Using genomic data and machine learning to predict antibiotic resistance: a tutorial paper

List of authors: Faye Orcales[1,2,#], Lucy Moctezuma[1,3,#], Meris Johnson-Hagler[1], John Matthew Suntay[1,2], Jameel Ali[1], Kristiene Recto[1], Pleuni Pennings, Ph.D[1]

[1]San Francisco State University
[2]University of California, San Francisco
[3]California State University, East Bay
[#]These authors are first authors

Abstract

Antibiotic resistance is a global public health concern. Bacteria have evolved resistance to most antibiotics, which means that for any given bacterial infection, the bacteria may be resistant to one or several antibiotics. For effective treatment and to control the spread of resistant strains of bacteria, accurate and efficient detection of resistance is important. It has been suggested that genomic sequencing and machine learning (ML) could make resistance testing more accurate and cost-effective. Given that ML is likely to become an ever more important tool in medicine, we believe that it is important for pre-health students and others in the life sciences to learn to use machine learning tools. This paper provides a step-by-step tutorial to learn four different ML models to predict drug resistance for *E. coli* isolates based on genomic data. The tutorial is accessible to beginners, doesn't require any software to be installed as it is based on Google Colab notebooks, and can be used in undergraduate and graduate classes.

Introduction

Machine learning (ML) is the development of algorithms that can make predictions without human intervention (1). It is used for various medical purposes thanks to the amount of data available (1). For example, studies have suggested that ML methods can identify tumors from images that radiologists have missed (2).

In recent years, many research groups have used ML methods to predict whether infections are caused by drug-resistant strains or not. For example, Lewin-Epstein and colleagues applied ML algorithms to patients' electronic medical records to predict antibiotic resistance (3); and Feretzakis and colleagues developed a model that predicts a bacterium's susceptibility to antibiotics based on the patient's demographic characteristics and previous resistance test results (4).

There is also a lot of interest in using ML in combination with genomic data to predict antibiotic resistance. Genomic data, including information on gene presence and absence and single nucleotide polymorphisms (SNPs), contain a lot of information that can be leveraged to predict phenotypes such as drug resistance. Several studies have been quite successful in using ML and genomic data to predict drug resistance in different microbial species. For example, Moradigaravand and colleagues used various ML algorithms to predict *E.coli* resistance against 11 antibiotics using whole genome sequencing data (5). Ren and colleagues led a similar study using SNP data and focused on 4 antibiotics (6). In another study Khaledi and colleagues used genomic and gene expression data to predict *Pseudomonas aeruginosa* resistance to 4 antimicrobial drugs (7).
The goal of this tutorial is to introduce the user to ML as applied to genomic data to predict drug resistance, so they may potentially use it in their own future research. We use the data and approaches from Moradigaravand et al (5).

In this paper, we will first give a brief introduction to antibiotic resistance. Next, we will describe the genomic data that can be used to train a machine learning algorithm. Then we will introduce basic concepts of machine learning, such as the different algorithms available and ways to improve machine learning performance. Finally, we will go over our tutorials, which show how to build a machine learning analysis pipeline for predicting antibiotic resistance from genomic data.

This tutorial is based on a project we did as research students in the CodeLab at San Francisco State University. The project used a dataset from Dr. Moradigaravand and colleagues (5). His dataset used genetic data and different machine learning models to predict *E.coli* resistance in bacteremia patients from the UK. The models included in this tutorial are: extreme gradient boosted tree (GBT), random forest (RF), logistic regression (LR), and neural network (NN). For each of these models there is a Google Colab Notebook, which users can work with in a browser with no need for software installations on their own computer. There are two additional notebooks for data processing and to visualize the results. Our goal is to create

an easily understandable tutorial for anyone who wishes to learn more about machine learning and its applications in predicting antibiotic resistance.

Drug resistance and the Moradigaravand dataset

Penicillin was one of the first antibiotics that helped cure bacterial infections such as strep throat, gonorrhea, and meningitis (8). Prior to the discovery of antibiotics, the average lifespan in the United States was merely 56.4 years, whereas in contemporary times, it has risen to approximately 80 years (9). However, in part due to the over-prescribing and misuse of antibiotics, resistant strains of bacteria have evolved and become quite common, reducing the usefulness of antibiotics. As drug resistance has become more common, identifying drug resistance phenotypes for pathogens is important for successful treatment of patients. Ideally, a doctor would know quickly which antibiotics they should prescribe to successfully treat their patient.

Antimicrobial susceptibility tests are laboratory procedures used to identify which antibiotic can help treat an infection (10). A sample is taken from the infected area and these tests are performed through disk diffusion or minimal inhibitory concentration methods, usually requiring incubation for at least one day, depending on the bacteria (10). These tests determine whether an isolate is susceptible, intermediate, or resistant to a range of drugs, which can then guide treatment. Antibiotic susceptibility testing is widely used, but a limitation to this procedure is the time it can take to get results. Genomic sequencing combined with machine learning has been proposed as an alternative. In this tutorial we will explain one approach to do this.

We'll be working with a dataset that was collected over many years in the United Kingdom (5). For 1936 patients who suffered from *E. coli* bloodstream infections, bacteria were isolated and tested against a range of antibiotics in the lab. An example of the phenotypic data can be seen in Table 1. Our goal here is to try and predict these antibiotic phenotypes using the genomic data from the same isolates.

Table 1. Example of antibiotic phenotypic data from the Moradigaravand dataset.

Strains	Ceftazidime	Cefotaxime	Ampicillin	Amoxicillin	Amoxicillin Clavulanate	Tazobactam	Cefuroxime	Cephalothin	Gentamicin	Tobramycin	Trimethoprim	Ciprofloxacin
Isolate 1	S	S	S	S	S	S	S	S	S	S	S	S
Isolate 2	S	S	R	R	R	S	S	S	S	S	R	R
Isolate 3	S	S	S	R	S	S	S	S	S	S	S	S
Isolate 4	S	S	R	S	R	S	S	S	S	S	S	R
Isolate 5	S	S	R	R	S	S	S	S	S	S	R	S
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.

S: Susceptible to the antibiotic
R: Resistant to the antibiotic

The genomic data used to predict resistance

In this tutorial, we follow the approaches taken by Dr. Moradigaravand and colleagues (5). While this is not the only way to use genomic data for machine learning purposes, it is a common way and a good place to start learning.
Bacterial genomes are usually sequenced using next-generation sequencing techniques. From that raw data, different bioinformatics tools are used to summarize the genomic data in different ways. To build a machine learning model to predict drug resistance, we need to choose what information from each isolate to use for the model to learn from. Moradigaravand and colleagues used two types of genomic data for this purpose: (1) a gene presence-absence table and (2) a population structure table. The third type of data they used is not genomic: (3) the year of isolation.

The gene-presence absence table summarizes for each bacterial isolate which genes are present or absent. Bacteria like *E coli* can gain and lose genes through horizontal gene transfer, leading to different gene content in each isolate. Genes that confer resistance are among the ones being lost and gained, which is why a gene presence absence table is expected to be relevant for predicting drug resistance, see Table 2.

Table 2. Example of gene presence-absence table

Strains	gene 1	gene 2	gene 3	gene 4	gene 5	[...]	gene 17199
Isolate 1	1	0	1	0	1	...	0
Isolate 2	1	1	1	1	0	...	0
Isolate 3	0	0	1	1	0	...	1
.	.	.	.	.	.	...	.
.	.	.	.	.	.	...	.
.	.	.	.	.	.	...	.

1: Presence of gene in isolate
0: Absence of gene in isolate

The population structure data represents which isolates are closely related (genetically similar) to each other (Figure 1). Population structure can be useful to predict drug resistance when related bacterial strains tend to have similar resistance profiles. The population structure data was extracted by first aligning the sequences from each isolate to a reference genome. Then, for all possible pairs of isolates, the genomes are compared to each other to calculate the number of differing sites for each pair (SNP distance). Using the adegenet package in R, these pairwise SNP distances are stored in a distance matrix, which is then used to cluster isolates together if their SNP distance is less than a cutoff value, as illustrated in Figure 1 (11).

Finally, we use the year of isolation for each isolate. Resistance levels change over time and relationships between resistance and genetic backgrounds can also change over time, which could mean that the year of isolation carries useful information for predicting drug resistance.

The variables that are used as predictors in a machine learning model are usually referred to as features. There are three groups of features in this dataset: gene presence and absence (G), population structure (S) and year of isolation (Y).

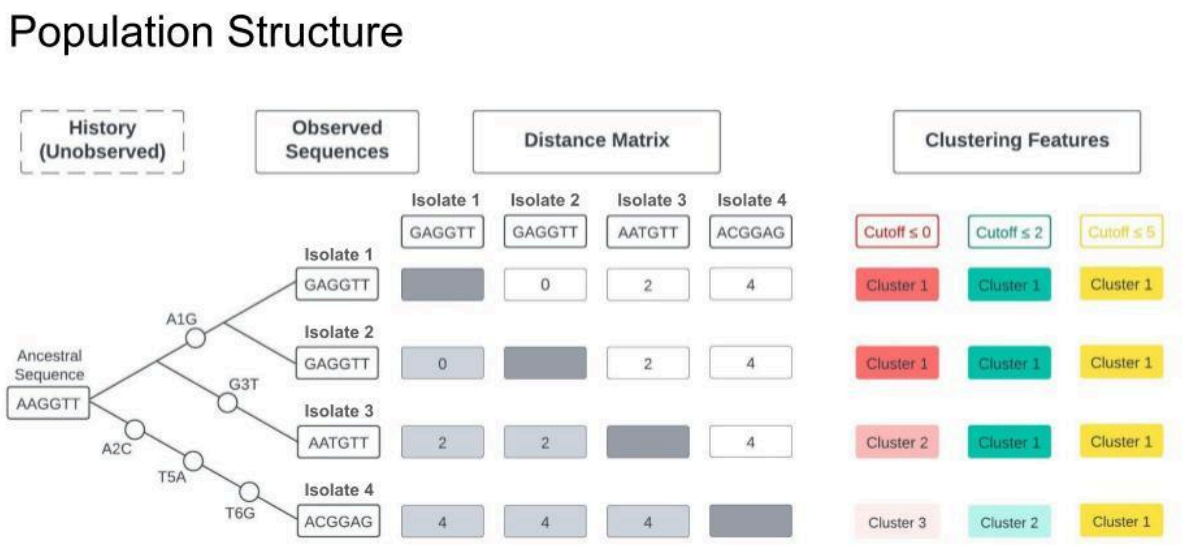

Figure 1. **Pipeline used to construct population structure data.** This figure shows from left to right an example for an ancestral sequence which mutated and led to related sequences in 4 isolates. Observed sequences are compared to each other, resulting in a distance matrix, which is then used for clustering. For each possible cut-off value, the resulting clusterings are listed in a separate column.

Introduction to Machine Learning

With this paper we aim to make machine learning more accessible for those who would like to learn it. The authors of this paper were a group of students at San Francisco State University that were excited to learn more about machine learning and its potential in antibiotic resistance research. At the start of this project, the authors were new to machine learning. As we developed the project, we gained experience in Python, genomic data, and creating machine-learning models. This tutorial is a culmination of our efforts, learning journeys, and aims to serve those in a similar learning phase.

Basic Concepts

ML algorithms (or models, we use the words interchangeably) learn from data similarly to how humans learn. The three most common ways that machine learning models learn are through reinforcement learning, unsupervised learning, and supervised learning. This tutorial explores supervised learning algorithms and covers how to prepare the data, how to split the data in

training and testing datasets, how to train the different ML models and how to assess how good the predictions are that we made.

In supervised learning, the algorithm learns by recognizing patterns from a labeled dataset. “Labeled” here means that the dataset has an “answer key”. In our case, this means that we work with isolates for which we know whether they are resistant or susceptible to different drugs (see Table 1). Before training a ML model the entire dataset is split into a training dataset and a testing dataset. Usually, around 70-80% of the data are used as the training dataset and the rest is used for testing. The training dataset is used for the algorithm to learn the patterns associated with resistance (or susceptibility). After that, the testing dataset is used to determine how well the algorithm can predict the label (resistant or susceptible).

The training data contain **features** that the model will attempt to learn patterns from and the **labels** it learns to predict. The features can be formatted in different ways, depending on the type of data used, such as pixel arrangements from an image or a set of words in a sentence. For this tutorial, the features are organized as tabular data, where each row is an observation (here a bacterial isolate) and each column is a feature (as described earlier in the manuscript, see Table 2 and Figure 1).

After the ML model is trained on the training data, the features of the **testing data** will be used afterwards for the ML model to make predictions with. The predictions will then be compared with the known labels for the testing dataset. To illustrate how supervised learning works, consider a hypothetical model that can predict house prices. This supervised learning algorithm will be given a number of features describing houses, such as number of bedrooms, size of the property, and more, as well as the associated prices of the target label. After the algorithm has been trained with the training dataset, it is used to predict house prices from the features of the testing dataset. Then we, as the user, observe what housing price predictions were made and check how well the predictions match the true values.

Supervised learning is generally used for two kinds of tasks: regression and classification. A **regression task** requires the user to model the relationship between features and a numerical variable (e.g., house price). A **classification task** on the other hand is when a class label (e.g., dog vs cat) is predicted based on features. In the previous example, predicting house prices would be a regression task. If the model were to predict only if the house was either cheap or expensive, then it would be considered a classification task.

In this tutorial, we use supervised algorithms to perform a classification task (resistant or susceptible). The model will predict whether strains of *E. coli* are resistant or susceptible across a spectrum of antibiotics. A set of six Google Colab notebooks will provide a thorough walkthrough of each of the algorithms used in that study for those who are new to machine learning.

Four Supervised Classification Models

Within the Google Colab notebooks, the user will be guided on how to implement Logistic Regression, Random Forest, Extreme Gradient-Boosted Trees and Neural Networks models to predict antibiotic resistance in *E. coli*. Each algorithm will be briefly explained below and further explanations are provided in the notebooks. For each of the models we will guide the user in assessing the quality of the predictions.

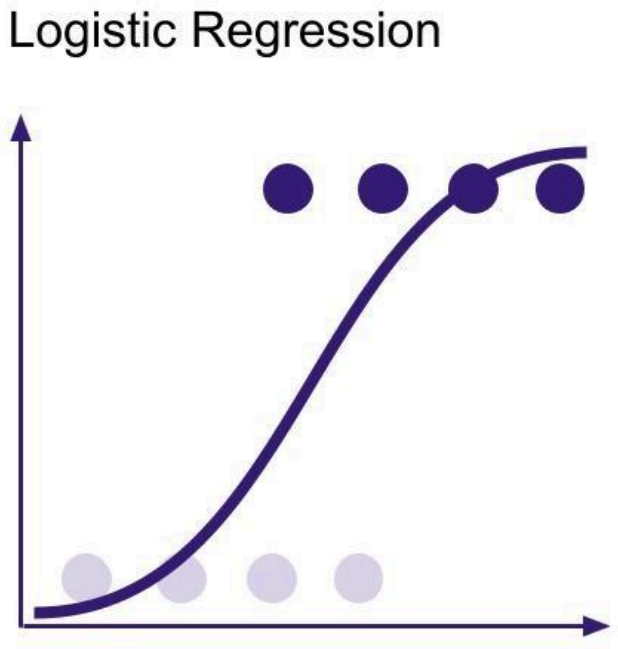

Figure 2. Logistic Regression A sigmoid function that is representative of a logistic regression model.

Logistic Regression is a classification model that predicts the probability for a binary outcome (2 classes), such as susceptible and resistant (see Figure 2). Logistic regression models have specific probability thresholds that the model uses to decide what class a particular observation belongs to: the susceptible or resistant class. The default threshold for logistic regression is 0.5. Above this value, the model predicts one class; below, it predicts the other class (12).

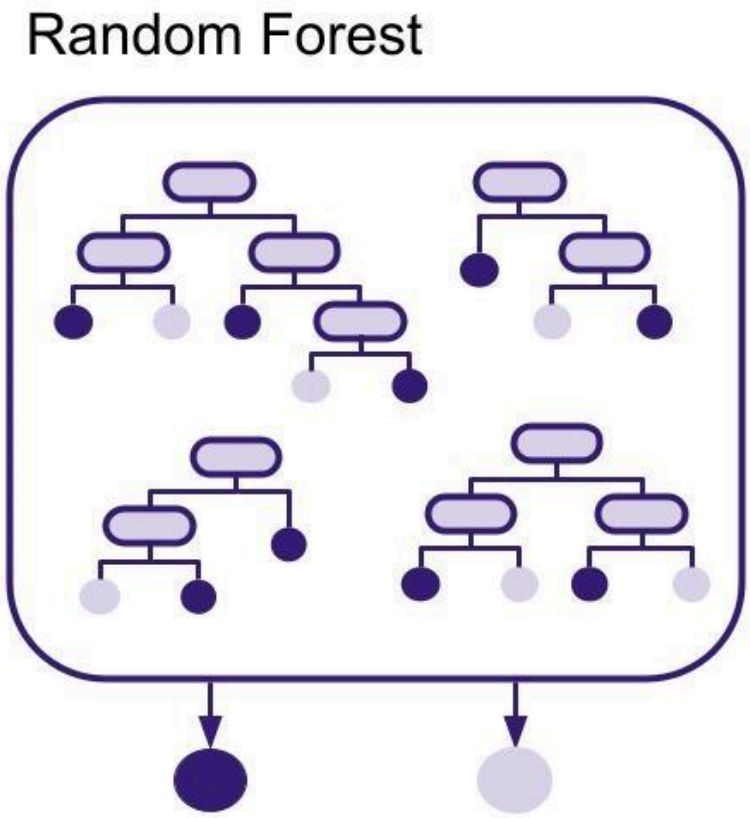

Figure 3. Random Forest Multiple decision trees that are the basis of a random forest model.

A **Random Forest** is considered an ensemble method that combines outputs from multiple simpler models, in this case it combines outputs from a collection of Decision Trees (see Figure 3). Each tree independently classifies samples using a series of yes/no questions, and the final decision for each observation is based on the most common prediction among the trees (13). Yes/no questions used for decision trees are based on the features, for example, whether an isolate carries a specific gene or whether an isolate is part of a specific cluster. While Random Forests are used for both regression and classification tasks, this tutorial will focus on using Random Forests for classification. Each tree in the Random Forest is created using a random subset of observations (a selection of data points chosen with replacement) and a random subset of features chosen without replacement. For example, if a dataset has 10 features and 100 observations, one of the decision trees in the random forest might pick 89 unique observations and then duplicate some of these observations until it has 100 total observations. It would then randomly pick 9 features out of the 10 available, which the decision tree would use to make its predictions. This process is repeated for each decision tree in the random forest. To make predictions for the test dataset, observations are classified by all decision trees in the random forest. If an observation is classified as resistant by 80 trees and classified as susceptible by 20 trees, then the final decision of the Random Forest will be resistant.

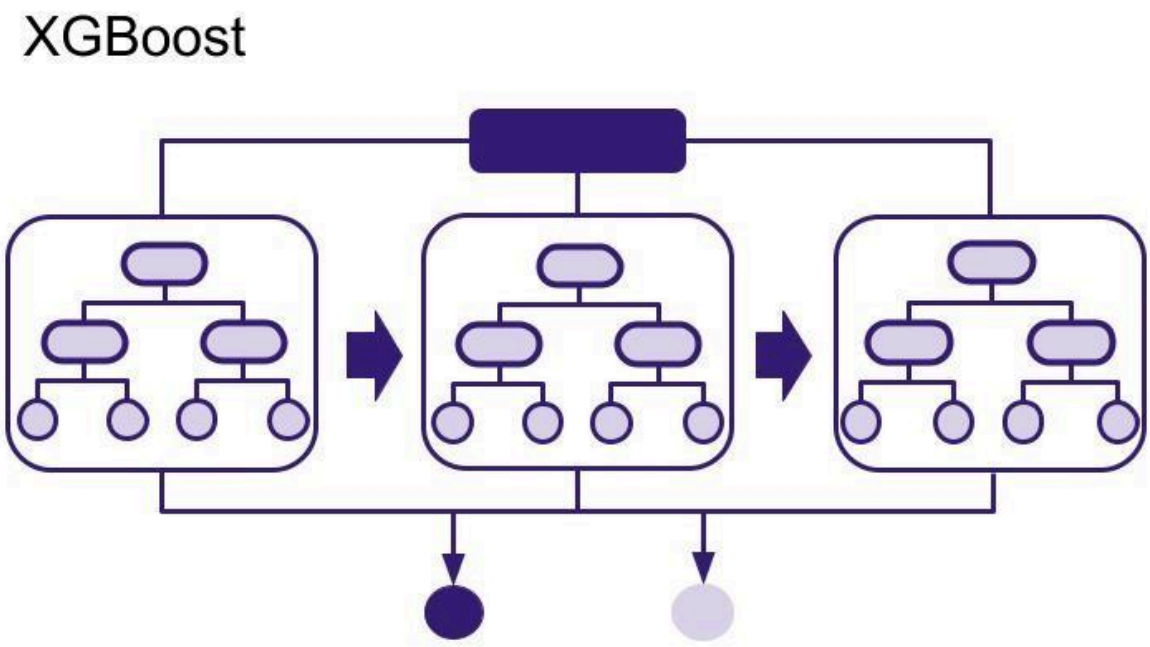

Figure 4. Extreme Gradient Boosted Tree Decision trees created sequentially in Extreme Gradient Boosted Tree (XGBoost)

An **Extreme Gradient Boosted Tree** is another ensemble method that can be used for regression and classification tasks (see Figure 4). Like Random Forests as it also uses several Decision Trees. Each tree also uses a random subset of observations, but this time without replacement. This means that the unique observations picked by a decision tree will not be duplicated. One of the main differences between Random Forests and Extreme Gradient Boosted Trees is that the individual trees here do not make predictions on the target label, instead they attempt to make predictions on pseudo-residuals. Through this method the trees are created sequentially, with each next tree attempting to improve the prior one by reducing the residuals (i.e., the difference between the original classes in our training data and the ones predicted for each iteration of a decision tree) (14). At the end of the training, all trees are used in the final prediction.

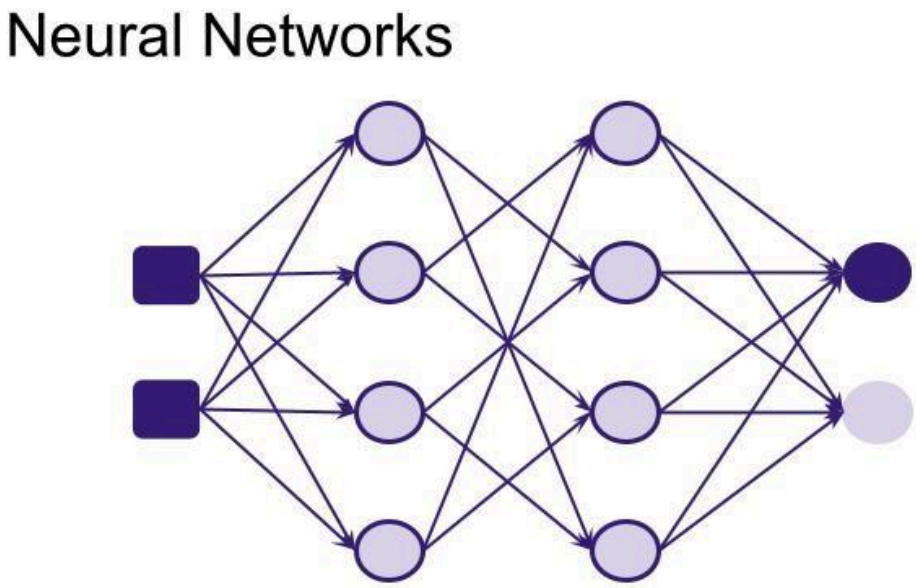

Figure 5. Neural Networks composed of different nodes inspired by neurons.

Deep Learning or Neural Networks, is a method where the algorithm learns to make predictions through a trial-and-error approach as well as an iterative analysis of training samples (see Figure 5). They are loosely inspired by how biological neurons are connected and signal each other. This model serves as the foundation for different types of Neural Network architectures. Different kinds of data can be used to train neural networks, such as images and sounds, etc. Depending on the type of data and task, a specific Neural Network architecture can be used. But in this tutorial, we will learn about how neural networks deal with tabular data (15).

Evaluating model performance

After training the model, It is important to evaluate its performance. There are many evaluation metrics but we will use some of the most common ones used for classification tasks which can be derived from a confusion matrix: accuracy, recall and precision. These metrics are used to quantify how well the model performs. In Figure 6 we illustrate how different metrics are calculated from a confusion matrix using a toy example.

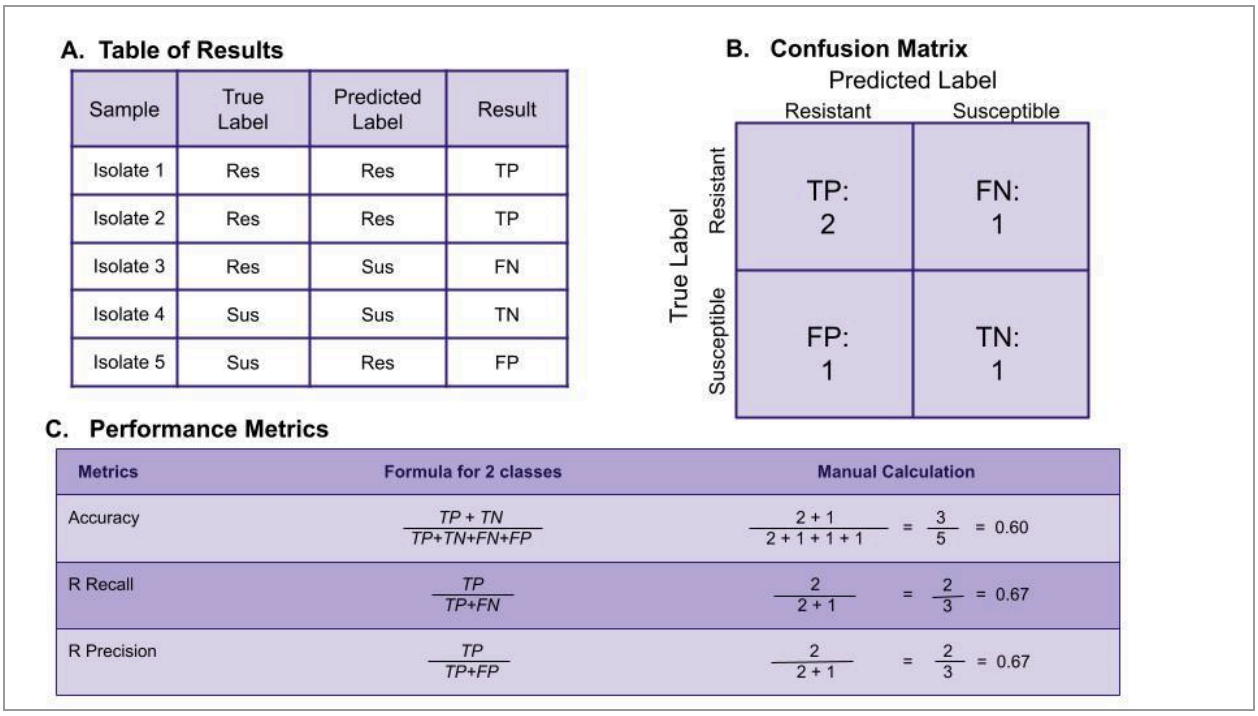

Figure 6. Example of how confusion matrix results are used to calculate evaluation metrics. (A) Determination of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) labels can be seen for toy examples. **(B)** A simple example of a confusion matrix showing total numbers of TP, TN, FP and FN. **(C)** Numbers from the confusion matrix are used to calculate various evaluation metrics.

The most common metrics for classification models are based on the amount of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) predictions. A true positive or true negative outcome is when a model predicts the resistant or susceptible class (respectively) correctly from the test data we gave the model. Conversely, a false positive or false negative outcome is when a model predicts the resistant class incorrectly (e.g. classifying the strain as susceptible when it is actually resistant results in a false negative).
Three of the most common evaluation metrics that can be calculated from these true/false positive/negative outcomes are accuracy, recall, and precision. **Accuracy** is the number of correct classifications over the total number of predictions made. **Recall** is the number of correct classifications made for a particular class over all predictions of that class – for our case, this means the fraction of all resistant isolates that were correctly classified as resistant. **Precision** is the number of true classifications made for a particular class over the total number of predictions for that class – in our case, this means the fraction of correctly labeled resistant isolates divided by all isolates that were predicted to be resistant. The formulas used to calculate these metrics is provided in Figure 6.

Generally, a model that consistently scores close to 1 for accuracy, recall and precision is considered to be well-performing. For the case of resistance, it may be most important to have high recall for resistance (making sure we “catch” all of the resistant strains) – even if the other evaluation metrics are not that high. A resistant *recall* score close to 1 means that the model

was able to correctly identify many resistant strains (high true positive cases), while also having low instances of predicting a strain was susceptible when it was actually resistant (low false negative cases). A resistant *precision* score close to 1 would also correctly identify many resistant strains (high true positive cases), however it differs from recall in that it tells us that the model also has low instances of predicting a strain was resistant when it was actually susceptible (low false positive cases).

Overview of the tutorials

The typical machine learning analysis pipeline involves preparing data, using that data to train a machine learning model, and extracting metrics to visualize how well the model performed. The tutorials provided will go through each step of this pipeline. Code within the tutorials is written in Python, as this is the most commonly used language in machine learning. In order to make the tutorials more accessible, we have provided them in Google Colab format as this allows for the notebooks to be used across any operating system. Furthermore, all of the libraries and tools (listed in Table 3) used in the tutorials are free to access.

Table 3. Python Libraries used in tutorials

Python Libraries	Documentation links
pandas	https://pandas.pydata.org/docs/user_guide/index.html
numpy	https://numpy.org/doc/stable/user/index.html#user
random	https://docs.python.org/3/library/random.html
functools	https://docs.python.org/3/library/functools.html
os	https://docs.python.org/3/library/os.html
sklearn	https://scikit-learn.org/stable/user_guide.html
matplotlib	https://matplotlib.org/stable/users/index
seaborn	https://seaborn.pydata.org/tutorial.html
graphviz	https://graphviz.org/documentation/
xgboost	https://xgboost.readthedocs.io/en/stable/python/index.html
tensorflow	https://www.tensorflow.org/guide
keras	https://keras.io/guides/

Included in this paper are six python notebooks. All notebooks and the data can be found in a GitHub repository. The first one is loading and preparing the data (Data Cleaning). The objective of that notebook is to clean and preprocess the data before feeding it into the machine learning algorithms. In addition, we want to understand the features we will be using to find resistance and susceptibility. We will be looking at how different combinations of these features affect accuracy of the ML models. In the next four notebooks, we will apply the following machine learning algorithms: The Logistic Regression (Logistic Regression), Random Forest (Random Forest), Extreme Gradient Boosted Tree (Extreme Gradient Boosted Tree), and Neural Network (Deep Learning) notebooks explain how the models work and their respective functions are implemented to Moradigaravand's dataset. Finally in the last notebook, we will focus on comparing the different ML models and visualizing the results (Results Visualization).

Once the model has been trained, its performance is tested by having it predict the labels of the test dataset. Then evaluation metrics are calculated to be able to compare the different models. The final part of our tutorial shows the user how to use the Matplotlib and Seaborn libraries to create bar graphs for the accuracy, precision, or recall scores that were collected (see Figure 7).

The results in Figure 7 show that, over all, the extreme gradient-boosted tree model does very well, and has the highest accuracy and recall for many of the drugs. We also see that gene presence-absence data are most likely to be included in the best-scoring models, whereas year of isolation is least likely to be included. Finally, we notice big differences between drugs and between resistance and susceptibility. This may be explained by the fact that for some of the drugs, the number of resistant strains is low, which makes it hard for the models to "learn" to recognize them. For more discussion on these results we refer to the paper that inspired this tutorial (5):

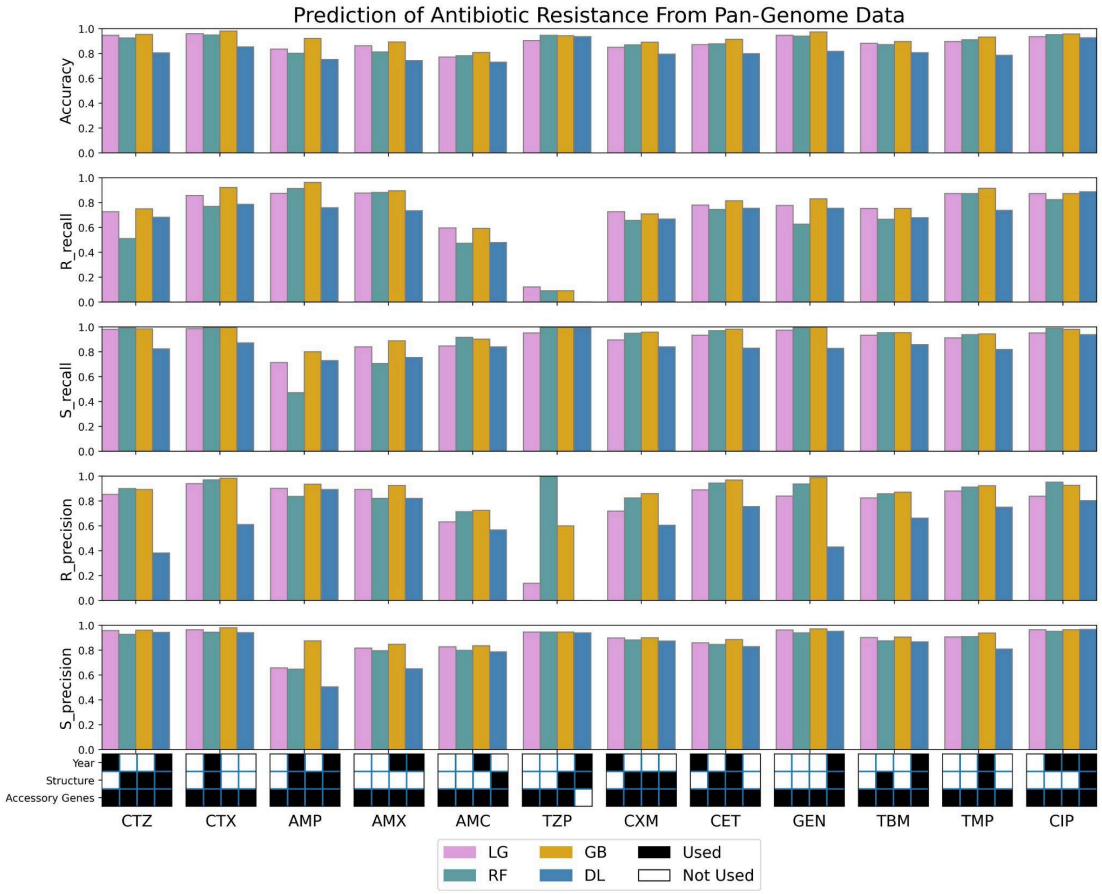

Figure 7. Example of a data visualization result from the final tutorial notebook. The figure shows various bar graph subplots, with each one showing results for an individual evaluation metric. Only the highest-performing combination of features for each metric are shown. The color key refers to the machine learning models (pink for logistic regression, green for random forest, yellow for extreme gradient-boosted trees, and blue for deep learning). The black and white grid on the very bottom indicates which features were used to result in the highest performing model that predicted each antibiotic: ceftazidime (CTZ), cefotaxime (CTX), ampicillin (AMP), amoxicillin (AMX), amoxicillin-clavulanate (AMC), piperacillin-tazobactam (TZP), cefuroxime (CXM), cephalothin (CET), gentamicin (GEN), tobramycin (TBM), trimethoprim (TMP), ciprofloxacin (CIP).

Discussion

The objective of this manuscript and the tutorial was to create a user-friendly guide to the use of machine learning for predicting which isolates antibiotic resistance. We designed 6 notebooks to introduce concepts and methods that are needed in such an analysis. Concepts and methods include data wrangling, machine learning classification models, and data visualization. Pre-processed data were provided within the first notebook to provide a starting point and context for our tutorial. All notebooks contain a step-by-step explanation detailing how we developed a machine learning pipeline to analyze *E. coli* whole genome sequencing data in order to predict resistance. In order for new users to better develop an understanding of machine learning methods and applications, modular code chunks were implemented and commented on in all notebooks. All notebooks for this tutorial are open access and available

for execution in Google Colaboratory. Our tutorial was designed in order to help students and trained biologists bridge the gap between biology and computer science.

We encourage individuals to use the methods learned from our tutorial to perform their own novel investigations. The tutorial provides guidance on constructing baseline models therefore users can improve model performance by performing hyperparameter tuning. Hyperparameter tuning involves systematically testing for the ideal parameter values to boost a model's performance. For example, users may want to find the best tree depth for the ensemble models in our tutorial, or how many layers should our Neural Network contain. There are many parameters within each model that can be tweaked to improve the models we have provided. In addition, different machine learning classifiers can be deployed to predict antibiotic resistance.

Two other models that are commonly used by researchers are the Support Vector Machines (SVM) and K-Nearest Neighbors (KNN). These models function differently than the ensemble models and Neural Network models provided in our tutorials. Similar to how Logistic Regression contains a set threshold to determine resistant bacterial samples, SVM aims to calculate the optimal threshold in order to predict resistance based on the number of features. KNN predicts resistance by grouping samples together that contain similar features.

Our tutorial only scratches the surface of how researchers can analyze resistance in bacteria. In addition to building upon our methods, users can also apply these methods to other bacterial datasets. There are many bacteria across the world that are developing antibiotic resistance that need to be analyzed such as *P. aeruginosa*, *M. Tuberculosis*, *C. auris* and many others. With the advent of improved sequencing technology and analytical capabilities researchers gain greater insights into bacterial resistance mechanisms. Through a better understanding of the genetic components involved in antibiotic resistance, researchers are able to develop better tools to detect resistance in bacteria that are becoming more difficult to treat. We encourage the use of our tutorial to support the improvement of better outcomes for patients.

Acknowledgements

We thank Dr Scott Roy and Dr Danesh Moradigaravand for technical help. Faye Orcales was supported by an NSF REPS supplement (award #1655212, awarded to Pleuni Pennings), Meris Johnson-Hagler was supported by Bristol-Myers Squibb Black Excellence in STEM Scholars and Genentech Foundation Scholars #G-7874540, Jameel Ali was supported by NIH MS to Bridges Doctorate T32-GM142515, John Matthew Suntay was supported by NIH MARC T34-GM008574, and Kristiene Recto was supported by Genentech Foundation Scholars #G-7874540.

Data and code

All notebooks and the data can be found in a GitHub repository.

References

1. Anahtar MN, Yang JH, Kanjilal S. Applications of Machine Learning to the Problem of Antimicrobial Resistance: an Emerging Model for Translational Research. McAdam AJ, editor. J Clin Microbiol. 2021 Jun 18;59(7):e01260-20.
2. Feretzakis G, Loupelis E, Sakagianni A, Kalles D, Lada M, Christopoulos C, et al. Using Machine Learning Algorithms to Predict Antimicrobial Resistance and Assist Empirical Treatment. Stud Health Technol Inform. 2020 Jun 26;272:75–8.
3. Lewin-Epstein O, Baruch S, Hadany L, Stein GY, Obolski U. Predicting Antibiotic Resistance in Hospitalized Patients by Applying Machine Learning to Electronic Medical Records. Clin Infect Dis. 2021 Jun 1;72(11):e848–55.
4. Feretzakis G, Sakagianni A, Loupelis E, Kalles D, Skarmoutsou N, Martsoukou M, et al. Machine Learning for Antibiotic Resistance Prediction: A Prototype Using Off-the-Shelf Techniques and Entry-Level Data to Guide Empiric Antimicrobial Therapy. Healthc Inform Res. 2021 Jul 31;27(3):214–21.
5. Moradigaravand D, Palm M, Farewell A, Mustonen V, Warringer J, Parts L. Prediction of antibiotic resistance in Escherichia coli from large-scale pan-genome data. Darling AE, editor. PLOS Comput Biol. 2018 Dec 14;14(12):e1006258.
6. Ren Y, Chakraborty T, Doijad S, Falgenhauer L, Falgenhauer J, Goesmann A, et al. Prediction of antimicrobial resistance based on whole-genome sequencing and machine learning. Bioinformatics. 2022 Jan 3;38(2):325–34.
7. Khaledi A, Weimann A, Schniederjans M, Asgari E, Kuo T, Oliver A, et al. Predicting antimicrobial resistance in *Pseudomonas aeruginosa* with machine learning‐enabled molecular diagnostics. EMBO Mol Med [Internet]. 2020 Mar 6 [cited 2022 Aug 11];12(3). Available from: https://onlinelibrary.wiley.com/doi/10.15252/emmm.201910264
8. Gaynes R. The Discovery of Penicillin—New Insights After More Than 75 Years of Clinical Use. Emerg Infect Dis. 2017 May;23(5):849–53.
9. Ventola CL. The antibiotic resistance crisis: part 1: causes and threats. P T Peer-Rev J Formul Manag. 2015 Apr;40(4):277–83.
10. Bayot ML, Bragg BN. Antimicrobial Susceptibility Testing. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2023 [cited 2023 Jul 5]. Available from: http://www.ncbi.nlm.nih.gov/books/NBK539714/
11. Jombart T. adegenet: a R package for the multivariate analysis of genetic markers. Bioinforma Oxf Engl. 2008 Jun 1;24(11):1403–5.
12. GeeksforGeeks. Logistic Regression in Machine Learning [Internet]. GeeksforGeeks. 2017 [cited 2023 Jul 20]. Available from: https://www.geeksforgeeks.org/understanding-logistic-regression/
13. Breiman L. Random Forests. Mach Learn. 2001;45(1):5–32.
14. Kattine A. Introduction to Boosted Trees — xgboost 2.0.2 documentation [Internet]. 2022 [cited 2023 Dec 16]. Available from: https://xgboost.readthedocs.io/en/stable/tutorials/model.html
15. Maharani D, Murfi H. Deep Neural Network For Structured Data - A Case Study Of Mortality Rate Prediction Caused By Air Quality. J Phys Conf Ser. 2019 Mar;1192:012010.