PREDICTING FRAUDULENT INSURANCE CLAIMS USING MACHINE LEARNING

Davor Stefanovski

Faculty of Electrical Engineering and Information Technologies

University of Ss. Cyril and Methodius in Skopje, Macedonia

davorflute@gmail.com

## Abstract

Car accidents are inevitable, but sometimes policyholders attempt to fake accidents to claim money from their insurance companies. In this project, I developed models to help companies determine whether a claim is fraudulent. The dataset preparation involved fixing missing values, adding new features, and encoding categorical data. To optimize model performance efficiently, I selected the best features and performed hyperparameter optimization. After training and comparing various models, I found that a Voting Classifier, combining two base models with complementary strengths, provided the best results for this imbalanced problem. Despite the model's effectiveness, it is intended to be used alongside human judgment, thus enhancing the company's ability to identify fraudulent claims.

## Introduction

In this project, I utilized the 'Vehicle Insurance Claim Fraud' dataset by Shivam Bansal (2022), which I sourced from Kaggle. The problem I aim to solve is predicting whether an insurance claim for car damage is fraudulent or non-fraudulent. This can be super beneficial to insurance companies, as a tool to help the agents whether to initiate an investigation or just to pay the funds and avoid additional costs. I chose this project because the highly unbalanced dataset, with a ratio of fifteen non-fraudulent claims to one fraudulent claim, presented a significant challenge. The prediction is based on 32 attributes related to the accident, the policyholder, the claimant, the policy, various time intervals, and other factors related to the claim. First, I preprocess the data, then analyze it to extract additional beneficial features. Next, I select models for prediction, tune them, and evaluate their performance. The best-performing model should not only prioritize accuracy but also focus on identifying fraudulent entries, as companies aim to minimize losses from fraudulent claims.

## Materials and Methods

The 'Vehicle Insurance Claim Fraud' dataset, as I said was sourced from Kaggle (Bansal, 2022). It contains 32 attributes and about 15420 entries with highly unbalanced dataset.

Missing values were handled using mean imputation for the Age feature and the faulty entry, where “DayOfWeekClaimed” and “MonthClaimed” was ‘0’ was completely deleted. Categorical variables were encoded using one-hot encoding, label encoding and mapping. Additional features were engineered, including “Time Interval” which is the time interval between the accident and the claim.

Various models were selected, including Decision Tree, Random Forest, Ada Boosting, Balanced Random Forest and Voting, which was a combination of Balanced Random Forest and Ada Boosting. The dataset was split into training and testing sets using an 80/20 split with stratification. After one evaluation I did a RFECV feature selection for the three best performing models and used the reduced datasets for the new trainings. Hyperparameters were tuned using Randomized Search to find optimal hyperparameters with a small execution time. Resampling techniques like SMOTE and Balanced Random Forest as a cost-sensitive learning approach were employed to address the class imbalance.

Models were evaluated using many metrics such as accuracy, precision, recall, F1-score, ROC-AUC and balanced accuracy, as well as the confusion matrix of the predictions. The focus was on recall, precision for fraudulent claims and the sum of all the recalls, also known as Balanced Accuracy. The performance of different models was compared to identify the three best-performing ones, which a further investigated and tuned to get even better results.

## Results

**At first glance, the dataset didn’t seem to have missing values,** but after taking a closer look, I decided that I needed to address some values as missing. I found a faulty entry which didn’t mean a lot to the whole model, so I removed it. And the others, in order not to lose too many entries, I filled them with some logical values.

**Next, I did Exploratory Data Analysis**, which consisted of heatmap between the numerical features, target distribution graphs for all the features, target count plot graphs for all the features and after encoding the categorical features – another heatmap. The results of the initial heatmap were disappointing. The most correlated numerical feature was deductible, which is the funds that the person must pay before the insurance continues. The features “BasePolicy” (Figure 1.), “PoliceReportFilled” (Figure 2.) and “NumberOfCars” showed excellent distribution of the target. When it comes to the fraudulent claim count plot in respect of the categorical value, interesting facts showed up:

* Most fraudulent claims are submitted by married people.
* The cars used for these fraudulent claims are mostly:
  + More than 6 years old
  + Valued between 20 thousand and 30 thousand dollars (Figure 3.)
  + Sedans (Figure 4.)
* Policyholders between 30 and 40 years old appear to submit more fraudulent claims than others.

The last heatmap, after I encoded the categorical features showed that “Fault”, “PolicyType\_Sedan – Liablity", “VehicleCategory - Sedan”, “VehicleCategory - Sport”, “BasePolicy - All Perils” and “BasePolicy - Liability” features show mild correlation with the target.

**For this project I chose multiple training models at the beginning**, and after the preliminary evaluation, I kept only three for further optimization. I divided the data using an 80-20 stratified train-test split ratio, trained the models, and subsequently compared the results. The model I used to reference the others to was Decision Tree, then I trained:

* Random Forest without any dataset balancing.
* AdaBoost with a base estimator of Decision Tree.
* Balanced Random Forest model imported from the imblearn library.
* Combination of Balanced Random Forest and AdaBoost using Voting.

As we can see from the comparison of the results (Table 1.), the Voting and Balanced Random Forest algorithms gave the best-balanced accuracy score. I tried replacing the Balanced Random Forest with oversampling the minority class using SMOTE and training Random Forest, but it gave worse results, while costing more time, so I continued using the previous pipe.

**For the feature selection part**, I tried both Univariate selection using “SelectKBest()” function and recursive selection with cross validation. I was trying the select 15 features and these are the features chosen for the AdaBoost algorithm:

* 'Month'
* 'WeekOfMonth'
* 'DayOfWeek'
* 'DayOfWeekClaimed'
* 'MonthClaimed'
* 'WeekOfMonthClaimed'
* 'Age'
* 'Fault'
* 'PolicyNumber'
* 'RepNumber'
* 'PastNumberOfClaims'
* 'AgeOfVehicle'
* 'AddressChange\_Claim'
* 'PolicyType\_Sedan - Liability'
* 'BasePolicy\_Liability'
* 'IntervalInWeeks'.

In the meantime, the features chosen with the Balanced Random Forest were:

* 'Month'
* 'WeekOfMonth'
* 'DayOfWeek'
* 'DayOfWeekClaimed'
* 'MonthClaimed'
* 'Age'
* 'Fault'
* 'VehiclePrice'
* 'PolicyNumber'
* 'RepNumber'
* 'Deductible'
* 'DriverRating'
* 'PastNumberOfClaims'
* 'BasePolicy\_Liability'
* 'IntervalInWeeks'

For the training of the Voting, I used the features that were chosen with the Balanced Random Forest, since it had more weight to its vote than AdaBoost.

**Before training the new models, I did hyperparameter optimization** using RandomizedSearchCV(). For the optimization of the AdaBoost model, I started with optimizing the estimator, which was decision tree. The output I got was a model with “max\_depth” = 5, “min\_samples\_split” = 5 and I used that model to optimize the AdaBoost. The parameters that my search found were: “learning\_rate” = 0.1 and “n\_estimators” = 800. In the optimization of Balanced Random Forest, I got a combination of parameters, where “min\_samples\_leaf” = 2 and “n\_estimators" = 200. In the end, I used the optimized models as the voters and optimized the weights of their votes. The search chose 0.6 as the weight of the Balanced Random Forest and 0.4 as the weight of the AdaBoost.

**I finished this project by evaluating and comparing all the models that I trained**. The results after the feature selection and hyperparameter optimization were significantly better than before. In the table (Table 2.), we can see that the voting shows the best results when it comes to Balanced Accuracy, so this model is my final choice for predicting this problem.

## Discussion

The feature “Age” had 320 entries valued “0”, which is obviously some sort of error. Because the age of the person who claimed the insurance was often around the age of the policy holder, I decided to fill these entries with the average of that interval. The problem with the “MonthClaimed” and “DayOfWeekClaimed” values that were “0” was completely solved by deleting the 1516th entry, since it was the only entry.

The first heatmap showed that the numerical features are almost not correlated to the target and yet I tried to use them because they might work well together. In the data analysis I found out that very little of the policy holders who bought liability policy claimed fraudulently. Probably, because those policies cover medical bills as well, which are not funded if the claimer is medically fine. Also, the number of vehicles seems to contribute well to the target. Very little fraud was committed by people who own more than 1 car and that has to do with their financial stability. Here, I found out that the police did a good job, with 0 frauds when a policeman filled a report. Further research showed that most of the frauds were committed by married claimants, aged 30-40 and with cars that fall into the lower-mid class that are often older than 6 years. This made a lot of sense, which gave me more trust in the dataset. Finally, after I encoded the features, I got a little better correlation heatmap.

I encoded AccidentArea, Sex, Fault, PoliceReportFiled, WitnessPresent, AgentType binary with LabelEncoder(). Then, I had a lot of features like: 'Month', 'DayOfWeek', 'DayOfWeekClaimed', 'MonthClaimed', 'VehiclePrice', 'Days\_Policy\_Accident', 'Days\_Policy\_Claim', 'PastNumberOfClaims', 'AgeOfVehicle', 'AgeOfPolicyHolder', 'NumberOfSuppliments', 'AddressChange\_Claim', 'NumberOfCars' that were ordinal, so to get the most value out of their meaning, I encoded them ordinally with mapping. The other categorical features were neither binary nor ordinal, so I had to encode them with one hot encoding in order not to confuse the model.

Since “Days\_Policy\_Accident” and “Days\_Policy\_Claim” are expressed in a strange way, where most of the entries fall in the category “30 days or more”, I extracted a new feature “TimeInterval” for the time interval between the day when the policy was made and the day of the claim. That function was not easy. It works by calculating the number of weeks between those timestamps, but in the current year – so it is not 100% accurate, but since that feature was selected with every selection method – it was a great idea.

In the model training section, I chose the reference to be Decision Tree because the Dummy classifier would give very good accuracy while failing all the fraudulent entries, which are of extreme importance. I knew that Random Forest would have excellent accuracy, but it obviously needs some kind of balancing the dataset. At first sight, oversampling the minority class and training regular Random Forest works similarly to training Balanced Random Forest in the first place, but Balanced Random Forest Worked a lot better. I suppose that this happened because of the way the balanced dataset is made. When I use the “SMOTE” function, it creates new entries by combining k-nearest neighbors, which may create incorrect entries, thus resulting in worse performance. Balance Random Forest doesn’t even create new entries, it samples the fraudulent and non-fraudulent entries unevenly. The motive behind AdaBoost was trying to improve the performance on the misclassified entries, by giving them more priority in the next estimator. Finally, Voting was trained in order to combine the strengths of both the AdaBoost and Balanced Random Forest.

The reason for choosing “RFECV” over “SelectKBest” was because training the model with its specific selected features will give the best performing model, but it may need some more computational resources. The selected features between the different models and the ones selected by “SelectKBest” were similar and they all included my extracted feature “IntervalInWeeks”.

I used “RandomizedSearchCV” for hyperparameter optimization just because when using it with enough iterations, you can get a better idea of where the global maximum lies. After that, my plans were using “GridSearchCV” for fine tuning the parameters. I did all this tuning of the hyperparameters on the validation set using Cross Validation, in order not to overfit my model on the test set.

In the end, the best performing model was Voting after selecting the features and optimizing the hyperparameters. In the plot (Figure 5.), we can see that there are a few misclassified non-fraudulent entries, but that was the sacrifice that was needed in order to have a model which will excellently classify the fraudulent entries. This “threshold” can be further adjusted by tuning the vote weights of both voters and it depends on the intentions and policies of the company.

## Tables and Figures

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

Figure 1. Figure 2.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 3. Figure 4.

Table 1.

A screenshot of a computer

Description automatically generated

Table 2.

