Enhancing Antimicrobial Resistance Prediction with Convolutional Neural Networks

Rasmus Freund
Bioinformatics Research Center

AARHUS UNIVERSITY

Supervisor
Palle Villesen

Abstract

As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the architectonic of practical reason. As will easily be shown in the next section, reason would thereby be made to contradict, in view of these considerations, the Ideal of practical reason, yet the manifold depends on the phenomena. Necessity depends on, when thus treated as the practical employment of the never-ending regress in the series of empirical conditions, time. Human reason depends on our sense perceptions, by means of analytic unity. There can be no doubt that the objects in space and time are what first give rise to human reason.

Introduction

Antimicrobial resistance (AMR) presents one of the most daunting challenges in global public health, threatening to render ineffective the very drugs designed to protect us from bacterial infections. The pervasiveness and impact of AMR are profound, as bacteria continue to evolve mechanisms to survive against antibiotics, which have historically revolutionized the management of infectious diseases.

In 2019, an estimated 4.95 million deaths were associated with bacterial AMR, with 1.27 million directly attributable to drug resistance [1]. The predominant pathogens contributing to AMR-related deaths include *Eschericia coli*, *Staphylococcus aureus*, and *Klebsiella pneumoniae*, among others, which are responsible for a significant proportion of the mortality associated with drug-resistant infections. These bacteria are particularly dangerous due to their ability to resist multiple drugs, which complicates treatment options and increases the risk of severe outcomes [1].

Given the critical need for rapid and accurate antimicrobial resistance testing, this thesis explores advanced machine learning approaches to predict AMR directly from MALDI-TOF mass spectra of clinical isolates. Central to this work is the use of the Database of Resistance Information on Antimicrobials and MALDI-TOF Mass Spectra (DRIAMS), a comprehensive and publicly available dataset created by a collaborative effort of researchers from ETH Zürich, the University of Basel, and other institutions [2].

From 2016 to 2018, DRIAMS compiled over 300,000 mass spectra and more than 750,000 antimicrobial resistance phenotypes from clinical isolates collected across four diagnostic laboratories in Switzerland. This extensive dataset encompasses 803 different species of bacterial and fungal pathogens and is organized into four subcollections (DRIAMS-A to DRIAMS-D) [3]. DRIAMS-A, the largest subcollection, serves as the primary focus of this thesis and contains 145,341 mass spectra linked to 71 different antimicrobial drugs.

To address the challenges posed by AMR, this thesis adopts a multi-stage approach, starting with the application of standard machine learning models to predict bacterial species from MALDI-TOF spectra. Building on this foundation, the focus then shifts to predicting AMR using a variety of machine learning techniques. As the complexity of the problem increases, more advanced techniques are employed to better capture the intricate patterns in the data and potentially enhance prediction accuracy and reliability. The following sections will explore these techniques in detail.

Understanding Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) focused on developing algorithms that allow computers to learn from and make predictions or decisions based on data. Unlike traditional programming, where specific instructions are coded by humans, ML systems improve their performance on tasks through experience.

The essence of ML lies in its ability to identify patterns and relationships with large datasets, which might be too complex or subtle for humans to discern. This capability is particularly valuable in fields like bioinformatics, where the volume and complexity of data, such as those found in genomic sequences or mass spectra, require advanced analytical methods.

Basic Concepts of Machine Learning

At its core, ML can be divided into three main types: supervised learning, unsupervised learning, and reinforcement learning.

- **Supervised Learning**: In supervised learning, the algorithm is trained on a labeled dataset, meaning that each training example is paired with an output label. The goal is to learn a mapping from inputs to outputs that can be used to predict the labels of new, unseen examples. Common algorithms include linear regression, decision trees, and neural networks [4].

- **Unsupervised Learning**: Unsupervised learning deals with unlabeled data. The algorithm tries to learn the underlying structure of the data without explicit instructions on what to predict. Techniques like clustering and dimensionality reduction fall under this category. Probabilistic models, such as Gaussian Mixture Models (GMM) and algorithms like k-means and with principal component analysis (PCA), are often used to uncover hidden patterns in the data [5].

- **Reinforcement Learning**: In reinforcement learning, an agent learns to make decisions by performing actions in an environment to maximize cumulative reward. The agent, which can be a software program or a robot, interacts with the environment by taking actions and receiving feedback in the form of rewards or penalties. This feedback helps the agent learn the optimal strategy to achieve its goals over time [6]. While reinforcement learning represents a significant area of ML research, it is not utilized in this thesis. Its mention here serves to provide a comprehensive overview of the main types of machine learning.

Regularized Linear Regression: Ridge and LASSO

Regularized linear regression techniques, such Ridge and LASSO (short for Least Absolute Shrinkage and Selection Operator), address overfitting by introducing a penalty term to the least squares objective function. Overfitting occurs when a model is too complex and captures noise in the training data, leading to poor generalization on unseen data. These methods are particularly useful in high-dimensional scenarios, where they can improve generalization and perform feature selection.

Mathematical Formulation

In standard linear regression, the objective is to minimize the sum of squared residuals:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - X_i\beta)^2 \tag{1}$$

where X is the matrix of input features, y is the vector of target values, and β is the vector of coefficients.

Ridge Regression modifies the objective by adding a regularization term that penalizes large coefficients:

$$\min_{\beta}\left[\sum_{i=1}^{n}(y_i - X_i\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right] \tag{2}$$

Here, $\sum_{j=1}^{p}\beta_j^2$ represents the L2 norm (Euclidean norm) of the coefficients, which is the sum of the squared values of the coefficients. The L2 norm penalizes large coefficients, encouraging them to be small but not necessarily zero.

LASSO Regression, on the other hand, adds a regularization term that penalizes the absolute values of the coefficients:

$$\min_{\beta}\left[\sum_{i=1}^{n}(y_i - X_i\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right] \tag{3}$$

Here, $\sum_{j=1}^{p}|\beta_j|$ represents the L1 norm (Manhattan norm) of the coefficients, which is the sum of the absolute values of the coefficients. The L1 norm can drive some coefficients to exactly zero, performing feature selection.

Properties and Benefits

- **Bias-Variance Tradeoff**: Both Ridge and LASSO introduce bias by shrinking the coefficients, but this can reduce the model's variance and improve generalization performance [7] (see Figures 1 and 2).

- **Handling Multicollinearity (Ridge)**: Ridge Regression mitigates multicollinearity (high level of correlation between several input features) by shrinking coefficients, thus providing more stable estimates [8, 9].

- **Feature Selection (LASSO)**: LASSO can shrink some coefficients to exactly zero, thus performing feature selection and simplifying the model. This is particularly beneficial in high-dimensional datasets [10].

To effectively apply Ridge and LASSO regression, it is crucial to select the appropriate regularization parameter λ - cross-validation is a robust technique for determining an optimal value [7]. This process involves:

1. **Splitting the data**: Divide the dataset into k folds (e.g., $k = 5$ or $k = 10$)
2. **Training and Validation**: Train the model on $k-1$ folds and validate it on the remaining fold. This process is repeated k times, with each fold serving as the validation set once.
3. **Averaging Performance**: Calculate the average performance metric (e.g., mean squared error) across all k folds for different values of $lambda$
4. **Selecting** λ: Choose the λ that minimizes the average validation error, ensuring the model generalizes well to unseen data

Elastic Net Regression

Elastic Net Regression is a regularized regression technique that linearly combines the penalties of Ridge and LASSO. It is particularly useful when dealing with high-dimensional data where multicollinearity is present and when feature selection is necessary.

Mathematical Formulation

Elastic Net modifies the objective of standard linear regression by adding two regularization terms:

$$\min_{\beta} \left[\sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right] \tag{4}$$

Here, λ_1 controls the LASSO penalty (L1 norm) and λ_2 controls the Ridge penalty (L2 norm). When both λ_1 and λ_2 are zero, Elastic Net reduces to OLS regression. By adjusting these parameters, Elastic Net can balance

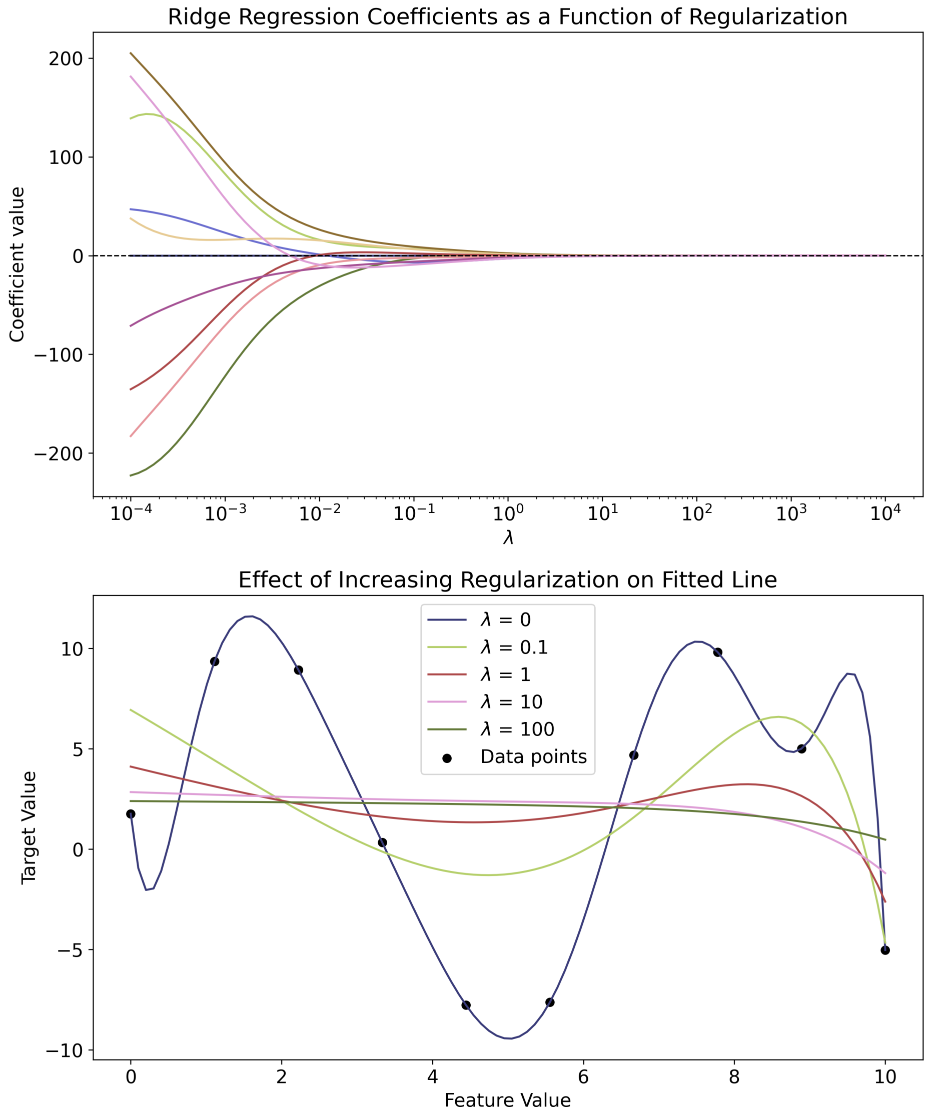

Figure 1: Ridge Regression Coefficients and the Effect of Regularization. The **top plot** shows Ridge Regression coefficients as a function of the regularization parameter λ. Each line represents a different feature's coefficient, demonstrating how increasing λ causes the coefficients to shrink towards zero. The **bottom plot** illustrates the effect of λ on the fitted non-linear model for 10 data points (synthetic data). As λ increases, the model transitions from overfitting (high variance) to better generalization (low variance), as seen by the smoothing of the fitted lines.

between the benefits of Ridge and LASSO, shrinking some coefficients while performing feature selection [11].

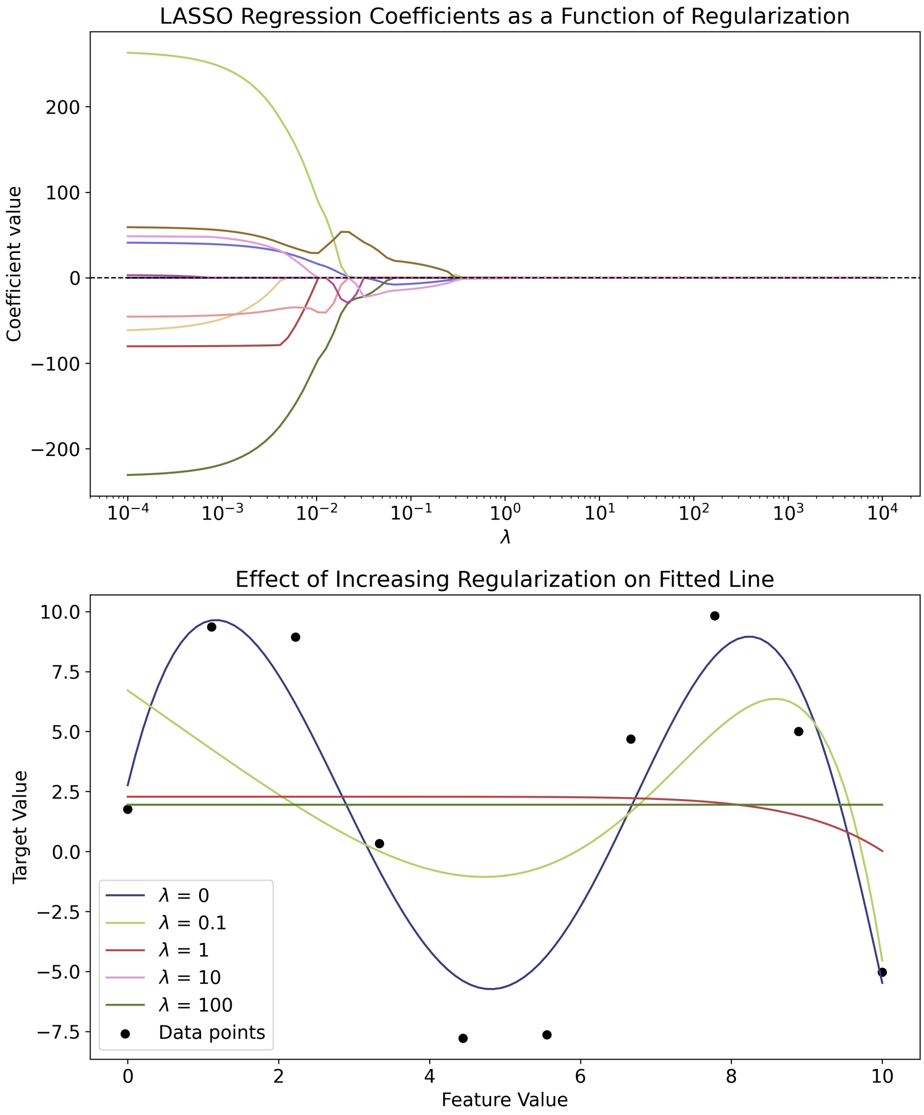

Figure 2: LASSO Regression Coefficients and the Effect of Regularization. The **top plot** shows LASSO Regression coefficients as a function of the regularization parameter λ. Each line represents a different feature's coefficient, demonstrating how increasing λ causes some coefficients to shrink to zero, effectively performing feature selection. The **bottom plot** illustrates the effect of λ on the fitted non-linear model for 10 data points (synthetic data). As λ increases, the model transitions from overfitting (high variance) to better generalization (low variance), as seen by the smoothing of the fitted lines.

Random Forest

Random Forest is an ensemble learning method used for both classification and regression tasks. It operates by constructing multiple decision trees during training, and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees. We will first explore how decision trees are created and move on to explaining the Random Forest method. Note that only the classification case will be explained, as this is the focus of the current work.

Mathematical Formulation - Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a feature, each branch represents a decision rule, and each leaf node represents an outcome. The path from the root to a leaf represents classification rules [12]. An example of a simple decision tree is shown in Figure 3; the Gini impurity value in the tree will be explained in the text below.

In constructing a decision tree, we aim to partition the feature space into regions where the response variable is as homogeneous as possible. This is done by recursively splitting the data based on certain criteria until a stopping condition is met.

