Case 1

02582 Computational Data Analysis

s233576 & s226765

March 30, 2025

Introduction and Data Overview

The objective of our first case study is to develop a predictive model for a response variable Y based on a set of features X. This dataset consists of 100 observations, each represented by a 100-dimensional feature vector along with a corresponding response value. Additionally, a separate dataset, X_{new}, containing 1000 new observations was provided, requiring predictions for Y.

Given the high dimensionality of the dataset (100 features and only 100 observations), the curse of dimensionality can pose a significant challenge, potentially leading to overfitting and making it difficult to generalize the model to new data. Thus, we focus on models that can handle such challenges effectively.

Through exploratory data analysis, we observed that the distributions of some of the features exhibit skewness or outliers, suggesting the need for data standardization. We also found high correlations among some of the numerical features, indicating potential redundancy, which could be addressed through dimensionality reduction techniques. Furthermore, we identified a considerable amount of missing values in the dataset that require proper handling.

Therefore, our task involves preprocessing our datasets, exploring various algorithms and selecting an appropriate model for accurate predictions. The full notebook for this analysis can be found here.

Model and methods

Our modeling approach was designed to balance predictive accuracy with computational efficiency while mitigating the risks of overfitting. We explored different algorithms and techniques before moving to a robust model selection and validation process using nested cross-validation.

Considering the high-dimensional nature of the dataset, we selected models that could effectively manage this challenge. While KNN regression is highly sensitive to high-dimensional spaces where the number of features exceeds the number of observations, regularized linear models such as Lasso (solved using Cyclical Coordinate Descent or Least Angle Regression (LARS)), Ridge or

ElasticNet are better suited to mitigate overfitting. Additionally, we tested Decision Tree Regressors, which are capable of capturing complex relationships between features but struggle with high-dimensionality, leading to an inferior performance in this case.

Regarding the nested cross-validation, the outer loop assesses model performance on unseen data, providing a generalized error for each model, which ensures an unbiased estimate of each model's performance. The outer loop calculates the mean root mean squared error (RMSE) of the test folds, which is defined as $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$, where n is the number of observations in the test set and $\hat{y}_i$ is the predicted outcome.

Meanwhile, the inner loop optimizes the models' hyperparameters, with the selected optimal parameters being used for evaluation in the outer loop. The model selection and validation occur in the final inner loop, where the RMSEs for the training and validation folds and the optimal parameters are saved. This process is illustrated in Figure 1. By using this approach, we minimize the risk of overfitting, which enhances the reliability of our model selection process.

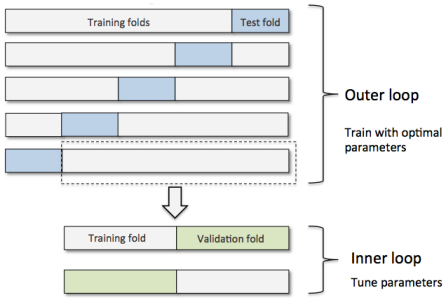

Figure 1: Illustration of Nested CV

It should be noted that preprocessing was performed separately for each fold to prevent data leakage and ensure unbiased model evaluation. This included KNN imputation for handling missing numerical values and standardization, which is crucial for regularized linear models, which is particularly critical for regularized linear models as they are sensitive to the scale of input features. Differences in scale can lead to biased parameters, worsening model accuracy and violating model assumptions. To ensure consistency, the scaling parameters derived from the training data were applied to the test data. By integrating these preprocessing steps into the model pipeline, we ensured consistent transformations across training and test data, improving model reliability.

Dimensionality reduction with PCA

Since each observation consists of 100 features, we investigated the potential benefits of dimensionality reduction using PCA. By performing PCA on the dataset, we found that retaining slightly more than 35 principal components preserved 95% of the total variance, indicating that a significant portion of the information could be maintained in a lower-dimensional space. Given this result, we considered incorporating PCA into our model selection pipeline to reduce feature redundancy and improve computational efficiency. However, PCA did not enhance predictive performance when applied within our modeling framework and was ultimately discarded.

Model selection and validation

The inner loop handled model selection by tuning hyperparameters using *GridSearchCV*. Each model was trained and validated across different folds, selecting the configuration that minimized RMSE validation. This step ensured that models were optimized before final evaluation.

To compare model performance across different hyperparameter settings, we analyzed the mean RMSE for training and validation sets in the final inner loop. The linear regression served as a baseline but clearly overfitted, with a near-zero training RMSE and a validation RMSE of approximately 35. Similarly, the Decision Trees performed poorly across all configurations, with high validation RMSE and significant overfitting. Their ability to capture complex relationships seemed to be outweighed by their poor generalization in high-dimensional data.

