Vehicle Insurance Fraudulent Claim Detection

Fraser Wong, Jaisreet Khaira, Richard Wijaya

1. The Problem

When involved with an auto accident, vehicle insurance policies are needed to protect people from financial problems. Insurance companies are giant financial companies who collect premiums from policyholders and use the money to pay for the losses of the people who are involved in an accident. Usually, this is in the form of paying for fixing the car, replacing the vehicle, or paying for medical bills. While this is very helpful for policyholders, there are risks for the insurance companies, including frauds. It might be possible for people to 'fake' an accident and submit a claim to get 'free' money from the insurance.

Since insurance companies use statistical calculations to come up with a premium to charge the policyholder, they would need to collect a lot of data. Some categories of data are more significant than others, which is something that will be explored in this project. For instance, age is among the most important factors when determining insurance premiums. Therefore, we believe that the age of a policy holder plays a significant role in the process of determining fraud. We will attempt to find whether or not this is the case. This project will use machine learning models to predict fraudulent claims, determine the model that is best suited for the job, and then remove the policy holder's age and run the models again, to determine if it's significant.

Another purpose of this project is to demonstrate different ways to deal with imbalanced data. Vehicle insurers get a lot of claims and most of them are not considered fraudulent. The data that is used in this project reflects that, which means only a small fraction of the total data points are considered fraud. This is one example of imbalanced data and some measures need to be taken to make a useful model.

2. The Data

The data we used for this project is a dataset we retrieved from Kaggle, that contains the details of 15,420 unique insurance claims. In addition, each insurance claim contains a final verdict on whether or not the claim is fraudulent. In the dataset, this verdict is stored in the column called FraudFound_P, where the value 1 means a fraud is found and the value 0 means a fraud is not found.

Initially inspecting the dataset, we discovered several duplicate and irrelevant columns that we could remove. First we removed the PolicyNumber, RepNumber, and AgentType columns since they were not related to our objective, but rather just different types of identifications for the policy.

While inspecting the unique values of each column, we noticed several concerns. The columns DayOfWeekClaimed and MonthClaimed contained the value 0. These columns should have contained strings representing the day and the month that a claim was made. There is 1 row of each of these cases, so after considering the large number of data, we decided to remove these rows.

Another concern was about the values in the column Age, which included 0. It is not possible for a person aged 0 to have a car, so we considered these data input errors. There are 320 rows with this value, which we considered a significant number of rows. However, there is another column called AgeOfPolicyHolder, which categorizes the policyholder according to their age group. When the value of

Age is 0, the value of AgeOfPolicyHolder is ‘16 to 17’, so we decided to replace the value 0 in Age with 17 to make sure the data is more useful.

The majority of columns in the dataset contained categorical values, which do not work well with the majority of machine learning models because they do not take non-numerical inputs. In order to change this, we implemented a technique known as One-hot encoding to convert the categorical variables into a numerical format. One-hot encoding works by representing each category as a binary vector, where all elements are zero, except for the element that is representing the category. For example, the column VehicleCategory contained 3 categories: Sedan, Sport, and Utility.

The tables below are examples of how one-hot encoding transforms categorical variables into numerical values that can be used to build the models.

Before One-Hot Encoding

Policy Holder	Vehicle Category
Person 1	Sedan
Person 2	Sport
Person 3	Utility

After One-Hot Encoding

Policy Holder	Sedan	Sport	Utility
Person 1	1	0	0
Person 2	0	1	0
Person 3	0	0	1

This technique transformed the dataset into a format that can be used as input in our machine learning algorithms. This transformation can be applied to the dataset before fitting the model or as part of the pipeline of fitting the model.

3. Analysis Techniques

We employed several steps of data analysis in this project as follows.

1. Obtain Significance of Columns

First we analyzed the significance of each column in relation to the fraud status. To do this, we performed Chi-Squared tests on each of the columns against the FraudFound_P column. The columns with high p-value means that its value is not associated with the value of FraudFound_P. Specifically, we paid close attention to the “AgeOfPolicyHolder” column as we want to determine its relation to the rate of fraud. Its resulting p-value was 6.15×10^{-5}, which is well below the 0.05 threshold for significance. Therefore our prediction is that the machine learning models that we use will be significantly affected by the removal of this column.The next steps involved finding an ideal model to test our prediction.

2. Train-Test Split

Before fitting any model, we splitted the data into training and testing sets. This is to make sure that the accuracy score that we obtain is not the result of overfitting and bias towards the data that we have. The

split makes sure that the accuracy score that we get with the model will reflect real data that the model would not have seen.

