02582 Computational Data Analysis - Case 1

Elysia Gao *(s222445)*, Christian Bakos Leirvåg *(*s222681)

March 20, 2023

INTRODUCTION & DATA DESCRIPTION

We evaluated the dataset consisting of 100 observations (Y, x) of response Y (a vector) and features X (a 100-dimensional feature matrix), where 95 features are continuous and 5 features are categorical. In addition, we have 1000 new observations (x_{new}) for which we want to predict y_{new} and estimate the prediction error using root mean squared error (RM$\hat{\text{S}}$E). To accomplish this, we built a predictive regression model of Y on x using Python, Python packages, and R. We outline the procedure which includes addressing missing values and handling categorical data by use of a pre-processing pipeline in Python, using R to further analyze and visualize our data and determine if our data is most likely linear or non-linear, building a robust method for Model Selection by use of nested cross-validation, Model Validation through Out of Bag Bootstrapping, and finally prediction of the X_{new} and estimation of the prediction error.

MISSING VALUES

We evaluated the missingness patterns in the dataset by visualizing the missing data. Since no discernible patterns were present, we assumed that missing data was missing at random (MAR) or missing completely at random (MCAR) and used simple imputation with mean and mode to fill in missing values. See Figure 1 and 2 that visually depict the proportion of missing values where red squares indicate missing values and blue squares indicate present values. Mean was used for imputation of continuous data, assuming normally distributed data, while mode was used for imputation of categorical data. This approach is commonly used when missingness is random and helps to avoid introducing bias in the analysis. For our analysis, we used SimpleImputer, which performs simple imputation with mean and mode separately for continuous and categorical variables which is defined in the pre-processing pipeline used during model selection.

In our case, the dataset contains 14.7% missing data, and the use of simple imputation is deemed adequate given the low to moderate percentage of missing data.

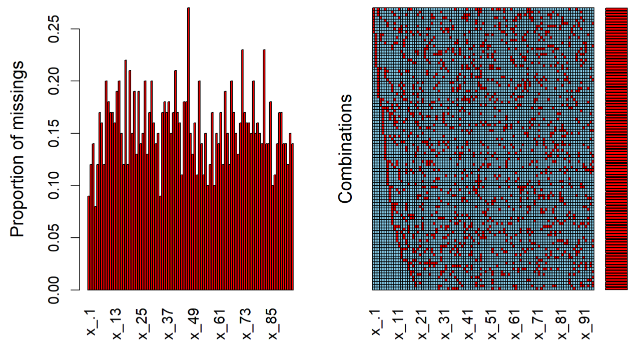

Figure 1: The first plot shows the proportion of missing data for each feature, while the second plot visualizes the combinations of missing data. Red squares indicate missing values in the 100 x 95 grid for continuous data. Blue squares represent present values.

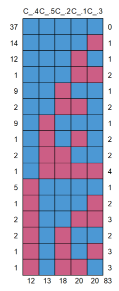

Figure 2: Plot of Missing Categorical Values. Red squares indicate missing values where blue squares represent present values in the 100 x 5 grid.

In addition to using a simple imputation approach, another option would be to use a model that incorporates Bayesian methods or robustness to missing data. Bayesian models allow for the quantification of uncertainty in both the parameters and the imputed values and can be a powerful tool for dealing with missing data. Similarly, robust methods can be used to handle missing data when the assumptions of simple imputation are not met. Thus, during model selection, we include the Bayesian Ridge Regression model as one of the models tested.

FACTOR HANDLING

We chose to use one-hot encoding to handle categorical variables because it allows the model to interpret the categorical variables as a set of binary variables rather than ordered values. With one-hot encoding, a new column is created for each category, where the value is 1 if the observation matches with the category or 0 otherwise. This ensures that the regression model properly handles the categorical variables and accurately interprets their relationship with the target variable. In our case, with 100 original features and 17 additional features from one-hot encoding, one-hot encoding was necessary for accurate interpretation of our categorical variables in our regression model. We incorporated one-hot encoding in our pre-processing pipeline used during model selection (further discussed in Model Selection).