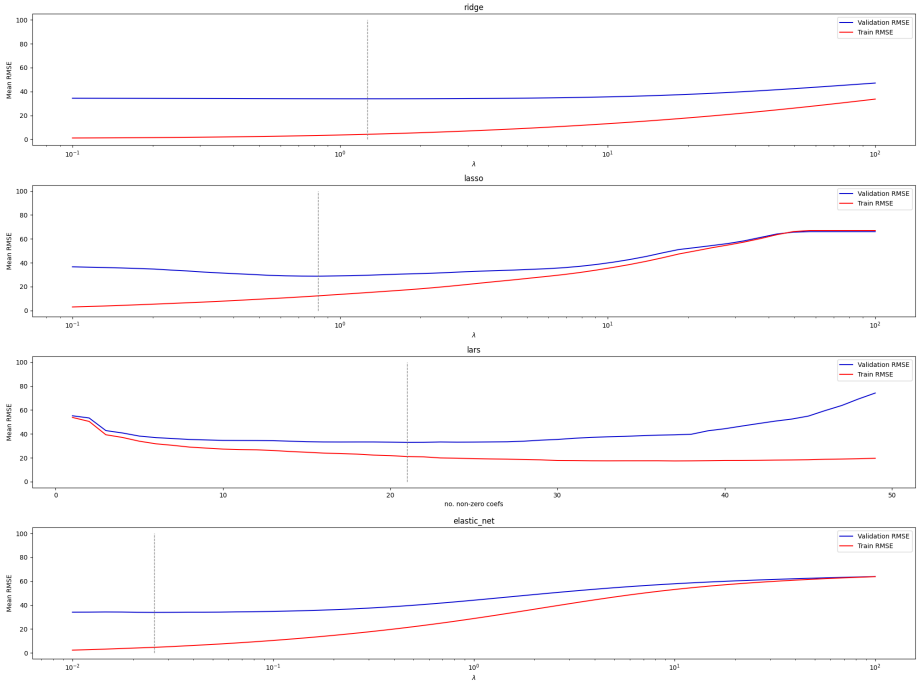

Figure 2: RMSE comparison plot for regularized regressions

To analyze the regularized models, Figure 2 illustrates the mean RMSE trends for training and validation sets, with vertical dashed lines marking the optimal hyperparameters based on validation RMSE. Ridge regression demonstrated more stability, but the gap between training and validation RMSE hints at overfitting. ElasticNet followed a similar pattern, showing higher validation RMSE across most λ values, making it less competitive.

At an optimal number of coefficients, Lasso regression with LARS algorithm exhibited relatively low RMSE and a minimal gap between training and valida-

tion errors. However, Lasso regression with gradient descent emerged as the best model, achieving the lowest validation RMSE while effectively handling high dimensionality through L1 regularization. Lasso regression minimizes the following objective function: $\hat{\beta} = \arg\min_\beta \left(\frac{1}{2n} \sum_{i=1}^{n} \left(y_i - x_i^T \beta \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$, where λ controls the amount of regularization, shrinking irrelevant feature coefficients to zero and thus reducing overfitting while improving generalization. This makes Lasso with gradient descent the optimal choice for this dataset.

To select a final λ for the Lasso regression, we apply the one standard error rule to choose a more regularized model. This rule suggests selecting the simplest model whose error is no more than one standard error above the error of the best model, helping to prevent overfitting while maintaining performance.

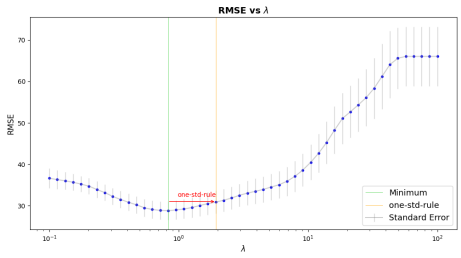

Figure 3: RMSE vs λ for Lasso

In this case, a higher λ value leads to more regularization, making the model simpler by shrinking the coefficients toward zero. This method helps prevent overfitting while maintaining performance by favoring a slightly more regularized model. In this case, the best-performing λ was 0.828, while the one standard error rule suggested 1.93 (Figure 3).

The outer loop results also indicated that Lasso was the best choice, as it consistently achieved the lowest error, demonstrating superior generalization performance on unseen data and no signs of overfitting.

Factor handling

To make categorical variables suitable for predictive modeling, one-hot encoding was used. One-hot encoding creates binary indicator variables for each category. By applying this method, we preserved the categorical structure while preventing misleading assumptions about relationships between different categories. This encoding also inherently handled missing values by creating a separate category for them.

During preprocessing, we identified that one categorical feature, C_{02}, contained only unique values or missing values. Since this feature did not provide any meaningful pattern or contribution to the predictive model, we removed it before proceeding with encoding.

Missing values

Missing data can significantly impact model performance, requiring careful handling to maintain data integrity. In this study, a considerable amount of missing values were present in both numerical and categorical features, requiring distinct approaches. Analyzing the plots obtained using the missingno

Python library, we observed that there did not seem to be any clear pattern in the missing information except for the categorical variables: if one categorical variable was missing, all categorical variables were missing for a particular observation.

Regarding the categorical features, as previously mentioned, one-hot encoding handles missing values. This ensured that missingness was represented in the dataset without introducing distortions while preserving information about the absence of data.

To determine the most suitable method for imputation in the numerical features, we tested multiple imputation techniques, including mean, median, k-nearest neighbors (KNN) and iterative imputation. To assess their effectiveness, the distributions of five randomly chosen numerical variables before and after each imputation method was applied were compared. We observed that mean and median imputation tended to distort the original distribution, often introducing skewness, as expected, considering these methods simply replace missing values with central tendency measures. In contrast, KNN and iterative imputation better preserved the underlying structure of the data, with iterative yielding the most similar distributions. However, this method is computationally expensive so, given the relatively small dataset size, we opted for KNN imputation as a balanced approach that maintains the relationships between variables while being computationally more feasible.

Results

After identifying and tuning our final model (the Lasso regression with gradient descent and a λ of 1.93), we trained it on the entire X dataset to obtain the final model.

It was found that X_{new} also had missing values in all variables, similar to the training set. Its numerical features presented more outliers, which was expected given that this dataset is 10 times larger. However, the distribution of each variable seemed somewhat aligned with the ones from the training dataset. The same preprocessing techniques were used to ensure consistency: categorical features were one-hot encoded, C_{02} was removed, KNN imputation handled missing values in numerical features, and the numerical variables were scaled.

The estimated final RMSE for new observations, obtained from the nested cross-validation's outer loop, was 26.99. This value serves as an approximation of the EPE, helping us quantify the model's predictive accuracy.

Once trained, the final model was used to make predictions on the test dataset X_{new}. The predicted values represent the model's estimate of the target variable for each observation in the new dataset.