Given data that consists of p features and a response, for each of N observations (x_i, y_i) for $i = 1, 2, \ldots, N$, with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, we model the response by partitioning the feature space into M regions $R_1, R_2, \ldots, R_M$ and assigning class c_m to each region. The example in Figure 3 is split as follows:

1. Feature 2 is split at $X_2 = 0.26$ where the feature space at or below this point is assigned to class 0.
2. Feature 1 is now split at $X_1 = 1.459$.
3. Finally, feature 1 is again split, now at $X_1 = -1.043$.

As visualized, this creates four regions in the feature space, which

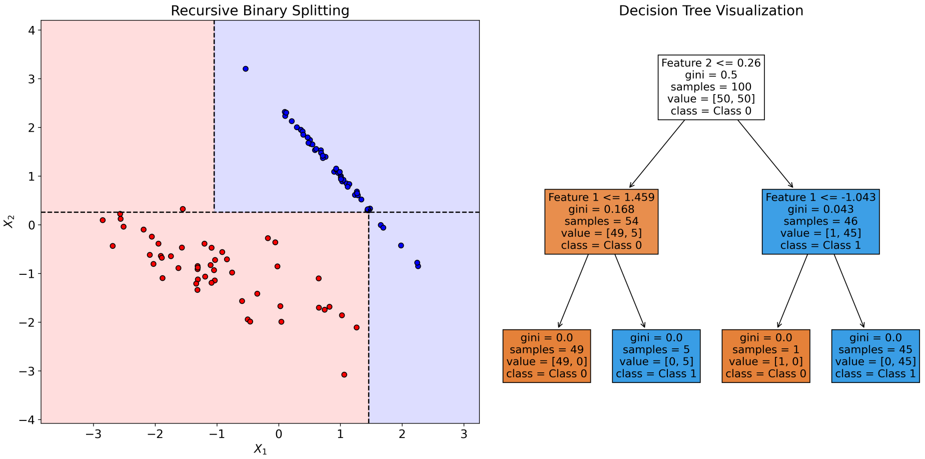

Figure 3: Recursive Binary Splitting and Decision Tree Visualization.
This figure demonstrates the process of recursive binary splitting and the resulting decision tree on data that has two features: X_1 and X_2. The left plot shows the decision boundaries created by a decision tree trained on a synthetic dataset, illustrating how the feature space is recursively split into regions, as indicated by the dashed lines. The right plot visualizes the corresponding decision tree, with each internal node representing a decision rule based on feature thresholds, and leaf nodes representing class outcomes. Nodes are color-coded by the majority class, with class distributions and impurity measures (Gini index) displayed.

allows us to model the response as:

$$\hat{f}(x) = \sum_{m=1}^{4} c_m I(x \in R_m) \tag{5}$$

where c_m is the class assigned to each region, and $I(\cdots)$ is the indicator function, which is 1 if x is in region R_m, and 0 otherwise. Since the data has only two classes, c_m will be either class 0 or class 1 for each region.

But how do we decide where to split along each of the features? This is where we get into the splitting criterion, where there are two popular methods for classification tasks:

- **Gini impurity**:

$$G = \sum_{i=1}^{C} p_i(1 - p_i) \tag{6}$$

where p_i is the probability of class i in the node. This measures how often a randomly chosen element would be incorrectly classified.

- **Entropy**:

$$H = -\sum_{i=1}^{C} p_i \log(p_i) \tag{7}$$

This measures the disorder or randomness of the data points in the node.

Since it is preferable to minimize both the Gini impurity and the entropy, the following step works for either criterion. Consider a feature j and a split point s, and define the regions:

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j > s\} \tag{8}$$

The goal is then to find a j and s that minimize the function:

$$\min_{j,s} \left[\frac{N_{R_1}}{N} C_{R_1} + \frac{N_{R_2}}{N} C_{R_2} \right] \tag{9}$$

where N_{R_1} and N_{R_2} are the number of samples in R_1 and R_2, and C_{R_1} and C_{R_2} is the splitting criterion value of these regions.

After finding the best split, we now assign the class label for each region R_m as the majority class in that region:

$$c_m = \arg\max_k \sum_{x_i \in R_m} I(y_i = k) \tag{10}$$

that is, the class label assigned for a given region is the class that appears most frequently within that region [7, 12].

Utilizing the functions mentioned so far for producing a decision tree could, however, lead to a problematic situation. One way to optimize the tree would be to grow it deep enough to assign each data point to its own node, resulting in an overly complex model. To prevent this, several methods can be employed:

- **Maximum depth**: Directly limit how deep the tree is allowed to grow.

- **Minimum samples per leaf**: Require a minimum number of samples in each leaf to avoid splits that result in nodes with very few data points.
- **Minimum samples per split**: Require a minimum number of samples to be present at a node before it can be split further.
- **Maximum leaf nodes**: Limit the number of leaf nodes in the tree to prevent excessive growth.

Once again, determining the optimal value for these hyperparameters is typically done through cross-validation.

Methods

Results

Discussion

References

[1] Christopher J L Murray et al. "Global Burden of Bacterial Antimicrobial Resistance in 2019: A Systematic Analysis". In: *The Lancet* 399.10325 (Feb. 2022), pp. 629–655. ISSN: 01406736. DOI: 10.1016/S0140-6736(21)02724-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S0140673621027240 (visited on 05/13/2024).

[2] Caroline V. Weis et al. *Database of Resistance Information on Antimicrobials and MALDI-TOF Mass Spectra (DRIAMS)*. Dryad, 2021. DOI: 10.5061/dryad.bzkh1899q. URL: https://datadryad.org/stash/dataset/doi:10.5061/dryad.bzkh1899q.

[3] Caroline Weis et al. "Direct Antimicrobial Resistance Prediction from Clinical MALDI-TOF Mass Spectra Using Machine Learning". In: *Nature Medicine* 28.1 (Jan. 2022), pp. 164–174. ISSN: 1078-8956, 1546-170X. DOI: 10.1038/s41591-021-01619-9. URL: https://www.nature.com/articles/s41591-021-01619-9 (visited on 12/11/2023).

[4] M. I. Jordan and T. M. Mitchell. "Machine Learning: Trends, Perspectives, and Prospects". In: *Science* 349.6245 (July 17, 2015), pp. 255–260. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aaa8415. URL: https://www.science.org/doi/10.1126/science.aaa8415 (visited on 05/15/2024).

[5] Zoubin Ghahramani. "Probabilistic Machine Learning and Artificial Intelligence". In: *Nature* 521.7553 (May 28, 2015), pp. 452–459. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14541. URL: https://www.nature.com/articles/nature14541 (visited on 05/15/2024).

[6] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, Massachusetts: MIT press, 2018. ISBN: 978-0-262-03924-6. URL: http://incompleteideas.net/book/the-book-2nd.html.

[7] Fariha Sohil, Muhammad Umair Sohali, and Javid Shabbir. "An Introduction to Statistical Learning with Applications in R: By Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, $41.98, eISBN: 978-1-4614-7137-7". In: *Statistical Theory and Related Fields* 6.1 (Jan. 2, 2022), pp. 87–87. ISSN: 2475-4269, 2475-4277. DOI: 10.1080/24754269.2021.1980261. URL: https://www.tandfonline.com/doi/full/10.1080/24754269.2021.1980261 (visited on 05/16/2024).

[8] Arthur E Hoerl and Robert W Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: 10.2307/1267351.

[9] Donald W Marquardt and Ronald D Snee. "Ridge Regression in Practice". In: *The American Statistician* 29.1 (1975), pp. 3–20. DOI: 10.1080/00031305.1975.10479105.

[10] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (Jan. 1, 1996), pp. 267–288. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/j.2517-6161.1996.tb02080.x. URL: https://academic.oup.com/jrsssb/article/58/1/267/7027929 (visited on 05/17/2024).

[11] Hui Zou and Trevor Hastie. "Regularization and Variable Selection Via the Elastic Net". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2 (Apr. 1, 2005), pp. 301–320. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/j.1467-9868.2005.00503.x. URL: https://academic.oup.com/jrsssb/article/67/2/301/7109482 (visited on 05/17/2024).

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York, 2009. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. URL: http://link.springer.com/10.1007/978-0-387-84858-7 (visited on 05/15/2024).