MODEL AND METHOD

Checking linearity

In our analysis, we focused on linear regression models after confirming that the normality, independence, homoscedasticity, and linearity assumptions were satisfied for general linear models [Madsen and Thyregod, 2011]. The histogram of y showed a normal distribution, and the q-q plot of y demonstrated linearity, confirming normality for the dependent variables. See Figure 3 and 4 below as reference.

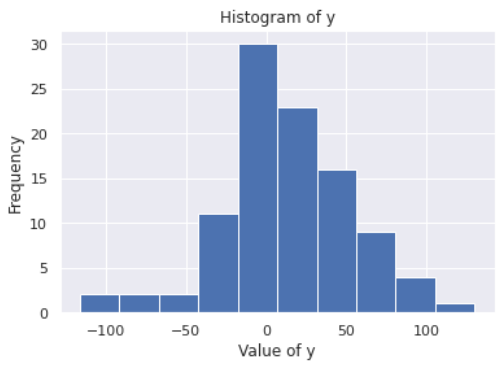

Figure 3: Histogram of Dependent Variable, y

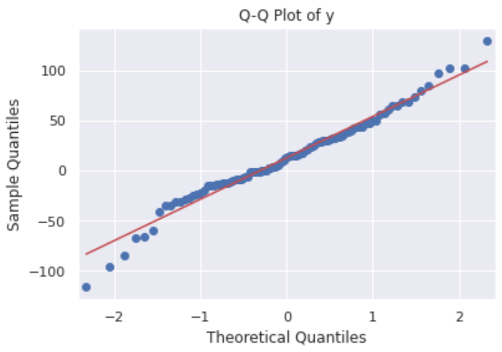

Figure 4: Q-Q Plot of Dependent Variable,y

Furthermore, plots of a linear model with all variables in R showed randomness in the residual vs. fitted plot and scale-location plot while the q-q residual plot was near linear, supporting the suitability of a general linear model.

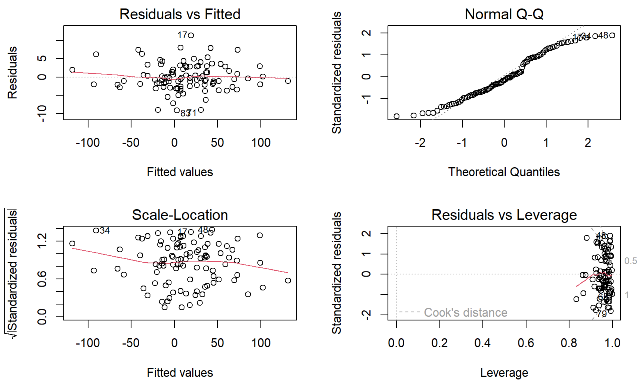

Figure 5: Residual Plots for the General Linear Model `model <- lm(y ~ ., data = X)`

Normality, independence, homoscedasticity, and linearity are crucial assumptions for linear models. Normality of the dependent variable ensures that the residuals of the model are also normally distributed. Independence of the residuals ensures that they are not correlated with each other, which could lead to incorrect estimates of the model parameters. Homoscedasticity ensures that the variance of the residuals is constant across all values of the dependent variable, and linearity ensures that the relationship between the dependent variable and the independent variables is linear. These assumptions allow us to make valid inferences and predictions using the linear model [Møller, 2023].

Thus, the residual plots further confirmed that the assumptions of normality, independence, and homoscedasticity were satisfied. The residual vs. fitted plot showed randomness in the residuals, satisfying the assumption of independence (residuals indepedent of one another) and also linearity. The scale-location plot demonstrated constant variance of residuals across all values of the dependent variable, fulfilling the homoscedasticity (equal variance) assumption. The q-q residual plot was nearly linear, confirming the normality assumption for the residuals.

Therefore, we chose to investigate the linear regression models in our model selection process (Linear Regression, Ridge Regression, Lasso Regression, Elastic Net Regression, and Bayesian Ridge Regression). As a sanity check, we also tested non-linear regression models (Decision Trees, Random Forests, KNN) but found results to generally be worse.