3. Resampling

One issue that we needed to address in our data set is the uneven distribution of fraudulent claims to non-fraudulent claims. There is only a small fraction of fraudulent claims in the data set, 923 data points, compared to 14,497 data points of non-fraudulent claims. This was a problem because the algorithms we applied would be biased towards predicting the majority class (non-fraudulent), resulting in a misleading accuracy score. A model that flags claims as non-fraudulent all the time is not useful for the insurer.

To avoid this problem, we tried different methods to balance the number of fraudulent and non-fraudulent claims. The first method is oversampling, where we resampled the fraudulent claims with replacement, increasing the number of fraudulent claims in the data set to match the number of non-fraudulent claims. The second method is undersampling, where we sampled the non-fraudulent claims, selecting only a fraction of the fraudulent claims data to match the number of the fraudulent claims. These resampling methods are done after the train-test split, using the training data. The resampled data is then fitted to several models.

4. Training Models

Using the resampled data sets, we tried fitting 6 machine learning algorithms to figure out which would work best, namely, logistic regression, naive Bayesian classifier, K-Nearest Neighbors classifier, decision tree classifier, random forest, and boosting classifier. As our data contained many categories, we primarily focused on decision trees and gradient boosting algorithms as they are well suited for high-dimensional data. We also decided to include logistics regression, naive Bayesian, and K-nearest neighbor algorithms to see how accurate they would be.

5. Evaluate Models

To evaluate how well the model can flag fraudulent claims, we used several metrics on the prediction made using the testing data. The first one is an accuracy score, which is the percentage of the prediction that matches the actual fraud status in the test data. This number might not reflect the usefulness of the model since we are working with imbalanced data. The second metric is a classification report on the prediction, which includes the precision, recall, and F-1 score. We want to maximize all these values to make sure we make the correct fraud classification on the relevant claims.After testing these models, we concluded that the model with the best results is random forest.

6. Light Gradient Boosting Algorithm

The results of all the models we were using, which will be discussed in detail in the next section, were not good. The precision and recall values were too low for the models to be useful. Therefore, we tried to find another model that we can use to make predictions. The next algorithm that we tested was called light gradient-boosting, which is based on decision trees and implements boosting. This algorithm can automatically work with imbalanced data, so we don’t need to do resampling. After fitting, we evaluated the performance using the metrics.

Using this algorithm, instead of directly predicting whether the data would be classified as fraud or not, we decided to apply a threshold to the predicted probability. For example, if the probability that the data is considered fraud is higher than the threshold, we will classify it as a fraud. Tuning this threshold resulted in better precision and recall results

7. Analyze Column Age Significance

After fitting all these models, we came to the conclusion that the LightGBM model has the best prediction results. In our final stage of analysis, we will compare the accuracy scores with and without the AgeOfPolicyHolder column in our dataset. Here we will determine if the column plays a significant role in the case of determining fraud. We will reach a conclusion based on whether or not the validation score exceeds the rate if all columns contributed equally to the accuracy score. This would be 1/33 or 3.03%.

4. Results

After doing Chi-Squared tests the resulting p-value is summarized in the table below.

Column	P-Value
Month	0.0017
WeekOfMonth	0.654
DayOfWeek	0.1184
Make	<0.001
AccidentArea	<0.001
DayOfWeekClaimed	0.64
MonthClaimed	<0.001
WeekOfMonth-Claimed	0.498
Sex	<0.001
MaritalStatus	0.798

Column	P-Value
Age	<0.001
Fault	<0.001
PolicyType	<0.001
VehicleCat	<0.001
VehiclePrice	<0.001
Deductible	<0.001
DriverRating	0.648
Days_Policy_ Accident	0.021
Days_Policy_ Claim	0.181
PastNumberOf-Claims	<0.001

Column	P-Value
AgeOfVehicle	0.003
AgeOfPolicyHolder	<0.001
PoliceReportFiled	0.06
WitnessPresent	0.439
NumberOf-Suppliments	<0.001
AddressChange	<0.001
NumberOfCars	0.66
Year	0.008
BasePolicy	<0.001

Based on the result above, not all columns affect the categorization of fraudulent claims. The columns whose p-values are large indicates that the column and the fraud status are independent. If we focus on the column of interest, AgeOfPolicyHolder, the p-value is very small, which might confirm its significance in making fraud predictions. However, we still need to do further analysis.

The tables below summarizes the model scores using the oversampled and undersampled datasets.

Fraud Accuracy (Oversampled)				
Algorithms	Validation Score	Precision	Recall	F-1 Score
Logistic Regression	0.631	0.13	0.87	0.24
Naive Bayesian	0.633	0.14	0.88	0.24
K-Nearest Neighbors	0.736	0.07	0.25	0.11
Decision Tree	0.931	0.45	0.37	0.38
Random Forest	0.923	0.38	0.32	0.33
Boosting	0.955	0.75	0.44	0.57

Fraud Accuracy (Undersampled)				
Algorithms	Validation Score	Precision	Recall	F-1 Score
Logistic Regression	0.630	0.13	0.88	0.23
Naive Bayesian	0.626	0.13	0.88	0.23
K-Nearest Neighbors	0.661	0.07	0.32	0.11
Decision Tree	0.690	0.13	0.70	0.24
Random Forest	0.610	0.14	0.96	0.24
Boosting	0.817	0.25	0.96	0.38

The validation scores above are quite high, which might indicate a good model. However, since the data is imbalanced, it is likely that the model made a lot of non-fraudulent predictions, which will still generate high accuracy scores. When we look at the precision and recall values, they are very low, indicating the models that we have are not useful to be used to predict fraudulent claims.

Since none of the above models resulted in good enough precision and recall values, it seemed like resampling methods are not sufficient to address the data imbalance problem. Therefore, we decided to find another method that can combine some of the models above and address the imbalanced dataset issue, possibly from outside the sci-kit package. The LightGBM algorithm combines the boosting method with decision trees and can take imbalanced data to train it. The table below shows the scores that we obtained using this algorithm on the testing set.

The precision and recall values are much better than using the previous models. Most importantly, they are both equally relatively high. Therefore, this is the best model to be used to detect fraudulent vehicle insurance claims and the one we used to determine the role of the AgeOfPolicyHolder column.

Fraud Accuracy (Includes AgeOfPolicyHolder Column)				
Algorithms	Validation Score	Precision	Recall	F-1 Score
Light Gradient Boosting	0.937	0.55	0.86	0.74

Our last analysis involves dropping the AgeOfPolicyHolder column. Here are the results of the Light Gradient Boosting model after the column was dropped

Fraud Accuracy (No AgeOfPolicyHolder column)				
Algorithms	Validation Score	Precision	Recall	F-1 Score
Light Gradient Boosting	0.934	0.53	0.84	0.64

After removal of the AgeOfPolicyHolder column, the validation score of the LGBM model dropped by 0.3%, precision score dropped by 2% and recall by 2%. These drops can be considered small.

5. Conclusions

After running both models, the difference between the validation score of the dataset with age, and the one without age is 0.3%. There is also little difference in the precision and recall scores, which correlates to the lack of movement in the validation score. Therefore, we can conclude that age is not significantly more important in determining fraud compared to other columns.

The ratio of fraud and non-fraud data in the dataset resulted in a highly imbalanced dataset that was difficult to work with. Models would have good validations scores, but their fraud prediction accuracy would be poor, making their scores misleading. We attempted to mitigate this problem by using resampled data to train the models, but were still unsatisfied with the results. The resampling models exhibited a tendency to perform poorly in detecting fraud; however, the claims they did flag as fraudulent tended to indeed be fraudulent.

Subsequently, we explored alternative algorithms to yield more favorable outcomes. Among these, the light gradient boosting model similarly displayed this inclination, yet through parameter tuning, we managed to lower the threshold for fraud detection. This also had the effect of decreasing the detection accuracy, but we believe we found a good tradeoff, between detection, and accuracy.

6. Limitations

There are some challenges regarding the imbalanced data that we have. With only around 6% of the data being fraudulent cases, most machine learning models would tend to bias predicting the non-fraud cases, resulting in misleading accuracy. When we tried oversampling and undersampling the data set, we expected the models to give better precision and recall values. However, looking for other models and algorithms helped build a better prediction machine.

If we had more time, we would like to address these imbalanced data issues. First, by obtaining more data, if possible. Another way to address this issue is by trying to engineer some features based on the columns that we have. Out of the 33 explanatory features, we can probably do more in-depth analyses regarding their relationships and come up with a feature that can be more useful.

References

Dataset: https://www.kaggle.com/datasets/shivamb/vehicle-claim-fraud-detection

Brownlee, Jason. "8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset - MachineLearningMastery.com." *Machine Learning Mastery*, 19 August 2015, https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/. Accessed 4 August 2023.

"Insurance." *Wikipedia*, https://en.wikipedia.org/wiki/Insurance. Accessed 4 August 2023.

"LightGBM Documentation." *Welcome to LightGBM's documentation! — LightGBM 4.0.0 documentation*, https://lightgbm.readthedocs.io/en/stable/index.html. Accessed 4 August 2023.

"LightGBM (Light Gradient Boosting Machine)." *GeeksforGeeks*, 23 May 2023, https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/. Accessed 4 August 2023.

Morales, Pedro. "Column Transformer with Mixed Types — scikit-learn 1.3.0 documentation." *Scikit-learn*, https://scikit-learn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html. Accessed 4 August 2023.