# DETECTION OF ANTIBIOTIC RESISTANCE IN BACTERIA VIA MACHINE LEARNING APPROACHES

by

Hamdi Erkut

B.S., Computer Engineering, Boğaziçi University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2019

# ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis supervisor Assoc. Prof. Arzucan Özgür and co-supervisor Assoc. Prof. Elif Özkırımlı Ölmez for their support and guidance throughout my master's education. It has been a great honor for me to work with them while learning a great deal from them. My sincere gratitude to Assist. Prof. Fatma Başak Aydemir, Prof. Özlem Keskin Özkaya, and Prof. Ramazan Yıldırım for kindly accepting to participate in my thesis jury. I am very gladful to each member of TABI Lab for academic discussion and their precious feedbacks.

I am extremely grateful to my parents for their love, patience and sacrifices that they have done during my education. Also I want to express my thanks to my sister for her endless emotional support.

Last but not least, I would like to thank to my friends: Avni Madenüs, Ozan Genç, Eray Eren, Atakan Yüksel, Merve Yüce, Ekin Yurdakul, Mustafa Sertbaş for all chitchat.

The numerical calculations reported in this thesis were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

# ABSTRACT

## DETECTION OF ANTIBIOTIC RESISTANCE IN BACTERIA VIA MACHINE LEARNING APPROACHES

Tuberculosis, which is sometimes referred as white plague, is one of the most dangerous diseases caused by bacteria in our era. The species causing the sickness is in the family of Mycobacteriaceae and called mycobacterium tuberculosis. Bacteria are able to acquire resistance to antibiotics, so mortality rate among tuberculosis patients is increasing. This thesis examines different machine learning algorithms to detect antibiotic resistance to four first-line drugs in tuberculosis treatment. Variants on 23 target genes are included as input for each model. The base mycobacterium tuberculosis genome, which is used to detect variants on each sample in the data set, is the genome with id h37rv. Bacteria having h37rv as genome, are susceptible to all first-line antibiotics. Different machine learning algorithms are investigated and compared to each other. We observe that traditional machine learning algorithms have higher performance than multilayer perceptrons do. The impact of different data representations used in information retrieval on antibiotic resistance detection is also examined and we can not find any clear evidence for them to improve machine learning models' performances. Additionally, the contributions of mutations are ranked via the SHAP methodology used in the interpretation of machine learning models. We propose ten mutations with the highest SHAP values for each target drug as resistance determinants.

# ÖZET

## YAPAY ÖĞRENME YAKLAŞIMLARIYLA BAKTERİLERDE ANTİBİYOTİK DİRENÇ TESPİTİ

Beyaz veba olarak da anılan verem günümüzde en ölümcül bakteri kaynaklı hastalıklardan biridir. Bu hastalığa sebep olan tür Mycobacteriaceae ailesinden verem çubuk bakterisidir. Bakterilerin antibiyotiklere karşı direnç kazanabilmeleri yüzünden verem hastaları arasındaki ölüm oranı artmaktadır. Bu tez farklı yapay öğrenme algoritmalarının, verem tedavisinde kullanılan dört birinci sıra antibiyotiğe karşı geliştirilmiş direnç tespitinde kullanımını incelemektedir. Her bir model için 23 hedef gen üzerindeki değişiklikler girdi olarak kullanılmıştır. h37rv kimliğine sahip olan verem çubuk bakterisi genomu değişiklik tespitinde temel olarak alınmıştır. h37rv kimlikli kalıtsal materyale sahip olan bakteriler birinci sıra antibiyotiklerin hepsine duyarlıdır. Çeşitli yapay öğrenme algoritmaları incelendi ve birbiriyle kıyaslandı. Geleneksel modellerin performanslarının çok katmanlı algılayıcılardan daha yüksek olduğunu gözlemledik. Bilgi erişimi alanında kullanılan çeşitli veri gösterimlerinin antibiyotik direnç tespiti üzerindeki etkileri de incelenmiş ancak makine öğrenmesi modellerinin performanslarını arttıracak bir etkiye sahip olduğu çıkarımı yapılamamıştır. Ek olarak SHAP ismi verilen makine öğrenmesi anlamlandırma tekniği ile mutasyonların antibiyotik direnç tahminine katkıları incelenmiştir. Her bir hedef ilaç için en yüksek SHAP değerine sahip on mutasyonu direnç berlirleyici olarak ileri sürüyoruz.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| J | Cost function |
| $L_p$ | Lagrangian primal function |
| $L_d$ | Lagrangian dual function |
| $\Delta w$ | Gradients for w |
| $\sigma_{z_i}$ | Softmax for $z_i$ |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| ABR | Antibiotic Resistance |
| AMR | Antimicrobial Resistant |
| bagging | Bootstrap Aggregating |
| BM | Best Matching |
| DNA | Deoxyribonucleic Acid |
| FFNN | Feedforward Neural Network |
| HGT | Horizontal Gene Transfer |
| IDF | Inverse Document Frequency |
| INDEL | Insertion or Deletion of Bases |
| GATK | Genome Analysis Toolkit |
| MLP | Multilayer Perceptrons |
| MTB | Mycobacterium Tuberculosis |
| NCBI | National Center for Biotechnology Information |
| NPL | Natural Language Processing |
| RF | Relevance Frequency |
| SNP | Single-nucleotide Polymorphism |
| SVM | Support Vector Machine |
| TF | Term Frequency |
| VCF | Variant Call Format |
| XGBoost | Extreme Gradient Boosting |

# 1. INTRODUCTION

An antibiotic called penicillin was discovered by Sir Alexander Fleming in 1928. Penicillin was utilized to treat diseases derived from bacterial pathogens after its discovery. Many lives were saved from infected wounds during World War II thanks to penicillin. However, penicillin resistance has been developed by the pathogens over time and penicillin resistance has became a clinical issue [9]. Then a new kind of antibiotics, beta-lactam antibiotics, was discovered. Yet, again, bacteria eventually developed tolerance to beta-lactam antibiotics. This process has repeated itself with newly introduced antibiotics.

In addition to the tolerance developed by bacteria on their own, there is another way for them to gain resistance to antibiotics: gene transfer. Horizontal gene transfer is the basis for bacterial evolution. Bacteria are able to gain antibiotic resistance due to this evolutionary ability. Gene exchange provides bacteria with dynamic genomes. Consequently, pathogens' genomes should be observed continuously. Identification of antibiotic resistance would decrease both treatment time and cost by preventing improper treatment.

With the discovery of the first generation of antibiotics, antimicrobial chemotherapy has been the primary method of defense against diseases that are caused by bacteria. The method has saved millions of lives. However, the effectiveness of our fundamental fighting method against bacterial pathogens is decreasing over time due to overuse and misuse of antimicrobial drugs. Unfortunately, the mortality from illnesses caused by bacteria is currently increasing. Additionally, it is observed that the risk of acquiring infections caused by antimicrobial resistant pathogens is increasing. Doctors may refuse to conduct basic surgeries in the near future due to high probability of mortality caused by infections acquired in hospitals.

Antibiotic resistance also bears huge economic burdens on the society. According to some estimates, antimicrobial resistance costs to US health system between $21 billion and $34 billion dollars every year [10]. Another study from Thailand claimed that average hospitalization costs were increased from $108 to $528 [10] due to patients infected by antimicrobial resistant (AMR) pathogens.

Currently methods used to detect antibiotic resistance in pathogens require empirical antibiotic therapies that take days to identify the pathogens and their resistance to antibiotics [11]. To keep antibiotic resistance down, it is important to detect its roots properly and to use suitable antibiotic agents. The above mentioned detection methods cause a delay in treatment. However, it is possible to speed up antibiotic resistance detection via methodologies utilizing machine learning techniques and whole genome sequencing technologies.

While there are studies related to antimicrobial resistance in pathogens, they were mostly conducted from microbiology and biochemistry perspectives. The topic is still a relatively new area of research for computer scientists. Thanks to developments in multi-core processors and data analysis methodologies, it becomes possible to decrease both time and cost of the detection of antimicrobial resistance in pathogens.

In this thesis the contribution of machine learning methods to the study of antibiotic resistance detection in mycobacterium tuberculosis is examined. We investigate support vector machines (SVM) with linear and rbf kernel, logistic regression, random forests, xgboost and feedforward neurak networks. We also examine whether different data representations used in natural language processing domain such as TF-IDF, TF-RF, BM25TF-IDF, BM25TF-RF lead any improvements on model performances. Lastly we inspect the mutations that have the highest impact on the prediction of antibiotic resistance via machine learning interpretation methods. By doing so, we aim to detect mutations that lead to antibiotic resistance in tuberculosis bacteria.

# 2. BACKGROUND

## 2.1. Biological Background

### 2.1.1. Methods to Acquire Antibiotic Resistance

Most of currently used antibiotic compounds are produced by some microbes as an ability to protect their environment from other organisms [12]. Actually there is an ongoing battle in bacteria kingdom. Some bacteria would develop resistance to their enemies' evolutionary advantages. It is observed that different bacteria start to develop resistance to antibiotics independent from each other. Therefore, it is an undeniable fact that evolutionary mechanisms play an important role in antibiotic resistance development.

The process of antibiotic resistance development is intensified by human-caused influences. The decrease in the cost to produce antibiotics caused the increase in the use of antibiotics. The gradual increase in the usage of antibiotics in anthropogenic activities such as agriculture, aquaculture and non-human applications such as waste disposal, results in higher selection and maintenance pressures on bacteria populations. Misuse and overuse of antibiotics have increased the development of antibiotic resistance among bacteria by rising selective pressures on pathogenic organisms. In addition to accelerated evolutionary process, there is a question required to be answered: what are the biological origins of antibiotic resistance development besides evolutionary pressures on bacteria?

Changes occurring on bacterial DNA may result in the development of resistance to a specific antibiotic. There are different ways for bacteria's DNA to be changed. Random mutations on genetic materials of a bacterium are one of the DNA change methods that may result in the development of antibiotic resistance. For instance, point mutations on gyrA, gyrB, parC, and parE genes of bacterial DNA [13] cause antibiotic resistance. It is also possible for a mutation to cause a phenotype which is responsible

for intrinsic resistance (defined as resistance that is a common character among bacteria species [14]). To illustrate, existence of efflux pump, which is a transportation protein used to extrude substrates, has an impact on the development of antibiotic resistance, they may transmit antibiotics from the cell to the external environment [15].

Another possibility for a variation of bacterial DNA is horizontal gene transfer [16] occurring between different organisms from the same generation. It has three different sub-types namely conjugation, transformation and transduction. Conjugation, in which two bacteria change their genetic materials directly, is one of the horizontal gene transfer methods . These two cells build a cellular bridge (conjugation pilus) between their cytoplasms and genetic materials (generally circular DNA pieces called plasmids) are transferred via this bridge. Transformation is another way to acquire genetic materials from external sources. The main source for these external DNA pieces is dead bacteria's DNA. Transduction is the last HGT method in which DNA of a bacterium is transferred by a bacteriophage (viruses targeting at bacteria). In this type of genetic material acquisition, bacteriophages become natural transmitters for genes of their hosts and transfer some segments of their hosts' genetic materials [17].

There is a phenomenon called host-controlled restriction and modification [17] which is referring to genetic set-up in bacteria identifying and destructing invader DNA. This mechanism works as a natural limit for horizontal gene transfer, but between 1.6 and 32.6 percent of microbial genomes are acquired by horizontal gene transfer according to some estimations [18]. Therefore, HGT plays an important role in spread of antibiotic resistance among bacteria populations in spite of the host-controlled restriction and modification.

### 2.1.2. Mobile Genetic Materials in Bacteria

Plasmids are required to be investigated to understand the origin of antibiotic resistance because they are genetic materials transferred during conjugation. They are relatively small, circular, double-stranded DNA molecules. They are separated from

bacterial chromosomes and replicated independently [12]. To carry antibiotic resistance, a plasmid is required to contain genes or mutations which cause the resistance.

Transposons are gene systems that change their sites within DNA or jump from one DNA to another one [12]. It is also possible for a transposon to jump from plasmids to bacterial chromosomes and vice versa. This is why they are sometimes referred as 'jumping genes' [19]. Resistance causing transposons contain at least one resistant gene, otherwise they would not have any impact on antibiotic resistance.

Gene cassettes are circular, free, non-replicating genetic molecules. Gene cassettes generally contain only one gene. Integrons are genetic elements having specific sites which gene cassettes can be integrated at by site-specific recombination [20]. It is possible to divide integrons into two categories namely mobile and chromosomal. Chromosomal integrons are relatively bigger than mobile integrons, but the mobile ones are the ones which are generally responsible for the dissemination of antibiotics resistance [21].

## 2.2. Machine Learning Background

### 2.2.1. Logistic Regression

Logistic regression is one of the widely used machine learning algorithms. In addition to its computational simplicity, it is easy to understand and implement. It is possible to define logistic regression as the most primitive neural networks, it is simply a single neuron (see Figure 2.1 for details).

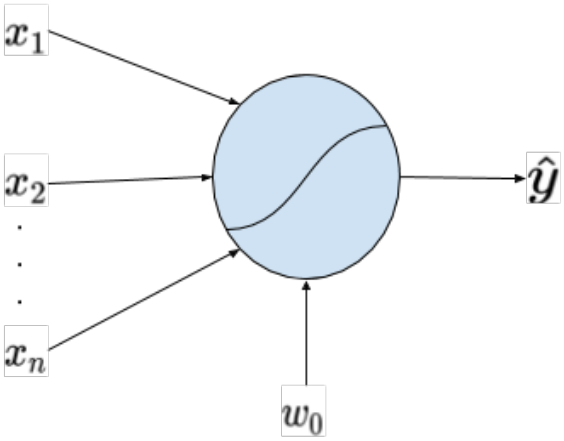


Figure 2.1: Neuron used in logistic regression

The Optimization (finding the best possible set of weights) is done in two steps namely forward propagation and backward propagation. In forward propagation, the class that the sample belongs to and the cost would be estimated. Firstly, let us investigate the formula used in forward propagation. Weighted combination of features is calculated via

$$\hat{y} = w^t * x + w_0 \tag{2.1}$$

'$w$' is a vector containing weights for each feature. It is also possible to state that this vector represents how much a feature is important in class prediction. $w_0$ is called bias. This value is independent from the features. After $\hat{y}$ is estimated, it is transformed by a function known as sigmoid, or logistic function shown in Figure 2.2.

$$p(C_1|X) = \frac{1}{1 + e^{-}\hat{y}} \tag{2.2}$$

Class prediction is done according to the value calculated by the sigmoid transformation. If it is greater than 0.5, the sample will be assigned as class 1 ($C_1$), otherwise it would assigned to class 2.

