Data Mining and Predictive Analytics
(BUDT758T)

Devising Predictive Models to Identify Potential Vehicle Insurance Fraud

Team Members:

Ying Chai, Jingping Guo, Jing Lin, Yifei Li, Shenger Zhang

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

	Typed Name	Signature
Contact Author	Shenger Zhang	*Shenger Zhang*
	Ying Chai	*Ying Chai*
	Jingping Guo	*Jingping Guo*
	Jing Lin	*Jing Lin*
	Yifei Li	*Yifei Li*

1. Executive Summary

1)   Background

According to the Coalition Against Insurance Fraud, "Insurance fraud is one of the largest crimes in the United States - at least $80 billion is stolen each year." Among those insurance fraud issues, auto insurance fraud components are one of the largest proportions of insurance loss. According to a study conducted by Versik, a data and analytics firm, it shows that fraud costs the auto insurance companies at least $29 billion annually. The Insurance Research Council (IRC) completed a study in which they estimated that paid claims for auto insurance personal injury benefits were fraudulently increased by $5.6 billion to $7.7 billion in 2012, compared to $4.3 billion to $5.8 billion in 2002.[1]

Maryland is one of the areas most affected by insurance fraud. A 2021 study completed by Clearsurance[2] using data from the Federal Trade Commission (FTC) found that Maryland ranked sixth in the United States for the most auto fraud and scams, with 1,893 cases of auto fraud occurring annually, not to mention those cases that haven't been detected. The study estimates that "Maryland has 313 auto-related cases per million Maryland residents."

Thus, in the insurance industry, detecting fraud claims has long been a topic of exploration.

2)   Report Summary

This report is based on 12,643 auto insurance fraud data in the oracle database for supervised insurance fraud prediction model training, aiming at predicting possible insurance fraud by analyzing user profiles, insurance policies, accident types, etc. This report will pre-process the data by exploring the data and selecting meaningful features. We will compare the results of logistic regression, Random Forest, Classification Trees and Xg Boost algorithms. Ultimately, the optimal model will be selected by comparing the accuracy and sensitivity indicators of the test value.