MODEL SELECTION

We used a nested cross-validation approach for model selection to ensure robustness while avoiding the risk of overfitting to the validation data (that can occuring during a single split). The outer loop is used to evaluate the performance of each model using test data that is not seen during training, and an inner loop is used to optimize the hyperparameters of each model using validation data that is not seen during training or testing.

We began our model selection process by defining the number of iterations for the cross-validation procedure to be 5. We used the number of splits for the inner cross-validation, which was set to 5, and defined the inner cross-validation itself using KFold with shuffling.

To pre-process the data, we used a pipeline that involved imputation, one-hot encoding, scaling, and PCA (principal component analysis). We used the SimpleImputer to fill in missing values, applied StandardScaler to normalize the data, applied OneHotEncoder to one hot encode categorical data, and finally applied PCA to perform dimension reduction, since there were more features (117) than observations (100) and also as a tool to avoid overfitting. PCA also reduced multicollinearity which was necessary to do for fitting linear models. See below Figure that depicts the correlation matrix between the features in X. Note that multicollinearity exists as shown in Figure 6 and thus, PCA is necessary for multicollinearity and dimensionality reduction. The number of PCA components was set to the cut-off of 90% explained variance.

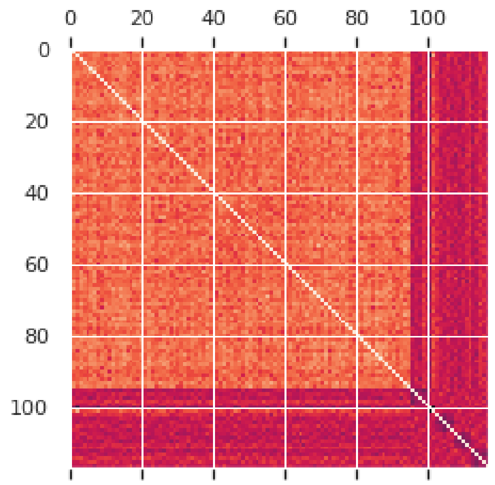

Figure 6: Heatmap depicting correlation between features in X (after imputation and one hot encoding)

To ensure that our model selection process was reliable, we took care to ensure that observations were independent [Clemmensen, 2023]. We performed the specific procedure highlighting the preprocessing procedure in order to avoid data leakage in the cross-validation loop. Specifically, we randomly permuted the data and split intro training, validation, and test sets for each of M iterations (outer loop). We fit the imputer, one-hot encoder, scaler, and PCA transformers on the training set for each fold of the inner loop and applied them to the validation set. We trained the model on the training fold set, and evaluated the performance on the validation fold set for each hyperparameter combination in the grid. We selected the hyperparameters that gave the best performance on the validation fold set. Next, we fit the imputer, scaler, one-hot encoder, and PCA transformers on the outer loop training and validation sets and applied them to the test set. Lastly, we evaluated the performance of each model on the test set. Overall, this approach ensured that transformers are fit and applied separately on each fold of the cross-validation loop in order to prevent data leakage and produce reliable estimates of model performance.

We also followed a method to ensure that our model selection process was robust. The method (which is outlined in pseudocode below [Clemmensen, 2023]) involved randomizing the data by permuting it, and then splitting it into three sets: training, validation, and testing. We trained the model on a range of tuning parameters using the training data, selected the best model based on the validation data, and then estimated the error associated with the model using the test set.

Note that we used the same random state m for splitting the data intro training/validation/test sets as well as for the inner cross-validation folds for each model. This ensures that the splits are consistent across all the models and that each model is evaluated on the same data splits, making the results between models directly comparable.

Algorithm 1 Nested Cross-Validation for Model Selection

```
1:  for model in models list do
2:      for $m = 1$ to $M$ do
3:          Randomize the data by permuting it.
4:          Split the data into three sets: training, validation, and
            test.
5:          for Each train, validation index in the inner CV split do
6:              Pre-process the training data using the pre-processing
                pipeline.
7:              Perform grid search over hyperparameters and calcu-
                late EPE for each fold.
8:              Fit the model on the pre-processed training data.
9:              Evaluate the model on the validation data.
10:             Select the best model based on its performance on the
                validation set.
11:         end for
12:         Test the selected model on the test set to estimate its
            error.
13:     end for
14:     Calculate the mean and standard deviation of the test errors
        over all $M$ iterations.
15: end for
16: Pick final best model.
```

To perform hyperparameter tuning, we used RandomizedSearchCV, which randomly searched over a range of hyperparameters. The evaluation metric used to calculate the error was negative mean squared error. RandomizedSearchCV was preferred over GridSearchCV since it is more computationally efficient and can lead to better models by exploring a wider range of hyperparameters. The best model was selected based on the smallest absolute value of the mean EPE calculated over the validation folds.

To ensure that the splits were critical, our code also implemented permutation of the data before splitting and normalization of the training data separately for each fold. Additionally, we used a five-fold inner cross-validation to further avoid overfitting.

Finally, our code calculated the mean and standard deviation of the test errors over the M iterations, providing an estimate of the expected performance for each model. We iterate this process over each model (Linear Regression, Lasso Regression, Elastic Net Regression, Ridge Regression, and Bayesian Ridge Regression, KNN Regression, Decision Trees Regression, Random Forests Regression) and determine the best final model as that with the lowest mean EPE (RMSE). The final parameter selection was based on the median of the 5 sets of best hyperparameters. Lastly, we trained our final best model on the full dataset using the best hyperparameters to be used for the prediction y_{new} given X_{new}. This process ensured that the model selection and evaluation were performed in a rigorous and unbiased manner, resulting in the identification of the best model for our analysis that performed well on independent test data.

The results from the model selection procedure are as follows:

Model Name	*Mean Test Error*	*Std Test Error*
Ridge	36.5697	3.76353
Lasso	29.176	3.01712
Elastic Net	31.8275	5.73302
Bayesian Ridge	**26.781**	**3.30676**
Linear	29.8976	3.73835
KNN Reg	40.0424	8.35872
Decision Trees Reg	47.6924	10.4751
Random Forest Reg	37.57	6.8691

Table 1: Mean and Standard Deviation of Test Errors over M iterations for each tested Regression Model

As shown in Table 1, the linear and regularized linear models performed the best as expected in terms of having a lower mean EPE (from our analysis of data visualization and checking of assumptions of a general linear model). Listed below are also the results from the Model Selection procedure. We calculated training, validation, and test error% to potentially see any issues with over or underfitting.

Model Name	*Train Error %*	*Val Error %*	Test Error %
Ridge	1.841%	2.670%	1.540%
Lasso	1.868%	2.555%	3.567%
Elastic Net	1.722%	2.351%	3.053%
Bayesian Ridge	**1.689%**	**2.551%**	**2.873%**
Linear	1.808%	2.665%	2.697%
KNN	1.689%	2.741%	2.234%
Decision Trees	1.598%	2.700%	3.558%
Random Forest	1.700%	2.918%	3.857%

Table 2: Training, Validation, and Test Error % for Each Tested Model

MODEL VALIDATION

We used bootstrapping to perform model validation, which is a method of model averaging. In this method, we randomly drew data sets with replacement from the training data. We performed this B times ($B = 100$ in our case) to produce B bootstrap data sets. Then, we refitted the model to each of the bootstrap data sets and examined the behavior of the fits over B replications. This process is used to overcome the issue of overfitting, which occurs when a model fits the training data too closely, resulting in poor performance on new or unseen data [Clemmensen, 2023].

To estimate the performance of the model on unseen data, we used out-of-bag samples. These are the data points that are not included in the bootstrap sample for a particular replication. When measuring the error at (x_i, y_i), we selected only bootstrap replications that do not contain (x_i, y_i). This approach provides a better estimate of the model's performance on unseen data. We used the mean squared error (MSE) as a metric to estimate the expected prediction error (EPE).

The mean out-of-bag boostrapping error (RMSE) is 24.830. This result is close to our model selection RMSE and lower, which is a good indication that our model is performing well on new and unseen data. The distribution of out-of-bag bootstrapping errors is also normally distributed, which is also a good indication that our model is not overfitting the training data and is generalizing well to new data. See Figure 7 for reference.

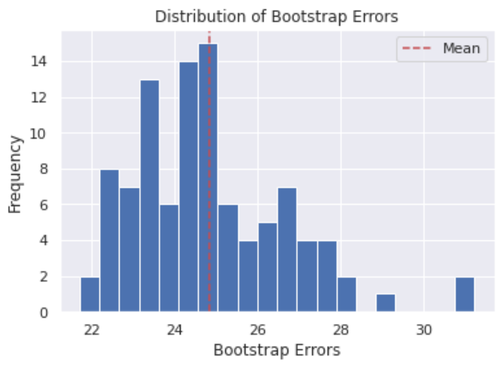

Figure 7: Histogram of Out-of-bag Bootstrapping Sample Errors

Estimated EPE

We estimated the root mean squared error (RMSE) for our Bayesian Ridge Regression model when predicting y_{new} values, without access to the true y_{new} values. To achieve this, we utilized a 5-fold cross-validation approach to evaluate the performance of our model on the available data.

We used cross-validation as it is widely-used for assessing the generalization capability of a model on unseen data.

The following steps were performed to estimate the RMSE using 5-fold cross-validation. We pre-processed the entire dataset X by applying the same pre-processing pipeline. This involved imputing missing values, one-hot encoding categorical features, scaling the data, and applying principal component analysis (PCA) to reduce dimensionality. We then performed 5-fold cross-validation on the Bayesian Ridge Regression model using the preprocessed dataset. We used the negative root mean squared error scoring metric to obtain the RMSE for each fold. Note that the cross validation score function returns the negative RMSE values by convention, so we negated the results to obtain the actual RMSE values. Lastly, we calculated the average RMSE and its standard deviation across the five folds to provide an estimate of the model's performance on unseen data.

The estimated RMSE, obtained using 5-fold cross-validation, provides an approximation of the model's performance on new data, such as y_{new}. While this method does not directly compute the RMSE for $y_{newpred}$, it offers a reliable estimate of the model's generalization capability. We chose this approach because it allows us to assess the model's performance in the absence of true y_{new} values. Once the true y_{new} values become available, the actual RMSE for $y_{newpred}$ can be calculated directly.

RESULTS

The best model for our prediction was Bayesian Ridge Regression with the hyperparameters $\alpha_1 = 0.0001, \alpha_2 = 1 \times 10^{-5}, \lambda_1 = 0.001, \lambda_2 = 1 \times 10^{-5}$ over 36 PCA components of the 117 one-hot encoded features which had the lowest mean test error out of all tested models as shown in the Tables 1 and 2 in Model Selection. Although other linear and regularized models performed somewhat similarly, we additionally chose this model to further add flexibility and robustness to our predictions as Bayesian Ridge Regression takes into uncertainty in coefficients which is helpful for our missing data. Lastly, our predicted RMSE was 27.310.

Testing Phase

Notably, we had also made preliminary tests with several other options to lower mean RMSE during our Model Selection phase including testing our regression models (neural network based regressor, adaboost regressor etc.), running model selection without PCA transformation, changing the number of CV folds and M iterations, using a different method of hyperparamter tuning (tried gridsearchCV as well), nested cross validation with two K-Fold loops, an ensemble of linear and regularized linear models, using boost methods on top of our base estimators. With preliminary results, we did not see a big reduction in mean RMSE, therefore, we chose Bayesian Ridge Regression as stated for simplicity but further analysis regarding any of the aforementioned methods may lead to better results.

References

[1] Line Clemmensen. Model selection (computational data analysis) slides. pages 26–41, 2023.

[2] Henrik Madsen and Poul Thyregod. General linear models. page 87. Chapman Hall/CRC, 2011.

[3] Jan Kloppenborg Møller. General linear models (advanced data analysis) slides. 2023.