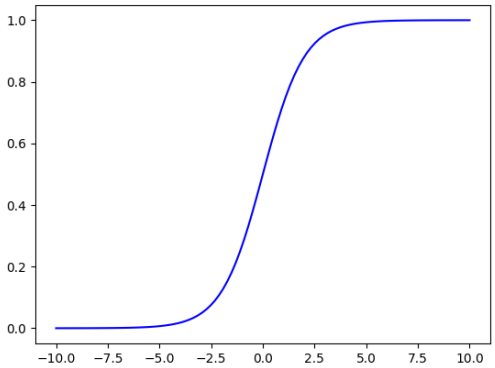


Figure 2.2: Sigmoid (logistic) function

To find the optimal weights for the features, it is required to define a cost function which would be used in backward propagation. Cross-entropy is used as the cost function in logistic regression. The cost function (see Equation 2.4 for details) is estimated by average of the loss function (see Equation 2.3 for details) over the whole training set.

$$L(\hat{y}, y) = y * \log \hat{y} + (1 - y) \log (1 - \hat{y}) \tag{2.3}$$

$$J = -\frac{1}{m} \sum_{i=1}^{m} L(\hat{y}, y) \tag{2.4}$$

The aim of the backward propagation is to find the global minimum of the cost function. This is why each weight is slightly updated according to the derivative of the cost function according to the weight itself in every iteration of backward propagation. Bias should also be included in backward propagation. Moreover, its optimal value is supposed to be found in order to build a successful classifier.

$$\Delta w = \frac{\partial J}{\partial w} \tag{2.5}$$

$$w = w + \Delta w \tag{2.6}$$

Logistic regression can be used in multiclass classification problems as well. To execute logistic regression for multiclass prediction, 'softmax' (see Equation 2.7) is used as the activation function instead of the logistic function. 'K' in the softmax formula corresponds to the class count in the problem. The sample is assigned to the class with the maximum value of the resulting softmax values.

$$\sigma_{z_i} = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for i=1,...,K} \tag{2.7}$$

### 2.2.2. Support Vector Machines

Support vector machines (SVMs) are discriminant based classification algorithms proposed by Vapnik [22]. In other words, the aim of SVMs is to find the best separator hyperplane between classes. To find the optimal discriminator, the margin (distance of hyperplane to the closest points to itself) is tried to be maximized for better generalization [23]. Additionally it is possible to observe that SVM can be used in both binary and multiclass classification problems.

For the binary classification case, every observation in the dataset is a tuple containing features and the class that it belongs to. $X = \{x^t, r^t\}$ where $r^t \in \{-1, 1\}$. Equations for the discriminator and the distance of a point to the discriminator can be seen in equations 2.8 and 2.9 respectively.

$$r^t(w^T * x^t + w_0) = 0 \tag{2.8}$$

$$dist(x, discriminator) = \frac{r^t(w^T * x^t + w_0)}{\|w\|} \tag{2.9}$$

Support vectors are originated from points located on margin boundaries. They are circled in the Fig 2.3. The aim of the algorithm is to maximize the distance between points deriving the support vectors and the hyperplane as it was stated earlier. In the Equation 2.10, $\rho$ is the value which we are trying to maximize during the training.

$$\frac{r^t(w^T * x^t + w_0)}{\|w\|} \geq \rho \tag{2.10}$$

It is possible to convert the equation in the following form

$$r^t(w^T * x^t + w_0) \geq \rho\|w\| \tag{2.11}$$

To handle the possibility of infinitely many solutions to $\rho\|w\|$ due to scaling $\|w\|$, the expression is fixed to 1. To maximize $\rho$, the minimization of $\|w\|$ is required. Then the maximization problem becomes a minimization problem under a constraint because of the duality [22]. The optimization problem seen below can be solved via lagrangian multipliers.

$$\text{Minimize } \frac{1}{2}\|w\|^2 \text{ subject to } r^t(w^T * x^t + w_0) \geq 1 \text{ for all t} \tag{2.12}$$

The lagrangian primal function (see Equation 2.13) is tried to be minimized with respect to $w$ and $w_0$.

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{t=1}^{N} \lambda[r^t(w^T * x^t + w_0) - 1] \tag{2.13}$$

$$= \frac{1}{2}\|w\|^2 - \sum_{t=1}^{N} \lambda r^t(w^T * x^t + w_0) + \sum_{t=1}^{N} \lambda \tag{2.14}$$

After taking derivatives of $L_p$ with respect to $w$ and $w_0$ and plugging them into the equation, we get lagrangian dual objective function [24].

$$L_d = -\frac{1}{2}\sum_t \sum_s \lambda^t \lambda^s r^t r^s (x^t)^T x^s + \sum_t \lambda \tag{2.15}$$

$L_d$ in Equation 2.15 is required to be maximized with respect to $\lambda^t$ [25]. After solving the equation, N of $\lambda$ would be found but most of them would be equal to 0. $x^t$ whose $\lambda$s are greater than 0 is called support vectors. To decide the discriminant, Equation 2.16 is required to be solved for all support vectors and their average is chosen as the discriminant [25].

$$w_0 = r^t - w^T x^t \tag{2.16}$$

It is not possible to find a proper discriminator via the formula mentioned above (they are valid for maximal margin classifier), if classes are not perfectly separable by a linear discriminant. This issue is solved via soft margin in support vector classifiers. Soft margin allows some observations which may be locate inside of the margin or wrong side of the discriminator. Soft margin is applied via introducing a new variable into equations used in maximal margin classifier.

$$r^t(w^T * x^t + w_0) \geq (1 - \epsilon^t) \text{ with } \epsilon^t \geq 0 \text{ and } \sum_{t=1}^{N} \epsilon^t \leq C \tag{2.17}$$

C in the Equation 2.17 is a hyperparameter which is required to be tuned. After inserting relaxing variable into lagrangian primal function, the equation becomes:

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{t=1}^{N} \epsilon^t - \sum_{t=1}^{N} \lambda[r^t(w^T * x^t + w_0) - 1 + \epsilon^t] - \sum_{t=1}^{N} \mu\epsilon^t \tag{2.18}$$

It should take derivatives of $L_p$ with respect to parameters and set them to 0. The values that make derivatives of $L_p$ equal to 0 would be plugged into the equation of $L_p$. The difference between maximal margin classifier is the term related to relaxing variable ($\epsilon$ in Equation 2.17). By doing so we get Lagrangian dual objective function:

$$L_d = -\frac{1}{2}\sum_t \sum_s \lambda^t \lambda^s r^t r^s (x^t)^T x^s + \sum_t \lambda \tag{2.19}$$

where $\sum_t \lambda^t r^t = 0$ and $0 \leq \lambda^t \leq C\,, \forall t$. Support vectors and the discriminator are found like in maximal margin classifiers.

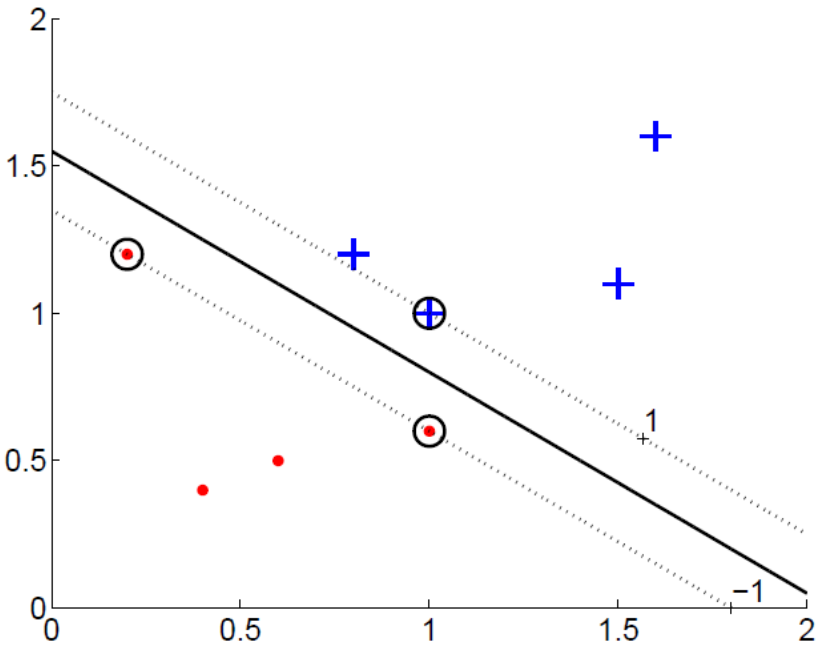


Figure 2.3: The optimal hyperplane in svm [7]

Since not all problems can be solved via linear discriminators, there is a need for more flexible models than linear ones. To get more flexible models, input space is enlarged via transformation functions. This idea is the basis for support vector machines. Actually it is not required to set a transformation function, knowing the kernel function (computing the inner product of vectors in the enlarged feature space [26]) would be enough to continue to find discriminator. The procedure is the same with support vector classifiers after choosing the kernel. In another words, a linear

discriminant is tried to fit into observations enlarged by a transformation function. Most popular kernel functions in SVM literature are pth-degree polynomial and radial basis functions.

### 2.2.3. Random Forests

In machine learning, ensemble methods refer to techniques combining results of different machine learning algorithms. Bagging, in another words bootstrap aggregation, is an ensemble technique to reduce the variance in models having high variances [24]. Basically bagging aims to prevent overfitting by reducing variance. Results of many estimators are averaged in bagging. These estimators are suggested to be unbiased models, relatively weak models [24]. Any simple models such as decision tree with prediction error lower than 1/2 are referred as weak models in this context. These estimators should be better than random guessing.

In the classical bagging, M different base learners are trained on different subsets of the data. These subsets are chosen uniformly and the replacement is allowed. Replacement means that a sample can co-occur in multiple subsets which are not mutually exclusive. After M base learners training, the final decision is made by averaging of estimations of base learners as in Equation 2.20. In classification problems, majority voting is used to decide the model output instead of averaging as observed in Equation 2.21.

$$\hat{f}_{bag}(x) = \frac{1}{M}\sum_{m=1}^{M}\hat{f}_m(x) \tag{2.20}$$

$$\hat{f}_{bag}(x) = majority\_voting\{\hat{f}_b\}_{b=1}^{B} \tag{2.21}$$

Random forests is an improvement for classical bagging which tries to decorrelate base learners by choosing features that would be used to train trees, randomly [27]. In another words, each tree in random forests is grown on the different subset of features.

Therefore, the correlation between trees is decreased. The rest of random forests is the same with vanilla bagging (see Figure 2.4 for details).

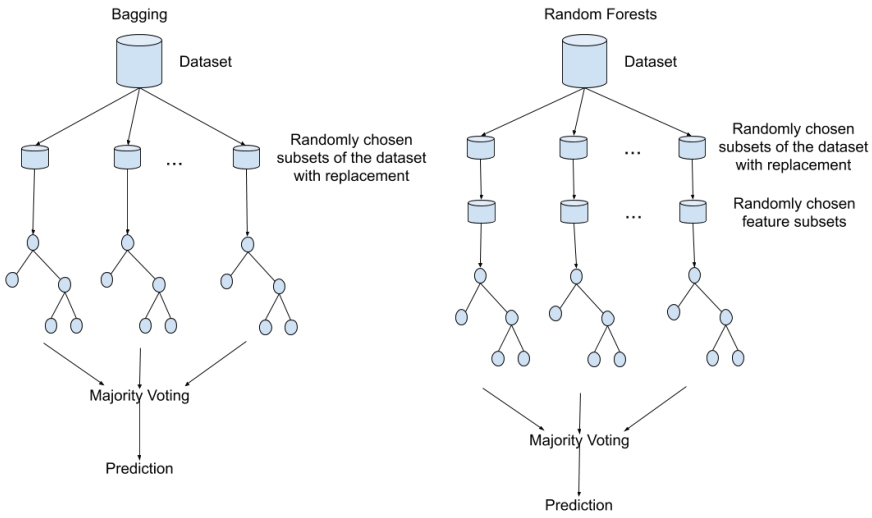


Figure 2.4: Bagging vs random forests

### 2.2.4. XGBoost

There is a concept in machine learning algorithms called boosting. In boosting, many weak estimators trained successively are gathered to create a single strong estimator. Boosting starts with training a simple model on the dataset. Then a new simple estimator, which focuses on deficiencies of previous estimators, is introduced into the system. New models are introduced into the system until there is no improvement in results or estimator count hits the upper limit for estimators (this is a hyperparameter). The logic in the gradient boosting is the same with ordinary boosting. While new estimators are added into the system, the gradient descent algorithm is used to minimize arbitrary loss function which is used to properly combine all estimators.

### 2.2.5. Feedforward Neural Networks

Feedforward neural networks or multilayer perceptrons are essential structures in deep learning approaches. In this type of architectures, it is aimed to approxi-

mate a function with respect to our dataset. In feedforward neural networks, feedback connections to neurons is not observed. When feedback connections are allowed, the architecture is defined as recurrent neural networks [28].

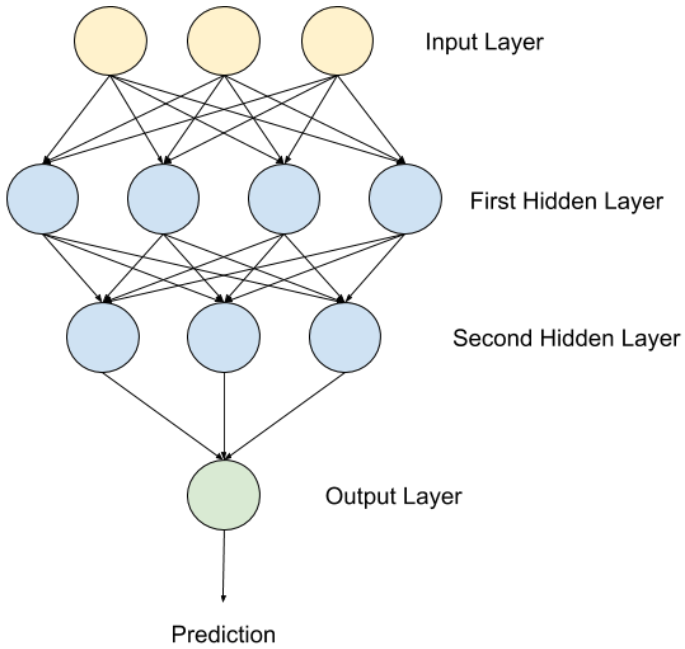


Figure 2.5: A feedforward neural network example

Feedforward neural networks aim to find optimum weights and biases for all hidden units. To conduct an optimization over architectures, it is required to define a loss function for the model. For instance if the problem is binary classification, it is suggested to use binary cross entropy loss. The first step in the training is to make prediction about the output and to calculate the loss with the current state of the model. This process is referred as forward propagation. After completing the forward propagation, an optimizer is used to find optimum weights and biases for every neuron. The backward propagation is an optimization problem in which the loss function is minimized. It starts with differentiation of the loss function with respect to neurons in the output layer so gradients for output neurons are calculated. Then each gradient flows through all related neurons in all layers by the chain rule in the backward direction.

Multilayer perceptrons, as the name suggests, are comprised of more than one hidden layers that consist of perceptrons referred as neurons hidden units (see Figure 2.1 for what neuron means in the context of multilayer perceptrons). Each neuron can be thought as a function which converts input vector to a scalar value. It calculates the result of an activation function fed by linear combination of neuron features. Additionally, a layer is an analogy for a vector-to-vector converter. Features for the first hidden layer is the dataset itself, on the contrary, features for the second and further layers are outputs of previous layers. It is required to break linearity in deep learning architectures because the function that the model tries to learn is generally a non-linear function. This is why non-linear activation functions are used in hidden units in feedforward neueal networks.

There are many different activation functions used to break linearity in deep neural network architectures. Mostly used non-linear activation functions in the literature are tanh (see Figure 2.6 for details), relu (see Figure 2.7 for details), leaky relu (see Figure 2.8 for details), sigmoid (see Figure 2.2 for details) and softmax. Generally sigmoid and softmax are used in the output layer if the responsibility of the model is to conduct a classification. Optimization for relu and leaky relu is easier than tanh, sigmoid, and softmax because they are simply the combination of two different lines.

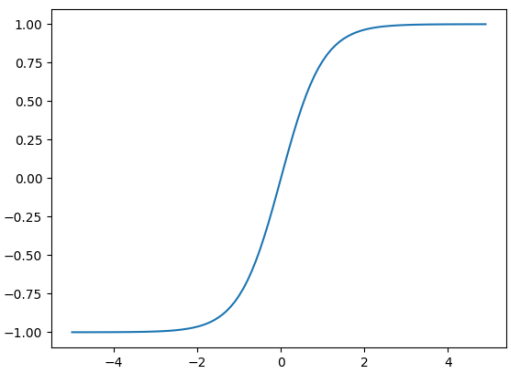


Figure 2.6: Tanh

Deep feedforward neural networks are generally more complex architectures than the ones used in traditional machine learning like svm, random forests etc. When

the complexity in the architecture increases, the model becomes more vulnerable to overfitting. There are different solutions for overfitting. The dropout technique is one of mostly used overfitting precautions in neural networks. Basically some randomly chosen hidden units are deactivated during each training epoch. To deactivate a neuron in the network, its results are multiplied by zero. By doing so its impact on the loss function is ignored for that specific training epoch.

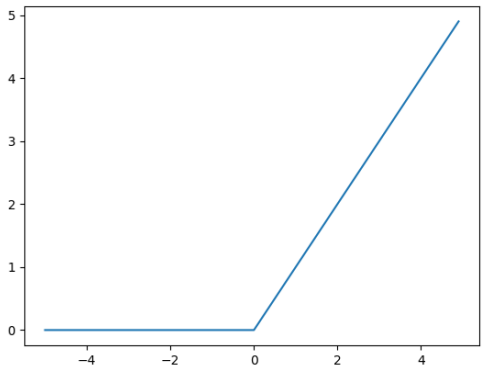


Figure 2.7: ReLU

Additionally there is a method called early stopping to prevent overfitting. If the training continues many epochs, at some point the error on validation set starts to increase while the error on the training set still decreases. In the literature this is called that the model starts to memorize inputs. Early stopping aims to stop the training just before that break point.

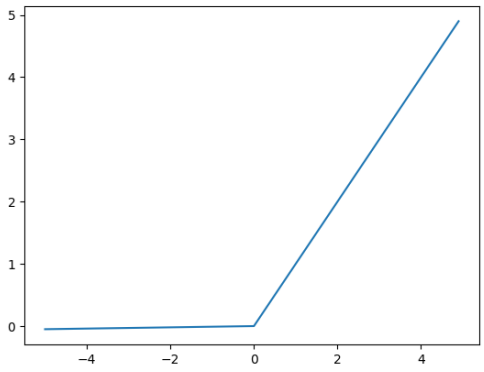


Figure 2.8: Leaky ReLU

### 2.2.6. Estimation of Feature Importance

There are different ways to estimate feature importance for a machine learning model such as permutation feature importance, DeepLIFT. We will focus on the SHAP methodology in our research.

2.2.6.1. Shapley Values and SHAP. SHAP is an abbreviaiton for shapley additive explanations. It is a method to explain how a model predicts with respect to its features. The approach is based on the coalitional game theory [29]. Each feature is thought as a player in the coalition. Shapley values are referring to contribution of each feature to the prediction of the model for a sample.

## 2.3. Literature Review

### 2.3.1. Filtering Process to Find Mutations Causing Antibiotic Resistance [1]

Researchers gathered 3651 complex genome sequence of tuberculosis bacteria from different regions of the world. H37Rv was chosen as reference genome because the bacterium with this DNA is susceptible to all antibiotic families investigated in the research. All paired-end reads were mapped with stampy to the reference genome. Isolates with less than 88% mapped coverage of the reference genome were excluded. Identification of resistance causing single nucleotide mutation over genome-wide association is challenging so they focused on 23 candidate genes and their promoter regions. During the research, characterization of mutations on these target 23 genes as resistance or benign was the aim. We will focus on the same target genes.

Some filters were applied to all loci differing from H37Rv. Firstly sequences differed from H37Rv just on the null-call were eliminated and they would not be included in further investigations. Then synonymous SNPs were filtered out. Another filter was applied to exclude lineage or phylogeny mutations. Remaining mutations were characterized as resistance-determining or as benign. To identify resistant determinant

all genes related to a drug were focused. A mutation was identified as resistant after applying mentioned filters to all genes related to a drug on a sequence known as resistant to a drug. After applying this whole process on all training set, researchers found mutation list that may cause antibiotic resistance.

A mutation library, which contained variants causing antibiotic resistance and not having impact on antibiotic resistance development, was created in this case. The library contains 2512 mutations observed on the mentioned target genes.

### 2.3.2. Deep Learning to Predict Antibiotic Resistance Gene from Metagenomic data [2]

ARGs were gathered from different databases namely ARDB, CARD and UNIPROT. Genes coming from ARDB and CARD were assigned as known genes (genes from these two databases can be thought as features) and these two databases contain information about antibiotic resistance that the gene may be responsible for and antibiotic group that the gene belongs to. There were 102 antibiotics which were grouped under 30 antibiotic categories. Genes coming from UNIPROT would be used in training after preprocessing and they were queried in the database with KW-0046 keyword. Before preprocessing, all duplicate sequences were removed with CD-HIT (discarding all except one that have 100% identity and the same length). While retrieving UNIPROT genes, their metadata information was also gathered to annotate them unless they were already gathered. UNIPROT genes were annotated according to their metadata. Levenshtein distance was used to calculate similarities between gene metadata and antibiotic categories. After text mining, there were two gene groups, one could be categorized with respect to their metadata and the other one could not be successfully annotated (tagged unknown in the paper). After this refinement process, this annotations were validated by ARDB and CARD. If a gene from UNIPROT (after refinement) has more than or equal to 90% identity to ARDB-CARD gene over its whole length, it was tagged as High. Similarly if a gene from UNIPROT has more than or equal to 50% identity and e-value was lower than 1e-10 to ARDB-CARD gene, it was tagged as Mid. DIAMOND (a program similar to BLAST) was used during validating

UNIPROT genes according to ARDB-CARD. The remaining genes would not be used in training of the proposed deep learning models. SNPs (single nucleotide polymorphisms) between UNIPROT genes were also filtered out. After these preprocessing procedures, there were 14933 genes remaining at total.

When preprocessing was completed, a feature matrix whose rows corresponding to homology similarity of UNIPROT genes to ARDB-CARD genes was calculated. UNIPROT genes were aligned to ARDB-CARD genes by using DIAMOND, the bit score was calculated and used as the similarity indicator. Proposed deep learning models were trained with this feature matrix.

There were two proposed architectures: DeepARG-LS which was developed to classify ARGs based on full gene length and DeepARG-SS which was developed to classify ARGs based on short sequence reads. 70% of the dataset was assigned as training set and the remaining part of the dataset was assigned as validation set. The baseline model contained 4 dense hidden layers containing 2000, 1000, 500 and 100 hidden units respectively. Input layer was consisted of 4333 hidden units referring to genes from ARDB and CARD databases. To avoid the overfitting, random hidden units were removed via dropout technique. The Output layer contains 30 units corresponding to 30 antibiotic resistance categories. The softmax activation function was used to calculate the probability of input sequence to each AR categories.

In DeepARG-SS input genes were split into 100 nucleotid-long sequences. An overall precision of 0.97 and a recall of 0.91 were achieved among the 30 antibiotic categories tested. In comparison, the best hit approach achieved an overall 0.96 precision and 0.51 recall. In DeepARG-LS input genes were used as a whole. It was expected that their results would be better because longer read sequences contain more information than short read sequence did. DeepARG-LS achieved a high precision (0.97+-0.03) and an almost perfect recall (0.99+-0.01) for antibiotic categories that were highly represented in the database. Comparatively, the best hit approach achieved a perfect

precision (1.00+-0.00) but a much lower recall (0.48+-0.2) for these categories. Both models did not perform well for categories with less genes. Further validations were conducted on different databases such as MEGARes

### 2.3.3. WDNN to Find Drug Resistance from Whole Genome Sequencing [3]

In the research, WHO network of supranational reference laboratories and the ReSeqTB knowledgebase were used in the training of the initial model included 3,601 MTB isolates. Besides, there were 792 MTB isolates as a separated validation set. 10 different antitubercular drugs would be included in the research. Researches defined 'predictor' to use them in the input layer of proposed WDNN architecture. 28 pre-selected antibiotic resistant genes and promoter regions were targeted by the study. They found 6342 frameship insertion-deletions and SNPs. 156 of these changes existed at least 30 of 3601 MTB isolates so they were selected as predictor. The remaining changes were aggregated into 141 derived categories. 56 of these categories were present in at least 30 isolates so they were chosen as predictor as well. In total, they used 222 predictors to train their model and further analyses (0 or 1 with respect to existence).

Wide and deep neural networks are the combination of logistic regression and deep multilayer perceptrons. In WDNN logistic regression and deep multilayer perceptrons are trained separately and merged in the final classification layer of the model. In deep multilayer perceptrons proposed in the paper contained two hidden layers (both contained 512 hidden units) with the ReLU activation function, monte carlo dropout, and L1 regularization. The network was trained with stochastic gradient descent using Adam optimizer. There were 222 hidden units in the input layer corresponding to 222 predictors. In the concatenation layer there were 734 hidden units. 11 antibiotic drugs were investigated during the research so there were 11 units in the output layer. The sigmoid was used as the activation function in output layer.

During the research different machine learning approaches were trained to compare their proposed model: multitask WDNN. Average sensitivities and specificities for rifampicin and isoniazid were as follows: 97.1% and 95.9% (multitask WDNN), 95.6%

and 95.4% (random forest), 96.7% and 95.7% (regularized logistic regression), 96.3% and 94.3% (preselected mutations MLP), and 97.2% and 95.2% (single task WDNN). Model performance trends were similar for other eight anti-tubercular drugs.

### 2.3.4. ML to Classify Antibiotic Resistance from DNA Sequencing Data [4]

Previously identified 23 candidate genes and their 100 base pair upstream regions were targeted by researchers in this study. Isolates used in the study [1] were used in the research and methods mentioned in the paper [1] were used in DNA sequencing. All SNPs (single nucleotide polymorphisms) were found. The existence of a SNP in an isolate was represented by a binary value: 1 for existence and 0 for absence. 2629 SNPs were found in total. They separated into 3 different feature sets. In the first one all SNPs were used, in the second one SNPs that were previously suspected of being resistance-determinants were included, the last one contained direct determinant genes for drugs. During the research, 11 drugs were investigated.

To understand underlying structure of genetic variations of bacterial isolates, both PCA and SL-PCA (sparse logistic version of PCA) were utilized. Both method decreased the dimensionality of 2639 to 2. Researchers continued with SL-PCA because it was more informative. One method to predict whether an isolate is resistant to a drug or not was called 'direct association'. In this method, an 'OR' rule was employed, if an isolate contained any mutations associated with a given drug in the libraries (Dream TB and the library mentioned in the paper written by T. Walker [1]) it was assigned as resistant to the drug.

During the research 7 different machine learning methods were investigated: logistic regression with L1 and L2 regularization, SVMs with L2 regularization and radial basis function kernel, random forest, a product-of-marginal models, a class-conditional Bernoulli mixture model. In average all machine learning based-methods had higher AUC results than 'direct association' method. In this type of researches sensitivity is

the most important criterion because failure to detect actual antibiotic resistant bacterium may result in harm for a patient. Machine learning algorithms contributed to increase in sensitivity.

### 2.3.5. Application of ML to MTB Drug Resistance Analysis [5]

Traditional machine learning methods such as logistic regression, gradient tree boosting, support vector machines, and random forests were used in this case. Dataset contained 13402 isolates, it was expanded version of the one used in paper named 'Machine learning for classifying tuberculosis drug-resistance from DNA sequencing data' [30]. The dataset was highly imbalanced, isolates were mostly extracted from susceptible bacteria. They tried to figure out importance of mutations in isolates in addition to conducting a classification.

The feature extraction was done as T. Walker explained in his research [8]. Before starting model training, dimensionality was reduced via sparsity constraints such as PCA (principal component analysis) and NMF (non-negative matrix factorization). Imbalance in data was handled by equalizing sample sizes for both classes (resistant and susceptible). To make two class equal sized, susceptible isolates were sub-sampled randomly.

The training was run by the following methodology: 100 iterations 5-fold cross validation and results were reported accordingly. After the classification, 10 most important features (mutations) were also reported for all antibiotics included by this research.

### 2.3.6. Deep Learning to Detect Cooccurent AMR in MTB [6]

In this case, the dataset created by The CRyPTIC Consortium [31] was used to predict co-occurent resistance of mycobacterium tuberculosis. Mutations on target genes defined by T. Walker [8] were used as features. An end-to-end model was proposed by researchers to conduct multilabel classification for antibiotic resistance for

4 first-line antibiotics namely isoniazid, rifampicin, ethambutol and pyrazinamide. A deep denoising autoencoder named DeepAMR was developed. The architecture contained 3 stacked encoders which extracted information. The feature sizes dropped from 5823 to 20. After feature extraction 4 mlp with 2 layers were used to predict whether the sample had resistance for a drug. Training was completed in two steps. The autoencoder was trained initially. After autoencoder training, mlps for each antibiotic were finetuned. Train-test set split was proportional to 70-30.

SNPs ranking was conducted to check variant importance in antibiotic resistance development. The approach based on permutation feature importance. They reported a metric estimating sensitivity drop when a feature was permuted.

# 3. DATASET

## 3.1. Dataset

We focus on the mycobacterium tuberculosis in this research because it is one of the most vital bacteria-causing diseases across the world. Therefore, we included 13860 mycobacterium tuberculosis isolates gathered from different region of the world as stated in baseline papers [8,31]. Dataset are formed by fastq [32] files storing biological sequences and their corresponding quality scores. An example from one of isolates used in our research can be seen below:

@SRR2099940.1 1 length=151

ACCCGGAACCAAGACTCGGAACTAACGAGAACCAGGGAGATACGTCGTTGA

+SRR2099940.1 1 length=151

CCCFFFFFHHHHHIIJJJJJJJJJJIJJJJJIJJJIIJIIIJJIJJHGHFFFFDDDDDDDDDDD

First and third lines in the fastq example, are simple header lines. They contain information about ncbi assigned id for this organism and sometimes information about the machine that reads DNA sequences. The second line contains the actual short DNA reads. As you can see from the example, there are only 4 characters (A, T, C, G) referring to nucleotides that can be observed on DNA. The last line contains read qualities for each nucleotide. Quality score for each nucleotide is represented by a character. All possible quality characters can be seen below. The lowest quality score is represented by exclamation mark and the highest quality score is shown by tilde. All other quality scores are between them and they are located in ascending order. All characters representing quality can be seen below:

!”#$%&’()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]ˆ_‘

abcdefghijklmnopqrstuvwxyz{|}˜

It is not possible to process fastq files directly in our research. Therefore, a data preprocessing pipeline was implemented to extract necessary genomic information about target bacteria. Variants are extracted from raw sequence reads distributed

via fastq files. All models that we investigate during this research, use only variants extracted by the preprocessing pipeline that we developed. Details can be seen in Section 4.1.

4 first line antibiotics used in tuberculosis treatment namely isoniazid, rifampicin, ethambutol and pyrazinamide are main concerns in this research. This is because they are firstly used drugs in the tuberculosis treatment. There are some missing labels for antibiotics in the dataset. Each bacteria isolate is not examined for each target drugs in our baseline researches [31]. Therefore, we need to filter out isolates that do not contain a label for drugs that we examine. We conduct this filtering mechanism for each target drug separately since each antibiotic is investigated independently of the others.

