Car insurance claim project

Carolina OLIVEIRA COSTA

Presentation of the data and the associated insurance problem

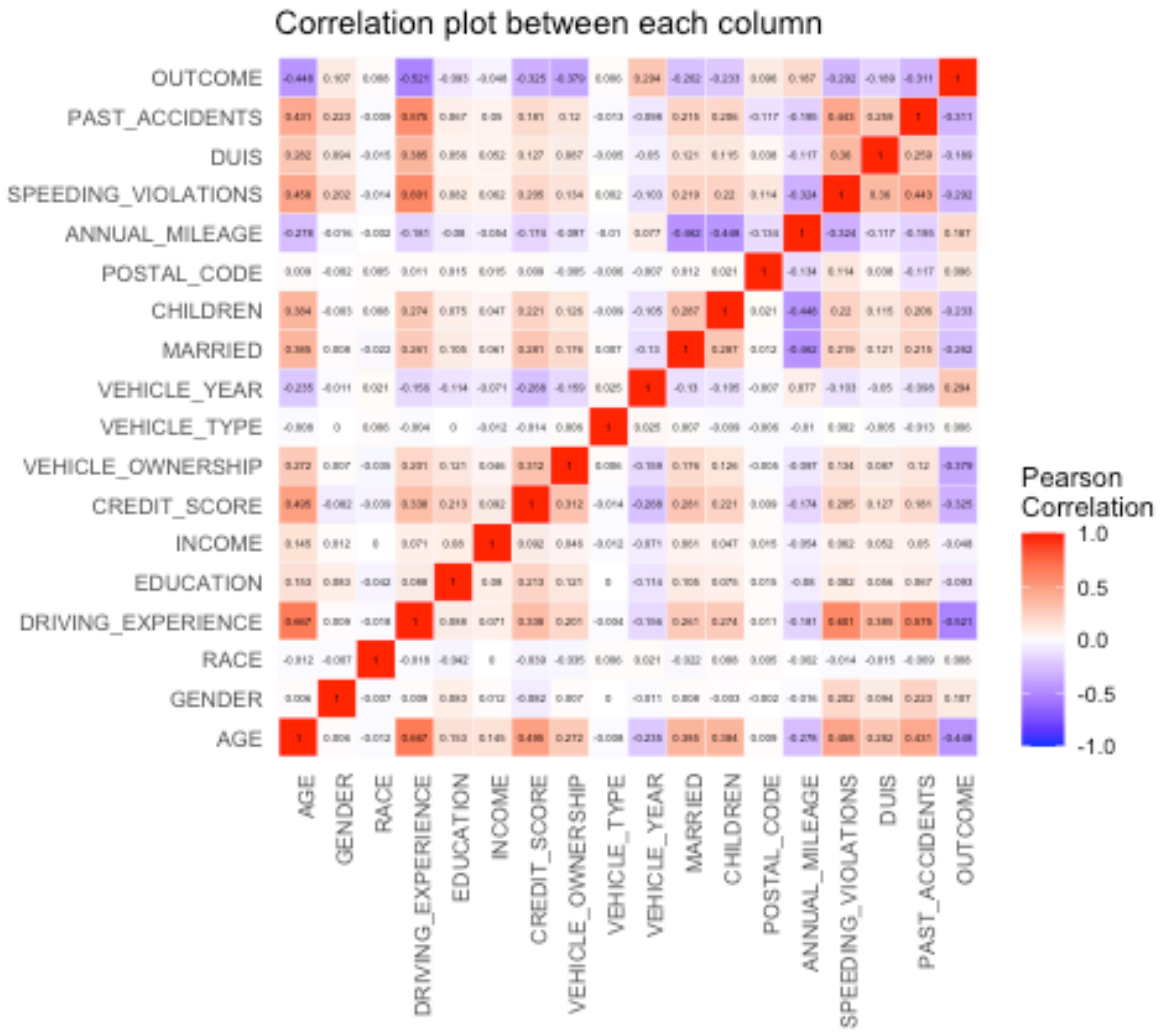
The chosen data set is the Car Insurance Data from Kaggle. It is contains 19 columns and 10000 rows. The columns describe personal informations about the policyholder, like his/her ID number, age, gender, … Some of them are categorical (e.g. Driving experiences and Race) and some numerical ones (eg. Credit score). However there is one column we would like to predict and that one is the column, which is named Outcome. The outcome is if the policyholder has claimed his/her loan or not. Many insurance companies offer many services than just insurance and lending money to someone is a very thoughtful task. Giving money to the wrong person can be a big loss depending on the amount loaned and the number of those "wrong" people since they will never give that money back. Hence it is also good to find a good model that can help the insurance company find who and who not lend money to.

