Danmarks Tekniske Universitet

Case 1 - Computational Data Analysis

Sunday 21st March

Authors:
Nicolaj Hans Nielsen
Jonah Tabbal

Study No:
s184335
s210158

1 Introduction

In this report, we will investigate how to construct a model that suits a given data sets best. We are to find this model on a data set consists of 100-dimensional feature matrix and one response variable and we are given 100 observations. The data consist of 95 numerical features and 5 categorical features and in the categorical features, we encounter missing data. We will describe the models tested, handling of missing values, features selection, model validation with a method on how to estimate the predictive performance of the model using $\widehat{RMSE}$. In the end, we will import a new data set with 1000 observations and make predictions on that.

2 Model and method

In this section, we will briefly describe the models considered, the model selection, model validation, handling of missing data, and handling of factors in the features.

2.1 Models

Initially, we studied multiple model; Elastic net, gradient boosting, XGboosing, regression trees with, and without bagging, however, it turned out that Lasso and random forest were the best. These will be described in detail below.
XGBoost is an optimized version of gradient boosting which generally performs better and we had really good preliminary results. However, as the model is outside the scope of the models considered in this course, we abounded further analysis on it.

Lasso This is a linear regression method that regularize the parameters using L_1 regularization $\beta_{LASSO} = \arg\min_\beta \|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$ here λ is a hyperparameter, we have to tune. As we increase λ, the solution will become more sparse. If we instead used a L_2 regularization of both L_1 and L_2, we fit a ridge or elastic net where we also would obtain shrinkage. However, we got the best preliminary results with Lasso.

Random Forest Here we use bagged decision trees and try to reduce the correlation between the trees. The correlation is reduced by restricting the possible set of splitting features at each tree node. At a tree node take m randomly sampled without replacement from all the features p and find the optimal features by using either the *gini coeffient* for classification of MSE for regression.

2.2 Model selection and validation

First we handle missing values, then preprocess the data, do feature selection then we tune hyperparameteres and based on the preliminary results based on RMSE on training data and the stability of the models, we took out best for further analysis, in our case *Lasso* and *Random Forest*.

2.3 Missing values

The data contains missing values only for categorical variables and we considered multiple ways to handle these. We thought of multiple ways to handle these. We didn't want to remove the observations containing missing values as we only had a limited number of observations. First, we tried to fill the missing values as we know there is only few options, namely H,I,J,G,K. So we first tried to look at the distribution of these value to fill them randomly but proportionally to their frequency. However, as we do not know the origin of the data and the behaviour of the categorical, we abandoned the idea. We opted to replace the missing value with a new category name, "Missing".

2.4 Factor handling

We used two data transformation steps in our preprocessing pipeline. First , for the numerical values we used the transformer `StandartScaler()`. Machine learning algorithm doesn't perform too well when the attributes have very different scale. We used standardization to make sure our data has same scale. Our second step in our transformation

is the encoding of the categorical data. At first we tried an `OrdinalEncoder()` to convert the categories from text to numbers. However, with ordinal encoder, two nearby values will be considered similar when running the algorithms. Since we don't know the origin of the data we cannot assume that it is the case. We chose to use `OneHotEncoder()` to resolve this issue, the dataset being small, it won't impact the training time. We got way better results with the One hot encoding so we decided to use it for the modeling.

2.5 Feature selection

To avoid data leakage, we kept the feature selection process and the hyperparameters tuning inside the inner CV loop. We used recursive feature elimination and cross-validated the selection of the best number of features for our model. We ended up using the following features 'x42', 'x51', 'x52', 'x73'. The results were consistent with the list of correlated variables to the response y during preprocessing so we decided to go ahead and use those.

2.6 Hyperparameters tuning

For the Lasso, we used the one-standard-error rule to select the λ for the model because we want a sparse model that describes the model reasonably well without imposing too much bias. We found that it should be 10.610.

We did a GridsearchCV to tune the hyperparameters of the Random forest. We tune the hyperparameters after the feature selection process as the final model will only contain these features. The best parameters for the Random forest were 50 trees with a maximum depth of 10.

2.7 Model Validation

As a metric of predictive performance, we will try to estimate the $\widehat{RMSE}$, that the model would give rise to if presented to the new data x_{new}. To do that we:

1. Take out the best i models
2. Do repeated 10-fold cross validation for each model and get a list of $\mathbf{RMSE}_{model\ i} = \{RMSE_{rep\ 1}^{fold\ 1}, RMSE_{rep\ 1}^{fold\ 2}, RMSE_{rep\ j}^{fold\ i}, \ldots, RMSE_{rep\ N}^{fold\ 10}\}$. We also try repeated bootstrapping.
3. Plot and asses the distribution of $\mathbf{RMSE}_{\mathbf{model\ i}}$
4. Pick model i with lowest variance and take out the best measure of centrality for the distribution of $\mathbf{RMSE}_{\mathbf{model\ opt}}$.
5. Make prection interval for $RMSE$, calculate the RMSE on the test data and see if within the prediction interval.
6. Stipulate that the best measure of centrality is the $\widehat{RMSE}$ or consider $RMSE_{test}$ as data is sparse

Specifically, we used *Lasso* and *random forest*, we did 10-fold CV and repeated 100 times and repeated bootstrapping where we first sampled 63% without replacement of the training data, then did 100 bootstrap samples of the 63% of the data. Then we trained on the bootstrap samples and calculated $RMSE$ on the remaining 37% of the data. We do both because we have sparse data and our estimates might be biased.

3 Results

We made histograms of all the $RMSE_{CV}$ and $RMSE_{Bootstrap}$ for both models. As we anticipated, the bootstrap were more optimistic, and therefore we used the results of the CV method for further analysis, however, both $RMSE$ are approximately normal distributed. We estimated the parameters for the normal dist. for each model:

$$\hat{\mu}^{RF} = 31.435 \quad \hat{\sigma}^{RF} = 11.568$$
$$\hat{\mu}^{Lasso} = 35.725 \quad \hat{\sigma}^{Lasso} = 10.167$$

Based on the result above, we choose the **random forest**.

We can now check the model on the test set. We can even specify a prediction interval for the $RMSE$ using $\mu_{RMSE} \pm z_{0.025}\sigma_{RMSE}$. We get $PI(RMSE_{test}) = 31.435 \pm 22.673$. On the test data, we get an $RMSE_{test} = 34.57$ hence the methods seems reasonable. We are not to provide prediction interval for the $RMSE$ but only $\widehat{RMSE}$. As we have sparse data, we speculate that $\hat{\mu}^{RF}$ might be biased and too optimistic and when tested on unseen data $RMSE_{test}$ was slightly greater. The difference is small but we take the more conservative of the two estimators and stipulate $RMSE_{test} = \hat{\mu}_{RMSE} = 34.57081$ as our result.

With the *random forest* using the tuned hyperparameters we trained the *random forest* on the entire available data set, then imported the new dataset and made the prediction, $\hat{y}_{new}$.