Car insurance claim project

Carolina OLIVEIRA COSTA - 0180395513

Presentation of the data and the associated insurance problem

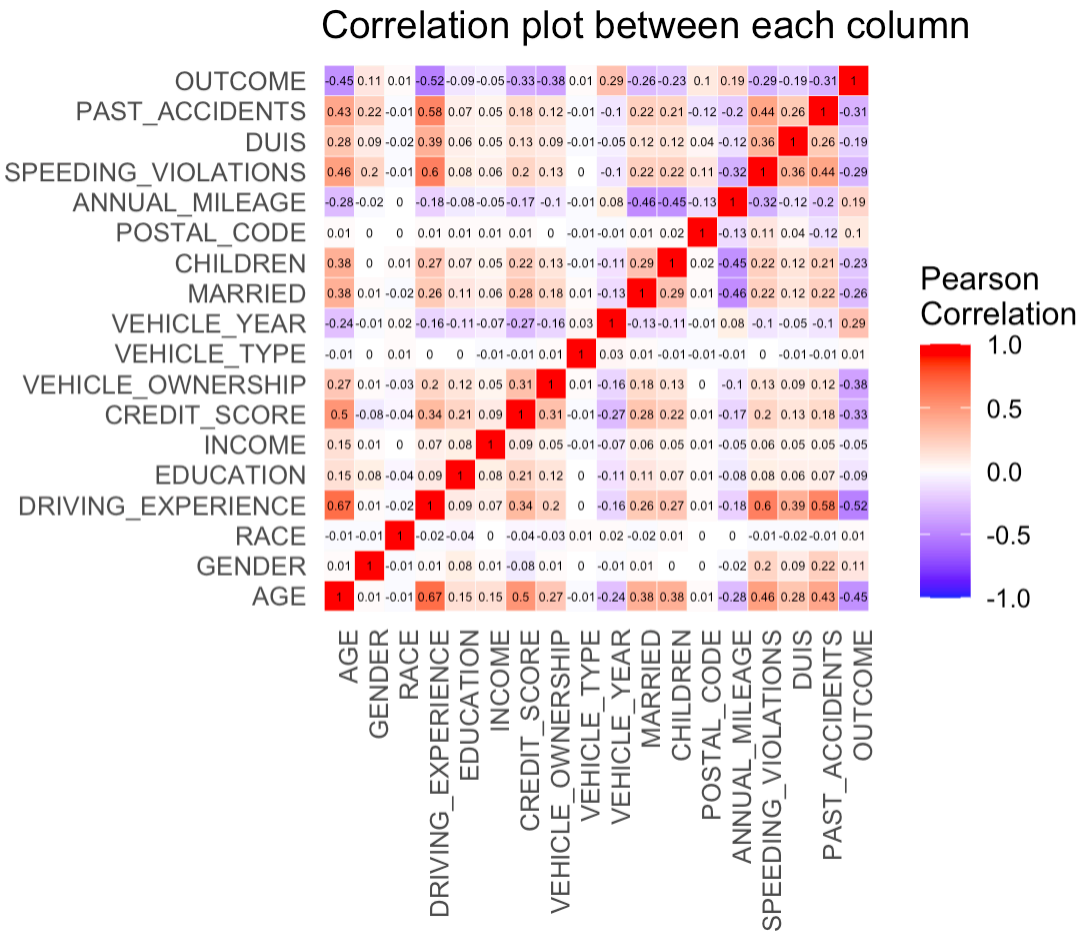
The chosen data set is the Car Insurance Data from Kaggle. It contains 19 columns and 10000 rows. The columns describe personal informations about the policyholder, like his/her ID number, age, gender, … Some of them are categorical (e.g. driving experiences and race) and some numerical ones (eg. credit score). However there is one column we would like to predict, that one is the column which is named *outcome*. The outcome expresses if the policyholder has claimed his/her loan or not. Looking at this data through the perspective of an insurance company, we can train a model to help them improve many services. Some of those services can be of course an insurance contract, but it can also be offers of financial products. By financial products, we mean lending money to someone for example. Giving money to the "wrong" person can be a big financial loss depending on the amount loaned. This "wrong" person is defined as someone who is not able or willing to repay the loan**.** Hence it is super important to find a good model that can help the insurance company find only good clients to lend money to. As everyone knows, loans can be very profitable to the ones lending the money, because of the interest rate which can be very high.

Preliminary analysis of the data

Before we start talking about models, 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 be dealt with later.

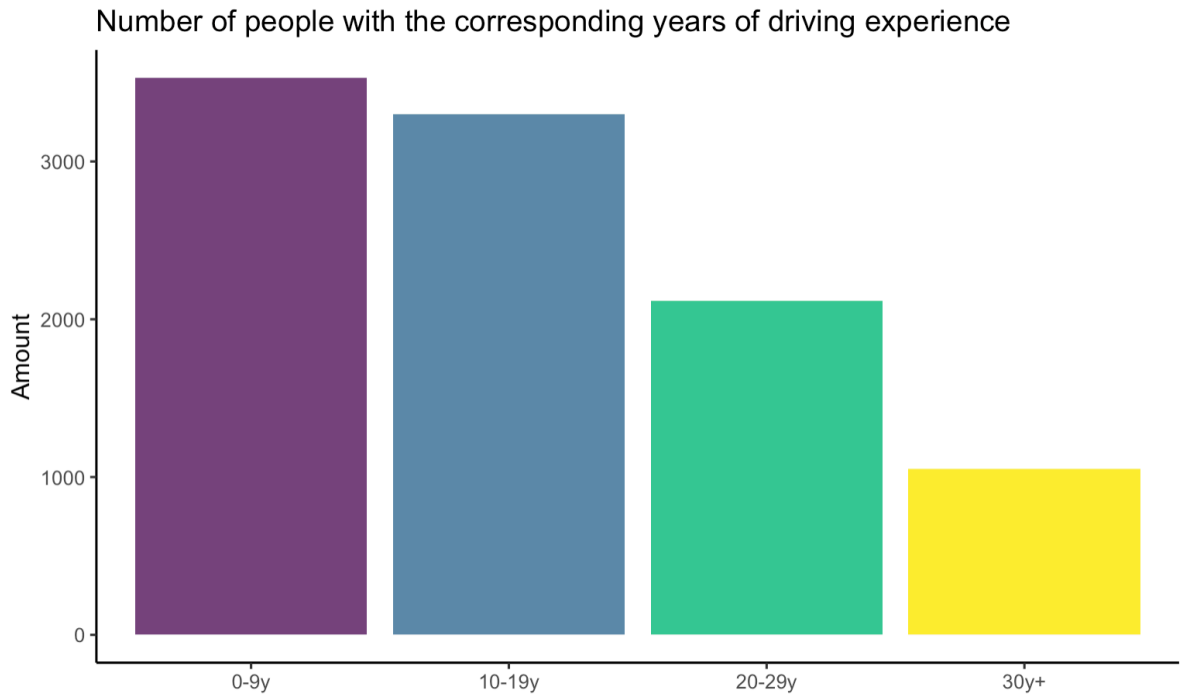
Then we plot the correlation plot between each column leaving the ID column out since the ID number is always unique. The ID number is not helpful to predict, hence it doesn't make sense to fit it to the model.



If we only consider the correlation values between the outcome column and the other columns, we can get the columns with the most and the least influence on the prediction. Those most influencing columns are the driving experience, vehicle ownership and credit score. On the other side, 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

Without 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 their categorical options. 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, where 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 an amount 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.

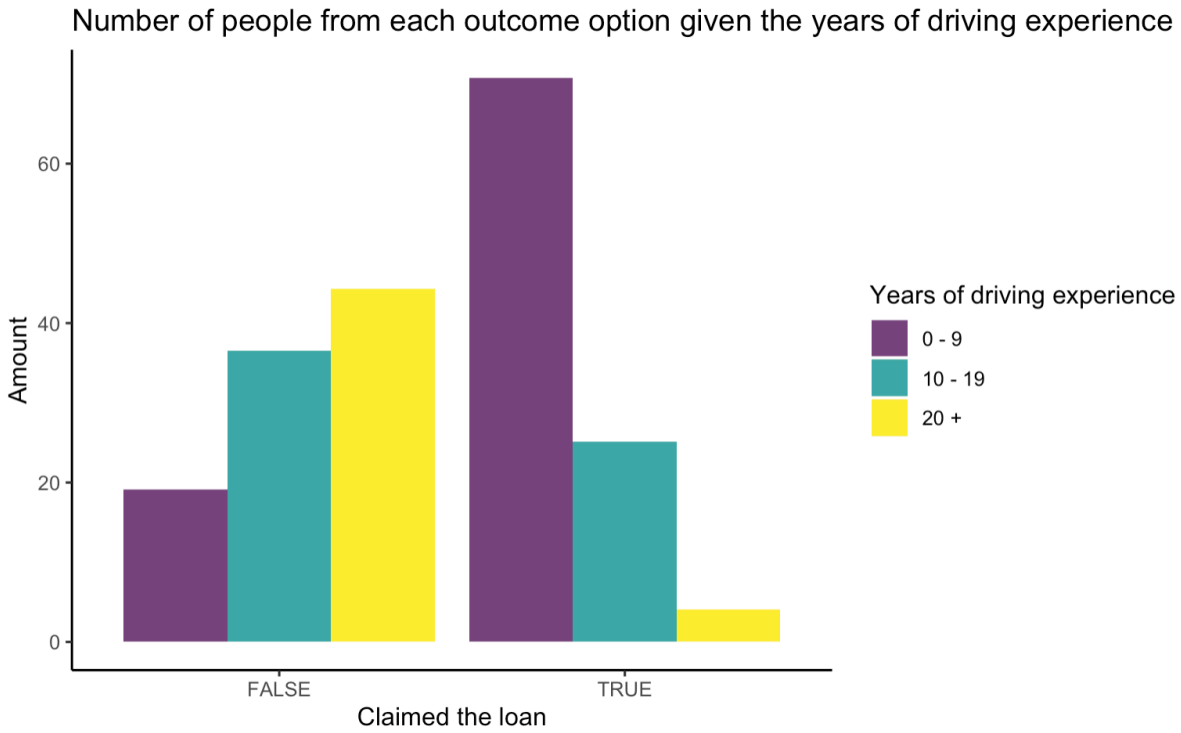
Missing values

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 the next section. Furthermore, it can handle both categorical and numerical columns well. After checking the distribution of the data for each column before and after generating data, we decided to accept the generated data since the distribution maintained its structure.

Plots

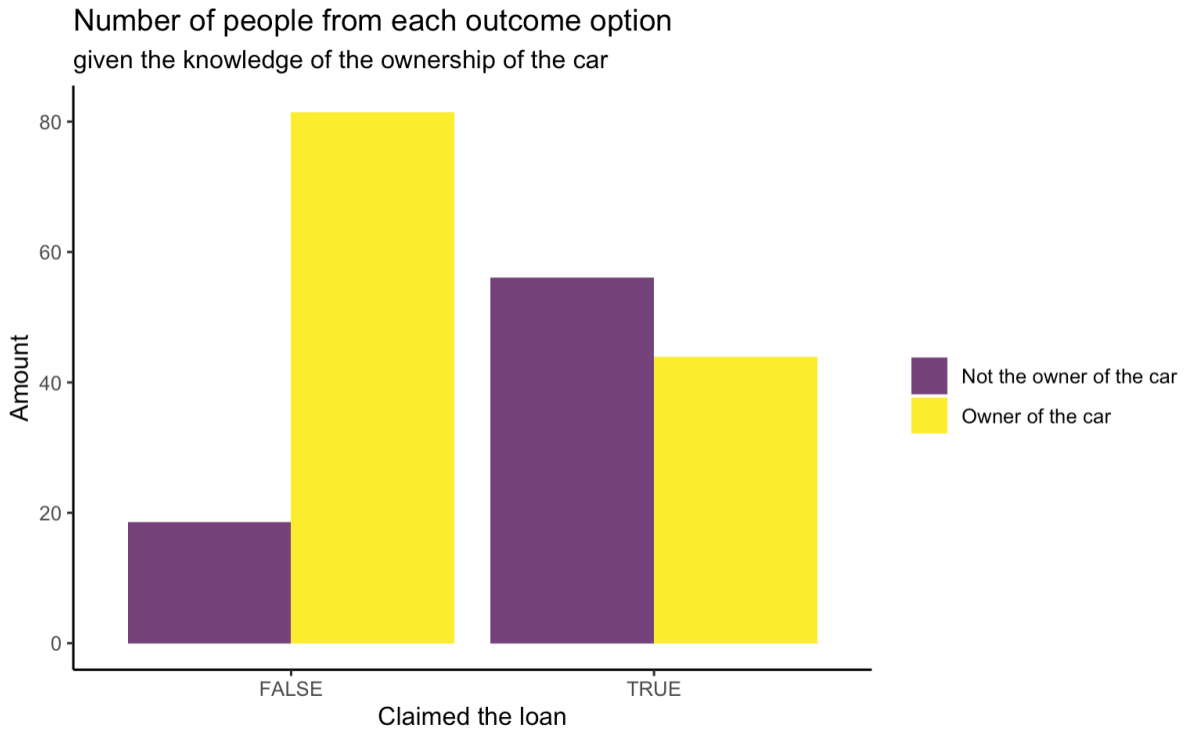
We have so many columns, but there are some columns more important than others. That is why we will only focus on the 3 most correlated column to the column we want to predict.

The first column is the years of driving experience. We already talked about this feature before, so we can immediately show the plot:



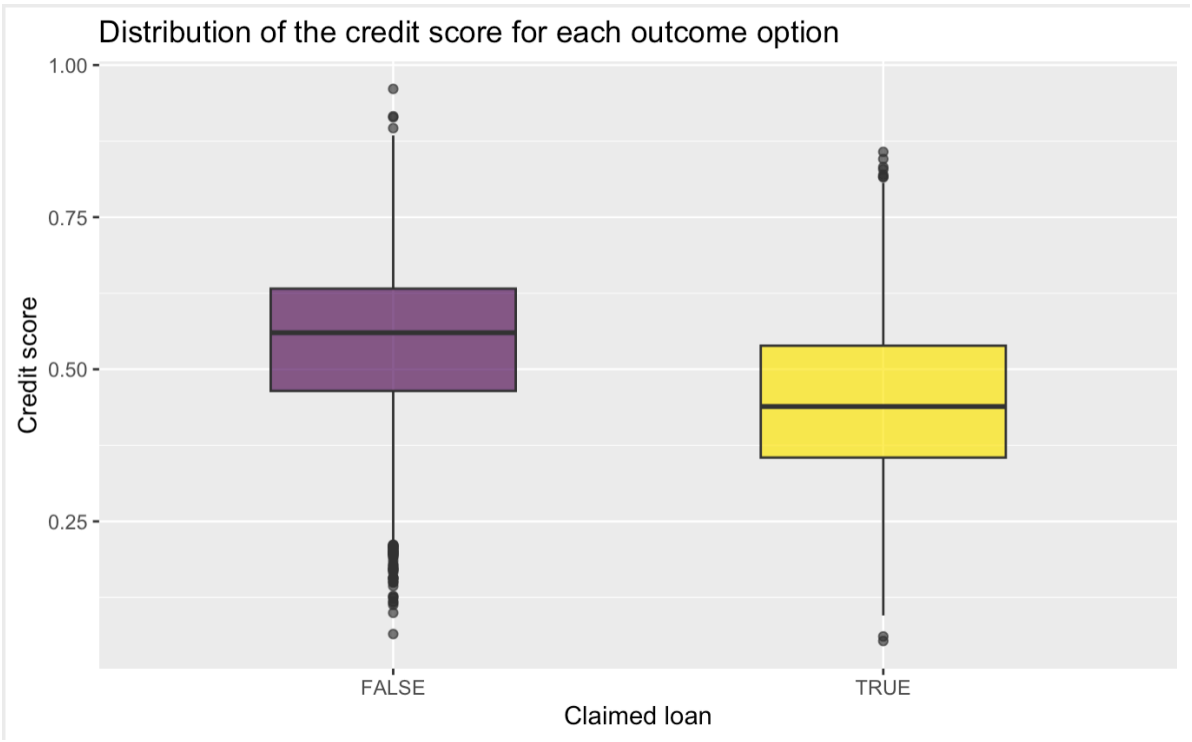
It is interesting to see that the less years of driving experience one has, the more likely it is to have claimed a loan.

Moving on to the ownership of the vehicle, it is a binary categorical column. The value 1 represents that the data is about the owner of the car and value 0 represent otherwise. There exist many reason to pay an insurance plan for the car even when one doesn't own it, e.g. parents paying insurance for their children etc.



This plot shows some shocking information. The probability of the owner of the vehicle get a loan is less than the non-owner of the vehicle.

On top of that, the column known to have an influence on the decision of the insurer to give a loan or not is the credit score. The credit score is between 0 and 1. The closer the credit score is to 1, the more likely it is to get a loan. But the following plot shows us the opposite.



More explanation about the above plot: The boxes represent where the most frequent values are, for example for the class "not claimed a loan", the majority of people have credit scores between ~ 0.45 *and* ~ 0.63*. The line is between the box is represents the middle number in a sorted list of all values. Thus again for the class "not claimed a loan", this line is higher than* 0.50.

We finished the data analysis, the next step is the model selection and training the model.

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 do 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 five chosen model: **Random Forest, Support Vector Machine, AdaBoost** and **Gradient boosting machine**. We will explain shortly the models without going too much into details and then show their results.

For each model, we will perform a cross-validation of 10-folds 5 times. K-Fold Cross-validation splits the training data into K subsets. At each iteration it evaluates the model at each different subset after the model trained on the other 4 subsets. Each model created can have very different values for each parameter. Again the model with the largest accuracy "wins".

We train each model with our train data. Next we predict the testing data and evaluate this prediction.

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 columns and by the mean for continuous columns from the latter collection.

Support Vector Machine

The support vector machine, also named SVM, is another algorithm for classification. Each data has a coordinate position in a more dimensional map. The model tries to generate a line (or hyperplane) between the data points that separates the two predicting classes.

AdaBoost

The AdaBoost algorithm uses Decisions Trees. The first iteration assigns each observation with equal weight. We evaluate the fitted tree. After that, new weight are assign to the each observation. Those observations more difficult to predict will get heavier weights and those observation more easy to predict will get lighter weights. And so on until it gets a tree with a small error which is exactly the difference between the predicted value and the actual value.

Gradient Boosting machine

This algorithm is kind of an extension of the previous model. Instead of having the difference between the predicted value and the actual value, the algorithm can use any other loss function. The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data.

Analysis of the results and conclusion

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

Algorithms	Accuracy	Specificity
Random Forest	0.8258	0.8506
Support Vector Machine	0.8123	0.8047
AdaBoost	0.8313	0.8207
Gradient Boosting Machine	0.8298	0.8200

Choosing the best model is not an easy task, however the metrics **accuracy** and **specificity** will help us to do so. The **accuracy** is the division of number of correct predictions by total number of predictions. We desire this number to be high. Furthermore, the **specificity** is another metric which expresses the ability of a test to correctly identify policyholders who *haven't* claimed a loan. This number should also be clearly high.

The best model is one that can maximize both metrics. SVM has the worst accuracy and worst specificity. Random Forest has the highest specificity, but AdaBoost has the highest accuracy. Besides AdaBoost and Gradient Boosting Machine have almost the same values for both metrics. The accuracy value of the model Random Forest which already contains the highest specificity is very close to the model with the highest accuracy. Therefore for this prediction, Random Forest is the best appropriate model.

References

Articles

[1] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, Asdrubal Lopez: A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing, Volume 408 (2020)

Websites

[1] https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

[2] https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[3] https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/#:~:text=What%20is%20the%20AdaBoost%20Algorithm,are%20also%20called%20Decision%20Stumps.