Preliminary analysis of the data

Before we start talking model, we will need to take a closer look at the data.

First, we have to note that there is 982 missing data in the column of the credit score and 957 missing data in the column of the annual mileage. This issue has to dealt with later.

Then we plot the correlation plot between each column leaving the ID out since the ID will not be a feature for the prediction.



If we only the correlation values between the outcome column and all of the rest of the columns, we can get the columns with the most influence and with the least influence on the prediction. Those most influencing columns are the driving experience, vehicle ownership and credit score and the least influencing columns are race, age and education. We can forget about the 3 latter ones, but it is nice to visualize our data and this is what we are going to do later.

Imbalanced categories

Before taking any decision or making any change to the data set, we take a quick look at each column, more specifically the categorical columns, and its values. We plot the count of each category for each column. Most of the categorical columns seem to have very close in numbers for each category option, others can have very large numbers for each option. There is one column we will try to fix that problem. This column is the years of driving experience. Here is the plot:

If we merge the categories "20-29y" and "30y+", we will get a count around the number of the other categories of the same column. We will name the new merged category "20y+".

Categories to numbers

The multiple categorical columns have to be converted to numbers before fitting them to the model. The model only understand numbers and not text. The ones with a logical order will be numbers from 1 to the number of categorical options of the corresponding column. For example the education column has the options: none, high school and university. The number representation is 1, 2 and 3. This method is called label encoding. Moreover, there exists another method, called One Hot Encoding. One Hot Encoding is helpful for column with no logical categorical order. The data set contains such a column, the vehicle type. One Hot Encoding makes the data set sparse, but this algorithm will make it easier for humans to understand the conversion to numbers. If we would do a label encoding to the vehicle type, it would probably assign "sedan" to 1 and "sport car" to 2. There is no real logical reason behind this choice. So the One Hot encoding is the best option for the latter column. The One Hot Encoding creates new columns. Each new column is for each category option and its values are either 0 or 1. For each row, only one column of the same One Hot Encoding will contain a 1 and the others will be 0. The column with the 1 is the category the data belonged. Now we are ready to deal with missing values.

Missing values

Furthermore, there a significant value of correlation between the outcome and both credit score and the annual mileage. The more logical next step is to fill the missing values in this 2 previous columns. For that there is an algorithm in R, call missForest. It uses a random forest algorithm to predict all the missing values. For more details how the random forest algorithm works, an explanation is given in next section.

Plots

Brief description of the models used

After the analysis, we split the data into training and test. The training data is 80% of the whole data and the test data is the rest 20%. With the training data we train the model and with the test data we will evaluate how good our model is.

Machine Learning models don't well on data which contain an imbalanced size of the predicting class. After balancing the data with a function in R, called upSample, the data set gets duplicated rows for the minority class.

The main task of the model is to do some classification prediction. So we will be able to pick the best performed model between three chosen model: Random Forest, Support Vector Machine and Naïve Bayes. We will explain shortly the models without going too much into details and then show their results.

Random Forest

The random forest algorithm is a collection of decision trees. One question one might ask now is what is a decision tree. The answer is: a decision tree is a tree where the nodes are the decisions and the edges are the tests. Those trees contain multiple levels at which one can do tests and decisions. Hence the random forest generates many trees based on different subsets of the training data with multiplication of the same rows. At the end, we will get one decision tree. The decisions will be chosen by majority vote for categorical and by the mean for continuous from the latter collection.

We train the random forest algorithm with our train data. Next we predict the same training data and evaluate this prediction.

Support Vector Machine

The support vector machine, also named SVM, is another algorithm for classification. It tries to generate a line that separates the two predicting classes when drawn in a hyperplane.

We train the SVM algorithm with our train data. Next we predict the same training data and evaluate this prediction.

Naïve Bayes

First note that the Naïve Bayes algorithm assumes independence between all of features. It calculates the posterior probabilities using the Bayes Theorem for each outcome class. Then it predicts the class such that this probabilities is the largest.

We train the Naïve Bayes algorithm with our train data. Next we predict the same training data and evaluate this prediction.

Analysis of the results and conclusion

As expected, the random forest algorithm has the best accuracy. Thus it will be the algorithm which we will perform a cross-validation with 5-folds. K-Fold Cross-validation splits the training data into K subsets. At each iteration it evaluates the model at each different different subset after the model trained on the other 4 subsets. Each model created can have very different value for each parameter. The model with again the largest accuracy "wins".

We are ready now to evaluate the final model on the test data. This is the following result: