Escherichia Coli Ceftriaxone Resistance

D. Bugatti, N. Dupertuis, A. Giacomuzzi, N. Oberholzer
Foundations of Data Science
Department of Health Science and Technology
ETH Zürich

Abstract

STILL NEED TO WORK ON THIS:

1 Overview and Preprocessing

1.1 Introduction to the Dataset

Antibiotic resistance in bacteria is a critical issue in modern medicine. A study conducted by Mihankhah et al. [2017] revealed that an astonishing 87.5% of the bacteria sampled in North Iran were resistant to at least one antibiotic. It is evident that this percentage has likely increased until now, underscoring the urgency of addressing this topic. Another study published in 2022 successfully compiled a dataset containing mass spectrometer results from more than 300'000 samples of bacteria and fungi. This dataset was then used to train a machine learning model capable of predicting whether a bacteria is resistant to a specific type of antibiotic; with the ultimate goal of prescribing the most effective treatment to a patient [Weis et al., 2022]. In this project, we only selected the samples used to train for the resistance to Ceftriaxone of Escherichia Coli, with the aim of building and testing different models onto real-world scenarios.

1.2 Data Wrangling and Visualization

The dataset consists of an integer label, either 1 or 0, indicating resistance to the antibiotic; mass spectrometry results, which are floats ranging from 0 to 1, which have been most likely normalized; and a column named "Unnamed : 0" which has elements of type string. This column contained a serial number ending with either "MALDI_1" or "MALDI_2", suggesting it might relate to the machines used for the observations. To prevent overfitting, we decided to remove this column. Additionally, we checked the dataset for NA and duplicates but found none. The reformatted dataset now consists in 5999 features and one label spanning over 1386 samples.

1.2.1 Avoiding the Curse of Dimensionality

As the dataset contains more features than samples, a feature selection must be performed before fitting with either a filter, wrapper or embedded method. To ensure that some features can be removed without losing too much information, we created a Pearson's correlation matrix of the features and plotted its heatmap with matplotlib [Hunter, 2007] and seaborn [Waskom, 2021].

Report Submitted to
Foundations of Data Science: D-HEST

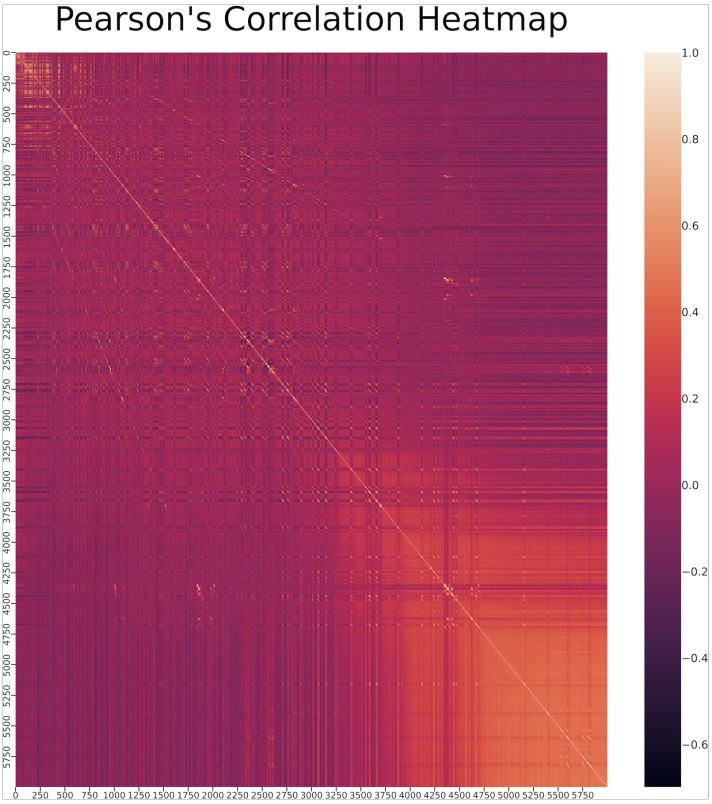

Figure 1: Heatmap of Pearson's correlation matrix

Except the trivial white line that divides the heatmap in half, there are numerous white and dark blue spots, which indicate high positive or negative correlation. Thanks to the graph we can safely assume that most of the variable are correlated, and therefore a filter will not only diminish the probability of overfitting (as mentioned earlier, a higher number of features than samples must always be avoided), but also focus the attention of the model onto non-collinear and highly explanatory features.

1.2.2 Assessing Class Imbalance

Another issue that must be addressed is the class imbalance; 81.5% of the bacteria samples are resistant to Ceftriaxone, and only 18.5% are not. There are several ways to help the model focus more onto the minority class. One solution is undersampling the majority class, but we ruled this out, because our dataset already has a relatively low number of samples. Another solution is oversampling the minority, but we rejected it because the duplicated samples would have identical values, therefore increasing the chance of overfitting which is already a concern with this dataset. The solution we chose as best is balancing the class weight in the model. This method doesn't involve dataset alterations, it instead adjusts the training process by giving more importance to mistakes made on the minority class. Specifically, by multiplying such errors by a weight, usually set inversely proportional to the minority class frequency.

1.2.3 Dataset Split

The dataset has been split into training and testing sets with an 85% ratio for training and a 15% ratio for testing. This leaves 208 samples for testing and 1178 samples for training.

2 Models

The following models have been trained and evaluated on the same dataset split, enabling an equal comparison and evaluation of their performance. They have been developed with Scikit-learn's [Pedregosa et al., 2011], Pandas's [pandas development team, 2020] and numpy's [Harris et al., 2020] algorithms and documentations.

2.1 SVM

Support Vector Machine (SVM) makes his prediction based on hyperplane divisions on the multidimensional feature space. It is therefore extremely important for the success of the model, to select only the highly explanatory features.

2.1.1 Model Workflow Explanation

We created a flowchart which should help visualize the model.

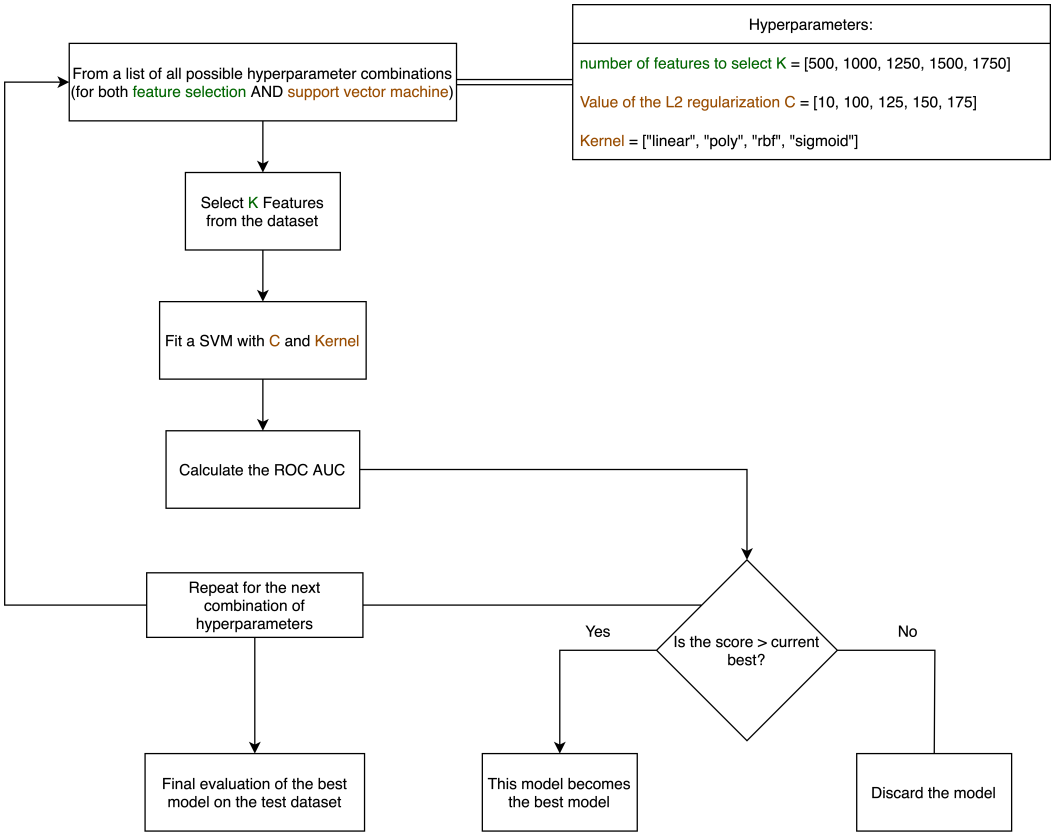

Figure 2: Flowchart of the implemented SVM Model

As SVM is so dependent on the selected features, we decided to implement a wrapper method which unifies feature selection and SVM model tuning into a grid search cross validation environment to iterate the method for every combination of hyperparameters possible for both feature selection and SVM. At first, a number K of features are selected to train the SVM. The non selected features are discarded. The selection algorithm we deemed best is ANOVA, because it selects the features whose variance contributes the most to the variance in the label. Since SVMs require highly explanatory features, this selection method aligns perfectly with their requirements. After the filtering process, a Support Vector Machine (SVM) model is fitted with a combination of two hyperparameters: C, which represents the strength of the penalty applied through L2 regularization, and Kernel, which determines the shape of the division between classes.

Thanks to pipelining, the same grid search cross validation (5 folds) was used to iterate between all of the combinations of the three hyperparameters:

- Feature selection's K: [500, 1000, 1250, 1500, 1750]
- SVM's C: [10, 100, 125, 150, 175]
- SVM's Kernel: ["linear", "poly", "rbf", "sigmoid"]

The best model was then selected using the average from each fold of the ROC AUC score.

2.1.2 Model Evaluation

The model that performed the best during cross validation was an SVM with rbf kernel and C coefficient equal to 125 fitted on 1500 out of the 5999 total features. After retraining the model on the whole training dataset, we evaluated on the test dataset and obtained the following results:

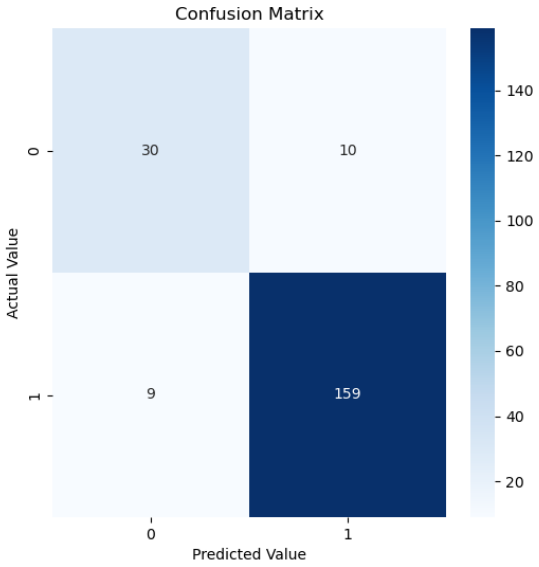

Figure 3: Confusion Matrix of the SVM

The statistics calculated from the confusion matrix are the following:

- Accuracy: 0.9086538461538461
- Precision: 0.9408284023668639
- Recall: 0.9464285714285714
- F1: 0.9436201780415431

Accuracy is the lowest of the scores, and this is explained by looking at the confusion matrix. The model is quite good at predicting positive bacteria samples, but performs quite worse in predicting negative samples. As accuracy gives the same importance to both positive and negative samples, it is comprehensibly the lowest score. The calculated ROC AUC score was instead a 0.9149. Overall, the model performed reasonably well, however there still might be space for improvements on the hyperparameters. Since the filtering process was itself part of the model and caused the model to perform better, it is not only fair, but also right to compare the model with specific feature selection with the other models, which have selected their features in a different way.

2.2 Random Forest

2.3 Logistic Regression

2.4 K Nearest Neighbors

3 Conclusion

References

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Abbas Mihankhah, Rahem Khoshbakht, Mojtaba Raeisi, and Vahideh Raeisi. Prevalence and antibiotic resistance pattern of bacteria isolated from urinary tract infections in northern iran. *J Res Med Sci*, 22:108, 2017. ISSN 1735-1995 (Print); 1735-7136 (Electronic); 1735-1995 (Linking). doi: 10.4103/jrms.JRMS{_}889{_}16.

The pandas development team. pandas-dev/pandas: Pandas. February 2020. doi: 10.5281/zenodo.3509134. URL `https://doi.org/10.5281/zenodo.3509134`.

F. Pedregosa, 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*, 12:2825–2830, 2011.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60): 3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

Caroline Weis, Aline Cuénod, Bastian Rieck, Olivier Dubuis, Susanne Graf, Claudia Lang, Michael Oberle, Maximilian Brackmann, Kirstine K. Søgaard, Michael Osthoff, Karsten Borgwardt, and Adrian Egli. Direct antimicrobial resistance prediction from clinical maldi-tof mass spectra using machine learning. *Nature Medicine*, 28(1):164–174, 2022. doi: 10.1038/s41591-021-01619-9. URL `https://doi.org/10.1038/s41591-021-01619-9`.