Master's Degree in Informatics Engineering
Dissertation
Final Report

Prediction of Mitochondrial Toxicity Indexes for Pharmacological Compounds

João Rodrigues de Campos

jrcampos@student.dei.uc.pt

Supervisor: Francisco Pereira
Co-Supervisor: Ernesto Costa

Coimbra, July 3, 2017

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Dedicatory

This thesis is dedicated to my loving daughter who came to us during my research. To me she represents the future and complexity of human nature and the need to further protect it

Abstract

The pharmaceutical industry is facing new challenges. The development of a new drug takes around 12 years to reach the market, with a cost of around £1.5bn per drug and only around 1 in each 5.000 drugs manages to reach it. The toxicity of the drugs on the biological tissue is one of the critical points in the drug development process, as it may cause the termination or recall of a drug. To measure such toxicity, the assessment of the perturbation of the mitochondria is one of the techniques that can be used. This dissertation aims to create a Machine Learning (ML) model that early predicts the toxicity levels of pharmacological compounds. Such a model could then be used to identify and prevent the development of new drugs with a toxic composition. This work is done in cooperation with the MitoXT group, based at the UC-BIOTECH, Center for Neuroscience and Cell Biology, that will provide a dataset containing the information regarding the toxic effects of pharmacological compounds on the mitochondria.

Keywords: Machine Learning, Classification, Clustering, Mitochondria, Drug Toxicity

Resumo

A indústria farmacêutica enfrenta novos desafios. O desenvolvimento de um novo medicamento leva cerca de 12 anos para chegar ao mercado, com um custo aproximado de £ 1,5 biliões e apenas 1 em cada 5.000 o consegue alcançar. A toxicidade dos medicamentos no tecido biológico é um dos pontos críticos no processo de desenvolvimento de fármacos, uma vez que pode causar o seu término ou retração. Para medir essa toxicidade, a avaliação da perturbação das mitocôndrias é uma das técnicas que podem ser utilizadas. Esta dissertação visa criar um modelo gls ML que permita prever precocemente os níveis de toxicidade de compostos farmacológicos. Esse modelo poderá então vir a ser usado para identificar e prevenir o desenvolvimento de novos medicamentos com uma composição tóxica. Este trabalho é feito em cooperação com o grupo MitoXT, com base no UC-BIOTECH, Centro de Neurociências e Biologia Celular, que irá fornecer um conjunto de dados que contém informações sobre os efeitos tóxicos de compostos farmacológicos nas mitocôndrias.

Acknowledgement

I would first like to thank my advisors, Francisco Pereira and Ernesto Costa, for their excellent collaboration and dedication. Their valuable guidance and knowledge provided me with the tools to conduct my research and successfully complete my dissertation.

I am also grateful to the team at MitoXT without whom it would not have been possible to fathom the complexity of this problem. Their patience and comprehension with my lack of knowledge on the subject was uncanny.

I would also and above all like to thank my wife for her unconditional support and eternal patience while i was wandering in my research. Also a special thanks to my baby daughter whose midnight meals made me company and awake during the long nights of work.

Finally, to my friends who suffered from my famous endless doubts and comments over the biological nature of the problem and made an effort to pretend to be interested.

Contents

List of Figures

List of Tables

List of Acronyms

ADP	Adenosine Diphosphate.
AI	Artificial Intelligence.
ATP	Adenosine Triphosphate.
AUC	Area Under Curve.
CHAID	Chi-squared Automatic Interaction Detector.
CS	Computer Science.
CURE	Clustering Using REpresentatives.
DBSCAN	Density-based spatial clustering of applications with noise.
DIMD	Drug Induced Mitochondrial Dysfunction.
DT	Decision Tree.
EA	Evolutionary Algorithm.
ECAR	Extracellular Acidification Rate.
ECG	Electrocardiogram.
EDA	Exploratory Data Analysis.
FA	Factor Analysis.
FDA	Fisher Discriminant Analysis.
FN	False Negative.
FP	False Positive.
GA	Genetic Algorithm.
GP	Genetic Programming.
HC	Hierarchical Clustering.
ID3	Iterative Dichotomizer 3.
k-NN	k Nearest Neighbors.
LDA	Linear Discriminant Analysis.
MAE	Mean Absolute Error.
MDLP	Minimum Description Length Principle.
MDS	Multidimensional Scaling.
MI	Mutual Information.
ML	Machine Learning.
MPT	Mitochondrial Permeability Transition.

MSE	Mean Squared Error.
mtDNA	Mitochondrial DNA.
NN	Neural Network.
OCR	Oxygen Consumption Rate.
OXPHOS	Oxidative Phosphorylation.
PCA	Principal Component Analysis.
RF	Random Forest.
RFE	Recursive Feature Elimination.
ROC	Receiver Operating Characteristics.
ROS	Reactive Oxygen Species.
RSS	Residual Sum of Squares.
SMOTE	Synthetic Minority Over-sampling Technique.
SSE	Sum of Square Errors.
SVD	Single Value Decomposition.
SVM	Support Vector Machine.
TP	True Positive.

Chapter 1

Introduction

1.1 Motivation

With the passage of time the pharmaceutical industry is facing new challenges. Typically the development of a new drug takes around 12 years from the initial discovery stage to reach the market, with a cost of around £1.5bn per drug, with it growing almost exponentially every year. Despite all that, still only around 1 in each 5.000 drugs manages to reach the market. Developing a drug normally takes six stages: pre-discovery, drug-discovery, pre-clinical testings and phase one, two and three of clinical trials (*The Guardian - Healthcare Network* 2016). Even after the drug is approved and launched in the market there is the risk of unforeseen consequences that could force a recall of the product, incurring on extremely high costs for the company, not to mention the possible threat to human health.

The toxicity of the drugs on the biological tissue is one of the critical points in the drug development process. The assessment of the perturbation of the mitochondria is one of the techniques used to measure such toxicity. Mitochondria are organelles that exist in most eukaryotic cells, being responsible for cellular respiration and the production of Adenosine Triphosphate (ATP). Its impairment may result in cell death and organ failure, with liver failure and cardiovascular lesions as the most common consequences. In order to measure how the mitochondria are affected it is possible to use various indicators, such as the values of Oxygen Consumption Rate (OCR), Extracellular Acidification Rate (ECAR), ATP generation, mitochondrial membrane potential, amongst others (Cyprotex 2013).

There are multiple techniques to identify mitochondrial dysfunction however many are expensive, lengthy and require multiple experiments (Cyprotex 2013). Still, even after the experiments are made the decision of whether there is evidence of dysfunction is mainly done through a visual and statistical data analysis of key indicators of mitochondrial function.

1.2 Context

In today's society we generate and store an exponentially increasing amount of data. With it comes the need to transform these data into different types of knowledge. Large companies, such as *Google*, *Amazon* and *Netflix*, invest strongly in models that try to extract this knowledge for any number of purposes, be it to improve its search heuristics and results, or to make better recommendations. Other companies, such as *PayPal*, try to detect other serious subjects such as frauds and irregularities in real time. These interests combined with the recent evolution of technology led to the expansion of an Artificial Intelligence (AI) area known as Machine Learning (ML). Although it has been around for years, the powerful technologies that are available nowadays allowed it to apply its algorithms on the larger datasets, bringing it (back) to the spotlight.

Albeit the amount of data collected has risen, so has its inherent complexity, alongside the fact that everything is gathered regardless of its structure. Many of the created datasets present certain characteristics that make it difficult to process, such as inconsistency, unbalanced data, small datasets, missing features, high and low dimensionality, amongst others, which if not handled properly may lead to bad or misleading conclusions. As the interest in creating models that may adapt to a system is growing, the need to create techniques and solutions to these constraints has led to the development of certain practices that can be applied in all types of learning.

Given this context it seems only natural to apply ML techniques to data regarding mitochondrial (dys)function to create models that can make predictions on the toxicity of new compounds.

1.3 Objectives

This project aims to overcome the existing limitation on detecting and predicting mitochondrial dysfunction and to take advantage on the ability of ML algorithms to extract knowledge that would otherwise not be found. Its main goal is to develop a ML model that is able to predict the toxicity of pharmacological compounds present in the drugs, using mitochondrial indicators both to prevent the development of new drugs with potential consequences and to analyze drugs that are already in the market with toxic compounds. This will be done through the study of the dataset provided by the MitoXT group, based at the UC-BIOTECH, Center for Neuroscience and Cell Biology. This dataset contains the information regarding the toxic effects of pharmacological compounds on the mitochondria, mainly on their ability to generate energy (Adenosine Triphosphate (ATP)) through aerobic cellular respiration (Oxidative Phosphorylation (OXPHOS)), as well as the Extracellular Acidification Rate (ECAR). As with many biological datasets, it presents some challenges, such as limited samples, unbalanced data, high variance, and so forth. As such, one important step of the study consists on a careful preprocessing of the data. Afterwards some ML algorithms will be applied and the results and performances compared.

For the final model to be useful it must be able to identify the toxicity of the drugs with a satisfactory success rate. Hence, its threshold of success is to correctly identify at least 50% of the samples exposed to compounds with known mitochondrial toxicity, otherwise the model will be deemed unfit. If the development of the model is indeed successful it will afterwards be integrated in a software program and delivered to the MitoXT team to assist in the development of new drugs.

Complementary, we perform a thorough and comprehensive analysis of common ML algorithms and techniques, and their application in a real case scenario. In a more ambitious mood there is also the goal of trying to develop a program that will be able to predict a toxicity index using techniques from other AI areas. To do so a population-based metaheuristic optimization approach with biological inspired mechanisms, also known as Evolutionary Algorithm (EA), will be used through a Genetic Programming (GP) implementation.

1.4 Contributions

The contributions from this work can be divided between the biological and the ML fields. For the biological area the main contributions are:

- a thorough analysis of the experiments data using statistical techniques which provide insights into the mitochondrial effects of the drugs
- the development of tools that ease the interpretation, extraction and analysis of the experiments results
- a new method to detect Drug Induced Mitochondrial Dysfunction (DIMD) in the early stages of drug development and that is able to give insights into the similarity of the dysfunction induced by the drugs
- a preliminary GP model to predict a toxicity index

As for the ML field, they are:

- a comprehensive literature review on ML concepts, techniques and algorithms
- a thorough study of the data preparation methods and a comparison of the standard ML problems and algorithms performance
- the development, implementation, testing and analysis of a fusion expert based ensemble algorithm that considers the performance of each algorithm and their predictions' probabilities

1.5 Structure

This document is organized as follows:

- In *Chapter 2* there is a state-of-the-art revision of the current techniques and approaches used in the biological field to study the toxic effects of compounds, with emphasis on the detection of DIMD
- *Chapter 3* holds a state-of-the-art revision of the current ML techniques and algorithms
- *Chapter 4* describes and analyses the experiments process and their data
- *Chapter 5* contains the configurations and implementations of the ML techniques and algorithms used and in *Chapter 6* their results are thoroughly analyzed
- *Chapter 7* contains a discussion of the results obtained and in *Chapter 8* the final conclusions and future work are presented

Chapter 2

Drug Development

Due to the extensive development time and cost of a new drug it is becoming increasingly important to identify potential toxicities in an early stage of its development process. Still, the main purpose of drug toxicity testing is to create methods that can protect the public health from the hazardous effects of drugs.

The development of a new drug has multiple stages that spread for several years, as depicted in *Figure 2.1*. Typically, from discovery to manufacturing, takes around 12 years and can reach up to costs of approximately £1.5bn per drug (*The Guardian - Healthcare Network* 2016). Besides the risk and financial cost of a withdrawal due to unforeseen side effects, the risk of developing a drug that can compromise human health is a constant concern. It is however particularly difficult to completely guarantee that there will be no adverse side effects.

Toxicity is often missed precisely because it is a rare occurrence. If a specific drug causes a side effect in 0.1% of patients, more than 10 000 would have to be exposed for it to even become realistic (Dykens and Will 2007).

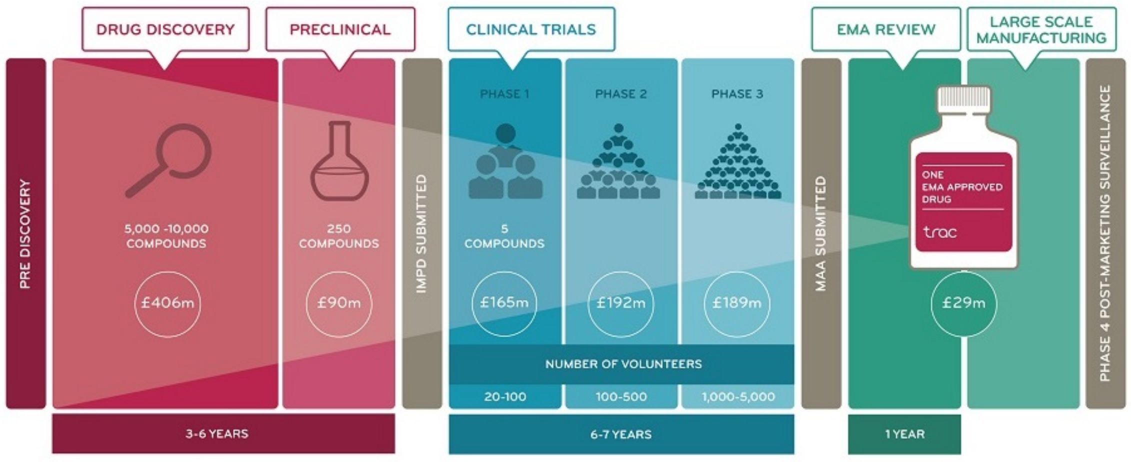


Figure 2.1: Drug Development Process, from (*Trac Drug Development Process* 2017)

Toxicity prediction is an essential step in the development process and is carried in its first stages, the preclinical phase. The studies used to predict such toxicity can be separated into three categories: **In silico**, which means performed entirely in a computer or via computer simulation, **In vitro**, those performed in a controlled laboratory environment outside of a living organism and **In vivo**, which are performed using a whole living organism. Each of these categories has its own advantages and disadvantages. An approach to improve the results it is to combine predictions from different high-throughput assays. Although there is not any preclinical experiment that is expected to completely predict the potential of human health hazards, the combination of *in silico*, *in vitro* and *in vivo* can strongly improve its assessment (*Differences between in vitro, in vivo and in silico studies* 2017).

2.1 Drug-Induced Toxicity

Drug induced toxicity remains one the main reasons for late stage attrition of drugs. The liver and the heart have been shown to be the main targets of drug-induced adverse effects. Due to the fact that many of them only manifest during clinical trials or post-launch, cardiotoxicity and hepatotoxicity are one of the main areas of investigation and focus (Cyprotex 2013). The toxic effects of a drug are a complex multitude of different and interlinking pathways, which makes it extremely difficult to fully understand them.

There are many ways a drug may induce toxicity. Reactive Oxygen Species (ROS) are derived from molecular oxygen and can be either radical or non-radical. The balance between ROS and antioxidants is important for maintaining the vital cellular and biochemical functions. If there is an excess of ROS it is known as oxidative stress and if there is a depletion is called reductive stress. Oxidative stress can lead to macromolecular damage and / or disruption of redox signaling and control. It has several targets, such as lipid membranes, DNA and proteins, which can lead to cardiotoxicity, nephrotoxicity, ototoxicity, skeletal myopathy and hepatotoxicity (Cyprotex 2013).

Reactive metabolite mediated toxicity can be dose dependent, where the toxicity is usually predictable, or idiosyncratic, where the toxicity is unpredictable, dose independent and probably does not manifest in preclinical tests. Due to the fact that the liver is the main organ involved in drug metabolism, it is often a target of such metabolites toxic effects, although they may also affect other organs such as skin and blood (Cyprotex 2013).

Another effect of drug induced toxicity is changes to cell cycle. The primary function of the cell cycle is to duplicate DNA in the chromosomes into two genetically identical daughter cells. This process is comprised of several steps which are controlled through checkpoints based on highly complex signaling pathways. Drug induced cell cycle toxicity affects this process by interfering in some of the steps (Cyprotex 2013).

Apoptosis is the process of programmed cell death. It is usually a beneficial process through which the system removes unwanted cells, however it can be also be a detrimental process. It is a complex system and is controlled through a range of extracellular or intracellular signals. Two major signaling pathways that lead to apoptosis have been identified and they are mitochondrial and receptor mediated. Drug induced toxicity to these pathways may active pro-apoptotic signals that lead to increased cell death (Cyprotex 2013).

Depending on their mode of action, carcinogens can be classified as genotoxic or non-genotoxic. Genotoxic carcinogens have a direct interaction of the chemical with DNA and / or the cellular apparatus that regulates the fidelity of the genome. Non-genotoxic on the other hand act as a non DNA damaging mechanisms in which a variety of cellular processes may be involved (Cyprotex 2013).

Cardiotoxicity is one of the main causes of toxicity related drug attrition. Initial physiological effects of drugs often manifest themselves as Electrocardiogram (ECG) changes and cardiac arrhythmias. Cardiomyopathy often occurs in the later stages of cardiac dysfunction with cardiac hypertrophy a common observation which can lead to heart failure.

2.2 Mitochondrial Dysfunction

Mitochondria are tiny but essential organelles to the survival of cells. They are responsible for the generation of almost all of our energy in the form of ATP (Lane 2006). Mitochondria can be thought of as cellular powerhouses converting energy release through substrate oxidation into a form usable by cellular processes (Dykens and Will 2008). With the exception of the cell nucleus, the mitochondria are the only organelles in animal cells to have their own genome, known as Mitochondrial DNA (mtDNA). They also have many other functionalities such as fatty acid oxidation, heme synthesis, calcium signaling and apotosis (Dykens and Will 2008). They are for that matter one of the most important elements in our body. They are composed of two membranes that separate the inter-membrane and the inner compartment where the mitochondrial matrix is (Rotella 2012), as illustrated in *Figure 2.2*.

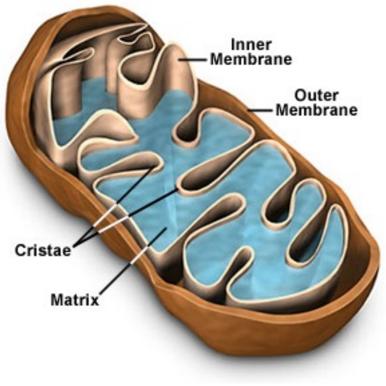


Figure 2.2: Mitochondria Structure, from (*Molecular Expressions - Mitochondria* 2017)

Mainly there are three distinct processes through which the human body generates almost all of its energy (ATP). **Glycolysis**, one of such processes, is an anaerobic pathway that happens in the cytosol and yields 2 ATP. Briefly, it converts glucose into pyruvate, which can either be converted to lactate through lactic acid fermentation or it can be used by another pathway to generate ATP (Alberts et al. 2002). The remaining two processes happen in the mitochondria and consist in two pathways: the **citric acid cycle** (also known as the Krebs cycle) which mainly uses the pyruvate generated by glycolysis and **Oxidative Phosphorylation (OXPHOS)**, an aerobic metabolism which generates most of the ATP. Through aerobic respiration the net yield is approximately 36 ATP for a single molecule of glucose (Ristow and Cuezva 2009).

OXPHOS consists of two independent processes: oxidation of reduced substrates and phosphorylation of Adenosine Diphosphate (ADP). The inner-membrane is where it occurs. It contains the complexes I, II, III, IV and V and its process can be seen in *Figure 2.3*. The OXPHOS machinery consists of two proton-pumping systems that are able to translocate protons across the membrane. The complex V, also known as ATP synthase as it catalyzes ATP production, is driven by an electrochemical proton gradient ($\Delta\psi$) between the inside and the outside of the inner membrane (Dykens and Will 2008).

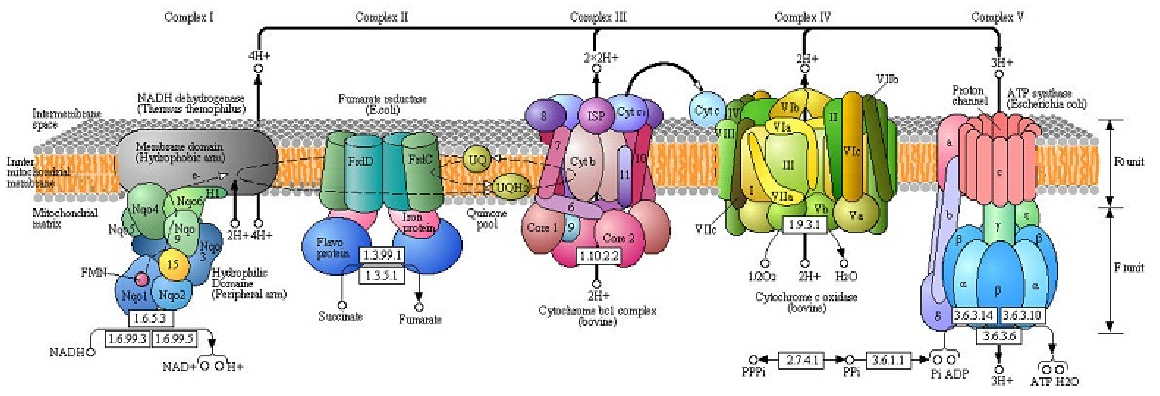


Figure 2.3: Oxidative Phosphorylation (OXPHOS), from (*Oxidative Phosphorylation* 2017)

Due to the complexity and structural details of the mitochondria there are many mechanisms through which DIMD can happen. Some drugs may inhibit the complexes involved in OXPHOS, conditioning the production of ATP. They can also uncouple electron transport from ATP synthesis by disrupting the proton gradient. It can also happen by inhibition of the enzymes involved in fatty acid oxidation or the citric acid cycle, causing the depletion of substrates. Some drugs may limit mtDNA replication while others may interfere with mtDNA encoded proteins (Rotella 2012). In summary, there are several mechanisms that can lead to DIMD.

The main toxic effects caused by DIMD (Cyprotex 2013) are:

- **Inhibition of Protein Complexes** - certain drugs are able to inhibit Complex I, Complex II / III, Complex IV or Complex V, impairing the mitochondrial ATP production
- **Uncoupling of Electron Transport from ATP Synthesis** - some drugs uncouple the electron transport from ATP synthesis by shuttling protons across the inner membrane
- **Inhibition of Mitochondrial Membrane Transporters** - inhibition of such transporters may cause substrate depletion and compromise ATP synthesis
- **Inhibition of Krebs Cycle Enzymes and Fatty Acid Metabolism** - the inhibition of Krebs Cycle Enzymes or fatty acid β-oxidation can also result in the depletion of substrates
- **Inhibition of mtDNA Replication and mtDNA-encoded Proteins Synthesis** - some drugs are able to interfere with mtDNA replication and mtDNA-encoded proteins synthesis
- **Oxidative Stress** - redox cycling, depletion of antioxidants or ROS can lead to the activation of cell death signaling and induction of mitochondrial membrane permeabilization
- **Mitochondrial Permeability Transition (MPT) Pore** - certain drugs may cause irreversible creation of a MPT pore leading to osmotic swelling and consequential cell death

As it can be seen, despite its critical role mitochondria are highly vulnerable to inhibition or uncoupling of the energy harnessing process, and have a high risk for permanent damage to the cell (Wallace and Starkov 2000). For that reason the perturbation of the mitochondria is one of the techniques used to measure the toxicity of a drug. DIMD is known to contribute to late stage compound attrition (Swiss et al. 2013). Although it is a fairly recent approach, the withdrawal of Cerivastatin and Troglitazone and a black box warning for Tolcapone showed the importance of testing drugs for mitochondrial impairment in early stages (Rotella 2012) . Since then multiple assays to detect such dysfunction have been developed, however many require expensive and specialized components and expertise.

2.2.1 In Vitro Methods to detect DIMD

The variety of mechanisms that may lead do DIMD means that there is no single assay that identifies them all. Thus, there are several techniques that can be used to analyze them, each one giving different insights into how the drug may induce toxicity. The main assays that are currently used to detect DIMD are (Rotella 2012) (Cyprotex 2013):

- **Mitochondrial Membrane Potential Measurements**
  A common method to detect DIMD is to monitor the mitochondrial membrane potential. For this, a variety of fluorescent dyes have been used in traditional fluorescence microscopy, automated high-content imaging, or high-throughput screening assays utilizing plate readers. A drawback of using this method to detect DIMD is that a decrease in the membrane potential is not necessarily a direct cause of mitochondria impairment.

- **Oxygen Consumption Measurements of Mitochondria**
  Due to the fact that OXPHOS consumes oxygen measuring the OCR is one of the techniques to study changes to its pathway. The first methods to analyze oxygen uptake were done with the Clark electrode, which had a limited throughput of approximately 10-20 compounds per day at a single dose. Nowadays however there are new technologies using oxygen sensors which can measure 96 or 384 wells in a single plate, giving a high-throughput screening of compounds.

- **Mitochondrial Mass**
  The number of mitochondria per cell can also be used to assess mitochondrial toxicity. An increase in mitochondrial mass may occur as a consequence of a response to increased energy production. Also a decrease in mitochondrial mass can also be a manifestation of early stage mitochondrial damage.

- **Measurement of mtDNA-Encoded Protein Levels in Cells**
  Drugs that interfere with mtDNA replication or mtDNA-encoded protein synthesis are rarely identified in 24-72h cell viability assays since both have lower turnover rates. Furthermore, the protein complexes involved in OXPHOS have to fall below a critical threshold in order to affect ATP synthesis, which varies between cell types. Since the mtDNA-encoded proteins are subunits of the complexes, they too have to fall below a critical threshold. Due to the time that these experiments take they are not ideal for high-throughput screening.

- **Metabolomics**
  Metabolomics is a scientific field that analyzes small molecules that constitute the metabolism (Roessner 2012). There is a growing interest in Metabolomics to investigate mitochondrial dysfunction. A non-radioactive tracer can be added to the culture

media of cells and its distribution into multiple metabolites produced by processes such as the Krebs cycles, lactate production and fatty acid oxidation can be monitored.

- **Assays for Measuring the Activity of Enzymes Involved in OXPHOS**
  In case there is a reduction of oxygen consumption with either isolated mitochondria or cells, the target of inhibition in OXPHOS can be identified either by classical biochemical models or by immunocapturing the individual protein complexes and determining each enzyme's activity in the presence of the test compounds.

- **Measurement of ATP Levels in Cells Grown in Glucose / Galactose Media**
  A way to measure the viability of the cells is through the measurement of ATP levels. This is usually done through immortalized cell lines. Although they possess the capability to be aerobic they have a preference to produce ATP *via* glycolysis when grown in glucose media. This is known as the Crabtree effect and to overcome this cells can be grown galactose containing media. Since the production of ATP through glycolysis is zero in galactose media the cells rely on OXPHOS which will make the cells more susceptible to DIMD.

- **Oxygen Consumption and Extracellular Acidification Measurements of Cells**
  Albeit oxygen sensors are able to identify changes to the OXPHOS in isolated mitochondria, a decrease in oxygen consumption alone can not be interpreted as mitochondrial toxicity. Dying cells, regardless of the mechanism of toxicity, will also display decrease in oxygen consumption. Therefore, pH sensors were developed to measure the ECAR and by doing so, the glycolytic rate. This provides the information required to identify which compounds directly impair the mitochondrial function and those that cause cytotoxicity.

There are a multitude of tests from different companies that use some of these techniques to attempt to detect DIMD. Two of them are particular recent and use the state of art on technology: the *Searhorse XF Cell Mito Stress Test* and the *Seahorse XF Glycolysis Stress Test*. Both measure the OCR and ECAR, one of the techniques just described, after adding a sequence of compounds that allow the analysis of some indicators for the OXPHOS and glycolytic pathway. As these tests allow plates with multiple wells it is also possible to use both culture media, glucose and galactose, to test the effects of drugs, which is another of the previously described methods. However, the processing of the results is mainly based on a visual analysis of the values distributions and deviations, usually in comparison to the control samples.

Chapter 3

Machine Learning

Although there is no consensus on what *intelligence* really is, it can be informally defined as: "Intelligence measures an agent's general ability to achieve goals in a wide range of environments" (Legg and Hutter 2006).

Artificial Intelligence (AI) has already been around for some time, and in conjunction with the previous definition it can be explained as the science of making intelligent machines, including intelligent computer programs (McCarthy 1987). From that evolved the idea of creating systems that can adapt and learn within an environment, that came to be known as Machine Learning (ML). Machine Learning (ML) is a subfield of Computer Science (CS) that gives computers the ability to learn without being explicitly programmed (Samuel 1959). Despite its recent intensive development in the last years, it has also been around for quite some time.

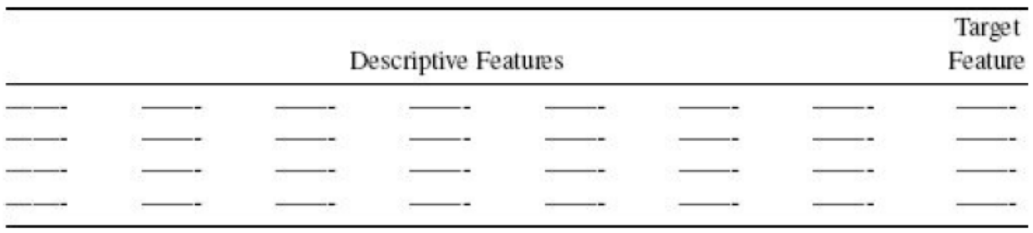
Depending on the type of feedback available to the learning system, ML tasks can be classified into **Supervised Learning**, **Unsupervised Learning**, **Semi-supervised Learning** and **Reinforcement Learning** (Ayodele 2010).

Different types of algorithms can be used to learn a computational model of a given problem. However, to create an accurate model it is crucial to thoroughly prepare the data, as it will directly influence its performance. Also essential to the analysis of ML algorithms is the ability to compare them and being able to state which are best at what.

3.1 Data Preparation

Due to the fact that nowadays most data gathering processes are loosely controlled, many of the datasets usually have irrelevant, noisy or unreliable, redundant or invalid data. Moreover, if these data are not carefully prepared the model may not be representative of the situation at study. For that matter, data pre-processing is one the most important steps in ML, usually consuming a considerable amount of the development time. Preparing the data includes several techniques, namely **cleaning**, **normalization**, **transformation**, **feature reduction** and **extraction** (Kotsiantis, Kanellopouloss, and Pintelas 2006).

In ML the basic requirements of the data for its models are fairly simple. It needs a dataset, large if possible, of historical **examples** of the scenario it represents, containing enough detail to describe it and its outcome. A standard structure is as the one in *Figure 3.1*. It is a simple table where the columns are divided into a set of **descriptive features** and a **target**, and each row represents an **instance**, that contains a value for each feature and target (Kelleher, Namee, and D'Arcy 2015).



Descriptive Features							Target Feature
——	——	——	——	——	——	——	——
——	——	——	——	——	——	——	——
——	——	——	——	——	——	——	——
——	——	——	——	——	——	——	——

Figure 3.1: Data Table, descriptive features and target, from (Kelleher, Namee, and D'Arcy 2015)

In order to prepare the data, some steps should be taken to allow the algorithm to perform to its best abilities in a generalized manner:

3.1.1 Data Cleaning

- **Noise Reduction and Outliers Detection**
  In ML noise is simply errors in the data or unpredictable random events. If the existing data has a lot of noise there are some methods that may be used to reduce it, such as *binning, clustering* and *regression*. To detect outliers there are several methods such as *Extreme Value Analysis*, *Probabilistic and Statistical Models*, *Linear Models*, *Proximity-based Models*, *Information Theoretic Models* and *High-dimensional Outlier Detection* (Aggarwal 2013).
- **Missing Values**
  Missing feature values is one of the most common situations one can find in real data. This is a subject that has already been thoroughly studied (Bruha and Franek 1996) (Grzymala-Busse and Hu 2001) that identifies the source of reason why the values are missing as one of the most important factors, as it will influence the choice amongst the different algorithms for each situation. There are a some options to deal with the missing values (Lakshminarayan, Harp, and Samad 1999):
  - **Ignoring Instances with Unknown Feature Values** - this method suggests to ignore the instances with at least one missing value
  - **Most Common Feature Value** - the value of the feature that occurs most often is selected and applied to all the unknown values of that feature
  - **Concept Most Common Feature Value** - the value of the feature that occurs the most within the entry class will be selected and applied to all the unknown values of that class for that feature
  - **Mean Substitution** - uses the feature's mean to fill the unknown values
  - **Regression or Classification Methods** - develop a regression or classification model based on the data available for that feature and use the other relevant features as predictors
  - **Hot Deck Imputation** - identify the most similar record with the one with missing values and use its values for the missing feature
  - **Method of Treating Missing Feature Values as Special Values** - consider the value of *unknown/missing* as a valid feature value

- **Data Inconsistency**
  In order to detect existing errors or inconsistencies in the data, a thorough analysis is required, as these may be silent errors. Using domain knowledge or expert help the dataset should be validated in order to detect inconsistencies or illegal values that may exist.

3.1.2 Data Transformation

- **Normalization**
  Normalization is the scaling of a feature values so that they fall under a certain range or relate to the feature values distribution. The most common methods to achieve this are:

  - **min-max scaling** - the data is scaled to a fixed range, usually 0 to 1:

    $$v' = \frac{v - min_x}{max_x - min_x}(new\_max_x - new\_min_x) + new\_min_x \tag{3.1}$$

    where v is the feature value to be scaled, min_x and max_x are the old minimum and maximum of the feature range, new_min_x and new_max_x are the intended minimum and maximum and v' is the scaled value of v

  - **z-score normalization** (or standardization) - all features will be rescaled so that they will have the properties of a standard normal distribution, with zero mean ($\mu = 0$) and a standard deviation of one ($\sigma = 1$):

    $$v' = \frac{v - \mu}{\sigma} \tag{3.2}$$

    where v is the value to be normalized, μ and σ are the current mean and standard deviation of the feature and v' is the normalized value of v

- **Aggregation and Decomposition**
  Some datasets possess features that individually do not provide much information, however if aggregated with others may give a good representation. On the other hand, the opposite may happen as well, where a feature represents a complex concept that may be more useful when split into its building parts.

- **Discretization**
  This is the process of dividing a continuous attribute into intervals, drastically reducing the number of possible values, resulting in a faster and effective ML process. It can be separated into two types (Agre and Peev 2002):

  - **Unsupervised Discretization**
    Discretize attributes discarding the class labels. Some of the unsupervised methods are *Equal Size*, *Equal Width*, *Equal Frequency* and *Clustering Discretization* (Rajashree Dash, Paramguru, and Rasmita Dash 2011).

- **Supervised Discretization**
  Takes into account the item class, finding intervals where most of the data instances have the same class and they differ amongst intervals. The most common are *entropy-based* and *Chi-squared* discretization (Rajashree Dash, Paramguru, and Rasmita Dash 2011).

3.1.3 Dimensionality Reduction

The complexity of any model usually depends on the number of inputs it has, as it determines both the time and space complexity to train it. In order to facilitate the development of a model it may be important to reduce its dimensionality (Alpaydin 2014):

- both the complexity and the size of many algorithms strongly depend on the number of input dimensions, *features*, of the problem, hence, reducing it decreases its complexity, time and space
- avoid processing unnecessary inputs
- simpler models have less variance, being more robust against noise and outliers
- reducing the number of features may allow for an easier understanding of the underlying process in the data, eventually allowing a visualization of it, and knowledge extraction

That being said, dimensionality reduction can be achieved through *feature extraction* and / or *feature selection*.

- **Feature Selection**
  Due to the way most of the information is gathered, without a purpose and a controlled supervision, most of the datasets have irrelevant or redundant features in them. The removal of these features is called *feature selection*. There are a few methods to perform feature selection, although they are usually grouped in three:
  - **Filter Methods**
    Filter methods use some form of statistical measures that assign a score to each feature, and the features are then ranked by their scores and either kept or discarded according to some threshold. Some of the most common methods include the *information gain*, *mutual information*, the *correlation and coefficient factors* and the *Chi squared test*.

    High-correlated features are those that depend on one-another and thus have similar information. Therefore keeping only one of the two columns will not drastically decrease the amount of information available (Silipo et al. 2014). To remove the highly correlated features, the correlation between all features is first measured and afterwards those that had a value above a given threshold are removed. One of the most common measures of correlation is Pearson's correlation, where m_i is the sample mean of feature x_i, s_i is its sample standard deviation, and n is the number of samples (Sa 2001):

$$r_{ij} = \frac{\sum_{k=1}^{n}(x_{k,i} - m_i)(x_{k,j} - m_j)}{(n-1)s_i s_j}, r_{i,j} \in [-1, 1] \tag{3.3}$$

Another method, the *Mutual Information (MI)* measures the dependence between two features. Similar to the correlation coefficient, it is 0 if the variables are independent and higher values indicate higher dependency. Unlike the correlation coefficient that can only handle linear dependence this is able to detect both linear and non-linear relationships (Sulaiman and Labadin 2015). Formally MI is defined by the probability density function of X, Y and joint variables (X, Y). However it can also be defined in terms of entropy (Sulaiman and Labadin 2015):

$$MI(X, Y) = h(Y) - h(Y|X) \tag{3.4}$$

And the entropy of X is expressed as:

$$h(X) = -\int f_x(x) \log^{f_x(x)} dx \tag{3.5}$$

- **Wrapper Methods**
  Wrapper methods on the other hand consider the feature selection as an optimization problem, where different combinations are prepared, evaluated and compared against other combinations. Wrapper methods are essentially solving the "real problem", resulting in a higher computational cost and with a tendency to create overfitting. Nonetheless it can find a better feature subset and it allows the process to detect possible interactions between variables. Some of the most common methods are *recursive feature elimination*, *sequential feature selection algorithms* and *genetic algorithms*

  Recursive Feature Elimination (RFE) methods perform dimensionality reduction using a given ML algorithm. Initially the algorithm is trained on n features and both weights and rankings are assigned to each feature. In the end, the feature whose weight is the smallest is removed. This process is repeated until only one feature remains (Guyon et al. 2002). Then subset with the best performance is chosen.

- **Embedded Methods**
  Embedded are quite similar to wrapper methods, as they manage to learn which features contribute more positively to the accuracy of the model while it is being created. The difference is that embedded methods have an intrinsic building metric that is used during learning, and also manage to avoid as much overfitting. Some of the most common methods are *L1 regularization* and *decision trees*

- **Feature Extraction**
  In feature extraction the goal is to find a new set of dimensions that are a combination of the original ones. These methods can be supervised or unsupervised, and some of the most widely used are *Principal Component Analysis (PCA)*, and *Linear Discriminant Analysis (LDA)* (Alpaydin 2014), which are both linear projection methods:

  - **PCA**
    An unsupervised method, that is a statistical procedure that takes a dataset consisting of a representation of points in a high-dimensional space and performs an orthonormal transformation to find the directions along which the variance is higher. These directions, called principal components, are ordered by the amount of variance they contain. The first component is the one that contains the

maximum variance and each succeeding component contains the highest possible variance under the constraint that it is orthogonal to the preceding components (Silipo et al. 2014).

- **LDA**
  Also known as *Fisher Discriminant Analysis (FDA)*, a supervised method, that tries to separate the samples of distinct groups by transforming them to a space that maximizes their between-class separability while minimizing their within-class variability (James et al. 2014)

Other methods, albeit less common, such as *Factor Analysis (FA)* and *Multidimensional Scaling (MDS)*, which are quite similar to PCA, and *Single Value Decomposition (SVD)* and *Sammon Mapping* are also available.

3.1.4 Instance Selection

Instance selection methods are used in order to help the algorithms to cope with the infeasibility of very large data sets. It becomes an optimization problem to minimize the data size while keeping the datamining quality. Instance selection can be achieved through *sampling*, *boosting*, *prototype selection* and *active learning* (Ghosh 2005). One of the major means of instance selection is sampling, whereby a sample is selected for testing and analysis, and randomness is a key element in the process (H. Liu and Motoda 2001). Some of the most well known are (Cano, Herrera, and Lozano 2005):

- **Random sampling** - Randomly selects a subset from the original set with equally distributed probabilities
- **Stratified sampling** - Applicable when the classes representations are not uniformly distributed, it increases the frequency of the minority classes in order to balance the discrepancy

It is well accepted that a "powerful computationally intensive procedure operating on a sub-sample of the data may in fact provide superior accuracy than a less sophisticated one using the entire data base" (J. H. Friedman 1997). Also, imbalanced datasets may lead the models to overfitting, removing its ability to generalize, hence there are specific solutions for instance selection on these sets.

Undersampling techniques seek to reduce the number of samples of the majority class in the dataset (Kubat and Matwin 1997). As a result the overall number of records in the dataset is decreased, also shortening training time. Whilst it reduces the model complexity, it may also lose some valuable information with the discarded items. One of the most effective techniques is **Random Undersampling** where samples of the majority class are randomly eliminated until the ratio between the majority and minority class is at the intended level. One of the problems with this approach is that it is not possible to control what information about the majority class is lost. Nonetheless, despite its simplicity it has been shown to be one of the most effective undersampling methods (A. Y.-c. Liu 2004).

Oversampling on the other hand seeks to increase the number of samples from the minority class. The obvious advantage is that it does not lose information since all the samples are kept. It does however increase the size of the dataset and consequently the time to train the model. Some of the methods to achieve this are:

- **Random Oversampling**
  Similar to *Random Undersampling*, *Random Oversampling* is a simple but effective technique. It works by choosing members from the minority class at random which are then duplicated on the new dataset. This happens with replacement, that is, the same sample may be chosen multiple times. The dataset from which the samples are chosen is always the original to avoid changing the initial probabilities (A. Y.-c. Liu 2004).

- **Synthetic Minority Over-sampling Technique (SMOTE)**
  SMOTE is an oversampling technique in which the minority class is oversampled through the creation of synthetic examples (Chawla et al. 2002). It takes each minority class sample and introduces synthetic examples along the line segments joining the *k* minority class nearest neighbors. This approach forces the decision region of the minority class to become more general (Chawla et al. 2002).

3.2 Data Analysis

- **Descriptive Statistics**
  Because data are organized in distributions, it is possible to analyze them through some statistics that describe or summarize them, taking into account its characteristics, such as centrality and spread, as in *Figure 3.2*.

$$\begin{tikzcd}
 & \boxed{\text{mean}} & & & \boxed{\text{Standard Deviation}} \\
\boxed{\text{Centrality}} \arrow[ur] \arrow[r] \arrow[dr] & \boxed{\text{median}} & & \boxed{\text{Spread}} \arrow[ur] \arrow[dr] & \\
 & \boxed{\text{mode}} & & & \boxed{\text{Variance}}
\end{tikzcd}$$

Figure 3.2: Descriptive Analysis

  Based on the data centrality:

  - **mean** - $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - **median** - midpoint of the distribution
  - **mode** - most frequent element(s)

  Based on the spread of the data:

  - **standard deviation**

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2} \tag{3.6}$$

  - **variance**

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2 \tag{3.7}$$

- **Exploratory Data Analysis (EDA)**
  Exploratory Data Analysis (EDA) has its focus on the data, mainly using graphical tools that provide a different perspective on the data. Through EDA it is possible to visualize the centrality and spread of the data, using *boxplots*, *scatterplots*, *histograms* and *runcharts*, amongst others. These graphical tools allow the user to perceive patterns, distributions, outliers and relationships between features that would not been easily detected or understood with other analysis.

3.3 Learning Types

The main concept of ML is that it is built from a set of data and tries to learn from it. However, the data that is available for a problem limits, or at least constraints, the type of the learning task.

3.3.1 Supervised Learning

By far the most common type of ML, this type of learning contains the data and the expected output for every combination of inputs. The goal is to map the relation between the input variables and its outputs in such a manner that for a new set of inputs it will be able to predict the output. The term "supervised" comes from the learning process, where the algorithm iteratively tries to learn to predict from the dataset and is corrected until it achieves an acceptable error rate. Based on the type of problem it can be one of the following:

- **Classification** - when the expected output for the input variables is a discrete value, such as a category or a class
- **Regression** - when the output is a continuous value

3.3.2 Unsupervised Learning

On the other hand, unsupervised learning has no such supervision, that is, for a combination of inputs there is no class or label for it. In this type of learning the aim is to find a structure or distribution in the input space so that recurring patterns can be found. It can be one the following, depending on the problem:

- **Clustering** - this method tries to discover the inherent groupings in the data
- **Association** - tries to find rules and relations between variables in large datasets to discover interesting patterns

3.3.3 Semi-supervised Learning

Semi-supervised learning falls between supervised and unsupervised learning, as it has both labeled and unlabeled data. Usually only a small set of data is labeled as normally it is too expensive to label all the entries. Semi-supervised learning may refer to either *transductive* or *inductive learning*, where transduction refers to the labeling of the unlabeled data provided through an approximate model, and the second to the prediction function for the entire input space (Chapelle, Schlkopf, and Zien 2010). Many real world problems fall into this category due to the cost of labeling all the entries in a large dataset.

3.3.4 Reinforcement Learning

Reinforcement learning algorithms interact with the environment, while learning a model that maps situations to actions, as to maximize the reward signal. It learns through those interactions and observing the results they produce. The model is not told explicitly which actions to take, in turn it must discover which actions yield the most reward by trying them, affecting not only the immediate reward but all the subsequent rewards (Sutton and Barto 1998). Trial-and-error are the two characteristics that distinguish this type of learning from the remaining.

3.4 Algorithms

There are many inductive learning problems, and one of the main differences between them is what type of prediction they try to make. Depending on the problem there are a few options on which algorithms to use. In this section some of the most known, and the ones that will most likely yield better results for the goal of this project, will be detailed. Although some of these algorithms are represented in one type of problem they may be used for others as well, with some adjustments.

To develop a ML predictor it is necessary to first create the model, and then train it to fit the problem data. Afterwards it must be evaluated to measure how well it can predict new unseen data. To do so, usually the dataset of the problem is divided in two, the **training** and the **test** set, which will be used for the fitting of the model and its evaluation, respectively. This subject will be further explained in *Section 3.5.1*.

3.4.1 Supervised

Classification problems are concerned with separating data into distinct classes. In these problems the target value is discrete, categorical. For these problems the algorithms try to build models that are able to separate the data into distinct classes through training with labeled data. The models are then used with new data and try to predict its classes from what it learned from the training set. As this type of algorithm requires labeled data it is considered a form of supervised learning. Some of the most common classification algorithms are:

- **Naive Bayes**
  The *Naive Bayes* algorithm is a classical demonstration of how generative assumptions and parameter estimations simplify the learning process. It makes the assumption that given the label (Y) of each sample, the features (X) are independent, that is (Shalev-Shwartz and Ben-David 2014):

$$P[X = x|Y = y] = \prod_{i=1}^{d} P[X_i = x_i|Y = y] \tag{3.8}$$

With this assumption the Bayes optimal classifier is (Shalev-Shwartz and Ben-David 2014):

$$h_{Bayes}(x) = argmax_{y \in \{0,1\}} P[Y = y] \prod_{i=1}^{d} P[X_i = x_i | Y = y] \tag{3.9}$$

- **Gaussian Process**
  The *Gaussian Process* is a generalization of the Gaussian probability distribution. While a probability distribution describes random variables as scalars or vectors a stochastic process governs the properties of functions. Loosely thinking, a function can be thought of as a very long vector, where each entry in the vector specifies the function value $f(x)$ at a particular point x (Rasmussen and Williams 2005). *Gaussian Process* can be used both for regression and classification. For the classification it uses a Laplace's method which in turn utilizes a Gaussian approximation, as properly detailed in (Rasmussen and Williams 2005).

- **Logistic Regression**
  Logistic Regression is a linear classifier that tries to predict the probability of a sample belonging to each class, that is, a value between 0 and 1. To do so, in *Logistic Regression* the logistic function, that has a form like *Figure 3.3*, is used (James et al. 2014):

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \tag{3.10}$$

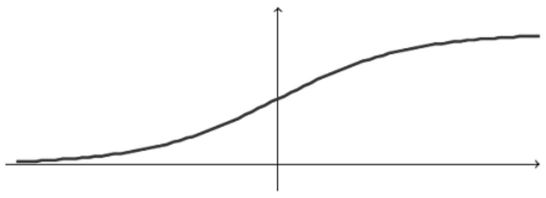


Figure 3.3: Logistic Function, from (Shalev-Shwartz and Ben-David 2014)

Which leads to:

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X} \tag{3.11}$$

The left-hand side of this equation is known as the *odds*, and can take any value from 0 to ∞, meaning very low and very high probability respectively. By taking the logarithm of both sides one arrives at (James et al. 2014):

$$log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X \tag{3.12}$$

where the left-hand side is called the *log-odds* or *logit*. The regression coefficients, β_0 and β_1 have to be estimated through the training data, and the most common way is through *maximum likelihood*, which tries to find the coefficients that maximize the likelihood (James et al. 2014):

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{3.13}$$

To transform it to a multiple logistic regression it can be generalized as (James et al. 2014):

$$log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{3.14}$$

Where $X = (X_1, ..., X_p)$ are p predictors. Then it can be rewritten to(James et al. 2014):

$$p(X) = \frac{e^{B_0 + B_1 X_1 + ... + \beta_p X_p}}{1 + e^{B_0 + B_1 X_1 + ... + \beta_p X_p}} \tag{3.15}$$

The regression coefficients can also be estimated through the maximum likelihood function.

- **Linear Regression**
  Regression problems are fairly similar to classification problems, and consequently also a form of supervised learning. While classification problems try to predict discrete values, the classes, regression tries to predict continuous values. These methods attempt to explicitly model the relationship between the inputs and the outputs. Some of the most common algorithms are the same as the ones for classification, such as Support Vector Machine (SVM), Decision Tree (DT) and Random Forest (RF), however there is also *Linear Regression*.

  Linear Regression tries to learn a linear function that best approximates the relationship between the inputs and the target. *Simple Linear Regression* is a straightforward approach of predicting Y based on a single input X, it assumes a approximately linear relation between X and Y (James et al. 2014):

$$Y \approx \beta_0 + \beta_1 X \tag{3.16}$$

  As with the logistic regression, β_0 and β_1 are coefficients, the *intercept* and *slope* respectively. In order to estimate them the most common method is through the *least squares* criterion, that tries to minimize the Residual Sum of Squares (RSS) for the coefficients (James et al. 2014):

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}))}{\sum_{i=1}^{n}(x_i - \bar{x}^2)} \tag{3.17}$$

$$\hat{\beta_0} = \bar{y} - \hat{\beta_1}\bar{x} \tag{3.18}$$

- **Decision Tree (DT)**
  DTs are top-down, recursive classifiers, and one of the most intuitive prediction models, mimicking how a human programmer would model it. They construct models based on values present in the training data using a decision tree as a predictive model,as the one in *Figure 3.4*. It makes predictions of the value associated with an instance by traveling from a root node to a leaf (Shalev-Shwartz and Ben-David 2014). If the target variable is continuous the models are called **regression trees** and **classification trees** otherwise.

  Due to the computational cost of trying to cover the whole solution space to find the tree that optimizes the problem, practical decision tree learning algorithms are based on heuristics. A general tree algorithm, similar to Iterative Dichotomizer 3 (ID3),

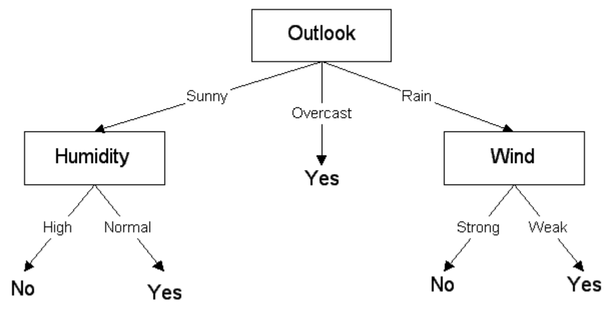


Figure 3.4: Decision Tree, from (*University of Florida The ID3 Algorithm* 2012)

starts with a tree containing a single leaf (*root*) and assigns it a label according to some rule. Then a recursive process begins, where on each iteration it measures some form of "gain" that quantifies the the improvement of splitting a leaf. Afterwards, amongst all possible splits it chooses the one that generates a higher gain and splits it, or chooses not to split it at all (Shalev-Shwartz and Ben-David 2014).

Different algorithms use different gain functions, being some of the most common **Train Error**, **Information Gain** and **Gini Index or Impurity** (Shalev-Shwartz and Ben-David 2014). Decision tree algorithms usually suffer from generating very large trees, creating low empirical risk but having a high true risk. A solution to avoid that is to limit the number of iterations, creating a bounded tree, or prune it, reducing the size but keeping the empirical error (Shalev-Shwartz and Ben-David 2014). Some of the most well known tree predictors are ID3, C4.5 and Chi-squared Automatic Interaction Detector (CHAID).

- **Support Vector Machine (SVM)**
  SVMs are binary classification algorithms that are able to predict both linear and non-linear data in high dimensional feature spaces. This high dimensionality raises both computation and sample complexity challenges. To face the sample complexity the SVM searches for "large margin" separators, by maximizing the distance between the samples of each class and the hyperplane that separates them (Shalev-Shwartz and Ben-David 2014).

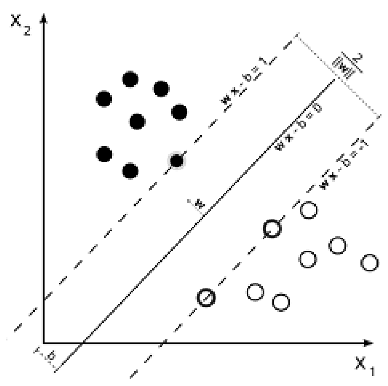


Figure 3.5: Hard Margin SVM, from (*University of Maryland, Hard Margin SVM* 2012)

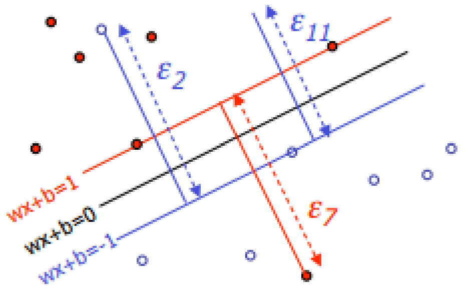


Figure 3.6: Soft Margin SVM, from (*Wikipedia Soft Margin SVM* 2008)

SVMs halfspaces can be calculated through *hard* or *soft margins*. Hard margin SVMs (*Equation 3.19*) require that all data points must be correctly classified, which can be difficult in noisy sets and lead to overfitting.

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 \\ s.t. \quad & y_i(w * x_i + b) \geq 1, \forall x_i \end{aligned} \tag{3.19}$$

Soft margin SVMs on the other hand introduce slack variables ε_i to allow the misclassification of some points, as in *Figure 3.6*, and can be found through *Equation 3.20* (III 2012):

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 + C\sum_{k=1}^{R} \varepsilon_k \\ s.t. \quad & y_i(w * x_i + b) \geq 1 - \varepsilon_i, \forall x_i \\ & \varepsilon_i \geq 0 \end{aligned} \tag{3.20}$$

If the data are not linearly separable it is also possible to map it to a high dimensional space, where it is easier to classify with linear decision surfaces. Mapping all the data to the high dimension space may be too expensive, so it is possible to use **kernels**, in order to avoid explicitly making the mappings. Some of the most used kernels are the *Polynomial* and the *Gaussian*, and their use can be observed in *Figure 3.7*.

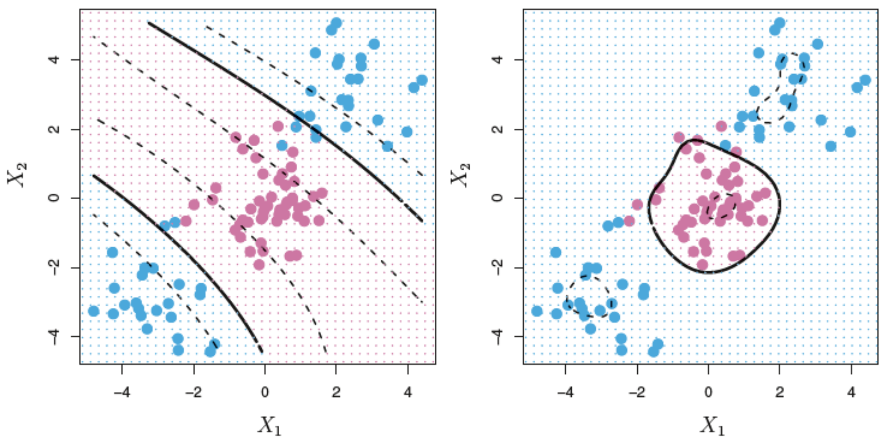


Figure 3.7: An SVM with a Polynomial (left) and Gaussian (right) kernels applied on a non-linear data, from (James et al. 2014)

As it can be seen the Gaussian kernel creates decision boundaries that resemble a circular shape, allowing it to separate the classes. The polynomial kernel creates very different hyperplanes from the polynomial, which in this case are almost linear decision borders. Different kernels are able to project the data to different spaces, allowing the model to create the necessary decision boundaries to model the data.

- **Neural Networks (NNs)**
  An artificial NN is a model of computation inspired in the structure of neural networks in the brain. It can be described as a directed graph, where the nodes are the neurons and the edges are the links between them. Each neuron has as input a weighted

sum of the outputs of the neurons connected to its incoming edges (Shalev-Shwartz and Ben-David 2014). The most common structure is a *feedforward* NN where the underlying graph does not contain cycles, as the example illustrated in *Figure 3.8*.

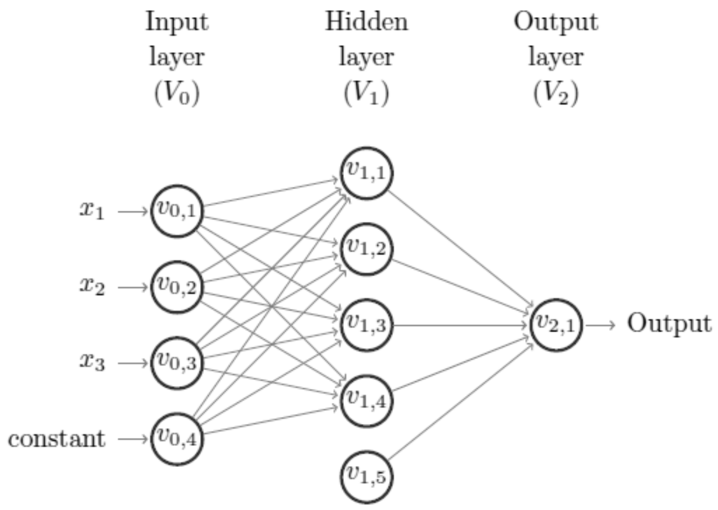


Figure 3.8: Feedforward Neural Network, from (Shalev-Shwartz and Ben-David 2014)

For a NN with N number of layers, the layers $V_0, ..., V_{N-1}$ are usually called *hidden layers*, V_0 the input and V_N is the output layer. In order to improve the weights of the edges some algorithms can be used, such as *backpropagation*. Briefly, it calculates the gradients of the loss function on an example (x, y) with respect to a weights vector w (Shalev-Shwartz and Ben-David 2014).

- **Genetic Programming (GP)**
  GP is a technique of machine learning inspired by the biological evolution theory, it is a type of Evolutionary Algorithm (EA). GP encodes programs as a set of genes, and based on an evolutionary cycle tries to generate solutions that are able to solve a problem. The resulting program can be thought of as a model, such as the ones generated from the other algorithms. An EA works on a large set of candidate solutions, also known as a population, and to be developed requires the following items to be defined:

  - **Representation** - In GP solutions may be represented as *regulatory networks*, *artificial regulatory networks*, *grammars* or *trees*, the latter as illustrated in *Figure 3.9*. The variables and constants (x, y and 3) are the leaves of the tree and are called *terminals*. The arithmetic operations ($+, *$ and *max*) are internal nodes, also called *functions*. The sets of allowed functions and terminals combined constitute the *primitive set* of a GP system (Poli, Langdon, and McPhee 2008).
  - **Initialization** - The are some options regarding how the initial population is generated, such as *Full*, *Grow* or *Ramped Half-and-Half*. *Ramped Half-and-Half* initializes half of the population using the *Full* method, which generates trees with the same depth for all leafs, and the other half using the *Grow* method, which allows the creation of trees with varied sizes and shape and with different depths (Poli, Langdon, and McPhee 2008).

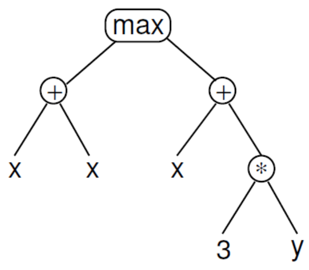


Figure 3.9: GP Syntax Tree Representation of $max(x + x, x + 3 * y)$, from (Poli, Langdon, and McPhee 2008)

- **Evaluation** - It is usually under the form of a fitness function, and it is problem dependent. It should try to measure how good the individual is, that is, how well does the model fit the data. The fitness function may be, or take into account, factors such as errors, time, accuracy, payoff or compliance
- **Variation** - Usually is done through **crossover** and/or **mutation** operators. *Point Mutation* is one of the most common mutations, it is the approximate equivalent of the bit-flip mutation commonly used in Genetic Algorithm (GA). It works by selecting a random node and replacing it with a different primitive of the same arity taken from the *primitive set* (Poli, Langdon, and McPhee 2008). *Subtree Crossover* is a common crossover operator, which is depicted in *Figure 3.10*.

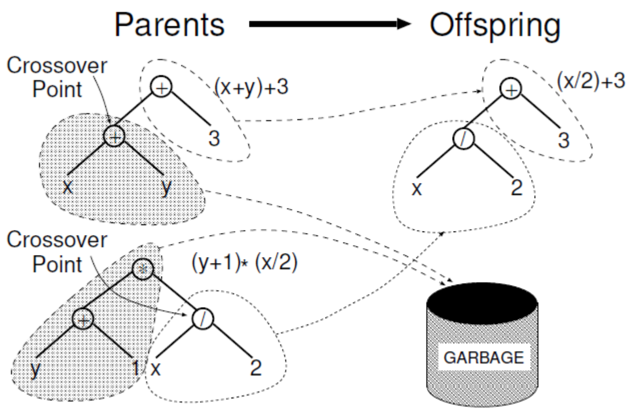


Figure 3.10: GP Subtree Crossover, from (Poli, Langdon, and McPhee 2008)

Given two parents it randomly selects a *crossover point* in each parent tree. It then creates the offspring by replacing the subtree rooted at the crossover point in a copy of the first parent with the subtree rooted at the crossover point from the second parent (Poli, Langdon, and McPhee 2008). Mutation operators randomly modify certain parts of the individuals.

Usually a GP algorithms works like as it is described in *Figure 3.11*:

1. **Population** - a set of individuals, each one representing a program is initially generate through one of the initialization methods.
2. **Parent's Selection** - a stochastic selection of a subset of individuals is made from the original population, accordingly to some heuristic, such as a deterministic tournament. These individuals constitute the **Parent's Population**, which will be used to create the next generation

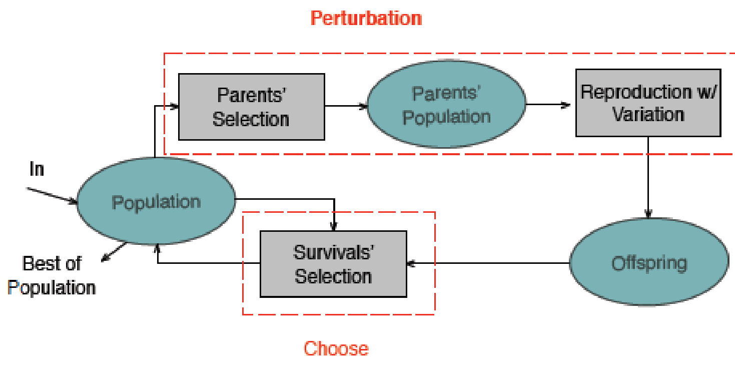


Figure 3.11: GP Cycle

3. **Reproduction With Variation** - the main evolutionary part of this approach, here the parents are combined, usually in pairs, using *crossover*. The new individuals may still suffer from further modifications through *mutation*. These new individuals constitute the **Offspring**.

4. **Survival's Selection** - the new individuals that will be placed in the population are chosen accordingly to some heuristic, such as *elitism* and/or *generational selection*. *Elitism* guarantees that the best defined percentage of the current population will be kept. *Generational selection* selects the remaining number of required individuals from the best of the offspring (Poli, Langdon, and McPhee 2008).

This cycle will run until it meets a stop criterion, such as maximum number of generations, stagnation of evolution or problem specific.

- **Ensemble**
Ensemble methods are a compilation of several weak independent algorithms that are trained and whose predictions are gathered to work as whole, as depicted in *Figure 3.12*. They contain a number of learners called based learners (weak learners, that is, not much better than a random classifier), which are usually generated from the training data by a base learning algorithm such as *decision tree*, *neural network* or other kinds of learning algorithms (Zhou 2012).

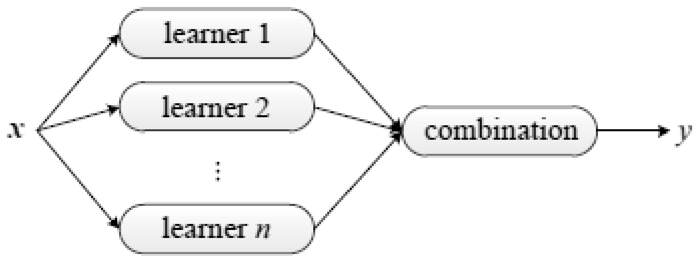


Figure 3.12: Ensemble Methods, from (Zhou 2012)

Usually an ensemble is constructed in two steps: generating the weak learners and then combining them. It is believed that a good ensemble is made of as accurate and diverse as possible base learners. The computational cost of the creation of the ensembles is not much higher than creating a single learner. Some of the most well known ensemble methods are based on DTs classifiers, such as *Boosting*, *Bagging* and *RFs*.

Bagging is an ensemble algorithm, which is an abbreviation for *Bootstrap AGGregatING*, and it is based on *bootstrap* and *aggregation*. It works on the premise that the combination of independent based learners will lead to a decrease of errors. It applies *bootstrap* sampling to generate the data subsets for training the base learners. To aggregate the learners for classification it combines the outputs with a voting strategy (Zhou 2012).

RF is an ensemble method that has proven to be particularly effective with large datasets. It is an extension of *Bagging*, being the difference in the incorporation of randomized feature selection. In the construction of a decision tree, at each step of split selection the RF selects a subset of features at random and then performing the split selection process (Zhou 2012). After a large number of trees is generated a vote for the most popular class is cast.

The construction of an ensemble can be separated in two approaches: **Classifier Selection** and **Classifier Fusion** (Nagi and Bhattacharyya 2013). In *classifier selection* the classifiers are trained to become experts in some area of the feature space. Then the output of the classifiers identified as the best for a specific classification problem is selected. Usually the input sample space is partitioned into smaller areas and each classifier learns the example in each area. *Classifier fusion* combines the outputs of the different classifiers. Each one of them has some knowledge of the entire feature space and tries to solve the same classification problem using different methods based on different training datasets, classifiers or parameters (Nagi and Bhattacharyya 2013).

To combine the outputs of the algorithms there are some options, such as **Majority Voting** and **Weighted Majority Voting**. *Majority voting* ensembles will make a decision based on a majority vote, that is, each classifier has equal vote and the most predicted target is the one chosen by the ensemble as a whole (Polikar 2012). Voting based methods operate on labels only, where $d_{t,j}$ takes a value of 0 or 1 depending on whether the class j was chosen by the classifier t or not, and the most predicted is the one returned by the ensemble:

$$y = argmax_{j \in \{1,2,\ldots,C\}} \sum_{t=1}^{T} d_{t,j} \tag{3.21}$$

However, if there is some evidence that certain classifiers are more qualified than others, giving more weight to their decision may improve overall performance. Hence, a weighted majority voting can be used, where there is a product between the classifier weight w_t and its decision $d_{t,j}$, and similar to the majority voting, the class that receives the highest weighted vote is selected as the final decision (Nagi and Bhattacharyya 2013).

$$y = argmax_{j \in \{1,2,\ldots,C\}} \sum_{t=1}^{T} w_t d_{t,j} \tag{3.22}$$

3.4.2 Unsupervised

Clustering problems don't have a class or label in the data, making it a form of unsupervised learning. Instead, these problems try to find structures that exist in the data, grouping the samples into groups of maximum commonality, by maximizing the intraclass similarity while minimizing interclass similarity (Han 2005), using some model approaches such as hierarchical and centroid-based. Some of the most common algorithms are:

- **Density-based spatial clustering of applications with noise (DBSCAN)**
  As its name says, DBSCAN is a density-based clustering algorithm that tries to identify and distinguish points in clusters from noise, based on its density. The main idea is that for each point in a cluster its neighborhood, based on a given radius, has to contain at least a minimum number of other points, that is, the density in the neighborhood has to exceed a defined threshold (Ester et al. 1996). In this algorithm, points are divided into *core points*, which have many neighbor points, *border points* which lie in the neighborhood of at least one core point, and noise points (Celebi 2015).

  First there is the need to formally define when a point should become a *core point*, by defining the ε-neighborhood (the set of points that are at most at a distance of ε from a point):

  $$N_\varepsilon(x_i) = \{x \in \mathbb{D} | d(x_i, x) \leqslant \varepsilon\} \tag{3.23}$$

  Then the core points within the point's ε-neighborhood can be defined as follows, with respect to a local density measure *minPts*:

  $$P = \{x \in D| \quad ||N_\varepsilon(x) \geqslant minPts\} \tag{3.24}$$

  A point p is called *directly density-reachable* from a core point x if $p \in N_\varepsilon(x)$. Also, it may be called *density-reachable* from a point x if there is a series of points $p_1, p_2, ..., p_n$ with $p = p_n$ and $x = p_1$ and for every (p_i, p_{i+1}) holds that p_{i+1} is directly density-reachable from p_i. It is also possible to say that if two points p and q are *density-connected* if there is a point o such that p and q are both density-reachable from o (Celebi 2015). Every point that is not density-connected to any other is considered noise. This is one of the most well known algorithms for clustering problems as it can handle data clusters of different shape and size and is also resistant to noise.

- **Hierarchical Clustering (HC)**
  HC uses linkage rules to produce a hierarchical sequence of clustering solutions. It can be either *agglomerative* (the most common, depicted in *Figure 3.13*), sequentially merges clusters, or *divisive*, where it sequentially splits them. The evaluation of the clusters similarity can be made with the following distances (Sa 2001):

  - **Euclidean norm** - $||x - m||_e = (\sum_{i=1}^{d}(x_i - m_i)^2)^{\frac{1}{2}}$
  - **Squared Euclidean norm** - $||x - m||_s = \sum_{i=1}^{d}(x_i - m_i)^2$
  - **City Block norm** - $||x - m||_c = \sum_{i=1}^{d} |x_i - m_i|$
  - **Chebychev norm** - $||x - m||_C = max_i(|x_i - m_i|)$
  - **Minkovsky norm** - $||x - m||_p = (\sum_{i=1}^{d}(x_i - m_i)^p)^{\frac{1}{p}}$

The linkage function uses the distance calculated by these formulas for any two pairs of objects to determine the proximity of objects. They can be one of the following (Sa 2001):

- **Single Linkage**
  The distance between two clusters is determined by the distance between the two nearest objects between them

$$d(w_i, w_j) = \min_{x \in w_i, y \in w_j} ||x - y|| \tag{3.25}$$

- **Complete Linkage**
  The opposite of the *Single Linkage*, it is determined by the distance between the two furthest objects between the clusters

$$d(w_i, w_j) = \max_{x \in w_i, y \in w_j} ||x - y|| \tag{3.26}$$

- **Group Average Linkage**
  The distance between two clusters is determined by the average of the distances amongst all pairs in them

$$d(w_i, w_j) = \frac{1}{C(n_i + n_j, 2)} \sum_{x, y \in \{w_i, w_j\}} ||x - y|| \tag{3.27}$$

- **Wards Linkage**
  Minimizes the variance of the cluster obtained by merging the two clusters

$$d(w_i, w_j) = \frac{1}{n_i + n_j} \sum_{x \in \{w_i, w_j\}} ||x - y||^2 \tag{3.28}$$

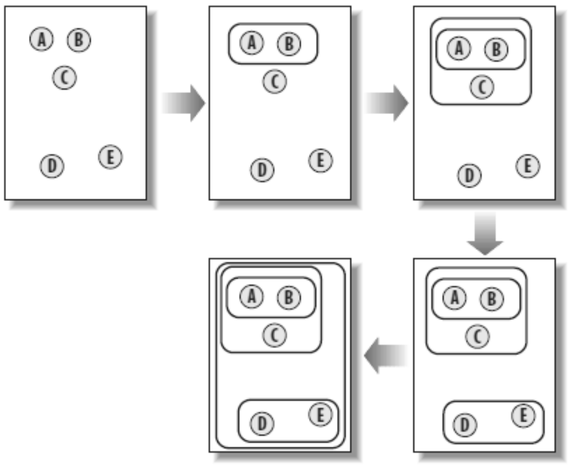


Figure 3.13: Agglomerative Hierarchical Clustering (HC), from (Segaran 2007)

- **Clustering Using REpresentatives (CURE)**
  CURE is an agglomerative hierarchical algorithm. Usually single-linkage implementation of bottom-up hierarchical algorithms are able to discover cluster of arbitrary shape

through the computation of distances between all pairs of points in the clusters for merging. CURE instead uses a set of representatives carefully chosen amongst the most distant to the center of the cluster and that represent its shape. Afterwards they are shrunk towards the center of the cluster reducing the impact of the outliers (Aggarwal 2015). CURE measures the similarity of two clusters based on the representative points without considering the internal closeness, such as density or homogeneity.

- **k-Means**
  k-Means is a centroid adjustment algorithm, as it iteratively tries to adjust the centroids of its clusters, which number must be defined beforehand. It begins with randomly placed centroids, and assigns every item to the nearest one. Afterwards, iteratively, they are moved into the average location of all the nodes assigned to them, and the assignments are done once again, until there are no more changes in the clusters, minimizing the function:

$$E = \sum_{j=1}^{C} \sum_{x_i \in w_j} ||x_i - m_j||^2 \tag{3.29}$$

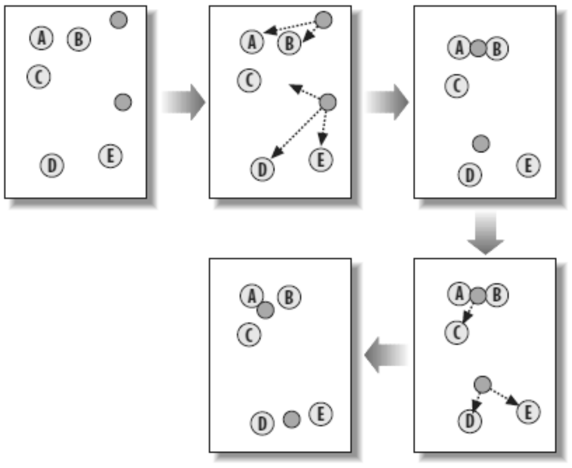


Figure 3.14: k-Means, from (Segaran 2007)

3.5 Algorithm Evaluation

A learning model is good if it produces good predictions on unseen examples. To do so, its performance is usually measured through its error / success rate.

Albeit some of the problems have undifferentiated costs for different errors, that is, the cost of misclassifying A as B is the same as misclassifying it as C (also know as a 0/1 loss function), that is not the case for some more complex problems. It may be the case that wrong decisions are not equally costly, requiring the implementation of a more complex error loss function to account for the problem specifics.

Despite the fact that the errors are one of the main criteria for evaluating an algorithm, it should be kept in mind that there are many others, some of them even dependent on the problem in hand. Some of them are (Turney 2002):

- the risk when errors are generalized using loss functions instead of 0/1 loss
- training time and space complexity
- testing time and space complexity
- interpretability, that is, whether the method allows knowledge extraction that can be checked and validated by experts
- easy programmability

Despite all of the factors that we may take into account when evaluating an algorithm, one should not forget that whatever conclusions that may arise are conditioned by the dataset with which the model was developed. Also, the comparison that can be made using the results is not domain independent, because we are not comparing the expected error rates of a learning algorithm in general, rather for a specific application and only as long as the sample used represents the target application. When it is said that a classification algorithm is good, it is only a qualification of how well its inductive bias matches the properties of the data (Alpaydin 2014). In fact, there is no universal algorithm that is on average the best for every situation, as stated by the *No Free Lunch Theorem* (Wolpert and Waters 1994).

3.5.1 Training and Testing

A fundamental problem in ML is how to obtain a realistic estimate of the prediction error of a model. This task is of particular relevance when the dataset is not large and the underlying distribution is not known (Borra and Di Ciaccio 2010). This estimate is important as it is based on its value that a model will be chosen instead of others due to having a better prediction performance.

As was previously stated, Machine Learning (ML) usually works with two main datasets: training, and test. The training set contains the data that are going to be used to fit the model while it is in training. Within the training set, a subset might also be used for **validation**, that is, it will be used to estimate the overfitting of the model. The test set estimates the generalization error of the final model (Hastie, Tibshirani, and J. Friedman 2009). The division between these two sets is not trivial, as when the model is in training it will not have access to the test set, hence the selection of the datasets must be representative of the problem. To create both sets there are some options (Sa 2001):

- **Resubstitution Method** - this method uses the whole set of data both for the training and testing of the algorithm. Due to the fact that they lack independence the error estimates obtained from this method are usually optimistic.
- **Holdout Method** - the available samples are randomly divided into two sets, normally with half each.
- **Partition / Leave-one-out Methods** - this approach divides the data in equally sized subsets which then rotate, serving for both training and testing the models. This way the subsets are independent of one another giving better error estimates for a big enough number of subsets.

  Cross Validation (CV) can be used to improve the prediction. Briefly, after the data is split in k disjoint subsets the model is then trained k times. Each time one of the subsets is left out and then used to compute the prediction error. For partition methods sometimes it is also important to pay attention to the representation of the classes in the resulting folds so that it does not disturb their prior probabilities. This is called *Stratification* (Alpaydin 2014). The final performance of the model is then the average performance obtained on each fold.

- **Bootstrap Method** - this method bases itself in the generation of artificial samples by randomly extracting samples with a uniform distribution for each class. In this case the error estimates are calculated with the original data set and the algorithm trained with large sets of the bootstrap samples, achieving estimates similar to the partition methods (Sa 2001).

With every model created using a dataset a problem arises, that is known as the *bias / variance dilemma*. Bias is a source of error derived from wrong assumptions made by the algorithm, as variance is a source of error due to the sensitivity to changes in the training set. Its behavior can be observed in *Figure 3.15*.

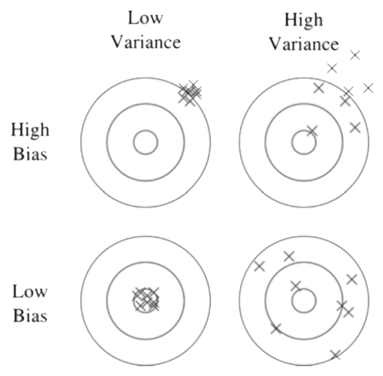


Figure 3.15: Bias and Variance, from (Domingos 2015)

A flexible model will allow for a low bias, although risking a higher variance. The opposite also happens, that is, if the variance is kept low the model may not fit the data and have a high bias. The dilemma part of this relation is to find a balance in the trade-off between the bias and the variance.

As a result of the existence of bias and variance in a model, two problems may derive from them, *underfitting* and *overfitting* respectively. **Overfitting** is the most common of both, and every model has a tendency to have it. It happens when the developed model is too flexible, learned too much, noise and random events included. This negatively impacts the performance of the model to new data as it limits its ability to generalize. On the other hand, when it can neither generalize nor can it not model the training data it is called **Underfitting**.

3.5.2 Performance Metrics

In order to evaluate an algorithm its performance must be measured. These can be divided by algorithms types, either *Classification* or *Regression*.

Classification Metrics

- **Confusion Matrix (CM)**
  If the loss function for the problem at study is binary, either 0/1 loss, all losses are equally bad, and the error calculations are based on a confusion matrix, as can be seen in *Figure 3.16*. They summarize the performance of a classifier in a two-dimensional matrix, indexed in one dimension by the true class and in the other by the predicted class (Sammut and Webb 2011). The samples that are correctly predicted are also known as **True Positive (TP)** and **True Negative (TN)**. The *positive* samples that are predicted as *negative* are the **False Negative (FN)** and the opposite are the **False Positive (FP)**.

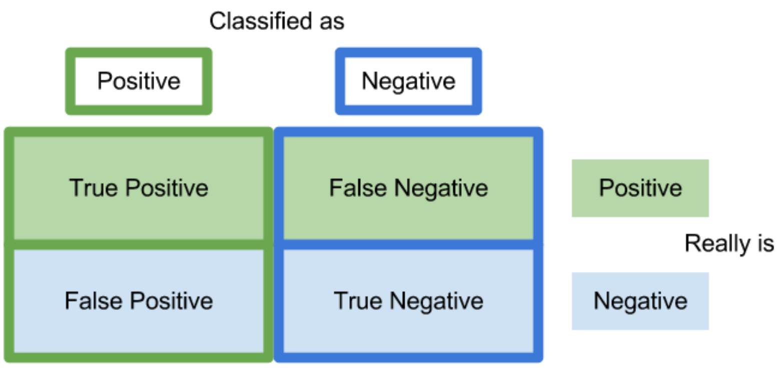
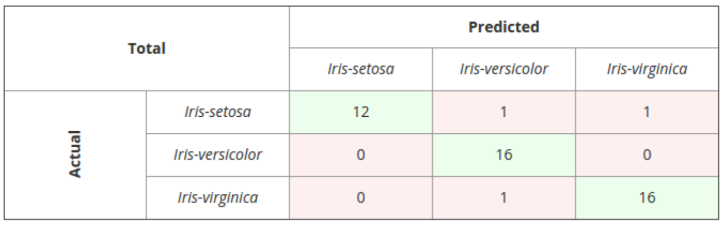


Figure 3.16: Confusion Matrix, from (*Wordpress Confusion Matrix* 2012)

- **Class Confusion Matrix**
  If the algorithm is intended to work on more than 2 classes, the previous binary confusion matrix should be replaced by a class class confusion matrix, which is a K x K such that its entry (i, j) contains the number of instances that belong to class i but were classified as class j as in *Figure 3.17*. This matrix allows to visualize which classes are being more commonly confused and to which classes.



Total		Predicted		
		Iris-setosa	*Iris-versicolor*	*Iris-virginica*
Actual	*Iris-setosa*	12	1	1
	Iris-versicolor	0	16	0
	Iris-virginica	0	1	16

Figure 3.17: Class Confusion Matrix, from (*WSO2 Class Confusion Matrix* 2012)

- **Error Rate**
  As such the error rate can be calculated as in *Formula 3.30*, where $N = TP + FP + TN + FN$, which represents the total number of validation samples. For problems that require a specific loss function this should be modified to take that into account.

$$ErrorRate = \frac{FN + FP}{N} \tag{3.30}$$

- **Loss Functions**
  These functions are used to represent the cost paid for inaccurate predictions, in order to minimize the expected risk. Some of the most common loss functions are *Square Loss*, *Hinge Loss*, *Logistic Loss* and *Cross Entropy Loss* (Hastie, Tibshirani, and J. Friedman 2009).

- **Accuracy**
  Accuracy is the most common metric to evaluate the performance of an algorithm. Its formula 3.31 creates a ratio of all the correct predictions made, $TN+TP$ related to all the predictions made, N. This ratio is only useful, and accurate, when there is a balanced number of observations in each class, and that both the predictions and prediction errors have the same importance.

$$Accuracy = \frac{TP+TN}{N} \tag{3.31}$$

- **Precision and Recall**
  A very common metric that measures the performance of an algorithm is the precision / recall, where both of them give a relation of the occurrences that the algorithm found. Precision 3.32 relates the correct occurrences the system found of a class, TP, against all the occurrences of that class that it found, $TP+FP$.

$$Precision = \frac{TP}{TP+FP} \tag{3.32}$$

  Recall 3.33 on the other hand relates the number of corrected occurrences the system found, TP, against all the occurrences of that class in the data, $TP+FN$.

$$Recall = \frac{TP}{TP+FN} \tag{3.33}$$

- **F-Score**
  This metric combines the precision and recall in one single function, giving a single value for a performance of an algorithm. The f-score, also known as the *balanced f-measure*, is the harmonic mean of the precision3.32 and recall3.33.

$$F-score = \frac{2*Precision*Recall}{Precision+Recall} \tag{3.34}$$

- **Sensitivity and Specificity**
  These metrics are usually used in medical areas, for example to identify if a patient has a given decease. Sensitivity is exactly the same as the recall3.33, however specificity 3.35 measures how good a model is in not finding what it doesn't want want to find. A system with a high specificity will have a very low number of false alarms.

- **Receiver Operating Characteristics (ROC)**
  To optimize a classifier it is usual to use a ROC curve, that displays the hit rate (sensitivity), $TP/(TP+FN)$, versus false alarm rate, $FP/(FP+TN)$, and has as form similar to 3.18. Each classification algorithm has a parameter, such as a threshold of decision, that can be tweaked to change the number of true positives versus false

positives. As there is an increase in true positives theres is also an increase in false alarms, and vice-versa, and depending on the gain or cost for the problem in hand is chosen a threshold that represents a point on this curve. Using the ROC curve it is also possible to calculate the Area Under Curve (AUC) metric, which is also a good single number performance for a system's performance.

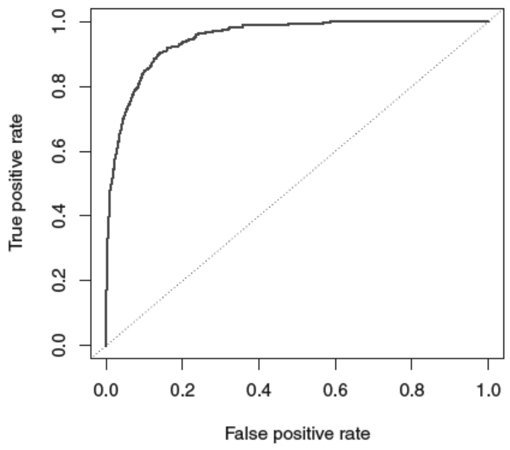


Figure 3.18: ROC Curve, from (James et al. 2014)

$$Specificity = \frac{TN}{TN + FP} \tag{3.35}$$

Regression Metrics

- **Mean Absolute Error (MAE)** 3.36 - it is the sum of the absolute differences between the prediction values and the real ones. It gives an idea of the dimension of the error but it does not take into account its direction.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{Y}_i - Y_i| \tag{3.36}$$

- **Mean Squared Error (MSE)** 3.37 - similarly to the MAE it gives an idea of the magnitude of the error, by averaging the squares of differences.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{3.37}$$

- **Coefficient of Determination** 3.41 - this metric, also known as *R Square* or *R^2*, provides an indication of how good of a fit a set of predictions is to the real values, based on the proportion of the total variation of outcome explained by the model.

$$\bar{Y} = \frac{1}{n}\sum_{i=1}^{n}Y_i \tag{3.38}$$

$$SS_{tot} = \sum_i (Y_i - \bar{Y})^2 \tag{3.39}$$

$$SS_{res} = \sum_i (Y_i - \hat{Y}_i)^2 \tag{3.40}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{3.41}$$

Clustering Metrics

Evaluating the performance of an algorithm for a clustering problem is not as straightforward as for classification or regression. Clustering metrics can be divided in two: **Internal Validation** and **External Validation**. External validation can be used if the data contains the targets of each instance. Some of such metrics are *Rand Index*, *Mutual Information*, *Homogeneity, Completeness* and *V-Measure* as detailed in (Tan, Steinbach, and Kumar 2005). In this scenario the metrics used for evaluation of the classification problems can also be used. On the other hand if the targets are not known, as it happens with most clustering problems, then the internal validation metrics must be used (Aggarwal 2015). Unsupervised validation criteria can still be further divided into two classes: **cohesion** and **separation** measures. *Cohesion* determines how closely related are the objects in the cluster, and *separation* how distinct and separated the cluster is from the remaining (Tan, Steinbach, and Kumar 2005). Some of these metrics are:

- **Sum of Square Errors (SSE)** - a measure of cluster *cohesion*, it measures how close are the items in a cluster to its centroids. It is probably the most common measure, for each point, the error is the distance to the nearest cluster (Tan, Steinbach, and Kumar 2005):

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist(m_i, x)^2 \tag{3.42}$$

  where x is a data point in cluster C_i, and m_i is the representative point for cluster C_i

- **Silhouette Coefficient** - this metric combines both *cohesion* and *separation*, that is, how close are the items in the cluster and how far apart are the items from each other. With this method each cluster is represented by its silhouette, through the positions of the instances that lie within the cluster. The silhouette coefficient for a single sample x_i is (Walde 2003):

$$S(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{3.43}$$

  where $a(x_i)$ is the mean distance between the instance x_i and all other points in the same cluster (within-cluster distance), and $b(x_i)$ is the mean distance between x_i and all other points in the nearest cluster (between-cluster distance). The silhouette score of a cluster is given by the mean of the individual scores of its samples as in *Equation 3.44*, and the score for all the data is given by the mean of the clusters scores, as in

Equation 3.45 (Walde 2003).

$$S(C_i) = \frac{1}{|C_i|} \sum_{x_j \in C_i} S(x_j) \tag{3.44}$$

$$S(C) = \frac{1}{k} \sum_{i=1}^{k} S(C_i) \tag{3.45}$$

3.5.3 Assessing and Comparing Algorithms Performance

After implementing and compiling the results for the target algorithms there is a need to compare them. For that there are several tests, depending on the characteristics of the data as represented by the graph in *Figure 3.19*. In short, the idea is to postulate that the performances of the algorithms to compare are the same, which is known as the *null hypothesis*, H_0, and ensure that the difference in performance between them is statistically significant, with some value of confidence.

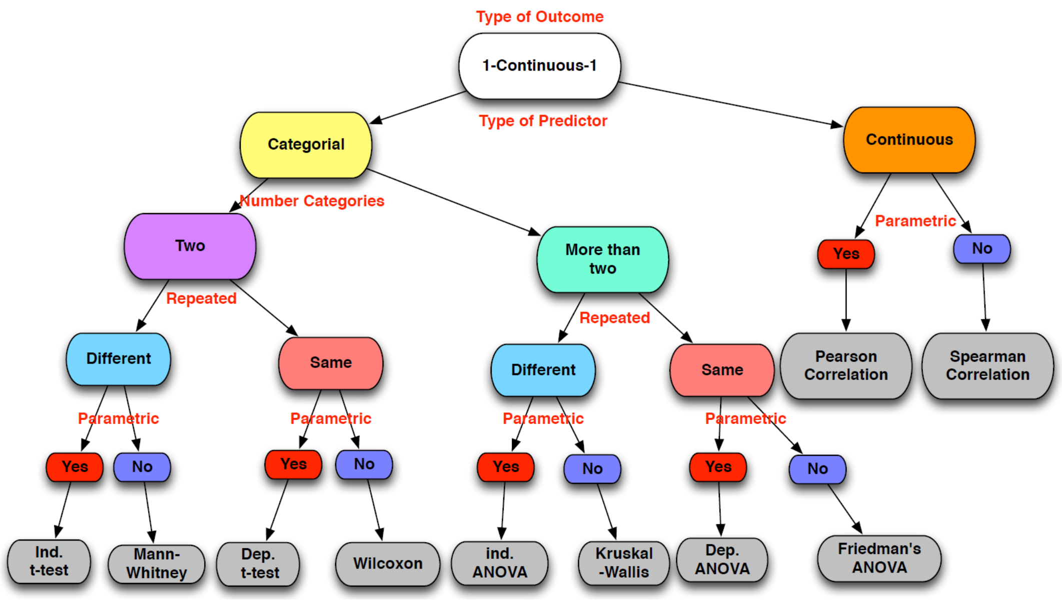
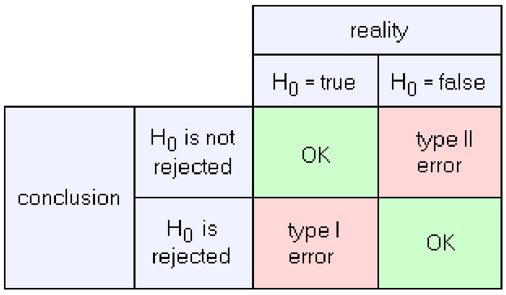


Figure 3.19: Statistical Tests

Due to the representation we have from the sample data, our conclusion on keeping or rejecting H_0 may be wrong, and those errors may be divided in two types 3.20. In the *Type II* error the (wrong) decision is to retain H_0, which is the same as doing nothing. However the *Type I* error is rejecting H_0, which means wrongfully rejecting a previous idea of truth (Field 2013).

When there are multiple algorithms that need to be compared it is necessary to use specific tests to do so. However, as multiple hypothesis testing requires several comparisons the probability of getting a significant result by chance adds up. To deal with that some corrections must be applied in order to adjust the confidence level so that the probability of observing significant results remains below the intended level, such as the *Bonferroni correction* and *Holm procedures* (Field 2013).



		reality	
		H_0 = true	H_0 = false
conclusion	H_0 is not rejected	OK	type II error
	H_0 is rejected	type I error	OK

Figure 3.20: Hypothesis Errors, from (*Fundamentals of Statistics* 2012)

It is worth noticing that when comparing different classification algorithms we are only testing whether they have the same expected error rate. If they do it does not mean that they made the same mistakes. The standard applications of these tests usually consider that all the misclassifications have the same cost, and if that is not the case the tests need to take into account the loss function for the problem.

Chapter 4

Biological Experimental Setup

As it was described in *Section 2.2*, there are several ways to identify Drug Induced Mitochondrial Dysfunction (DIMD). For this work the techniques used were the analysis of the *Oxygen Consumption and Extracellular Acidification Measurements of Cells 2.2.1*, combined with the *Measurement of ATP Levels in Cells Grown in Glucose / Galactose Media 2.2.1*.

4.1 Context

All cells have the ability to accelerate the metabolic processes and in aerobically poised cells mitochondial OXPHOS has the capacity to exceed bioenergetic demand. During exercise or stress they can increase the production of ATP as needed, up to a maximum. The reverse also happens, that is, they can halt OXPHOS to the minimum required to cell survival. Drug exposure however can impair the mitochondrial capacity to adapt and at some point a bioenergetic threshold is crossed, that leads to cell death (Dykens and Will 2008).

Tolcapone is a drug used in the treatment of Parkinson's disease. It is an effective drug, however it has been inferred that induces hepatotoxicity. After approximately 1 year on the market it led to three fatalities which granted it a withdrawal and black box warnings in some countries (Chen 2011). Tolcapone's toxicity has been associated with mitochondrial dysfunction due to ultrastructural changes it presents in the mitochondria (Sardão and Tiago B. Silva 2016). Another drug used in this study is Entacapone, which is similar to Tolcapone and is also used in Parkison's treatment, though is not associated with hepatotoxicity (P 2000). In order to surpass the Tolcapone toxic limitation the MitoXT group developed new compounds with profile similar to both Tolcapone and Entacapone (Silva et al. 2016). Based on an initial analysis there were indications that they present lower *in vitro* toxicity when compared with Tolcapone (Sardão and Tiago B. Silva 2016)

To analyze the toxicity of the new drugs the OCR and ECAR of living cells exposed to them were measured using the *Mito Stress* test and Glycolysis Stress test from Seahorse, ran in a XFe96 Extracellular Flux Analyzer. The set of drugs tested were: Tolcapone, Entacapone, and the new drugs, ABA, ABE, APA and APE (Sardão and Tiago B. Silva 2016).

4.2 Description

Assays that measure ATP levels to analyze cell viability are routinely used to investigate drug toxicity. Usually the cells used for such assays are done with immortalized cell lines due to its ease of usage. However, despite these cells having the ability to generate ATP aerobically (that is, through the mitochondria) they tend to generate it through glycolysis, which makes it harder to identify mitochondrial toxicants. To bypass this, cells can be grown in galactose where the net yield of glycolysis is zero, forcing them to use the OXPHOS pathway to generate ATP (Rotella 2012). Nonetheless, in the experiments used for this project both media were used to distinguish the toxic effects primarily through mitochondrial targets from those that cause toxicity from multiple targets.

The consumption of oxygen in the cells is a way to analyze how much Oxidative Phosphorylation (OXPHOS) is happening, and consequently ATP synthesis. However, the sole measurement of oxygen consumption is not enough to identify compound toxicity since there are other situations that can cause it to be altered, such as cell death. To complement the analysis pH sensors are used to measure the Extracellular Acidification Rate (ECAR), and therefore, the glycolytic rate of cells. With this information it provides the necessary data to distinguish between compounds that impair mitochondrial function from those that cause general cytotoxicity. This is due to the fact that generally there is an increase in the glycolytic rate when OXPHOS is impaired, whereas this does not happen when there is only cytotoxicity (Rotella 2012).

The *Mito Stress* test measures the OCR of the cells when exposed to specific compounds and intends to highlight particular indicators of the cells viability, as it can be seen in *Figure 4.1*. In order to measure the amount of oxygen that is being consumed due to the OXPHOS and as it is not the only process that uses oxygen, it is necessary to identify what is being consumed by other processes. To do so, inhibitors of the electron transport chain are added, and this is known as **Non-Mitochondrial Respiration**. The **Basal Respiration** is the respiration used to meet the endogenous ATP demand of the cell. **Proton Leak** respiration is due to protons that leak through the membrane but are not used to produce ATP. The **ATP-linked Production** is the respiration that is used to drive the ATP synthesis. **Maximal Respiration** corresponds to the maximum respiration rate of the cells working at full capacity. **Reserve / Spare Respiratory Capacity** is the difference between the basal and the maximal, and it indicates the ability of a cell to meet an increased energy demand (Divakaruni et al. 2014).

To measure the glycolytic pathway the *Glycolysis Stress* test was used. Glycolytic turnover is associated with acidification of the extracellular medium, hence the measurement of extracellular pH is used as an indicator of the glycolysis. Similarly to the oxygen consumption this test measures the changes in pH when the cells are exposed to specific compounds *Figure 4.2*. In order to identify the variations due to the glycolytic pathway it is necessary to find the **Non-Glycolytic Acidification**, which corresponds to the acidification made by other processes. The **Glycolysis** measurement represents the ECAR reached after the addition of saturating amounts of glucose. **Glyocolytic Capacity** is the maximum ECAR when OXPHOS is halted, driving the cells to use glycolysis at its maximum capacity. Lastly, **Glycolytic Reserve** indicates the cells ability to respond to an energetic demand (Seahorse 2017).

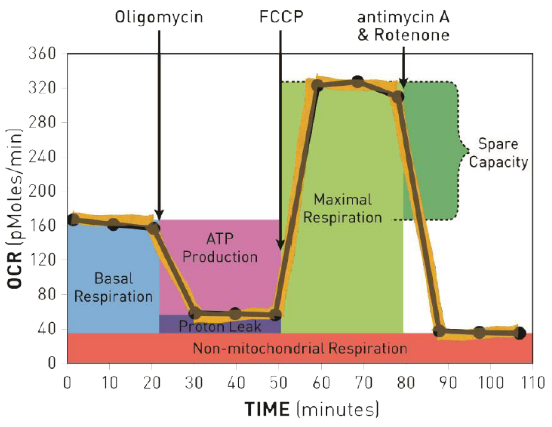


Figure 4.1: Respiration Function, from (*Agilent Technologies* 2016)

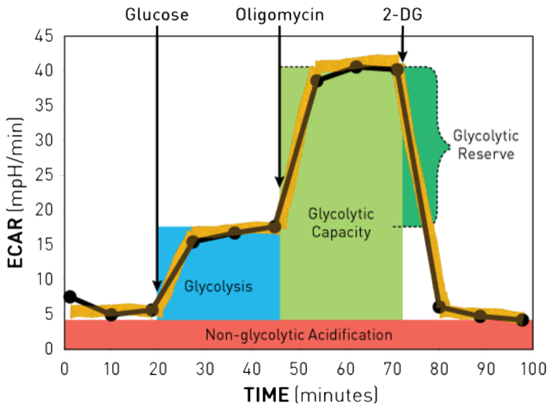


Figure 4.2: Glycolytic Function, from (*Agilent Technologies* 2016)

4.3 Setup

For these tests plates with dimensions 12 * 8, 96 wells, were used to hold the solutions for analysis, similar to the one in *Figure 4.3*.

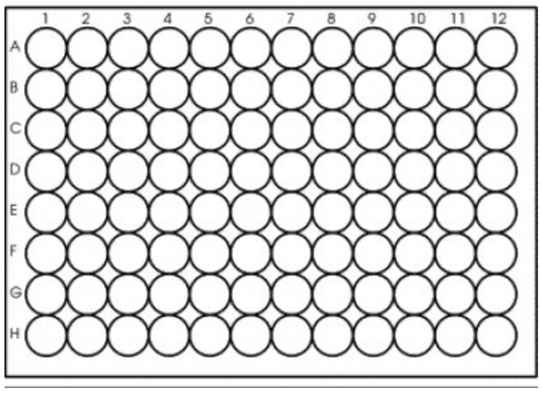


Figure 4.3: XF96 microplate, from (*WK Labe - Seahorse XFe96* 2017)

Each corner well does not contain any compound and is used as reference (A1, A12, H1 and H12). In total there were 3 *Mito Stress* and 4 *Glycolysis Stress* tests. The number of experiments for each test and combination of drug concentrations (10 and 50 μM) and media (glucose and galactose) are in *Table 4.1* and *Table 4.2*. These concentrations were defined based on a dose response analysis to identify which concentrations induced cell death. With 50μM it was already possible to measure some cell death whilst with 10μM it was not.

4.4 Results Processing

The *Mito Stress* test and the *Glycolysis Stress* test generate two files for each run: one where the configurations and measurements taken are stored, and another which holds the cell concentrations for each well. Both files come in a spreadsheet format with multiple sheets.

Medium	**Drug**	**Conc. μM**	**# Exp.**		**Conc. μM**	**# Exp.**
Glucose	Control		42			
Glucose	ABA	10.0	8		50.0	8
Glucose	ABE	10.0	8		50.0	8
Glucose	APA	10.0	8		50.0	8
Glucose	APE	10.0	8		50.0	8
Glucose	Entacapone	10.0	8		50.0	8
Glucose	Tolcapone	10.0	8		50.0	8
Galactose	Control		44			
Galactose	ABA	10.0	8		50.0	8
Galactose	ABE	10.0	8		50.0	8
Galactose	APA	10.0	8		50.0	8
Galactose	APE	10.0	8		50.0	7
Galactose	Entacapone	10.0	8		50.0	7
Galactose	Tolcapone	10.0	8		50.0	7

Table 4.1: *Mito Stress* - Experiments per Drug

Medium	**Drug**	**Conc. μM**	**# Exp.**		**Conc. μM**	**# Exp.**
Glucose	Control		68			
Glucose	ABA	10.0	10		50.0	10
Glucose	ABE	10.0	10		50.0	10
Glucose	APA	10.0	4		50.0	4
Glucose	APE	10.0	6		50.0	6
Glucose	Entacapone	10.0	14		50.0	14
Glucose	Tolcapone	10.0	14		50.0	14
Galactose	Control		69			
Galactose	ABA	10.0	10		50.0	10
Galactose	ABE	10.0	10		50.0	10
Galactose	APA	10.0	4		50.0	4
Galactose	APE	10.0	6		50.0	6
Galactose	Entacapone	10.0	14		50.0	14
Galactose	Tolcapone	10.0	14		50.0	12

Table 4.2: *Glycolysis Stress* - Experiments per Drug

The results files have a matrix on the first sheet with the labels of the experiment for each well, representing what was in each one (i.e. a well with the description "Tolcapone 10μM-Starving media Glu" means that in that well there was a solution with 10μM of Tolcapone on a glucose medium). Both experiments make 12 consecutive measures timed to execute on relevant time frames, such as the addition of a new compound, as described in *Section 4.2*. There are two types of measures that are collected and important for the project: the Oxygen Consumption Rate (OCR) and the Extracellular Acidification Rate (ECAR). On the second sheet there are 12 * 2 matrices of 96 cells each with the values collected for each well at each measurement.

Because it is not possible to guarantee that a given sample has a fixed number of cells in it, the returned results can not be used directly. Hence, the normalization files are necessary in order to normalize the values that are measured based on the cellular concentration. In these files there is a matrix with dimensions 12 * 8 (96) that matches the tray used in the experiment and in each cell there is the cellular concentration for the respective well.

For these data to be used to develop the predictive model they must be gathered and stored in a way that better fits the needs of the process. It should follow a structure where the first few columns refer to the initial conditions, such as, the compounds present in the solution and the type of the medium, and the remaining columns would be all the measured values, similar to *Table 4.3*.

IC1	**IC2**	**IC3**	**MV1**	**MV2**	**MV3**	**Target**

Table 4.3: Data Structure

Primarily the data that was delivered was being parsed by hand by the MitoXT group. However, due to the latency and required work to do so and the amount of data that needed to be converted, a specific parser was created to export the data from the experiments into the required structure. To do so, after a thorough analysis of the tests specifications and with expert support it was established that the measurements that were required to identify the addition and effects of the compounds were: 3rd, 6th, 9th and 12th, for both tests. Briefly, for each test the parser works as follows:

1. Open each experiment results file
   1.1. Load the labels 12 * 8 matrix present in the first sheet into a multidimensional array
      - parse the label to extract the compound, dosage and medium

   1.2. Parse each required measurement (3rd, 6th, 9th and 12th) 12 * 8 matrix in the measurements sheet
      - store each measurement for each well in the multidimensional array on the corresponding index (i.e. [0,0], [0,1], ...) both for the OCR and ECAR

   1.3. Open the normalization file of the experiment
      i. parse the 12 * 8 normalization matrix
      ii. store each cell value on the corresponding index in the multidimensional array
2. Save the all the gathered data for the test into a unique file in the required structure
   - concentration of the compound in the well
   - medium and its concentration
   - 4 measurements for OCR
   - 4 measurements for ECAR
   - well cell concentration
   - target of the sample (the compound it was exposed to)

Although the logic behind the parser was fairly simple, most of the files did not follow a standard naming convention nor all the tables followed the same positioning across files. This required a thorough and lengthy debugging of all the files in order to generalize the parser to all the variants that existed, as well as to detect small differences that would be otherwise impossible to detect. Examples of the experiments data tables can be seen in *Appendix A*.

Chapter 5

Data Preprocessing and Experimental Settings

To create the best possible models there are a variety of techniques that can be used. Although some of them do not have hyperparameters to configure some others require a prior analysis to choose the best parameters.

5.1 Data Preprocessing

To develop models that can make accurate predictions and are able to generalize the data need to be prepared. To do so, it is necessary to thoroughly analyze them and choose the adequate techniques that will improve their performance.

The data available from the experiments was stored in two files: one for the *Mito Stress* test and another for the *Glycolysis Stress* test, as explained in *Section 4*.

5.1.1 Data Cleaning & Transformation

Data cleaning and transformation is a process that can take up to 80% of the time involved in a ML project (Dasu and Johnson 2003). It is not just a "first-step" either, it must repeated many times during the development of the project, as new problems are identified or new data is collected (Wickham 2014). Without this the models may not work as expected and / or lead to wrong or invalid conclusions. To do so, a careful study and analysis of the problem was done with the support and expert knowledge of the MitoXT team.

After an initial analysis of the data it was observed that there were several wells in the experiments with low cell concentrations. If the concentration of the well is too low the measurements obtained will not be accurate due to lack of sensitivity, hence introducing noise in the data. Therefore, wells that presented a concentration below a given threshold defined by the experts (0.2) were removed. The data from the *Glycolysis Stress* tests were the ones that had more invalid samples. The remaining valid experiments can be seen in *Table 5.1*. By comparing with the initial results, seen in *Table 4.2*, the main losses in experiments were in the galactose medium, where some of the setups lost up to 50% of samples.

Outliers were also analyzed, however the choice was not to remove them. Due to the fact that this is a biological dataset an outlier may have relevance, indeed they could even be the

Medium	**Drug**	**Conc.** μM	**# Exp.**		**Conc.** μM	**# Exp.**
Glucose	Control		68			
Glucose	ABA	10.0	10		50.0	9
Glucose	ABE	10.0	10		50.0	10
Glucose	APA	10.0	4		50.0	4
Glucose	APE	10.0	6		50.0	6
Glucose	Entacapone	10.0	14		50.0	14
Glucose	Tolcapone	10.0	14		50.0	14
Galactose	Control		37			
Galactose	ABA	10.0	5		50.0	4
Galactose	ABE	10.0	7		50.0	6
Galactose	APA	10.0	4		50.0	4
Galactose	APE	10.0	4		50.0	4
Galactose	Entacapone	10.0	12		50.0	10
Galactose	Tolcapone	10.0	10		50.0	9

Table 5.1: *Glycolysis Stress* - Remaining Experiments per Drug

most important data. An outlier may represent an instance of an adverse effect, which will usually have a very low occurrences and may deviate strongly from the the rest of the data.

Another important step was the analysis of the consistency of the data. Once again, based on the literature and expert knowledge, the data were thoroughly analyzed to detect invalid values, be them by the values themselves or by relation to other measurements. By doing so some situations were detected, which were then formally justified by the experts. One of those was a *Glycolysis Stress* test which gave results that were visibly different from the remaining, which are those marked as outliers in *Figure 5.3*. In biology there is always the possibility of deviations amongst experiments in similar conditions. For this case, some of the possible explanations are that small changes in the incubation of the cells, such as temperature and CO_2 variations, may cause different media conditions which may lead to different cell metabolism and proliferation. Another factor that may cause the dispersion is that the cell lines used in the experiments have proliferation capabilities. That means that the cells from one week are not the same as the next, as these will be the result from the division of the previous cells. Another situation that appeared was the existence of negative ECAR values. As these refer to the extracellular acidification they should not be negative. After expert advise these values were identified as measurement errors due to lack of sensitivity and set to 0. Lastly, the relations between the measurements were controlled (i.e. if the maximum measure was lower than the basal, which normally should not happen). There were some samples that had situations like this, and with expert support they were reviewed one by one. Most of them fell under the category of sensitivity, that is, the differences were so small that they were considered the same. The remaining cases did not have such a direct explanation, however they were kept as they could, for example, represent a change in cell concentration during the experiment due to cell death caused by the drug, and as such be a relevant measure.

In the end the values used to create the models were normalized using a z-score normalization.

5.1.2 Data Analysis

After cleaning and validating the data the first step was to perform a *descriptive analysis*. With it is possible to observe how the features are distributed along their values. The means and standard deviations for the *Mito Stress* test and the *Glycolysis Stress* test are present in *Table 5.2* and *Table 5.3*

Feature	**Mean**	**Std. Deviation**	**Range [min,max]**
OCR_BAS	549	93	[284, 841]
OCR_OLI	243	60	[145, 607]
OCR_FCCP	977	287	[358, 1667]
OCR_ROT	144	27	[79, 215]
ECAR_BAS	82	28	[25, 144]
ECAR_OLI	154	34	[72, 252]
ECAR_FCCP	192	37	[91, 287]
ECAR_ROT	151	71	[28, 298]

Table 5.2: *Mito Stress* - Descriptive Analysis

Feature	**Mean**	**Std. Deviation**	**Range [min,max]**
OCR_BAS	708	625	[93, 2478]
OCR_GLU	594	517	[95, 2038]
OCR_OLI	528	524	[75, 1981]
OCR_2DG	483	496	[73, 1960]
ECAR_BAS	71	46	[3, 205]
ECAR_GLU	181	105	[0, 416]
ECAR_OLI	209	117	[14, 457]
ECAR_2DG	94	58	[0, 264]

Table 5.3: *Glycolysis Stress* - Descriptive Analysis

As it can be seen in *Table 5.2* the OCR values have a wide range of values. Some of the features, such as OCR_FCCP, have a high standard deviation when compared to their means. For *Table 5.3* however there are some things that immediately catch the eye. All the OCR values have an excessively wide range of values and standard deviations, specially when comparing their respective means.

Hence, to observe the distribution of the data and its high variance, a study of the features distribution was made. From *Figure 5.1* it is possible to observe that there is a considerable amount of variance amongst the features regarding OCR.

In order to understand if the variance could be explained by the effects of the drugs and / or media, a boxplot of the features only with Control samples on a glucose medium was generate *Figure 5.2*. Although the dispersion is somewhat diminished, there is still a significant variance amongst the OCR values. After excluding the possibility that the data could be have been wrongly parsed, it was accepted that it is a consequence of the nature of the problem (biology), where diversity and variation are constant influences. This high variance also means that a good part of the features range for the different drugs will overlap with each other, making it difficult to distinguish the drug effects from natural diversity. Combining this with the fact that there are not many samples for each condition makes it particularly challenging for ML algorithms.

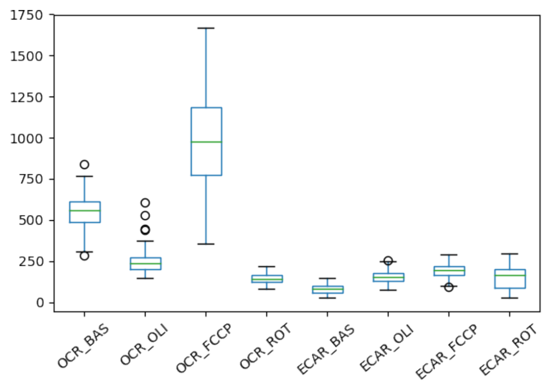


Figure 5.1: *Mito Stress* - Feature Boxplot

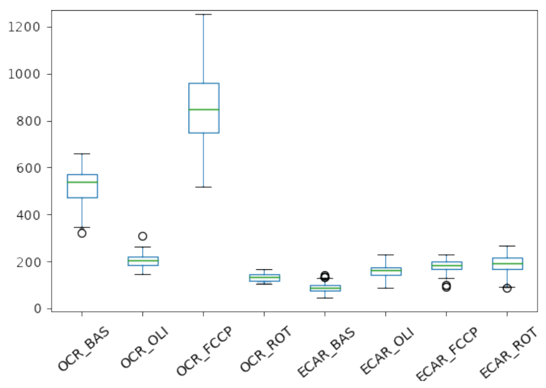


Figure 5.2: *Mito Stress* - Control Feature Boxplot

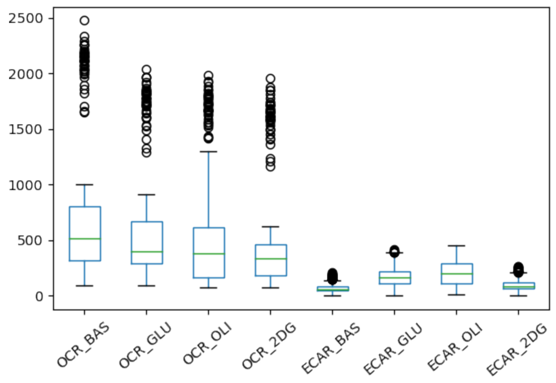


Figure 5.3: *Glycolysis Stress* - Feature Boxplot

The boxplot for the *Glycolysis Stress* test is yet another example of the inherent variability that exists in biological datasets. As it is visible in *Figure 5.3* there is a complete group of samples that are represented as outliers. Although every controllable experiment conditions are the same, due to factors that are not possible to control, such as cell reproduction, an entire experiment gives results that deviate considerably from the remaining. Nonetheless, the trend between the features / measurements appears to be the same across all experiments.

Then an analysis was made to the drug distribution of the samples, that is, how many items there were per drug per test. As it is possible to see in *Figure 5.4* and *Figure 5.5* the Control samples are in much larger numbers than the ones exposed to drugs. Another aspect that can be observed, mainly in *Figure 5.5* is the reduced number of samples for each class, specially taking into account that the numbers observed encompass both media (glucose and galactose) and concentrations (10μM and 50μM).

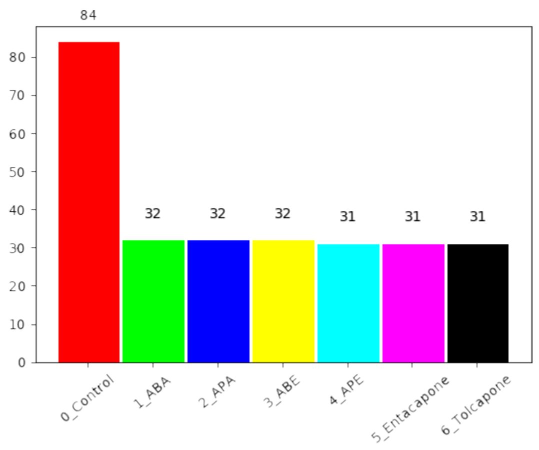


Figure 5.4: *Mito Stress* - Drugs Distribution

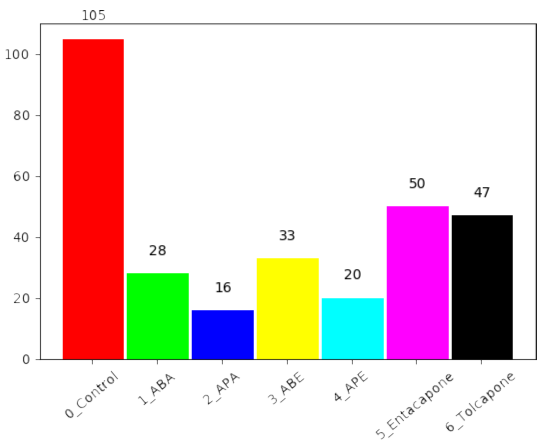


Figure 5.5: *Glycolysis Stress* - Drugs Distribution

Afterwards the averages by drug for both tests were plotted *Figure 5.6* and *Figure 5.7*. For the *Mito Stress* test the averages are very close to each other. For the *Glycolysis Stress* test however the variations caused by the drugs are more visible.

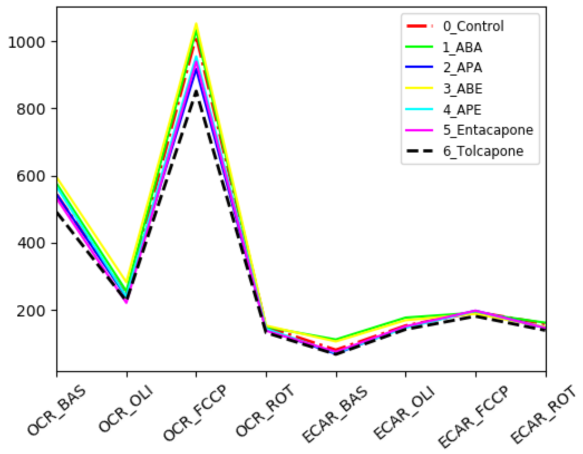


Figure 5.6: *Mito Stress* - Feature Averages by Drug

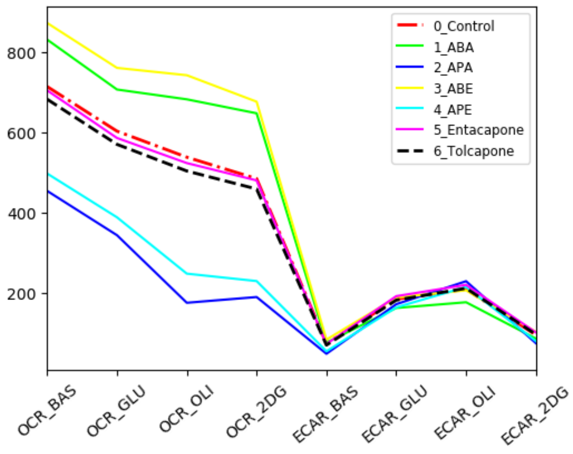


Figure 5.7: *Glycolysis Stress* - Feature Averages by Drug

Figure 5.8 and *Figure 5.9* represent the scatter plot matrix and correlation of each pair of features in the experiments data. The diagonal of the matrix shows a histogram of the feature values distribution. As it is illustrated in the *Figure 5.8* most of the features do not have high correlations (over 80%) with the exception of ECAR_BAS with ECAR_-OLI. On the other hand, for the *Glycolysis Stress* test, *Figure 5.9*, many features are highly correlated.

Due to the fact that both the *Mito Stress* test and the *Glycolysis Stress* test measure the same amount and type of events (4 OCR and 4 ECAR), the initial dataset used for this problem was a combination of both data. To do so another feature was added to identify the test a sample belongs to. The final structure of the dataset possesses 11 features and the target, which is the compound it was exposed to:

- 1 boolean feature to identify which test it refers to, either the *Mito Stress* or the *Glycolysis Stress*

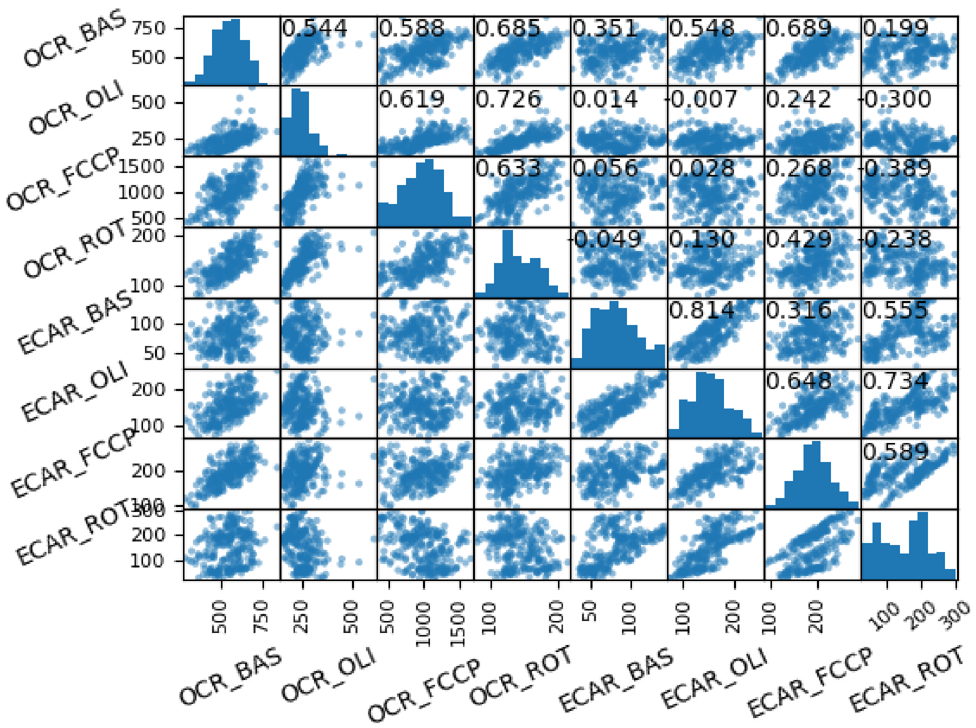


Figure 5.8: *Mito Stress* - Scatter Plot and Correlation

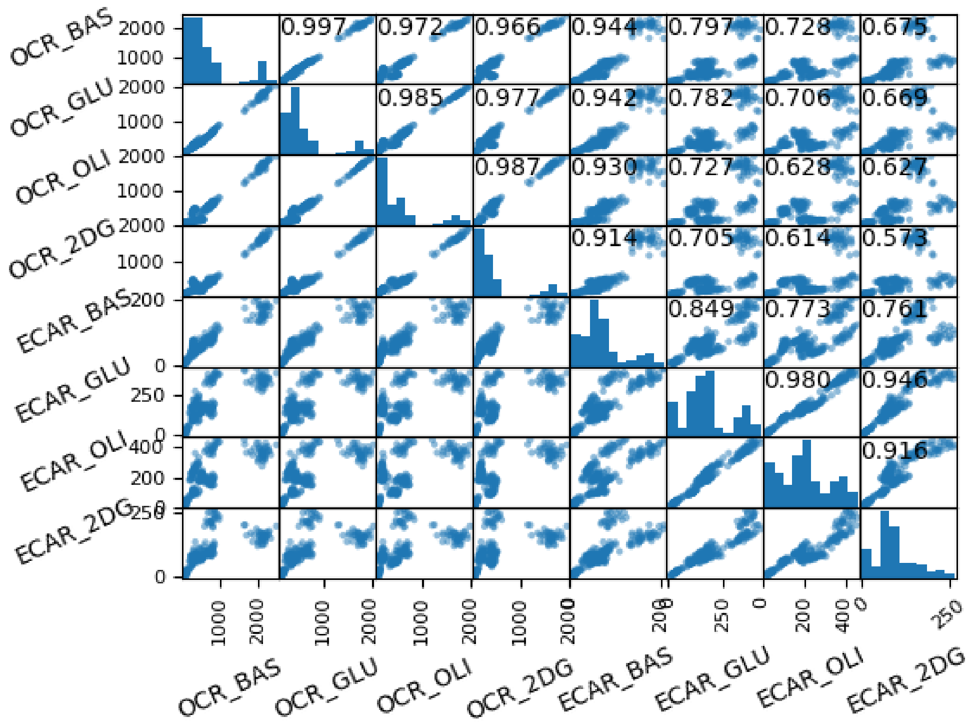


Figure 5.9: *Glycolysis Stress* - Scatter Plot and Correlation

- 1 boolean feature for the medium it used
- 1 numeric feature to define the concentration of the compound
- 4 OCR measures

- 4 ECAR measures
- 1 target value

Biological Analysis

To interpret the effects of the drugs some new graphs were generated. For the *Mito Stress* experiments the concentrations of 10μM and 50μM in a glucose medium were plotted, and can be seen in *Figure 5.10* and *Figure 5.11*. Only the measurements for the OCR are shown as they are the main focus of the *Mito Stress* test.

With the concentration of 10μM most compounds keep similar means to the Control's with the exception of ABA and ABE that have a higher oxygen consumption after OCR_FCCP. For the concentration of 50μM however the effects are exacerbated. The measures where it can be seen the most is at OCR_Basal and OCR_FCCP. Changes at the OCR_Basal level mean that the normal function of the cell is altered and changes after OCR_FCCP mean that the maximum respiratory capacity of the cell may be impaired. Only ABE keeps similar values to 10μM, and ABA measurements reduce considerably when compared to its effects in 10μM. APA, APE and Entacapone also induce some inhibition and Tolcapone shows a considerable deviation, both at OCR_Basal and OCR_FCCP.

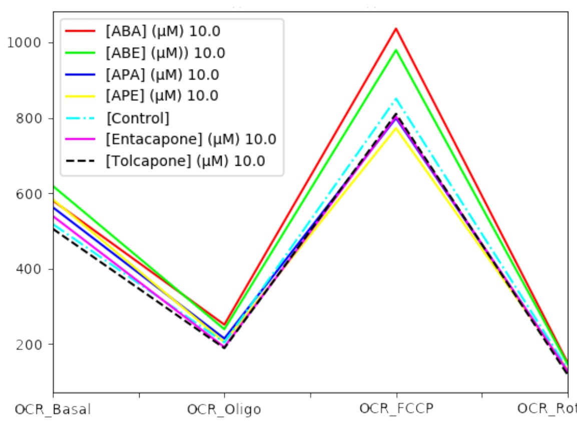


Figure 5.10: *Mito Stress* - 10μM Feature Averages by Drug in Glucose for

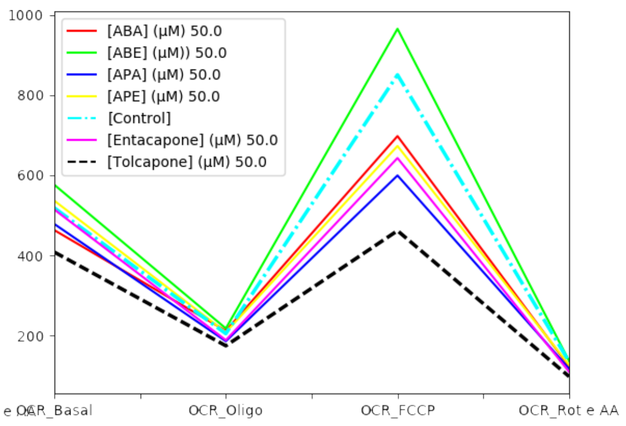


Figure 5.11: *Mito Stress* - 50μM Feature Averages by Drug in Glucose

Tolcapone is known to cause mitochondrial toxicity. Hence it would be expected that it would influence the most in the galactose medium as it forces OXPHOS. However, when the data were analyzed the cells in glucose were more affected than the ones in galactose, as can be seen in *Figure 5.12* and *Figure 5.13*. A possible, yet still theoretical, reason for these results is that instead of directly impairing the OXPHOS it may target something upstream of it, compromising the availability of the substrate.

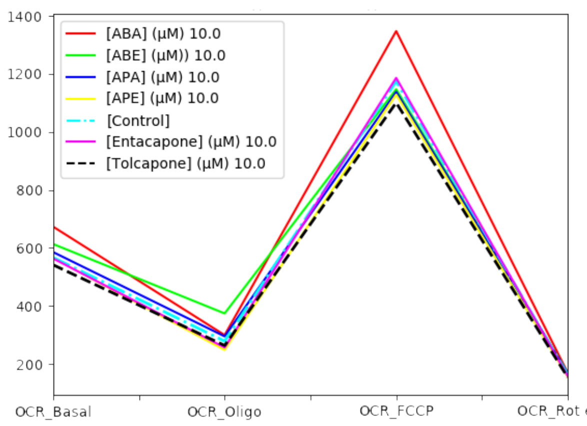


Figure 5.12: *Mito Stress* - 10μM Feature Averages by Drug in Galactose

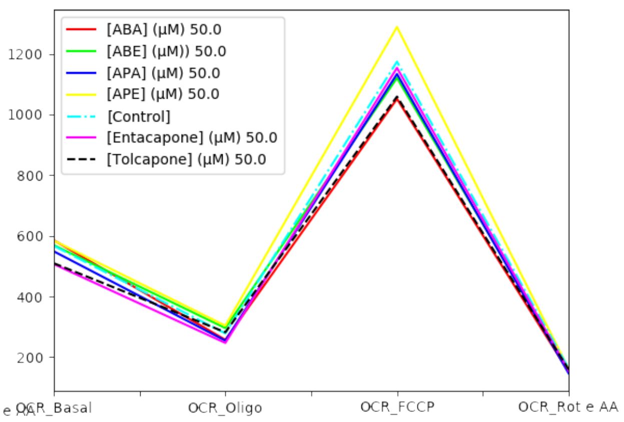


Figure 5.13: *Mito Stress* - 50μM Feature Averages by Drug in Galactose

For the *Glycolysis Stress* experiments a plot for the concentrations of 10μM and 50μM in a glucose medium were also generated, and can be seen in *Figure 5.14* and *Figure 5.15*. In both of them only the measurements for the ECAR were plotted as they are the main focus of the *Glycolysis Stress* test.

Both plots show that all compounds induce some dysfunction however the more visible are APA and APE. For the 50μM concentration the values do not change significantly, with ABA and ABE closer to the *Control*. The main difference is with the compound APA which presents a reduced value after the ECAR_Oligo.

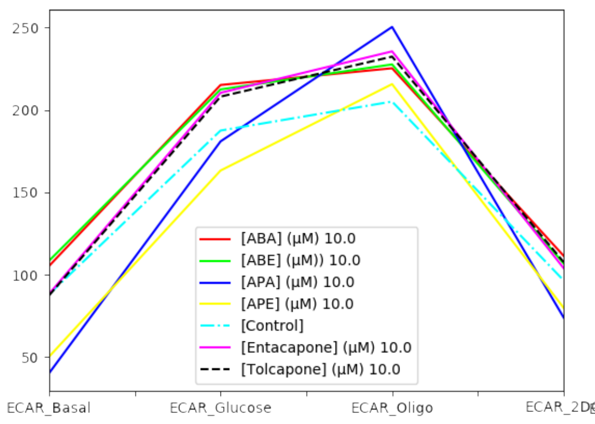


Figure 5.14: *Glycolysis Stress* - 10μM Feature Averages by Drug in Glucose

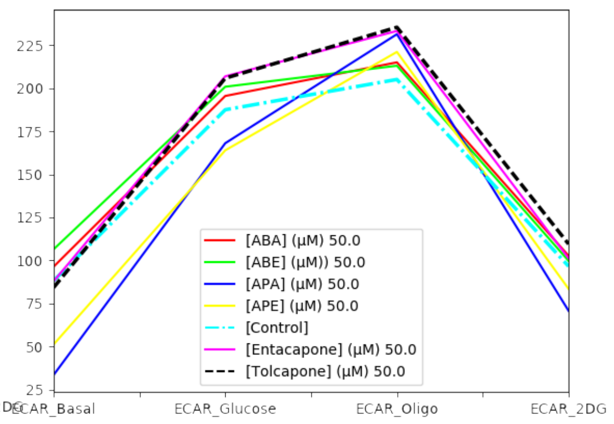


Figure 5.15: *Glycolysis Stress* - 50μM Feature Averages by Drug in Glucose

However, once again the results of the compounds in a galactose medium show steeper effects, as it can be seen in *Figure 5.16* and *Figure 5.17*. In 10μM concentrations most of the compounds have similar values to the Control, however ABA shows a significant reduction. In 50μM all the compounds show some deviation and the ABA reduces even further.

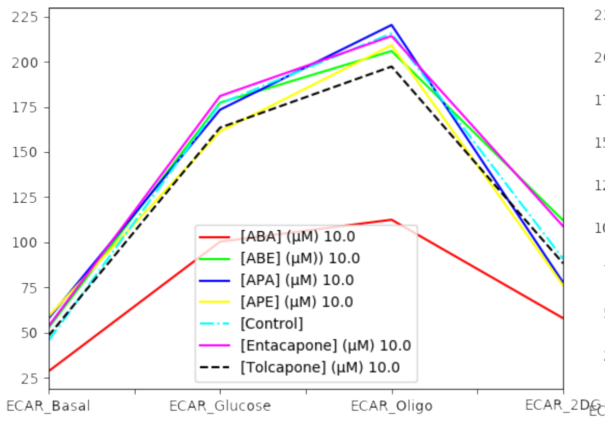


Figure 5.16: *Glycolysis Stress* - 10μM Feature Averages by Drug in Galactose

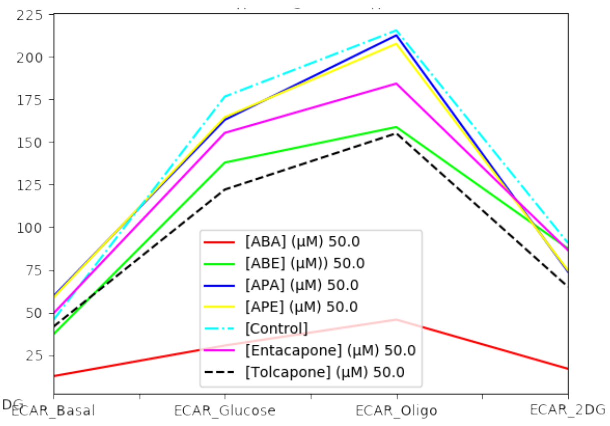


Figure 5.17: *Glycolysis Stress* - 50μM Feature Averages by Drug in Galactose

For the *Glycolysis Stress* test data a factor that must be taken into account is that the number of samples is very low and in some cases the records for a compound came from a single experiment.

5.1.3 Dimensionality Reduction

In order to improve the results obtained by the models and reduce its complexity a set of *feature selection* and *extraction* techniques were used. Although dimensionality reduction techniques are mainly used to reduce the complexity and time required to create the models they can also be used to remove irrelevant or noisy data, as well as improve the ratio between samples and features. Although the data that exist for this dataset are scarce these techniques were experimented nonetheless. The impact on computational performance generated by removing features was not taken into account due to the fact that all the datasets were small and thus were already swiftly processed. The methods applied were:

- **Correlation** - Based on the correlation values for the pairs of features, those higher than 80% were identified and one of the features of the pair was removed from the dataset. Using this method the features removed were:
  - **Mito Stress** - ECAR_BAS
  - **Glycolysis Stress** - OCR_BAS, OCR_GLU, OCR_OLI, OCR_2DG, ECAR_-BAS, ECAR_GLU, ECAR_OLI

  As it was previous analyzed most of the features in the *Glycolysis Stress* experiments had a high correlation, hence most of them were removed with this filter.

- **MI** - This method calculates the amount of information contained in each feature using the MI functions described in *Section 3.1.3*. Then the 20% that had the lowest relation with target variable were removed. Using this method the features removed were:
  - **Mito Stress** - Dosage, ECAR_FCCP
  - **Glycolysis Stress** - Dosage, OCR_GLU

  The *Dosage* feature is the one that contains the concentrations used for each sample.

- **Recursive Feature Elimination (RFE)** - Due to the effectiveness and the way Decision Tree (DT) algorithms uses features to develop its trees (Ratanamahatana and Gunopulos 2002) this algorithm was chosen to be used in this procedure. To choose the number of features to keep, a *cross-validation* approach was applied to each combination of features resulting from the RFE, and in the end the subset of features that produced better results were kept. The scoring function used for this method was the recall, to try to benefit those who could better predict the most elements of each class. Using this method the features removed were:
  - **Mito Stress** - Environment, OCR_BAS, OCR_FCCP, OCR_ROT, ECAR_OLI, ECAR_ROT
  - **Glycolysis Stress** - Environment, OCR_BAS, OCR_GLU, OCR_2DG, ECAR_BAS, ECAR_GLU, ECAR_OLI

  The *Test* and *Environment* features contain the type of test (either *Mito Stress* or *Glycolysis Stress*) and the media (either glucose or galactose).
- **Principal Component Analysis (PCA)** - Although the interpretability of the results is important in this project a PCA feature extraction was used. Due to the transformation from the initial features to the computed components the interpretability of the results is mostly lost as they do no longer represent direct variables from the problem. After the PCA was applied the principal components that represented 99% of the variance were kept. Using PCA the following components were used: **Mito Stress**: 9, and **Glycolysis Stress**: 5.

5.1.4 Data Balancing

To mitigate the imbalance in the dataset both undersampling and oversampling techniques were tested. The goal was to improve the results by avoiding the overfitting of the model to the overrepresented classes, therefore losing the ability to generalize to the remaining ones.

The methods used were *Random Undersampling*, *Random Oversampling* and SMOTE. For SMOTE the number of neighbors to be considered was kept as in its original article, 5 (Chawla et al. 2002).

5.2 Clustering

The first approach was to treat this work as a clustering problem. The intention was to verify if the algorithms would be able to create a decent separation between the drugs. The algorithms that will be analyzed further are *k-Means*, *DBSCAN*, *Hierarchical Clustering (HC)* and *CURE*. Some other algorithms such as *Birch* and *Gaussian Mixture* were also tested however they are neither algorithms of reference nor did their results merit further analysis. Nonetheless, they can be seen in *Appendix B*.

For the clustering problem the targets were removed from the data, using them only to calculate some external performance measures. All the algorithms were also used in combination with the data balancing and dimensionality reduction techniques described in *Section 5.1.3*.

5.2.1 k-Means

Due to the fact that *k-Means* requires the definition of the number of clusters, an optimization logic was implemented to find it by identifying the k which produced the highest silhouette score. The number of clusters tested laid within a range, between 2 and 14. Although the ideal number would be lower than the maximum tested, due to the nature of the clusters obtained by *k-Means* it would be possible that by having more clusters it would be able to create a better separation between the drugs (Tan, Steinbach, and Kumar 2005). Through this method the k value found was **2**.