In addition to missing information in the dataset, the dataset itself is highly imbalanced. To elaborate, there are more susceptible isolates in the dataset than resistant ones. This pattern is observed for all antibiotics investigated in the research. Therefore we should handle class imbalance in our dataset to conduct more precise detection of antibiotic resistance in bacteria forming our dataset. For details please check Section 4.2.

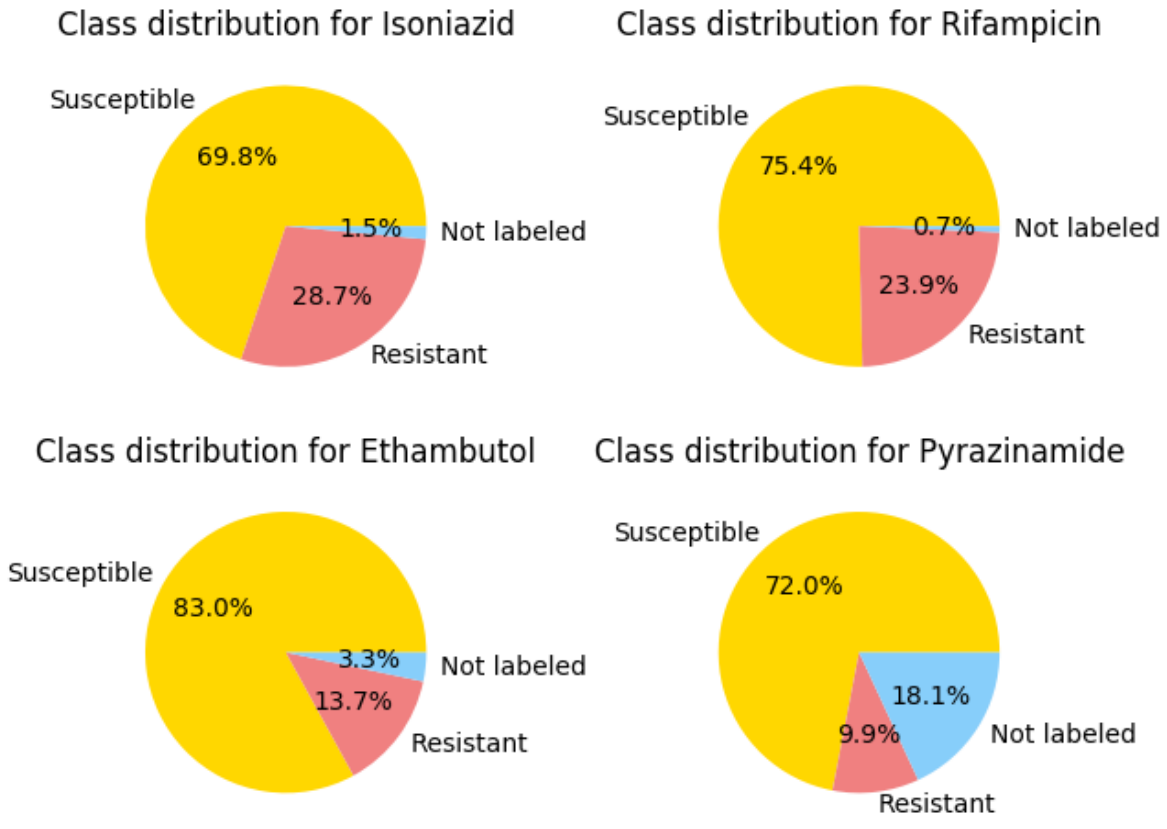


Figure 3.1: Class distributions for all target drugs

# 4. METHODS

We focus on 4 first-line antibiotics and 23 genes on which mutations are previously detected as resistance determinants during this research. Methods can be categorized under 3 topics namely, data preprocessing, classification and feature importance extraction.

## 4.1. Data Preprocessing

### 4.1.1. Preprocessing Pipeline

We developed a pipeline application to conduct data preprocesing. The pipeline is formed by 5 subprocesses namely downloading isolate read files, sequence alignment, bam cleaning, variant calling and variant annotation. First step is to download experiment files containing short reads gathered by next generation sequencing techniques [33] for each mycobacterium tuberculosis isolate investigated in this research.

First of all, a manager application is written to manage all tools included in the pipeline. Python is chosen as the language for the application. The application conducts command line calls for all other third party applications to prepare all required information from fastq files containing short reads derived from next generation DNA sequencing.

There are 3 required arguments which respectively refer to pipeline's own configuration file, the main directory containing all isolate folders and identifications of isolates to which the pipeline would be applied. Provided ids are supposed to satisfy one of the following formats: single value (348), comma-separated two values (348,400) or dash-separated two values (348-400). If single value is provided to the application then it applies the data preprocessing pipeline only to the given isolate. If the argument contains comma-separated two values, it divides the argument into two values by comma and applies the pipeline to these two ids. If the provided arguments contains

dash then the manager application divides the input into two values and applies the pipeline to all isolates between provided two values (including both of the inputs).

The pipeline uses a yaml [34] configuration file to trigger download module. The application manager assumes that each isolate has its own directory and all these isolate directories are contained by another base directory. As you might guess each isolate directory contains configuration file for the download module, it has two different method to download fastq files. The first one uses FTP and gets data from european nucleotide archieve. The other one uses NCBI Sratoolkit. Download module reads run ids from the configuration file if it uses NCBI Sratoolkit. It uses download links, if FTP is chosen. The default download method in the current version of the pipeline is NCBI Sratoolkit due to its download performance.

After downloading fastq files, they are aligned to a reference genome. H37Rv sequence of mycobacterium tuberculosis is the one we used in our research because it is susceptible to all first-line antibiotics. There are different aligners integrated with the pipeline manager application. It is possible to work with one of following aligners: Stampy [35], NextGenMap [36], BWA [37] and GSNAP [38]. GSNAP is the current default sequence alignment mapper, As it is defined in the research in which different aligners and variant callers are compared [39], it achieves high sensitivity without losing much in specificity. Installation of these mappers is prerequisite and should be done manually. After installation process for these mappers is completed, configuration for the aligner can be set in the configuration file which is provided the application with a command line argument. These mappers give output in sam [40] format. However, BamCleaner utilizes bam files as input so sam files created by alignment process are converted into bam [40] by SAMtools [40].

When bam files are ready, they are put into cleaning module of the pipeline manager. To clean bam files, GATK is used [41]. It has a recommended pipeline to clean files created by sequence alignment mappers [42]. In our approach, bam files are sorted, duplicates in bam files are marked and removed and bam files without duplicates are indexed via the preprocessing pipeline manager.

The preprocessing manager conducts variant callings to find possible SNPs and INDELs on cleaned bam files. All variants are stored in a vcf file [43]. Two different variant callers are used in the pipeline. First one is referred as SAMtools [40] and the other one is referred as Platypus [44]. They perform differently in detecting SNPs and INDELs. It is possible to extract features from different combinations of variants detected by these two methodologies. However, we will include SNPs detected by samtools and INDELs detected by platypus in our research.

Last step of the pipeline is normalizing the vcf files created by variant callers. It is used to detect whether any two indels are representing the same modification on the DNA or not. If two indels are the same then we are not required to include both of them in our research. Left-normalization is gaining popularity among researchers. Left normalization means that start position of a variant is shifted to left until it is not possible to do so and the mutation is represented as few nucleotides as possible.

### 4.1.2. Base Feature Matrices

We focus preselected 23 genes defined by T.Walker [8] in this research. There is at least one mutation on these genes that was previously detected to cause antibiotic resistance in mycobacterium tuberculosis [8]. Additionally, promoter regions of these genes are also included in our processes. Antibiotic and related resistance determinant genes can be seen in the table 4.1.

Variants that are not related to target genes are ignored in further steps of our research. After selecting variants only observed on target genes, we apply some filters to SNPs which are independent of used variant callers. Then the feature matrix is formed by remaining variants. First filter is related to read depth of variants. It should be greater than 90%, otherwise the SNP would be neglected in the rest of the research. The second filter is related to the observation frequency of a mutation. Mutations which are seen in only one isolate are filtered out by the second filter. We do not apply any filters to INDELS.

Table 4.1: Antibiotic Resistance and Gene Relations [8]

| **Isoniazid** | **Rifampicin** | **Ethambutol** | **Pyrazinamide** | **Other Antibiotics** |
|---|---|---|---|---|
| ahpC | rpoB | embA | pncA | gyrA |
| fabG1 | | embB | rpsA | gyrB |
| inhA | | embC | | rpsL |
| katG | | embR | | gidB |
| ndh | | iniA | | rrs |
| | | iniC | | tlyA |
| | | manB | | eis |
| | | rmlD | | |

All detected variants are iterated over for each variant detection methods. These variants constitute columns of the feature matrix. Additionally isolates form rows of the feature matrix. If a variant exists in an isolate, related feature matrix location is marked as '1', otherwise it is marked as '0'. We would call this feature matrix as binary data representation in next steps of this research. It is possible to see feature matrix size in the table 4.2:

Table 4.2: Base Feature Matrices

| **Variant Detection** | **Feature Matrix Size** |
|---|---|
| snp: SAMtools & indel: Platypus | 13860*11750 |

### 4.1.3. Additional Data Representations

Binary data representation assumes that each feature has the same weight. This means in case at hand that each mutation has a direct impact on the antibiotic resistance. However, some mutations on target genes may not have any impact on antibiotic resistance development. This is why finding an alternative data representation which differentiate weights of mutations may improve performances of our models. Term

weighting methods used in information retrieval help us to avoid the assumption that each mutation has the same influence on the development of drug resistance. We investigate 4 different term weighting methods generally used in natural language processing namely TF-IDF, TF-RF, BM25TF-IDF, BM25TF-RF.

4.1.3.1. TF-IDF based data representation. TF-IDF (term frequency - inverse document frequency) is used to estimate distance between the query and a document. All documents are sorted according to this evaluation and the owner of the query would face with most relevant documents to his/her query. By doing so, search quality is aimed to be improved.

TF is obtained by the division of the number of that a term is seen in a document to the total term count in the document. We mention the result of this division as $TF$ in the formula. IDF is the measure of how much information a term provides. In another words, IDF is used to increase weights of rare terms and decrease significance of common terms. It is calculated by division of the total document count (N) to the number of these documents that contain the term ($DF$).

$$TF - IDF = TF * IDF \tag{4.1}$$

$$TF = FT \tag{4.2}$$

$$IDF = \log(N/DF) \tag{4.3}$$

In our context, isolates are corresponding to documents and mutations are corresponding to terms. There is a difference between TF-IDF that we used and the one in the literature. A mutation can be seen only once in an isolate despite the fact that a term can be found multiple times in a document. Therefore, $TF$ was changed into $TF = 1/((\text{number of mutation on the isolate})+1)$. We summed divider with 1 to avoid 0 division issue. Additionally base of log in idf was set as 2 while some resources in the literature are using $e$ as the base for the logarithm.

4.1.3.2. TF-RF based data representation. Researchers try to find alternative data representation models to include predefinedly categorized information in the text categorization problem. In unsupervised weighting methods, the category of documents is neglected in a way that a term has the same weight for each document category. On the contrary, term weights in different categories are not required to be the same in supervised term weighting.

While calculating inverse document frequency (an unsupervised weighting method), the category of the document is ignored. In another words, IDF minds only the number of the documents in the corpus that contain the queried terms without checking their categories. Therefore, a term has the same weight for each document category. On the other hand, RF (relevance frequency) includes document categories containing the queried term in the calculation as well. TF-RF is used to estimate how important a term is for a document category [45].

Ratio of the number of a term is observed in documents from positive categories to the number of a term is observed in documents from negative categories is used to estimate RF value [46]. By doing so, weight of a term is based on categories of documents.

$$TF - RF = TF * RF \tag{4.4}$$

$$RF = \log_2(2 + \frac{a}{max(1, c)}) \tag{4.5}$$

'a' and 'c' in the RF formula are corresponding to observation count of the term in all documents with positive category and observation count of the term in all documents with negative category respectively. To understand difference between RF and IDF, it would be better to convert IDF formula into a form containing variables used in RF estimation: $IDF = log(\frac{N}{a+c})$.

As aforementioned, mutations and isolates are corresponding terms and documents respectively. In our context RF is calculated by division of the number of resistant isolates containing the mutation to the number of susceptible isolates containing the mutations. We sum 1 with the 'c' instead of choosing the maximum of 1 and 'c'. Thus, the RF formula becomes Equation 4.6

$$RF = \log_2(2 + \frac{a}{1+c}) \tag{4.6}$$

4.1.3.3. BM25TF-IDF based data representation. The algorithm is often called 'Okapi BM25' and BM is referring to 'best matching'. The algorithm is used by search engines to rank documents according to their relevance score to a query. In our case, we would try to score the relevance between variants and drug resistance. To calculate the relevance score, we apply BM25 mechanics on TF term. TF is calculated like defined in TF-IDF (see Equation 4.2 for details). $L_s$ is referring to the length of samples and $L_{ave}$ is referring to the average length of all samples in the entire collection. All samples in our dataset has same sized features. Therefore $L_s$ and $L_{ave}$ are equals and the equation 4.7 becomes the equation 4.8.

$$BM25TF = ((k+1) * TF)/(k * ((1-b) + b * (L_s/L_{ave})) + TF) \tag{4.7}$$

$$BM25TF = ((k+1) * TF)/(k + TF) \tag{4.8}$$

IDF is calculated as defined in the TF-IDF (see Equation 4.3 for details). The final score is calculated by multiplying BM25TF and IDF values.

$$BM25TF - IDF = BM25TF * IDF \tag{4.9}$$

4.1.3.4. BM25TF-RF based data representation. BM25TF was calculated as it was defined in BM25TF-IDF and RF (see Equation 4.5 for details) is calculated as it is defined in TF-RF. BM25TF-RF is calculated via multiplication of BM25TF and RF.

$$BM25TF - RF = BM25TF * RF \tag{4.10}$$

## 4.2. Detection of Antibiotic Resistance

To detect antibiotic resistance in mycobacterium tuberculosis we utilize different machine learning algorithms such as svm, random forests, xgboost, and feedforward neural networks. All algorithms use feature matrices containing information about only variants on target genes as input.

Tuberculosis resistance to 4 first-line antibiotics: isoniazid, rifampicin, ethambutol and pyrazinamide is investigated in this research. Each antibiotic is evaluated separately. The problem is thought as a binary classification for each antibiotic instead of a multilabels classification problem. There are two reasons for taking problem as a binary classification. The first one is missing information about labels in the dataset. Different samples in the dataset are lacking labels for different antibiotics. To ignore any isolate lacking label for any antibiotics would lead into serious decrease in the dataset size. The bigger dataset would give better results for feedforward neural networks. The second reason is that the feature importance extraction is easier if models are trained for a binary classification problem.

Additionally, the dataset is highly imbalanced which means susceptible and resistant classes are not of equally sized. The susceptible class has much more samples than the resistant class does. Imbalanced class pattern is observed for all antibiotics. To handle the imbalanced dataset issue, we define weights for susceptible and resistant classes (see Equations 4.11, 4.12 respectively for details) and use these values in the training process of models. For instance in feedforward neural network architectures we use class weights during the calculation of loss function so the optimization would

handle the imbalance in class samples. To simplify, when the model makes a wrong classification about resistant isolates, it will be penalized more than wrong classification about susceptible isolates.

$$w_{susceptible} = max(C_{susceptible}, C_{resistant})/C_{susceptible} \tag{4.11}$$

$$w_{resistant} = max(C_{susceptible}, C_{resistant})/C_{resistant} \tag{4.12}$$

Where:

$C_{susceptible}$: How many isolates belong to susceptible class.

$C_{resistant}$: How many isolates belong to resistant class.