A screenshot of a computer

Description automatically generated

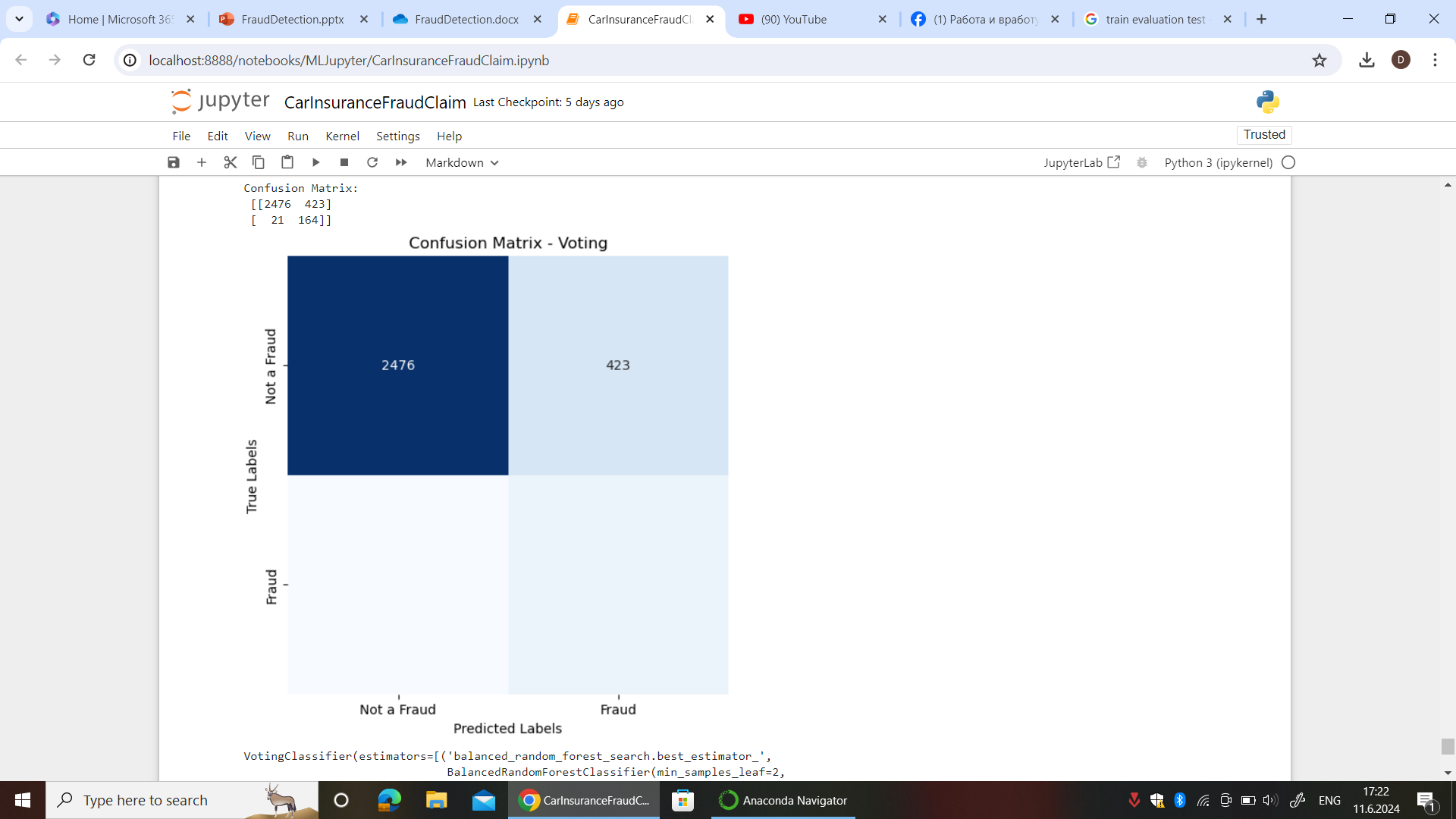


Figure 5.

## References

1. Shivam Bansal. 2022. Vehicle Insurance Claim Fraud Detection. https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection