Case 1

02582 Computational Data Analysis

s243636, s243933

March 27, 2025

Introduction/Data description

With this project, we aim to develop a predictive model for the response variable Y using a given feature matrix X. The dataset provided includes 100 observations, each with a response variable Y and a corresponding set of 100 features. Furthermore, there is also a separate dataset that contains 1,000 new observations (named x_{new}), which is intended to make predictions only. The goal of this case project is to then create an accurate predictive model and estimate its prediction error using Root Mean Squared Error (RMSE).

Overview of the data

- Training data (*case1Data.csv*): Contains 100 observations, each consisting of a response variable Y and a 100-dimensional feature matrix X.
- Test data (*case1Data_Xnew.csv*): Includes 1,000 new observations of X, for which predictions of Y need to be computed.

Note that all code and figures are in our GitHub repository.

Model and method

As Y is continuous, we use only regression models to make predictions. The 7 different models tested within the scope of this project are linear regression, ridge regression, lasso regression, elastic net, support vector regression, multi-layer perceptron regressor, and random forest regressor.

Model selection

In order to find the optimal model, we performed 2-layer k-fold cross-validation (CV) with $k = 5$. The inner layer of the cross-validation is used to estimate the best hyperparameters for a model using predefined hyperparameter grids. Using the best found hyperparameters for each model, we retrain the model using all of the training data from the outer fold before calculating the RMSE. These RMSE values along with the model and their hyperparameters are stored for later use. Once we run through all outer folds, the RMSE values are averaged for all 5 folds independently for each model giving us an initial estimate for the expected prediction error ($R\hat{M}SE$) for each model.

As we have more predictors than we do observations, we deemed it might be useful to perform dimensionality reduction in the form of PCA. The PCA within our implementation is set to use all of the components that would explain 90% of the variance within the data. The $R\hat{M}SE$ values from the process in the previous paragraph are computed for both PCA data and as well as regular preprocessed data. The results from this are shown in Table 1.

Model	$R\hat{M}SE$ with PCA	$R\hat{M}SE$ without PCA
Linear Regression	44.477	39.621
Lasso Regression	44.060	34.313
Ridge Regression	42.266	35.412
ElasticNet	41.924	36.187
MLP Regressor	48.440	42.422
Random Forest Regressor	68.660	47.322
Support Vector Regressor	45.109	37.766

Table 1: The models and their estimated prediction errors when trained using PCA components and using regular preprocessed training data.

Ideally, $R\hat{M}SE$ should be as low as possible. The best model therefore is Lasso Regression without any PCA with an $R\hat{M}SE$ of 34.313. In our case, as $p > n$, it was obvious models with some sort of regularization would, and do, perform better. As it can be seen from Table 1, lasso regression, ridge regression, and elastic net, all that have some sort of regularization, perform better than the other models. It is also interesting to observe that all of the models have lower $R\hat{M}SE$ values without PCA than they do with PCA.

Although we are now aware of the best model, in order to find the best hyperparameters for the Lasso model, we look through the RMSE values we obtained for each fold for the model during the 2-layer CV process. These results are displayed in Table 2.

Outer Fold	RMSE	Hyperparameters
1	29.876	{“alpha”: 1}
2	38.947	{“alpha”: 1}
3	29.177	{“alpha”: 1}
4	37.232	{“alpha”: 5}
5	36.335	{“alpha”: 1}

Table 2: The RMSEs and respective hyperparameters for the best model, i.e., Lasso Regression.

As the RMSE between the lasso models is smallest when the hyperparameter is {“alpha”: 1}, our final best model is Lasso(“alpha”=1). Alpha is the hyperparameter that multiplies the L1-norm within the model. This regularization helps prevent overfitting by lowering the coefficients of the less significant parameters towards to zero. The larger the alpha value, the stronger the regularization.

Missing values

As it can clearly be seen in Figure 1 and 2, our dataset included a vast amount of missing values. Specifically in the overview of Figure 2, it is visible that the missing values of the numerical features are randomly distributed across different rows and columns. Therefore, due to this randomness, we can impute the data with either the mean or the median of the variables, instead of simply deleting entire rows, without potentially losing valuable information. With that in mind, we then implemented specific transformations to handle these missing numerical values effectively.