Dataset is divided into two subsets namely training and test set. Training set contains 80% of the whole dataset and remaining 20% is assigned as test set. While separating the dataset, proportion of resistant isolates to susceptible ones in both training and test sets are equal to the ratio of resistant to susceptible isolates in whole dataset. To tune hyperparameters for every model, 10 fold cross validation is used on the training set. We use grid search as the hyperparameter optimization (see Appendix A for details about hyperparameters for each algorithm) technique. After hyperparameter optimization, each model is trained on the training set and evaluated on the test set separately.

Traditional machine learning algorithms are used from the python library named scikit-learn [47]. To utilize xgboost, the library, which has the same name with the algorithm, is used in this research [48]. To build multilayer perceptrons, we use pytorch [49]. Artificial neural network with 1, 2, and 3 hidden layers are built and compared to traditional machine learning algorithms in this research.

### 4.3. Extraction of Antibiotic Resistance Causing Mutations

The mutation key, represented in columns of the feature matrix, contains mutation location and nucleotide changes. We convert this information into mutation id defined by Human Genome Variation Society as our reference studies do.

We include 23 target genes (see Table 4.1 for details) on which mutations are detected as having an impact on antibiotic resistance in our research. Therefore, we can only face with mutations observed on genes and their promoter regions. Naming conventions for variants, that we follow, is as follows:

- If the mutation is SNP and it occurs on the coding region, it is named by concatenation of gene name, "_" symbol, aminoacid that is encoded unless the mutation occurs and aminoacid that is encoded after mutation
- If the mutation is SNP and it occurs on the promoter region, it is named by concatenation of gene name, "_" symbol, nucleotide in reference genome, location with respect to start position of start codon on encoding region of the gene, nucleotide after mutations. The location is negative value, it is the distance to the beginning of start codon.
- If the mutation is deletion, it is named by concatenation of gene name, "_" symbol and deletion start location on gene, "_" symbol, deletion end location on gene and deleted sequence.
- If the mutation is insertion, is named by concatenation of gene name, "_" symbol and insertion start location on gene, "_" symbol, insertion end location on gene and inserted sequence.

We use xgboost to detect possible reasons for the development of antibiotic resistance. SHAP (see Section 2.2.6.1 for details) is used to rank features according to their impacts on drug resistance development. Basically SHAP estimates shapley values referring to how much a feature contributes to the model output. 10 variants with the highest SHAP scores (see Table 5.1 for each antibiotic) are reported as possible resistance determinants.

# 5. EXPERIMENTS AND RESULTS

## 5.1. Classifier Comparison

Firstly we conduct experiments to decide which machine learning algorithm serves better for our first goal to detect antibiotic resistance in mycobacterium tuberculosis. To choose the best model, we train the all alternative models namely support vector machine with linear and rbf kernel, logistic regression, random forests, xgboost, feed-forward neural network with 1, 2 and 3 hidden layers on the same presplit train set. The training and the test set contain 80% and 20% of the whole data set respectively. Different metrics such as sensitivity, specificity, precision and f1 score are reported to compare models.

3 dummy classifiers are proposed as baselines. In the baseline 0, all samples are assigned as susceptible, in the baseline 1, all samples are chosen as resistant to investigated antibiotic. In the last one (baseline 2) each sample is randomly assigned as susceptible or resistant. A value is created between 0 and 1 uniformly for each sample. If the estimated value is smaller than or equal to ratio of the resistant samples to whole labeled samples in the dataset for investigated antibiotic, the sample is assigned as resistant, otherwise it is assigned as benign. These dummy classifiers are used to check whether investigated models are actually learning information from the data set or not. It is possible to see that all machine learning models surpass baseline classifiers, from Tables 5.1, 5.2, 5.3, 5.4. Therefore we deduce that all machine learning algorithms are actually learning information from variants in the data set.

Different models perform differently for different antibiotics. Traditional algorithms generally perform better classification than neural network approaches. For isoniazid (see Table 5.1 for details), random forests surpass all other models when sensitivity is at stake but it is not as powerful in specificity as it is for sensitivity. When we examine f1 score, we observe that the most successful models are followings: svm with linear and rbf kernel, logistic regression and xgboost. For our research xgboost is

the best option among these 4 models for isoniazid because even if their f1 scores are equal, xgboost has higher sensitivity than the other three has.

Table 5.1: Model Comparison for Isoniazid

| **Model** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
|---|---|---|---|---|
| baseline0 | 0.00 | 1.00 | 1.00 | 0.00 |
| baseline1 | 1.00 | 0.00 | 0.29 | 0.45 |
| baseline2 | 0.28 | 0.71 | 0.28 | 0.28 |
| svm_linear | 0.91 | 0.98 | 0.96 | 0.94 |
| svm_rbf | 0.92 | 0.98 | 0.96 | 0.94 |
| lr | 0.91 | 0.99 | 0.82 | 0.94 |
| rf | 0.96 | 0.92 | 0.82 | 0.88 |
| xgboost | 0.93 | 0.98 | 0.96 | 0.94 |
| ffnn-1d | 0.90 | 0.95 | 0.88 | 0.89 |
| ffnn-2d | 0.86 | 0.96 | 0.89 | 0.88 |
| ffnn-3d | 0.85 | 0.97 | 0.93 | 0.89 |

Rifampicin is another antibiotic included in this research. It is observed that traditional machine learning algorithms have better performance in average than feed-forward neutal networks do. Additionally random forests has the worst scores among traditional algorithms even in sensitivity. Interestingly multilayer perceptrons with 1 hidden layer surpasses all remaining algorithms in sensitivity. On the other hand it does not perform well in specificity, precision, and f1 score. It falls behind traditional machine learning approaches. In average, svm with linear and rbf kernel, logistic regression, and xgboost perform similarly (see Table 5.2 for details). Svm with linear kernel and logistic regression has slightly better f1 score than xgboost yet all of these 3 algorithms perform similarly with respect to sensitivity. Xgboost is chosen for further steps of this research because the difference in performance of these models is ignorable. Additionally decision based algorithms are easier to interpret and we use interpretaion methods to detect mutations that lead to antibiotic resistance.

Table 5.2: Model Comparison for Rifampicin

| **Model** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
|---|---|---|---|---|
| baseline0 | 0.00 | 1.00 | 1.00 | 0.00 |
| baseline1 | 1.00 | 0.00 | 0.24 | 0.39 |
| baseline2 | 0.23 | 0.76 | 0.23 | 0.23 |
| svm_linear | 0.91 | 0.99 | 0.96 | 0.94 |
| svm_rbf | 0.91 | 0.99 | 0.96 | 0.93 |
| lr | 0.91 | 0.99 | 0.96 | 0.94 |
| rf | 0.88 | 0.99 | 0.96 | 0.92 |
| xgboost | 0.91 | 0.99 | 0.95 | 0.93 |
| ffnn-1d | 0.92 | 0.91 | 0.77 | 0.84 |
| ffnn-2d | 0.86 | 0.98 | 0.93 | 0.90 |
| ffnn-3d | 0.88 | 0.98 | 0.94 | 0.91 |

We observe similar results for ethambutol with that we do for isoniazid and rifampicin. Multilayet perceptrons perform worse than traditional machine learning algorithms in the detection of antibiotic resistance. It is possible to see drastic decline in random forests performance for ethambutol in the table 5.3. It falls behind other traditional machine learning approaches. Xgboost surpasses all algorithms when it comes to f1 score. However, svm with linear and rbf kernels and logistic regression have higher sensitivity scores. We still choose xgboost to detect variants that cause resistance to ethambutol due to the fact that it is easier to explain how xgboost makes decision and similar performance scores.

Table 5.3: Model Comparison for Ethambutol

| **Model** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
|---|---|---|---|---|
| baseline0 | 0.00 | 1.00 | 1.00 | 0.00 |
| baseline1 | 1.00 | 0.00 | 0.14 | 0.25 |
| baseline2 | 0.14 | 0.87 | 0.15 | 0.15 |
| svm_linear | 0.91 | 0.91 | 0.62 | 0.74 |
| svm_rbf | 0.92 | 0.91 | 0.64 | 0.75 |
| lr | 0.91 | 0.91 | 0.63 | 0.75 |
| rf | 0.68 | 0.91 | 0.56 | 0.61 |
| xgboost | 0.89 | 0.92 | 0.66 | 0.76 |
| ffnn-1d | 0.76 | 0.92 | 0.62 | 0.68 |
| ffnn-2d | 0.66 | 0.95 | 0.70 | 0.68 |
| ffnn-3d | 0.62 | 0.96 | 0.74 | 0.68 |

We observe serious drop in performance for all models when pyrazinamide is at stake. It has the biggest subset of data that are not labeled so models are not able to properly learn information from the dataset. This results in a drastic performance drop. 18.1% of data are unlabeled yet positive class forms only 9.9% of the whole samples. It is the lowest value among all 4 antibiotics. Xgboost has a relatively higher performance than all other algorithms do. We observe the pattern that feedforward neural network approaches fall behind traditional machine learning algorithms. However, their specificity scores are comparable to values for traditional models. Xgboost is the approach that we choose to use in next steps of this research because it is one of investigated approaches with highest f1 and sensitivity scores.

Table 5.4: Model Comparison for Pyrazinamide

| **Model** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
|---|---|---|---|---|
| baseline0 | 0.00 | 1.00 | 1.00 | 0.00 |
| baseline1 | 1.00 | 0.00 | 0.12 | 0.22 |
| baseline2 | 0.11 | 0.88 | 0.11 | 0.11 |
| svm_linear | 0.74 | 0.94 | 0.64 | 0.69 |
| svm_rbf | 0.61 | 0.97 | 0.72 | 0.66 |
| lr | 0.75 | 0.95 | 0.66 | 0.70 |
| rf | 0.55 | 0.98 | 0.79 | 0.65 |
| xgboost | 0.70 | 0.95 | 0.68 | 0.69 |
| ffnn-1d | 0.63 | 0.97 | 0.77 | 0.69 |
| ffnn-2d | 0.61 | 0.96 | 0.71 | 0.66 |
| ffnn-3d | 0.66 | 0.95 | 0.65 | 0.66 |

Another observation is that predictions about susceptible class are generally more accurate. This situation leads us to the opinion that the issue about class imbalance could not be completely solved. Even if the impact of class imbalance is reduced, algorithms still tend to make more predictions of susceptible class.

When feedforward neural networks are compared to each other, we observe that multilayer perceptrons with 1 hidden layer has better sensitivity for isoniazid, rifampicin and ethambutol. However, multilayer perceptrons with 3 hidden layers has better sensitivity score for pyrazinamide. Additionally, when the model complexity is increased, the model predicts with higher specificity scores. Therefore we deduce complex FFNNs learn information about negative class better than they do for resistant class.

To sum up, we try different machine learning approaches including some neural network architectures and traditional ones to detect antibiotic resistance to 4 first-line drugs being used in treatment of mycobacterium tuberculosis. Machine learning algorithms investigated in this research, are actually learning information because all

of them surpass the proposed baseline estimators. Additionally traditional machine learning algorithms give better results for all target drugs than multilayer perceptrons do.

## 5.2. Data Representation Comparison

All algorithms, that we investigated in this research, are trained and evaluated with different data representations such as TF-IDF, TF-RF, BM25TF-IDF, BM25TF-RF. However, we only share results for xgboost below to not spoil the simplicity (see Appendix B for all data representation comparisons).

Although we expect some improvements in the classification performance with implementation of different data representations used in natural language processing, we do not observe any significant improvements in the model performance (see Table 5.5 for details). Therefore, we decide to continue with the binary feature matrix to find variants potentially causing antibiotic resistance.

Table 5.5: Data Representation Comparison for XGBoost

| **Data Representations** | **Sensitivity** | **Specificity** | **Precision** | **F1 Score** |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.93 | 0.98 | 0.96 | 0.94 |
| TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| **Rifampicin** | | | | |
| binary | 0.91 | 0.99 | 0.95 | 0.93 |
| TF-IDF | 0.90 | 0.98 | 0.95 | 0.92 |
| TF-RF | 0.90 | 0.98 | 0.95 | 0.92 |
| BM25TF-IDF | 0.90 | 0.98 | 0.95 | 0.92 |
| BM25TF-RF | 0.90 | 0.98 | 0.95 | 0.92 |

*Continued on next page*

Table 5.5 – *Continued from previous page*

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Ethambutol** | | | | |
| binary | 0.89 | 0.92 | 0.66 | 0.76 |
| TF-IDF | 0.90 | 0.92 | 0.66 | 0.76 |
| TF-RF | 0.90 | 0.92 | 0.66 | 0.76 |
| BM25TF-IDF | 0.90 | 0.92 | 0.66 | 0.76 |
| BM25TF-RF | 0.90 | 0.92 | 0.66 | 0.76 |
| **Pyrazinamide** | | | | |
| binary | 0.70 | 0.95 | 0.68 | 0.69 |
| TF-IDF | 0.66 | 0.95 | 0.64 | 0.65 |
| TF-RF | 0.66 | 0.95 | 0.64 | 0.65 |
| BM25TF-IDF | 0.66 | 0.95 | 0.64 | 0.65 |
| BM25TF-RF | 0.66 | 0.95 | 0.64 | 0.65 |

## 5.3. Important Mutations for Antibiotic Resistance

Different machine learning approaches are investigated during this research but xgboost is chosen to find resistance determinant mutations because its performance is above the average of all investigated algorithms (see Section 5.1 for details) and its structure is relatively simpler to explain. Due to its decision tree based structure, it is easier to interpret. We use SHAP (SHaply Additive exPlaniation) values (see Section 2.2.6.1 for information about SHAP) to understand which mutations are important in the development of antibiotic resistance for each drug. It is possible to see the most important 10 mutations in the development of antibiotic resistance for each target drug in Table 5.6. SHAP values are estimations representing how much each feature contributes to the final classification .

We rank all features according to SHAP values estimated for xgboost. Then variants with the highest SHAP values are chosen for future investigations. 10 mutations chosen for each antibiotic by xgboost (see Figure 5.1 for estimated SHAP values) can be seen in Table 5.6. Mutations annotated with * (like katG1_S315T*) actually lead to antibiotic resistance for the investigated drug according to dreamtb database.

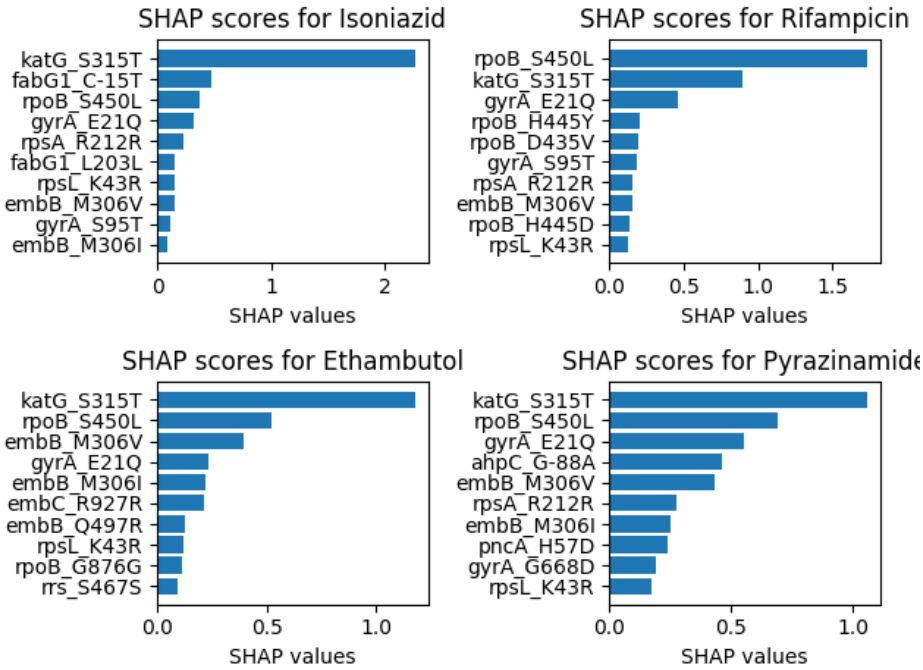


Figure 5.1: SHAP values estimated for 10 most important mutations

We took dreamtb database [50] as the ground truth for the relation between antibiotic resistance and resistance determinants. The database was curated from researches about antibiotic resistance in mycobacterium tuberculosis in the literature. It contains 997 mutations that lead to antibiotic resistance to 4 first-line drugs. We use the database to validate variants proposed by our model.

Table 5.6: Antibiotics Resistance Causing Mutations

| **Isoniazid** | **Rifampicin** | **Ethambutol** | **Pyrazinamide** |
|---|---|---|---|
| katG_S315T* | rpoB_S450L* | katG_S315T | katG_S315T |
| fabG1_C−15T* | katG_S315T | rpoB_S450L | rpoB_S450L |
| rpoB_S450L | gyrA_E21Q | embB_M306V* | gyrA_E21Q |
| gyrA_E21Q | rpoB_H445Y* | gyrA_E21Q | ahpC_G−88A |
| rpsA_R212R | rpoB_D435V* | embB_M306I* | embB_M306V |
| fabG1_L203L | gyrA_S95T | embC_R927R | rpsA_R212R |
| rpsL_K43R | rpsA_R212R | embB_Q497R* | embB_M306I |
| embB_M306V | embB_M306V | rpsL_K43R | pncA_H57D* |
| gyrA_S95T | rpoB_H445D* | rpoB_G876G | gyrA_G668D |
| embB_M306I | rpsL_K43R | rrs_S467S | rpsL_K43R |

Additionally we try to understand the success of our model. Therefore, we estimate precision values for different top n features such as 10, 20, 40 and 60. There are two different precision values. The first one estimates the ratio of resistance causing variants for the investigated antibiotics to the count of chosen mutations. On the other hand the second one estimates the ratio of resistance causing mutations for any drugs investigated in this research to the count of chosen mutations (see Table 5.7 for details). We also calculate precision values for the model proposed in the reference study [5]. The reference model is an implementation of logistic regression. The model in the reference study has better precision scores than our xgboost does. Details can be seen in the table 5.8.

Table 5.7: Precision for Top n Features

| **Top n Features** | **Investigated Drug** | **All Drugs** | **Drug** |
|---|---|---|---|
| 10 | 0.20 | 0.50 | Isoniazid |
| 20 | 0.20 | 0.55 | |
| 40 | 0.20 | 0.45 | |
| 60 | 0.17 | 0.37 | |

*Continued on next page*

Table 5.7 – *Continued from previous page*

| Top n Features | Investigated Drug | All Drugs | Drug |
|---|---|---|---|
| 10 | 0.40 | 0.60 | Rifampicin |
| 20 | 0.30 | 0.50 | |
| 40 | 0.18 | 0.35 | |
| 60 | 0.15 | 0.32 | |
| 10 | 0.20 | 0.40 | Ethambutol |
| 20 | 0.25 | 0.45 | |
| 40 | 0.28 | 0.43 | |
| 60 | 0.25 | 0.38 | |
| 10 | 0.10 | 0.50 | Pyrazinamide |
| 20 | 0.05 | 0.40 | |
| 40 | 0.03 | 0.30 | |
| 60 | 0.01 | 0.30 | |

We use dreamtb as the source for mutations leading into development of antibiotic resistance. Precision estimations are done according to dreamtb and the database was lastly updated at May 26, 2014. Therefore, we can not directly deduce that variants proposed by our model are not related to antibiotic resistance to target drugs so we extract a list containing difference between variants chosen by our model and the reference study. Then we filter out variants which exist in dreamtb database. We present these mutations as our hypothesis (see Table 5.12 for the whole list). In the next step of this research, we try to find proof that these mutations actually play a role in the development of antibiotic resistance.

Table 5.8: Precision Comparison for 10 Chosen Variants According to Dreamtb

| | **Precision For** | | |
|---|---|---|---|
| **Variant Detector** | **Investigated Drug** | **All Drugs** | **Drug** |
| XGBoost | 0.20 | 0.50 | Isoniazid |
| Reference | 0.50 | 0.50 | |
| XGBoost | 0.40 | 0.60 | Rifampicin |
| Reference | 0.75 | 0.75 | |
| XGBoost | 0.20 | 0.40 | Ethambutol |
| Reference | 1.00 | 1.00 | |
| XGBoost | 0.10 | 0.50 | Pyrazinamide |
| Reference | 0.50 | 0.50 | |

There is another database referred as mycoresistance containing mutations and related researches in the literature [51]. It contains more recent papers than dreamtb. Mutations which are proposed by our model are validated by papers included by this database. It is possible to see all reference papers validating our hypothesis in the table 5.12. Precision for variants proposed by our model and the model in the reference study [5] are estimated according to mycoresistance as well (see Table 5.9 for details). Additionally, we calculate precision scores for 10 randomly chosen mutations. It is used to check accuracy of our proposals. As it can be seen from Tables 5.10 and 5.10, precision scores of variants proposed by our model are higher than precision scores estimated for randomly chosen 10 mutations. Therefore, we deduce that our model detects mutations that actually play a role in the development of antibiotic resistance.

Table 5.9: Precision Comparison for 10 Chosen Variants According to Mycoresistance

| | **Precision For** | | |
|---|---|---|---|
| **Variant Detector** | **Investigated Drug** | **All Drugs** | **Drug** |
| XGBoost | 0.90 | 0.90 | Isoniazid |
| Reference | 0.80 | 0.90 | |
| XGBoost | 1.00 | 1.00 | Rifampicin |
| Reference | 0.80 | 0.80 | |
| XGBoost | 0.30 | 0.70 | Ethambutol |
| Reference | 0.80 | 0.80 | |
| XGBoost | 0.20 | 0.90 | Pyrazinamide |
| Reference | 0.40 | 0.60 | |

Table 5.10: Precision Scores of XGBoost for 10 Randomly Chosen Variants

| | **Precision For** | | |
|---|---|---|---|
| **Variant Detector** | **Investigated Drug** | **All Drugs** | **Drug** |
| XGBoost | 0.00 | 0.10 | Isoniazid |
| XGBoost | 0.00 | 0.10 | Rifampicin |
| XGBoost | 0.10 | 0.20 | Ethambutol |
| XGBoost | 0.00 | 0.10 | Pyrazinamide |

Additionally we increase the threshold to include 20 mutations with the highest SHAP values in the proposals. Then we estimate precision scores for proposed 20 mutations according to mycoresistance database. Results can be seen in the table 5.11. We could not compare xgboost with the reference algorithm because we do not know its proposals when the threshold is increased to 20. Although precision scores are decreased, most of mutations in the proposal are related to the development of antibiotic resistance for isoniazid and rifampicin. However, small portion of proposed mutations for ethambutol and pyrazinamide are actually related to the antibiotic resistance. We focus on 10 mutations that we used to estimate precision scores in the table 5.9 due to higher precision score of our model.

Table 5.11: Precision Scores of XGBoost for 20 Chosen Variants

| | **Precision For** | | |
|---|---|---|---|
| **Variant Detector** | **Investigated Drug** | **All Drugs** | **Drug** |
| XGBoost | 0.80 | 0.85 | Isoniazid |
| XGBoost | 0.75 | 0.80 | Rifampicin |
| XGBoost | 0.35 | 0.70 | Ethambutol |
| XGBoost | 0.10 | 0.60 | Pyrazinamide |

In the table 5.12, there is a column, which can take a binary (true or false) value, named MDR/XDR. If any mutation in the table 5.12 has 'TRUE' in MDR/XDR column, this mutation leads into resistance to both isoniazid and rifampicin. MDR is referring to bacteria which have resistance to at least both isoniazid and rifampicin [6]. XDR is used for bacteria that are resistant to at least one of the injectable second-line antibiotics in addition to isoniazid, rifampicin and any fluoroquinolones [6].

All mutations in our hypothesis are not included by dreamtb and the reference paper as leading into antimicrobial resistance but they are detected by our model. Additionally, we observe that gyrA, rpsL play a role in the development of antibiotic resistance to isoniazid and rifampicin although they do not have any impact on antibiotic resistance to target drugs according to T.Walker [8] and dreamtb. rpsA is found to cause resistance to isoniaid and rifampicin in addition to pyrazinamide. Eventhough some mutations are detected to lead to the development of antimicrobial resistance by our model, there are no evidence in mycoresistance database to support this hypothesis. These mutations are shown by 'No reference' phrase in the 'Reference Papers' column in the table 5.12. It is worth to investigate these mutations because predictions of xgboost about resistance determinant variants are accurate.

Table 5.12: Proof for Proposed 10 Mutations

| **Mutation** | **Reference Papers** | **MDR/XDR** | **Drug** |
|---|---|---|---|
| gyrA_S95T | [52–54] | True | Isoniazid |
| gyrA_E21Q | [55] | True | |
| rpsA_R212R | [56] | True | |
| rpsL_K43R | [57] | True | |
| embB_M306V | [58, 59] | True | |
| embB_M306I | [59, 60] | True | |
| gyrA_S95T | [52–54] | True | Rifampicin |
| gyrA_E21Q | [55] | True | |
| rpsA_R212R | [56] | True | |
| rpsL_K43R | [57, 61, 62] | True | |
| embB_M306V | [58, 59] | True | |
| rpoB_G876G | No reference | False | Ethambutol |
| rrs_S467S | No reference | False | |
| gyrA_E21Q | No reference | True | |
| rpsL_K43R | No reference | True | |
| embC_R927R | No reference | False | |
| gyrA_G668D | No reference | False | Pyrazinamide |
| gyrA_E21Q | No reference | True | |
| ahpC_G-88A | No reference | False | |
| rpsA_R212R | [56] | True | |

# 6. DISCUSSION

In this thesis, we compared different machine learning algorithms to classify mycobacterium tuberculosis samples according to their resistance to 4 first-line antibiotics. Investigated algorithms included traditional algorithms as well as neural network approaches. Algorithms, that we studied during this research, can be seen as follows support vector machine with linear and rbf kernel, logitic regression, random forests, XGBoost, feedforward neural network with 1, 2, and 3 hidden layers.

We check whether machine learning approaches learn actually information from the data set by comparing them with simple baseline classifiers. To conduct this comparison we define 3 different base estimators. In the first one all samples are labelled as susceptible to the drug. In the second one all samples are marked resistant. The last estimator classify a sample randomly with respect to its ratio of resistant to susceptible labels for the antibiotic. All machine learning approaches surpass these baseline classifiers for all experiments conducted for each antibiotic separately. Therefore, we deduce that all of our machine learning models are actually learning from the data set.

Traditional algorithms interestingly have better classification performances than feedforward neural networks do. However there is no clear winner among machine learning models that we investigate during this study, different algorithms overcome others for different antibiotics. For instance, Random forests has the highest sensitivity for isoniazid. However, it falls behind other algorithms for other drugs. XGBoost is one of algorithms with best scores in average. It is always one of the most successful algorithms in experiments conducted for each antibiotics.

In feedforward neural networks, complex models learn more information about susceptible class. When the hidden layer size is increased in multilayer perceptrons, it generally results in higher specificity (except pyrazinamide). It is possible to deduce that complex MLPs is better to conduct predictions about negative class. Even if we try to handle class imbalance in the data set via introducing class weights into the

system, it is not fully solved. Since in almost all cases, sensitivity scores fall behind specificity. That is to say, almost in all experiments, success rate in prediction of negative class is higher than resistant class.

Another point that we examine during this research is whether different data representations used in NLP lead to any improvement in the performance of investigated machine learning approaches. We evaluate TF-IDF, TF-RF, BM25TF-IDF, BM25TF-RF. According to our experiments, there is no clear indicator to performance increase in results of examined models. We observe different effects of each data representation on different models (see Appendix B for details).

We investigate whether our trained model can detect antibiotic resistance causing variants. There is a database named dreamtb containing information about mutations on mycobaterium tuberculosis and whether they play a role in the development of antibiotic resistance. We accept the database as the ground truth. We calculate precision scores on variants proposed by our xgboost and the reference paper [5]. Predictions conducted in the reference paper are more accurate according to dreamtb (see Table 5.8 for details). We also compare variants detected by xgboost algorithm with variants in dreamtb. Our related deductions can be seen below:

- 2 out of 10 the most important variants with respect to xgboost are actually resistance causing mutations for isoniazid.
- 4 out of 10 the most important variants with respect to xgboost are actually resistance causing mutations for rifampicin.
- 2 out of 10 the most important variants with respect to xgboost are actually resistance causing mutations for ethambutol.
- 1 out of 10 then most important variants with respect to xgboost is actually resistance causing mutations for pyrazinamide.

Additionally, we conduct further investigations about variants not included by dreamtb and the reference study [5] to prove that our model predictions about resistance determinants are accurate. We validate mutations proposed by xgboost by

mycoresistance database which is an alternative to dreamtb. It contains pubmed ids for papers about mutations detected to cause the development of antimicrobial resistance. Some variants proposed by our model are validated by mycoresistance although they predicted as not leading resistance to target drugs by dreamtb and the reference paper [5]. On the other hand there are some mutations that we are not able to find any proof that they cause antibiotic resistance to drugs investigated in this research (see Table 5.12 for evidence of proposed mutations). We detect that some genes such as gyrA, rpsL and rpsA are misclassified by T.Walker [8]. We find that these three genes have an impact on the development of resistance to isoniazid and rifampicin.

# 7. CONCLUSION AND FUTURE WORKS

In this study we implemented and evaluated different machine learning algorithms to detect antibiotic resistance in mycobacterium tuberculosis. We included only variants observed on some preselected genes (23 genes previously detected to play a role in antibiotic resistance development). Different algorithms was evaluated and xgboost had relatively high scores among them. Eventhough the impact of class imbalance was decreased, it could not be solved completely. We also analyzed different term frequency based data representations used in the information retrieval. We did not observe serious improvements in model performances. In addition to investigation mentioned above, we also tried to detect variants that play a role in development of drug resistance to target drugs via interpretation methods for machine learning algorithms and we proposed some mutations that have an impact in the development of antibiotic resistance to target drugs. We also identified some genes playing a role in the development of resistance to isoniazid and rifampicin which were not covered by T.Walker [8].

As future work, we will include all variants observed on bacteria samples in the data set instead of 23 target genes. We may be ignoring some important information that might help to detect antibiotic resistance. Additionally, we will pay attention to architecture utilizing convolutional neural networks because an end-to-end algorithm may recognize some unknown relations among variants observed on mycobacterium tuberculosis DNAs.

We will also try to interpret samples in the data set which are not labeled by laboratory experiments via semi-supervised learning techniques. In addition, we will investigate whether techniques similar to word embedding used in NLP improve performances of machine learning approaches in bioinformatics domain as well.

# REFERENCES


1. Walker, T. M., T. A. Kohl, S. V. Omar, J. Hedge, C. D. O. Elias, P. Bradley, Z. Iqbal, S. Feuerriegel, K. E. Niehaus, D. J. Wilson *et al.*, "Whole-genome sequencing for prediction of Mycobacterium tuberculosis drug susceptibility and resistance: a retrospective cohort study", *The Lancet infectious diseases*, Vol. 15, No. 10, pp. 1193–1202, 2015.