5.2.2 Density-based spatial clustering of applications with noise (DBSCAN)

As described in *Section 3.4.2*, DBSCAN requires the definition of two parameters, the ϵ and $minPts$. In order to select the best values a basic approach was implemented. It analyzes the distance from a point to its k^{th} nearest neighbor, referred as k-dist. Points that belong to the same cluster will have a low k-dist, and the remaining will have a high value. The computation of the k-dist with $k = 4$, a number already proven to be effective (Ester et al. 1996), for all the data points was made. Afterwards it was sorted in ascending order and the values were plotted, expecting to see a sharp change at the value of k-dist that corresponds to a suitable value of ϵ. Using the k value as $minPts$ the points for which k-dist is less than ϵ will be labeled as core points, while the others will be labeled as noise or border points (Tan, Steinbach, and Kumar 2005). Also, the metric used to calculate the distance between the instances was the **euclidean**. As it can be seen in *Figure 5.18* there is a fairly sharp change when the value of the distance is approximately 0.92. Therefore, the chosen values for DBSCAN were $\epsilon = 0.92$ and $minPts = 4$.

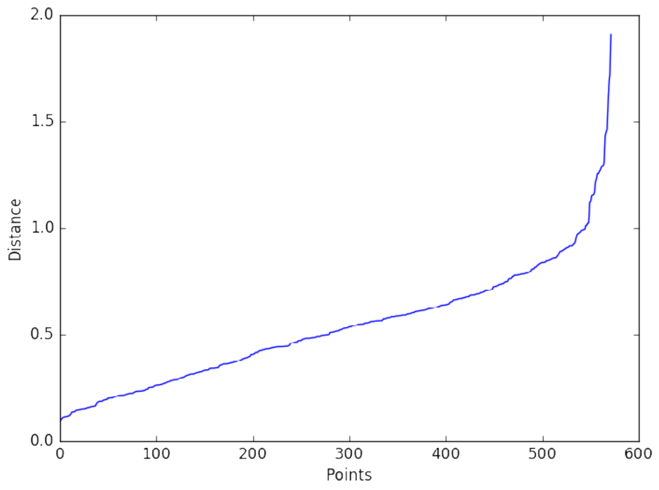


Figure 5.18: k-dist plot for $k = 4$

5.2.3 Hierarchical Clustering (HC)

The type of HC used was the **agglomerative**, also known as *bottom-up*, in which the individual points are successively merged into higher-level clusters (Aggarwal 2015). The choice of which linkage method and metric to use was made by testing the common combinations:

average, complete and ward linkages, with cityblock, euclidean and chebchevy distance metrics. Then the pair that possessed the higher cophnetic correlations was chosen. Briefly, this correlates the real pairwise distances of all the instances to those implied by the hierarchical clustering (Saraçli, N. Doğan, and İ. Doğan 2013). In this case the chosen linkage was **average** with the **euclidean** distance metric. To chose the number of clusters to be used the silhouette coefficient was also studied with the same technique implemented for *k*-Means. This also defined that the number of clusters that produced a higher silhouette score was **2**.

5.2.4 Clustering Using REpresentatives (CURE)

To use this algorithm it is necessary to define some parameters, which were mainly chosen based on the existing literature (Guha, Rastogi, and Shim 1998):

- *k* **clusters** - to define the number of clusters an identical approach to the one used for *k*-Means was used, based on the silhouette score, which defined it as **2**
- **Representative points** - the number of representative points was set to **10**, a value shown to be effective
- **Compression** - for the shrink factor the value of **0.5** was used, halfway between its defined range

5.3 Classification

Some algorithms were tested for the classification approach to this problem. The list of algorithms used: *Naive Bayes*, *Gaussian Process*, *Logistic Regression*, *Decision Tree (DT)*, *k Nearest Neighbors (k-NN)*, *Support Vector Machine (SVM)*, *Neural Network (NN)* and *Genetic Programming (GP)*. The ensemble methods *Bagging* and *Random Forest (RF)* were also tested. Additionally an ensemble comprising some of these methods was implemented. Some of the parameters of the various algorithms were chosen based on existing literature.

5.3.1 Naive Bayes, Gaussian Process and Logistic Regression

The *Naive Bayes* used is a **Gaussian Naive Bayes**, where the likelihood of the features is assumed to Gaussian.

For the *Gaussian Process* a **RBF** kernel was chosen. To optimize the kernel parameters the **Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS)-B** optimizer is used. As this is a multi-class problem the One-Versus-Rest (OVR) schema was used, where each class has a classifier and the one that has a higher value is the chosen class.

As for the *Logistic Regression* the only configuration made was to use the **L2 regularization**. To make a prediction it also uses an OVR approach.

5.3.2 k Nearest Neighbors (k-NN)

For the k Nearest Neighbors (k-NN) implementation used a k of **5** was chosen due to the number of experiments for each configuration in the data. The metric used to calculate the distance between the neighbors was the **minkowski**.

5.3.3 Support Vector Machine (SVM)

The SVM method used was configured with a **Radial Basis Function (RBF)** kernel. The gamma value for the kernel was set with a relative number: $\frac{1}{n\_features}$. Also it uses One-Versus-One (OVO) to develop its classifiers. That means that involves creating a model for each pair of classes, which results in $\frac{N(N-1)}{2}$ classifiers. When testing, a majority voting of the votes of each classifier will define the predicted class (Gidudu, Hulley, and Marwala 2007).

5.3.4 Neural Network (NN)

For the NN algorithm a couple of parameters must be defined. The NN was created with **1 hidden layer**, thus making it a two-layer NN which is proved to be very efficient and are considered an universal function approximators (III 2012). The activation function used is the **rectified linear unit** function $f(x) = max(0, x)$. The solver used to optimize the weights is the **LBFGS** and the regularization parameter was set as 0.0001. A constant learning rate of 0.001 was used.

5.3.5 Decision Tree (DT), Bagging and Random Forest (RF)

To create the DT models the **Gini Impurity** was used to measure the quality of the split at each node, and the best one was always chosen. The algorithm used was the **Classification and Regression Trees (CART)** which is similar to C4.5, however instead of the entropy based approach it uses the *Gini index*, a generalization of the binomial variance (Loh 2011).

As for Bagging, **10 DT** were used as base learners. The DT algorithm was chosen due to their general ability to create good classifiers across datasets.

The RF was configured with a maximum number of **10 estimators** and also the **Gini Impurity** for the measure of the node quality.

5.3.6 Expert Ensemble

To overcome the limitation of each individual model and to exploit its capabilities an ensemble method was implemented. The first implementation was a **Majority Voting Ensemble**.

Another approach to combine the decisions of the individual classifiers was developed on the concept that some classifiers perform better than others, inspired by the work in (Nagi and Bhattacharyya 2013). The approach they propose combines n classifiers into an ensemble based on an estimation of their performance. The main idea is that it can give different weights to classifiers based on their ability to predict a part of the problem better than the remaining. The method they suggest has an architecture of a two-layer model, as illustrated in *Figure 5.19*.

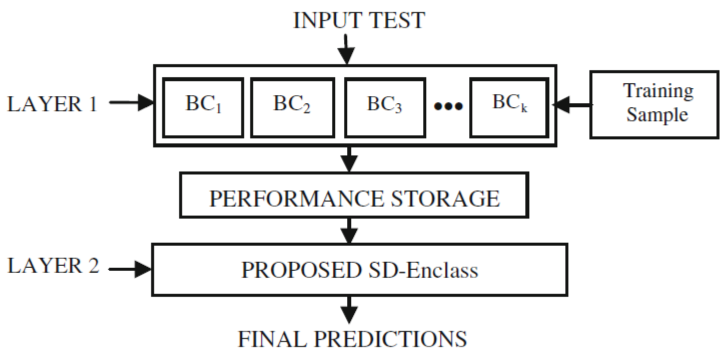


Figure 5.19: Expert Ensemble Architecture, from (Nagi and Bhattacharyya 2013)

Their approach creates a specific training dataset for measuring the performance of the algorithm, by removing redundant information. It then trains the classifier with a 10-fold *cross-validation*. The classifiers are then tested, and the one with the highest performance (*precision*) for a given class becomes the expert of such class. When making a prediction it has several rules to decide which class is returned if any of the classifiers is an expert in the class it predicted, and in ties looks at the probabilities obtained during training.

Thus, the second ensemble implemented for this project follows some of those ideas. It is also a two-layer system as in *Figure 5.19*. In the first layer the models are trained with the entire training set and its performance is also measured using 10-fold *cross-validation*. However, instead of using the metric as in the original article (*precision*) the *recall* is used. As the intention is to create models that would be able to classify as most of a given class as possible and *recall* attempts to measure just that. The main difference from its inspiration is on the voting heuristic. In the original article the experts had a definite decision, that is, if there was an expert in the predicted classes, that would be the choice. We on the other hand thought that a weighted expertise and the algorithms confidence in each class would create better decision borders. All the classifiers contribute to the class weights with their confidence from its probabilities and its expertise for that class. The reasoning behind it is that by only using the final decision of the algorithm we are ignoring the confidence (or lack of it) that the classifier has in its decision. As an example, if the probabilities are: 34% for class A, 33% for class B and C, then the class A would be chosen even if the classifier almost had the same probabilities for each class. In the end the class that has the highest weight is chosen. It implements the following formula:

$$y = argmax_{j \in \{1,2,\ldots C\}} \left[\sum_{i=1}^{T} e_{t,j} * w_e * p_{t,j} * w_p + \sum_{t=1}^{T} d_{t,j} * w_s \right] \tag{5.1}$$

where:

- $e_{t,j}$ is the expertise of classifier t for class j, which if j is not its predicted class will take the value of 1 (only contributing with its probability)
- $p_{t,j}$ is the probability returned by the classifier t for the class j

- w_e, w_p and w_s are weights for the expertise, class prediction probability and support respectively

As it can be seen by the formula it is a voting process inspired in both weighted and majority voting.

5.3.7 Genetic Programming (GP)

To study how an evolutionary approach would fare in such a problem, a Genetic Programming (GP) algorithm was developed. The implementation made treats this as a *symbolic regression* problem, which is a process of mechanically creating a computer program that fits certain numerical data (Poli, Langdon, and McPhee 2008).

In this case the purpose was to evolve a program that would be able to distinguish between the classes. The implementation made had the following configurations:

- **Representation** - Syntax trees were chosen for the representation
- **Initialization** - *Ramped Half-and-Half* method was used to initialize the population
- **Population Size** - The population size chosen was 500, as it is considered a reasonable minimum for a GP approach (Poli, Langdon, and McPhee 2008)
- **Mutation Probability** - The mutation probability used was of 10%, as it is an accepted and tested good value (Poli, Langdon, and McPhee 2008)
- **Mutation Operator** - The operator used for mutation was *point mutation*
- **Crossover Probability** - The crossover probability used was of 90%, as suggested in (Poli, Langdon, and McPhee 2008)
- **Crossover Operator** - The crossover operator chosen was *subtree crossover*.

  Typical GP primitive sets lead to trees with an average branching factor, that is, the number of children of each node, of at least two. Hence, the majority of nodes will be leafs, which if the choice of the crossover point is made by uniform probability leads to frequently exchanging only very small amounts of genetic materials. To prevent this, 90% of the times a function node is selected and leaves only 10%, as suggested in (Koza 1992)
- **Maximum Tree Size** - In order to prevent bloat and excessive growth of the problems a maximum tree size of 1000 nodes was set
- **Generations** - The number of generations chosen was 100, due to the fact that usually the most productive search is performed in the initial generations, and if it is not found then, it is not likely to be found in a reasonable amount of time (Poli, Langdon, and McPhee 2008)
- **Terminal Set** - The terminal set was comprised of a number of variables that matches the number of features of the dataset, and an *ephemeral random constant* uniformly chosen from the range [-5, 5], as suggested in (Poli, Langdon, and McPhee 2008)
- **Function Set** - The function set used included the basic arithmetic functions: $+, *, /$ and exponents of 2 and 3

- **Parents Selection** - The selection of parents to reproduce is made through *tournament selection* of 3 individuals
- **Offspring Selection** - The offspring that will constitute the next generation is made through *generational selection* and 2% of *elitism*
- **Fitness Function**
  Since *symbolic regression* outputs a continuous value it is necessary to discretize it to convert it into a classification model. A discretization method that had promising results was using the *Minimum Description Length Principle (MDLP)* (Fayyad and Irani 1993). In short, it creates multi-interval bins using a entropy minimization heuristic. After the discretization, a majority voting of the samples that were in the ranges of each bin was applied to define the target class for each bin. To evaluate the fitness of the solution the real classes of the samples were compared against the predictions, and the performance metric *recall* was used.

  The second implementation of GP attempted to develop a program that would be able to predict a toxicity index. To do so a different fitness function was developed. The main idea was to create a sequence of contiguous bins that would be able to separate the different drugs. For each individual (program) a bin was created for each class, where the minimum and maximum of each bin was based on the lowest and highest value for the samples of that class, as illustrated in *Figure 5.20*. However this would lead to overlapping bins, thus to avoid it the intersection point of the sequential bins was calculated and the boundaries redefined.

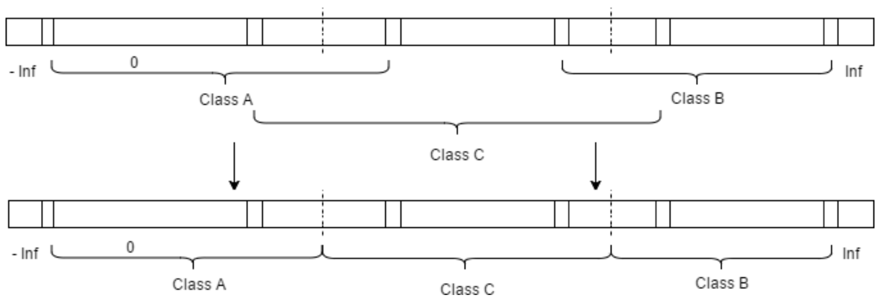


Figure 5.20: GP Bins Definition

  Afterwards, similar to the other approach the samples target would be compared to the predictions and the *recall* was used to measure the performance of the model. To guide the evolution of the programs individuals in which the minimum of the lower bin was lower than 0 and those that predicted very high indexes (above 10^8) were penalized. Also, solutions to which the Control bin was not the one with the lowest indexes and those that did not have the Tolcapone as the most toxic were also penalized.

5.4 Training and Testing

As the dataset for this problem is not very large the recommended approach to evaluate the performance of the algorithm is *cross-validation* (Alpaydin 2014). *Stratification* was also used to keep the representation of the classes from the original dataset in the resulting folds. The number of folds used was **10**, which was based on previous studies where it was proved that it produces good results across multiple models (Borra and Di Ciaccio 2010). Also, each method used for classification was ran **30** times with different seeds for the random generators, to allow the reproducibility of the experiments and to minimize the possible bias from random values.

Chapter 6

Results

To analyze the performance of the different algorithms with the data from the Mito Stress and *Glycolysis Stress* tests some combinations of the data were used. As both tests measure the same type and amount of information (OCR and ECAR) an extra dataset was created that combines them. Hence, three separate datasets were studied: one exclusively with the **Mito Stress** test data, another only with the **Glycolysis Stress** test data and a third that combines both **Mito Stress and Glycolysis Stress** test data. For this last dataset an extra feature was added to identify to which test each sample belongs to. This dataset will be referred to from now on as the **complete** dataset.

6.1 Clustering

For the clustering study the target of each sample, that is, the drug it was exposed to, was removed from the dataset that would be used to create the models. They were however used to calculate some external performance measures and to analyze which samples were in each cluster. The algorithms used were *k-Means*, *DBSCAN*, *Hierarchical Clustering (HC)* and *Clustering Using REpresentatives (CURE)*.

As part of the goal of this project was to study the new drugs toxic effects, and how they compare to Entacapone and Tolcapone, the ideal clustering would separate them from the remaining, indicating that their effects were indeed different. Also, if the generated clusters grouped the new drugs with the Control it could be interpreted as that they did not have enough mitochondrial effects to distinguish between them. Some of the remaining results can be seen in *Appendix B*.

6.1.1 *k*-Means

The silhouette coefficient was used to select the number of clusters to be used. The coefficient for each number of clusters for the *Complete* can be seen *Table 6.1*. As it can be seen, $k = 2$ generates the clusters with the highest score. In order to analyze the composition of each cluster the class distribution of the samples in each one was plotted. In *Figure 6.1* it is possible to see that most of the classes fell under a single cluster. We also analyzed the class distribution for 7 clusters, one for each condition, as illustrated in *Figure 6.2*. Still, the samples from the different classes are scattered across the clusters, with no visible separation amongst them.

# Clusters	Silhouette
2	0,612
3	0,380
4	0,362
5	0,416
6	0,480
7	0,454
8	0,429
9	0,446
10	0,438
11	0,451
12	0,459
13	0,419

Table 6.1: *k*-Means *Complete* Silhouette Coefficients

# Clusters	Silhouette
2	0,610
3	0,554
4	0,486
5	0,484
6	0,527
7	0,485
8	0,459
9	0,478
10	0,476
11	0,506
12	0,541
13	0,527

Table 6.2: *k*-Means *Glycolysis Stress* Silhouette Coefficients

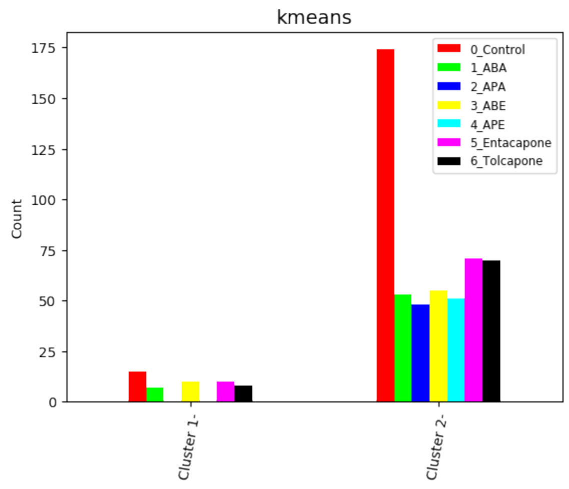


Figure 6.1: *k*-Means *Complete* Class Distribution - 2 Clusters

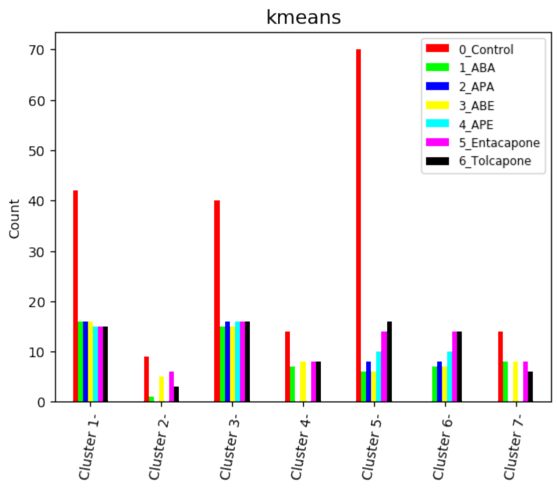


Figure 6.2: *k*-Means *Complete* Class Distribution - 7 Clusters

Although *k*-Means is a very known and commonly used algorithm it is also a simple and therefore has some limitations. One of them is that it assumes that the clusters have a spherical structure and hence does not perform well with clusters of arbitrary shape (Aggarwal 2015). A further analysis was made to the data from each test separately. The *Mito Stress* test data was not able to perform any better, however with *Glycolysis Stress* there were results that merited attention. The number of clusters chosen based on the silhouette coefficient was the same as the *Complete*, as it can be seen in *Table 6.2*. The distribution of the cluster formed was also similar to the *Complete*, as it can be seen in *Figure 6.3*. Nonetheless, after a thorough analysis of the clusters that formed with other cardinality a pattern appeared. As it can be seen in *Figure 6.4*, it is visible that two pairs of drugs are regularly grouped together, with minimum overlapping between them. The samples of pairs of drugs ABA & ABE and APA & APE are found in the same clusters, and only 4 instances of APE show up in clusters mainly of ABA & ABE. There is not however any particular association between the pairs and the controls, Entacapone or Tolcapone. Additionally, by analyzing these results it suggest that the clustering of ABA and ABE with the *Control* is more common than with APA and APE.

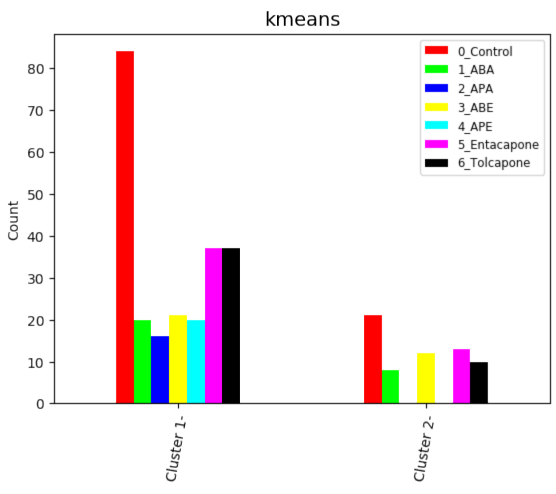


Figure 6.3: *k*-Means *Glycolysis Stress* Class Distribution - 2 Clusters

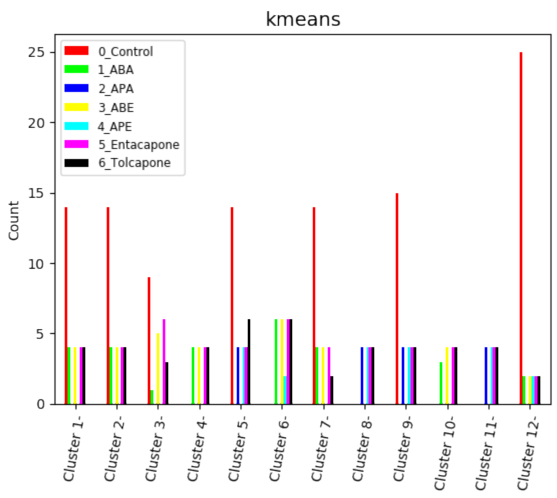


Figure 6.4: *k*-Means *Glycolysis Stress* Class Distribution - 12 Clusters

Both the techniques of data balancing and dimensionality reduction detailed in *Section 5.1.4* and *Section 5.1.3* respectively were applied, yet the results were not altered.

On a later stage, after the classification algorithms were studied, another attempt was made with new datasets that comprised the differentials between the measures, as will be further describe in *Section 6.2.3*. For the *Glycolysis Stress* dataset some improvements were seen. The number of clusters that had a higher silhouette changed to 6. As it can be seen in *Figure 6.5* these clusters were able to make a good separation between the pairs of drugs previously identified, with the exception of the same number of APE's that appear in a cluster with ABA & ABE. Also, after a thorough analysis it was found that with 3 clusters the results would be as *Figure 6.6*, which creates fewer and bigger clusters by keeping the same mixture of 4 APE's.

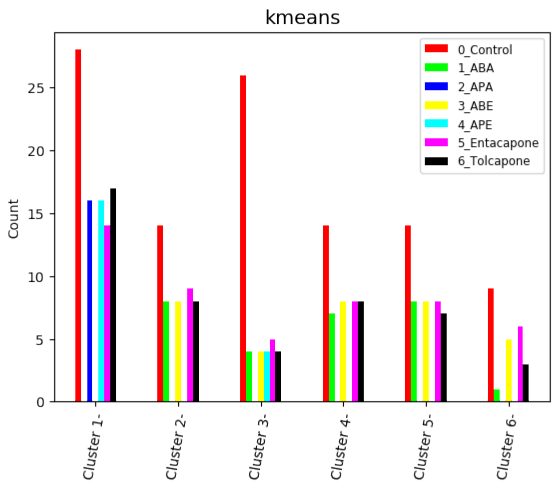


Figure 6.5: *k*-Means Differential - *Glycolysis Stress* Class Distribution - 6 Clusters

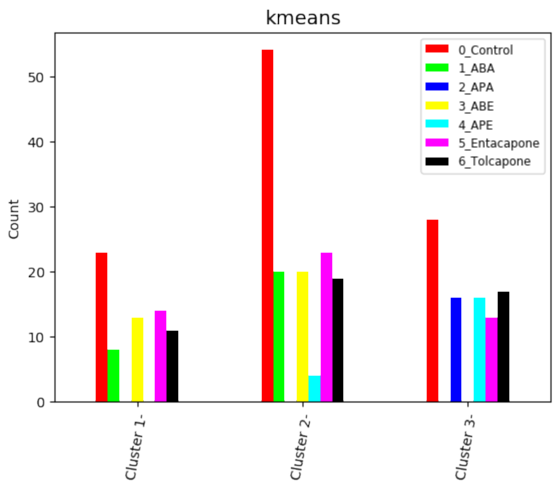


Figure 6.6: *k*-Means Differential - *Glycolysis Stress* Class Distribution - 3 Clusters

6.1.2 Density-based spatial clustering of applications with noise (DBSCAN)

DBSCAN is an algorithm that has been shown to perform well on several datasets with clusters of varying shape and size. The $minPoints$ was set at 4 based on the literature and then the ϵ was set based on the distances of the $4th$ neighbors. With these parameters the algorithm found 17 clusters, as can be seen in *Figure 6.7*. Once more, the clusters are not able to separate the effects between the compounds into distinct clusters. Analyzing the datasets individually allowed the algorithm to be able to separate the drugs ABA & ABE and APA & APE with the *Glycolysis Stress* as in *Figure 6.8*.

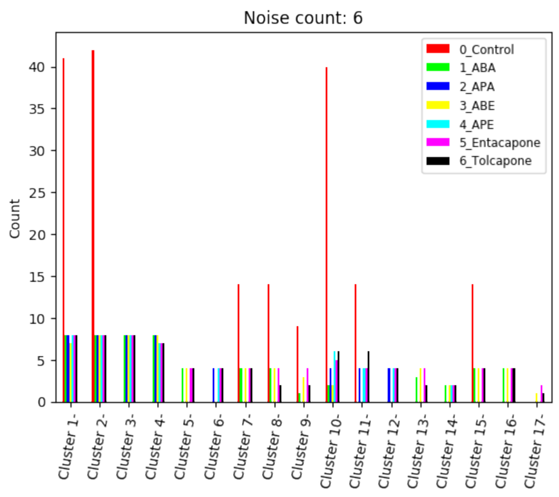


Figure 6.7: DBSCAN *Complete* Class Distribution - 17 Clusters

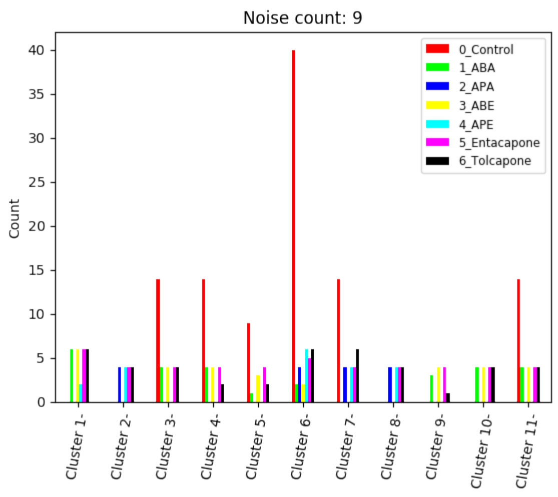


Figure 6.8: DBSCAN *Glycolysis Stress* Class Distribution - 11 Clusters

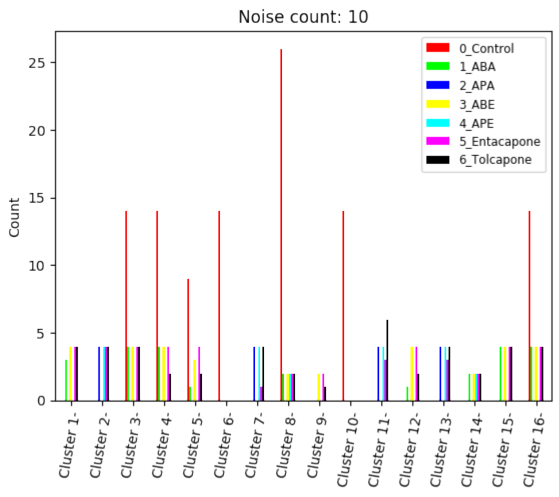


Figure 6.9: DBSCAN Differential - *Glycolysis Stress* Class Distribution - 16 Clusters

Once again on a later stage tests were performed by using the differentials dataset and the results improved slightly, providing some more information. In *Figure 6.9* it is possible to observe that the clusters separate both pairs of drugs and the algorithm also managed to create clusters that completely separate Control samples. Again, data balancing and dimensionality reduction techniques did not produce any improvements.

DBSCAN is known to also have some limitations. One of them, and certainly one of the reasons it was not able to perform better is related to the way it uses densities to define the clusters, which make it difficult to deal with clusters of varying density.

6.1.3 Hierarchical Clustering (HC)

One of the main reasons why HC it is an interesting algorithm is that it provides different levels of clustering granularities which in turn give different application specific insights (Aggarwal 2015).

Figure 6.10 shows a dendogram of the clusters generated by HC and their distances that result from the configurations made. The silhouette score was also used to determine the number of clusters, which also resulted in 2. The coefficients for each number of clusters can be seen in *Table 6.3* and the distribution of the samples classes for each cluster can be seen in *Figure 6.11*.

# Clusters	Silhouette
2	0,625
3	0,426
4	0,379
5	0,451
6	0,430
7	0,460
8	0,457
9	0,439
10	0,421
11	0,446
12	0,404
13	0,385

Table 6.3: HC *Complete* Silhouette Coefficients

# Clusters	Silhouette
2	0,371
3	0,387
4	0,411
5	0,444
6	0,475
7	0,469
8	0,446
9	0,428
10	0,409
11	0,419
12	0,435
13	0,424

Table 6.4: HC Glycolysis Differentials Dataset Silhouette Coefficients

Once again the results obtained are not able to acceptably separate the drugs by clusters. Hierarchical methods are usually sensitive to small mistakes done during the merging of the clusters. Some of the linkage methods are particular known for successively merging neighboring clusters due to the presence of a small amount of noise (Aggarwal 2015).

By analyzing the *Mito Stress* and *Glycolysis Stress* test data separately no improvements were found. However, once again on a later, stage using the differentials for the *Glycolysis Stress* dataset there were some improvements. These results however are not an improvement on what was already found with *k*-Means and DBSCAN. The best silhouette score for this dataset was for 6 clusters, as it can be seen in *Table 6.4*. With this number HC was also able to create clusters that would separate the ABA & ABE and APA & APE with the same 4 APE's mixed with ABA & ABE *Figure 6.12*.

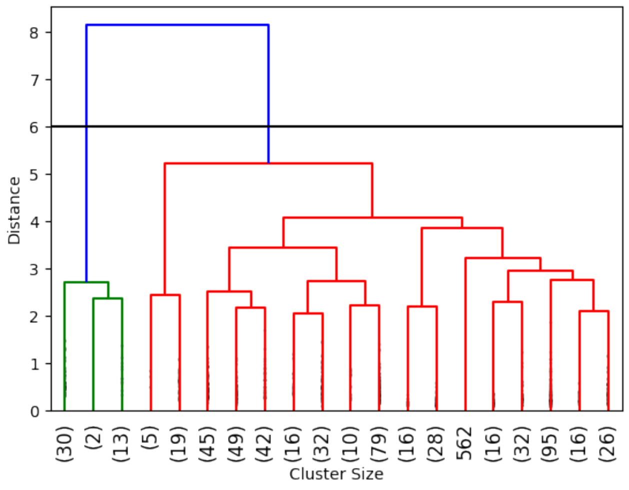


Figure 6.10: HC Dendogram with 2 Clusters Cut

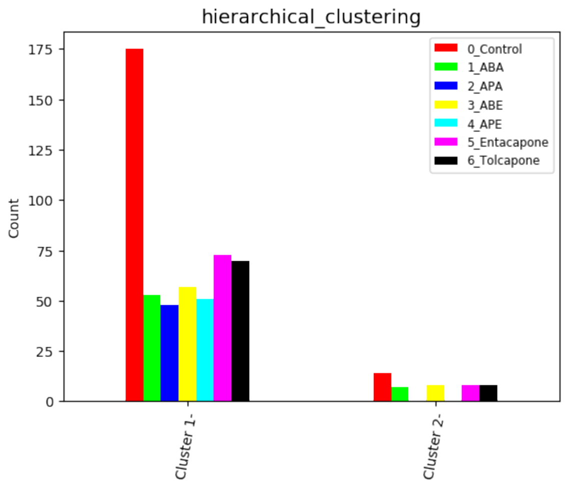


Figure 6.11: HC *Complete* Class Distribution - 2 Clusters

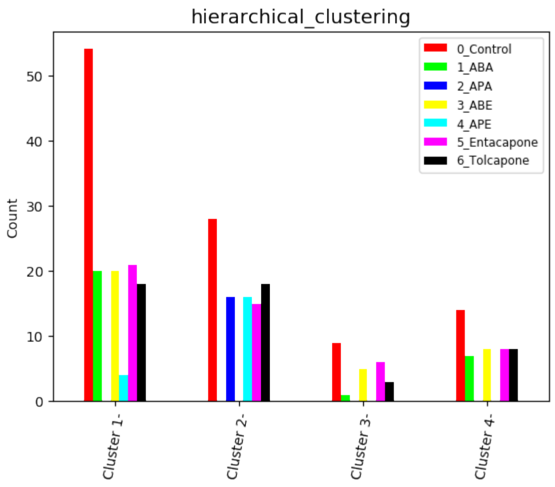


Figure 6.12: HC Differential - *Glycolysis Stress* Class Distribution - 4 Clusters

Again, data balancing techniques and dimensionality reduction did not improve the results.

6.1.4 Clustering Using REpresentatives (CURE)

The last clustering algorithm used was CURE. One of the reasons for the choice of this algorithm was that it is able to identify clusters of varying shape and size regardless of their density (Aggarwal 2015). The silhouette score was also used once again and for the combined dataset, the number of clusters that produced the highest score was 2, as listed in *Table 6.5* and illustrated in *Figure 6.13*. With this number of clusters, like the previous algorithms, most of the samples were mixed in the same cluster. Similar to the previous algorithms analysis were made with the individual tests data, data balancing and dimensionality reduction techniques that yielded no improvements. However using the differentials for the *Glycolysis Stress* dataset similar improvements to the ones observed in the remaining algorithms were found. The highest silhouette score for this dataset was for 6 clusters, as can be observed in *Table 6.6* and illustrated in *Figure 6.14*.

# Clusters	Silhouette
2	0,625
3	0,425
4	0,424
5	0,451
6	0,481
7	0,440
8	0,427
9	0,404
10	0,374
11	0,413
12	0,387
13	0,382

Table 6.5: CURE *Complete* Silhouette Coefficients

# Clusters	Silhouette
2	0,395
3	0,436
4	0,376
5	0,510
6	0,513
7	0,517
8	0,456
9	0,425
10	0,399
11	0,436
12	0,442
13	0,431

Table 6.6: CURE Glycolysis Differentials Dataset Silhouette Coefficients

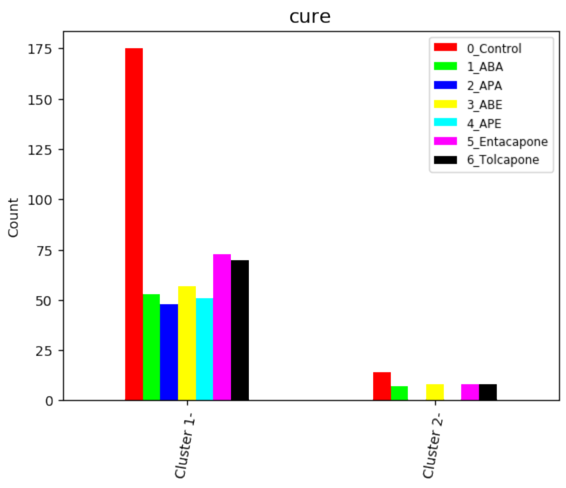


Figure 6.13: CURE *Complete* Class Distribution - 2 Clusters

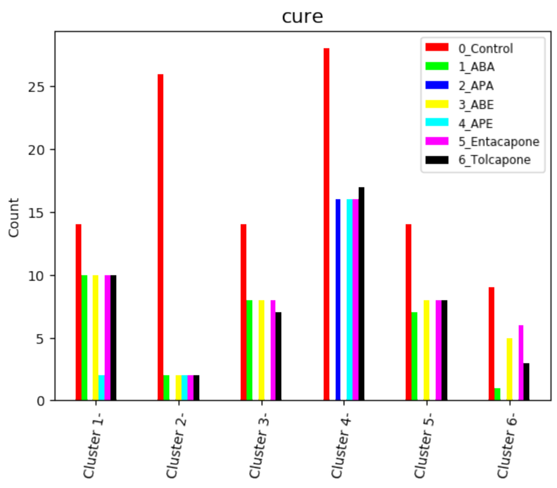


Figure 6.14: CURE Differential - *Glycolysis Stress* Class Distribution - 6 Clusters

6.2 Classification

By treating this as a classification problem the purpose becomes to create models that are able to classify the samples based on the compounds they have been exposed to. As with clustering, the same three datasets were used.

6.2.1 Initial Configuration

Because the toxicity of the new drugs is not yet thoroughly studied a problem arises when trying to determine which classes to predict for each sample. The tests that were previously made on the new drugs by the expert team indicated that they had somewhat similar therapeutic effects and are most likely less nefarious than Tolcapone (Sardão and Tiago B. Silva 2016). Hence the samples exposed to the new drugs were grouped as a single class. Thus the list of 4 classes that would used for the samples, based on the compound they were exposed to, were: **Control**; **ABA, ABE, APA and APE**; **Entacapone**; **Tolcapone**.

The classes distribution was as illustrated in *Figure 6.16*, *Figure 6.17* and *Figure 6.15*.

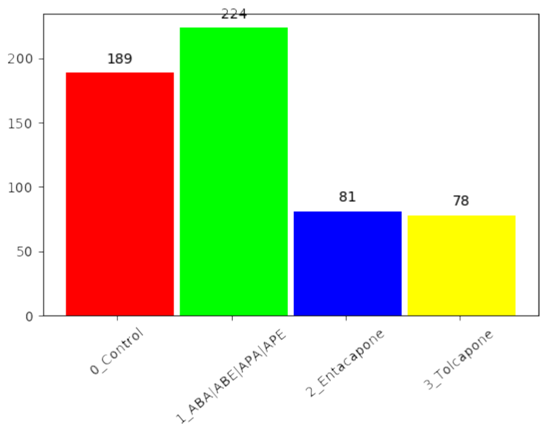


Figure 6.15: *Complete* Class Distribution

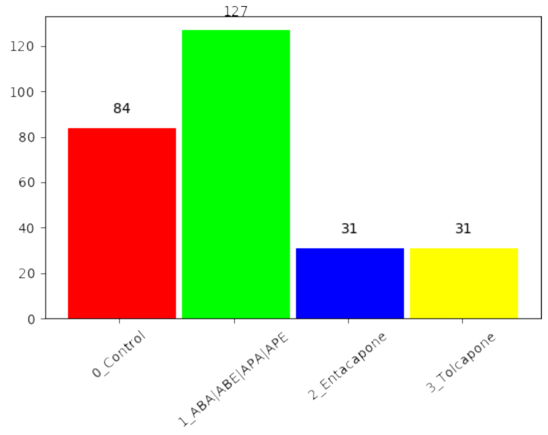


Figure 6.16: *Mito Stress* Class Distribution

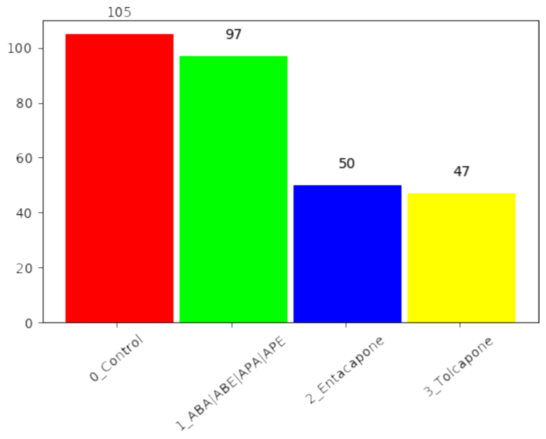


Figure 6.17: *Glycolysis Stress* Class Distribution

Afterwards, the classification algorithms discussed in *Section 5.3* were ran on all three datasets. One of the main tools used to analyze the performance of the algorithms in classifying the samples from the different classes were confusion matrices. Due to the fact that in this problem each drug may have an associated toxicity it is of utmost importance to understand which classes are being confused with one another as it may indicate similar effects.

The algorithms RF and NN were the ones that had better and consistent results across the datasets, hence they will be used to illustrate the development of the models. The remaining algorithms results can be seen in *Appendices Section B.2.1*. The results of the models created with these algorithms for the *Complete* dataset can be seen in *Figure 6.18* and *Figure 6.19*. They can clearly distinguish the controls from the samples exposed to compounds, correctly classifying 93% of the Control samples with RF and 94% with NN, with a minimal number of other samples being confused as Control. This is a very good indicator that despite all the variance present in the data the algorithms are still able to identify the impaired cells. They are not however able to acceptably distinguish between the drugs.

RF has a high percentage, 80%, of correct classifications of the class ABA|ABE|APA|APE however almost all the remaining samples are also classified as it, with 55% and 45% for Entacapone and Tolcapone respectively. NN was the only algorithm that was somewhat able to distinguish between the classes, with 71%, 36% and 45% for ABA|ABE|APA|APE, Entacapone and Tolcapone respectively.

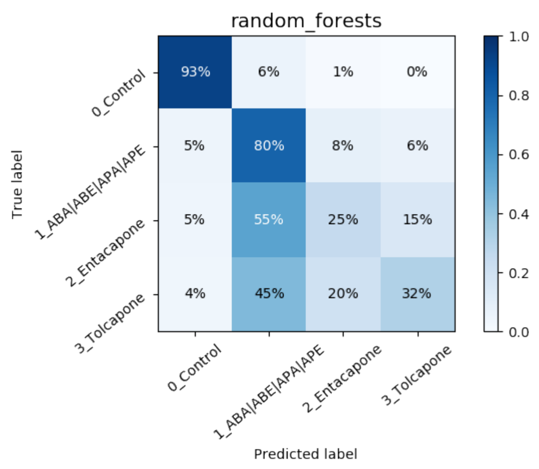


Figure 6.18: *Complete* RF CM

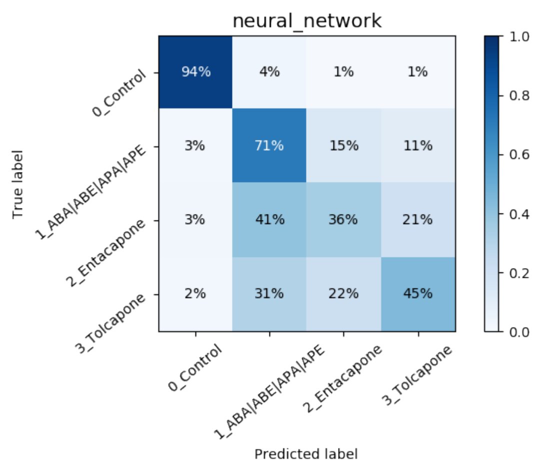


Figure 6.19: *Complete* NN CM

By using only the data from the *Mito Stress*, as illustrated in *Figure 6.20* and *Figure 6.21*, the results were similar. However they lost some of the ability to classify the Control, with 82% and 88% for RF and NN. Still the NN algorithm was slightly better at distinguishing amongst the classes with 74%, 39%, 52% for ABA|ABE|APA|APE, Entacapone and Tolcapone respectively.

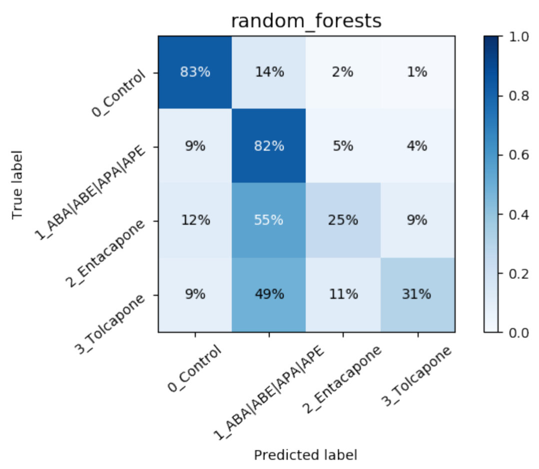


Figure 6.20: *Mito Stress* RF CM

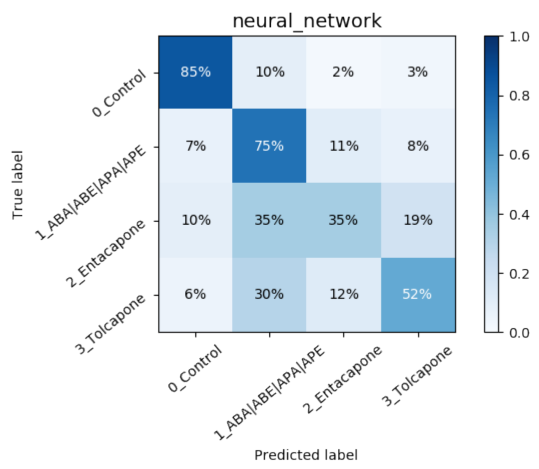


Figure 6.21: *Mito Stress* NN CM

With the data from the *Glycolysis Stress* test alone an interesting result appears, as can be seen in *Figure 6.22* and *Figure 6.23*. With them some of the algorithms, such as RF and NN, are able to completely separate the control samples from the remaining. Although their ability to classify the other samples did not improve, the fact they can entirely distinguish the controls is of particular interest. It is so because ultimately the main purpose is to identify if the compounds create mitochondrial dysfunction regardless of its intensity. Hence, with such classifier if the prediction given by it is "Control" it means that the drug has no detectable effects on the mitochondrial function.

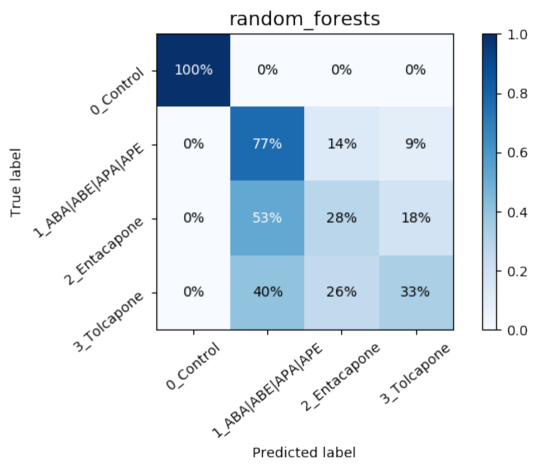


Figure 6.22: *Glycolysis Stress* RF CM

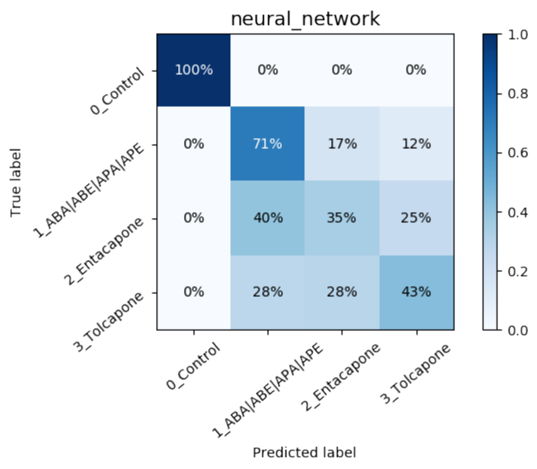


Figure 6.23: *Glycolysis Stress* NN CM

Data Balancing

Random Undersampling

When compared with the initial results of the *Complete* data it can be seen in *Figure 6.24* and *Figure 6.25* that both algorithms slightly improved in classifying the samples exposed to drugs. The RF model is able to correctly classify 43% for both Entacapone and Tolcapone against the initial 25% and 32%. However the classification of ABA|ABE|APA|APE is considerably reduced from 80% to 55%. A similar analysis can be made for the NN model. The predictions of Entacapone and Tolcapone went from 36% and 45% to 44% and 51% respectively and the ABA|ABE|APA|APE decreased from 71% to 50%. Also the ability to identify the control samples was lower in both algorithms.

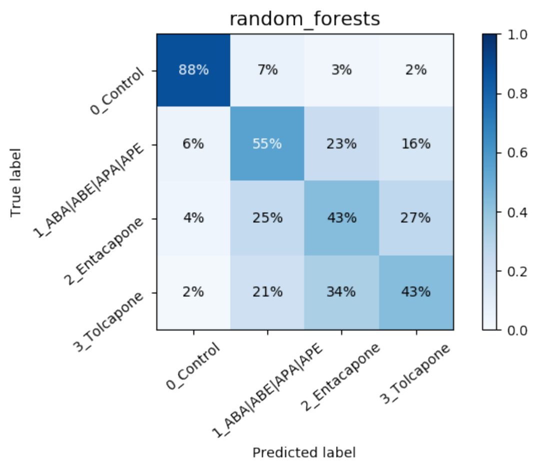


Figure 6.24: *Complete* Undersampling RF CM

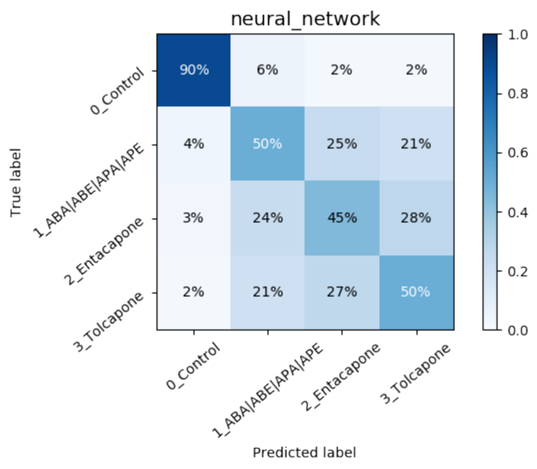


Figure 6.25: *Complete* Undersampling NN CM

For the *Mito Stress* test dataset the classifications of the RF model for Entacapone and Tolcapone improved from 25% and 31% 47% and 41% respectively. For the NN model the predictions for Entacapone improved from 35% to 50% and Tolcapone from 52% to 56%. However for both algorithms this improvement came ate the expense of the ability to classify the Control and ABA|ABE|APA|APE samples, which decreased considerably. Nonetheless the NN was the first algorithm to classify all the classes above 50%.

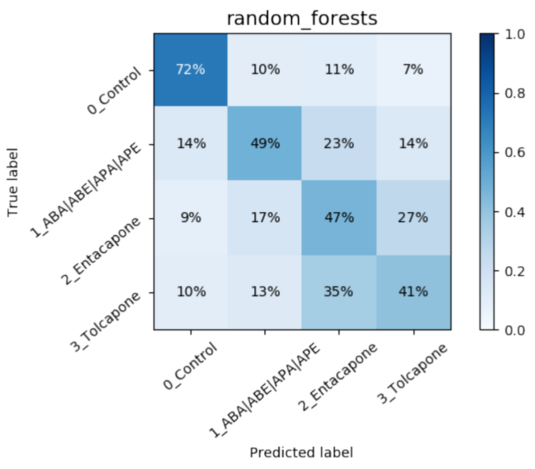


Figure 6.26: *Mito Stress* Undersampling RF CM

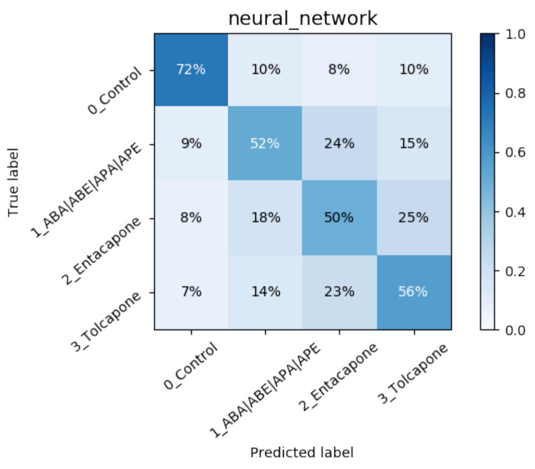


Figure 6.27: *Mito Stress* Undersampling NN CM

For the *Glycolysis Stress* test dataset the results were identical. For both RF and NN models the classifications for Entacapone and Tolcapone improved however the predictions of ABA|ABE|APA|APE decreased.

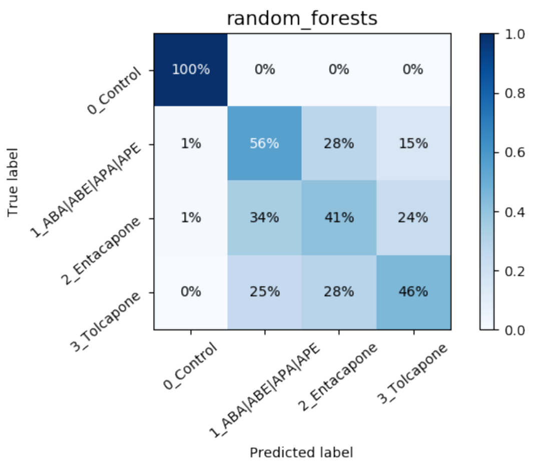


Figure 6.28: *Glycolysis Stress* Undersampling RF CM

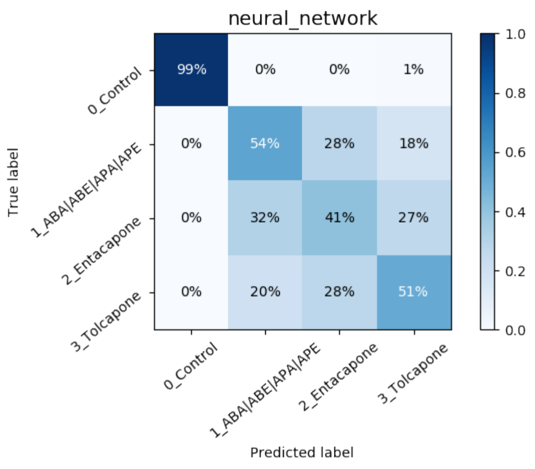


Figure 6.29: *Glycolysis Stress* Undersampling NN CM

Random Oversampling

For the *Complete* dataset the results obtained through *Random Oversampling* remained very similar, as it can be seen in *Figure 6.30* and *Figure 6.31*. For the RF model the prediction of ABA|ABE|APA|APE slightly decreased from 80% to 73% and both Entacapone and Tolcapone increased from 25% and 32% to 34% and 37% respectively. For the NN model there were no significant differences, just Entacapone increasing from 36% to 37% and ABE|ABE|APA|APE and Tolcapone slightly reducing their performance.

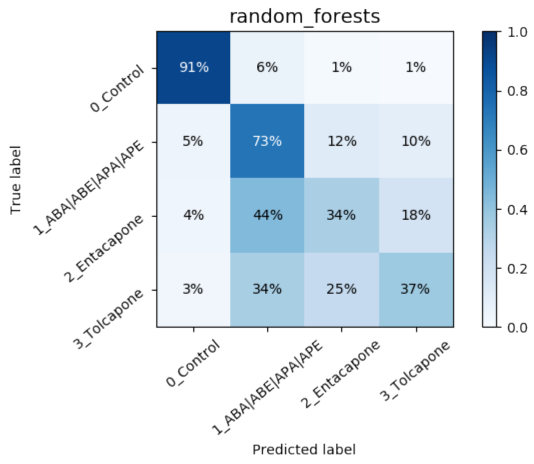


Figure 6.30: *Complete* Oversampling RF CM

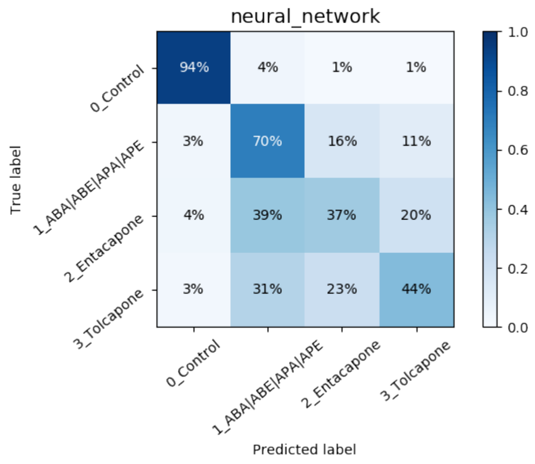


Figure 6.31: *Complete* Oversampling NN CM

There were however better results for the *Mito Stress* test dataset. The previous analyzed RF and NN models did not improve but two other algorithms, *Gaussian Process* and SVM produced good results. Their initial confusion matrices can be seen in *Figure 6.32* and *Figure 6.33* respectively. Both models were able to achieve classifications above 50% for almost all classes, as depicted in *Figure 6.34* and *Figure 6.35*. The k-NN and *Logistic Regression* models also showed similar improvements, but not as good.

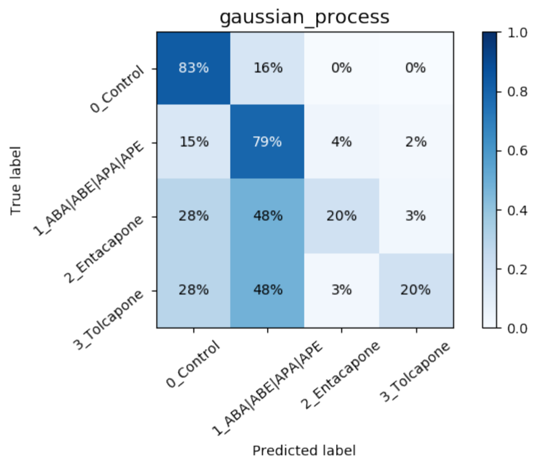


Figure 6.32: *Mito Stress* Gaussian Process CM

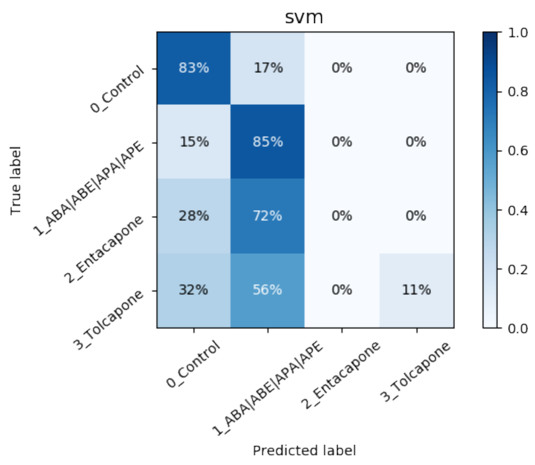


Figure 6.33: *Mito Stress* SVM CM

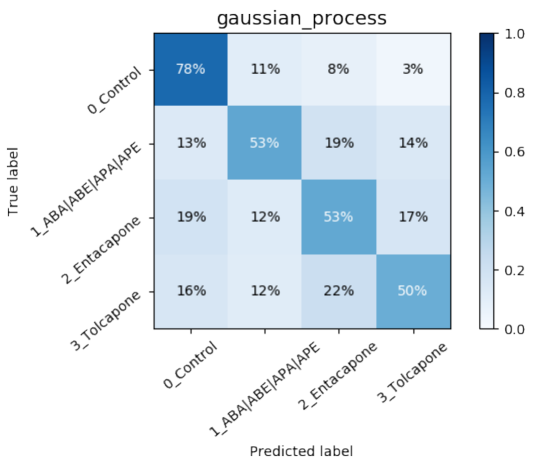


Figure 6.34: *Mito Stress* Oversampling Gaussian Process CM

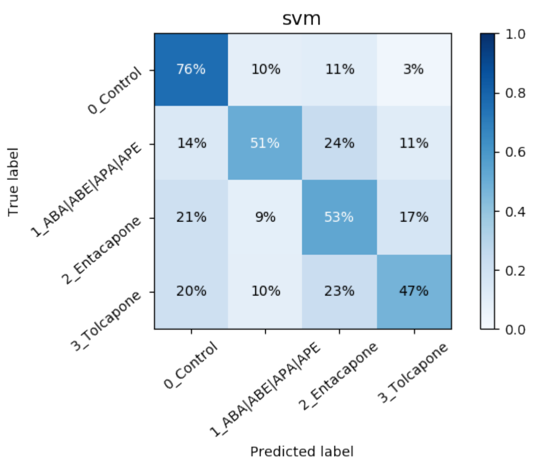


Figure 6.35: *Mito Stress* Oversampling SVM CM

For the *Glycolysis Stress* test dataset the results of the RF model, seen in *Figure 6.36*, slightly increased for Entacapone and Tolcapone from 28% and 33% to 35% and 39% respectively, and decreased from 77% to 72% in the ABA|ABE|APA|APE predictions. For the NN model an increase from 35% and 43% to 37% and 44% was seen for Entacapone and Tolcapone respectively.

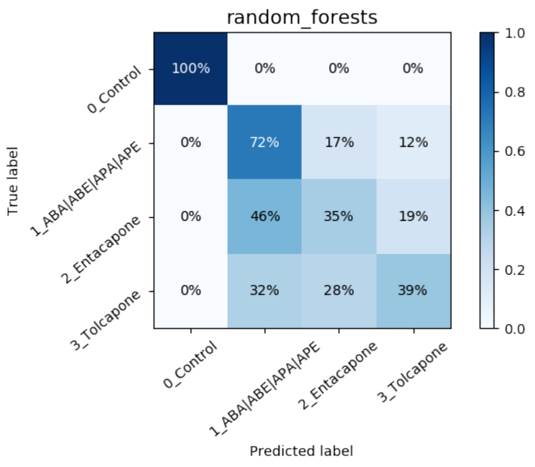


Figure 6.36: *Glycolysis Stress* Oversampling RF CM

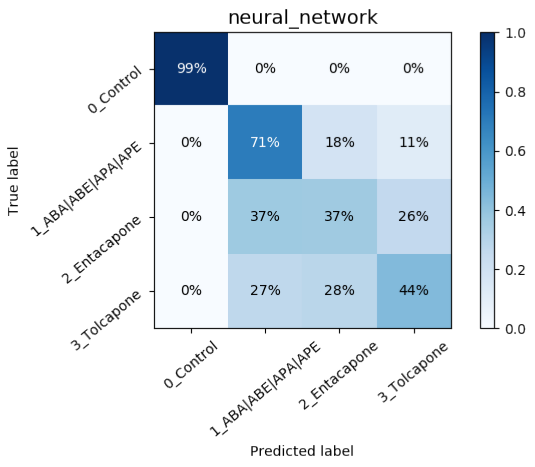


Figure 6.37: *Glycolysis Stress* Oversampling NN CM

SMOTE

For the *Complete* data this oversampling technique did not induce many changes in the results. The RF model, as illustrated in *Figure 6.39*, had some loss of performance in the class ABA|ABE|APA|APE from 80% to 69%. There was a slight increase for Entacapone and Tolcapone, from 25% and 32% to 36% and 40% respectively. For the NN model there were some losses for ABA|ABE|APA|APE from 71% to 69% and an increase for Entacapone from 36% to 37% as well as for Tolcapone from 45% to 50% as can be seen in *Figure 6.39*.

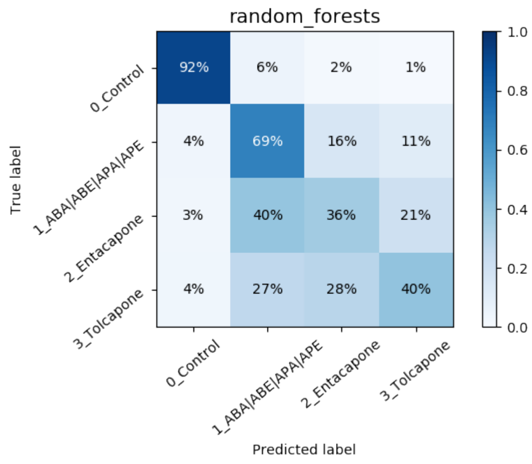


Figure 6.38: *Complete* SMOTE Dataset RF CM

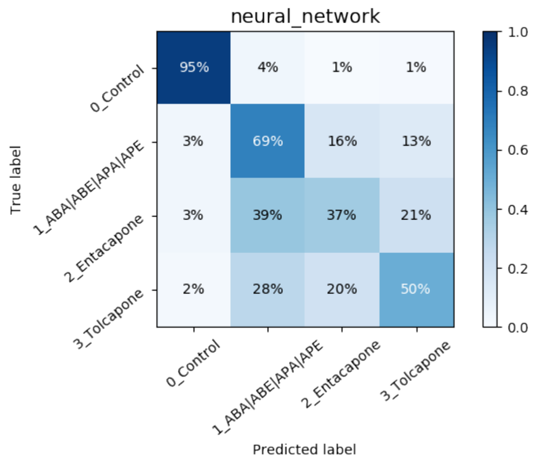


Figure 6.39: *Complete* SMOTE NN CM

Similar to what was seen with *Random Oversampling* for the *Mito Stress* test dataset the RF and NN models did not improve, however there were a couple of algorithms that presented promising results. The *Gaussian Process* and SVM, as can be seen in *Figure 6.40* and *Figure 6.41*. Compared with the results obtained with *Random Oversampling*, ABA|ABE|APA|APE increased to 54% and Tolcapone to 56%. The SVM model on the other hand did not show as much changes in comparison however Entacapone reached 52% and Tolcapone 50%.

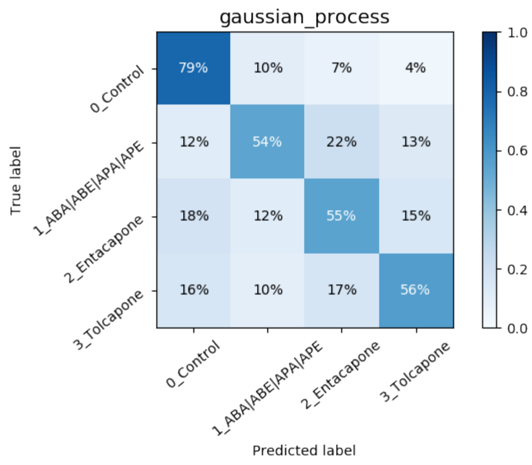


Figure 6.40: *Mito Stress* SMOTE Gaussian Process CM

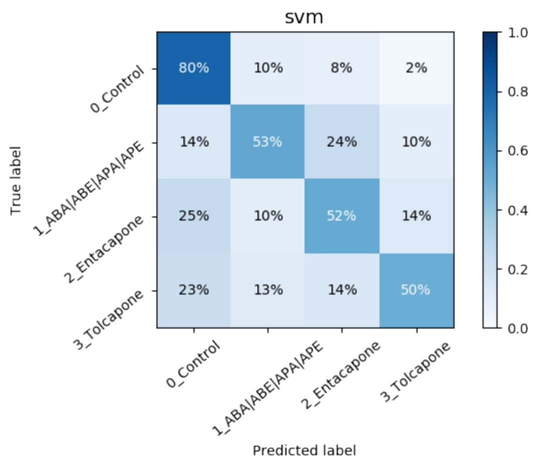


Figure 6.41: *Mito Stress* SMOTE SVM CM

For the *Glycolysis Stress* test dataset the results did not improve much for any of the models, as it can be seen in *Figure 6.42* and *Figure 6.43*. RF lost 11% for the ABA|ABE|APA|APE class and increased Entacapone and Tolcapone from 28% and 33% to 34% and 43% respectively. NN slightly improved Entacapone from 35% to 36% and Tolcapone from 43% to 46%.

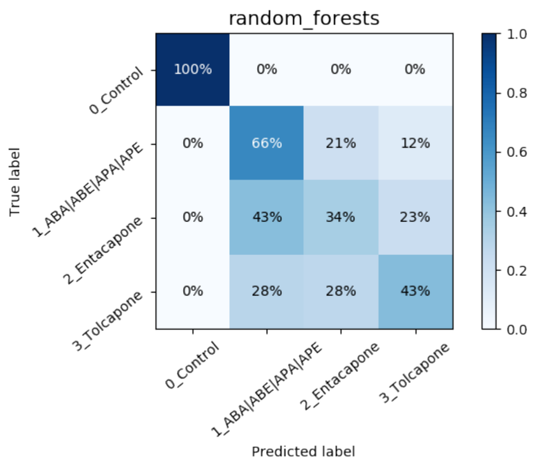


Figure 6.42: *Glycolysis Stress* SMOTE RF CM

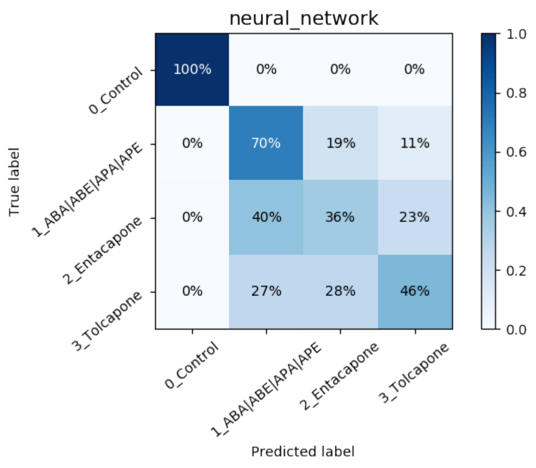


Figure 6.43: *Glycolysis Stress* SMOTE NN CM

6.2.2 Dimensionality Reduction

All the dimensionality reduction techniques described in *Section 5.1.3* were applied to the three datasets. None of them produced any improvements worth mentioning and most of the models in fact had worst results. The results can be seen in *Appendices Section B.2.1*.

6.2.3 Differentials

Due to the fact that none of the described approaches had significantly improved the initial results, a new perspective on the data was required. Thus, instead of directly using the measurements, which as previous analyzed had considerable variation for similar conditions, the differentials between the measurements were used.

For each test there are some indicators of particular interest, as detailed in *Section 4.2*. These indicators of the metabolic pathways function can be obtained by extracting values from the relations between the measurements for each test. For the *Mito Stress* these are based on the OCR and are:

- **Basal Respiration** - the difference between the OCR_BAS and OCR_ROT
- **ATP Production** - the difference between OCR_BAS and OCR_OLI
- **Proton Leak** - the difference between OCR_OLI and OCR_ROT
- **Maximal Respiration** - the difference between the OCR_FCCP and OCR_ROT
- **Spare Capacity** - the difference between the OCR_FCCP and OCR_BAS
- **Non-mitochondrial Respiration** - the OCR_ROT value directly
- **Bioenergetic Health Index (BHI)** - serves as a dynamic index of bionergetics health, and is a relation between the *Spare Respiratory Capacity*, the *ATP Production*, the *Non-Mitochondrial Respiration* and the *Proton Leak* (Chacko et al. 2014)

However for this test there are still the ECAR measurements. As for these there are no specific indicators the direct differentials between them were used (i.e ECAR_2 - ECAR_1, ECAR_3 - ECAR_2, and so on). The first ECAR measurement was also kept to avoid losing the initial condition.

For the *Glycolysis Stress* test the indicators used were based on the ECAR:

- **Glycolysis** - the difference between ECAR_GLU and ECAR_2DG
- **Glycolytic Capacity** - the difference between ECAR_OLI and ECAR_2DG
- **Glycolytic Reserve** - the difference between ECAR_OLI and ECAR_GLU
- **Non-glycolytic Acidification** - the ECAR_2DG value directly

For the OCR measurements an identical approach to the other test data was used. With these new datasets however it was no longer possible to combine the data from both tests as the differentials for each one differ in number and meaning.

The boxplots for each dataset can be seen in *Figure 6.44* and *Figure 6.45*. With this approach most of the measurements are no longer so dispersed, mainly for the *Glycolysis Stress* test data. As it is possible to observe even the experiment that had a significant offset from the average, which can still be seen in the OCR_1 feature, is no longer noticeable, indicating it indeed followed a similar range of effects as the remaining experiments.

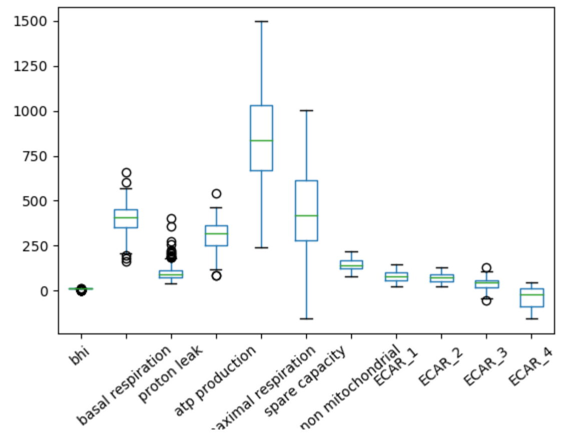


Figure 6.44: Differential - *Mito Stress* Feature Boxplot

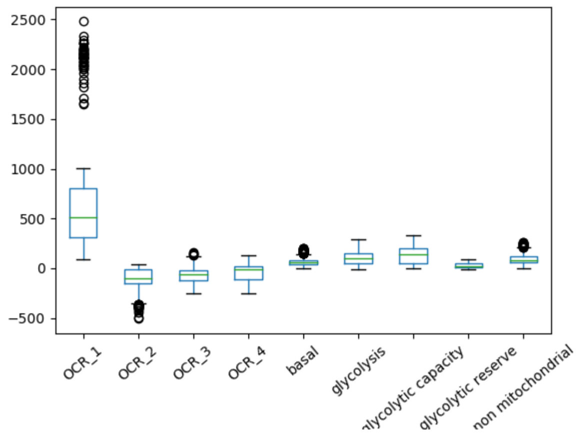


Figure 6.45: Differential - *Glycolysis Stress* Feature Boxplot

For the *Mito Stress* test data most of the models had a slight improvement in the results, similar to *Figure 6.46* and *Figure 6.47*. The results for the remaining algorithms can be seen in *Appendices Section B.2.2*. The NN model was mainly able to improve the predictions for the lower represented classes. The class ABA|ABE|APA|APE improved from 75% to 76%, Entacapone from 35% to 41% and Tolcapone from 52% to 54%. RF however lost performance and it did not reduce the confusion between the underrepresented classes.

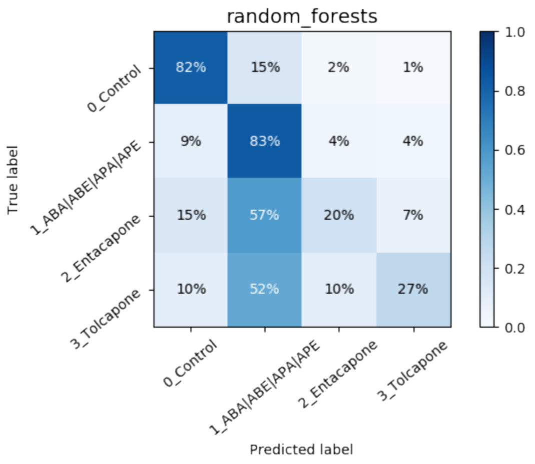


Figure 6.46: Differential - *Mito Stress* RF CM

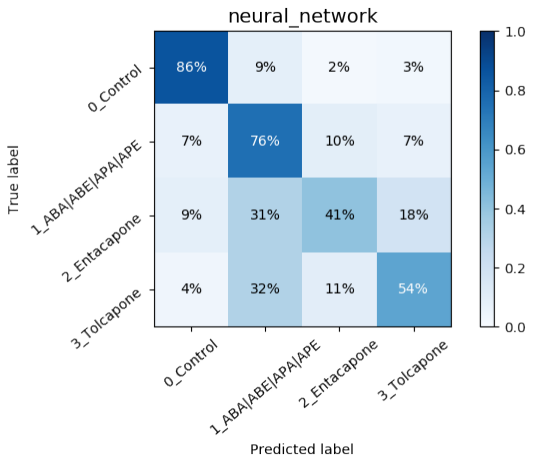


Figure 6.47: Differential - *Mito Stress* NN CM

For the *Glycolysis Stress* test data the results also did not change significantly, still some of the algorithms had a slight increase in classifying the underrepresented classes, such as *Figure 6.48* and *Figure 6.49*. The RF model improved the ABA|ABE|APA|APE samples from 77% to 79% and Tolcapone from 33% to 37%. The NN model lost some performance in ABA|ABE|APA|APE from 71% to 70%, but managed to improve Entacapone and Tolcapone from 35% and 43% to 41% and 52% respectively.

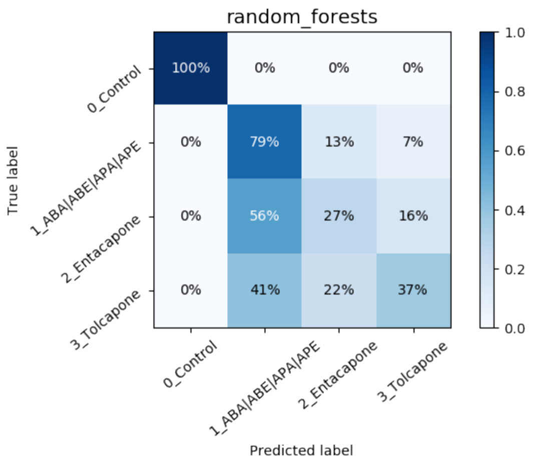


Figure 6.48: Differential - *Glycolysis Stress* RF CM

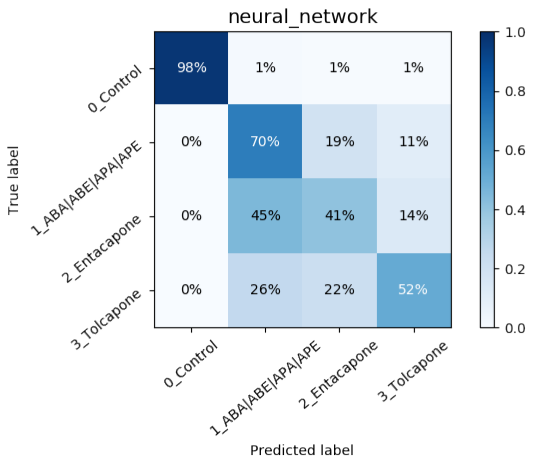


Figure 6.49: Differential - *Glycolysis Stress* NN CM

6.2.4 Individual Targets

Although some algorithms were able to classify the data better than others, the results were still poor. Hence an analysis was made to see how the algorithms would do when trying to predict each drug individually. By doing so new information was found regarding the effects of the new drugs.

As it is possible to see in *Figure 6.50* and *Figure 6.51*, for the *Mito Stress* test data the drugs ABA and ABE, as well as APA and APE, are often confused with each other. It is also possible to observe that the drugs ABA and ABE are only confused with the controls while APA and APE are often confused with Entacapone and Tolcapone.

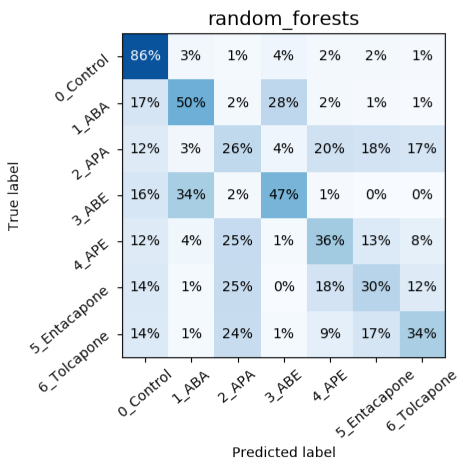


Figure 6.50: Individuals - *Mito Stress* RF CM

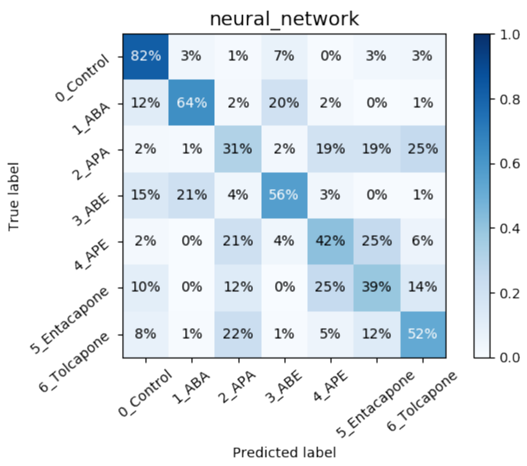


Figure 6.51: Individuals - *Mito Stress* NN CM

This is also visible in *Figure 6.52* and *Figure 6.53* which are the results with the data from the *Glycolysis Stress* test. In them it is possible to observe that almost none of the samples exposed to drugs was confused with the controls. However it shows that the drugs ABA and ABE are also confused with Entacapone and Tolcapone alongside APA and APE.

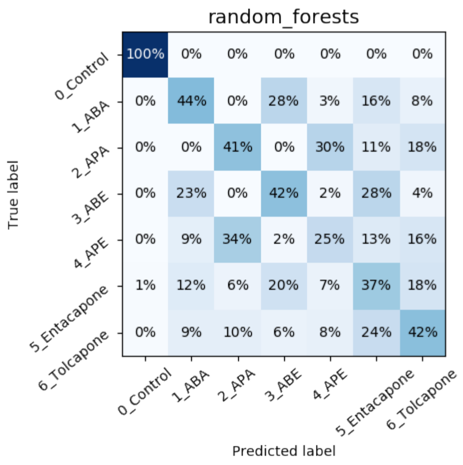


Figure 6.52: Individuals - *Glycolysis Stress* RF CM

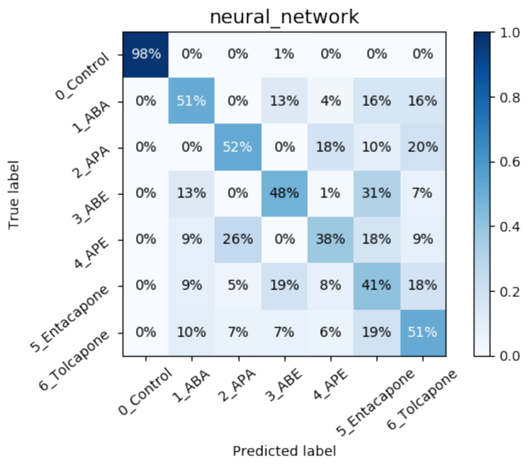


Figure 6.53: Individuals - *Glycolysis Stress* NN CM

The results for the remaining algorithms can be seen in *Appendices Section B.2.3*.

6.2.5 Grouped Drugs

After some analysis with the expert team it was conclude that the pairs with similarities shared a molecular structure and therefore it was plausible, and interesting, that they were being confused. Hence, the next attempt was to group the samples exposed to ABA and

ABE in a new class, and the ones exposed to APA and APE in another. By doing so the distribution of the classes in the datasets was as illustrated in *Figure 6.54*, *Figure 6.55*.

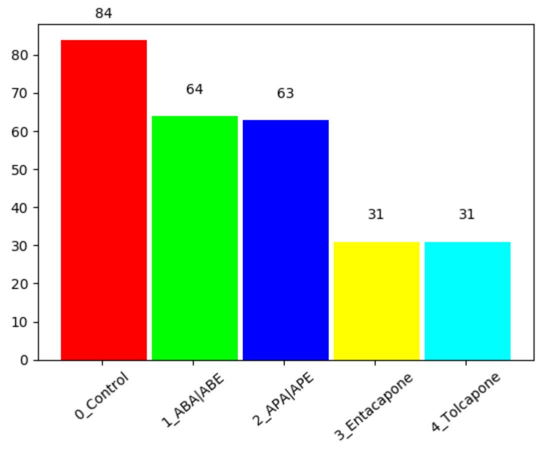


Figure 6.54: Groups - *Mito Stress* Class Distribution

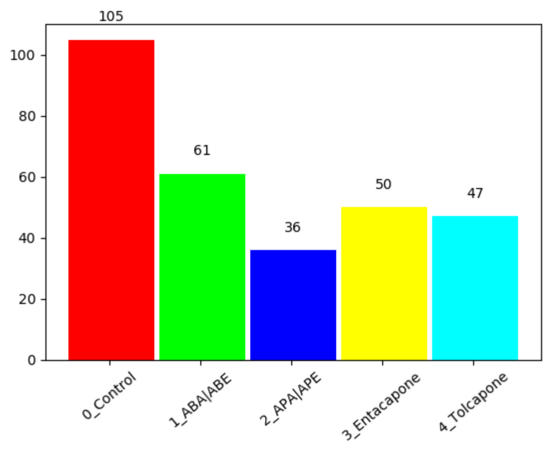


Figure 6.55: Groups - *Glycolysis Stress* Class Distribution

The results of this change with the *Mito Stress* test data can be seen in *Figure 6.56* and *Figure 6.57*. The RF model is not able to separate the Entacapone and Tolcapone samples, which are being misclassified as APA|APE. However it is possible to see that by grouping those two pairs of compounds the model was able to classify them considerably well, with 83% for ABA|ABE and 69% for APA|APE. Still, the NN model performed slightly better. For the class ABA|ABE it correctly classified 79% with little confusion with other classes, 60% for APA|APE, 41% for Entacapone and was able to correctly classify 52% of the Tolcapone samples.

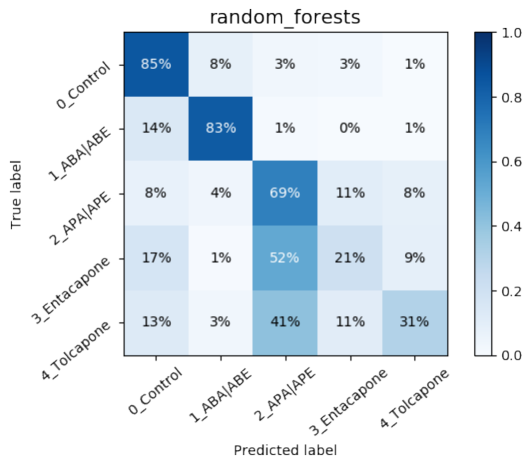


Figure 6.56: Groups - *Mito Stress* RF CM

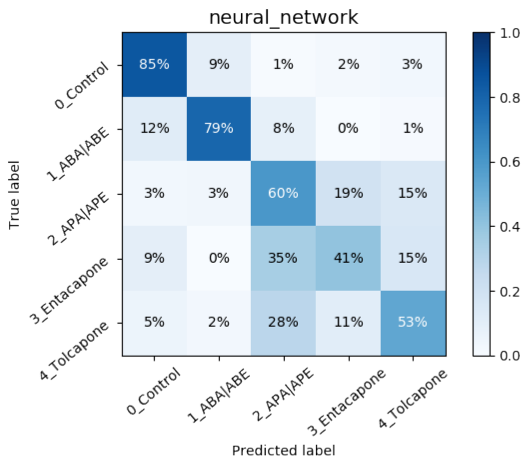


Figure 6.57: Groups - *Mito Stress* NN CM

By looking at the *Glycolysis Stress* test data it is also possible to see that the classifiers were able to acceptably distinguish between the samples from ABA|ABE and APA|APE, as can be seen in *Figure 6.58* and *Figure 6.59*. As with the data from the *Mito Stress* test here the RF model was not able to separate the Entacapone and Tolcapone samples. The

NN model was able to perform better but still only with 41% and 53% for Entacapone and Tolcapone respectively.

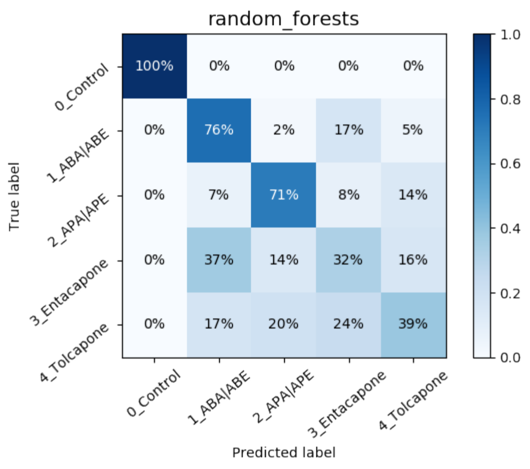


Figure 6.58: Groups - *Glycolysis Stress* NN CM

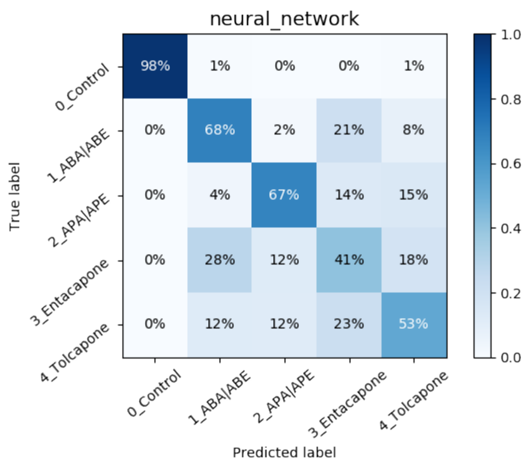


Figure 6.59: Groups - *Glycolysis Stress* NN CM

The results for the remaining algorithms can be seen in *Appendices Section B.2.4*.

6.2.6 No Controls

As there was already a way to classify the Control samples with almost 100% accuracy in both tests and the models were still having difficulty classifying the drugs samples, some tests were made by removing the controls from the dataset. The purpose was to reduce possible noise and thus allowing the algorithms to specialize on the drugs.

For the *Mito Stress* test, as illustrated in *Figure 6.60* and *Figure 6.61*, there were some small improvements. The results for the remaining algorithms can be seen in *Appendices Section B.2.5*. For the RF classifier the samples of ABA|ABE were almost entirely correctly classified, 97%, with very low confusion with other classes. APA|APE also had a high value, 73%, however almost all of the Entacapone and Tolcapone samples were also classified as APA|APE. The NN model improved in predicting ABA|ABE, Entacapone and Tolcapone, however it lost in predicting APA|APE.

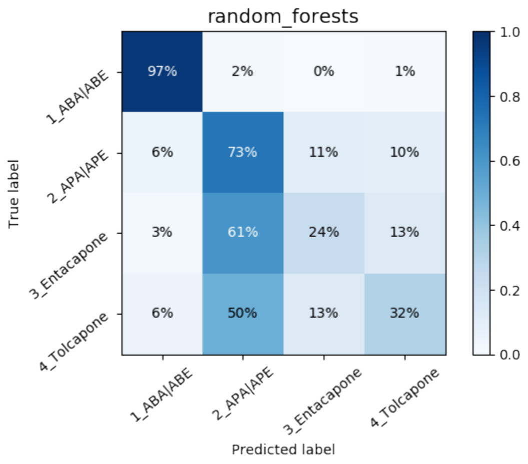


Figure 6.60: No Controls - *Mito Stress* RF CM

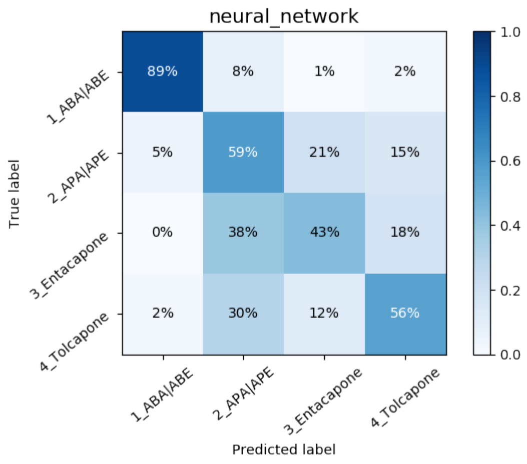


Figure 6.61: No Controls - *Mito Stress* NN CM

For the *Glycolysis Stress* test the results were not much better as can be seen in *Figure 6.62* and *Figure 6.63*. The RF classifier can still predict ABA|ABE and APA|APE acceptably well, however Entacapone and Tolcapone have very few correct predictions. The NN model improved in predicting APA|APE, Entacapone and Tolcapone and lost some performance for ABA|ABE.

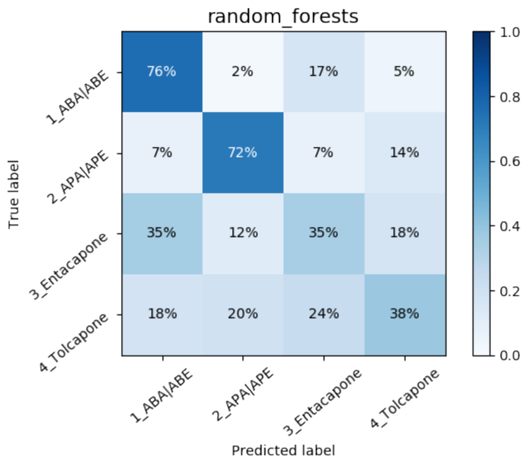


Figure 6.62: No Controls - *Glycolysis Stress* RF CM

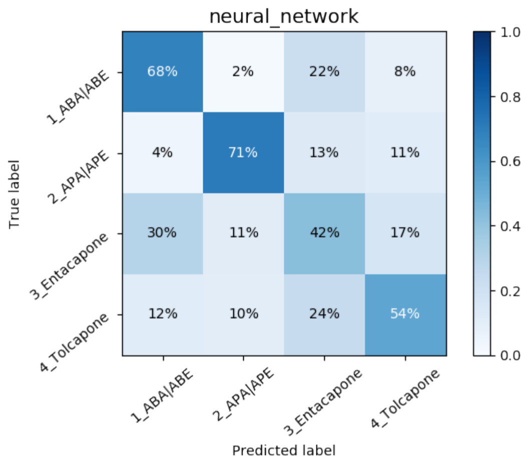


Figure 6.63: No Controls - *Glycolysis Stress* NN CM

As these are different datasets both data balancing and dimensionality reduction techniques were also tested. However, just like with the previous datasets, using the dimensionality reduction techniques did not produce any improvements on the classifiers performances. The remaining results can be seen in *Appendices Section B.2.5*.

Random Undersampling

Like with the original datasets the data balancing techniques produced some improvements. Although the final results are still not satisfactory it is possible to observe that *Random Undersampling* aids the models in predicting the underrepresented classes.

The results obtained for the *Mito Stress* data are depicted in *Figure 6.64* and *Figure 6.65*. The RF model gained some ability to classify Entacapone and Tolcapone, at the expense of the classifications of APA|APE. With this balancing the NN classifier loses some ability to identify the classes ABA|ABE and APA|APE and gained in the underrepresented, Entacapone and Tolcapone, increasing from 43% and 56% to 52% and 60% respectively.

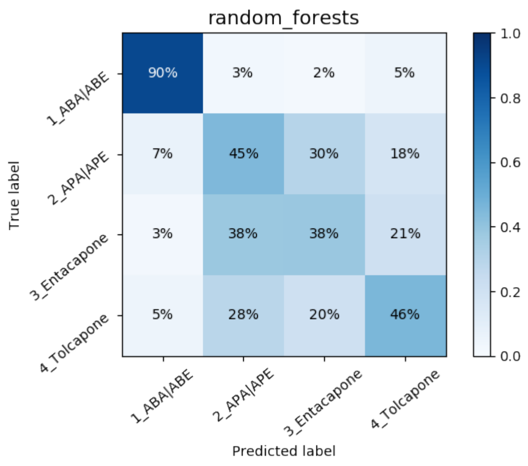


Figure 6.64: No Controls - *Mito Stress* Undersampling RF CM

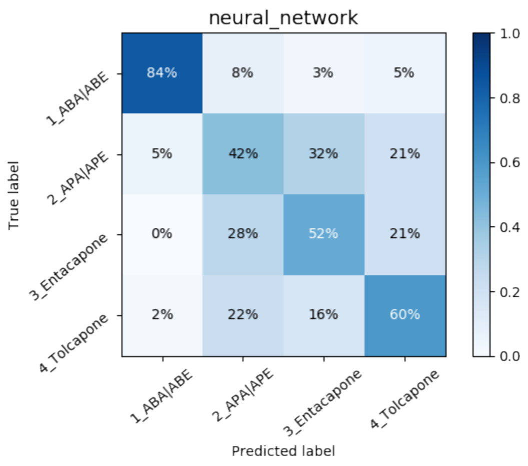


Figure 6.65: No Controls - *Mito Stress* Undersampling NN CM

For the *Glycolysis Stress* test however there were no visible improvements, and some losses can be observed in *Figure 6.66* and *Figure 6.67*.

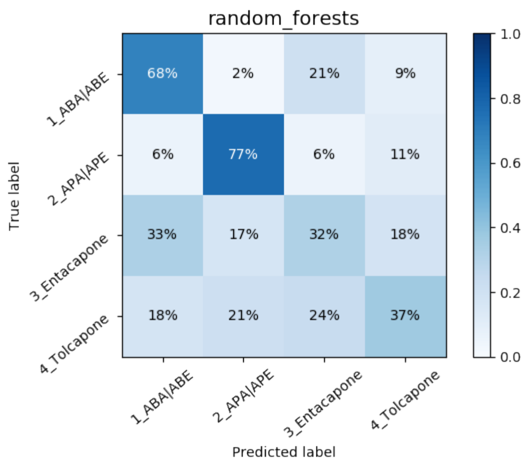


Figure 6.66: No Controls - *Glycolysis Stress* Undersampling RF CM

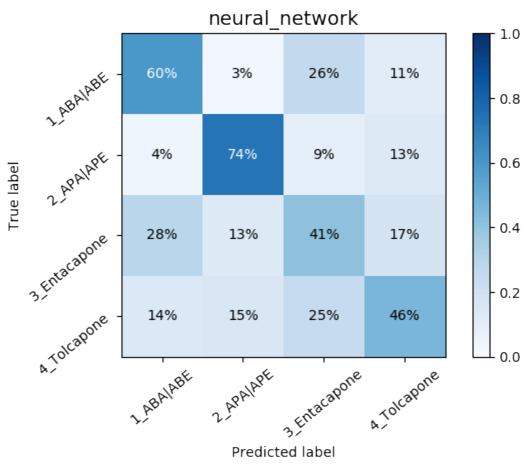


Figure 6.67: No Controls - *Glycolysis Stress* Undersampling NN CM

6.2.7 Random Oversampling

With the original dataset the oversampling techniques produced interesting results. With these there were also some algorithms that showed a considerable improvement, such as the *Gaussian Process*, which went from 26% and 39% for Entacapone and Tolcapone to 48%

and 53% respectively, as it is observable in *Figure 6.68*. The NN model kept the same results obtained with *Random Oversampling*, only increasing Entacapone by 1% *Figure 6.69*.

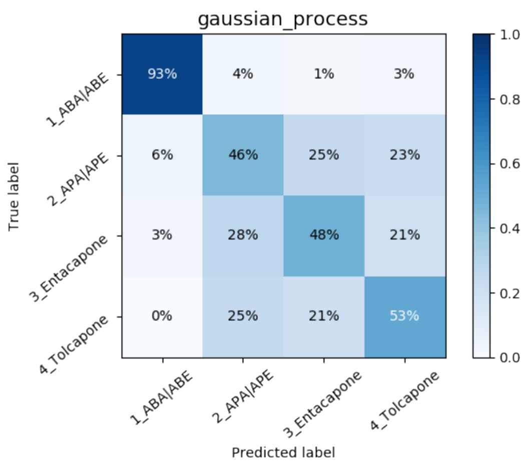


Figure 6.68: No Controls - *Mito Stress* Oversampling Gaussian Process CM

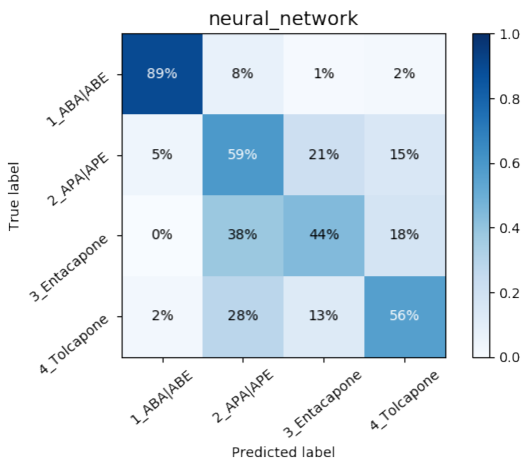


Figure 6.69: No Controls - *Mito Stress* Oversampling NN CM

Figure 6.70: Data Balancing - *Mito Stress* - Classification Algorithms

For the *Glycolysis Stress* test there was no visible improvement and some models lost performance, as illustrated in *Figure 6.71* and *Figure 6.71*.

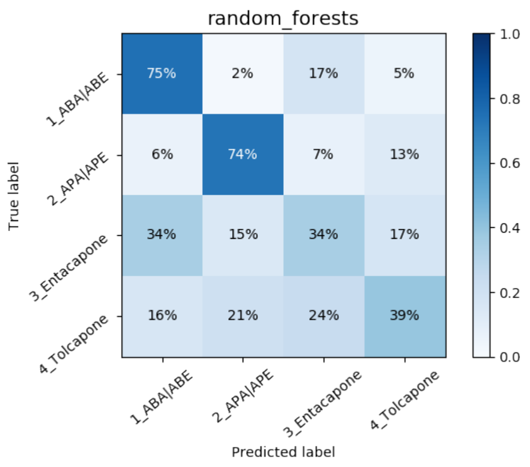


Figure 6.71: No Controls - *Glycolysis Stress* Oversampling RF CM

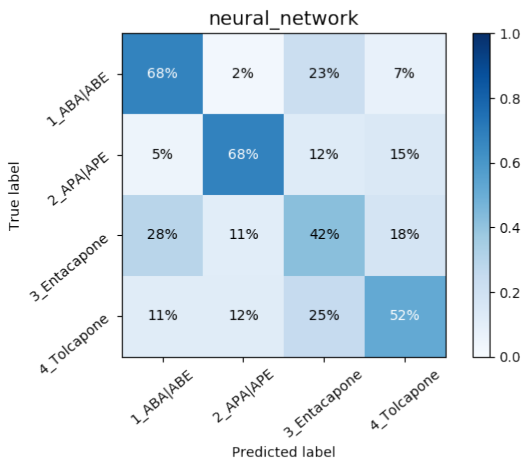


Figure 6.72: No Controls - *Glycolysis Stress* Oversampling NN CM

6.2.8 SMOTE

With the SMOTE method and following the footsteps of the previous analysis, with *Mito Stress* data some algorithms produced slightly better results, such as*Gaussian Process*, *Figure 6.73*. The NN model however kept approximately the same results, as can be seen in *Figure 6.74*.

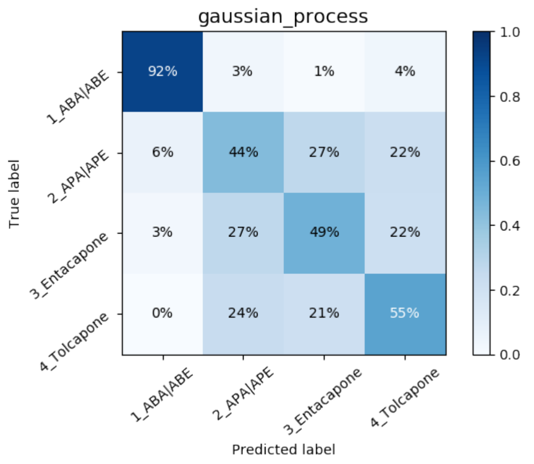


Figure 6.73: No Controls - *Mito Stress* SMOTE Gaussian Process CM

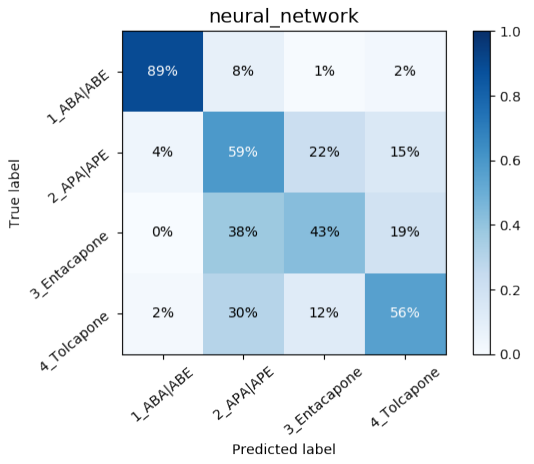


Figure 6.74: No Controls - *Mito Stress* SMOTE NN CM

For the *Glycolysis Stress* dataset the NN classifier lost some performance and the RF model gained some, as can be seen in *Figure 6.76* and *Figure 6.75* respectively, improving in most classes, mainly from 70% to 80% in APA|APE.

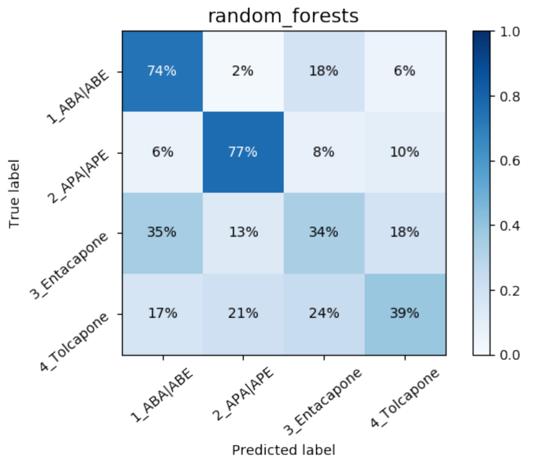


Figure 6.75: No Controls - *Glycolysis Stress* SMOTE RF CM

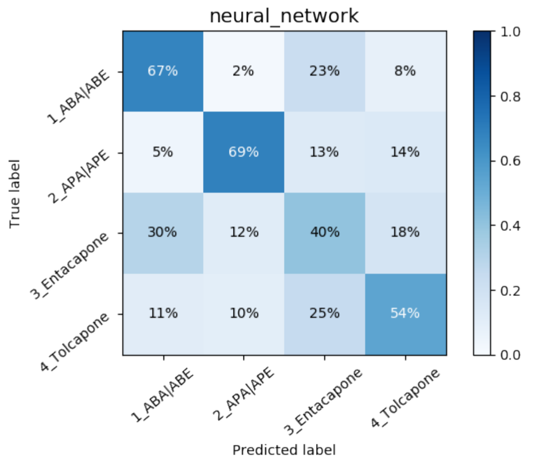


Figure 6.76: No Controls - *Glycolysis Stress* SMOTE NN CM

6.2.9 Expert Ensemble

Although some of the techniques previously applied generated better results to some extent, most of them improved the classification of some classes in exchange of some performance in others. Even still, the best algorithms had very poor overall performances, most of them struggling to go over the considered "random" 50%.

Different algorithms handle the problem of classification in different ways. Based on this fact the combination of some learners of different families may produce better results when combined than any of them individually. Hence an attempt was made to combine the classifiers that performed acceptably well in each test as an ensemble and see how this could improve the final classifier performance. To do so, and due to fairly large number of algorithms and combinations that were tested, a list was made of those that performed better. As it was previously covered there were some algorithms that were able to completely distinguish between the Control samples. Hence, an algorithm with that ability (DT with the *Glycolysis Stress* test data) was used to classify them, and the remaining algorithms of the ensemble would be used to classify the other classes.

The initial approach to the ensemble, as previously described, was a voting majority ensemble. After a thorough test and analysis, the initial algorithms chosen were:

- **NN** - one model with the *Glycolysis Stress* test data and another with *Mito Stress* test data with SMOTE balancing
- **RF** - with the *Glycolysis Stress* test data
- **Bagging** - this algorithm was used with the *Mito Stress* test with random undersampling balance
- **Gaussian Process** - with the *Mito Stress* test data with SMOTE balancing

The results obtained, illustrated in *Figure 6.77*, were a considerable improvement over the ones obtained from the individual models. For the simple majority voting the results of the ensemble were 100% correctly classified Control samples without any misclassification of other classes, 98% for ABE|ABE, 68% for APA|APE, 52% for Entacapone and 55% for Tolcapone. Some other results for different ensembles can be seen in *Appendices Section B.2.6*. Although these results were good when compared to individual performances they were still poor results. Hence the expert ensemble was implemented and the performance improved even further as can be seen in *Figure 6.78*. Initially all the components in the expert ensemble formula, seen in *Equation 5.3.6*, had equal weights of 1. Comparing with results from the voting ensemble all the classes that were having difficulties were improved. Entacapone increased from 52% to 54% and Tolcapone from 55% to 66%, however APA|APE decreased from 68% to 63%. Considering the characteristics of the dataset in hand and the initial results for individual classifiers these were finally some promising results.

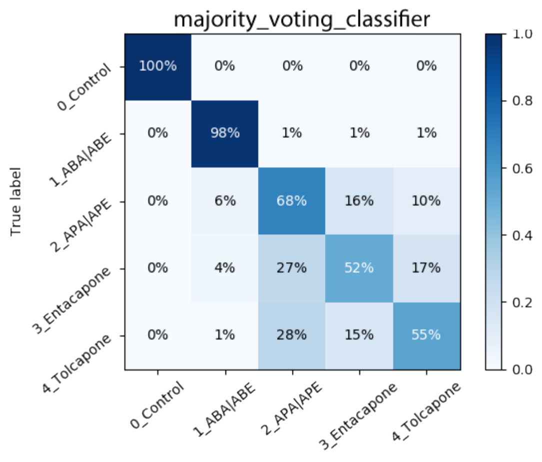


Figure 6.77: Voting Ensemble CM

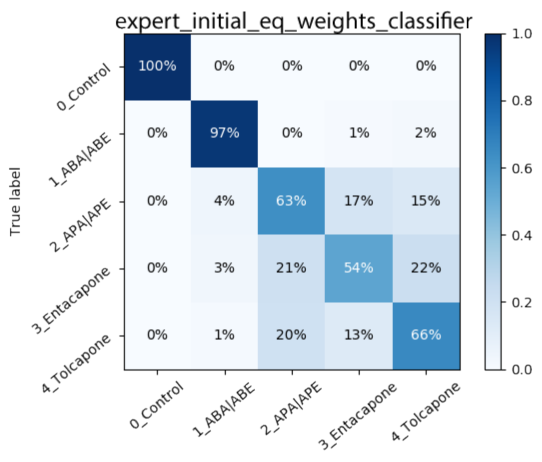


Figure 6.78: Expert Ensemble CM

Nonetheless, to perform an even thorougher analysis of the best combinations, a set of algorithms and environments were chosen:

Test	Structure	Preparation	Algorithm
Glycolysis	ABA|APA & ABE|APE	None	Bagging, Neural Network, Random Forest and SVM
Mito	ABA|APA & ABE|APE	Random Undersampling	Bagging, Decision Tree, Gaussian Process, Logistic Regression, Naive Bayes, Neural Network, Random Forest and SVM
Mito	ABA|APA & ABE|APE	SMOTE	Gaussian Process, k-NN, Logistic Regression, Neural Network and SVM
Mito	ABA|APA|ABE|APE	Random Undersampling	Gaussian Process, Logistic Regression, Neural Network and SVM
Mito	ABA|APA|ABE|APE	SMOTE	Gaussian Process, Logistic Regression, k-NN and SVM

Table 6.7: Ensemble Algorithms Base

In order to create the best ensemble the combinations between these algorithms were ran and then compared. After an analysis of the results, the combination that produced the best results, as illustrated in *Figure 6.80*, were:

- **Bagging** - with the *Glycolysis Stress* test data
- **NN** - one model with the *Glycolysis Stress* test data and another with the *Mito Stress* test data with *Random Undersampling* balancing
- **RF** - with the *Glycolysis Stress* test data
- **DT** - with the *Mito Stress* test data with *Random Undersampling* balancing
- **Gaussian Process** - with *Mito Stress* test data with *Random Undersampling* balancing
- **SVM** - with the initial grouping of the new compounds (ABA|ABE|APA|APE) and with *Mito Stress* test data with *Random Undersampling* balancing

As it can be seen in *Figure 6.79* the model was able to transversely classify all the classes. ABA|ABE class correct classifications decreased from 97% to 96%, APA|APE from 63% to 61% and Entacapone increased from 54% to 61% and Tolcapone from 66% to 71%. Still, afterwards, a sweep of the combinations of the weights for the parameters of the ensemble was done, and the best combination found was: $w_e = 1.78$, $w_p = 2.5$, $w_s = 1.2$. As it can be seen in *Figure 6.79* comparing with the previous results it generated even better results, with ABA|ABE improving to 97%, a near perfect distinction of the other classes, APA|APE and Entacapone improved to 62% and Tolcapone to 72%.

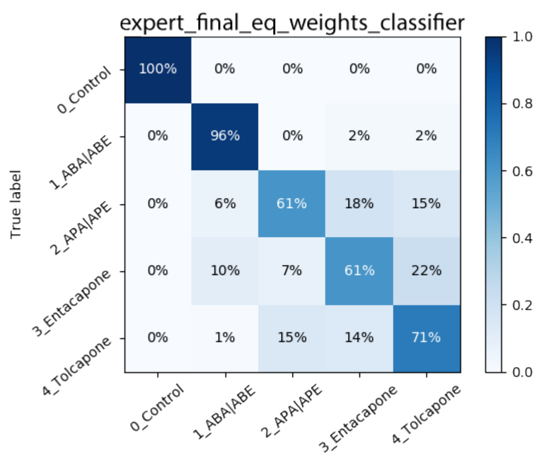


Figure 6.79: Expert Ensemble Equal Weights CM

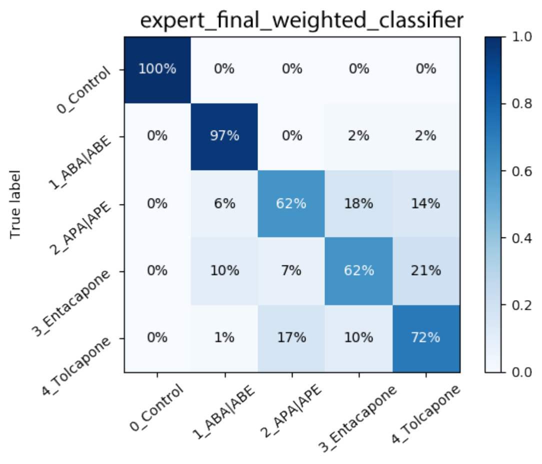


Figure 6.80: Expert Ensemble Best Combination CM

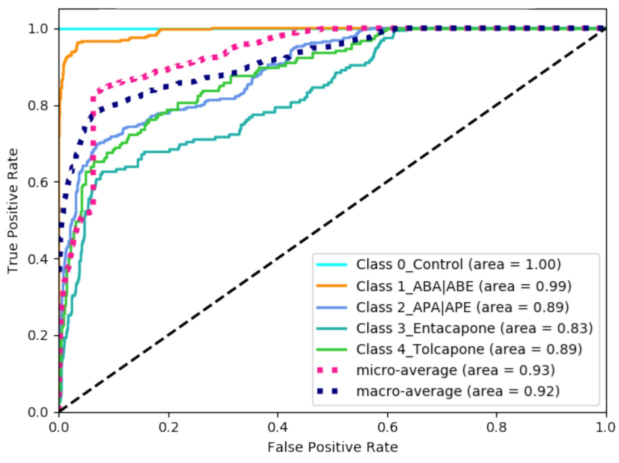


Figure 6.81: Expert Ensemble Best ROC

In *Figure 6.81* it is possible to observe the ROC curve by the different classes and the weighted average. As it can be seen the average ROC curve has a very good area under the curve of 0.94. In *Table 6.8* it is possible to see the values for the different performance metrics for the voting ensemble and the expert ensemble with equal and optimized weights. The performance metrics obtained by the voting ensemble were already acceptable as well, with an F1-score of 0.78, accuracy and recall of 0.79 and the precision of 0.80. The initial version of the expert ensemble was able to improve these results, by increasing the accuracy and recall to 0.81 and the F1-score and precision to 0.82. The best results in all the performance metrics, and agreeing with the confusion matrices analyzed, belongs the expert ensemble with the optimized weights, which improved the F1-score, accuracy and recall to 0.83 and the precision to 0.84.

6.2.10 Genetic Programming (GP)

Genetic Programming (GP) was the last development in the attempt of creating a classification model. Initially an implementation was made to create a classifier, similar to the

Perf. \ Alg.	Voting	Initial Ensemble	Expert Ensemble
F1	0,78	0,82	0,83
Accuracy	0,79	0,81	0,83
Precision	0,80	0,82	0,84
Recall	0,79	0,81	0,83

Table 6.8: Ensemble Best Classification Performance

ones obtained by the other classes. To achieve that, the MDLP was used to discretize the continuous value returned by the GP solutions into multiple bins. However, this implementation did not achieve an acceptable performance, as it was not able to distinguish between any of the drugs, instead almost every sample was considered as Control. The ranges of the bins that were created through the discretization were not useful either, as there was no visible pattern or logic amongst them. This poor performance was most likely due to the dispersion of the features values which made it difficult to create a program that was able to model such complexity. Data balancing and dimensionality reduction techniques were not able to improve the results either

Other GP implementation intended to give more insight on the results obtained by the classification models. Although the results previously documented tell that the effects of the compounds are different from each other, they do not provide any information regarding which are more toxic. The purpose of the GP approach was to evolve a program that would try to create an index that would allow to predict the compounds toxicity. For this, the *Complete* and the differentials datasets were used with the grouping of the pairs ABA|ABE and APA|APE. The results can be seen in *Table 6.9*.

Algorithm	Drug Index Order
Complete Dataset	APA|APE=>ABA|ABE=>Entacapone=>Tolcapone
Differential - Mito Stress	ABA|ABE=>APA|APE=>Entacapone=>Tolcapone
Glycolysis Differential	APA|APE=>Entacapone=>ABA|ABE=>Tolcapone

Table 6.9: Genetic Programming (GP) Datasets Compounds Toxicity

By looking at the table it is possible to observe that the compounds are indeed grouped differently across the datasets. For the *Mito Stress* test the solution places the compounds ABA|ABE with the lowest indexes followed by APA|APE, then Entacapone and finally Tolcapone. For the *Glycolysis Stress* test however the results were different. The compounds with the lowest indexes were APA|APE. ABA|ABE appears between Entacapone an Tolcapone. Using the *Complete* APA|APE appear as the drug with the lowest indexes, followed by ABA|ABE, then Entacapone and finally Tolcapone. These differences between the *Mito Stress* and the *Glycolysis Stress* test data corroborates the possibility that the compounds have different effects on each pathway.

6.2.11 Models Comparison

Although the evolution of the ML models was mainly based on the interpretation of the performance metrics and the confusion matrices, in order to undoubtedly state that the final model is better than the remaining a statistical comparison was made. The models used for the comparison were: **SVM** with SMOTE data balancing, **NN** with *Random Undersampling* data balancing, **Voting Ensemble** with the initial combination of algorithms and **Expert Ensemble** with the final combination of algorithms and weights. The performance metric used to compare them was the **F1-Score** as it is a measure that balances both the precision and recall.

The null hypothesis (H_0) that was to be tested was that the means of the performances of all four models were the same and the alternative hypothesis (H_1) was that not all of them were equal:

$$\begin{aligned} H_0 &: \exists(i,j) : \mu_i = \mu_j \\ H_1 &: \exists(i,j) : \mu_i \neq \mu_j \end{aligned} \tag{6.1}$$

As the subjects of the experiments were not the same throughout the process an unpaired statistical test must be used. Then, to decide between parametric and non-parametric tests the normality of the data was tested with the *Shapiro-Wilk normality test* which gave the following results: $W = 0.81731, p - value = 6.692e^{-11}$, that proved that the data did not follow a normal distribution. Thus, based on all these conditions the *Kruskal-Wallis* non-parametric test was chosen (Field 2013). The results for this test were: $chi - squared = 98.118, df = 3, p - value < 2.2e^{-16}$. With this p-value it was safe to state that there are differences between at least two of the algorithms for a significance level of 5%. To identify which models were indeed different the *Dunn test* with *Bonferroni* correction was used. The results can be seen in *Table 6.10*.

	SVM	**NN**	**Voting Ensemble**
NN	-2.917119 0.0106		
Voting Ensemble	-7.842078 0.0000	-4.924958 0.0000	
Expert Ensemble	-8.435894 0.0000	-5.518774 0.0000	-0.593815 1.0000

Table 6.10: *Dunn's test With Bonferroni Correction*

From these results it was possible to observe that SVM was different from the remaining three algorithms, NN was also different from both ensembles and that between these the null hypothesis holds for a significance level of 5%.

These results statistically prove that the ensembles used were indeed a significant improvement over the best individual models.

Chapter 7

Discussion

Due to the variability and small size of the dataset it was accepted from the start that it would be a challenge to create a model that would be able to generalize to the effects from the different drugs. Also, the concentrations of the compounds that were used in the experiments carried by the MitoXT team were of already non-toxic dosages, that is, they were concentrations for which there was no measurable or significant toxicity. This implies that whatever effects could arise would already be difficult to find.

7.1 Clustering

As previously stated, one of the main goals with clustering was to analyze how the samples would be organized in groups (clusters) based on their similarity. What was hopped to be seen was that the drugs known to have toxic effects would be in separate clusters and that the new drugs would either be in their own cluster or gathered with the controls, indicating that they had no significant effects.

However, none of the algorithms was able to make such a separation. The algorithms used were unsupervised, hence no knowledge of the samples compound was used to develop the model. Also, the dataset contained a small number of samples and a considerable amount of diversity and variability. These conditions combined made it unlikely for any algorithm to separate the samples by the effects of the drugs. Although very large datasets can also pose a challenge to clustering algorithms, if the dataset is too small and there is not enough samples to represent the problem then it becomes virtually impossible to create representative clusters.

Nonetheless, there were some insights that could be drawn from the results. Using DBSCAN it was possible to create clusters exclusively, or almost, of Control samples. This indicated that the despite the variance in the data the algorithm was still able to separate them from rest. With other algorithms it was also possible to observe that there were two pairs of drugs, ABA and ABE, and APA and APE, which were often, and mutually exclusively, clustered together. This suggests that the effects of the drugs in these pairs are most likely similar, whilst being distinguishable from the other pair.

7.2 Classification

As these are supervised problems, the algorithms were developed based on the target information in the dataset. However, even using it most of the algorithms struggled in generalizing, thus having low performances. Once again, the size, diversity and variance of the dataset were part of the reason why the classification algorithms were not able to thrive.

Algorithms from different families and multiple combinations of the data were used, yet none were able to create a model that could acceptably make predictions on unseen examples. As a result, data balancing and dimensionality reduction techniques were applied to attempt to overcome some of the limitations imposed by the dataset. Both undersampling and oversampling techniques were able to slightly improve some classifiers performance, which due to their nature are more sensitive to unbalanced datasets. It allowed those algorithms to better classify the original underrepresented classes. SMOTE, one of the oversampling methods, was the one that most improved the results. The dimensionality reduction techniques were used with the expectation that the removal of irrelevant and noisy features would allow the algorithms to focus on the useful data, however none were able to improve the classifiers performance.

Nonetheless, using the differentials between the measurements in the data increased most of the algorithms performance. Also, by making predictions using the individual drugs it was possible to identify similarities between the new drugs that were not initially defined. As the similar drugs share a common molecular structure (Silva et al. 2016) the confusion amongst them may indicate that the effects on the cells are identical. Their aggregation, ABA with ABE and APA with APE, allowed the algorithms to create better decision boundaries between the classes, thus improving their classification performances. These were the same drugs that were also found to be clustered together. Further, the fact that ABA and ABE are not confused with Entacapone nor Tolcapone in the *Mito Stress* but they are in the *Glycolysis Stress* test data may indicate that their effects are more alike in the glyoclytic pathway than the OXPHOS. It was also concluded that by exclusively using the *Glycolysis Stress* test data it was possible to completely distinguish the Control samples from the remaining.

The results obtained by the ensembles demonstrated once again the power and flexibility of such methods. By combining algorithms that were otherwise weak it was at last possible to achieve some interesting results. The majority voting ensemble proved that a simple implementation can generate a model with a higher performance than any of the algorithms did individually. The expert ensemble developed weights the predictions of the models, their performance in training and the probabilities of their predictions. With this approach it was possible to considerably improve the results. One of the reasons that allowed the ensembles to perform so well was the varied nature of their algorithms and how they were able to focus on different areas of the datasets.

For the best ensemble, by using the *Glycolysis Stress* test data it was possible to have 100% correct classifications of the Control samples. Hence, a DT model trained with those data was used for that the sole purpose. Models of the algorithms *Bagging*, NN, SVM and RF using these data were able to acceptably classify across the drugs. By using the *Mito Stress* test dataset without the Control samples and with the new drugs grouped in a single class the algorithms were better at distinguishing them from Entacapone and Tolcapone. By oversampling the data and thus removing its imbalance the *Gaussian Process* was also able to acceptably distinguish between the new drugs and Entacapone and Tolcapone. DT and NN however benefited from balancing the same data through undersampling.

As it is known, different algorithms are able to explore different areas of the problem, derived from their internal mechanisms. The fact that they were tested in multiple datasets that benefit certain characteristics allowed them to perform better where they excelled. After performing a statistical comparison between the best individual algorithms and the ensembles it was shown that the latter were indeed better. Although there were no significant differences between the ensembles, the expert ensemble with the final configurations was able to outperform the remaining in all performance metrics.

By using GP, although it does not give a formal statement of the drugs toxicity from a biological point of view, it is nonetheless an indication of their effects. The high variance in the dataset poses a challenge to the GP algorithm due to the complexity of developing a single formula that could account for all the diversity. Still, the results were consistent with the data previously obtained from the classifiers and exploratory analysis. For the *Mito Stress* dataset the results comply with the classifications predictions, where the compounds APA and APE were commonly misclassified as Entacapone or Tolcapone. For the *Glycolysis Stress* the results were also in agreement with the insights obtained from the classification models, where the compounds ABA, ABE, Entacapone and Tolcapone were misclassified with one another. What these results indicate is that the drugs ABA and ABE create less dysfunction in the OXPHOS pathway than APA and APE, but all four induce less changes than Entacapone and Tolcapone. However, the results obtained for the *Glycolysis Stress* test suggest that APA and APE generate less dysfunction on the glycolytic pathway than all the remaining drugs, and ABA and ABE induce more than Entacapone but less than Tolcapone. By combining the data from both tests it hints that APA and APE are the drugs that create less dysfunction in both metabolic pathways, followed by ABA and ABE, Entacapone and Tolcapone.

The results obtained were already validated and presented at the International symposium MitoPorto - Advances in mitochondrial research, University of Porto, May 26th, 2017, as can be seen in the *Appendix Section B.2.8*. Also, the main insights and results obtained regarding the comparative analysis between the different compounds will be submitted to an international journal from the computational biology/ML area.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The purpose of this work was to create a Machine Learning (ML) model to predict the toxicity of pharmacological compounds present in the drugs. This was to be done by detecting mitochondrial perturbation using data pertaining to the ability of the mitochondria to generate ATP through OXPHOS and the Extracellular Acidification Rate (ECAR). In the process of developing the models the standard ML techniques such as data preparation, data balancing and dimensionality reduction would be used. Both clustering and classification algorithms were to be tested to explore the data and create a model that would perform well and be able to generalized to new data.

Due to the intrinsic nature of the problem it was a challenge from the beginning, starting with the fact that the dataset available was considerably small and contained a high level of variance. In order to even understand and prepare the data a significant amount of time was spent studying and understanding the problem.

Throughout the development of this project a lot of effort was put into creating better ML models. With the original data it was necessary to invest a considerable amount of work into analyzing and preparing it so that it could be safely used with ML algorithms. Through the use of several data preparation techniques, from data balancing to dimensionality reduction, it was possible to iteratively improve some of the algorithms performance. Afterwards, by interpreting the data and using some key indicators between the measurements to reduce the overall variance of the data proved to be an effective way to boost their performance.

Using the ensemble paradigm has shown once again that the combination of weak learners can lead to very good results even with small datasets. The development of an expert ensemble heuristic that considered the performance of the algorithms in predicting each class, their probabilities and respective predictions demonstrated to be a compelling heuristic to combine the decisions of multiple algorithms in a comprehensive way. After performing a statistical analysis it was proven that the expert ensemble developed was indeed better than the best individual algorithms. Based on these results it is safe to say that the achieved model is above and beyond the threshold of success defined and viable to assist in the drug development process.

The resulting model that is able to predict the mitochondrial toxicity is an innovation in the field of drug discovery and toxicity prediction. There is no other method or tool that uses the *in vitro* measurements of ECAR and OCR to identify and predict the drugs toxicity through the ATP metabolic pathways. The fact that it is not only a binary classifier but can in fact distinguish by the degree of dysfunction is yet another remark. Also, the publishing of the preliminary results in a symposium specifically directed to the study of mitochondria highlights the interest and innovation of such a tool.

The use of a Genetic Programming (GP) algorithm also proved to be able to provide some insight on the development of a formula to predict the toxicity of the compounds. The results obtained by this model, although preliminary, were in agreement with the classifiers.

Hence, based on what was achieved it is safe to say that Machine Learning (ML) is indeed an area that can provide new insights into biological inspired datasets. With it is possible to extract knowledge even from small datasets with high variability and noise.

From a biological perspective, once again it was shown that the mitochondrial function can be a good indicator of drug toxicity. Through the use of the ML algorithms it was possible to completely distinguish the control samples and amongst those exposed to drugs to separate them by compound. In regard to the new drugs there are good indicatons that their effects are in fact different and cause less dysfunction than Tolcapone.

8.2 Future Work

The development done in this work was based on a small and limited dataset. The use of larger datasets with more drugs and experiments would be necessary to draw significant conclusions, both in terms of the new ensemble approach and the biological interpretation of the results. It would also be interesting to have enough experiments of multiple concentrations for each drug that would allow a study regarding the effects of the concentrations as well as the drugs.

Although many ML algorithms were used most of the parameters and configurations chosen for each one were based on existing literature and *adhoc* optimizations. A thorough study of the individual algorithms performance for multiple parameters would possible improve their results, which in turn could eventually improve the ensemble results.

Further work on the Genetic Programming (GP) would also be interesting as there were some promising preliminary results, however due to the data and time limitations of this project it was not possible to develop further. A thorough study of other GP approaches, such as representations and operators, using larger populations and generations could also yield better solutions. Such work however would also imply a strong support of a team with a sound biological background to validate and assist on the development of a formula that is both able to calculate the toxicity and make logic from a biological point of view.

Bibliography

Aggarwal, Charu C. (2013). *Outlier Analysis*. Springer.

Aggarwal, Charu C. (2015). *Data Mining: The Textbook*. Springer Publishing Company, Incorporated.

Agilent Technologies (2016). Accessed: 2017-05-03. url: `http://www.agilent.com`.

Agre, Gennady and Stanimir Peev (2002). "On Supervised and Unsupervised Discretization". In: *Bulgarian Academy of Sciences*.

Alberts, B. et al. (2002). *Molecular Biology of the Cell*. 4th. Garland.

Alpaydin, Ethem (2014). *Introduction to Machine Learning*. The MIT Press.

Ayodele, Taiwo Oladipupo (2010). "Types of Machine Learning Algorithms". In: *New Advances in Machine Learning*.

Borra, Simone and Agostino Di Ciaccio (2010). "Measuring the Prediction Error. A Comparison of Cross-validation, Bootstrap and Covariance Penalty Methods". In: *Comput. Stat. Data Anal.* 54.12.

Bruha and F. Franek (1996). "Comparison of various routines for unknown attribute value processing: covering paradigm". In: *International Journal of Pattern Recognition and Artificial Intelligence* 10.

Cano, Jose Ramon, Francisco Herrera, and Manuel Lozano (2005). "Strategies for Scaling Up Evolutionary Instance Reduction Algorithms for Data Mining". In: ed. by Ashish Ghosh and Lakhmi C. Jain, pp. 21–39. doi: `10.1007/3-540-32358-9_2`. url: `http://dx.doi.org/10.1007/3-540-32358-9_2`.

Celebi, M. Emre (2015). *Partitional Clustering Algorithms*. 1st ed. Springer International Publishing.

Chacko, B. K. et al. (2014). "The Bioenergetic Health Index: a new concept in mitochondrial translational research". In: *Clin. Sci.* 127.6.

Chapelle, Olivier, Bernhard Schlkopf, and Alexander Zien (2010). *Semi-Supervised Learning*. 1st. The MIT Press.

Chawla, Nitesh V. et al. (2002). "SMOTE: Synthetic Minority Over-sampling Technique". In: *J. Artif. Int. Res.* 16.1.

Chen, J. J. (2011). "Pharmacologic safety concerns in Parkinson's disease: facts and insights". In: *Int. J. Neurosci.* 121 Suppl 2.

Cyprotex (2013). *Mechanisms of Drug-Induced Toxicity*.

Dash, Rajashree, Rajib Lochan Paramguru, and Rasmita Dash (2011). "Comparative Analysis of Supervised and Unsupervised Discretization Techniques". In: *International Journal of Advances in Science and Technolog* 2.3.

Dasu, Tamraparni and Theodore Johnson (2003). *Exploratory Data Mining and Data Cleaning*. 1st ed. New York, NY, USA: John Wiley & Sons, Inc.

Differences between in vitro, in vivo and in silico studies (2017). Accessed: 2017-05-31. url: `https://mpkb.org/home/patients/assessing_literature/in_vitro_studies`.

Divakaruni, A. S. et al. (2014). "Analysis and interpretation of microplate-based oxygen consumption and pH data". In: *Meth. Enzymol.* 547.

Domingos, Pedro (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books.

Dykens, James A. and Yvonne Will (2007). “The significance of mitochondrial toxicity testing in drug development”. In: *Drug Discovery Today* 12.

Dykens, James A. and Yvonne Will (2008). *Drug-Induced Mitochondrial Dysfunction*. New York, NY, USA: John Wiley & Sons, Inc.

Ester, Martin et al. (1996). “A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press.

Fayyad, Usama M. and Keki B. Irani (1993). “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning.” In: *IJCAI*.

Field, Andy (2013). *Discovering Statistics Using IBM SPSS Statistics*. 4th. Sage Publications Ltd.

Friedman, Jerome H. (1997). “Data mining and statistics: What’s the connection”. In: *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*.

Fundamentals of Statistics (2012). Accessed: 2017-06-05. url: `http://www.statistics4u.info/fundstat_eng/cc_error_types.html`.

Ghosh, Ashish (2005). *Evolutionary computation in data mining*. Springer-Verlag Berlin Heidelberg.

Gidudu, Anthony, Greg Hulley, and Tshilidzi Marwala (2007). “Image Classification Using SVMs: One-against-One Vs One-against-All”. In: *CoRR* abs/0711.2914.

Grzymala-Busse, Jerzy W. and Ming Hu (2001). “A Comparison of Several Approaches to Missing Attribute Values in Data Mining”. In: ed. by Wojciech Ziarko and Yiyu Yao, pp. 378–385. doi: `10.1007/3-540-45554-X_46`. url: `http://dx.doi.org/10.1007/3-540-45554-X_46`.

Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim (1998). “CURE: An Efficient Clustering Algorithm for Large Databases”. In: *SIGMOD Rec.* 27.2.

Guyon, Isabelle et al. (2002). “Gene Selection for Cancer Classification Using Support Vector Machines”. In: *Mach. Learn.* 46.1-3.

Han, Jiawei (2005). *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Springer. Springer.

III, Hal Daumé (2012). *A Course in Machine Learning*.

James, Gareth et al. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Kelleher, John D., Brian Mac Namee, and Aoife D’Arcy (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press.

Kotsiantis, S. B., D. Kanellopouloss, and P. E. Pintelas (2006). “Data Preprocessing for Supervised Learning”. In: *International Journal of Computer Science*.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press. isbn: 0-262-11170-5.

Kubat, Miroslav and Stan Matwin (1997). “Addressing the Curse of Imbalanced Training Sets: One-Sided Selection”. In: *In Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann.

Lakshminarayan, Kamakshi, Steven A. Harp, and Tariq Samad (1999). “Imputation of Missing Data in Industrial Databases”. In: *Applied Intelligence* 11.

Lane, Nick (2006). *Power, Sex, Suicide - Mitochondria and the Meaning of Life*. Oxford University Press, USA.

Legg, Shane and Marcus Hutter (2006). “A formal definition of intelligence for artificial systems”. In: *In Proc. 50th Anniversary Summit of Artificial Intelligence*.

Liu, Alexander Yun-chung (2004). “The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets”. MA thesis. University of Texas at Austin.

Liu, Huan and Hiroshi Motoda (2001). *Instance Selection and Construction for Data Mining*. 1st ed. The Springer International Series in Engineering and Computer Science 608. Springer US.

Loh, Wei-Yin (2011). “Classification and regression trees”. In: *WIREs Data Mining and Knowledge Discovery*.

McCarthy, John (1987). “Generality in Artificial Intelligence”. In: *Commun. ACM* 30.12, pp. 1030–1035. issn: 0001-0782. doi: `10.1145/33447.33448`.

Molecular Expressions - Mitochondria (2017). Accessed: 2017-05-31. url: `https://micro.magnet.fsu.edu/cells/mitochondria/mitochondria.html`.

Nagi, Sajid and Dhruba Kr. Bhattacharyya (2013). “Classification of microarray cancer data using ensemble approach”. In: *Network Modeling Analysis in Health Informatics and Bioinformatics* 2.3.

Oxidative Phosphorylation (2017). Accessed: 2017-05-31. url: `http://gadau.lab.asu.edu/research/`.

P, Watkins (2000). “COMT inhibitors and liver toxicity.” In: *Neurology* 11 Suppl 2.

Poli, Riccardo, William B. Langdon, and Nicholas Freitag McPhee (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

Polikar, Robi (2012). “Ensemble Learning”. In: *Ensemble Machine Learning: Methods and Applications*. Ed. by Cha Zhang and Yunqian Ma. Boston, MA: Springer US.

Rasmussen, Carl Edward and Christopher K. I. Williams (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Ratanamahatana, Chotirat Ann and Dimitrios Gunopulos (2002). “Scaling up the naive Bayesian classifier: Using decision trees for feature selection”. In:

Ristow, Michael and Jose M. Cuezva (2009). *Cellular Respiration and Carcinogenesis*. Humana Press.

Roessner, U. (2012). *Metabolomics*. Intech.

Rotella, David (2012). *New Horizons in Predictive Toxicology. Current Status and Application*. Ed. by Alan G E Wilson. RSC Drug Discovery. The Royal Society of Chemistry.

Sa, J. P. Marques de (2001). *Pattern Recognition - Concepts, Methods And Applications*. Springer.

Sammut, Claude and Geoffrey I. Webb (2011). *Encyclopedia of Machine Learning*. 1st. Springer Publishing Company, Incorporated.

Samuel, A. L. (1959). “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM J. Res. Dev.* 3.3.

Saraçli, Sinan, Nurhan Doğan, and İsmet Doğan (2013). “Comparison of hierarchical cluster analysis methods by cophenetic correlation”. In: *Journal of Inequalities and Applications*.

Sardão, Vilma A. and et al Tiago B. Silva (2016). “Developing a novel generation of catechol Omethyltranferase (COMT) inhibitors without mitochondrial liabilities for the therapy of Parkinson’s disease”. In:

Seahorse (2017). *Seahorse XF Glycolysis Stress Test Kit User Guide*.

Segaran, Toby (2007). *Programming Collective Intelligence*. First. O’Reilly.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press.

Silipo, Rosaria et al. (2014). *Seven Techniques for Dimensionality Reduction*.

Silva, T. et al. (2016). "Development of Blood-Brain Barrier Permeable Nitrocatechol-Based Catechol O-Methyltransferase Inhibitors with Reduced Potential for Hepatotoxicity". In: *J. Med. Chem.* 59.16.

Sulaiman, M. A. and J. Labadin (2015). "Feature selection based on mutual information". In: *2015 9th International Conference on IT in Asia (CITA)*.

Sutton, Richard S. and Andrew G. Barto (1998). "Introduction to Reinforcement Learning". In:

Swiss, R. et al. (2013). "Validation of a HTS-amenable assay to detect drug-induced mitochondrial toxicity in the absence and presence of cell death". In: *Toxicol In Vitro* 27.6.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar (2005). *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

The Guardian - Healthcare Network (2016). Accessed: 2017-01-15. url: `https://www.theguardian.com/healthcare-network/2016/mar/30/new-drugs-development-costs-pharma`.

Trac Drug Development Process (2017). Accessed: 2017-05-31. url: `https://www.tracservices.co.uk/wp-content/uploads/2014/09/Trac-Drug-Development-Process-1.jpg`.

Turney, Peter D. (2002). "Types of Cost in Inductive Concept Learning". In: *CoRR* cs.LG/0212034.

University of Florida The ID3 Algorithm (2012). Accessed: 2017-01-15. url: `https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm`.

University of Maryland, Hard Margin SVM (2012). Accessed: 2017-01-15. url: `http://www.cs.umd.edu/~samir/498/SVM.pdf`.

Walde, Sabine Schulte im (2003). "Experiments on the Automatic Induction of German Semantic Verb Classes". Published as AIMS Report 9(2). PhD thesis. Institut fur Maschinelle Sprachverarbeitung, Universitat Stuttgart.

Wallace, K. B. and A. A. Starkov (2000). "Mitochondrial Targets of Drug Toxicity". In: *Annu Rev Pharmacol Toxicol* 40.

Wickham, Hadley (2014). "Tidy Data". In: *Journal of Statistical Software*.

Wikipedia Soft Margin SVM (2008). Accessed: 2017-01-15. url: `https://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`.

WK Labe - Seahorse XFe96 (2017). Accessed: 2017-01-15. url: `http://www.wklab.org/instrument-booking/seahorse-bioanalyzer`.

Wolpert, David H. and R. Waters (1994). "The Relationship between PAC, the Statistical Physics framework, the Bayesian framework, and the VC framework". In: *In*. Addison-Wesley.

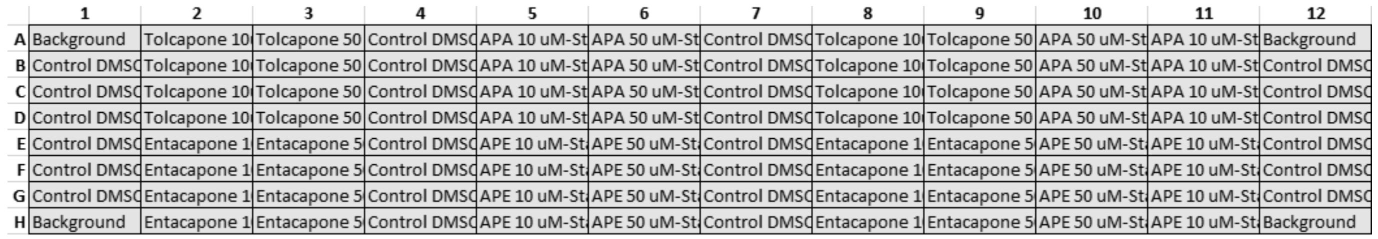
Wordpress Confusion Matrix (2012). Accessed: 2017-01-15. url: `https://uberpython.wordpress.com/2012/01/01/precision-recall-sensitivity-and-specificity`.

WSO2 Class Confusion Matrix (2012). Accessed: 2017-01-15. url: `https://docs.wso2.com/display/ML100/Model+Evaluation+Measures`.

Zhou, Zhi-Hua (2012). *Ensemble Methods: Foundations and Algorithms*. 1st. Chapman & Hall/CRC.

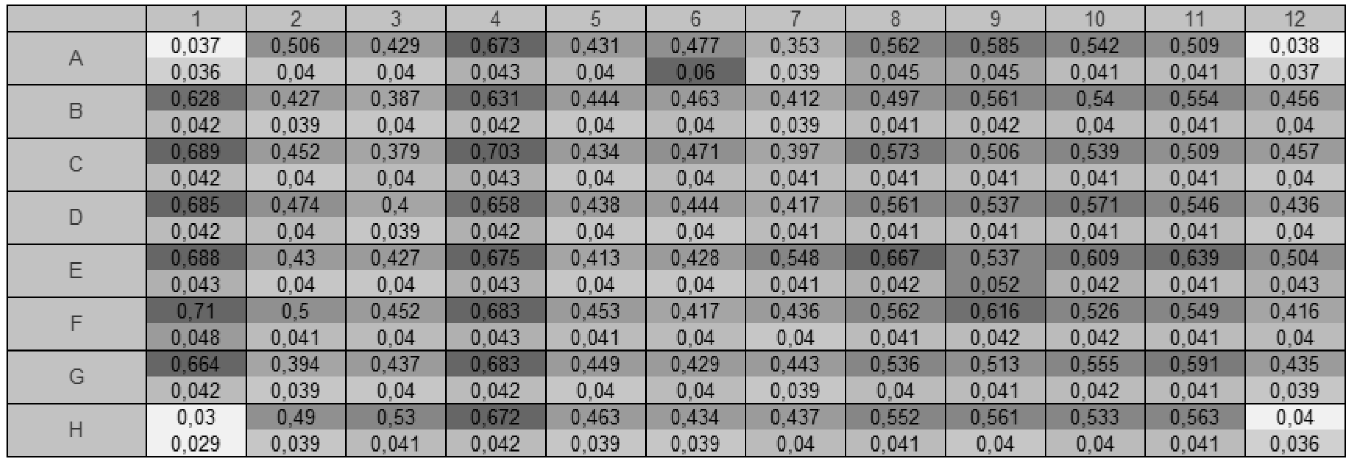
Appendix A

Biological Experiments Results



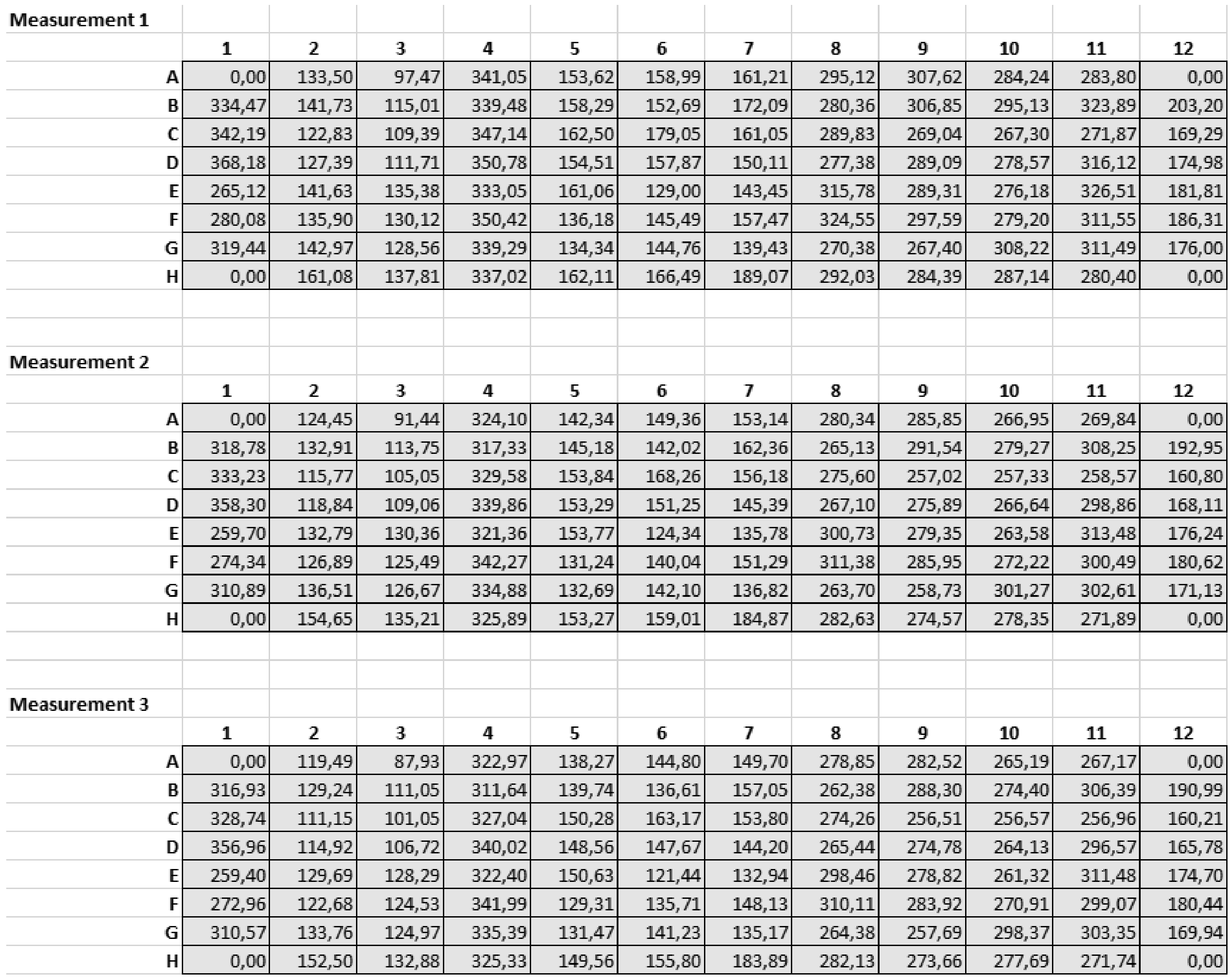
	1	2	3	4	5	6	7	8	9	10	11	12
A	Background	Tolcapone 10	Tolcapone 50	Control DMSO	APA 10 uM-St	APA 50 uM-St	Control DMSO	Tolcapone 10	Tolcapone 50	APA 50 uM-St	APA 10 uM-St	Background
B	Control DMSO	Tolcapone 10	Tolcapone 50	Control DMSO	APA 10 uM-St	APA 50 uM-St	Control DMSO	Tolcapone 10	Tolcapone 50	APA 50 uM-St	APA 10 uM-St	Control DMSO
C	Control DMSO	Tolcapone 10	Tolcapone 50	Control DMSO	APA 10 uM-St	APA 50 uM-St	Control DMSO	Tolcapone 10	Tolcapone 50	APA 50 uM-St	APA 10 uM-St	Control DMSO
D	Control DMSO	Tolcapone 10	Tolcapone 50	Control DMSO	APA 10 uM-St	APA 50 uM-St	Control DMSO	Tolcapone 10	Tolcapone 50	APA 50 uM-St	APA 10 uM-St	Control DMSO
E	Control DMSO	Entacapone 1	Entacapone 5	Control DMSO	APE 10 uM-St	APE 50 uM-St	Control DMSO	Entacapone 1	Entacapone 5	APE 50 uM-St	APE 10 uM-St	Control DMSO
F	Control DMSO	Entacapone 1	Entacapone 5	Control DMSO	APE 10 uM-St	APE 50 uM-St	Control DMSO	Entacapone 1	Entacapone 5	APE 50 uM-St	APE 10 uM-St	Control DMSO
G	Control DMSO	Entacapone 1	Entacapone 5	Control DMSO	APE 10 uM-St	APE 50 uM-St	Control DMSO	Entacapone 1	Entacapone 5	APE 50 uM-St	APE 10 uM-St	Control DMSO
H	Background	Entacapone 1	Entacapone 5	Control DMSO	APE 10 uM-St	APE 50 uM-St	Control DMSO	Entacapone 1	Entacapone 5	APE 50 uM-St	APE 10 uM-St	Background

Figure A.1: Experiment Data File - Plate Wells Description



	1	2	3	4	5	6	7	8	9	10	11	12
A	0,037	0,506	0,429	0,673	0,431	0,477	0,353	0,562	0,585	0,542	0,509	0,038
	0,036	0,04	0,04	0,043	0,04	0,06	0,039	0,045	0,045	0,041	0,041	0,037
B	0,628	0,427	0,387	0,631	0,444	0,463	0,412	0,497	0,561	0,54	0,554	0,456
	0,042	0,039	0,04	0,042	0,04	0,04	0,039	0,041	0,042	0,04	0,041	0,04
C	0,689	0,452	0,379	0,703	0,434	0,471	0,397	0,573	0,506	0,539	0,509	0,457
	0,042	0,04	0,04	0,043	0,04	0,04	0,041	0,041	0,041	0,041	0,041	0,04
D	0,685	0,474	0,4	0,658	0,438	0,444	0,417	0,561	0,537	0,571	0,546	0,436
	0,042	0,04	0,039	0,042	0,04	0,04	0,041	0,041	0,041	0,041	0,041	0,04
E	0,688	0,43	0,427	0,675	0,413	0,428	0,548	0,667	0,537	0,609	0,639	0,504
	0,043	0,04	0,04	0,043	0,04	0,04	0,041	0,042	0,052	0,042	0,041	0,043
F	0,71	0,5	0,452	0,683	0,453	0,417	0,436	0,562	0,616	0,526	0,549	0,416
	0,048	0,041	0,04	0,043	0,041	0,04	0,04	0,041	0,042	0,042	0,041	0,04
G	0,664	0,394	0,437	0,683	0,449	0,429	0,443	0,536	0,513	0,555	0,591	0,435
	0,042	0,039	0,04	0,042	0,04	0,04	0,039	0,04	0,041	0,042	0,041	0,039
H	0,03	0,49	0,53	0,672	0,463	0,434	0,437	0,552	0,561	0,533	0,563	0,04
	0,029	0,039	0,041	0,042	0,039	0,039	0,04	0,041	0,04	0,04	0,041	0,036

Figure A.2: Normalization Data File - Well Cell Concentrations



Measurement 1

	1	2	3	4	5	6	7	8	9	10	11	12
A	0,00	133,50	97,47	341,05	153,62	158,99	161,21	295,12	307,62	284,24	283,80	0,00
B	334,47	141,73	115,01	339,48	158,29	152,69	172,09	280,36	306,85	295,13	323,89	203,20
C	342,19	122,83	109,39	347,14	162,50	179,05	161,05	289,83	269,04	267,30	271,87	169,29
D	368,18	127,39	111,71	350,78	154,51	157,87	150,11	277,38	289,09	278,57	316,12	174,98
E	265,12	141,63	135,38	333,05	161,06	129,00	143,45	315,78	289,31	276,18	326,51	181,81
F	280,08	135,90	130,12	350,42	136,18	145,49	157,47	324,55	297,59	279,20	311,55	186,31
G	319,44	142,97	128,56	339,29	134,34	144,76	139,43	270,38	267,40	308,22	311,49	176,00
H	0,00	161,08	137,81	337,02	162,11	166,49	189,07	292,03	284,39	287,14	280,40	0,00

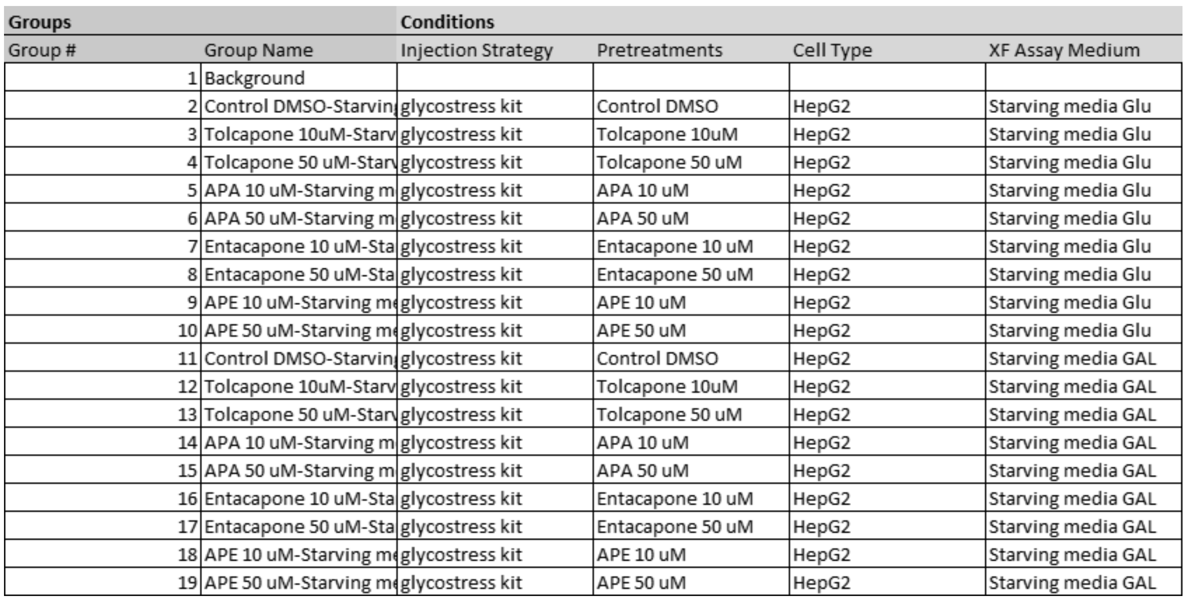
Measurement 2

	1	2	3	4	5	6	7	8	9	10	11	12
A	0,00	124,45	91,44	324,10	142,34	149,36	153,14	280,34	285,85	266,95	269,84	0,00
B	318,78	132,91	113,75	317,33	145,18	142,02	162,36	265,13	291,54	279,27	308,25	192,95
C	333,23	115,77	105,05	329,58	153,84	168,26	156,18	275,60	257,02	257,33	258,57	160,80
D	358,30	118,84	109,06	339,86	153,29	151,25	145,39	267,10	275,89	266,64	298,86	168,11
E	259,70	132,79	130,36	321,36	153,77	124,34	135,78	300,73	279,35	263,58	313,48	176,24
F	274,34	126,89	125,49	342,27	131,24	140,04	151,29	311,38	285,95	272,22	300,49	180,62
G	310,89	136,51	126,67	334,88	132,69	142,10	136,82	263,70	258,73	301,27	302,61	171,13
H	0,00	154,65	135,21	325,89	153,27	159,01	184,87	282,63	274,57	278,35	271,89	0,00

Measurement 3

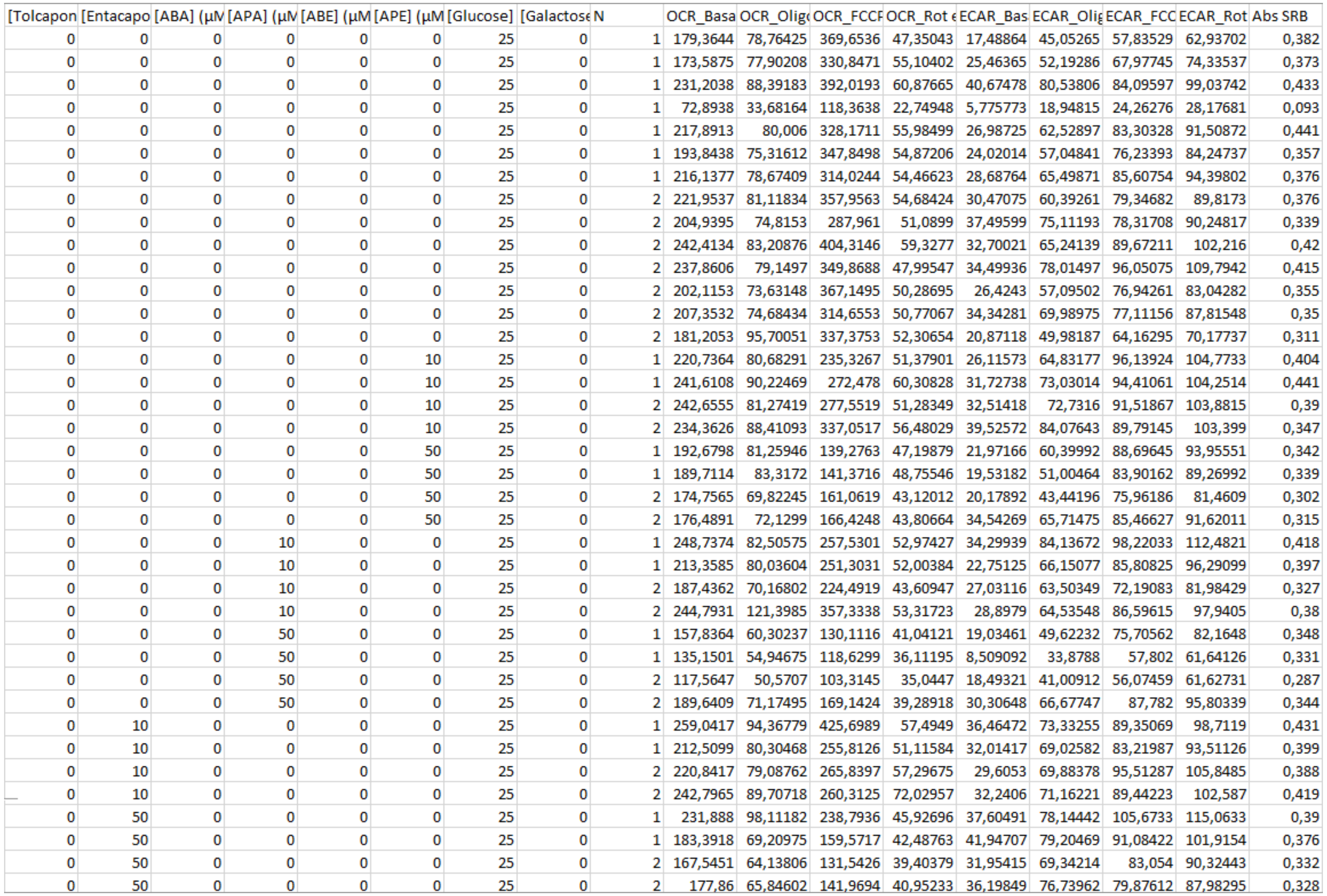
	1	2	3	4	5	6	7	8	9	10	11	12
A	0,00	119,49	87,93	322,97	138,27	144,80	149,70	278,85	282,52	265,19	267,17	0,00
B	316,93	129,24	111,05	311,64	139,74	136,61	157,05	262,38	288,30	274,40	306,39	190,99
C	328,74	111,15	101,05	327,04	150,28	163,17	153,80	274,26	256,51	256,57	256,96	160,21
D	356,96	114,92	106,72	340,02	148,56	147,67	144,20	265,44	274,78	264,13	296,57	165,78
E	259,40	129,69	128,29	322,40	150,63	121,44	132,94	298,46	278,82	261,32	311,48	174,70
F	272,96	122,68	124,53	341,99	129,31	135,71	148,13	310,11	283,92	270,91	299,07	180,44
G	310,57	133,76	124,97	335,39	131,47	141,23	135,17	264,38	257,69	298,37	303,35	169,94
H	0,00	152,50	132,88	325,33	149,56	155,80	183,89	282,13	273,66	277,69	271,74	0,00

Figure A.3: Experiment Data File - First 3 Measurements for OCR (Similar tables exist for ECAR



Groups		Conditions			
Group #	Group Name	Injection Strategy	Pretreatments	Cell Type	XF Assay Medium
1	Background				
2	Control DMSO-Starvin	glycostress kit	Control DMSO	HepG2	Starving media Glu
3	Tolcapone 10uM-Starv	glycostress kit	Tolcapone 10uM	HepG2	Starving media Glu
4	Tolcapone 50 uM-Starv	glycostress kit	Tolcapone 50 uM	HepG2	Starving media Glu
5	APA 10 uM-Starving m	glycostress kit	APA 10 uM	HepG2	Starving media Glu
6	APA 50 uM-Starving m	glycostress kit	APA 50 uM	HepG2	Starving media Glu
7	Entacapone 10 uM-Sta	glycostress kit	Entacapone 10 uM	HepG2	Starving media Glu
8	Entacapone 50 uM-Sta	glycostress kit	Entacapone 50 uM	HepG2	Starving media Glu
9	APE 10 uM-Starving m	glycostress kit	APE 10 uM	HepG2	Starving media Glu
10	APE 50 uM-Starving m	glycostress kit	APE 50 uM	HepG2	Starving media Glu
11	Control DMSO-Starvin	glycostress kit	Control DMSO	HepG2	Starving media GAL
12	Tolcapone 10uM-Starv	glycostress kit	Tolcapone 10uM	HepG2	Starving media GAL
13	Tolcapone 50 uM-Starv	glycostress kit	Tolcapone 50 uM	HepG2	Starving media GAL
14	APA 10 uM-Starving m	glycostress kit	APA 10 uM	HepG2	Starving media GAL
15	APA 50 uM-Starving m	glycostress kit	APA 50 uM	HepG2	Starving media GAL
16	Entacapone 10 uM-Sta	glycostress kit	Entacapone 10 uM	HepG2	Starving media GAL
17	Entacapone 50 uM-Sta	glycostress kit	Entacapone 50 uM	HepG2	Starving media GAL
18	APE 10 uM-Starving m	glycostress kit	APE 10 uM	HepG2	Starving media GAL
19	APE 50 uM-Starving m	glycostress kit	APE 50 uM	HepG2	Starving media GAL

Figure A.4: Experiment Data File - Initial Conditions Description



[Tolcapon	[Entacapo	[ABA] (µM	[APA] (µM	[ABE] (µM	[APE] (µM	[Glucose]	[Galactose	N	OCR_Basa	OCR_Olig	OCR_FCCF	OCR_Rot	ECAR_Bas	ECAR_Olig	ECAR_FCC	ECAR_Rot	Abs SRB
0	0	0	0	0	0	25	0	1	179,3644	78,76425	369,6536	47,35043	17,48864	45,05265	57,83529	62,93702	0,382
0	0	0	0	0	0	25	0	1	173,5875	77,90208	330,8471	55,10402	25,46365	52,19286	67,97745	74,33537	0,373
0	0	0	0	0	0	25	0	1	231,2038	88,39183	392,0193	60,87665	40,67478	80,53806	84,09597	99,03742	0,433
0	0	0	0	0	0	25	0	1	72,8938	33,68164	118,3638	22,74948	5,775773	18,94815	24,26276	28,17681	0,093
0	0	0	0	0	0	25	0	1	217,8913	80,006	328,1711	55,98499	26,98725	62,52897	83,30328	91,50872	0,441
0	0	0	0	0	0	25	0	1	193,8438	75,31612	347,8498	54,87206	24,02014	57,04841	76,23393	84,24737	0,357
0	0	0	0	0	0	25	0	1	216,1377	78,67409	314,0244	54,46623	28,68764	65,49871	85,60754	94,39802	0,376
0	0	0	0	0	0	25	0	2	221,9537	81,11834	357,9563	54,68424	30,47075	60,39261	79,34682	89,8173	0,376
0	0	0	0	0	0	25	0	2	204,9395	74,8153	287,961	51,0899	37,49599	75,11193	78,31708	90,24817	0,339
0	0	0	0	0	0	25	0	2	242,4134	83,20876	404,3146	59,3277	32,70021	65,24139	89,67211	102,216	0,42
0	0	0	0	0	0	25	0	2	237,8606	79,1497	349,8688	47,99547	34,49936	78,01497	96,05075	109,7942	0,415
0	0	0	0	0	0	25	0	2	202,1153	73,63148	367,1495	50,28695	26,4243	57,09502	76,94261	83,04282	0,355
0	0	0	0	0	0	25	0	2	207,3532	74,68434	314,6553	50,77067	34,34281	69,98975	77,11156	87,81548	0,35
0	0	0	0	0	0	25	0	2	181,2053	95,70051	337,3753	52,30654	20,87118	49,98187	64,16295	70,17737	0,311
0	0	0	0	0	10	25	0	1	220,7364	80,68291	235,3267	51,37901	26,11573	64,83177	96,13924	104,7733	0,404
0	0	0	0	0	10	25	0	1	241,6108	90,22469	272,478	60,30828	31,72738	73,03014	94,41061	104,2514	0,441
0	0	0	0	0	10	25	0	2	242,6555	81,27419	277,5519	51,28349	32,51418	72,7316	91,51867	103,8815	0,39
0	0	0	0	0	10	25	0	2	234,3626	88,41093	337,0517	56,48029	39,52572	84,07643	89,79145	103,399	0,347
0	0	0	0	0	50	25	0	1	192,6798	81,25946	139,2763	47,19879	21,97166	60,39992	88,69645	93,95551	0,342
0	0	0	0	0	50	25	0	1	189,7114	83,3172	141,3716	48,75546	19,53182	51,00464	83,90162	89,26992	0,339
0	0	0	0	0	50	25	0	2	174,7565	69,82245	161,0619	43,12012	20,17892	43,44196	75,96186	81,4609	0,302
0	0	0	0	0	50	25	0	2	176,4891	72,1299	166,4248	43,80664	34,54269	65,71475	85,46627	91,62011	0,315
0	0	0	10	0	0	25	0	1	248,7374	82,50575	257,5301	52,97427	34,29939	84,13672	98,22033	112,4821	0,418
0	0	0	10	0	0	25	0	1	213,3585	80,03604	251,3031	52,00384	22,75125	66,15077	85,80825	96,29099	0,397
0	0	0	10	0	0	25	0	2	187,4362	70,16802	224,4919	43,60947	27,03116	63,50349	72,19083	81,98429	0,327
0	0	0	10	0	0	25	0	2	244,7931	121,3985	357,3338	53,31723	28,8979	64,53548	86,59615	97,9405	0,38
0	0	0	50	0	0	25	0	1	157,8364	60,30237	130,1116	41,04121	19,03461	49,62232	75,70562	82,1648	0,348
0	0	0	50	0	0	25	0	1	135,1501	54,94675	118,6299	36,11195	8,509092	33,8788	57,802	61,64126	0,331
0	0	0	50	0	0	25	0	2	117,5647	50,5707	103,3145	35,0447	18,49321	41,00912	56,07459	61,62731	0,287
0	0	0	50	0	0	25	0	2	189,6409	71,17495	169,1424	39,28918	30,30648	66,67747	87,782	95,80339	0,344
0	10	0	0	0	0	25	0	1	259,0417	94,36779	425,6989	57,4949	36,46472	73,33255	89,35069	98,7119	0,431
0	10	0	0	0	0	25	0	1	212,5099	80,30468	255,8126	51,11584	32,01417	69,02582	83,21987	93,51126	0,399
0	10	0	0	0	0	25	0	2	220,8417	79,08762	265,8397	57,29675	29,6053	69,88378	95,51287	105,8485	0,388
0	10	0	0	0	0	25	0	2	242,7965	89,70718	260,3125	72,02957	32,2406	71,16221	89,44223	102,587	0,419
0	50	0	0	0	0	25	0	1	231,888	98,11182	238,7936	45,92696	37,60491	78,14442	105,6733	115,0633	0,39
0	50	0	0	0	0	25	0	1	183,3918	69,20975	159,5717	42,48763	41,94707	79,20469	91,08422	101,9154	0,376
0	50	0	0	0	0	25	0	2	167,5451	64,13806	131,5426	39,40379	31,95415	69,34214	83,054	90,32443	0,332
0	50	0	0	0	0	25	0	2	177,86	65,84602	141,9694	40,95233	36,19849	76,73962	79,87612	87,98295	0,328

Figure A.5: Parsed Data File - Results Table for ML

Appendix B

Machine Learning Results

B.1 Clustering

B.1.1 Birch

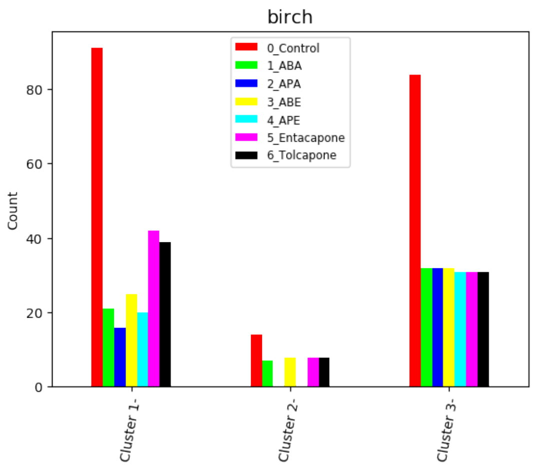


Figure B.1: Birch *Complete* CD 3 Clusters

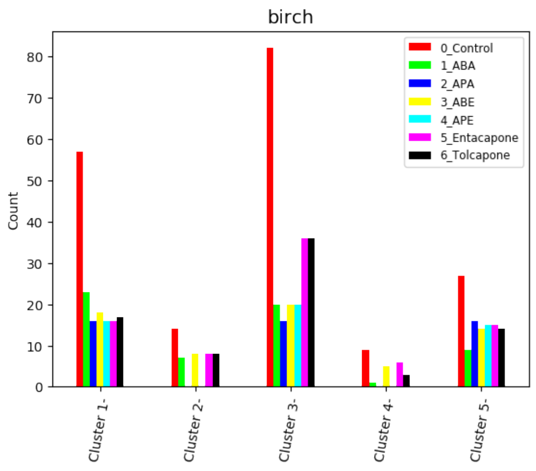


Figure B.2: Birch *Complete* CD 5 Clusters

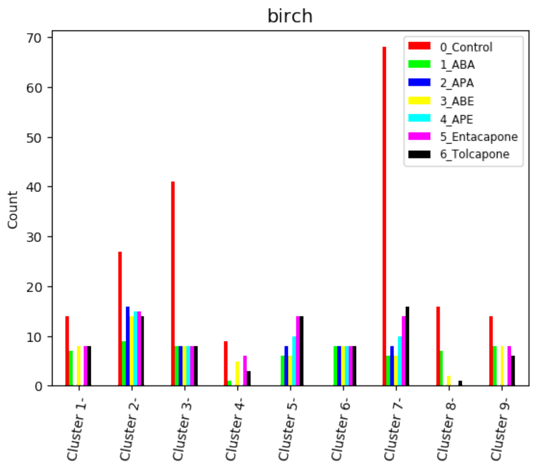


Figure B.3: Birch *Complete* CD 9 Clusters

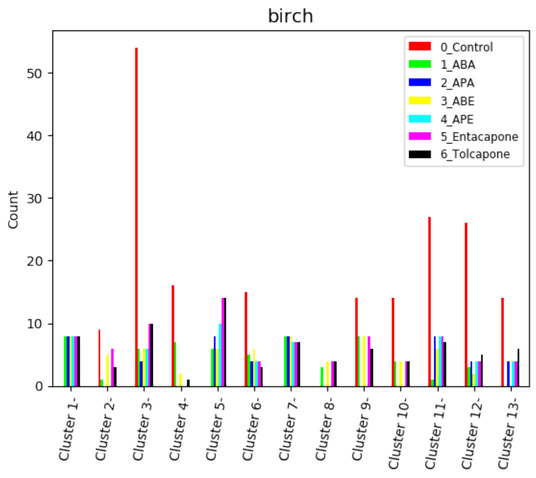


Figure B.4: Birch *Complete* CD 13 Clusters

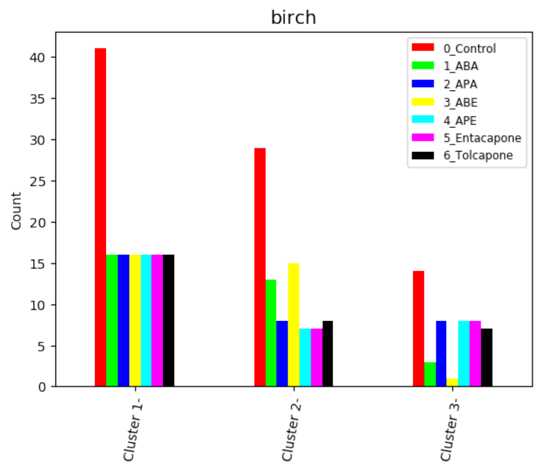


Figure B.5: Birch *Mito* CD 3 Clusters

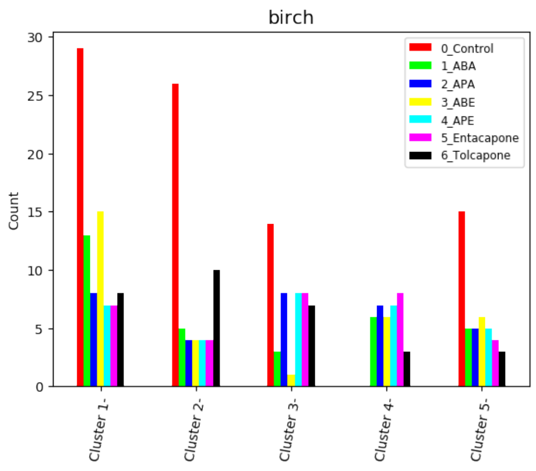


Figure B.6: Birch *Mito* CD 5 Clusters

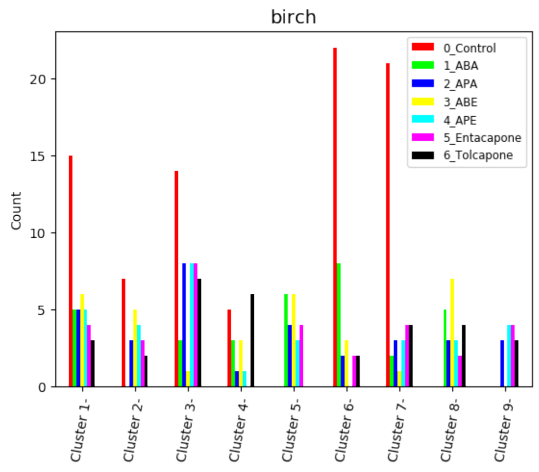


Figure B.7: Birch *Mito* CD 9 Clusters

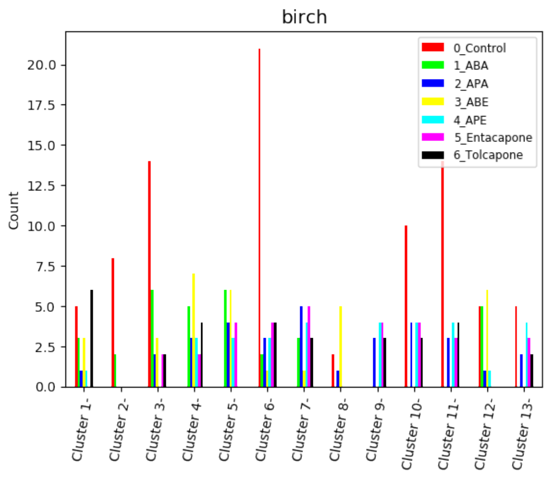


Figure B.8: Birch *Mito* CD 13 Clusters

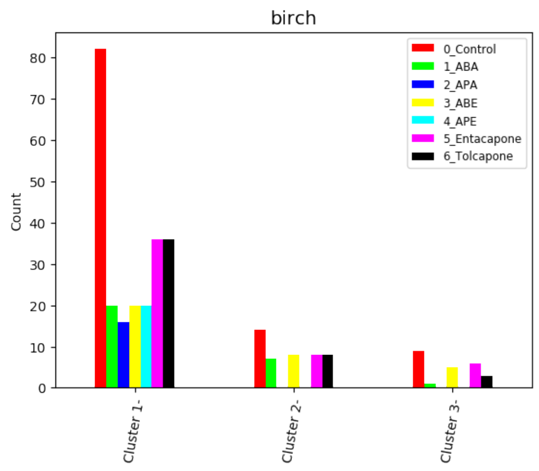


Figure B.9: Birch *Glycolysis* CD 3 Clusters

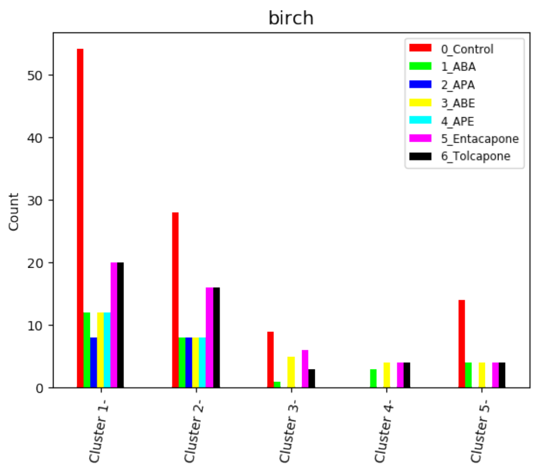


Figure B.10: Birch *Glycolysis* CD 5 Clusters

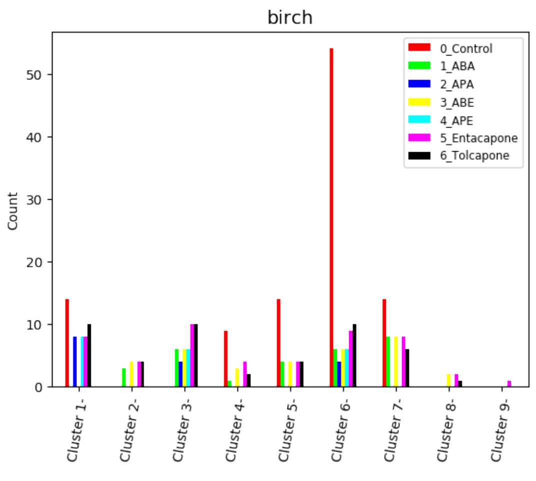


Figure B.11: Birch *Glycolysis* CD 9 Clusters

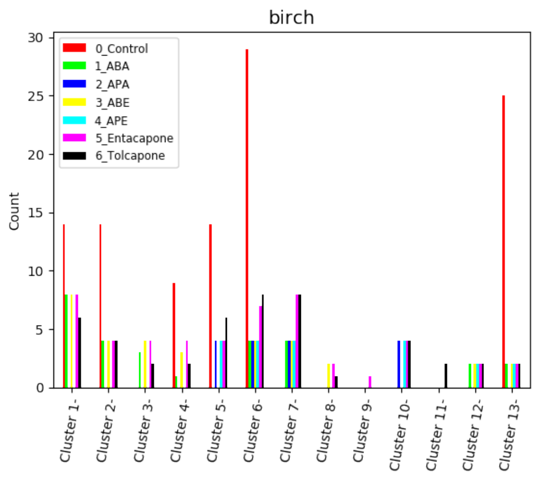


Figure B.12: Birch *Glycolysis* CD 13 Clusters

B.1.2 Gaussian Mixture

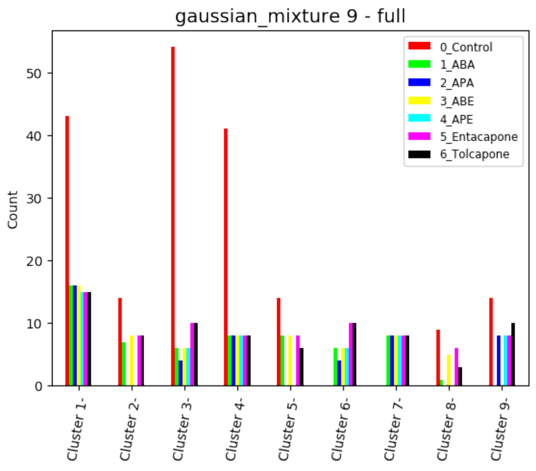


Figure B.13: Gaussian Mixture *Complete* CD

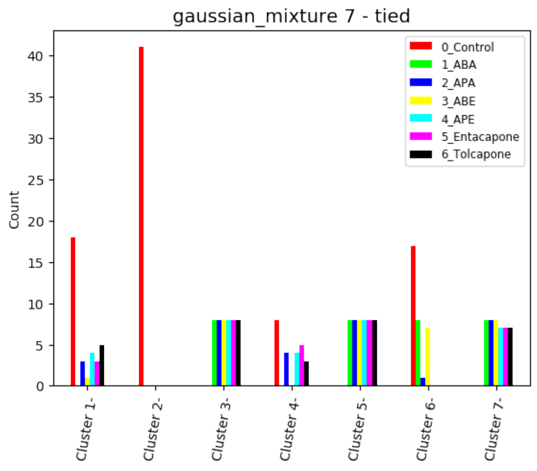


Figure B.14: Gaussian Mixture *Mito* CD

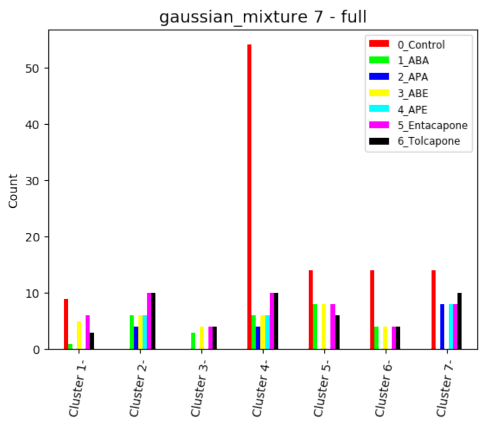


Figure B.15: Gaussian Mixture *Glycolysis* CD

B.1.3 *k*-Means

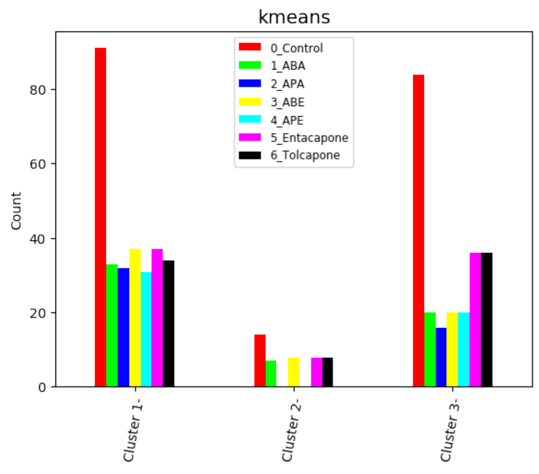


Figure B.16: *k*-Means *Complete* CD 3 Clusters

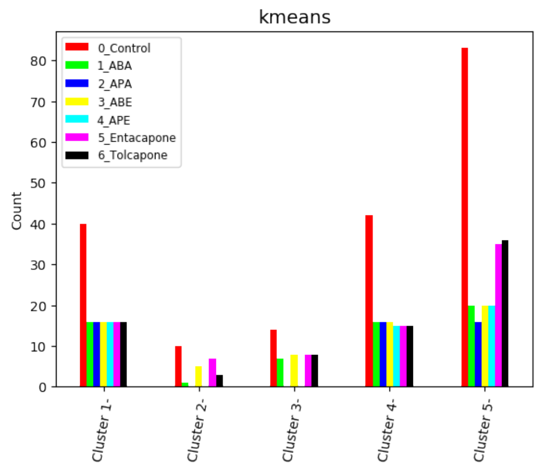


Figure B.17: *k*-Means *Complete* CD 5 Clusters

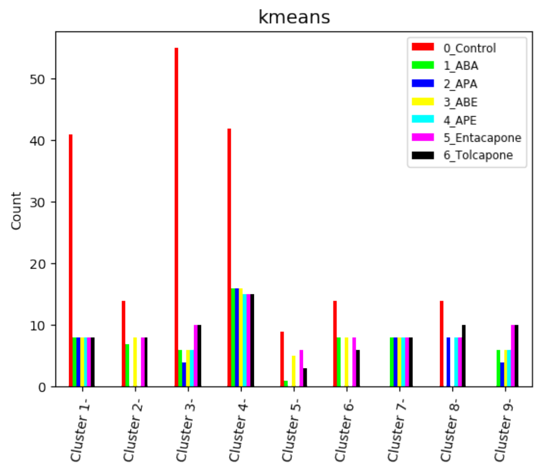


Figure B.18: *k*-Means *Complete* CD 9 Clusters

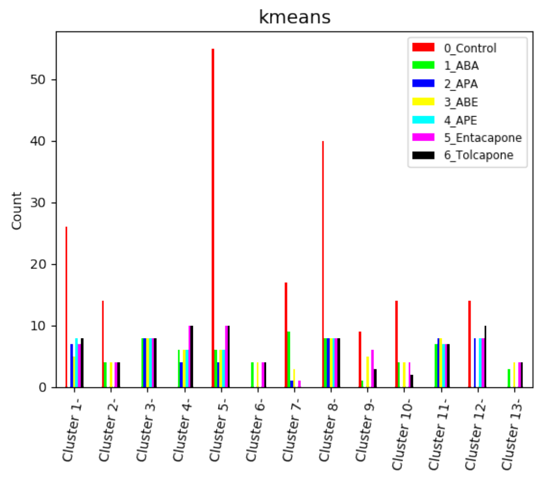


Figure B.19: *k*-Means *Complete* CD 13 Clusters

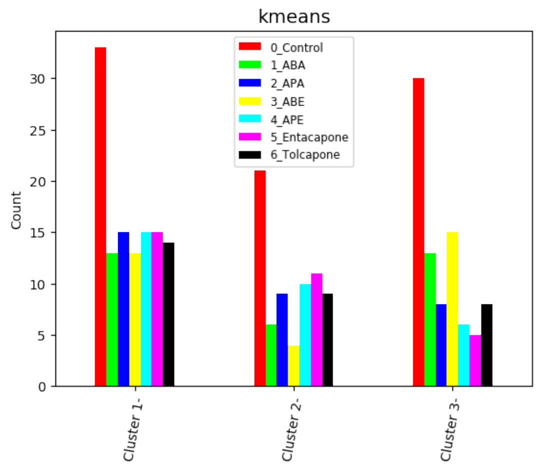


Figure B.20: *k*-Means *Mito* CD 3 Clusters

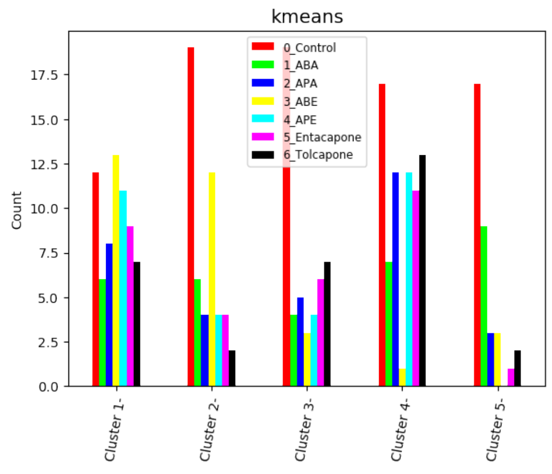


Figure B.21: *k*-Means *Mito* CD 5 Clusters

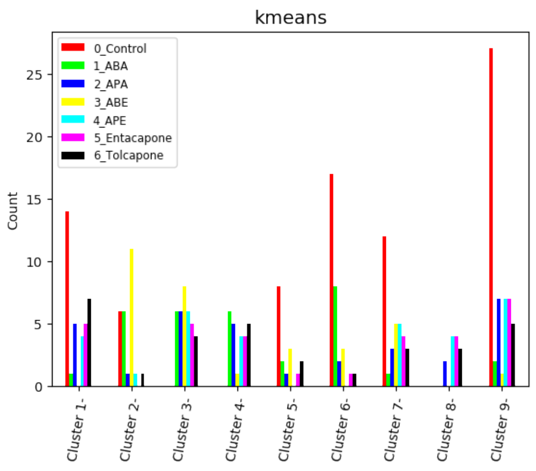


Figure B.22: *k*-Means *Mito* CD 9 Clusters

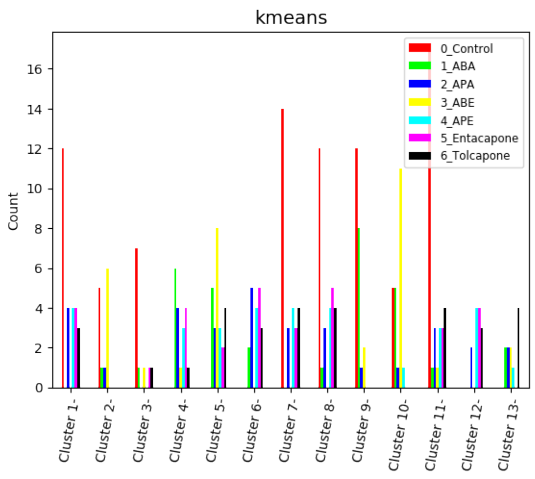


Figure B.23: *k*-Means *Mito* CD 13 Clusters

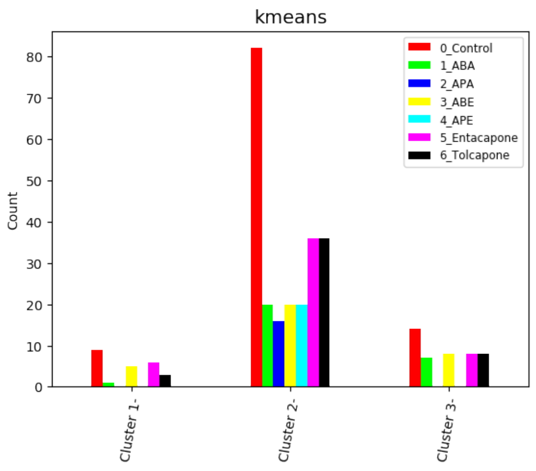


Figure B.24: *k*-Means *Glycolysis* CD 3 Clusters

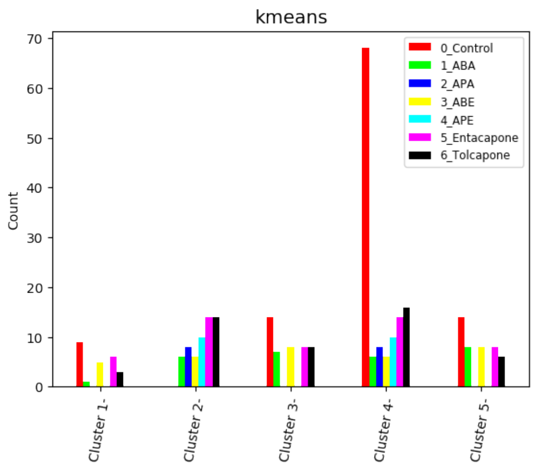


Figure B.25: *k*-Means *Glycolysis* CD 5 Clusters

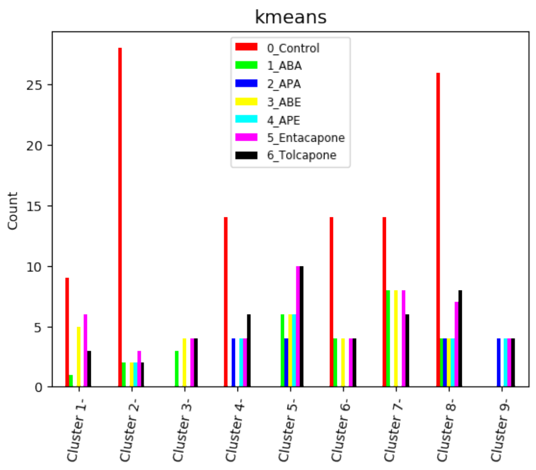


Figure B.26: *k*-Means *Glycolysis* CD 9 Clusters

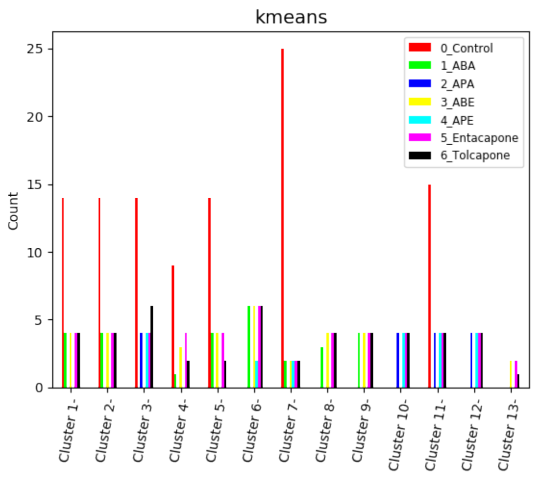


Figure B.27: *k*-Means *Glycolysis* CD 13 Clusters

B.1.4 Hierarchical Clustering (HC)

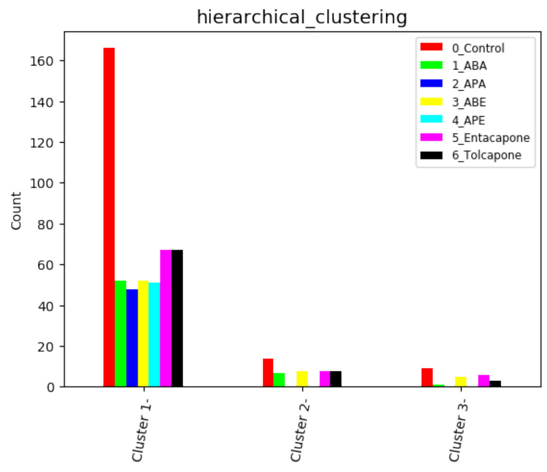


Figure B.28: HC *Complete* CD 3 Clusters

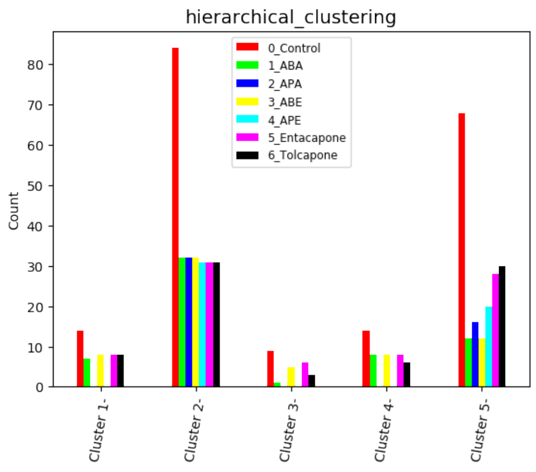


Figure B.29: HC *Complete* CD 5 Clusters

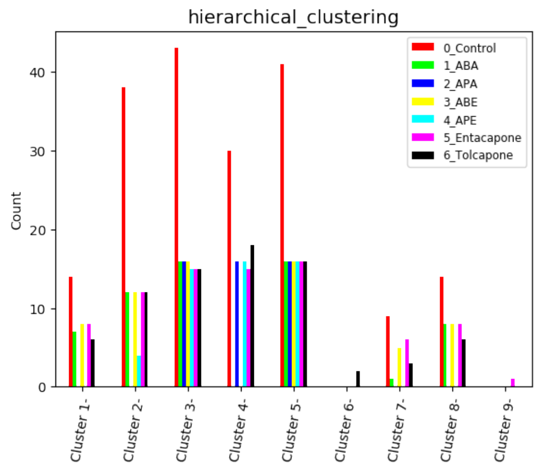


Figure B.30: HC *Complete* CD 9 Clusters

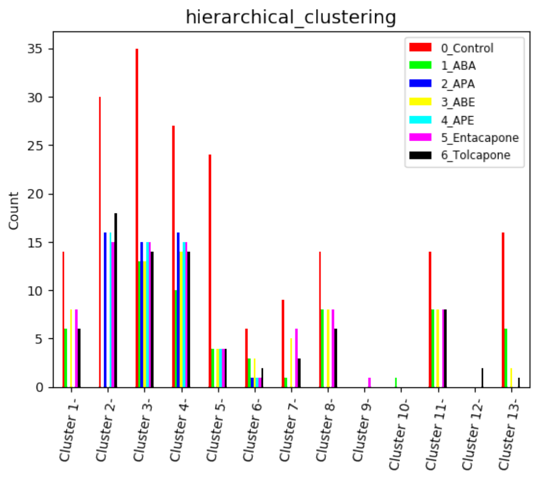


Figure B.31: HC *Complete* CD 13 Clusters

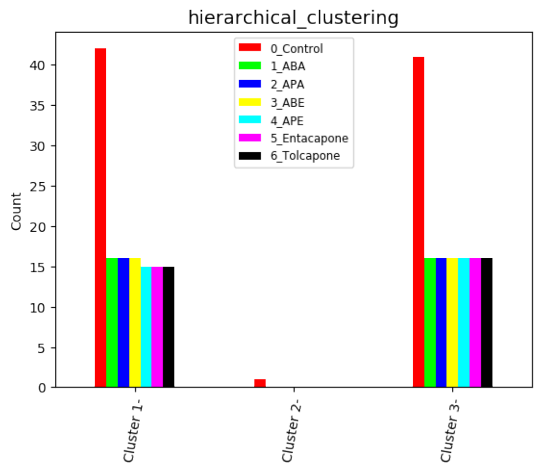


Figure B.32: HC *Mito* CD 3 Clusters

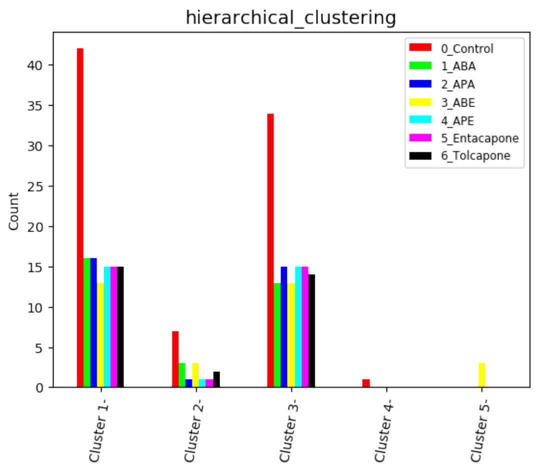


Figure B.33: HC *Mito* CD 5 Clusters

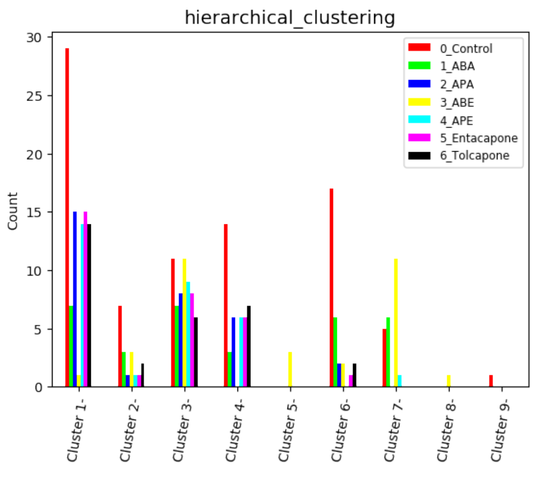


Figure B.34: HC *Mito* CD 9 Clusters

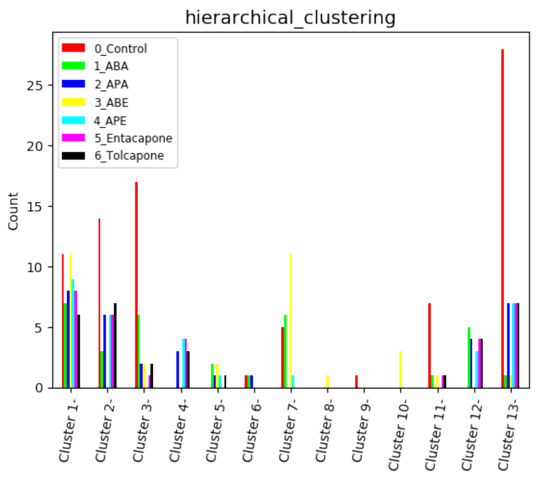


Figure B.35: HC *Mito* CD 13 Clusters

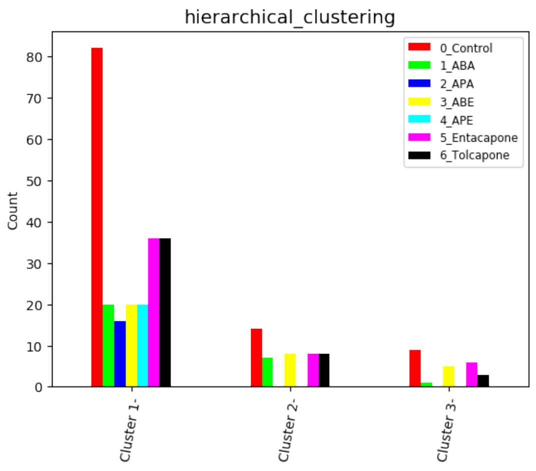


Figure B.36: HC *Glycolysis* CD 3 Clusters

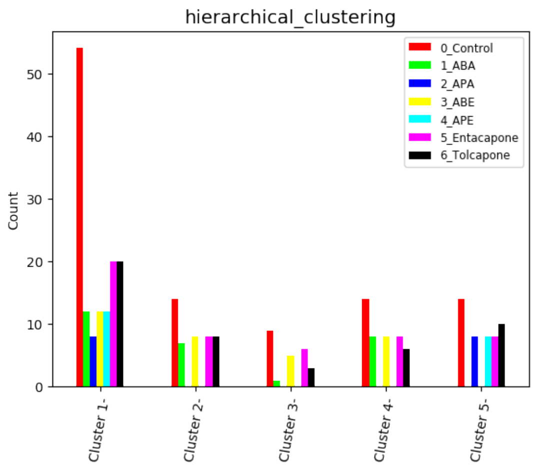


Figure B.37: HC *Glycolysis* CD 5 Clusters

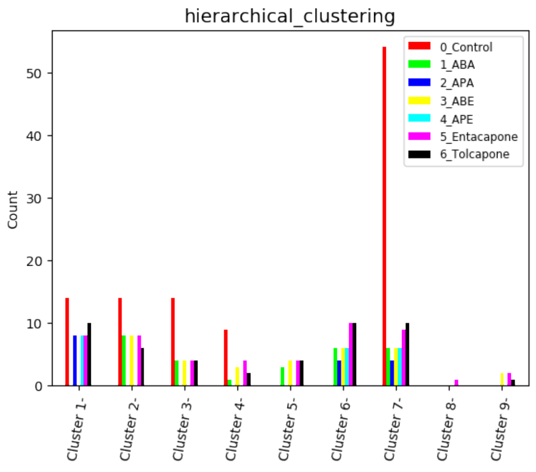


Figure B.38: HC *Glycolysis* CD 9 Clusters

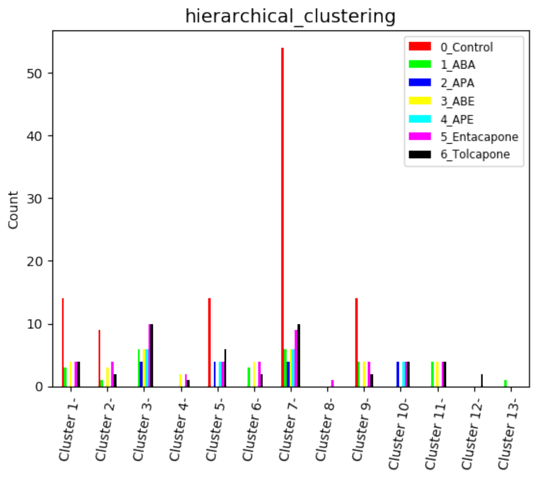


Figure B.39: HC *Glycolysis* CD 13 Clusters

B.1.5 Density-based spatial clustering of applications with noise (DBSCAN)

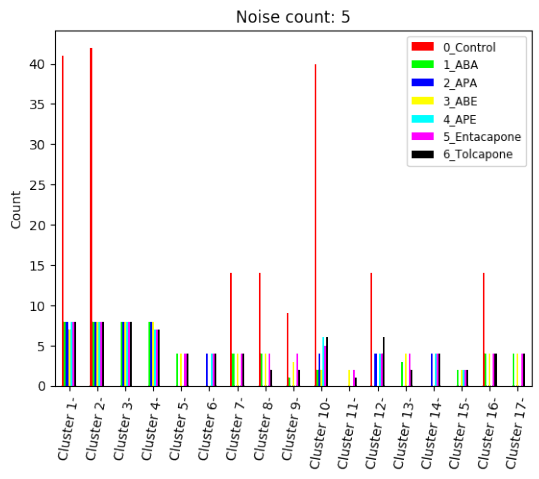


Figure B.40: DBSCAN *Complete* CD $minPts = 2, \epsilon = 0.9$

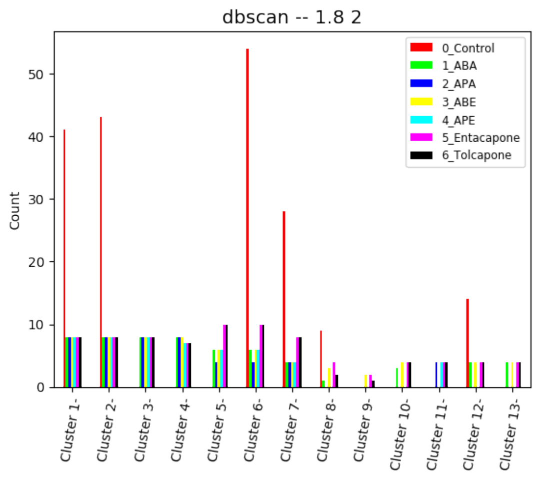


Figure B.41: DBSCAN *Complete* CD $minPts = 2, \epsilon = 1.8$

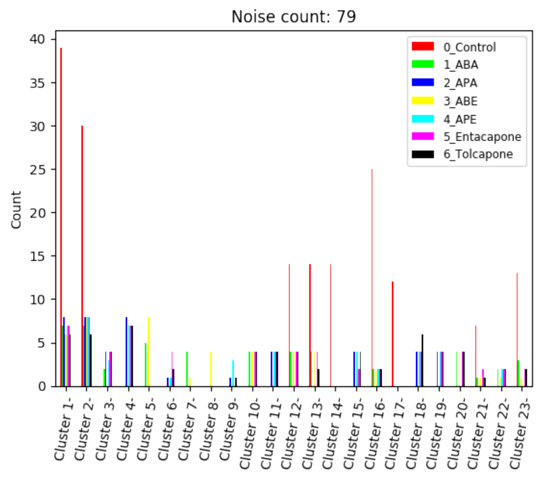


Figure B.42: DBSCAN *Complete* CD $minPts = 4, \epsilon = 0.6$

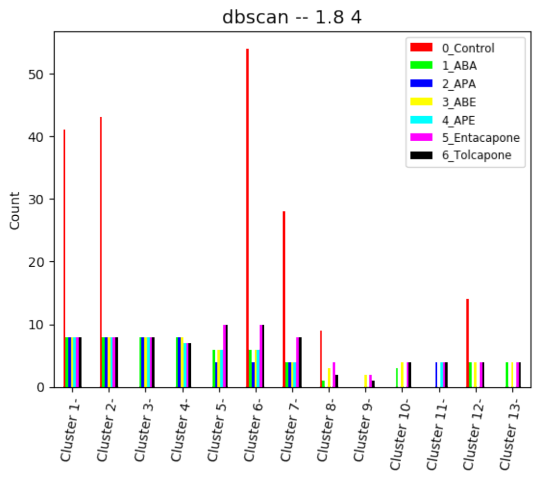


Figure B.43: DBSCAN *Complete* CD $minPts = 4, \epsilon = 1.8$

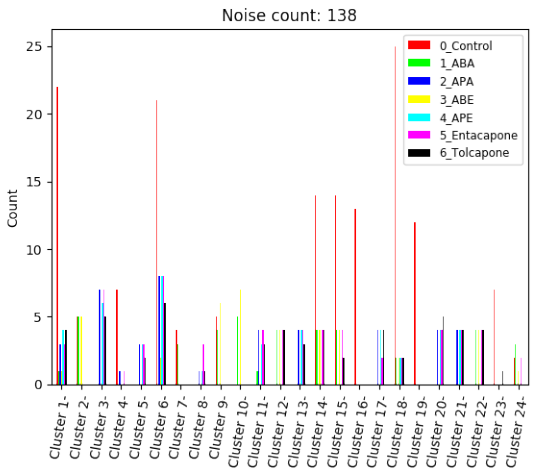


Figure B.44: DBSCAN *Complete* CD $minPts = 6, \epsilon = 0.6$

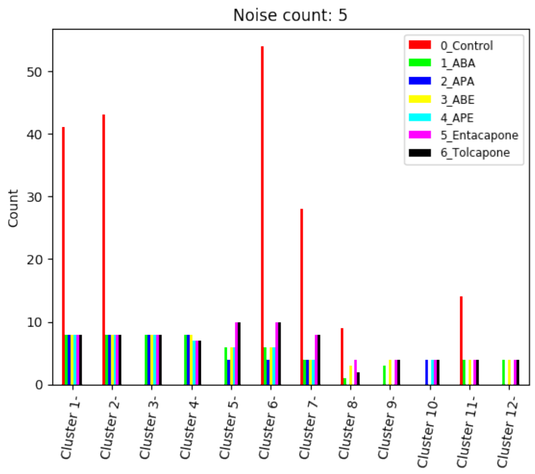


Figure B.45: DBSCAN *Complete* CD $minPts = 6, \epsilon = 1.8$

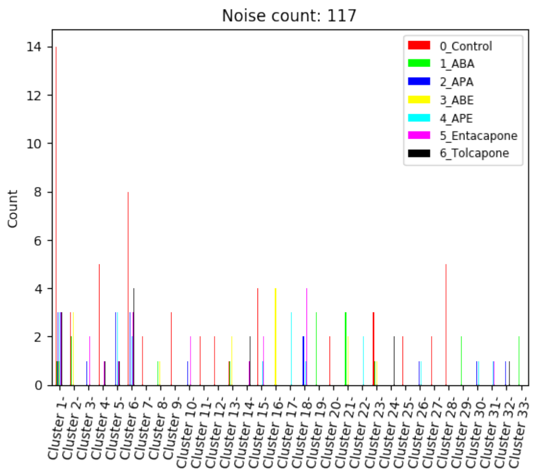


Figure B.46: DBSCAN *Mito* CD $minPts = 2, \epsilon = 0.9$

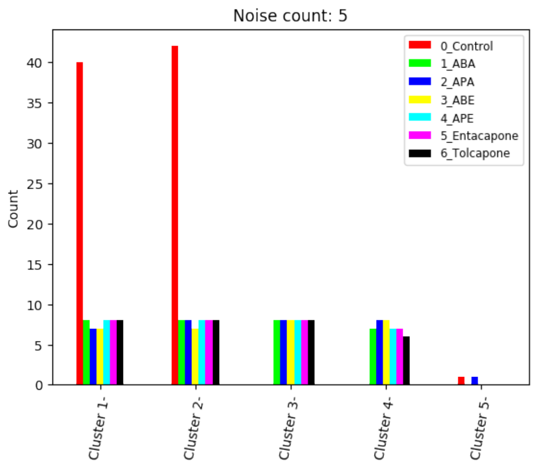


Figure B.47: DBSCAN *Mito* CD $minPts = 2, \epsilon = 1.8$

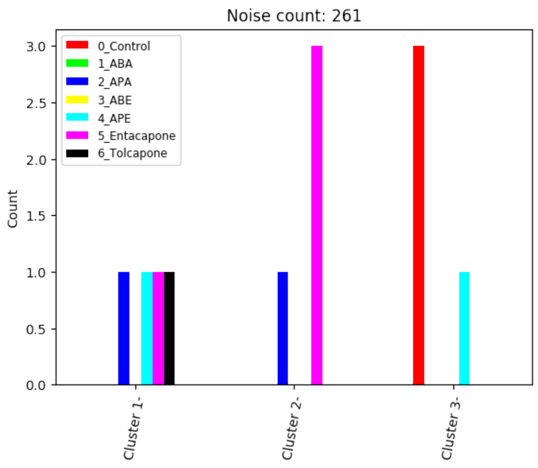


Figure B.48: DBSCAN *Mito* CD $minPts = 4, \epsilon = 0.6$

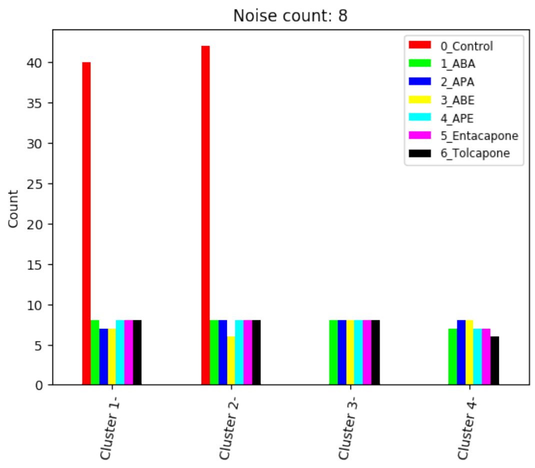


Figure B.49: DBSCAN *Mito* CD $minPts = 4, \epsilon = 1.8$

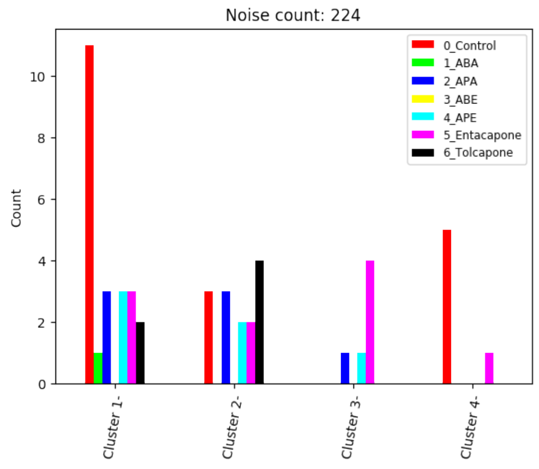


Figure B.50: DBSCAN *Mito* CD $minPts = 6, \epsilon = 0.6$

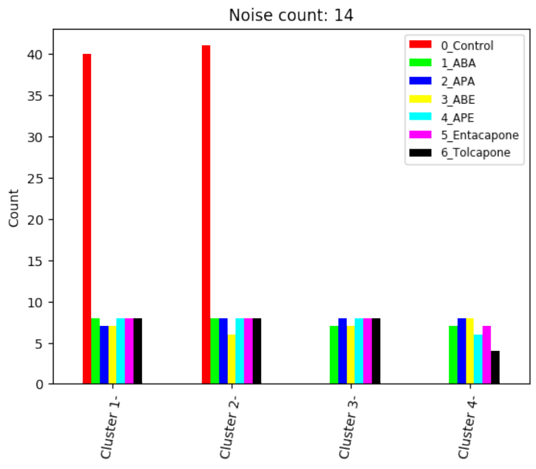


Figure B.51: DBSCAN *Mito* CD $minPts = 6, \epsilon = 1.8$

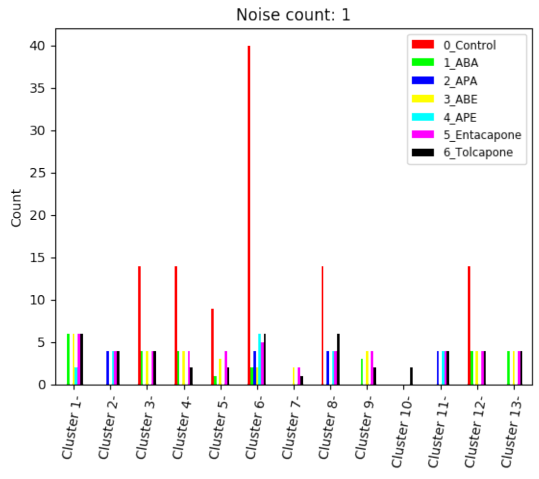


Figure B.52: DBSCAN *Glycolysis* CD $minPts = 2, \epsilon = 0.9$

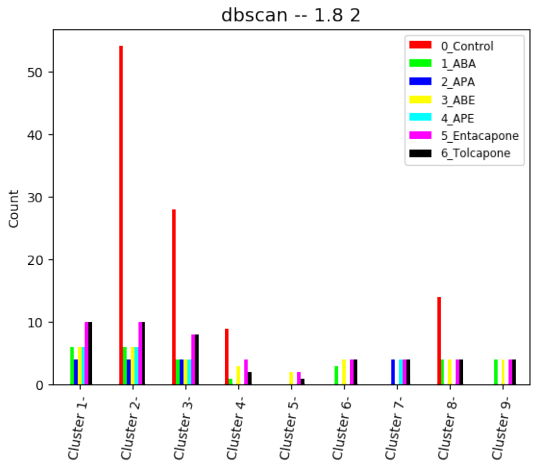


Figure B.53: DBSCAN *Glycolysis* CD $minPts = 2, \epsilon = 1.8$

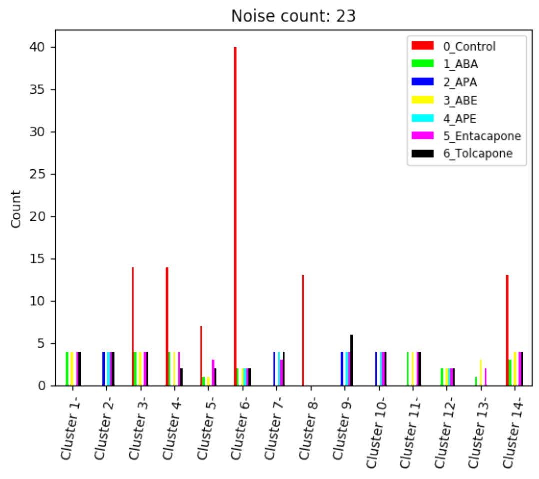


Figure B.54: DBSCAN *Glycolysis* CD $minPts = 4, \epsilon = 0.6$

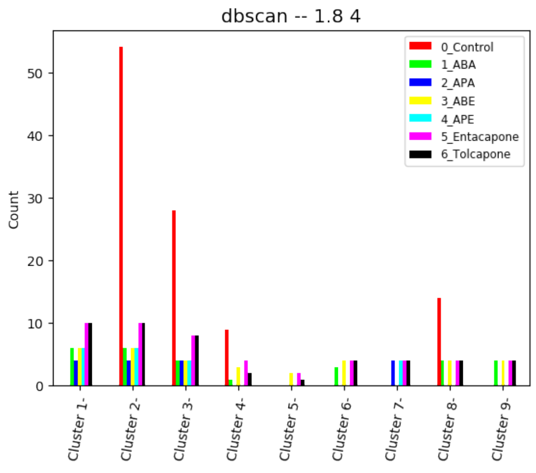


Figure B.55: DBSCAN *Glycolysis* CD $minPts = 4, \epsilon = 1.8$

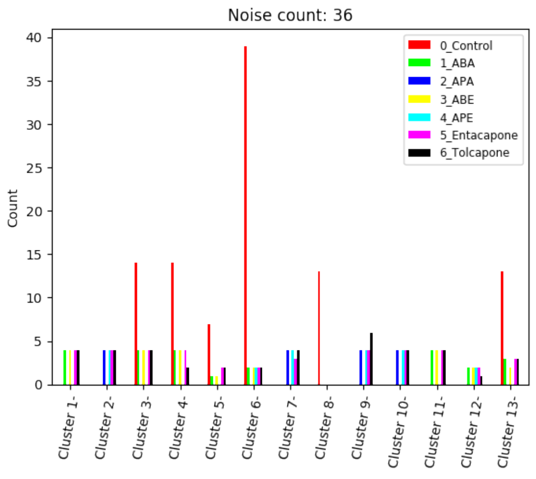


Figure B.56: DBSCAN *Glycolysis* CD $minPts = 6, \epsilon = 0.6$

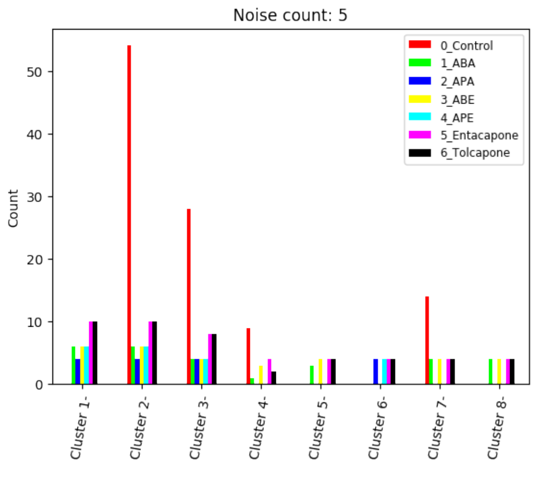


Figure B.57: DBSCAN *Glycolysis* CD $minPts = 6, \epsilon = 1.8$

B.1.6 Clustering Using REpresentatives (CURE)

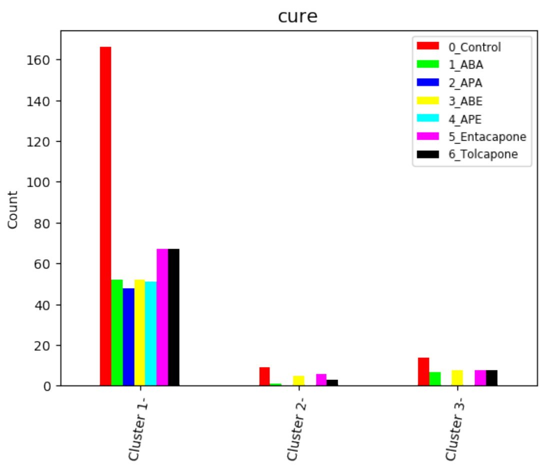


Figure B.58: CURE *Complete* CD 3 Clusters

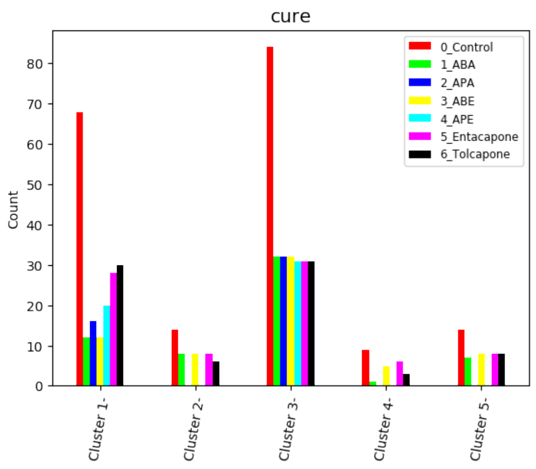


Figure B.59: CURE *Complete* CD 5 Clusters

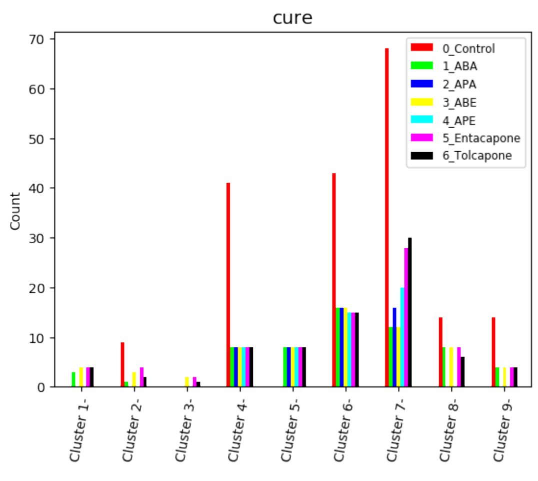


Figure B.60: CURE *Complete* CD 9 Clusters

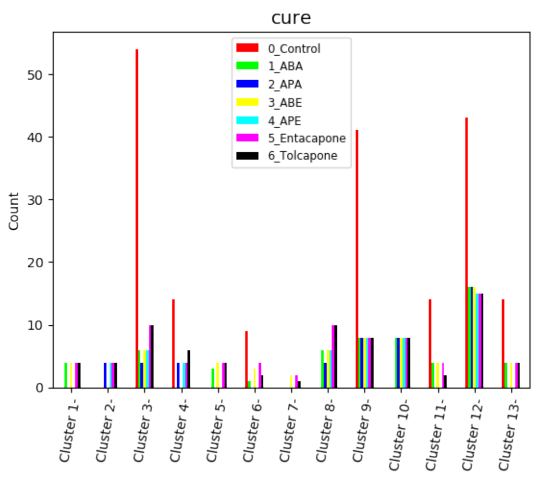


Figure B.61: CURE *Complete* CD 13 Clusters

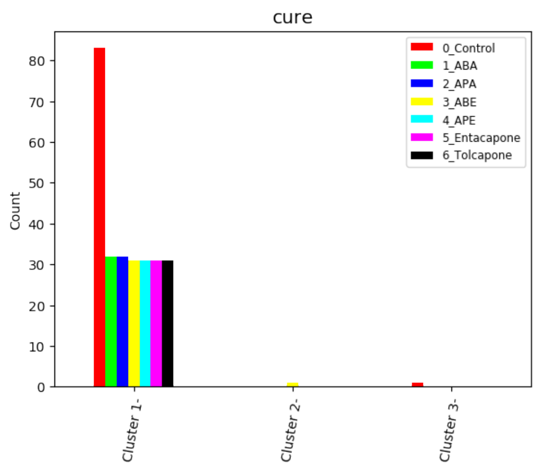


Figure B.62: CURE *Mito* CD 3 Clusters

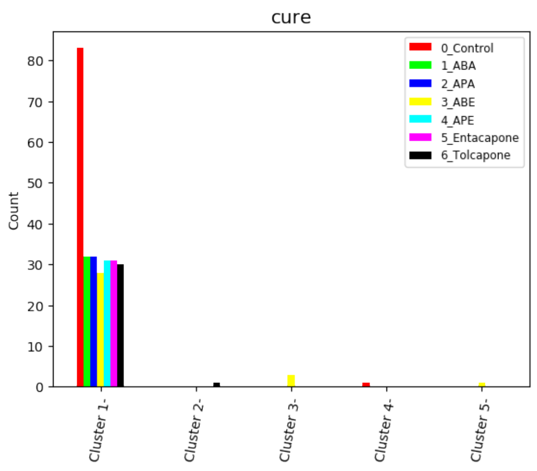


Figure B.63: CURE *Mito* CD 5 Clusters

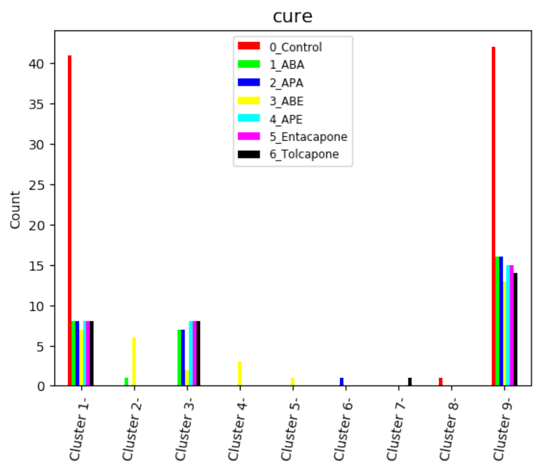


Figure B.64: CURE *Mito* CD 9 Clusters

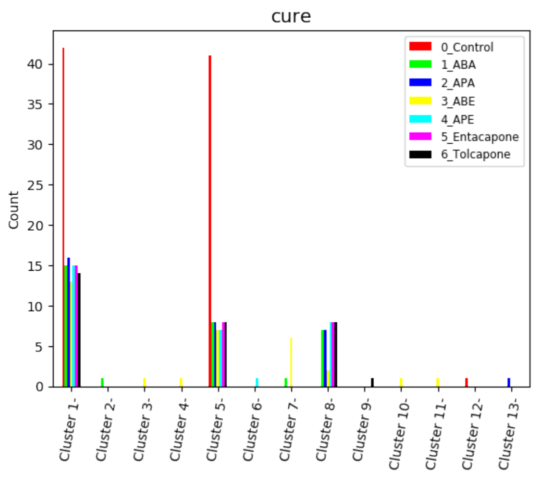


Figure B.65: CURE *Mito* CD 13 Clusters

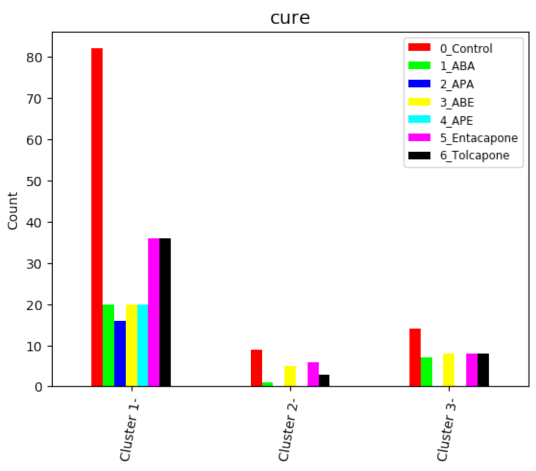


Figure B.66: CURE *Glycolysis* CD 3 Clusters

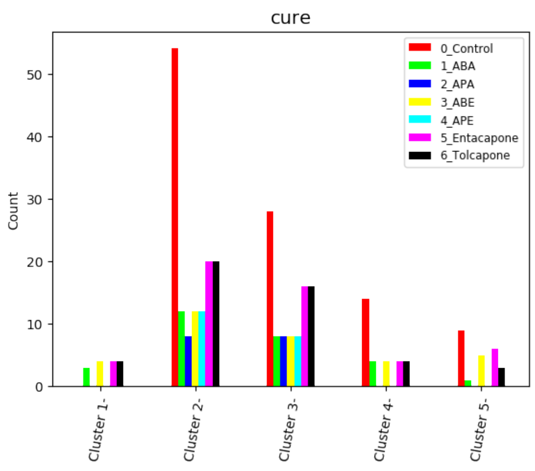


Figure B.67: CURE *Glycolysis* CD 5 Clusters

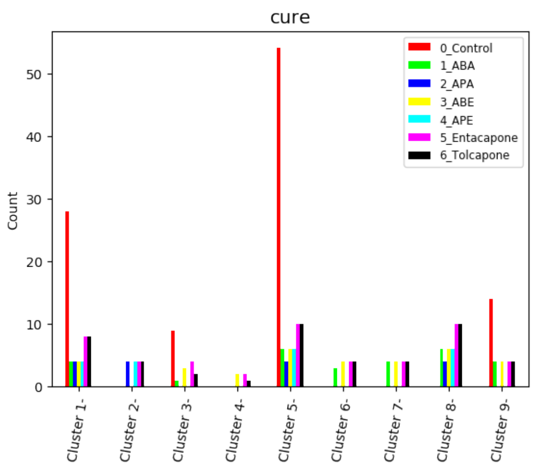


Figure B.68: CURE *Glycolysis* CD 9 Clusters

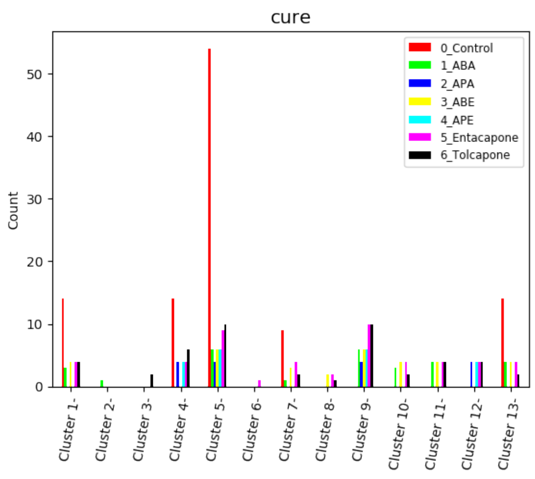


Figure B.69: CURE *Glycolysis* CD 13 Clusters

B.2 Classification

B.2.1 Initial Configuration

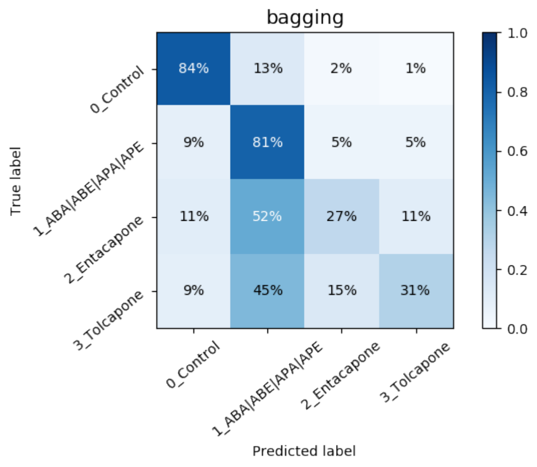


Figure B.70: *Mito* Bagging CM

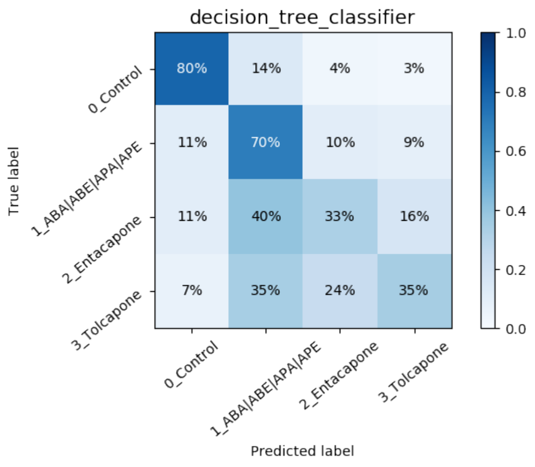


Figure B.71: *Mito* DT CM

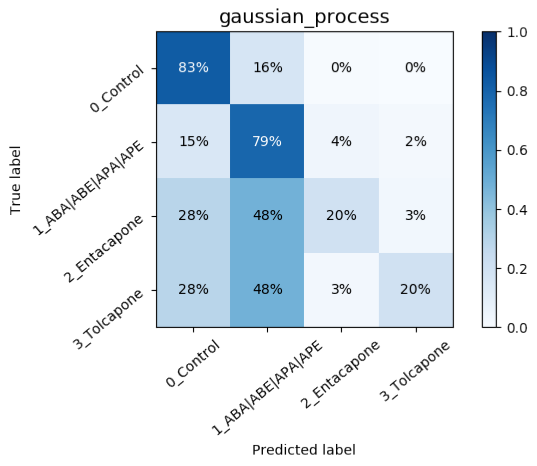


Figure B.72: *Mito* Gaussian Process CM

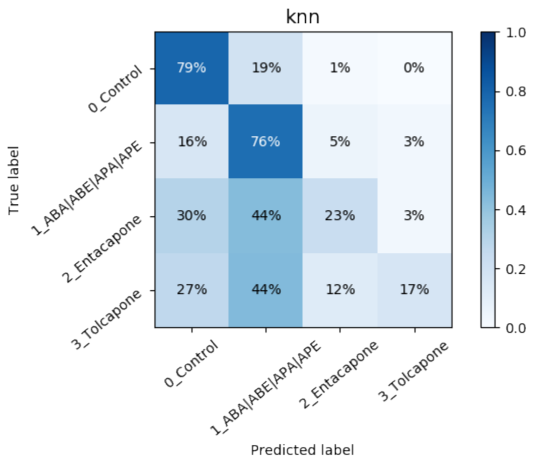


Figure B.73: *Mito* k-NN CM

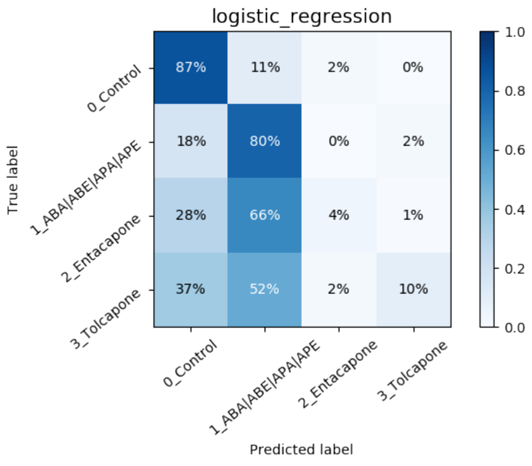


Figure B.74: *Mito* Logistic Regression CM

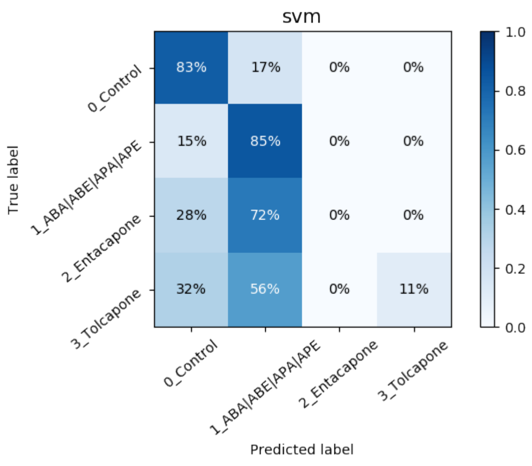


Figure B.75: *Mito* SVM CM

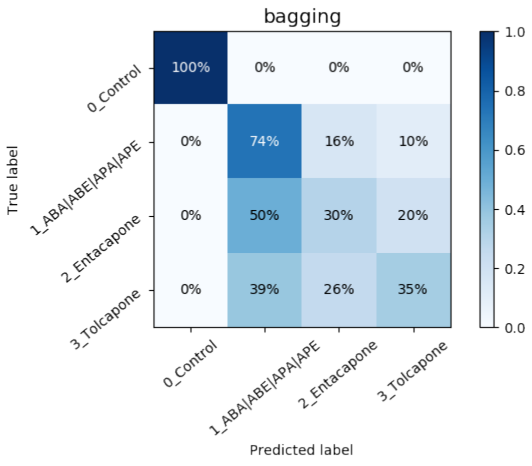


Figure B.76: *Glycolysis* Bagging CM

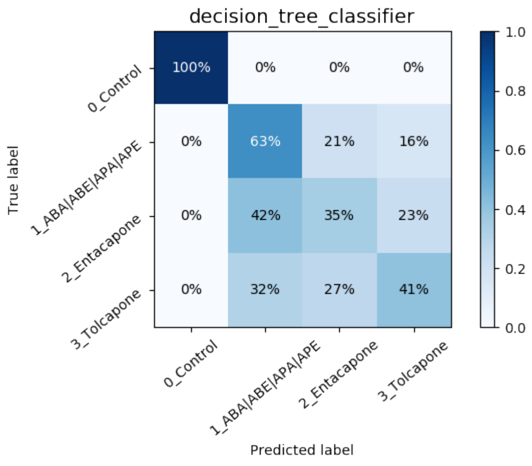


Figure B.77: *Glycolysis* DT CM

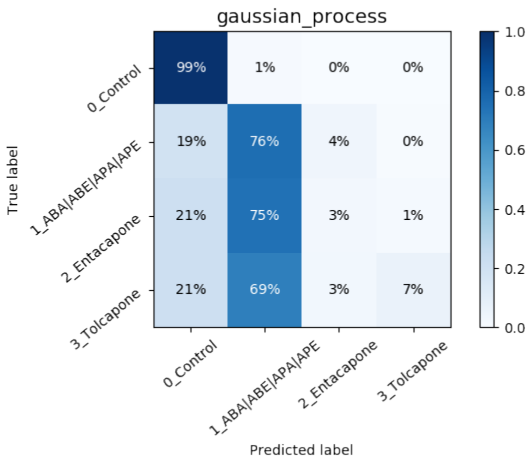


Figure B.78: *Glycolysis* Gaussian Process CM

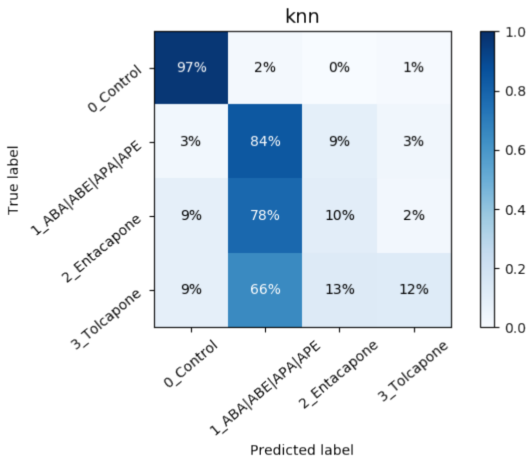


Figure B.79: *Glycolysis* k-NN CM

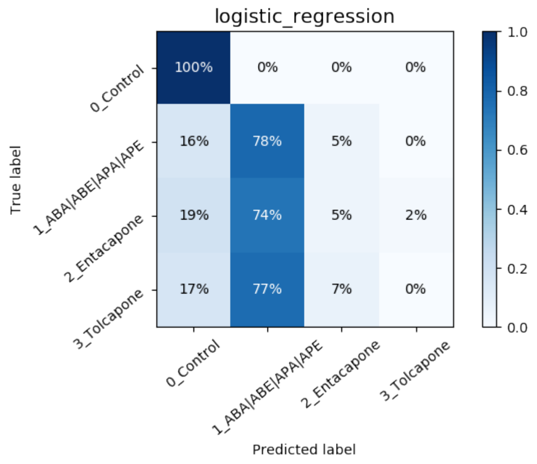


Figure B.80: *Glycolysis* Logistic Regression CM

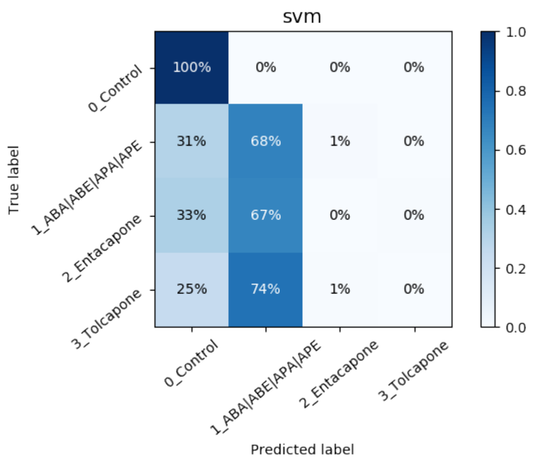


Figure B.81: *Glycolysis* SVM CM

Data Balancing

Random Undersampling

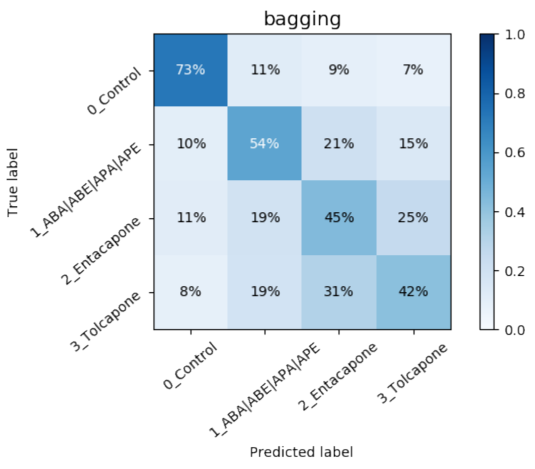


Figure B.82: *Mito* Undersamp. Bagging CM

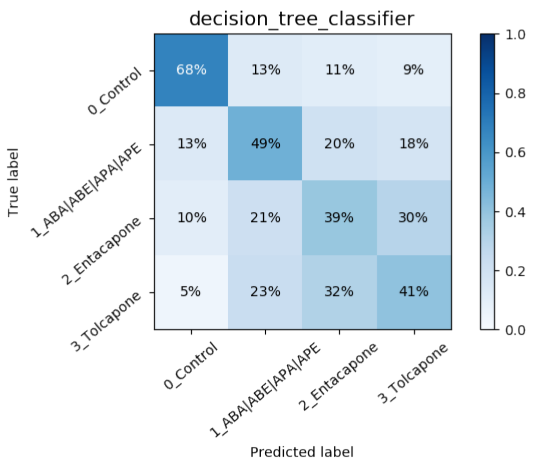


Figure B.83: *Mito* Undersamp. DT CM

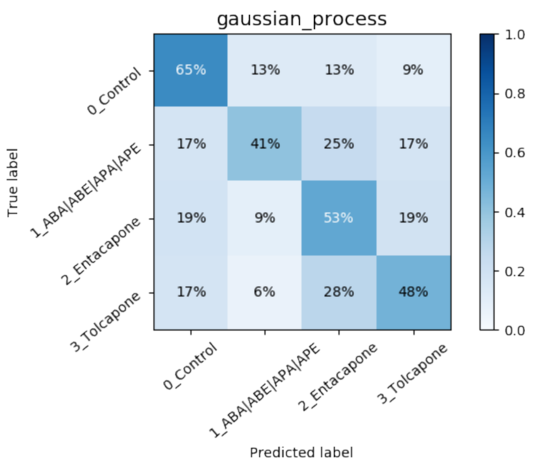


Figure B.84: *Mito* Undersamp. Gaussian Process CM

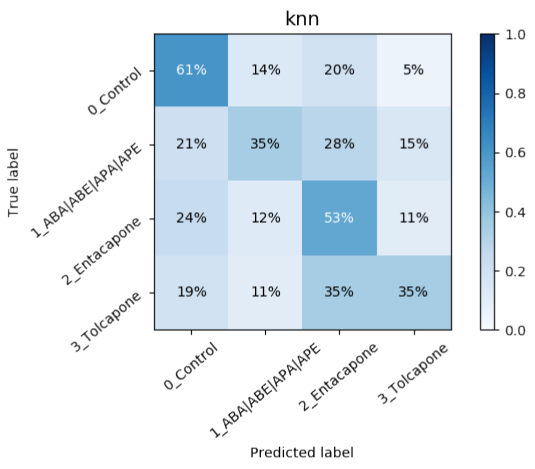


Figure B.85: *Mito* Undersamp. k-NN CM

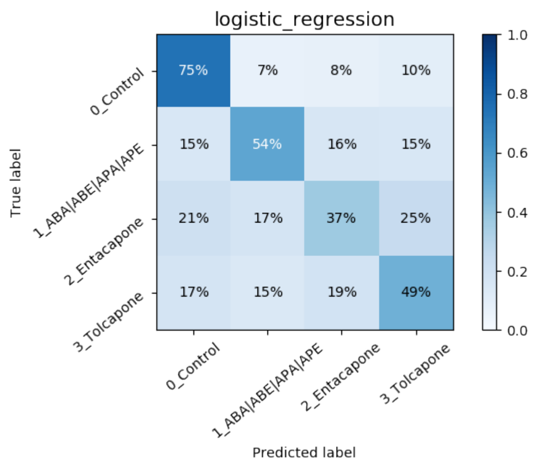


Figure B.86: *Mito* Undersamp. Logistic Regression CM

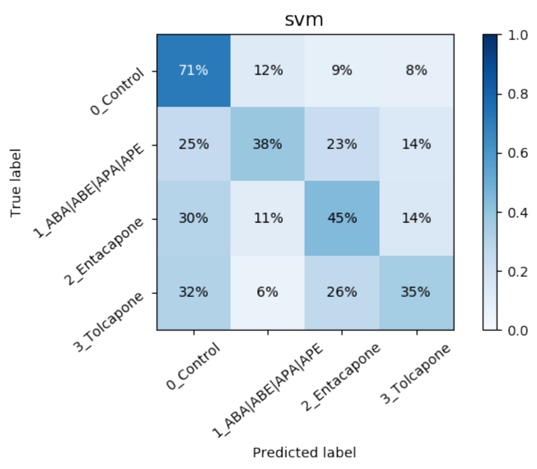


Figure B.87: *Mito* Undersamp. SVM CM

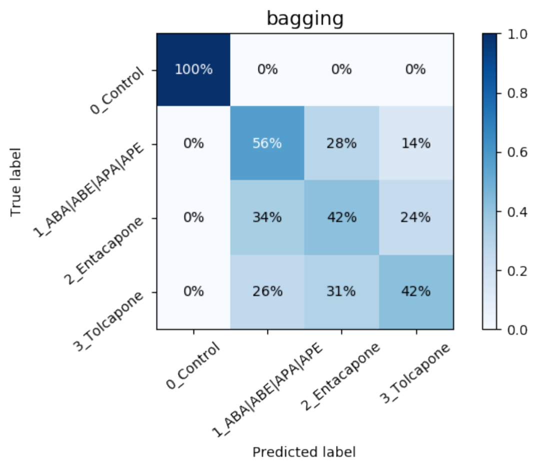


Figure B.88: *Glycolysis* Undersamp. Bagging CM

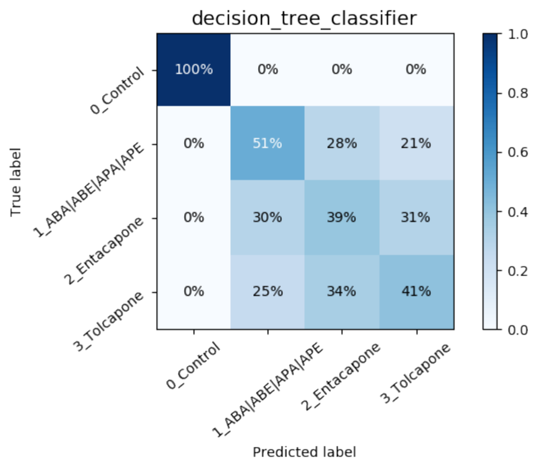


Figure B.89: *Glycolysis* Undersamp. DT CM

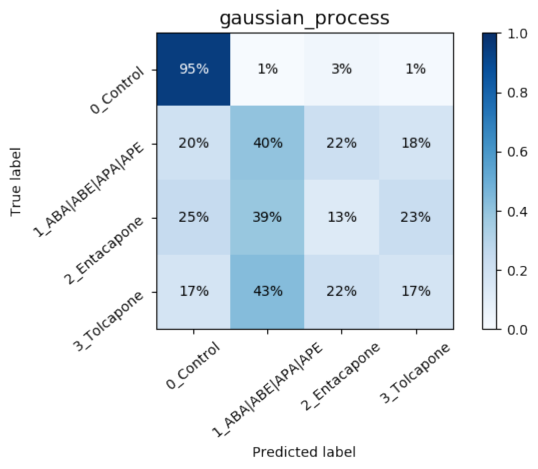


Figure B.90: *Glycolysis* Undersamp. Gaussian Process CM

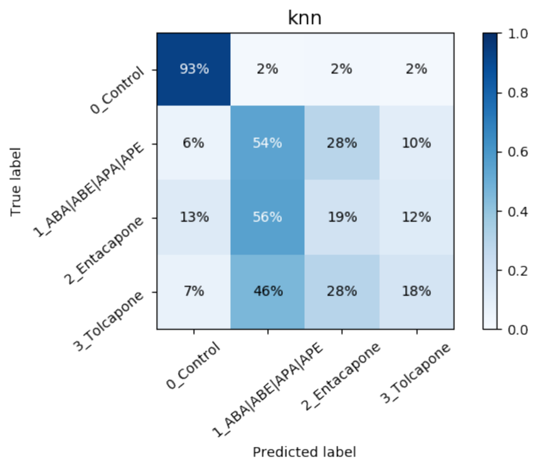


Figure B.91: *Glycolysis* Undersamp. k-NN CM

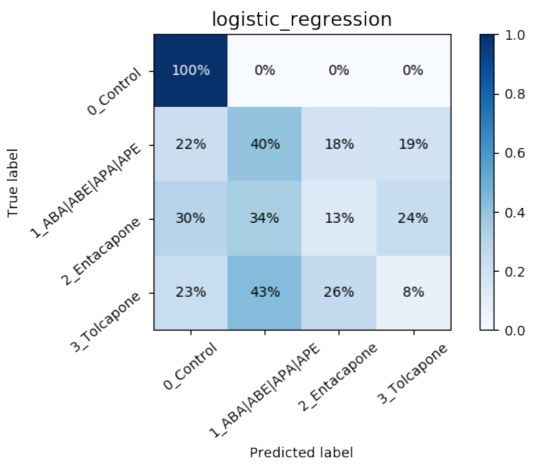


Figure B.92: *Glycolysis* Undersamp. Logistic Regression CM

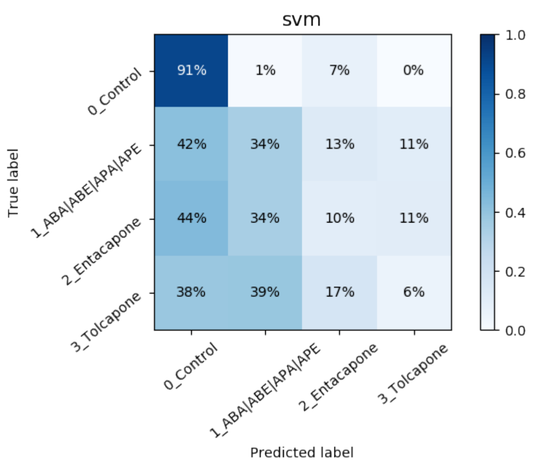


Figure B.93: *Glycolysis* Undersamp. SVM CM

Random Oversampling

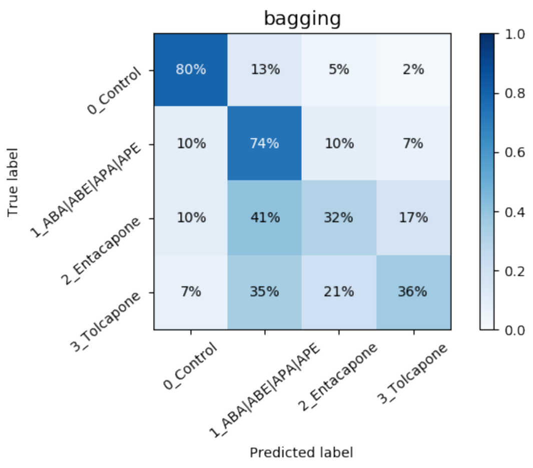


Figure B.94: *Mito* Oversamp. Bagging CM

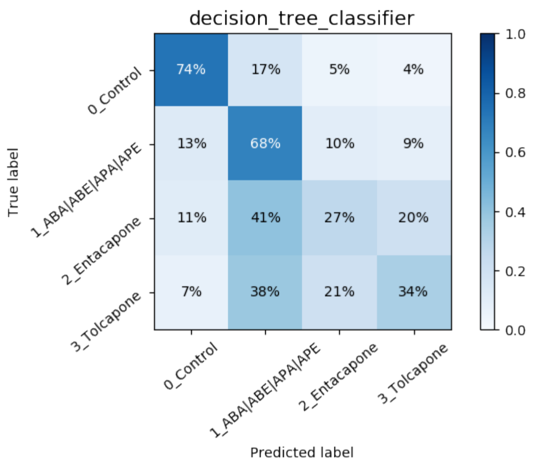


Figure B.95: *Mito* Oversamp. DT CM

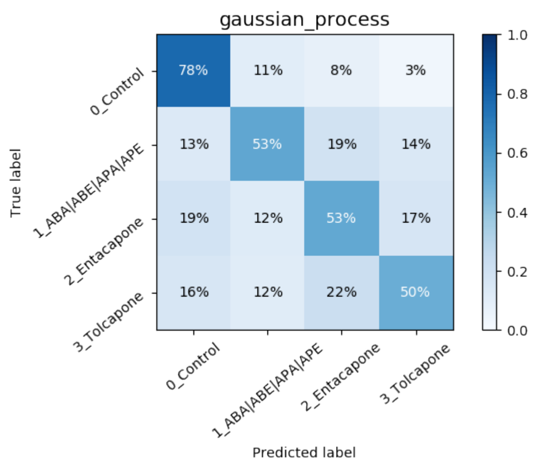


Figure B.96: *Mito* Oversamp. Gaussian Process CM

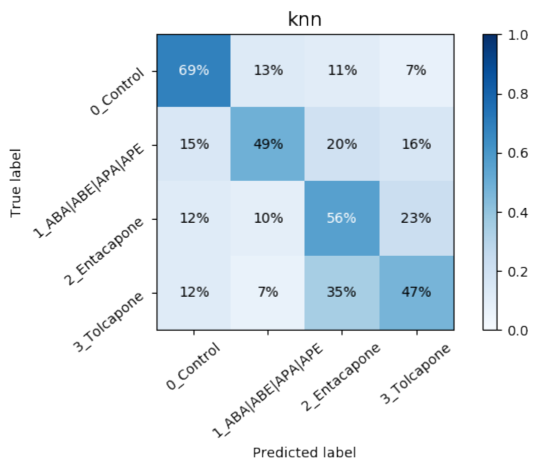


Figure B.97: *Mito* Oversamp. k-NN CM

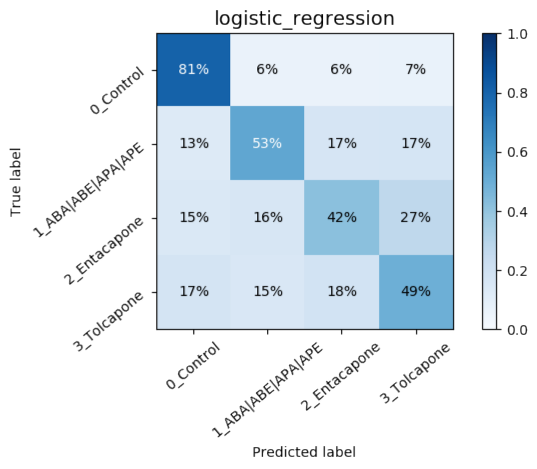


Figure B.98: *Mito* Oversamp. Logistic Regression CM

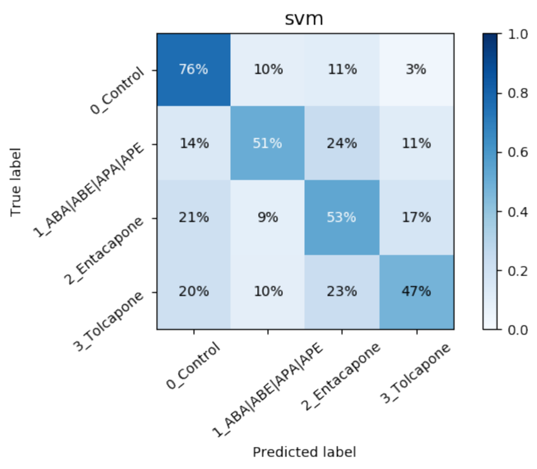


Figure B.99: *Mito* Oversamp. SVM CM

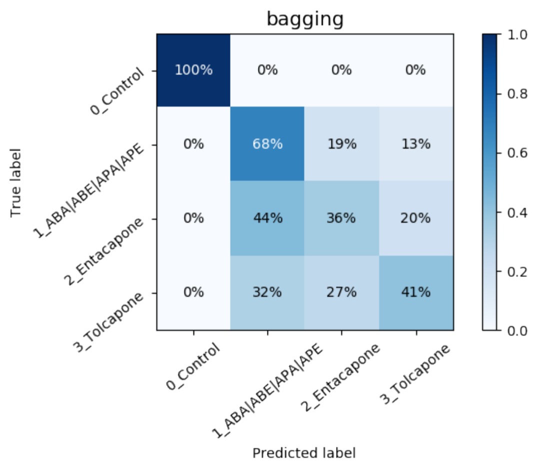


Figure B.100: *Glycolysis* Oversamp. Bagging CM

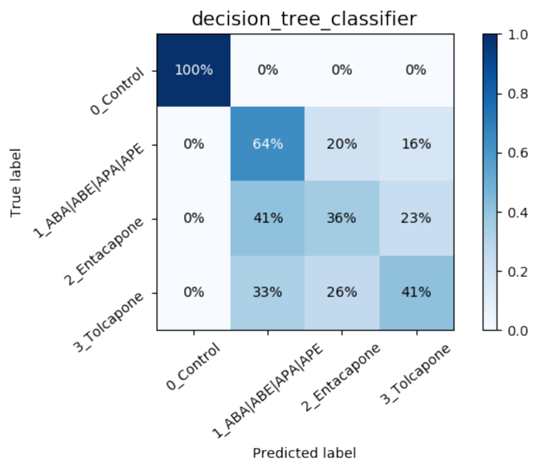


Figure B.101: *Glycolysis* Oversamp. DT CM

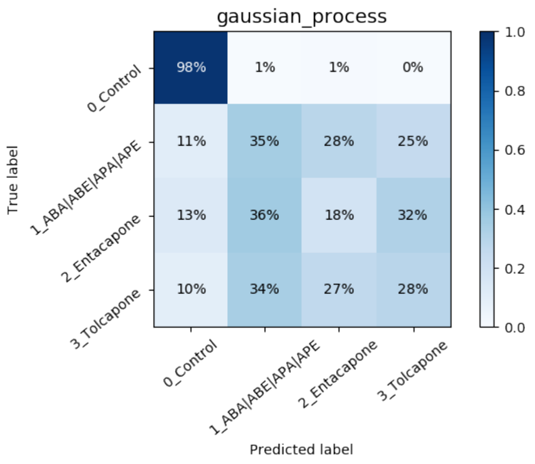


Figure B.102: *Glycolysis* Oversamp. Gaussian Process CM

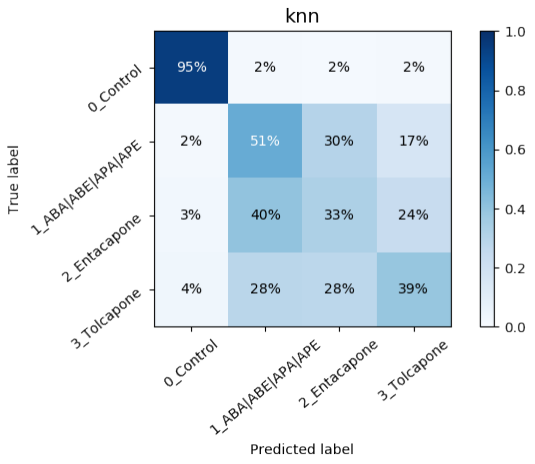


Figure B.103: *Glycolysis* Oversamp. k-NN CM

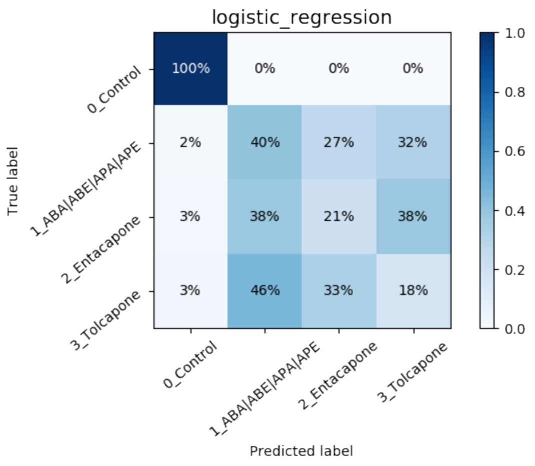


Figure B.104: *Glycolysis* Oversamp. Logistic Regression CM

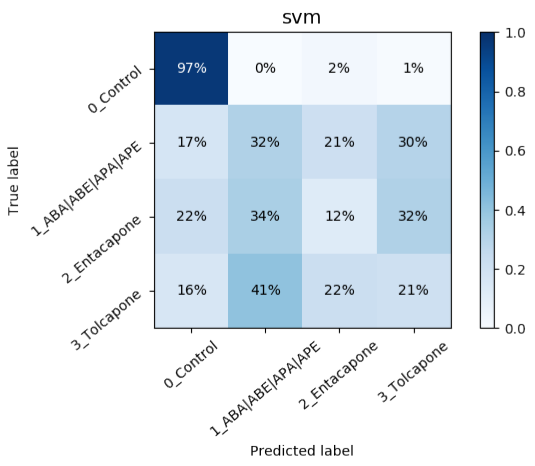


Figure B.105: *Glycolysis* Oversamp. SVM CM

Synthetic Minority Over-sampling Technique (SMOTE)

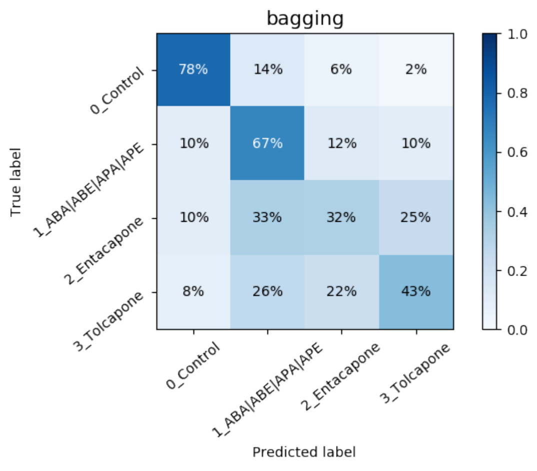


Figure B.106: *Mito* SMOTE Bagging CM

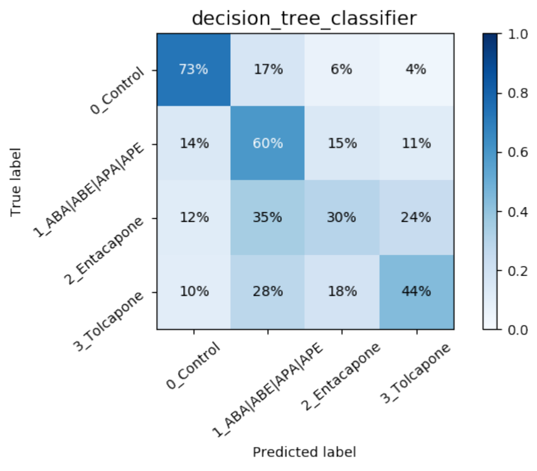


Figure B.107: *Mito* SMOTE DT CM

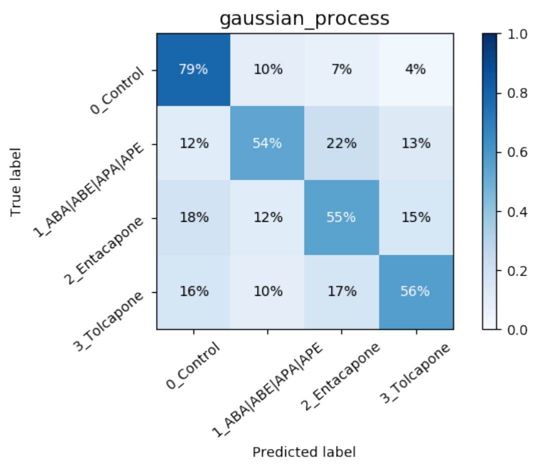


Figure B.108: *Mito* SMOTE Gaussian Process CM

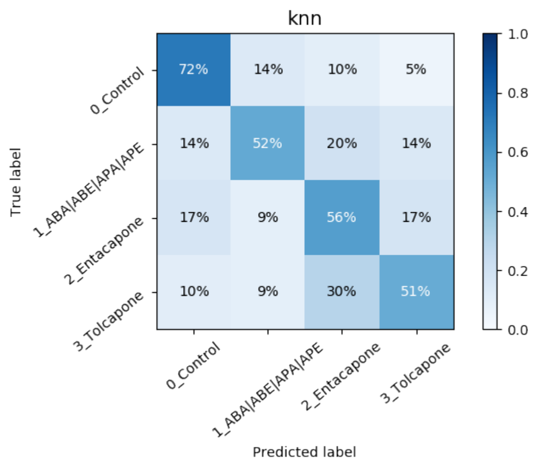


Figure B.109: *Mito* SMOTE k-NN CM

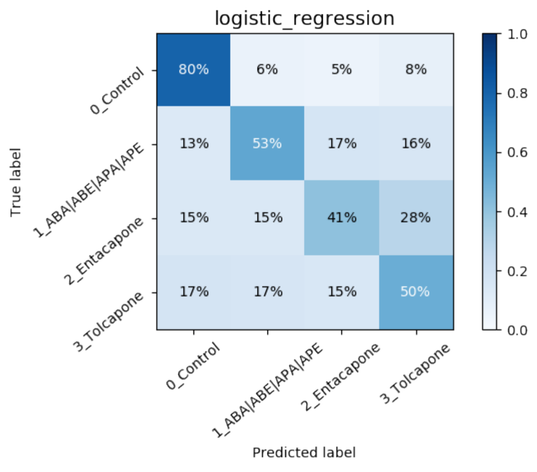


Figure B.110: *Mito* SMOTE Logistic Regression CM

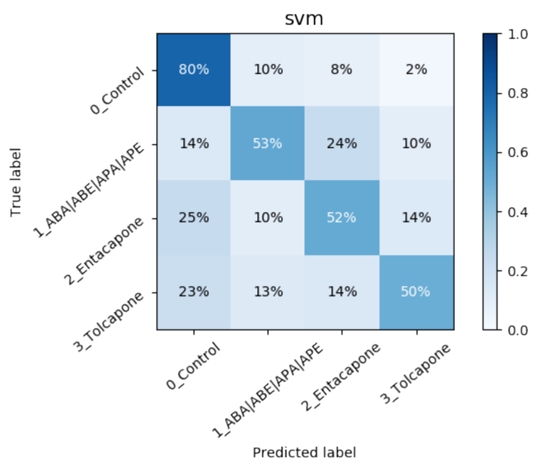


Figure B.111: *Mito* SMOTE SVM CM

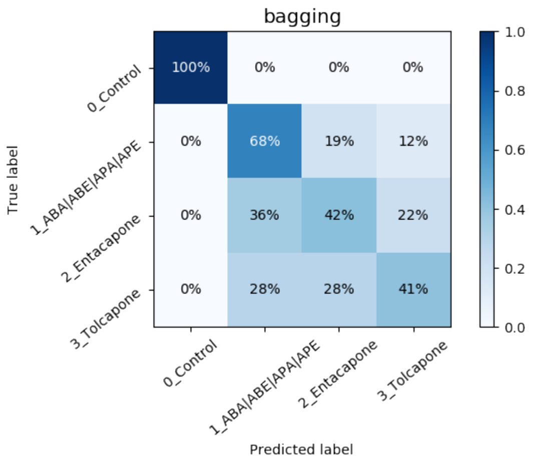


Figure B.112: *Glycolysis* SMOTE Bagging CM

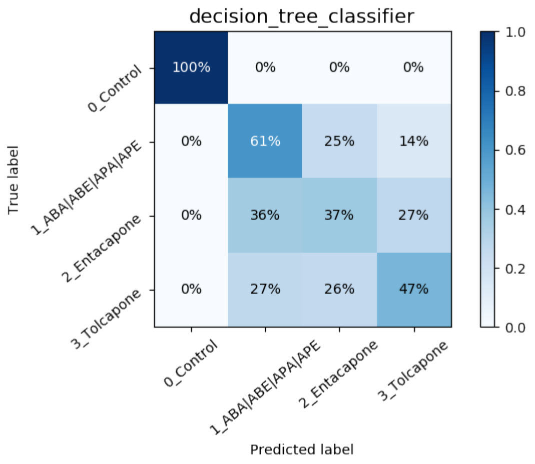


Figure B.113: *Glycolysis* SMOTE DT CM

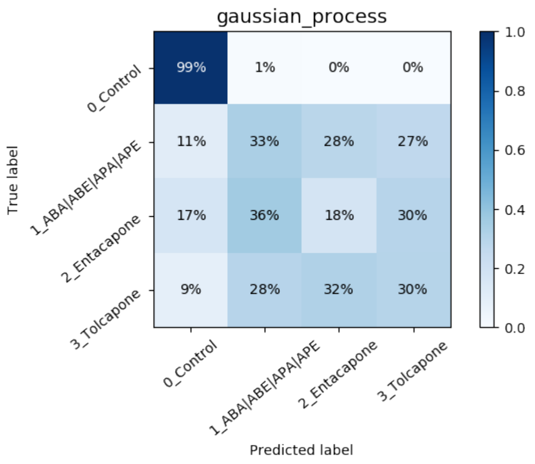


Figure B.114: *Glycolysis* SMOTE Gaussian Process CM

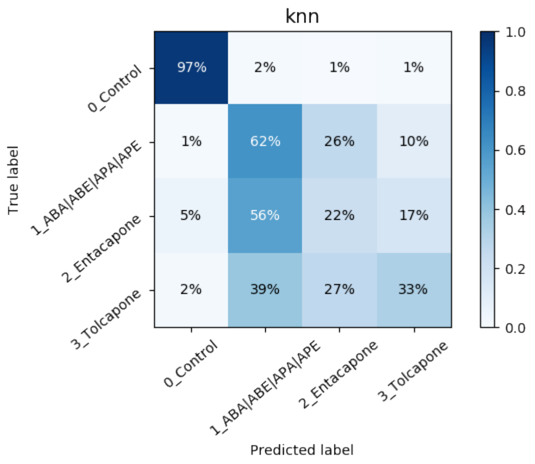


Figure B.115: *Glycolysis* SMOTE k-NN CM

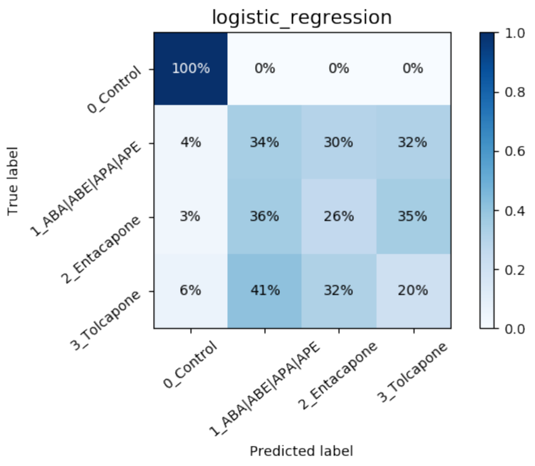


Figure B.116: *Glycolysis* SMOTE Logistic Regression CM

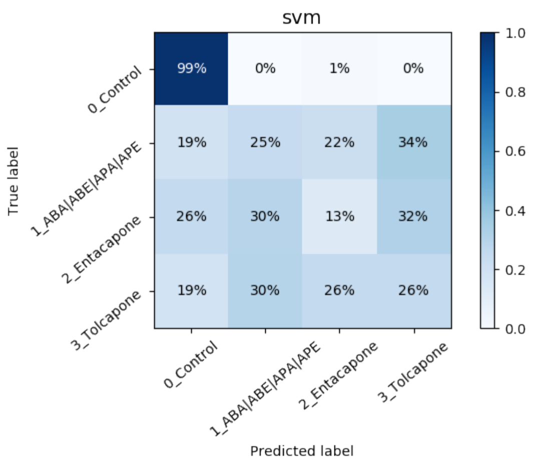


Figure B.117: *Glycolysis* SMOTE SVM CM

Dimensionality Reduction

Correlation

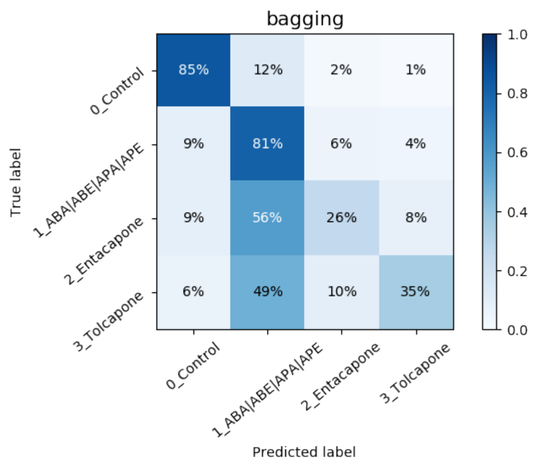


Figure B.118: *Mito* Correlation Bagging CM

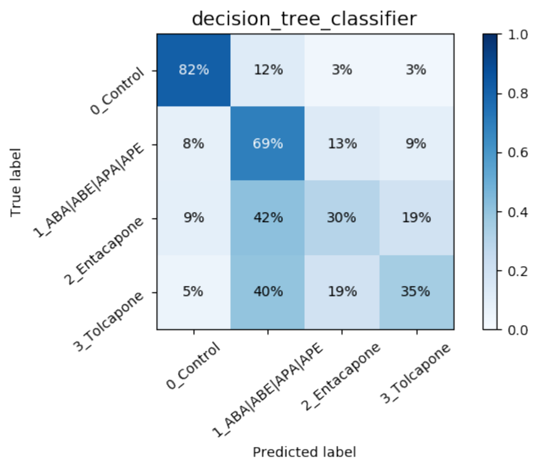


Figure B.119: *Mito* Correlation DT CM

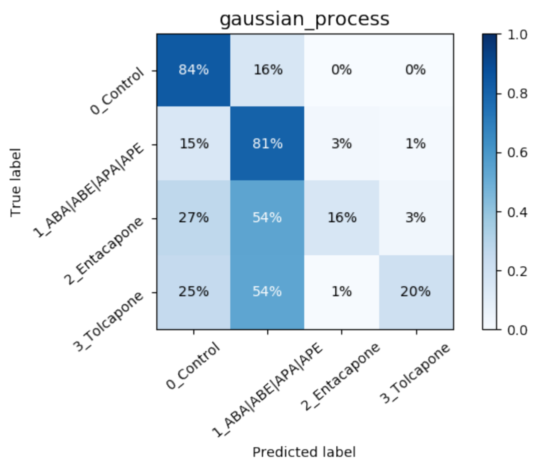


Figure B.120: *Mito* Correlation Gaussian Process CM

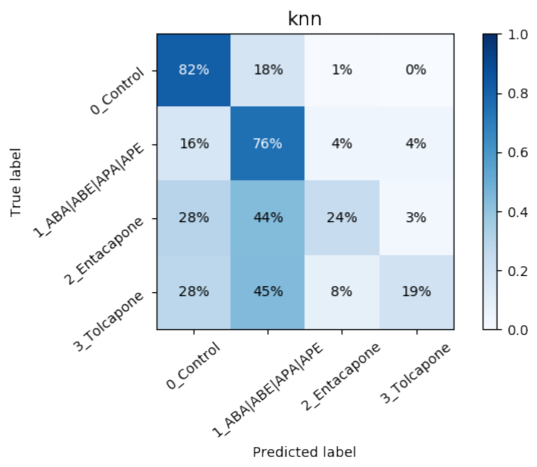


Figure B.121: *Mito* Correlation k-NN CM

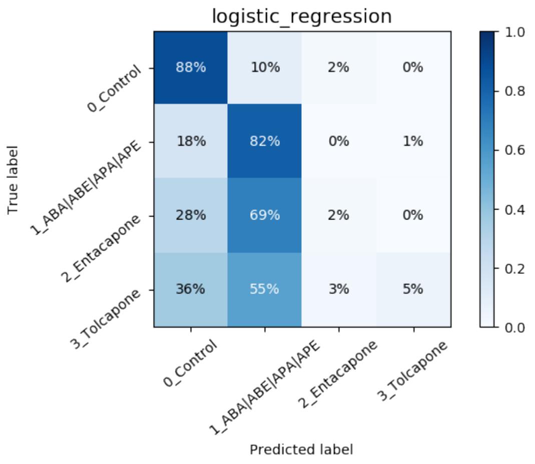


Figure B.122: *Mito* Correlation Logistic Regression CM

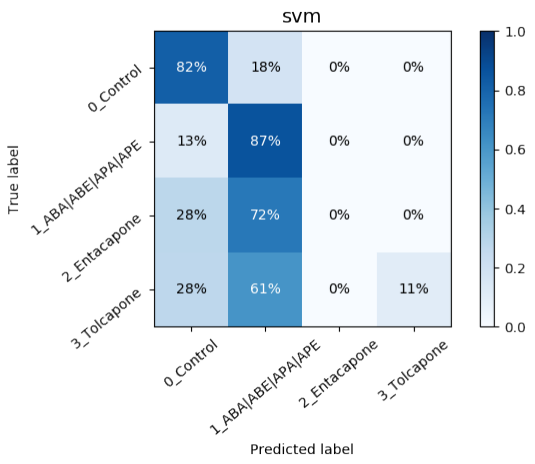


Figure B.123: *Mito* Correlation SVM CM

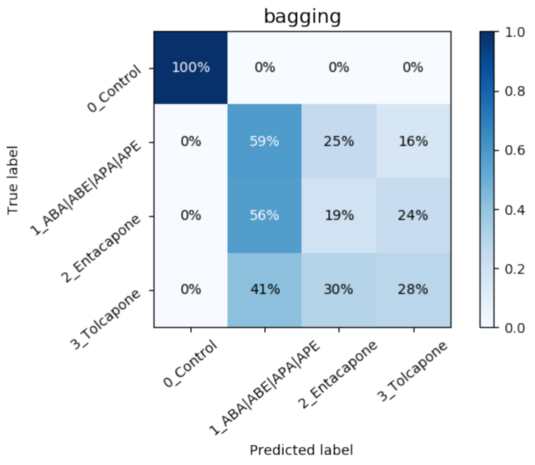


Figure B.124: *Glycolysis* Correlation Bagging CM

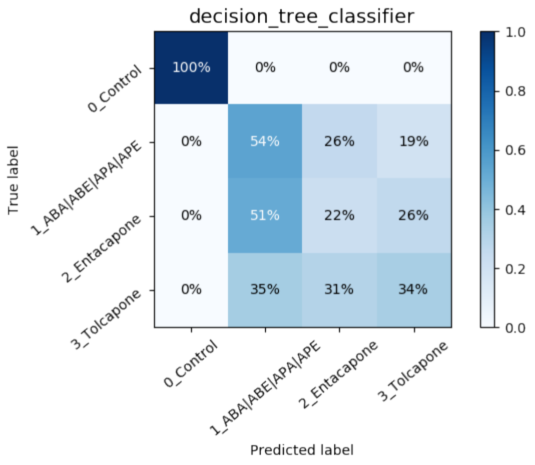


Figure B.125: *Glycolysis* Correlation DT CM

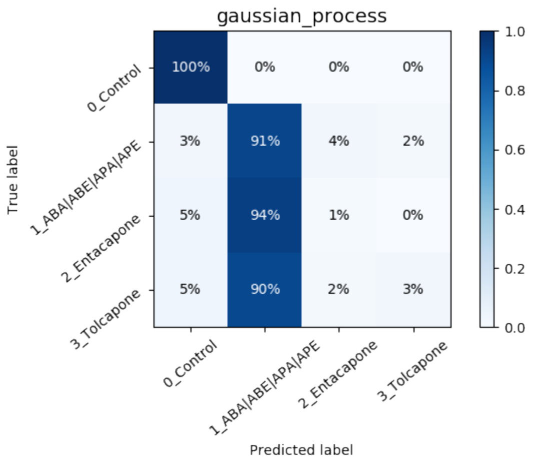


Figure B.126: *Glycolysis* Correlation Gaussian Process CM

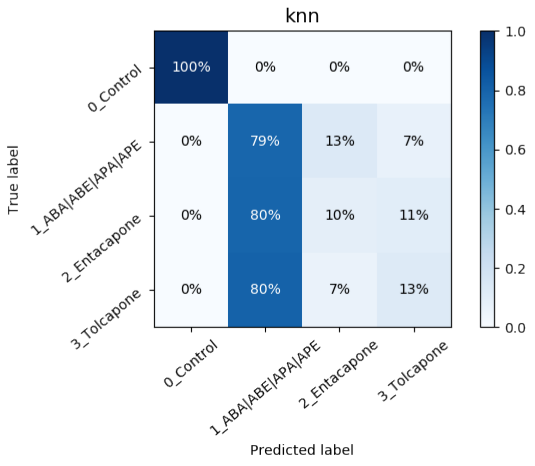


Figure B.127: *Glycolysis* Correlation k-NN CM

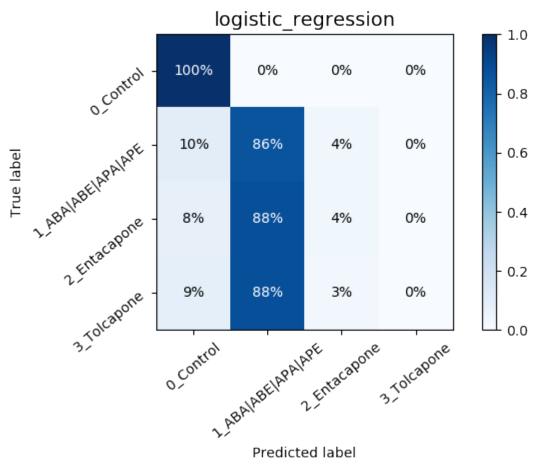


Figure B.128: *Glycolysis* Correlation Logistic Regression CM

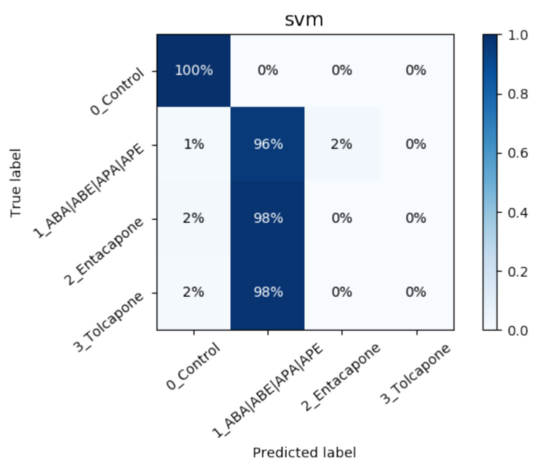


Figure B.129: *Glycolysis* Correlation SVM CM

Mutual Information (MI)

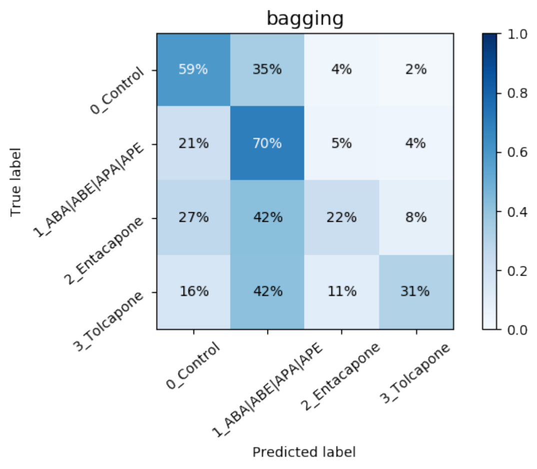


Figure B.130: *Mito* MI Bagging CM

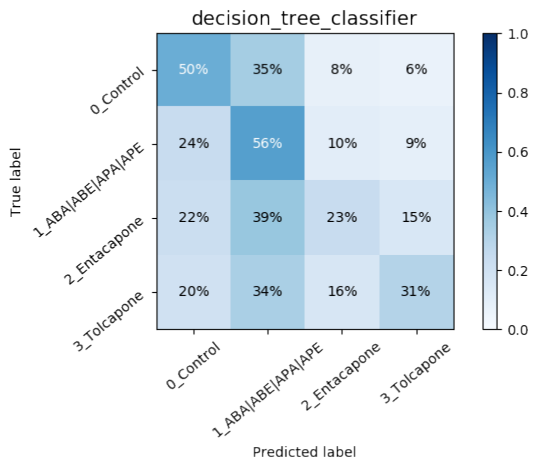


Figure B.131: *Mito* MI DT CM

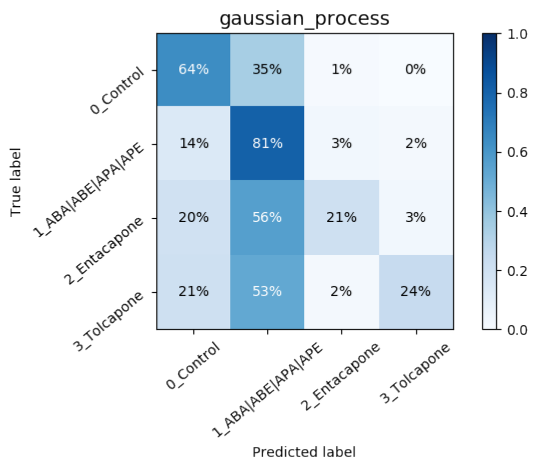


Figure B.132: *Mito* MI Gaussian Process CM

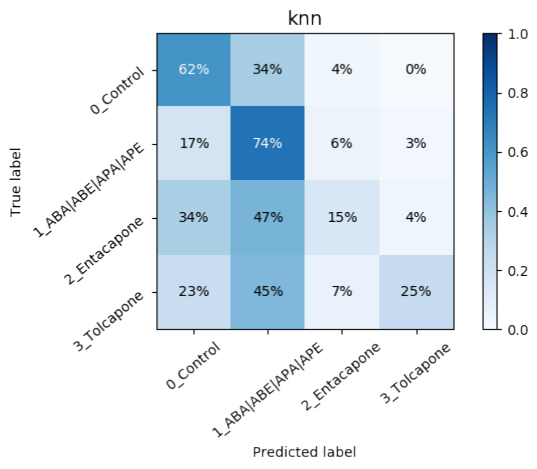


Figure B.133: *Mito* MI k-NN CM

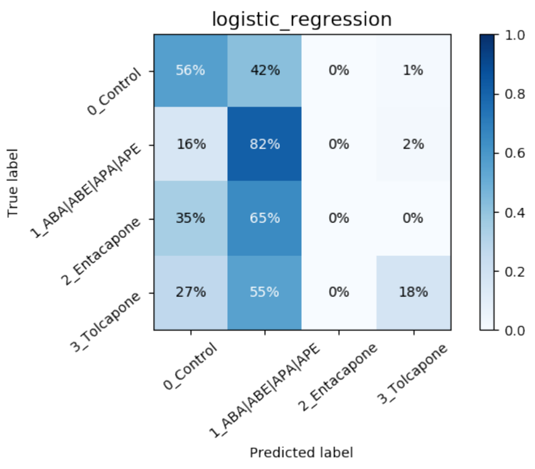


Figure B.134: *Mito* MI Logistic Regression CM

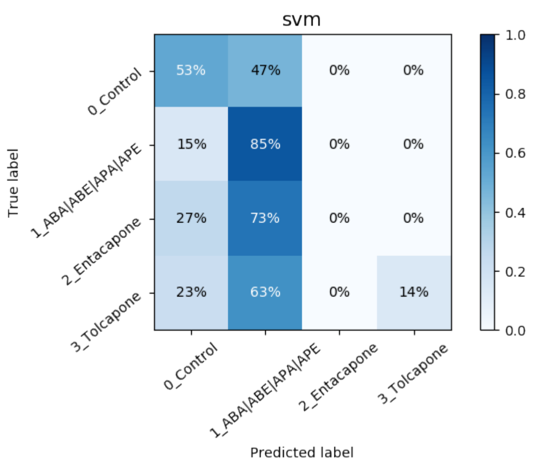


Figure B.135: *Mito* MI SVM CM

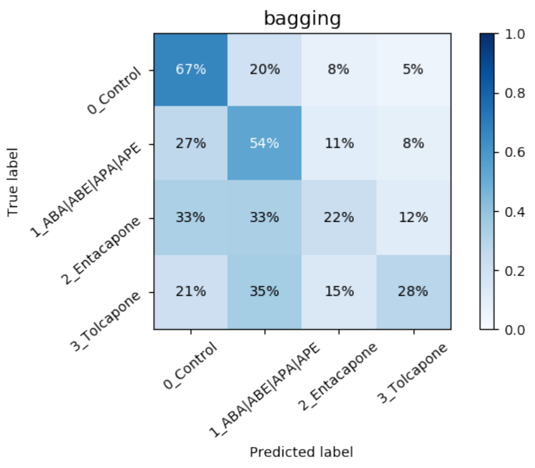


Figure B.136: *Glycolysis* MI Bagging CM

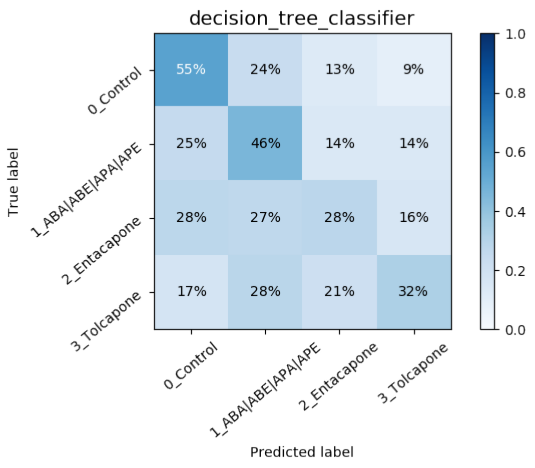


Figure B.137: *Glycolysis* MI DT CM

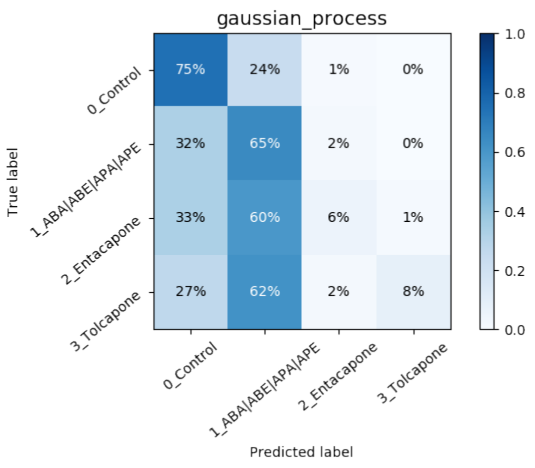


Figure B.138: *Glycolysis* MI Gaussian Process CM

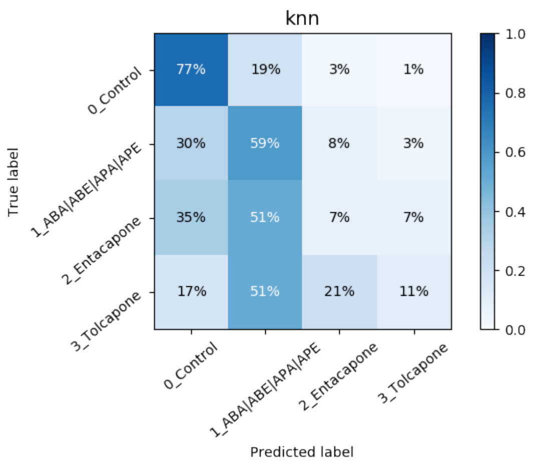


Figure B.139: *Glycolysis* MI k-NN CM

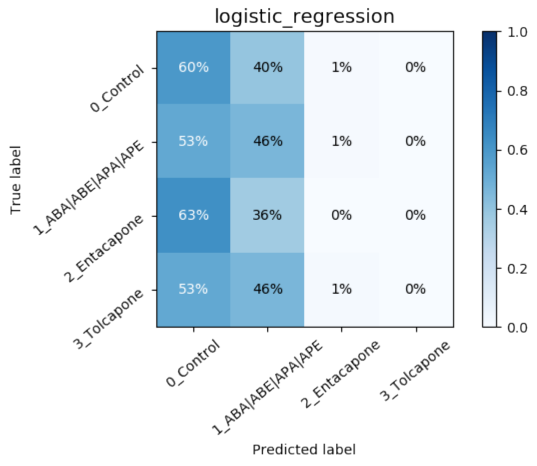


Figure B.140: *Glycolysis* MI Logistic Regression CM

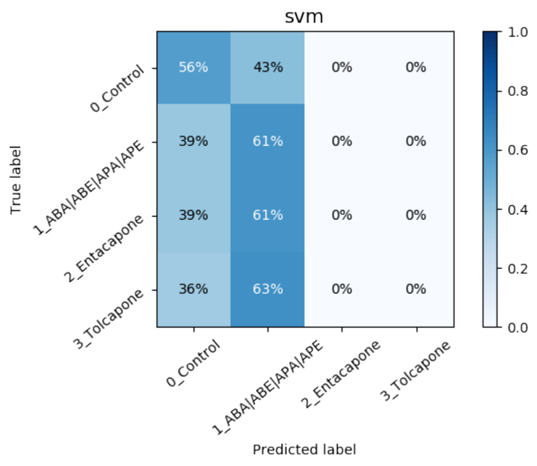


Figure B.141: *Glycolysis* MI SVM CM

Principal Component Analysis (PCA)

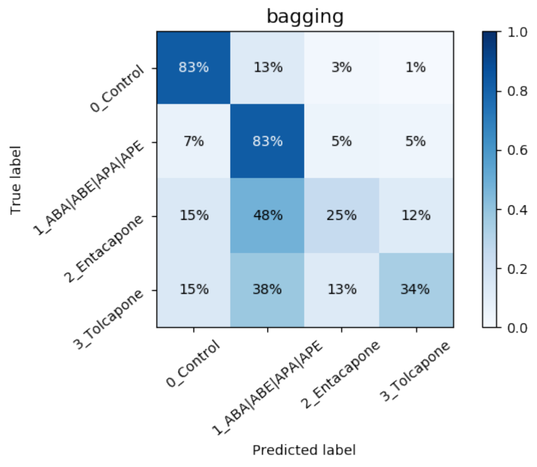


Figure B.142: *Mito* PCA Bagging CM

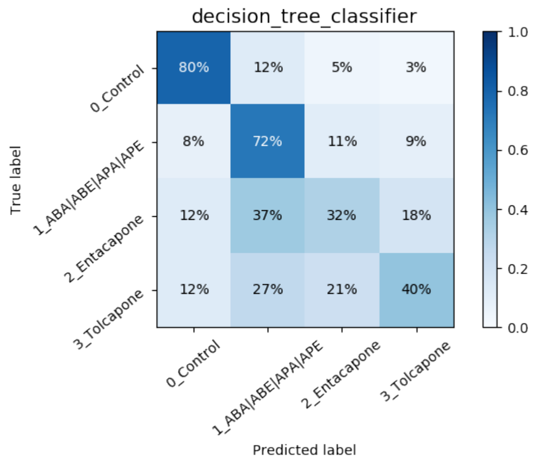


Figure B.143: *Mito* PCA DT CM

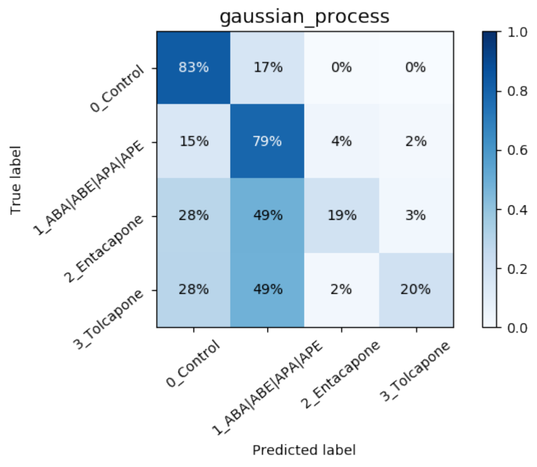


Figure B.144: *Mito* PCA Gaussian Process CM

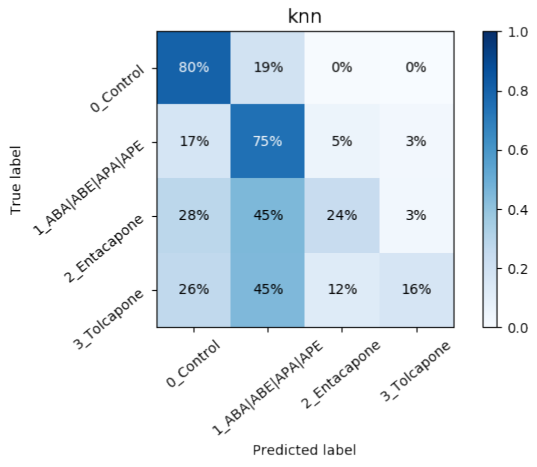


Figure B.145: *Mito* PCA k-NN CM

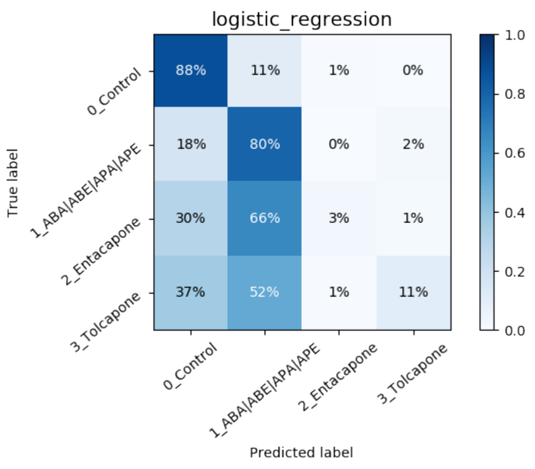


Figure B.146: *Mito* PCA Logistic Regression CM

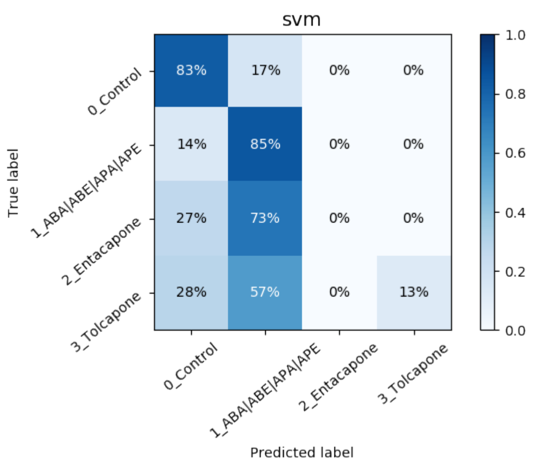


Figure B.147: *Mito* PCA SVM CM

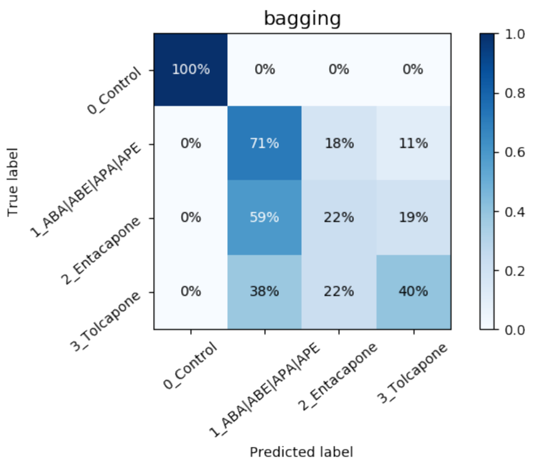


Figure B.148: *Glycolysis* PCA Bagging CM

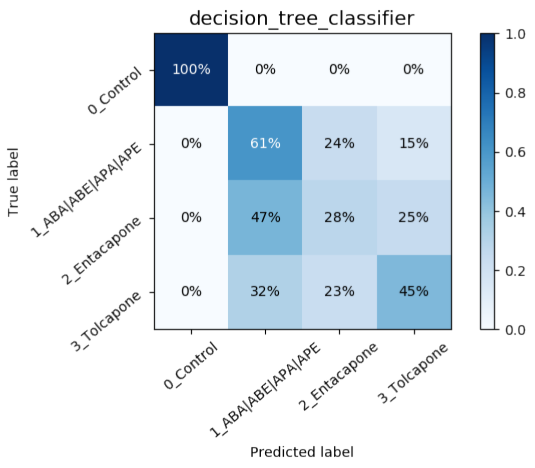


Figure B.149: *Glycolysis* PCA DT CM

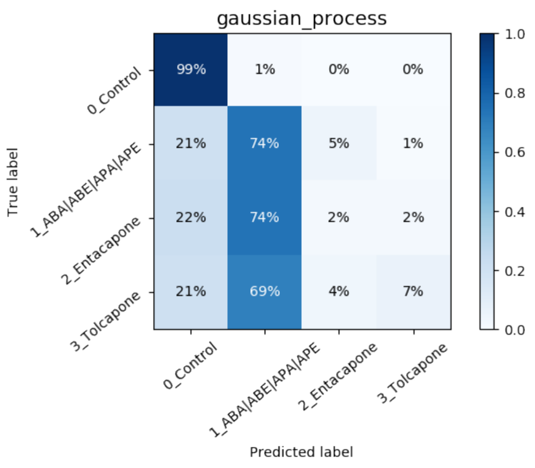


Figure B.150: *Glycolysis* PCA Gaussian Process CM

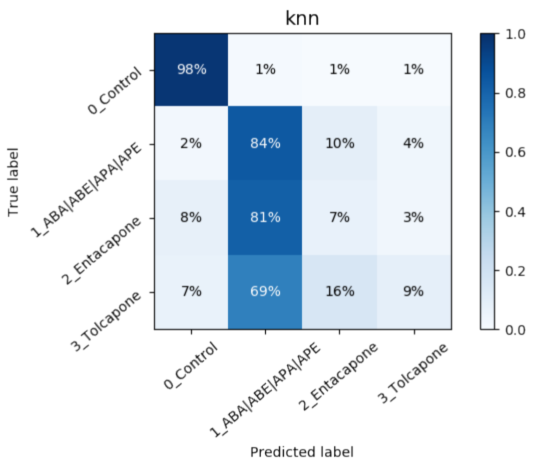


Figure B.151: *Glycolysis* PCA k-NN CM

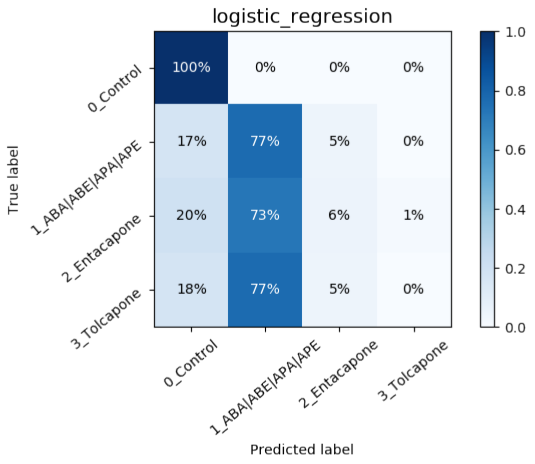


Figure B.152: *Glycolysis* PCA Logistic Regression CM

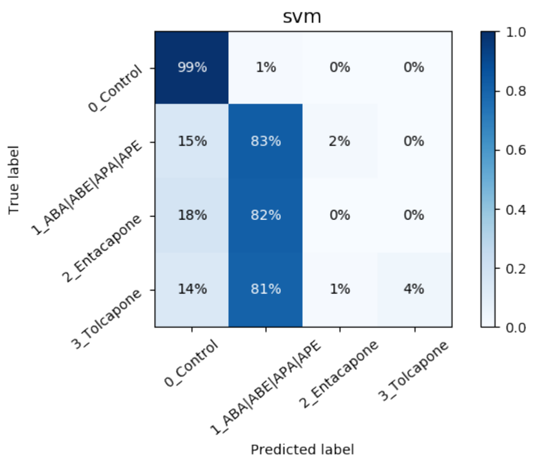


Figure B.153: *Glycolysis* PCA SVM CM

Recursive Feature Elimination (RFE)

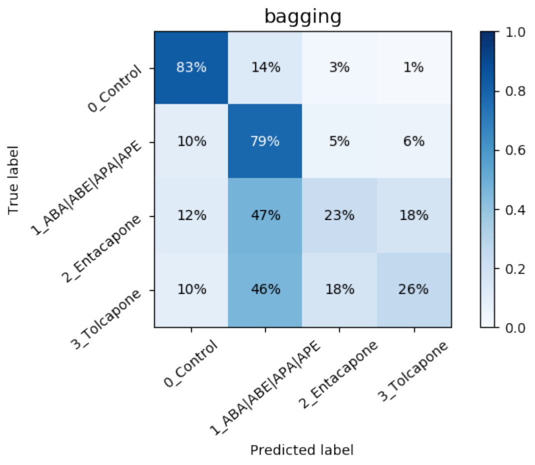


Figure B.154: *Mito* RFE Bagging CM

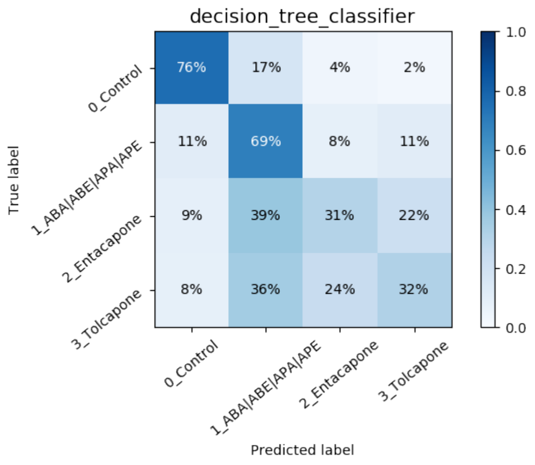


Figure B.155: *Mito* RFE DT CM

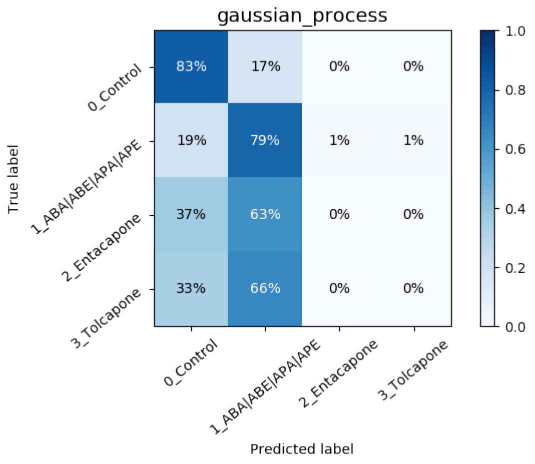


Figure B.156: *Mito* RFE Gaussian Process CM

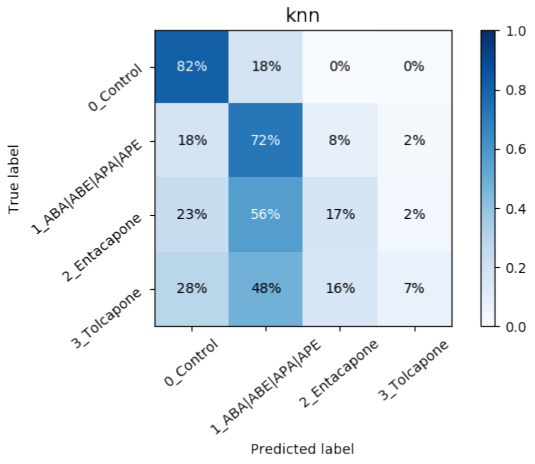


Figure B.157: *Mito* RFE k-NN CM

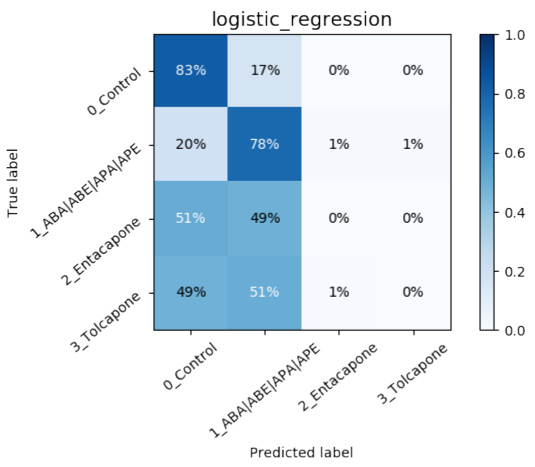


Figure B.158: *Mito* RFE Logistic Regression CM

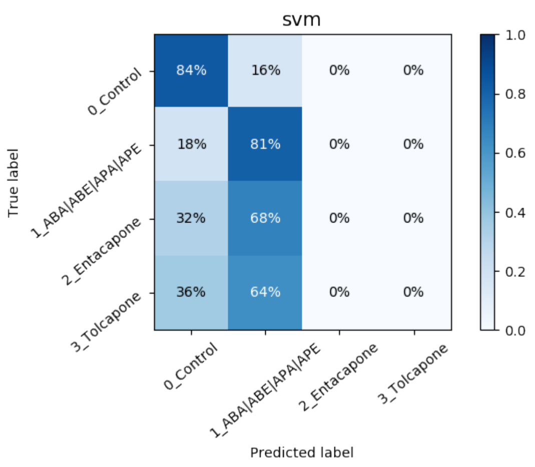


Figure B.159: *Mito* RFE SVM CM

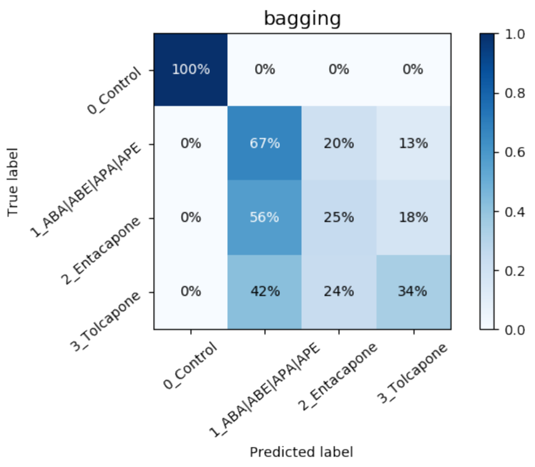


Figure B.160: *Glycolysis* RFE Bagging CM

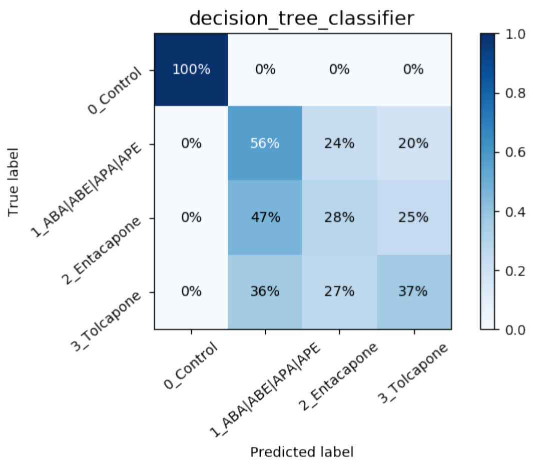


Figure B.161: *Glycolysis* RFE DT CM

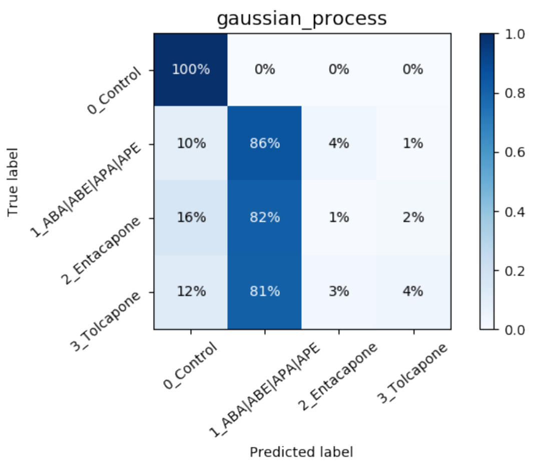


Figure B.162: *Glycolysis* RFE Gaussian Process CM

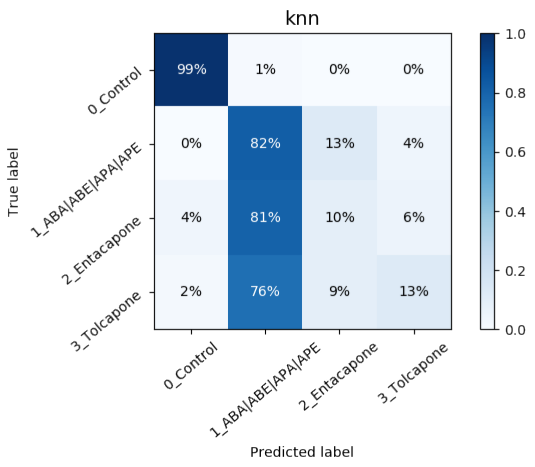


Figure B.163: *Glycolysis* RFE k-NN CM

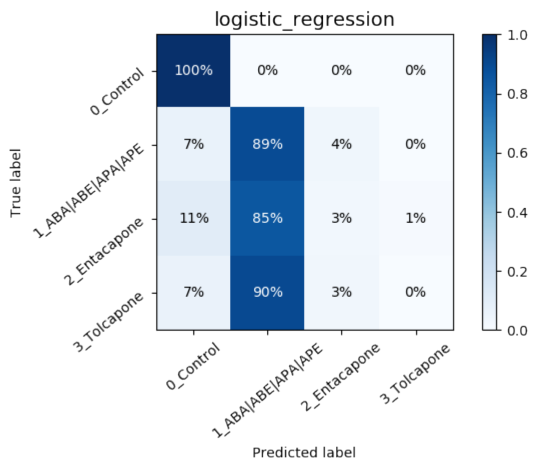


Figure B.164: *Glycolysis* RFE Logistic Regression CM

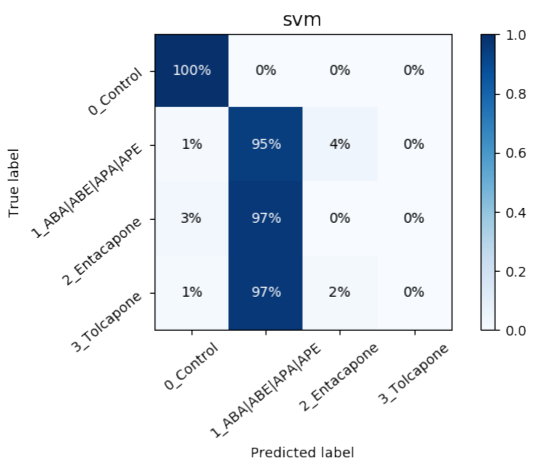


Figure B.165: *Glycolysis* RFE SVM CM

B.2.2 Differentials

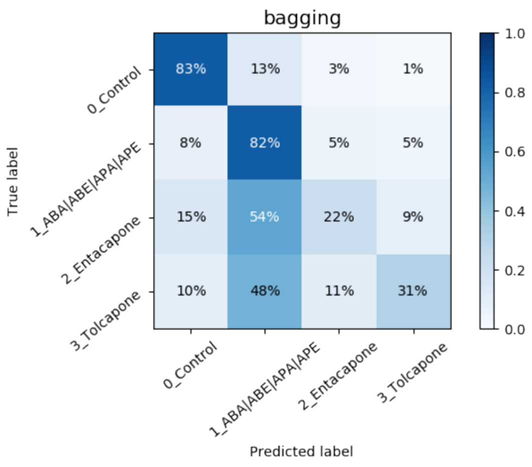


Figure B.166: Differential *Mito* Bagging CM

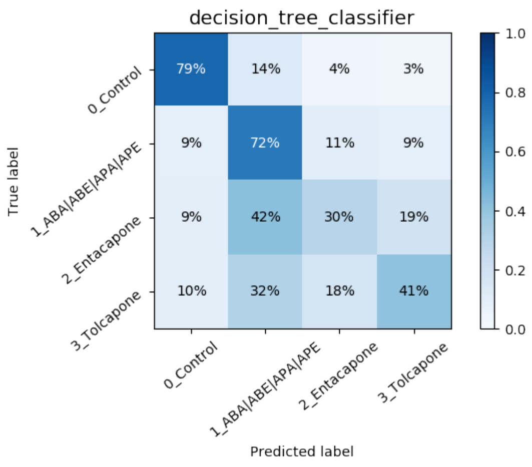


Figure B.167: Differential *Mito* DT CM

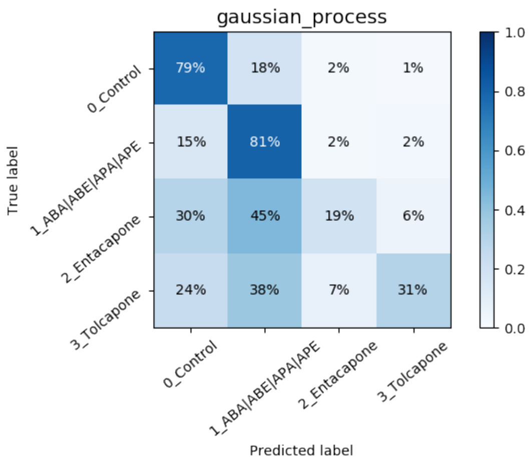


Figure B.168: Differential *Mito* Gaussian Process CM

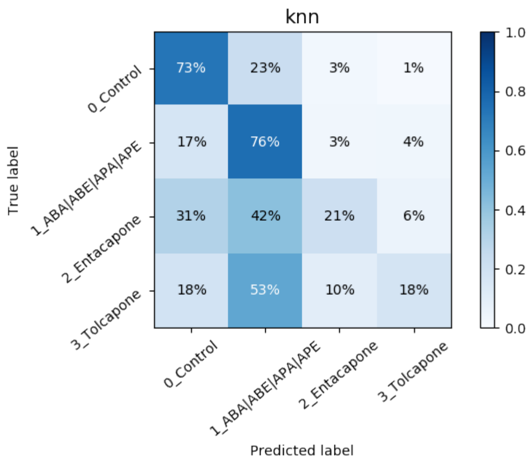


Figure B.169: Differential *Mito* k-NN CM

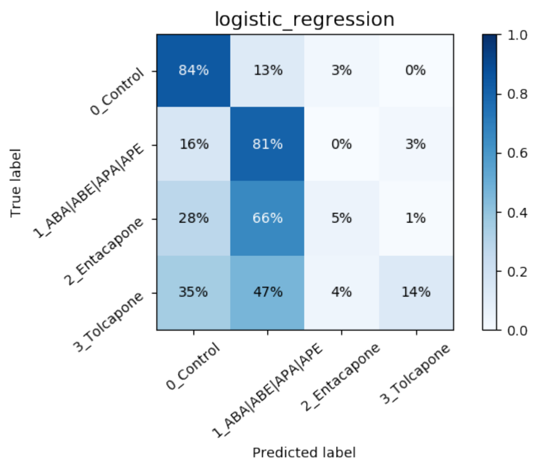


Figure B.170: Differential *Mito* Logistic Regression CM

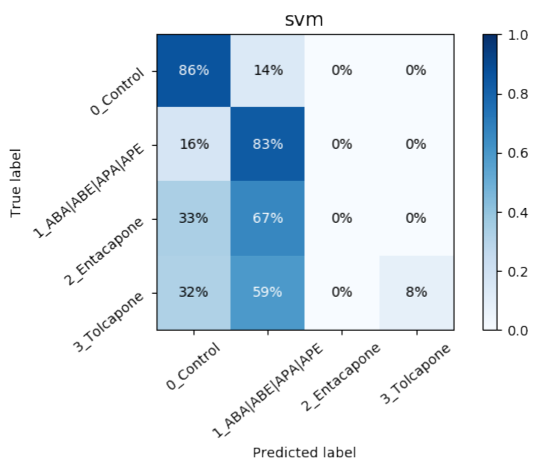


Figure B.171: Differential *Mito* SVM CM

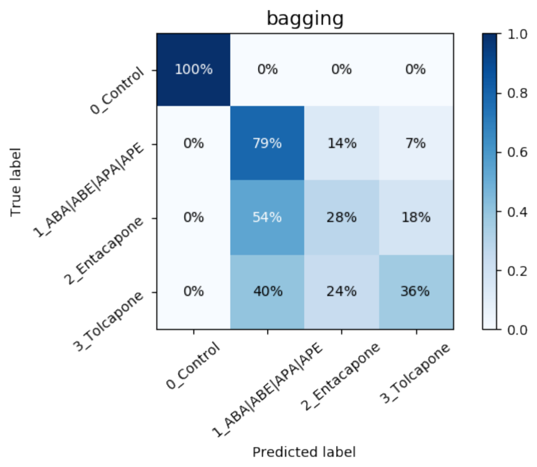


Figure B.172: Differential *Glycolysis* Bagging CM

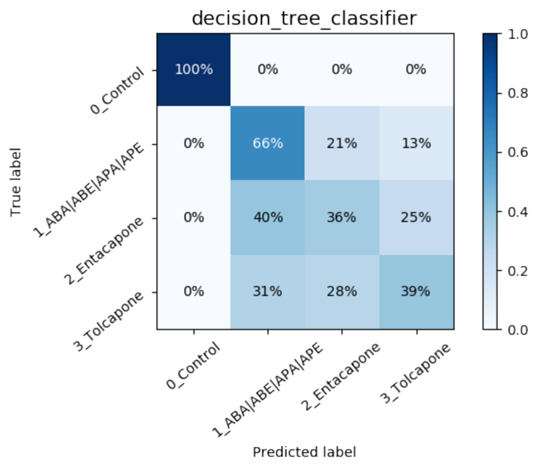


Figure B.173: Differential *Glycolysis* DT CM

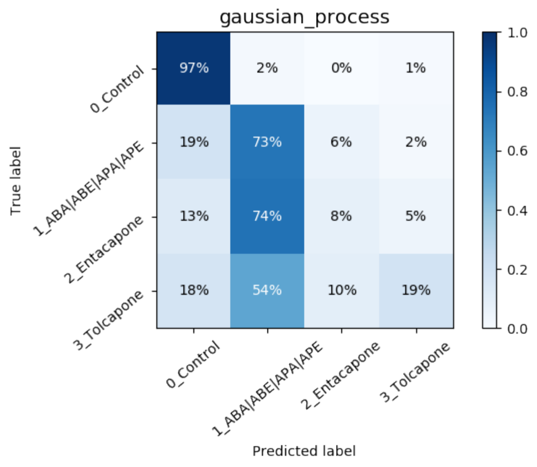


Figure B.174: Differential *Glycolysis* Gaussian Process CM

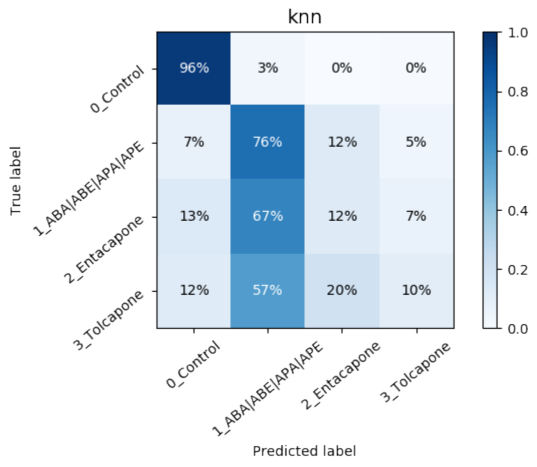


Figure B.175: Differential *Glycolysis* k-NN CM

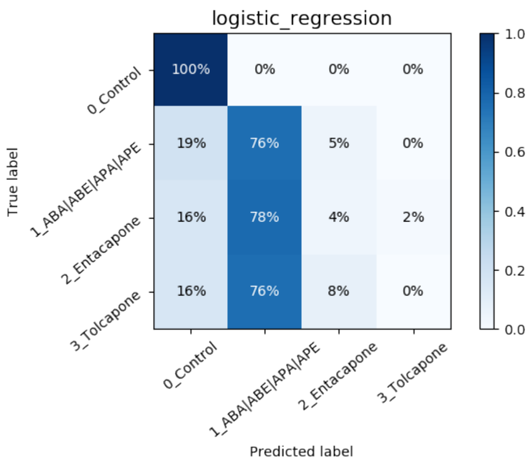


Figure B.176: Differential *Glycolysis* Logistic Regression CM

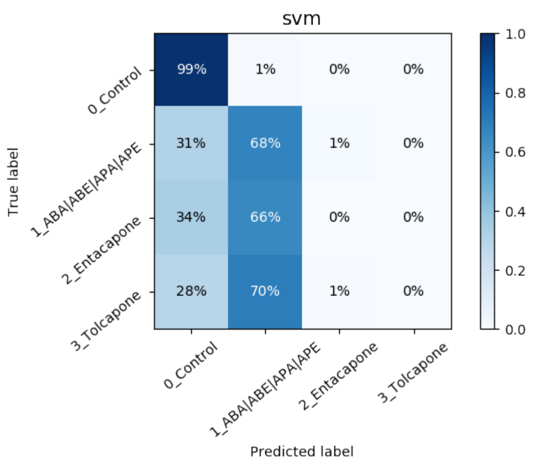


Figure B.177: Differential *Glycolysis* SVM CM

B.2.3 Individual Targets

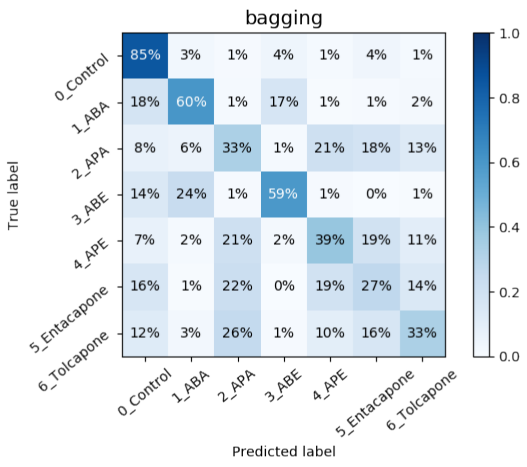


Figure B.178: Individuals *Mito* Bagging CM

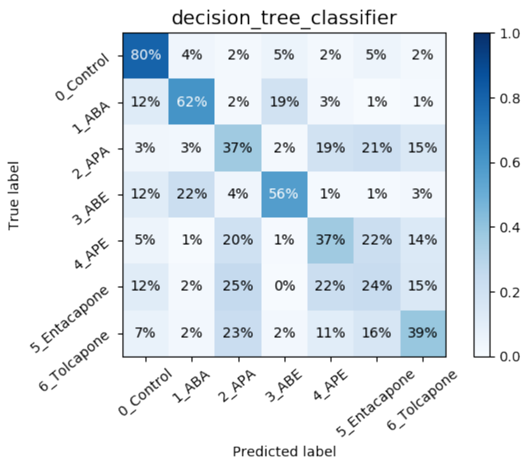


Figure B.179: Individuals *Mito* DT CM

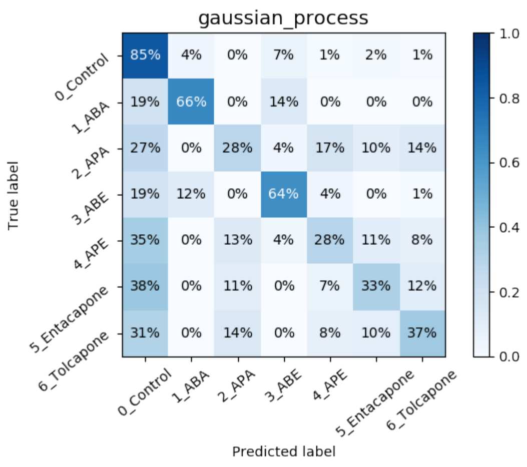


Figure B.180: Individuals *Mito* Gaussian Process CM

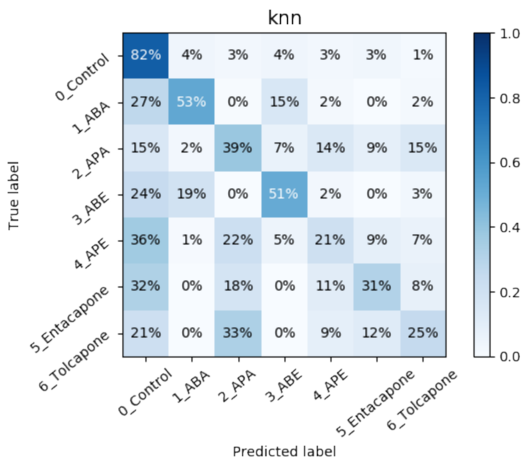


Figure B.181: Individuals *Mito* k-NN CM

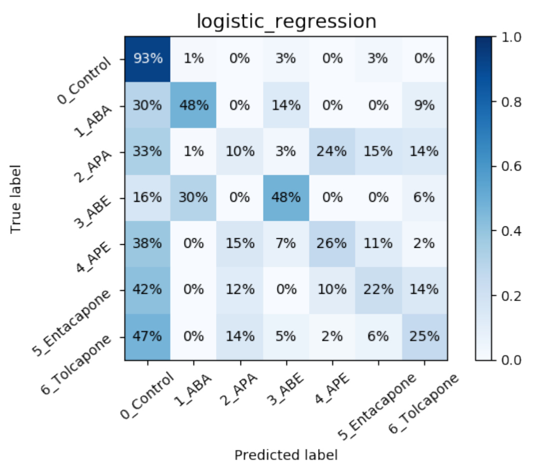


Figure B.182: Individuals *Mito* Logistic Regression CM

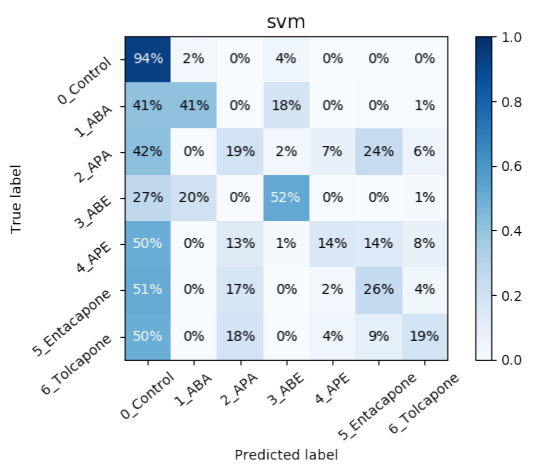


Figure B.183: Individuals *Mito* SVM CM

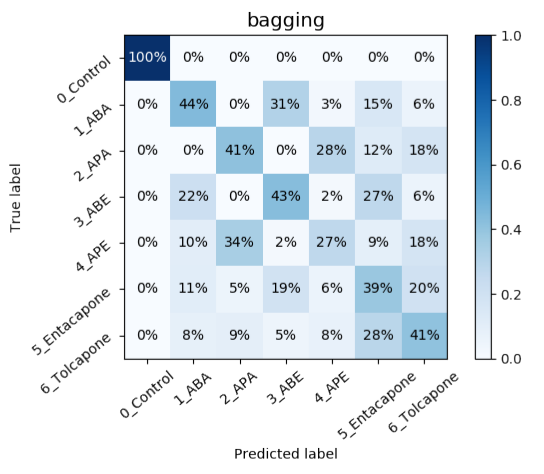


Figure B.184: Individuals *Glycolysis* Bagging CM

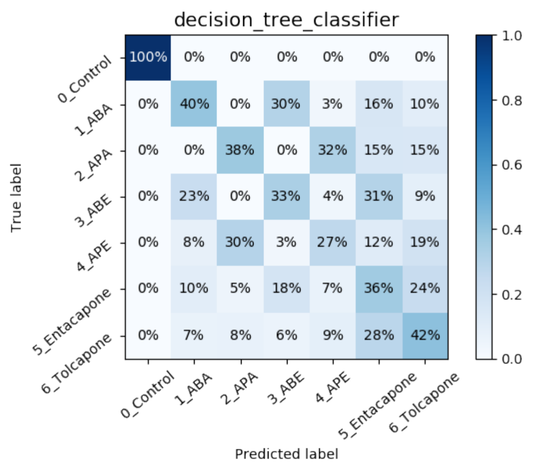


Figure B.185: Individuals *Glycolysis* DT CM

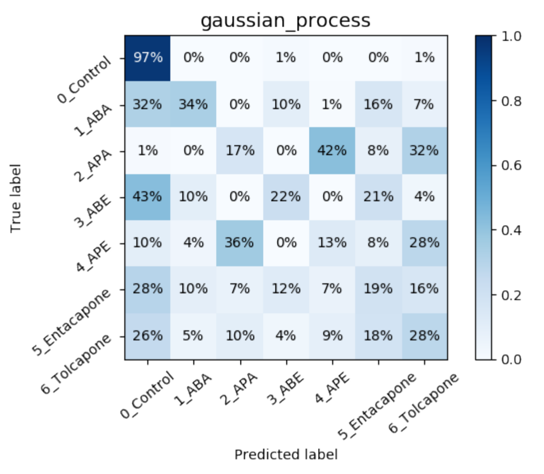


Figure B.186: Individuals *Glycolysis* Gaussian Process CM

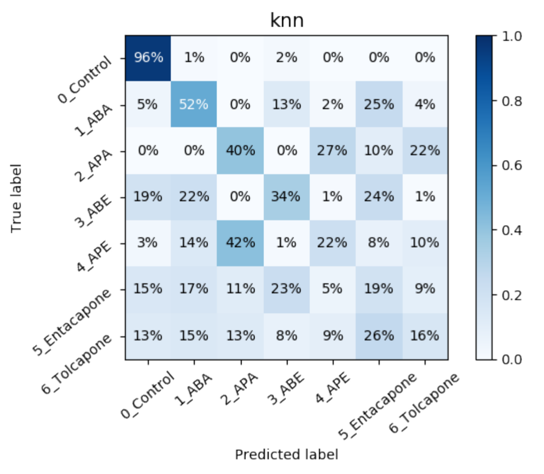
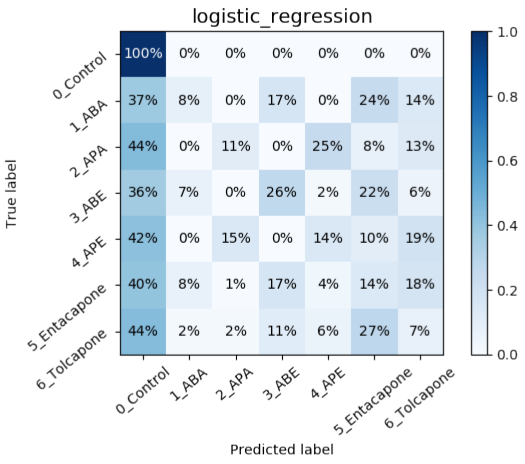
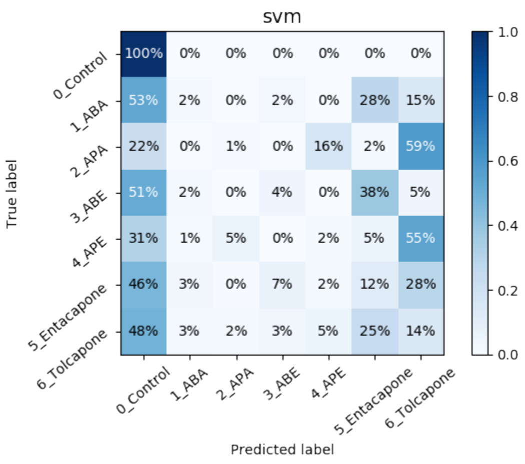


Figure B.187: Individuals *Glycolysis* k-NN CM



	0_Control	1_ABA	2_APA	3_ABE	4_APE	5_Entacapone	6_Tolcapone
0_Control	100%	0%	0%	0%	0%	0%	0%
1_ABA	37%	8%	0%	17%	0%	24%	14%
2_APA	44%	0%	11%	0%	25%	8%	13%
3_ABE	36%	7%	0%	26%	2%	22%	6%
4_APE	42%	0%	15%	0%	14%	10%	19%
5_Entacapone	40%	8%	1%	17%	4%	14%	18%
6_Tolcapone	44%	2%	2%	11%	6%	27%	7%

Figure B.188: Individuals *Glycolysis* Logistic Regression CM



	0_Control	1_ABA	2_APA	3_ABE	4_APE	5_Entacapone	6_Tolcapone
0_Control	100%	0%	0%	0%	0%	0%	0%
1_ABA	53%	2%	0%	2%	0%	28%	15%
2_APA	22%	0%	1%	0%	16%	2%	59%
3_ABE	51%	2%	0%	4%	0%	38%	5%
4_APE	31%	1%	5%	0%	2%	5%	55%
5_Entacapone	46%	3%	0%	7%	2%	12%	28%
6_Tolcapone	48%	3%	2%	3%	5%	25%	14%

Figure B.189: Individuals *Glycolysis* SVM CM

B.2.4 Grouped Drugs

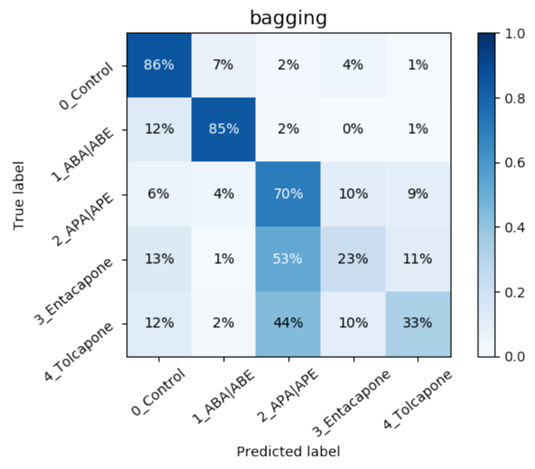


Figure B.190: Groups *Mito* Bagging CM

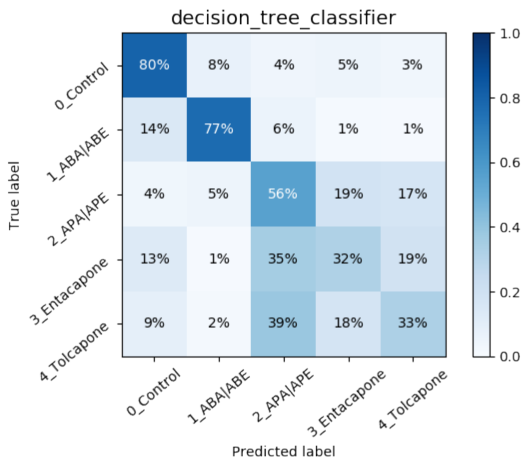


Figure B.191: Groups *Mito* DT CM

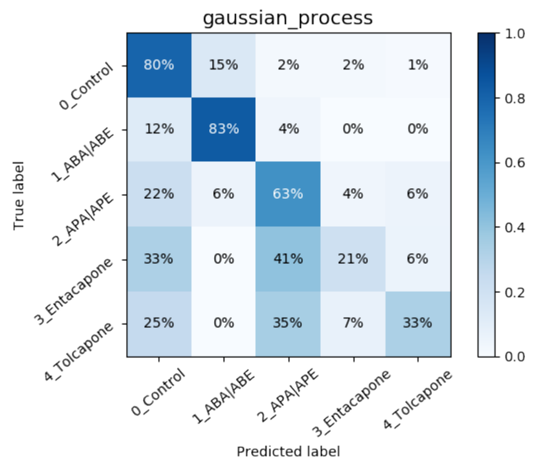


Figure B.192: Groups *Mito* Gaussian Process CM

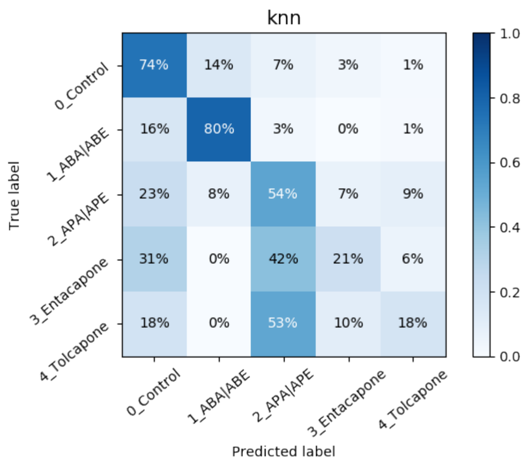


Figure B.193: Groups *Mito* k-NN CM

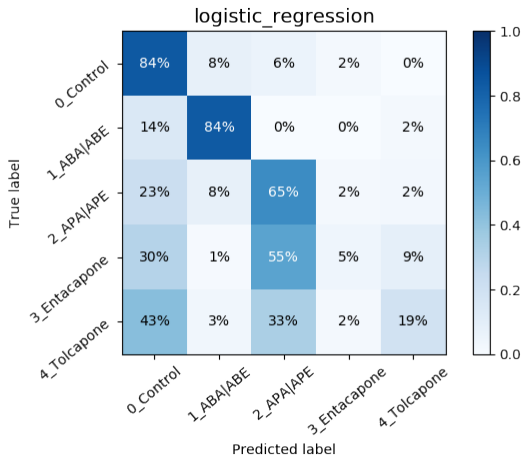


Figure B.194: Groups *Mito* Logistic Regression CM

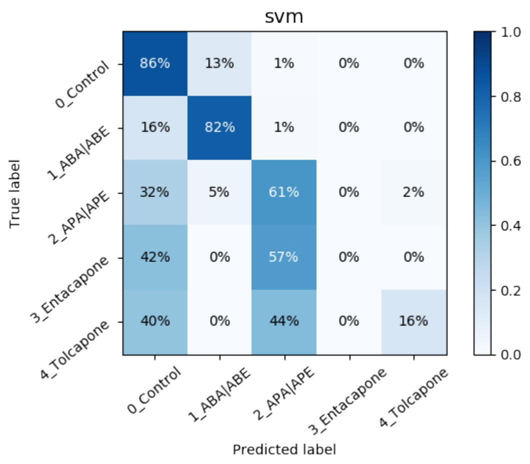


Figure B.195: Groups *Mito* SVM CM

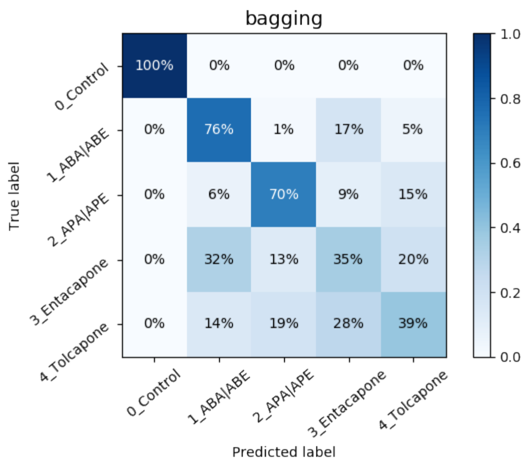


Figure B.196: Groups *Glycolysis* Bagging CM

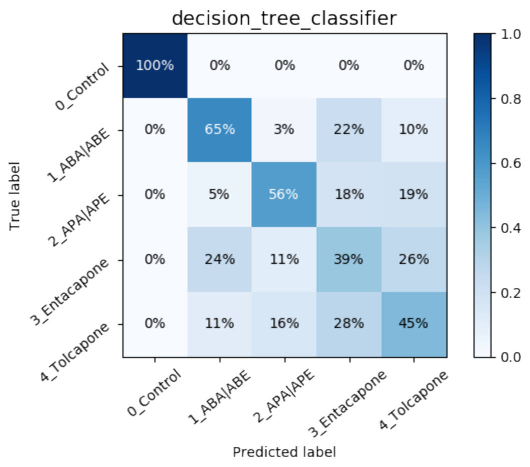


Figure B.197: Groups *Glycolysis* DT CM

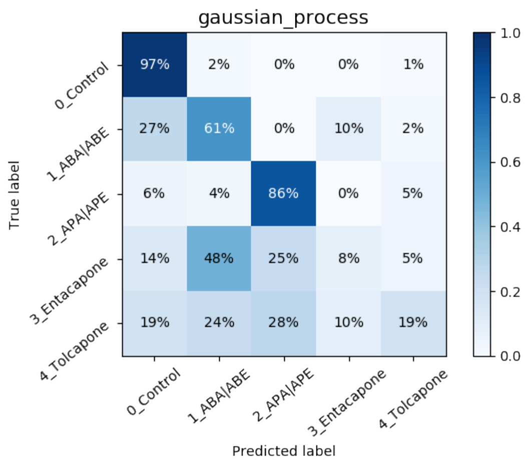


Figure B.198: Groups *Glycolysis* Gaussian Process CM

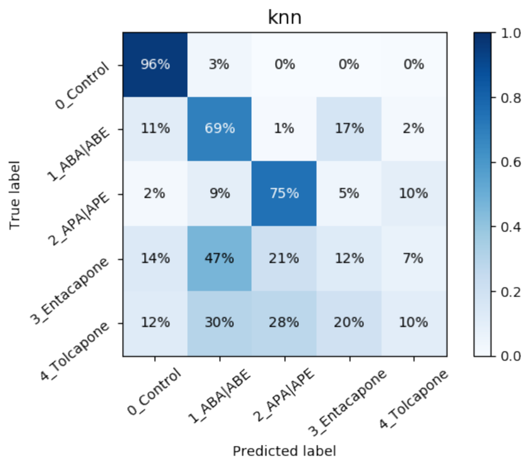


Figure B.199: Groups *Glycolysis* k-NN CM

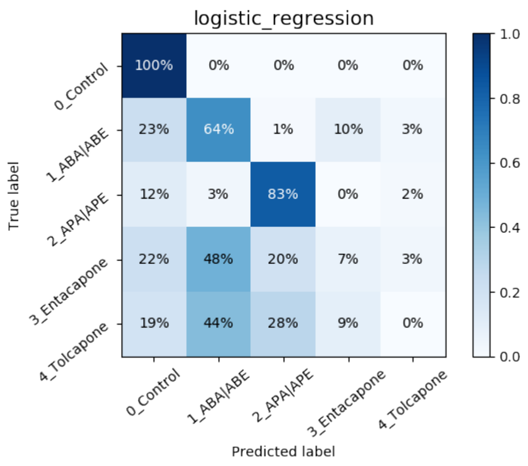


Figure B.200: Groups *Glycolysis* Logistic Regression CM

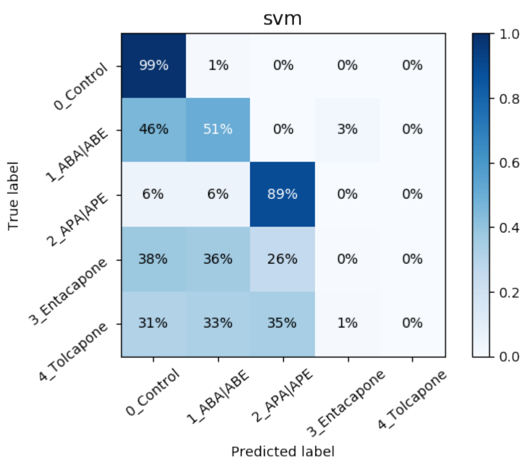


Figure B.201: Groups *Glycolysis* SVM CM

B.2.5 No Controls

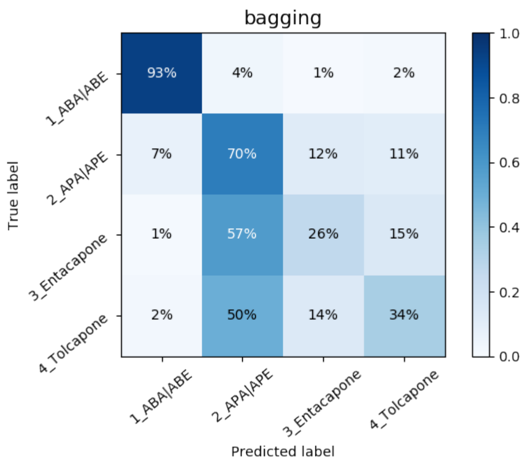


Figure B.202: No Controls *Mito* Bagging CM

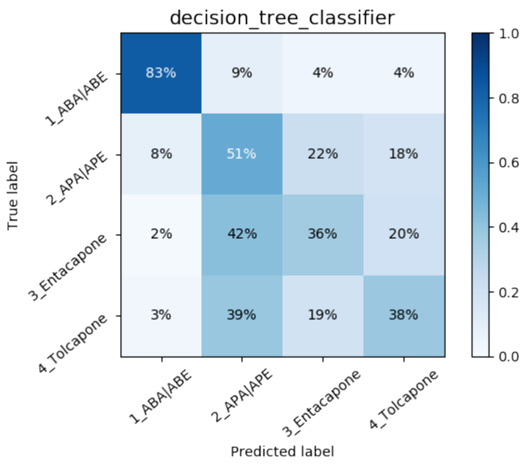


Figure B.203: No Controls *Mito* DT CM

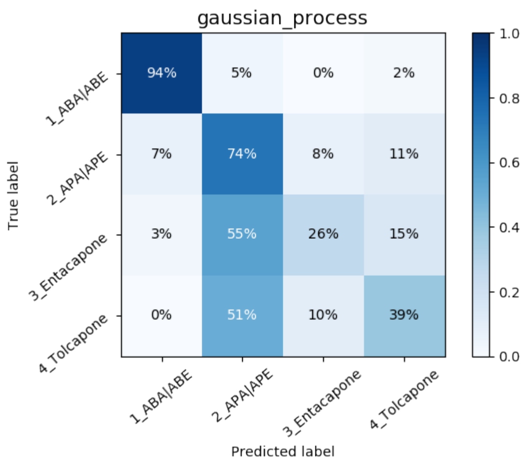


Figure B.204: No Controls *Mito* Gaussian Process CM

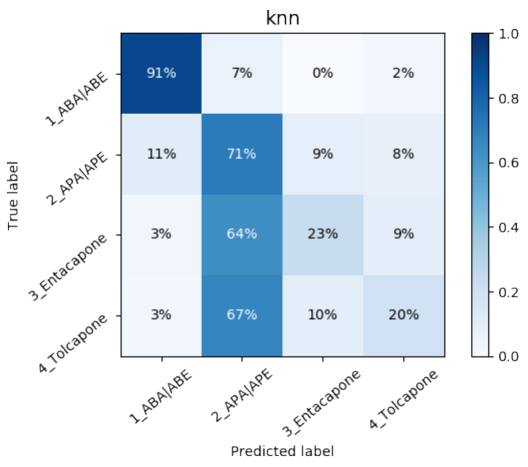


Figure B.205: No Controls *Mito* k-NN CM

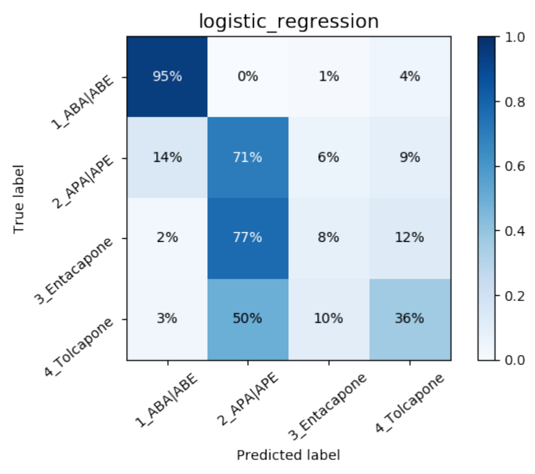


Figure B.206: No Controls *Mito* Logistic Regression CM

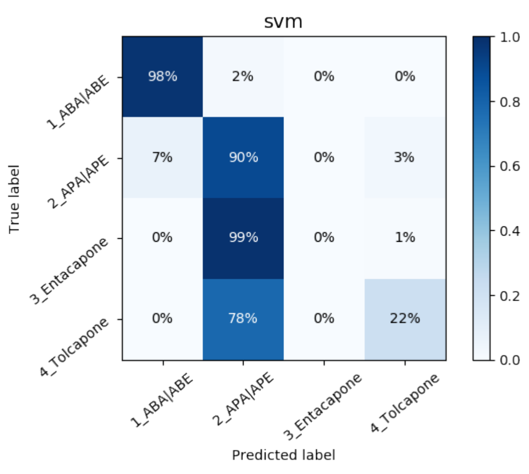


Figure B.207: No Controls *Mito* SVM CM

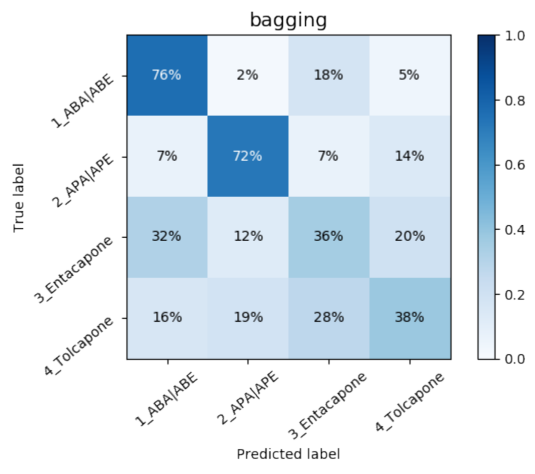


Figure B.208: No Controls *Glycolysis* Bagging CM

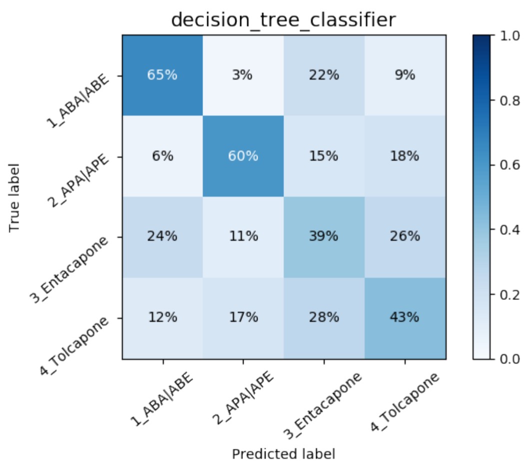


Figure B.209: No Controls *Glycolysis* DT CM

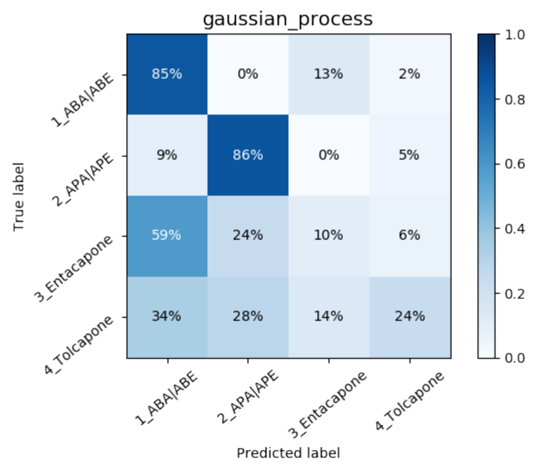


Figure B.210: No Controls *Glycolysis* Gaussian Process CM

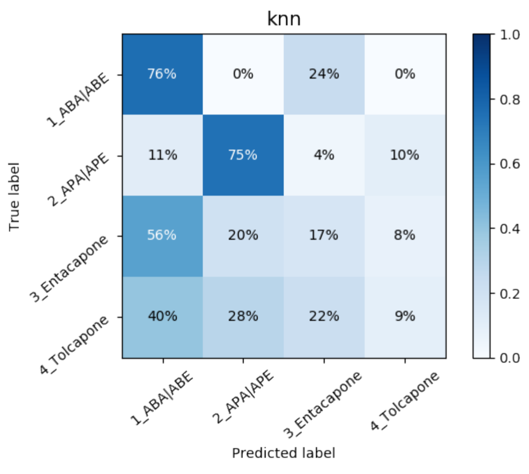


Figure B.211: No Controls *Glycolysis* k-NN CM

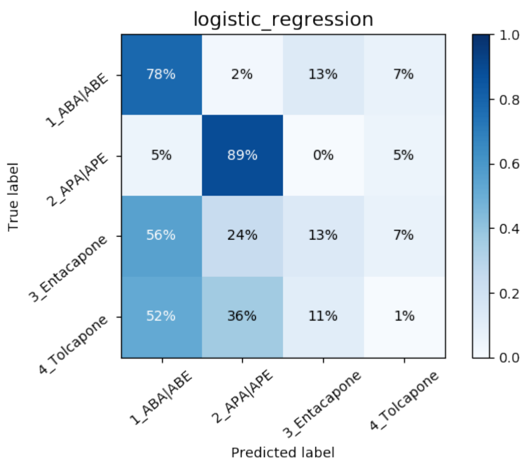


Figure B.212: No Controls *Glycolysis* Logistic Regression CM

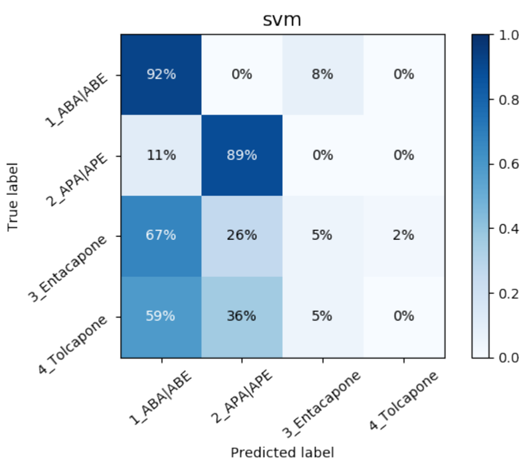


Figure B.213: No Controls *Glycolysis* SVM CM

Data Balancing

Random Undersampling

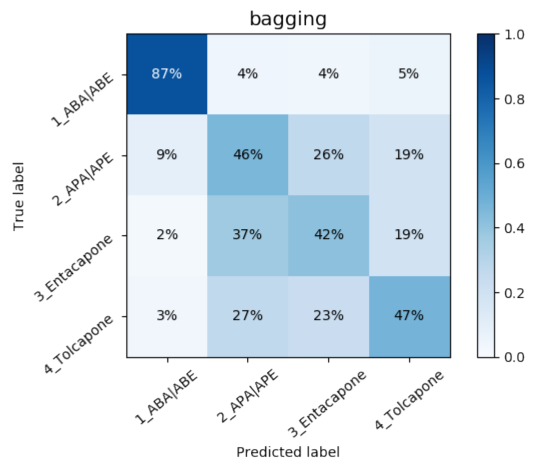


Figure B.214: No Controls *Mito* Undersamp. Bagging CM

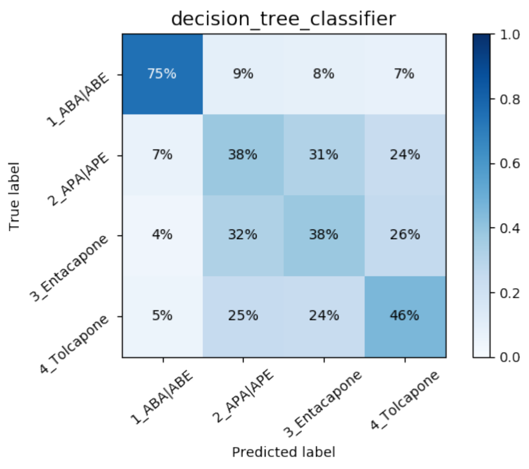


Figure B.215: No Controls *Mito* Undersamp. DT CM

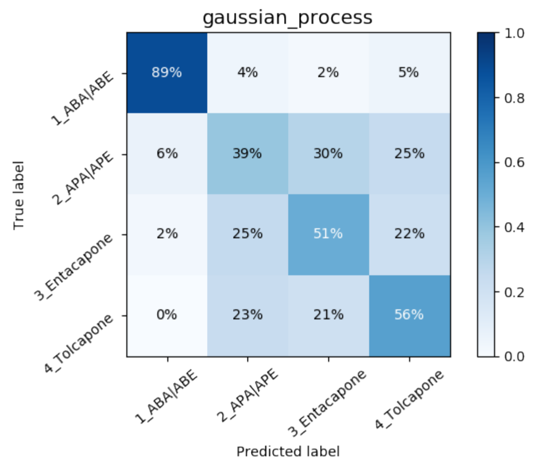


Figure B.216: No Controls *Mito* Undersamp. Gaussian Process CM

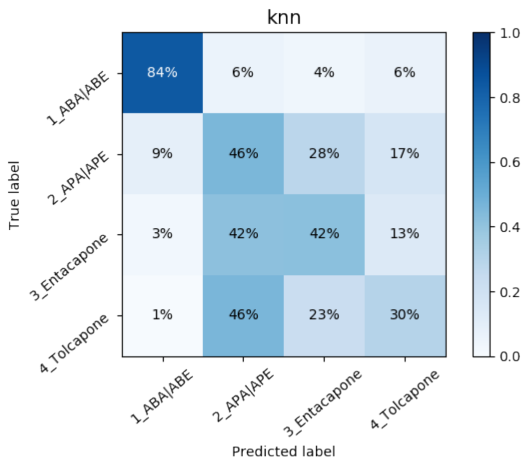


Figure B.217: No Controls *Mito* Undersamp. k-NN CM

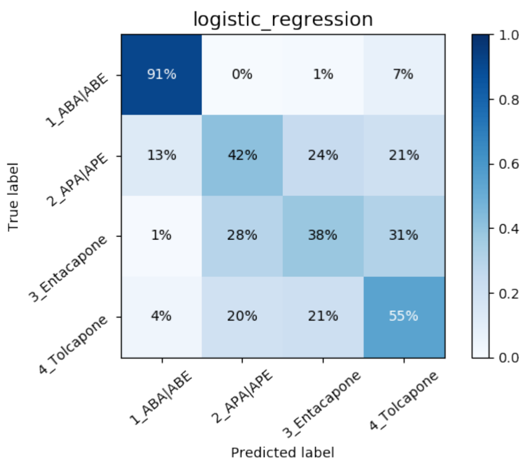


Figure B.218: No Controls *Mito* Undersamp. Logistic Regression CM

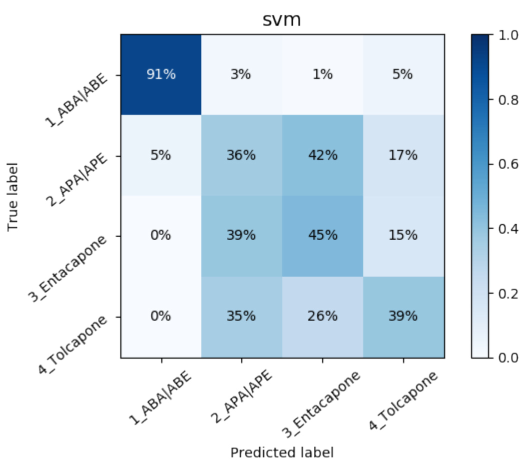


Figure B.219: No Controls *Mito* Undersamp. SVM CM

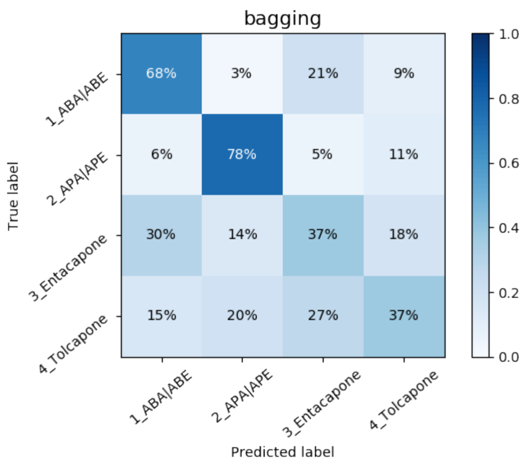


Figure B.220: No Controls *Glycolysis* Undersamp. Bagging CM

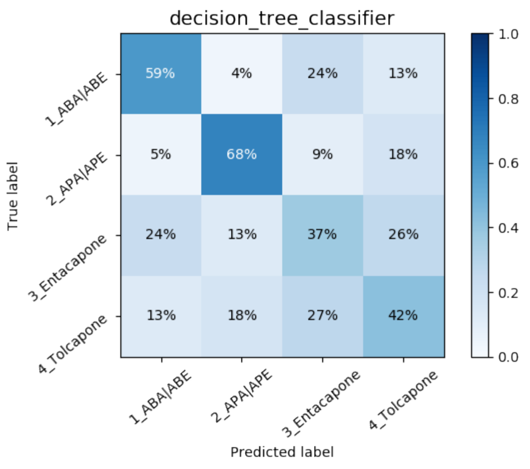


Figure B.221: No Controls *Glycolysis* Undersamp. DT CM

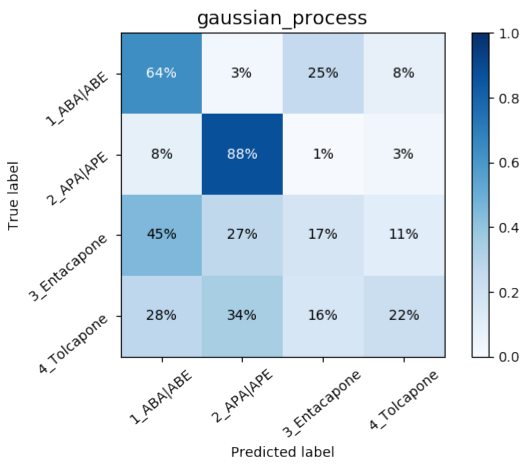


Figure B.222: No Controls *Glycolysis* Undersamp. Gaussian Process CM

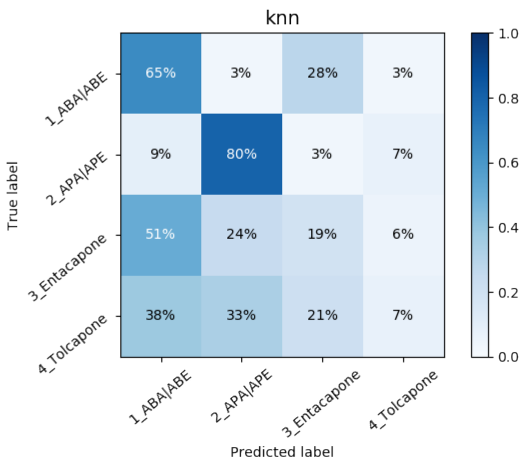
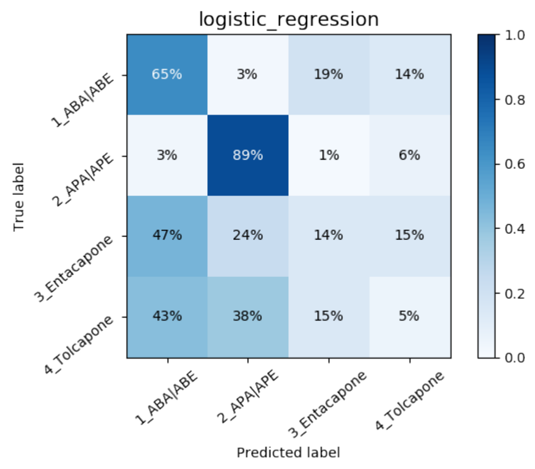


Figure B.223: No Controls *Glycolysis* Undersamp. k-NN CM



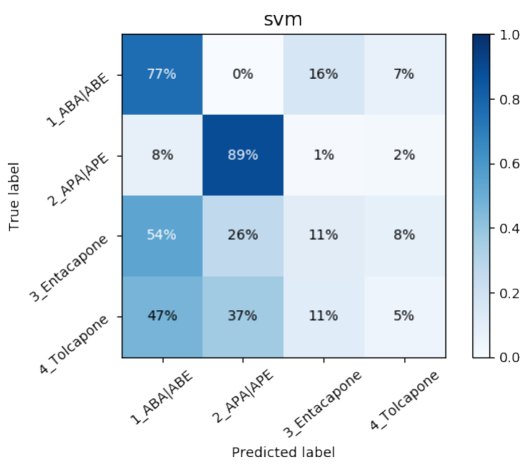
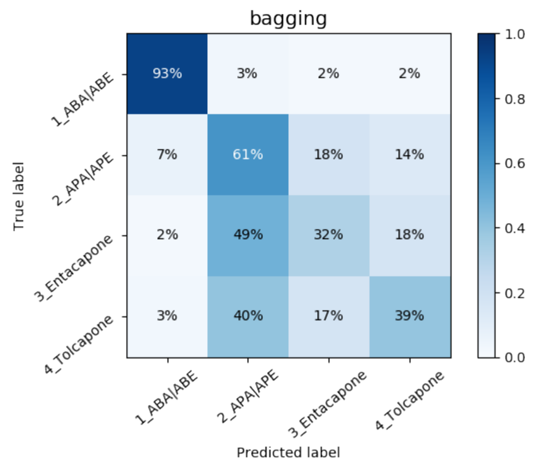


Figure B.224: No Controls *Glycolysis* Undersamp. Logistic Regression CM

Figure B.225: No Controls *Glycolysis* Undersamp. SVM CM

Random Oversampling



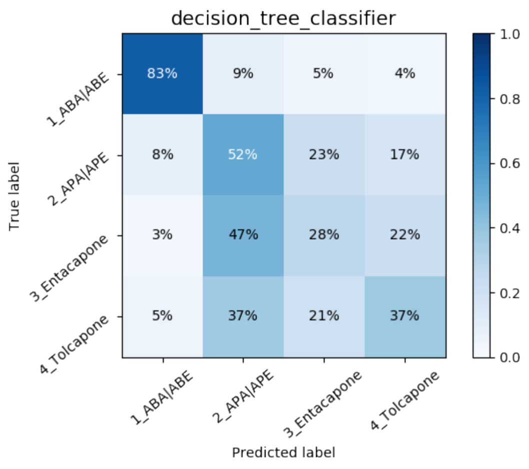


Figure B.226: No Controls *Mito* Oversamp. Bagging CM

Figure B.227: No Controls *Mito* Oversamp. DT CM

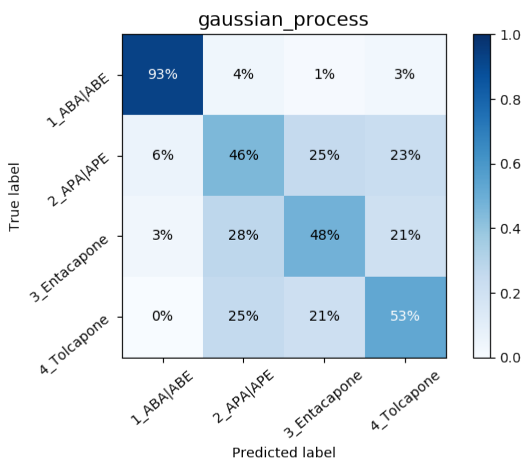


Figure B.228: No Controls *Mito* Oversamp. Gaussian Process CM

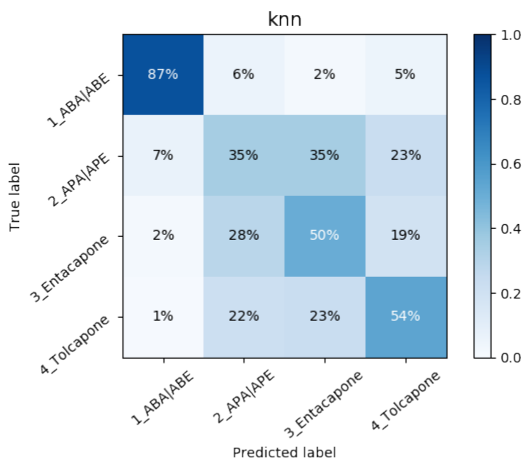


Figure B.229: No Controls *Mito* Oversamp. k-NN CM

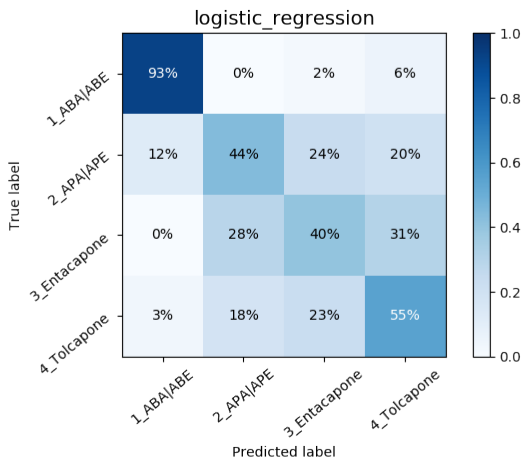


Figure B.230: No Controls *Mito* Oversamp. Logistic Regression CM

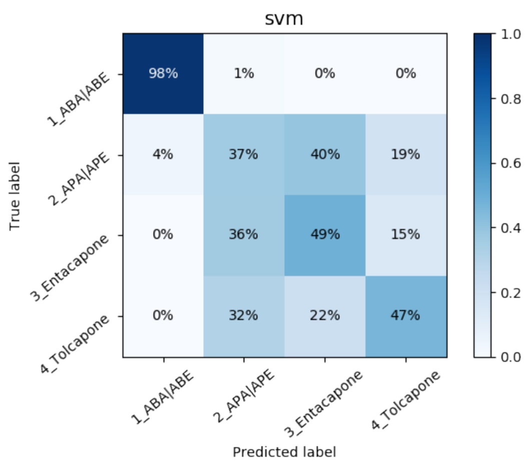


Figure B.231: No Controls *Mito* Oversamp. SVM CM

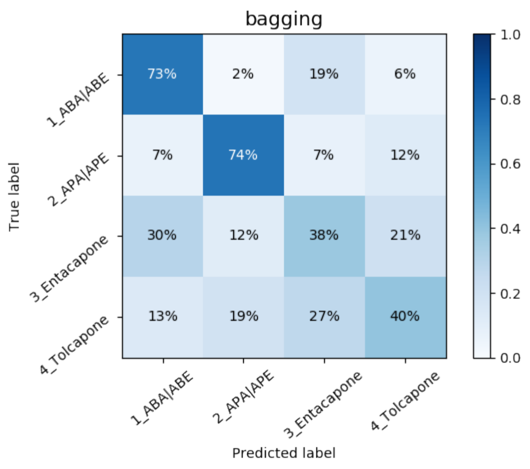


Figure B.232: No Controls *Glycolysis* Oversamp. Bagging CM

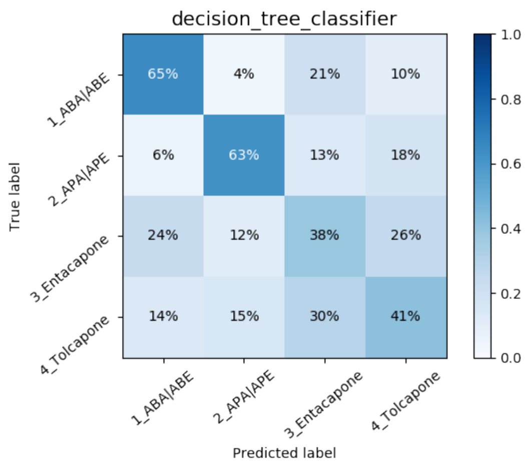


Figure B.233: No Controls *Glycolysis* Oversamp. DT CM

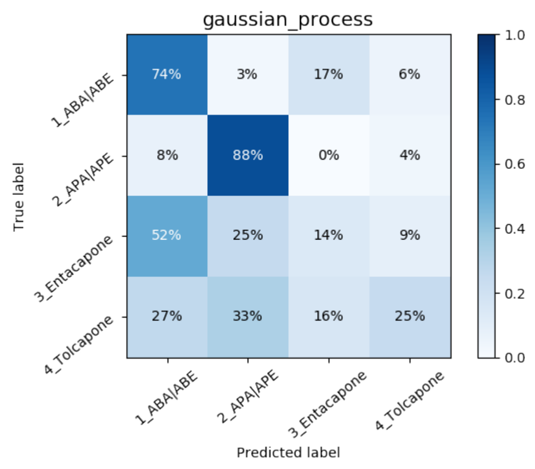


Figure B.234: No Controls *Glycolysis* Oversamp. Gaussian Process CM

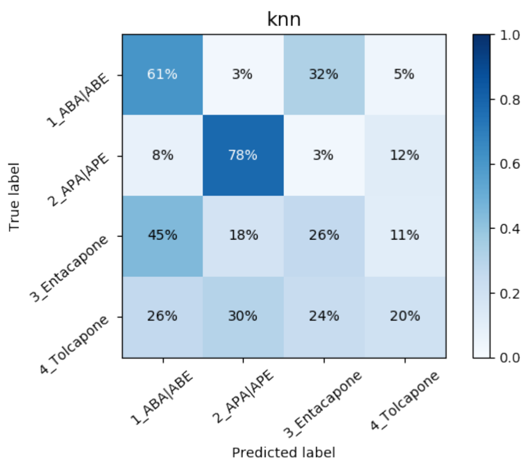


Figure B.235: No Controls *Glycolysis* Oversamp. k-NN CM

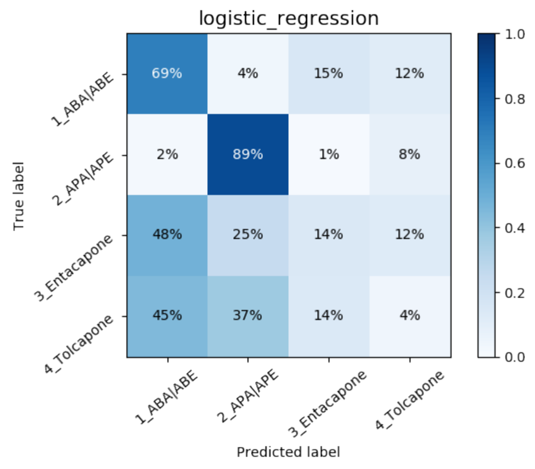


Figure B.236: No Controls *Glycolysis* Oversamp. Logistic Regression CM

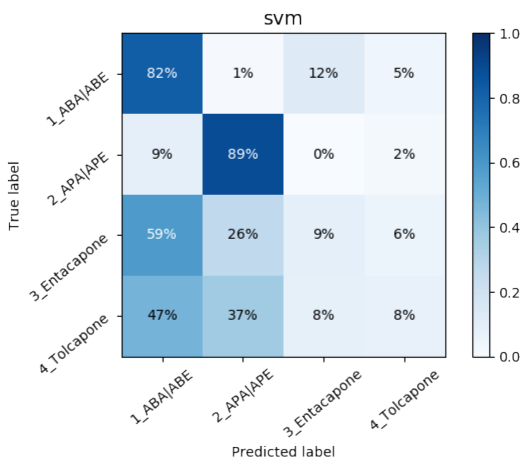


Figure B.237: No Controls *Glycolysis* Oversamp. SVM CM

Synthetic Minority Over-sampling Technique (SMOTE)

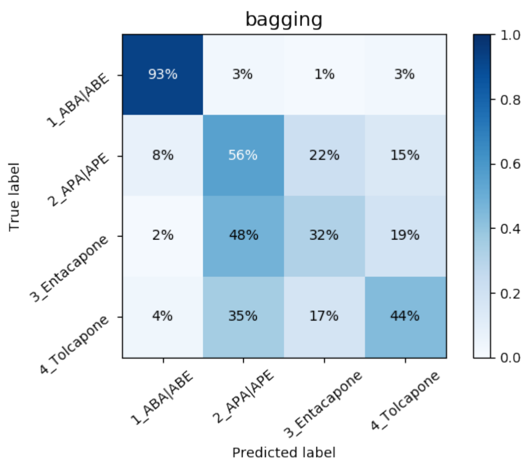


Figure B.238: No Controls *Mito* Oversamp. Bagging CM

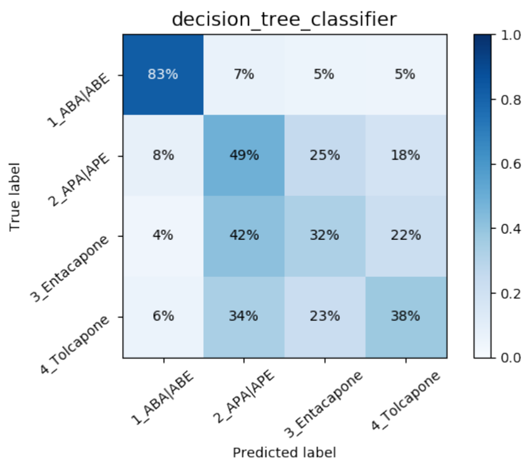


Figure B.239: No Controls *Mito* Oversamp. DT CM

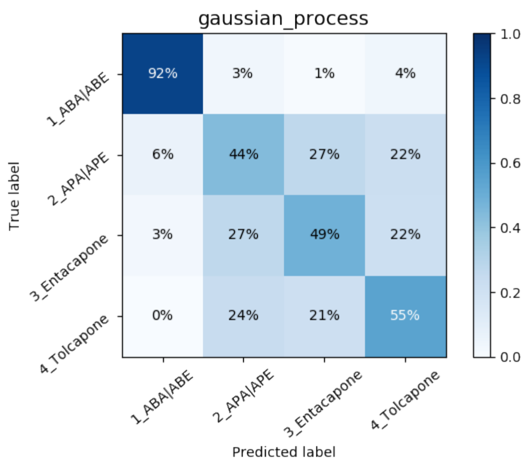


Figure B.240: No Controls *Mito* Oversamp. Gaussian Process CM

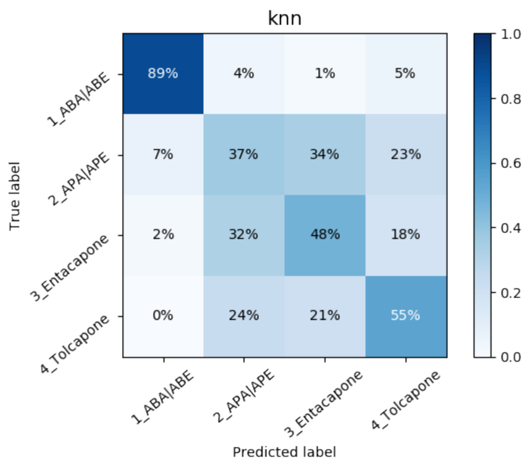


Figure B.241: No Controls *Mito* Oversamp. k-NN CM

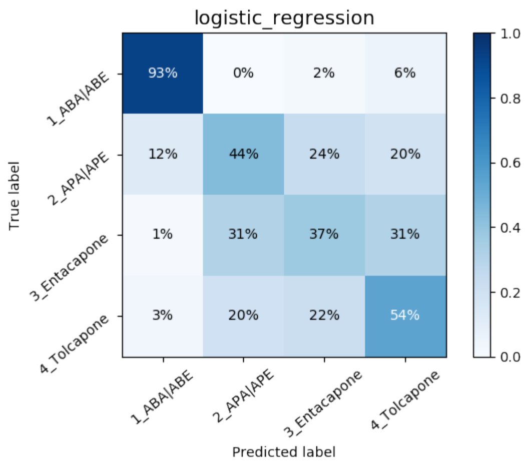


Figure B.242: No Controls *Mito* Oversamp. Logistic Regression CM

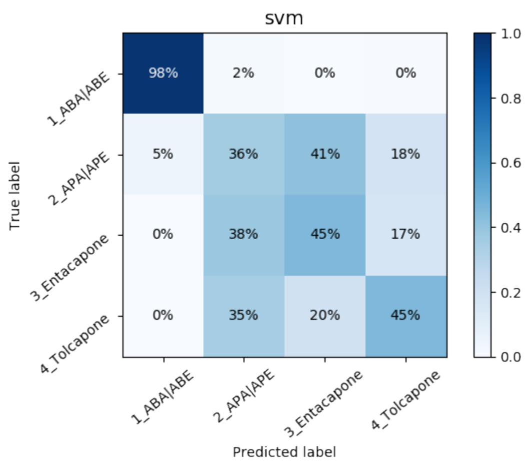


Figure B.243: No Controls *Mito* Oversamp. SVM CM

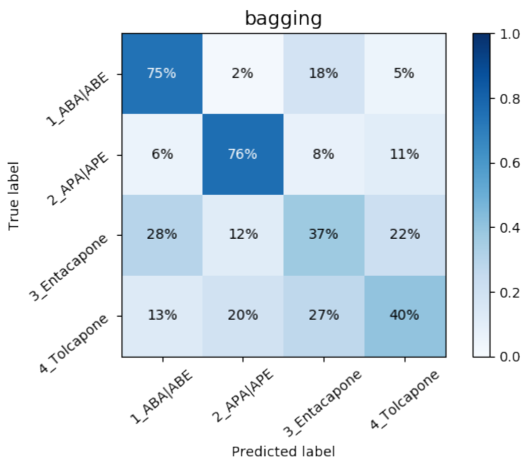


Figure B.244: No Controls *Glycolysis* Oversamp. Bagging CM

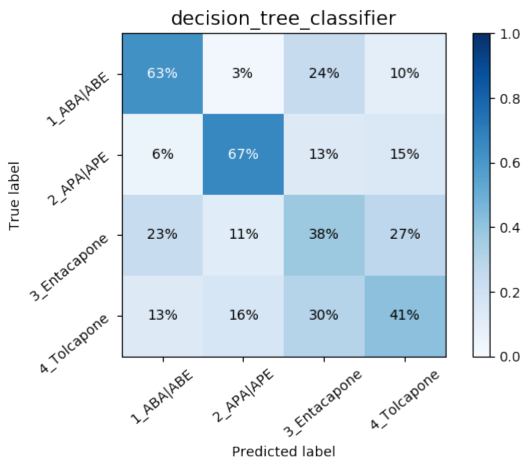


Figure B.245: No Controls *Glycolysis* Oversamp. DT CM

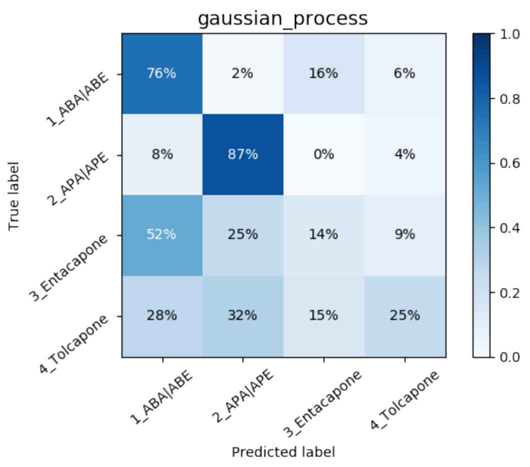


Figure B.246: No Controls *Glycolysis* Oversamp. Gaussian Process CM

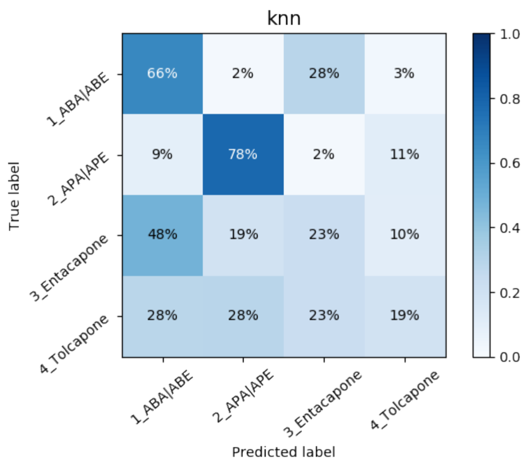


Figure B.247: No Controls *Glycolysis* Oversamp. k-NN CM

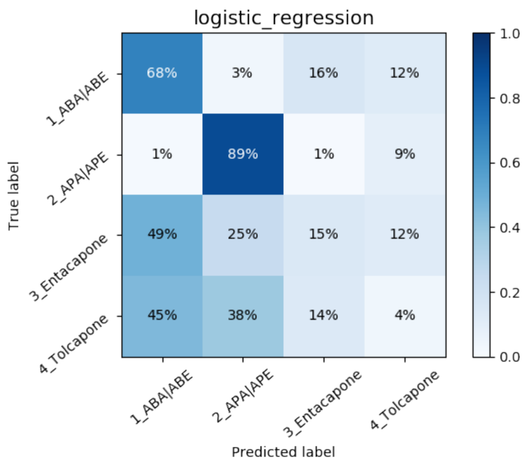


Figure B.248: No Controls *Glycolysis* Oversamp. Logistic Regression CM

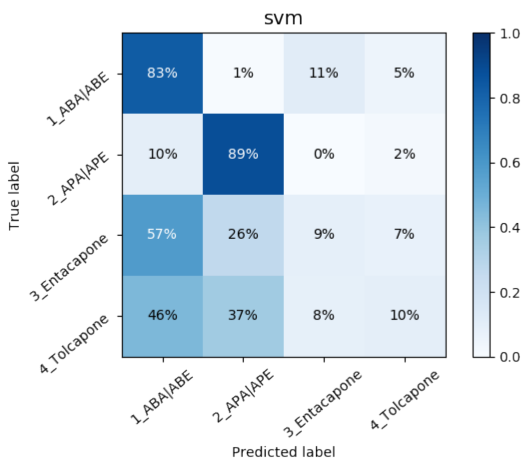


Figure B.249: No Controls *Glycolysis* Oversamp. SVM CM

B.2.6 Voting Ensemble

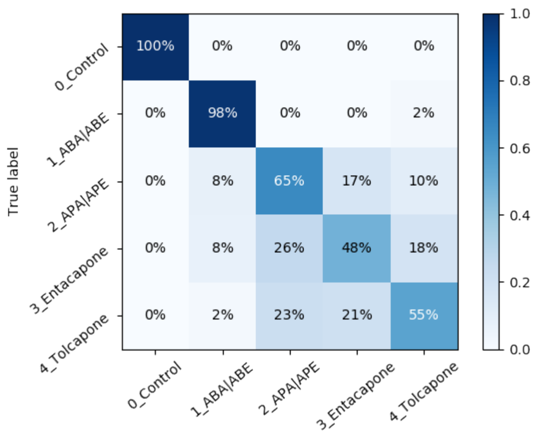


Figure B.250: Voting Ensemble #1 - NN, RF, Bag., Gau. Proc., Log. Reg., CM

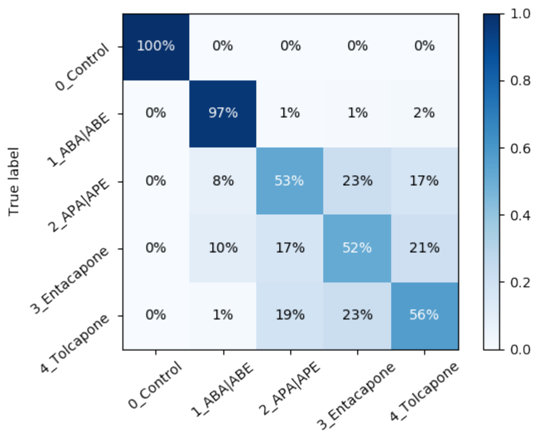


Figure B.251: Voting Ensemble #2 - NN, R, Bag., Gau. Proc., CM

B.2.7 Expert Ensemble

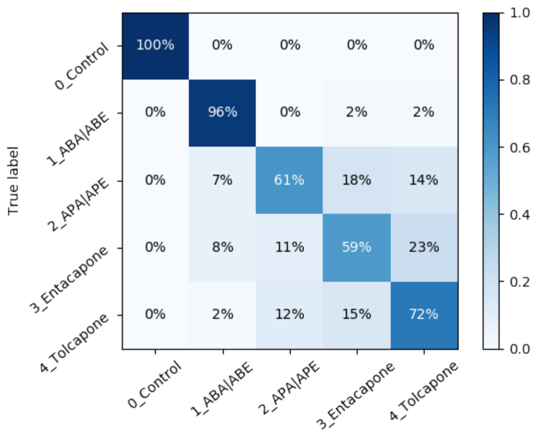


Figure B.252: Expert Ensemble #1 Equal Weights - Bag., NN, RF, DT, NN, Gau. Proc., SVM CM

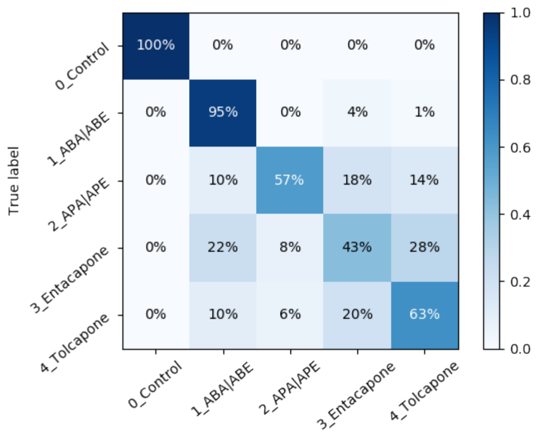


Figure B.253: Expert Ensemble #2 Equal Weights - Bag., NN, RF, DT, Gau. Proc., Gau. Proc. CM

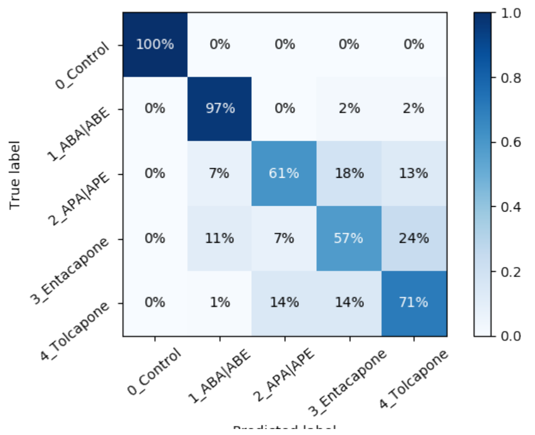


Figure B.254: Expert Ensemble #1 - Bag., NN, RF, DT, NN, Gau. Proc., SVM, 1.0, 0.4, 0.4 CM

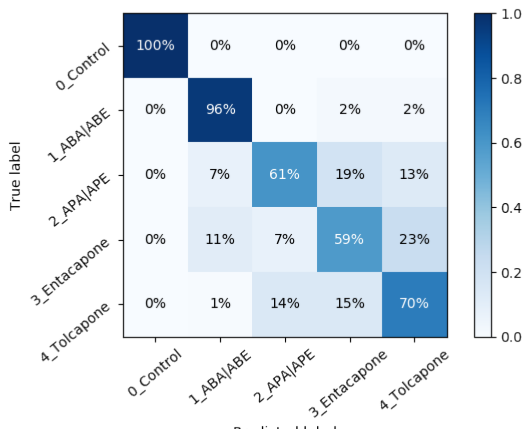


Figure B.255: Expert Ensemble #2 - Bag., NN, RF, DT, NN, Gau. Proc., SVM, 1.0, 0.4, 2.8 CM

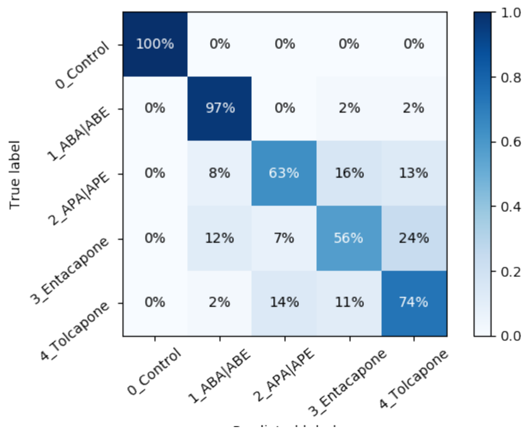


Figure B.256: Expert Ensemble #3 - Bag., NN, RF, DT, NN, Gau. Proc., SVM, 1.0, 1.2, 0.0 CM

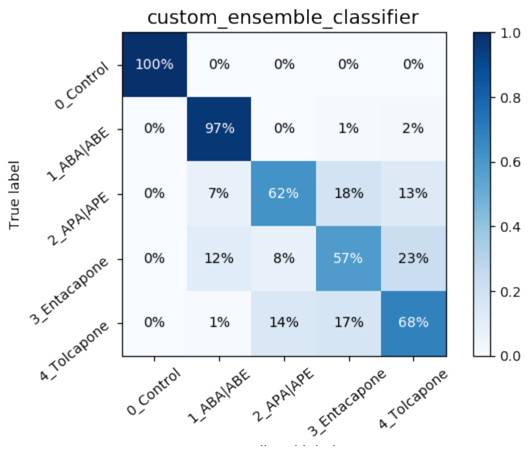


Figure B.257: Expert Ensemble #4 - Bag., NN, RF, DT, NNm Gau. Proc., SVM, 1.8, 0.2, 1.0 CM

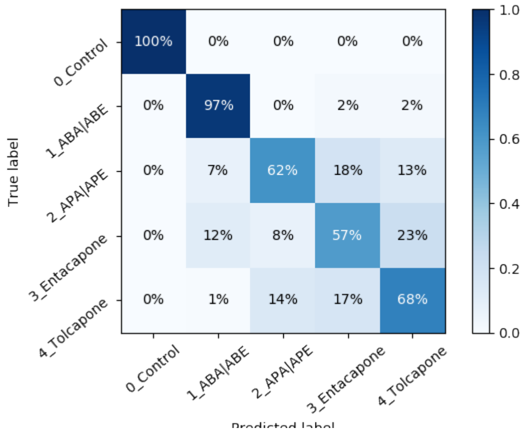


Figure B.258: Expert Ensemble #4 - Bag., NN, RF, DT, NNm Gau. Proc., SVM, 1.8, 0.2, 3.0 CM

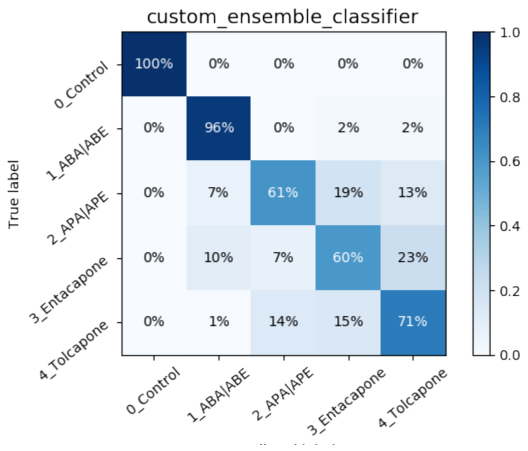
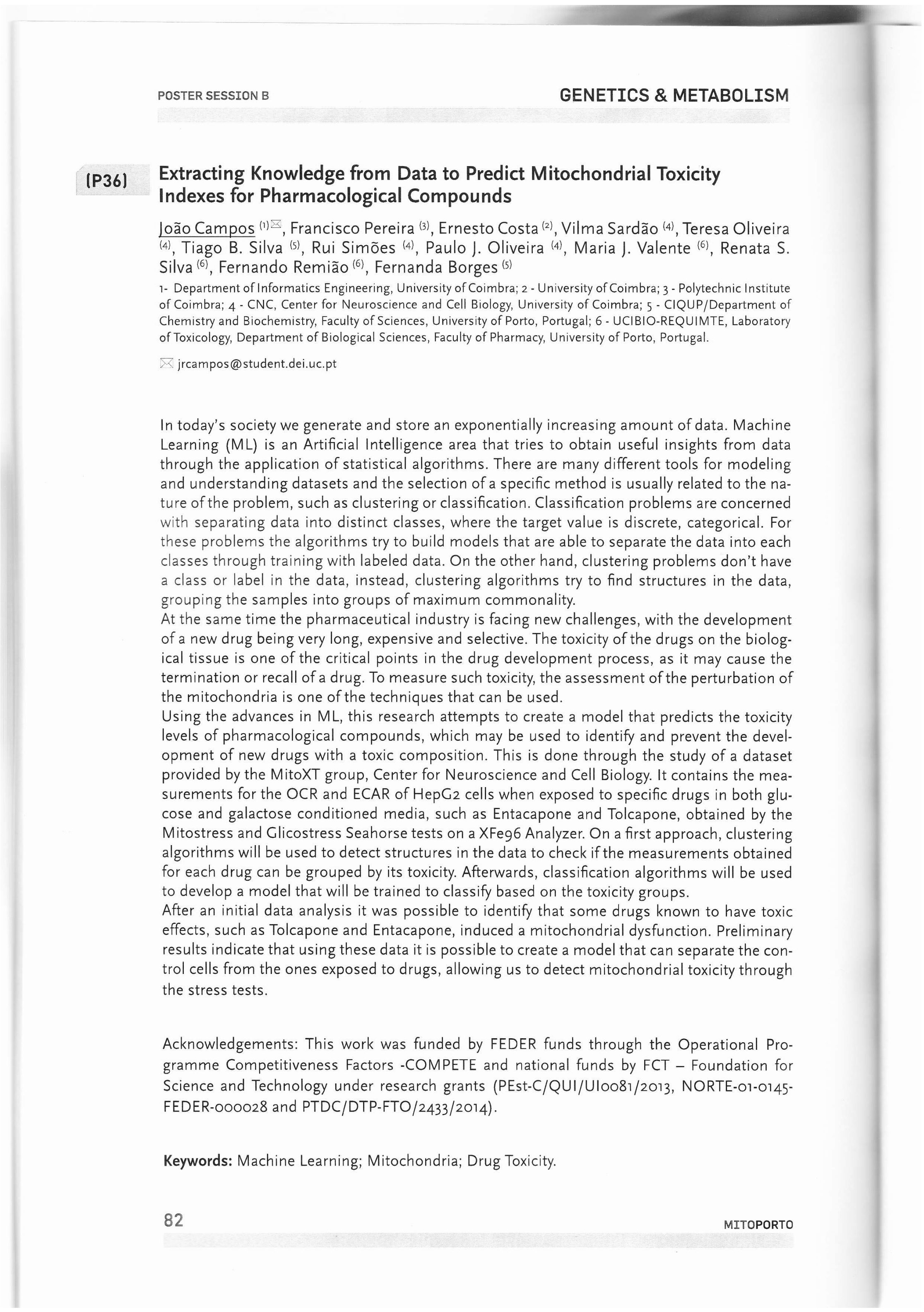


Figure B.259: Expert Ensemble #4 - Bag., NN, RF, DT, NNm Gau. Proc., SVM, 1.8, 1.0, 3.0 CM

B.2.8 MitoPorto



POSTER SESSION B GENETICS & METABOLISM

(P36) **Extracting Knowledge from Data to Predict Mitochondrial Toxicity Indexes for Pharmacological Compounds**

João Campos (1)✉, Francisco Pereira (3), Ernesto Costa (2), Vilma Sardão (4), Teresa Oliveira (4), Tiago B. Silva (5), Rui Simões (4), Paulo J. Oliveira (4), Maria J. Valente (6), Renata S. Silva (6), Fernando Remião (6), Fernanda Borges (5)

1- Department of Informatics Engineering, University of Coimbra; 2 - University of Coimbra; 3 - Polytechnic Institute of Coimbra; 4 - CNC, Center for Neuroscience and Cell Biology, University of Coimbra; 5 - CIQUP/Department of Chemistry and Biochemistry, Faculty of Sciences, University of Porto, Portugal; 6 - UCIBIO-REQUIMTE, Laboratory of Toxicology, Department of Biological Sciences, Faculty of Pharmacy, University of Porto, Portugal.

✉ jrcampos@student.dei.uc.pt

In today's society we generate and store an exponentially increasing amount of data. Machine Learning (ML) is an Artificial Intelligence area that tries to obtain useful insights from data through the application of statistical algorithms. There are many different tools for modeling and understanding datasets and the selection of a specific method is usually related to the nature of the problem, such as clustering or classification. Classification problems are concerned with separating data into distinct classes, where the target value is discrete, categorical. For these problems the algorithms try to build models that are able to separate the data into each classes through training with labeled data. On the other hand, clustering problems don't have a class or label in the data, instead, clustering algorithms try to find structures in the data, grouping the samples into groups of maximum commonality.

At the same time the pharmaceutical industry is facing new challenges, with the development of a new drug being very long, expensive and selective. The toxicity of the drugs on the biological tissue is one of the critical points in the drug development process, as it may cause the termination or recall of a drug. To measure such toxicity, the assessment of the perturbation of the mitochondria is one of the techniques that can be used.

Using the advances in ML, this research attempts to create a model that predicts the toxicity levels of pharmacological compounds, which may be used to identify and prevent the development of new drugs with a toxic composition. This is done through the study of a dataset provided by the MitoXT group, Center for Neuroscience and Cell Biology. It contains the measurements for the OCR and ECAR of HepG2 cells when exposed to specific drugs in both glucose and galactose conditioned media, such as Entacapone and Tolcapone, obtained by the Mitostress and Glicostress Seahorse tests on a XFe96 Analyzer. On a first approach, clustering algorithms will be used to detect structures in the data to check if the measurements obtained for each drug can be grouped by its toxicity. Afterwards, classification algorithms will be used to develop a model that will be trained to classify based on the toxicity groups.

After an initial data analysis it was possible to identify that some drugs known to have toxic effects, such as Tolcapone and Entacapone, induced a mitochondrial dysfunction. Preliminary results indicate that using these data it is possible to create a model that can separate the control cells from the ones exposed to drugs, allowing us to detect mitochondrial toxicity through the stress tests.

Acknowledgements: This work was funded by FEDER funds through the Operational Programme Competitiveness Factors -COMPETE and national funds by FCT – Foundation for Science and Technology under research grants (PEst-C/QUI/UI0081/2013, NORTE-01-0145-FEDER-000028 and PTDC/DTP-FTO/2433/2014).

Keywords: Machine Learning; Mitochondria; Drug Toxicity.

82 MITOPORTO

Figure B.260: MitoPorto

Extracting Knowledge from Data to Predict Mitochondrial Toxicity Indexes for Pharmacological Compounds

João R. Campos[1], Francisco Pereira [2,3], Ernesto Costa[1,2], Vilma Sardão[4], Teresa Oliveira[4], Tiago B. Silva[5], Rui Simões[4], Paulo J. Oliveira[4], Maria J. Valente[6], Renata S. Silva[6], Fernando Remião[6], Fernanda Borges[5]

[1]University of Coimbra; [2]CISUC, Department of Informatics Engineering, University of Coimbra; [3]Polytechnic Institute of Coimbra; [4]CNC, Center for Neuroscience and Cell Biology, University of Coimbra; [5]CIQUP/Department of Chemistry and Biochemistry, University of Porto; [6]UCIBIO-REQUIMTE, Laboratory of Toxicology, Department of Biological Sciences, University of Porto

Abstract

The toxicity of the drugs on the biological tissue is one of the critical points in the drug development process. To measure such toxicity, the assessment of the perturbation of the mitochondria is one of the techniques that can be used. Using the advances in Machine Learning, this research attempts to create a model that predicts the toxicity levels of pharmacological compounds through clustering and classification algorithms. The results indicate that using these data it is possible to create a model that can detect mitochondrial toxicity, allowing it to separate the control cells from the ones exposed to drugs.

Introduction

In today's society we generate and store an exponentially increasing amount of data. Machine Learning (ML) is an Artificial Intelligence area that tries to obtain useful insights from data through the application of statistical algorithms. At the same time the pharmaceutical industry is facing new challenges, with the development of a new drug being very long, expensive and selective. The toxicity of the drugs on the biological tissue is one of the critical points in the drug development process, as it may cause the termination or recall of a drug. To measure such toxicity, the assessment of the perturbation of the mitochondria is one of the techniques that can be used. Using the advances in ML, this research attempts to create a model that predicts the toxicity levels of pharmacological compounds, which may be used to identify and prevent the development of new drugs with a toxic composition.

Data Analysis

For this work a dataset was provided by the MitoXT group, Center for Neuroscience and Cell Biology. It contains the measurements for the OCR and ECAR of HepG2 cells when exposed to specific drugs in both glucose and galactose conditioned media, obtained by the Mito Stress and Glycolysis Stress Seahorse tests on a XFe96 Analyzer. The drugs tested include some to have known toxic effects, such as Entacapone and Tolcapone, and the newly developed CNCAPA, CNCABA, CNCAPE and CNCABE[1]. The measurements of the effect of each compound added in the tests are stored, as depicted in *Figure 1 (a)* and *(b)*, resulting in 4 measurements of OCR and ECAR for each well.

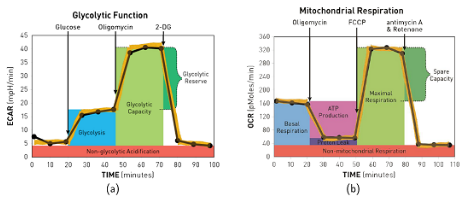


Figure 1: (a): Glycolytic Function[2] (b): Mitochondrial Respiration[2]

In total there were 3 Mito Stress and 4 Glycolysis Stress tests, adding up to 8 experiments for each combination of drug concentrations (10 and 50 μM) and media (glucose and galactose). After performing a descriptive analysis it was possible to observe the average effects that each drug has on the measurements. Figure 2 (a) and *(b)* contains the average measures for the drugs with a concentration of 50 μM in a glucose medium for both tests. As it is possible to observe in *Figure 2 (b)* Tolcapone strongly interferes with the mitochondrial function.

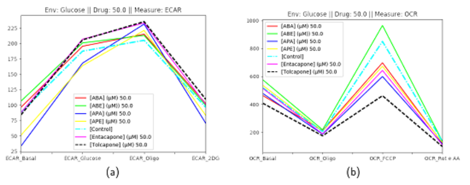


Figure 2: (a): Glycolysis Stress Averages (b): Mito Stress Averages

Machine Learning

There are many different tools for modeling and understanding datasets and the selection of a specific method is usually related to the nature of the problem, such as clustering or classification. Classification problems are concerned with separating data into distinct classes, where the target value is discrete, categorical. For these problems the algorithms try to build models that are able to separate the data into each classes through training with labeled data. On the other hand, clustering problems don't have a class or label in the data, instead, clustering algorithms try to find structures in the data, grouping the samples into groups of maximum commonality.

In ML the basic requirement of the data is that it contains enough detail to describe the scenario it represents and its outcome. A standard structure is a simple table where the columns are divided into a set of descriptive features and a target, and each row represents an instance. For this dataset the features used were: the type of test, the medium, the dosage and both the OCR and ECAR measures. To develop a ML predictor it is necessary to first create the model, and then train it to fit the problem data. Afterwards it must be evaluated to measure how well it can predict new data, usually by splitting the dataset in the training and the test set.

Algorithms

Although both classification and clustering approaches were made, classification yielded better results. Using multiple algorithms for classification the first step was to classify each experiment as either Control, Entacapone, Tolcapone or one of the new drugs. Using the measured values directly the algorithms managed to correctly classify most of the controls, however they were not able to satisfactorily predict the cells exposed to drugs. To minimize the variance between measurements the data from each test were analyzed separately and the indicators for each one, as described in *Figure 1 (a)* and *(b)*, were used. With this approach and with the data from the Glycolysis Stress test, the algorithms managed to identify all the control cells without any confusion with the ones exposed to drugs, as it can be observed in the Confusion Matrix (CM) in *Figure 3 (a)*.

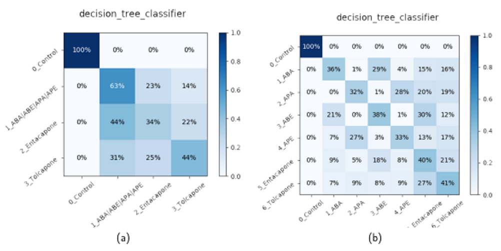


Figure 3: (a): Grouped Drugs DT CM - Step 1 (b): Individual Drugs DT CM - Step 2

Despite the better results the algorithms were still having difficulty in classifying the drugs. Thus, instead of separating the new drugs from the rest, a model was created to predict each drug separately. As it is possible to see in *Figure 3 (b)* there is confusion in the pairs CNCABA and CNCABE, as well as with CNCAPA and CNCAPE. This indicates that these pairs of drugs most likely induce similar changes on the metabolic system, hence the difficulty in distinguishing them. By grouping the identified pairs of drugs the algorithms started being able to correctly classify them, as it is can be seen in *Figure 4 (a)*

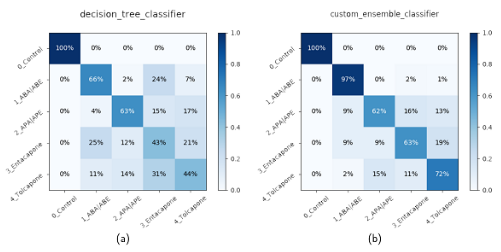


Figure 4: (a): Paired Drugs DT CM - Step 3 (b): Expert Ensemble CM

Although with the Glycolysis Stress test data the algorithms were able to entirely identify the controls, with the data from the Mito Stress test they were able to better distinguish between the drugs. Hence, an ensemble combining the the best algorithms was developed. Initially a simple version was implemented using only the majority voting of the predicted classes, which delivered good results. Nonetheless, a more complex version of the ensemble was developed, where each algorithm was given an expertise value for each class, giving more weight to the predicted classes by its expertise. The results improved further, as can be seen in *Figure 4 (b)*.

Conclusions

Using Machine Learning techniques to extract knowledge from biological data promises to give new insight to complex problems. With this dataset it was able to predict mitochondrial dysfunction and toxicity based on the data obtained from the Glycolysis and Mito Stress tests. The developed ensemble of algorithms was able to completely distinguish control cells from the ones exposed to drugs and acceptably separate the mitochondrial effects between them. Using such a model it is possible to test new drugs for mitochondrial toxicity.

References

1. Silva, T. *et al.* Development of bloodbrain barrier permeable nitrocatechol-based catechol o-methyltransferase inhibitors with reduced potential for hepatotoxicity. *Medical Chemistry* (2016). 2. Agilent technologies (2016). URL `http://www.agilent.com`.

Figure B.261: MitoPorto Poster

Appendix C

Technologies

The development of this project requires the use of several technologies. In order to choose which tools to use a research was made to gather all the pros and cons of each alternative. After weighting all the criteria that influenced the choice, the author chose Python as the programming language to be used. Some of the main reasons to choose it included the ease of usage, the availability of packages useful to the subject of this project as well the existing implementations and community for support. Within the Python environment several packages will be used, nonetheless these are the ones more related to subject:

- Pandas
- SciPy library
- Matplotlib
- Scikit
- NumPy

R was used to perform the all the statistical analysis.