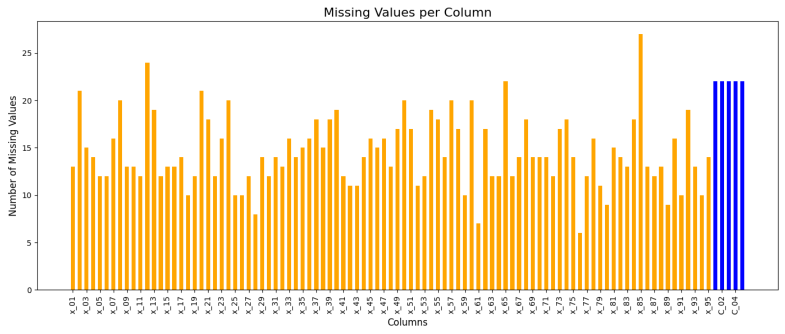

Figure 1: Missing values of X train per column

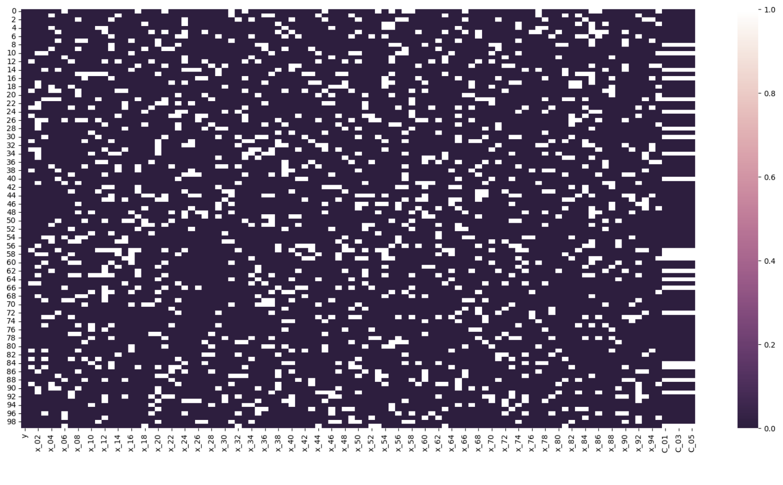

Figure 2: Overview of missing values of X train

For the numerical missing values, we started by identifying numerical columns - which started with "x" - for X train as well as for the test dataset. We then performed a skewness test, using the `skewtest` function, to see if the numerical features follow a normal distribution. If the feature at hand was normally distributed then its missing values were imputed with the mean, otherwise if skewed, the missing values were assigned the median to prevent further distortions. These transformations follow the preferred techniques mentioned in Section 9.6 of the book.

For the categorical variables, we chose different imputation strategies due to the contrasting nature of the features. As further mentioned in the next report section, missing values were handled by introducing a separate category to account for omitted information without introducing bias. This ensured that categorical features could later be encoded effectively, as detailed in Subsection 9.2.4 of the book.

Factor handling

In order to properly incorporate categorical variables into our predictive models, we started by identifying categorical features within the dataset - columns beginning with "C" - and analyzed their distributions.

A key insight from our exploratory analysis was that missing values in these features were not randomly distributed as numerical values were; rather, they tended to occur in the same observations as it can be seen in Figure 2. Therefore, and in line with the recommendations in Subsection 9.2.4 titled "Missing Predictor Values" of the book, we handled missing categorical values by assigning them to a new category. This ensured that these gaps were explicitly recognized rather than being treated as undefined or ignored. Based on this, we imputed the missing data with all 0s so that the encoding of these categorical features for the required numerical inputs of the prediction models would be easier.

Once the imputations of the missing values were addressed, we proceeded with encoding the categorical variables. We applied One Hot Encoding, which creates a new column for each category in each categorical variable. The values within these new columns are 1s if the category applies to the row, else it is 0. By converting categorical variables into numerical format, while preserving their own structure, we made sure that they could be effectively used in our predictive modeling pipeline.

Model validation

Now that model selection helped find the best model, the preprocessed training data is split into training and validation datasets. A newly instantiated Lasso("alpha"=1.0) model is trained on the split training set and a final estimate for $R\hat{M}SE$ is calculated on the validation set. We expect the $RMSE$ from the test set to be close to $R\hat{M}SE = 35.666$.

Results

As mentioned in the previous section, the estimate of our EPE, given as an estimate of the $RMSE$, is 35.666. However, we instantiate the best model yet again and train the Lasso model with the same hyperparameter with all of the preprocessed training data before making the predictions to ensure that all of the training data is being well utilized. The first 10 values of these predictions are shown in Table 3.

Predictions
174.69627148393525
270.8949992384187
267.08077788076315
206.58273172169345
128.7981305811395
72.37684715360487
168.47162537304837
186.36730023307004
231.89830397626213
275.7004541280792

Table 3: First 10 instances of our model predictions.