2. Arango-Argoty, G., E. Garner, A. Pruden, L. S. Heath, P. Vikesland and L. Zhang, "DeepARG: a deep learning approach for predicting antibiotic resistance genes from metagenomic data", *Microbiome*, Vol. 6, No. 1, p. 23, 2018.

3. Chen, M. L., A. Doddi, J. Royer, L. Freschi, M. Schito, M. Ezewudo, I. S. Kohane, A. Beam and M. Farhat, "Deep Learning Predicts Tuberculosis Drug Resistance Status from Whole-Genome Sequencing Data", *bioRxiv*, p. 275628, 2018.

4. Yang, Y., K. E. Niehaus, T. M. Walker, Z. Iqbal, A. S. Walker, D. J. Wilson, T. E. Peto, D. W. Crook, E. G. Smith, T. Zhu *et al.*, "Machine learning for classifying tuberculosis drug-resistance from DNA sequencing data", *Bioinformatics*, Vol. 34, No. 10, pp. 1666–1671, 2017.

5. Kouchaki, S., Y. Yang, T. M. Walker, A. Sarah Walker, D. J. Wilson, T. E. Peto, D. W. Crook, C. Consortium and D. A. Clifton, "Application of machine learning techniques to tuberculosis drug resistance analysis", *Bioinformatics*, Vol. 35, No. 13, pp. 2276–2282, 2018.

6. Shamout, F., T. Zhu, Y. Yang, D. A. Clifton, T. M. Walker, A. S. Walker, T. E. A. Peto, D. W. Crook and D. J. Wilson, "DeepAMR for predicting co-occurrent resistance of Mycobacterium tuberculosis", , 01 2019, `https://doi.org/10.1093/bioinformatics/btz067`.

7. Alpaydin, E., *Introduction to machine learning*, p. 354, MIT press, 2014.

8. Walker, T. M., T. A. Kohl, S. V. Omar, J. Hedge, C. D. O. Elias, P. Bradley, Z. Iqbal, S. Feuerriegel, K. E. Niehaus, D. J. Wilson *et al.*, "Whole-genome sequencing for prediction of Mycobacterium tuberculosis drug susceptibility and resistance: a retrospective cohort study", *The Lancet infectious diseases*, Vol. 15, No. 10, pp. 1193–1202, 2015.

9. Ventola, C. L., "The Antibiotic Resistance Crisis", *Pharmacy and Therapeutics.*, Vol. 40, No. 4, pp. 277–283, 2015.

10. WHO, *Antimicrobial resistance: global report on surveillance*, World Health Organization, 2014.

11. Didelot, X., R. Bowden, D. J. Wilson, T. E. A. Peto and D. W. Crook, "Transforming clinical microbiology with bacterial genome sequencing", *Nature Reviews Genetics*, Vol. 13, p. 601–612, 2012, + http://dx.doi.org/10.1038/nrg3226.

12. Bennett, P. M., "Plasmid encoded antibiotic resistance: acquisition and transfer of antibiotic resistance genes in bacteria", *British Journal of Pharmacology*, Vol. 153, No. S1, pp. S347–S357, https://bpspubs.onlinelibrary.wiley.com/doi/abs/10.1038/sj.bjp.0707607.

13. Lupo, A., S. Coyne and T. Berendonk, "Origin and Evolution of Antibiotic Resistance: The Common Mechanisms of Emergence and Spread in Water Bodies", *Frontiers in Microbiology*, Vol. 3, p. 18, 2012, https://www.frontiersin.org/article/10.3389/fmicb.2012.00018.

14. Woodford, N. and M. Ellington, "The emergence of antibiotic resistance by mutation", *Clinical Microbiology and Infection*, Vol. 13, No. 1, pp. 5 – 18, 2007, http://www.sciencedirect.com/science/article/pii/S1198743X14615500.

15. Webber, M. A. and L. J. V. Piddock, "The importance of efflux pumps in bacterial antibiotic resistance", *Journal of Antimicrobial Chemotherapy*, Vol. 51, No. 1, pp. 9–11, 2003, http://dx.doi.org/10.1093/jac/dkg050.

16. Davies, J. and D. Davies, "Origins and Evolution of Antibiotic Resistance", *Microbiology and Molecular Biology Reviews*, Vol. 74, No. 3, pp. 417–433, 2010, https://mmbr.asm.org/content/74/3/417.

17. Warnes, S. L., C. J. Highmore and C. W. Keevil, "Horizontal Transfer of Antibiotic Resistance Genes on Abiotic Touch Surfaces: Implications for Public Health", *mBio*, Vol. 3, No. 6, 2012, https://mbio.asm.org/content/3/6/e00489-12.

18. Boto, L., "Horizontal gene transfer in evolution: facts and challenges", *Proceedings of the Royal Society of London B: Biological Sciences*, Vol. 277, No. 1683, pp. 819–827, 2010, http://rspb.royalsocietypublishing.org/content/277/1683/819.

19. The Editors of Encyclopædia Britannica, "Transposon", *Encyclopædia Britannica*, 2018, https://www.britannica.com/science/transposon.

20. Bennett, P. M., "Integrons and gene cassettes: a genetic construction kit for bacteria", *Journal of Antimicrobial Chemotherapy*, Vol. 43, No. 1, pp. 1–4, 1999, http://dx.doi.org/10.1093/jac/43.1.1.

21. Domingues, S., G. J. da Silva and K. M. Nielsen, "Integrons", *Mobile Genetic Elements*, Vol. 2, No. 5, pp. 211–223, 2012, https://doi.org/10.4161/mge.22967, pMID: 23550063.

22. Cortes, C. and V. Vapnik, "Support-vector networks", *Machine learning*, Vol. 20, No. 3, pp. 273–297, 1995.

23. Alpaydin, E., *Introduction to machine learning*, p. 351, MIT press, 2014.

24. Friedman, J., T. Hastie and R. Tibshirani, *The elements of statistical learning*, Vol. 1, Springer series in statistics New York, 2001.

25. Alpaydin, E., *Introduction to machine learning*, p. 353, MIT press, 2014.

26. Friedman, J., T. Hastie and R. Tibshirani, *The elements of statistical learning*, p. 423, Springer series in statistics New York, 2001.

27. Robert, C., *Machine learning, a probabilistic perspective*, Taylor & Francis, 2014.

28. Goodfellow, I., Y. Bengio and A. Courville, *Deep learning*, MIT press, 2016.

29. Shapley, L. S., "A value for n-person games", *Contributions to the Theory of Games*, Vol. 2, No. 28, pp. 307–317, 1953.

30. Yang, Y., K. E. Niehaus, T. M. Walker, Z. Iqbal, A. S. Walker, D. J. Wilson, T. E. Peto, D. W. Crook, E. G. Smith, T. Zhu *et al.*, "Machine learning for classifying tuberculosis drug-resistance from DNA sequencing data", *Bioinformatics*, Vol. 34, No. 10, pp. 1666–1671, 2017.

31. "Prediction of Susceptibility to First-Line Tuberculosis Drugs by DNA Sequencing", *New England Journal of Medicine*, Vol. 379, No. 15, pp. 1403–1415, 2018, `https://doi.org/10.1056/NEJMoa1800474`, pMID: 30280646.

32. Fields, C. J., M. L. Heuer, N. Goto, P. J. A. Cock and P. M. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants", *Nucleic Acids Research*, Vol. 38, No. 6, pp. 1767–1771, 12 2009, `https://dx.doi.org/10.1093/nar/gkp1137`.

33. Metzker, M. L., "Sequencing technologies—the next generation", *Nature reviews genetics*, Vol. 11, No. 1, p. 31, 2010.

34. Ben-Kiki, O., C. Evans and B. Ingerson, "Yaml ain't markup language (yaml$^{\text{TM}}$) version 1.1", *yaml. org, Tech. Rep*, p. 23, 2005.

35. Lunter, G. and M. Goodson, "Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads", *Genome research*, Vol. 21, No. 6, pp. 936–939, 2011.

36. Sedlazeck, F. J., P. Rescheneder and A. Von Haeseler, "NextGenMap: fast and accurate read mapping in highly polymorphic genomes", *Bioinformatics*, Vol. 29, No. 21, pp. 2790–2791, 2013.

37. Li, H., "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM", , mar 2013, http://arxiv.org/abs/1303.3997.

38. Wu, T. D. and S. Nacu, "Fast and SNP-tolerant detection of complex variants and splicing in short reads.", *Bioinformatics (Oxford, England)*, Vol. 26, No. 7, pp. 873–81, apr 2010, http://www.ncbi.nlm.nih.gov/pubmed/20147302 http://www.ncbi.nlm.nih.gov/pubmed/20147302.

39. Tian, S., H. Yan, C. Neuhauser and S. L. Slager, "An analytical workflow for accurate variant discovery in highly divergent regions", *BMC Genomics*, Vol. 17, No. 1, p. 703, Sep 2016, https://doi.org/10.1186/s12864-016-3045-z.

40. Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis and R. Durbin, "The sequence alignment/map format and SAMtools", *Bioinformatics*, Vol. 25, No. 16, pp. 2078–2079, 2009.

41. McKenna, A., M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly *et al.*, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data", *Genome research*, 2010.

42. "GATK best practices workflow", https://software.broadinstitute.org/gatk/best-practices/workflow?id=11145, accessed: 2019-12-04.

43. Danecek, P., A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry *et al.*, "The variant call format and VCFtools", *Bioinformatics*, Vol. 27, No. 15, pp. 2156–2158, 2011.

44. Rimmer, A., H. Phan, I. Mathieson, Z. Iqbal, S. R. F. Twigg, A. O. M.

Wilkie, G. McVean, G. Lunter and G. Lunter, "Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications", *Nature Genetics*, Vol. 46, No. 8, pp. 912–918, aug 2014, `http://www.nature.com/articles/ng.3036`.

45. Lan, M., C. L. Tan, J. Su and Y. Lu, "Statistical representation models for mutation information within genomic data", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 31, No. 4, pp. 721–735, 2008.

46. Lan, M., C. L. Tan, J. Su and Y. Lu, "Supervised and traditional term weighting methods for automatic text categorization", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 31, No. 4, pp. 721–735, 2008.

47. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

48. Chen, T. and C. Guestrin, "Xgboost: A scalable tree boosting system", *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.

49. Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch", *NIPS-W*, 2017.

50. Sandgren, A., M. Strong, P. Muthukrishnan, B. K. Weiner, G. M. Church and M. B. Murray, "Tuberculosis drug resistance mutation database", *PLoS medicine*, Vol. 6, No. 2, p. e1000002, 2009.

51. Dai, E., H. Zhang, X. Zhou, Q. Song, D. Li, L. Luo, X. Xu, W. Jiang and H. Ling, "MycoResistance: a curated resource of drug resistance molecules in Mycobacte-

ria", *Database*, Vol. 2019, 2019.

52. Liu, C. H., H. M. Li, N. Lu, Q. Wang, Y. L. Hu, X. Yang, Y. F. Hu, P. C. Woo, G. F. Gao and B. Zhu, "Genomic sequence based scanning for drug resistance-associated mutations and evolutionary analysis of multidrug-resistant and extensively drug-resistant Mycobacterium tuberculosis", *Journal of Infection*, Vol. 65, No. 5, pp. 412–422, 2012.

53. Siddiqi, N., M. Shamim, A. Amin, D. Chauhan, R. Das, K. Srivastava, D. Singh, V. Sharma, V. Katoch, S. Sharma *et al.*, "Typing of drug resistant isolates of Mycobacterium tuberculosis from India using the IS6110 element reveals substantive polymorphism", *Infection, Genetics and Evolution*, Vol. 1, No. 2, pp. 109–116, 2001.

54. Dookie, N., A. W. Sturm and P. Moodley, "Moxifloxacin resistance in the F15/LAM4/KZN extensively drug-resistant strain of Mycobacterium tuberculosis", *Infection and drug resistance*, Vol. 7, p. 223, 2014.

55. Dookie, N., A. W. Sturm and P. Moodley, "Moxifloxacin resistance in the F15/LAM4/KZN extensively drug-resistant strain of Mycobacterium tuberculosis", *Infection and drug resistance*, Vol. 7, p. 223, 2014.

56. Akhmetova, A., U. Kozhamkulov, V. Bismilda, L. Chingissova, T. Abildaev, M. Dymova, M. Filipenko and E. Ramanculov, "Mutations in the pncA and rpsA genes among 77 Mycobacterium tuberculosis isolates in Kazakhstan", *The International Journal of Tuberculosis and Lung Disease*, Vol. 19, No. 2, pp. 179–184, 2015.

57. Aung, H. L., T. Tun, D. Moradigaravand, C. U. Köser, W. W. Nyunt, S. T. Aung, T. Lwin, K. K. Thinn, J. A. Crump, J. Parkhill *et al.*, "Whole-genome sequencing of multidrug-resistant Mycobacterium tuberculosis isolates from Myanmar", *Journal of global antimicrobial resistance*, Vol. 6, pp. 113–117, 2016.

58. Jnawali, H. N., S. C. Hwang, Y. K. Park, H. Kim, Y. S. Lee, G. T. Chung, K. H. Choe and S. Ryoo, "Characterization of mutations in multi-and extensive drug resistance among strains of Mycobacterium tuberculosis clinical isolates in Republic of Korea", *Diagnostic microbiology and infectious disease*, Vol. 76, No. 2, pp. 187–196, 2013.

59. Shi, D., L. Li, Y. Zhao, Q. Jia, H. Li, C. Coulter, Q. Jin and G. Zhu, "Characteristics of embB mutations in multidrug-resistant Mycobacterium tuberculosis isolates in Henan, China", *Journal of antimicrobial chemotherapy*, Vol. 66, No. 10, pp. 2240–2247, 2011.

60. Jadaun, G., R. Das, P. Upadhyay, D. Chauhan, V. Sharma and V. Katoch, "Role of embCAB gene mutations in ethambutol resistance in Mycobacterium tuberculosis isolates from India", *International journal of antimicrobial agents*, Vol. 33, No. 5, pp. 483–486, 2009.

61. Zhao, L.-l., H.-c. Liu, Q. Sun, T.-y. Xiao, X.-q. Zhao, G.-l. Li, C.-y. Zeng and K.-l. Wan, "Identification of mutations conferring streptomycin resistance in multidrug-resistant tuberculosis of China", *Diagnostic microbiology and infectious disease*, Vol. 83, No. 2, pp. 150–153, 2015.

62. Jagielski, T., H. Ignatowska, Z. Bakuła, Ł. Dziewit, A. Napiórkowska, E. Augustynowicz-Kopeć, Z. Zwolska and J. Bielecki, "Screening for streptomycin resistance-conferring mutations in Mycobacterium tuberculosis clinical isolates from Poland", *PLoS One*, Vol. 9, No. 6, p. e100078, 2014.

# APPENDIX A: Hyperparameters for Each Algorithm

Table A.1: Hyperparameters for SVM

| | **Algorithms** | |
|---|---|---|
| **Hyperparameters** | **Svm linear** | **Svm rbf** |
| C | [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000] | [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000] |
| gamma | - | [0.001, 0.1, 1, 10, 100] |

Table A.2: Hyperparameters for LR

| | **Algorithm** |
|---|---|
| **Hyperparameters** | **Logistic regression** |
| C | [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000] |
| penalty | [”l2”] |
| solver | [”newton-cg”, ”lbfgs”, ”liblinear”, ”sag”, ”saga”] |

Table A.3: Hyperparameters for Regression Tree Based Algorithms

| | **Algorithms** | |
|---|---|---|
| **Hyperparameters** | **Random forests** | **XGBoost** |
| n_estimators | [100, 250, 500, 1000] | [200, 250, 300, 350, 400, 500] |
| max_features | [”sqrt”, ”log2”, null] | - |
| max_depth | - | [2, 3, 4, 5, 6, 7, 8, 9, 10] |
| learning_rate | - | [10, 1, 0.1, 0.01] |

Table A.4: Hyperparameters for Feedforward Neural Networks

| | **Algorithm** | | |
|---|---|---|---|
| **Hyperparameters** | **FFNN-1d** | **FFNN-2d** | **FFNN-3d** |
| batch_sizes | [64, 256, 512, 1024, 2048] | [64, 256, 512, 1024, 2048] | [64, 256, 512, 1024, 2048] |
| hidden_units | [[16], [32], [64], [128], [1024], [4096], [8192]] | [[64, 64], [32, 32], [16, 16], [64, 32], [64, 16],[512, 128], [512, 512], [128,512]] | [64, 64, 64], [[64, 32, 16], [32, 32, 32], [16, 16, 16], [512, 256, 128], [512, 512, 515], [128, 256, 512]] |
| activation_functions | [[”relu”], [”leaky_relu”]] | [[”leaky_relu”, ”leaky_relu”], [”relu”, ”relu”]] | [[”leaky_relu”, ”leaky_relu”, ”leaky_relu”], [”relu”, ”relu”, ”relu”]] |
| learning_rates | [0.1, 0.01, 0.001] | [0.1, 0.01, 0.001] | [0.1, 0.01, 0.001] |
| optimizers | [”Adam”] | [”Adam”] | [”Adam”] |
| dropout_rates | [0.0, 0.25, 0.5] | [0.0, 0.25, 0.5] | [0.0, 0.25, 0.5] |

# APPENDIX B: Data Representations Comparisons for All Models

Table B.1: Svm with Linear Kernel

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.91 | 0.98 | 0.96 | 0.94 |
| TF-IDF | 0.92 | 0.99 | 0.96 | 0.94 |
| TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-IDF | 0.92 | 0.99 | 0.96 | 0.94 |
| BM25TF-RF | 0.92 | 0.98 | 0.95 | 0.94 |
| **Rifampicin** | | | | |
| binary | 0.91 | 0.99 | 0.96 | 0.94 |
| TF-IDF | 0.91 | 0.99 | 0.96 | 0.93 |
| TF-RF | 0.91 | 0.99 | 0.97 | 0.94 |
| BM25TF-IDF | 0.91 | 0.99 | 0.95 | 0.93 |
| BM25TF-RF | 0.91 | 0.99 | 0.96 | 0.94 |
| **Ethambutol** | | | | |
| binary | 0.91 | 0.91 | 0.62 | 0.74 |
| TF-IDF | 0.89 | 0.93 | 0.69 | 0.77 |
| TF-RF | 0.91 | 0.93 | 0.69 | 0.78 |
| BM25TF-IDF | 0.90 | 0.93 | 0.68 | 0.78 |
| BM25TF-RF | 0.89 | 0.93 | 0.69 | 0.78 |
| **Pyrazinamide** | | | | |
| binary | 0.74 | 0.94 | 0.64 | 0.69 |
| TF-IDF | 0.79 | 0.94 | 0.63 | 0.70 |
| TF-RF | 0.76 | 0.95 | 0.67 | 0.71 |
| BM25TF-IDF | 0.71 | 0.94 | 0.62 | 0.66 |
| BM25TF-RF | 0.80 | 0.94 | 0.66 | 0.72 |

Table B.2: Svm with Rbf Kernel

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.91 | 0.98 | 0.96 | 0.94 |
| TF-IDF | 0.92 | 0.99 | 0.96 | 0.94 |
| TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-IDF | 0.92 | 0.99 | 0.96 | 0.94 |
| BM25TF-RF | 0.92 | 0.98 | 0.95 | 0.94 |
| **Rifampicin** | | | | |
| binary | 0.91 | 0.99 | 0.96 | 0.94 |
| TF-IDF | 0.91 | 0.99 | 0.96 | 0.93 |
| TF-RF | 0.91 | 0.99 | 0.97 | 0.94 |
| BM25TF-IDF | 0.91 | 0.99 | 0.95 | 0.93 |
| BM25TF-RF | 0.91 | 0.99 | 0.96 | 0.94 |
| **Ethambutol** | | | | |
| binary | 0.91 | 0.91 | 0.62 | 0.74 |
| TF-IDF | 0.89 | 0.93 | 0.69 | 0.77 |
| TF-RF | 0.91 | 0.93 | 0.69 | 0.78 |
| BM25TF-IDF | 0.90 | 0.93 | 0.68 | 0.78 |
| BM25TF-RF | 0.89 | 0.93 | 0.69 | 0.78 |
| **Pyrazinamide** | | | | |
| binary | 0.71 | 0.94 | 0.64 | 0.69 |
| TF-IDF | 0.79 | 0.94 | 0.63 | 0.70 |
| TF-RF | 0.76 | 0.95 | 0.67 | 0.71 |
| BM25TF-IDF | 0.71 | 0.94 | 0.62 | 0.66 |
| BM25TF-RF | 0.80 | 0.94 | 0.66 | 0.72 |

Table B.3: Logistic Regression

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.91 | 0.99 | 0.97 | 0.94 |
| TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-RF | 0.91 | 0.99 | 0.96 | 0.94 |
| **Rifampicin** | | | | |
| binary | 0.91 | 0.99 | 0.96 | 0.94 |
| TF-IDF | 0.91 | 0.99 | 0.96 | 0.93 |
| TF-RF | 0.91 | 0.99 | 0.96 | 0.93 |
| BM25TF-IDF | 0.91 | 0.99 | 0.96 | 0.93 |
| BM25TF-RF | 0.91 | 0.99 | 0.96 | 0.94 |
| **Ethambutol** | | | | |
| binary | 0.91 | 0.91 | 0.63 | 0.75 |
| TF-IDF | 0.85 | 0.94 | 0.69 | 0.76 |
| TF-RF | 0.93 | 0.87 | 0.55 | 0.69 |
| BM25TF-IDF | 0.87 | 0.93 | 0.68 | 0.76 |
| BM25TF-RF | 0.87 | 0.94 | 0.70 | 0.78 |
| **Pyrazinamide** | | | | |
| binary | 0.75 | 0.95 | 0.66 | 0.70 |
| TF-IDF | 0.77 | 0.93 | 0.62 | 0.68 |
| TF-RF | 0.75 | 0.94 | 0.64 | 0.69 |
| BM25TF-IDF | 0.75 | 0.94 | 0.62 | 0.68 |
| BM25TF-RF | 0.74 | 0.95 | 0.65 | 0.70 |

Table B.4: Random Forests

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.95 | 0.91 | 0.82 | 0.88 |
| TF-IDF | 0.96 | 0.92 | 0.83 | 0.89 |
| TF-RF | 0.96 | 0.92 | 0.83 | 0.89 |
| BM25TF-IDF | 0.96 | 0.92 | 0.83 | 0.89 |
| BM25TF-RF | 0.96 | 0.92 | 0.83 | 0.89 |
| **Rifampicin** | | | | |
| binary | 0.88 | 0.99 | 0.96 | 0.92 |
| TF-IDF | 0.90 | 0.98 | 0.92 | 0.91 |
| TF-RF | 0.90 | 0.98 | 0.92 | 0.91 |
| BM25TF-IDF | 0.90 | 0.98 | 0.92 | 0.91 |
| BM25TF-RF | 0.90 | 0.98 | 0.92 | 0.91 |
| **Ethambutol** | | | | |
| binary | 0.68 | 0.91 | 0.56 | 0.61 |
| TF-IDF | 0.78 | 0.90 | 0.57 | 0.66 |
| TF-RF | 0.77 | 0.90 | 0.56 | 0.65 |
| BM25TF-IDF | 0.79 | 0.90 | 0.57 | 0.66 |
| BM25TF-RF | 0.79 | 0.90 | 0.57 | 0.66 |
| **Pyrazinamide** | | | | |
| binary | 0.55 | 0.98 | 0.79 | 0.65 |
| TF-IDF | 0.57 | 0.98 | 0.77 | 0.66 |
| TF-RF | 0.57 | 0.98 | 0.77 | 0.66 |
| BM25TF-IDF | 0.57 | 0.98 | 0.77 | 0.66 |
| BM25TF-RF | 0.57 | 0.98 | 0.77 | 0.66 |

Table B.5: XGBoost

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.93 | 0.98 | 0.96 | 0.94 |
| TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-IDF | 0.92 | 0.98 | 0.96 | 0.94 |
| BM25TF-RF | 0.92 | 0.98 | 0.96 | 0.94 |
| **Rifampicin** | | | | |
| binary | 0.91 | 0.99 | 0.95 | 0.93 |
| TF-IDF | 0.90 | 0.98 | 0.95 | 0.92 |
| TF-RF | 0.90 | 0.98 | 0.95 | 0.92 |
| BM25TF-IDF | 0.90 | 0.98 | 0.95 | 0.92 |
| BM25TF-RF | 0.90 | 0.98 | 0.95 | 0.92 |
| **Ethambutol** | | | | |
| binary | 0.89 | 0.92 | 0.66 | 0.76 |
| TF-IDF | 0.90 | 0.92 | 0.66 | 0.76 |
| TF-RF | 0.90 | 0.92 | 0.66 | 0.76 |
| BM25TF-IDF | 0.90 | 0.92 | 0.66 | 0.76 |
| BM25TF-RF | 0.90 | 0.92 | 0.66 | 0.76 |
| **Pyrazinamide** | | | | |
| binary | 0.70 | 0.95 | 0.68 | 0.69 |
| TF-IDF | 0.66 | 0.95 | 0.64 | 0.65 |
| TF-RF | 0.66 | 0.95 | 0.64 | 0.65 |
| BM25TF-IDF | 0.66 | 0.95 | 0.64 | 0.65 |
| BM25TF-RF | 0.66 | 0.95 | 0.64 | 0.65 |

Table B.6: Feedforward Neural Networks with 1 Hidden Layer

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.90 | 0.95 | 0.88 | 0.89 |
| TF-IDF | 0.87 | 0.97 | 0.92 | 0.89 |
| TF-RF | 0.88 | 0.97 | 0.93 | 0.90 |
| BM25TF-IDF | 0.87 | 0.97 | 0.91 | 0.89 |
| BM25TF-RF | 0.88 | 0.96 | 0.90 | 0.89 |
| **Rifampicin** | | | | |
| binary | 0.92 | 0.91 | 0.77 | 0.84 |
| TF-IDF | 0.87 | 0.97 | 0.91 | 0.89 |
| TF-RF | 0.88 | 0.96 | 0.86 | 0.87 |
| BM25TF-IDF | 0.85 | 0.97 | 0.90 | 0.87 |
| BM25TF-RF | 0.88 | 0.97 | 0.91 | 0.90 |
| **Ethambutol** | | | | |
| binary | 0.76 | 0.92 | 0.62 | 0.68 |
| TF-IDF | 0.82 | 0.90 | 0.59 | 0.69 |
| TF-RF | 0.81 | 0.94 | 0.70 | 0.75 |
| BM25TF-IDF | 0.81 | 0.88 | 0.53 | 0.64 |
| BM25TF-RF | 0.76 | 0.93 | 0.63 | 0.69 |
| **Pyrazinamide** | | | | |
| binary | 0.63 | 0.97 | 0.77 | 0.69 |
| TF-IDF | 0.65 | 0.96 | 0.70 | 0.67 |
| TF-RF | 0.68 | 0.96 | 0.71 | 0.69 |
| BM25TF-IDF | 0.65 | 0.96 | 0.71 | 0.68 |
| BM25TF-RF | 0.67 | 0.96 | 0.70 | 0.68 |

Table B.7: Feedforward Neural Networks with 2 Hidden Layers

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.86 | 0.96 | 0.89 | 0.88 |
| TF-IDF | 0.85 | 0.96 | 0.89 | 0.87 |
| TF-RF | 0.87 | 0.98 | 0.95 | 0.90 |
| BM25TF-IDF | 0.88 | 0.94 | 0.86 | 0.87 |
| BM25TF-RF | 0.87 | 0.97 | 0.92 | 0.89 |
| **Rifampicin** | | | | |
| binary | 0.87 | 0.98 | 0.93 | 0.90 |
| TF-IDF | 0.86 | 0.96 | 0.88 | 0.87 |
| TF-RF | 0.89 | 0.98 | 0.94 | 0.92 |
| BM25TF-IDF | 0.85 | 0.97 | 0.90 | 0.87 |
| BM25TF-RF | 0.89 | 0.97 | 0.91 | 0.90 |
| **Ethambutol** | | | | |
| binary | 0.66 | 0.95 | 0.70 | 0.68 |
| TF-IDF | 0.77 | 0.93 | 0.65 | 0.71 |
| TF-RF | 0.81 | 0.94 | 0.69 | 0.75 |
| BM25TF-IDF | 0.59 | 0.96 | 0.72 | 0.65 |
| BM25TF-RF | 0.73 | 0.95 | 0.72 | 0.72 |
| **Pyrazinamide** | | | | |
| binary | 0.61 | 0.96 | 0.71 | 0.66 |
| TF-IDF | 0.62 | 0.96 | 0.70 | 0.66 |
| TF-RF | 0.64 | 0.97 | 0.75 | 0.69 |
| BM25TF-IDF | 0.54 | 0.98 | 0.80 | 0.64 |
| BM25TF-RF | 0.56 | 0.98 | 0.78 | 0.65 |

Table B.8: Feedforward Neural Networks with 3 Hidden Layers

| Data Representations | Sensitivity | Specificity | Precision | F1 Score |
|---|---|---|---|---|
| **Isoniazid** | | | | |
| binary | 0.85 | 0.97 | 0.93 | 0.89 |
| TF-IDF | 0.86 | 0.97 | 0.93 | 0.89 |
| TF-RF | 0.83 | 0.98 | 0.94 | 0.88 |
| BM25TF-IDF | 0.89 | 0.96 | 0.89 | 0.89 |
| BM25TF-RF | 0.90 | 0.96 | 0.91 | 0.91 |
| **Rifampicin** | | | | |
| binary | 0.88 | 0.98 | 0.94 | 0.91 |
| TF-IDF | 0.88 | 0.98 | 0.93 | 0.91 |
| TF-RF | 0.89 | 0.98 | 0.95 | 0.92 |
| BM25TF-IDF | 0.88 | 0.98 | 0.93 | 0.90 |
| BM25TF-RF | 0.88 | 0.98 | 0.93 | 0.91 |
| **Ethambutol** | | | | |
| binary | 0.62 | 0.96 | 0.74 | 0.68 |
| TF-IDF | 0.66 | 0.96 | 0.71 | 0.68 |
| TF-RF | 0.72 | 0.95 | 0.72 | 0.72 |
| BM25TF-IDF | 0.69 | 0.96 | 0.72 | 0.70 |
| BM25TF-RF | 0.61 | 0.97 | 0.77 | 0.68 |
| **Pyrazinamide** | | | | |
| binary | 0.66 | 0.95 | 0.65 | 0.66 |
| TF-IDF | 0.56 | 0.97 | 0.73 | 0.63 |
| TF-RF | 0.69 | 0.95 | 0.67 | 0.68 |
| BM25TF-IDF | 0.62 | 0.96 | 0.69 | 0.66 |
| BM25TF-RF | 0.60 | 0.97 | 0.74 | 0.66 |