---

[1] https://www.iii.org/article/background-on-insurance-fraud

[2] https://clearsurance.com/blog/worst-states-for-auto-related-fraud

2. Data Description

We use the data from Kaggle data <u>Vehicle Insurance Claim Fraud Detection</u>, the original data is provided by Oracle. The data consists of 12643 lines of claims and the fraud label with other 32 variables. The type of variables and detailed descriptions are listed below (**Appendix**).

```
> glimpse(df)
Rows: 15,420
Columns: 33
$ Month                <chr> "Dec", "Jan", "Oct", "Jun", "Jan", "Oct", "Feb", "Nov", "Dec", "Apr", "Mar", …
$ WeekOfMonth          <int> 5, 3, 5, 2, 5, 4, 1, 1, 4, 3, 2, 5, 3, 5, 5, 4, 4, 5, 4, 4, 2, 2, 3, 3, 3, 3,…
$ DayOfWeek            <chr> "Wednesday", "Wednesday", "Friday", "Saturday", "Monday", "Friday", "Saturday…
$ Make                 <chr> "Honda", "Honda", "Honda", "Toyota", "Honda", "Honda", "Honda", "Honda", "Hon…
$ AccidentArea         <chr> "Urban", "Urban", "Urban", "Rural", "Urban", "Urban", "Urban", "Urban", "Urba…
$ DayOfWeekClaimed     <chr> "Tuesday", "Monday", "Thursday", "Friday", "Tuesday", "Wednesday", "Monday", …
$ MonthClaimed         <chr> "Jan", "Jan", "Nov", "Jul", "Feb", "Nov", "Feb", "Mar", "Dec", "Apr", "Mar", …
$ WeekOfMonthClaimed   <int> 1, 4, 2, 1, 2, 1, 3, 4, 5, 3, 3, 5, 3, 1, 1, 5, 1, 1, 5, 1, 1, 2, 5, 3, 3, 1,…
$ Sex                  <chr> "Female", "Male", "Male", "Male", "Female", "Male", "Male", "Male", "Male", "…
$ MaritalStatus        <chr> "Single", "Single", "Married", "Married", "Single", "Single", "Married", "Sin…
$ Age                  <int> 21, 34, 47, 65, 27, 20, 36, 0, 30, 42, 71, 52, 28, 0, 61, 38, 41, 28, 32, 30,…
$ Fault                <chr> "Policy Holder", "Policy Holder", "Policy Holder", "Third Party", "Third Part…
$ PolicyType           <chr> "Sport - Liability", "Sport - Collision", "Sport - Collision", "Sedan - Liabi…
$ VehicleCategory      <chr> "Sport", "Sport", "Sport", "Sport", "Sport", "Sport", "Sport", "Sport", "Spor…
$ VehiclePrice         <chr> "more than 69000", "more than 69000", "more than 69000", "20000 to 29000", "m…
$ PolicyNumber         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22…
$ RepNumber            <int> 12, 15, 7, 4, 3, 12, 14, 1, 7, 7, 7, 13, 11, 12, 3, 16, 15, 6, 6, 2, 3, 13, 8…
$ Deductible           <int> 300, 400, 400, 400, 400, 400, 400, 400, 400, 400, 400, 400, 400, 400, 400, 40…
$ DriverRating         <int> 1, 4, 3, 2, 1, 3, 1, 4, 4, 1, 3, 1, 1, 3, 1, 1, 4, 1, 1, 2, 1, 2, 3, 3, 3, 4,…
$ Days_Policy_Accident <chr> "more than 30", "more than 30", "more than 30", "more than 30", "more than 30…
$ Days_Policy_Claim    <chr> "more than 30", "more than 30", "more than 30", "more than 30", "more than 30…
$ PastNumberOfClaims   <chr> "none", "none", "1", "1", "none", "none", "1", "1", "none", "2 to 4", "none",…
$ AgeOfVehicle         <chr> "3 years", "6 years", "7 years", "more than 7", "5 years", "5 years", "7 year…
$ AgeOfPolicyHolder    <chr> "26 to 30", "31 to 35", "41 to 50", "51 to 65", "31 to 35", "21 to 25", "36 t…
$ PoliceReportFiled    <chr> "No", "Yes", "No", "Yes", "No", "No", "No", "No", "No", "No", "No", "No", "No…
$ WitnessPresent       <chr> "No", "No", "No", "No", "No", "No", "No", "No", "Yes", "No", "No", "No", "No"…
$ AgentType            <chr> "External", "External", "External", "External", "External", "External", "Exte…
$ NumberOfSuppliments  <chr> "none", "none", "none", "more than 5", "none", "3 to 5", "1 to 2", "none", "3…
$ AddressChange_Claim  <chr> "1 year", "no change", "no change", "no change", "no change", "no change", "n…
$ NumberOfCars         <chr> "3 to 4", "1 vehicle", "1 vehicle", "1 vehicle", "1 vehicle", "1 vehicle", "1…
$ Year                 <int> 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994, 1994,…
$ BasePolicy           <chr> "Liability", "Collision", "Collision", "Liability", "Collision", "Collision",…
$ FraudFound_P         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
```

3. Research Questions

Regarding the technology to detect insurance fraud, there's a lack of confidence in organizations' tools, resources, and knowledge to combat and manage globalized insurance fraud. The Coalition estimates that 43% of the insurance agents express they are "Somewhat confident" while only 21.2% of organizations are "Very confident" and 4.1% are "Extremely confident" to detect insurance fraud. Among all the technologies, automated red flags (88%), and predictive modeling (80%) are the tools that are mostly used to detect insurance fraud.

We use the insurance claims data provided by oracle with the intention of simulating the real situation of insurance companies in the daily claims process, and the data contains all the information that can be accessed during the claims process. We want to filter the features and find out the metrics that can improve the insurance fraud screening rate through the algorithm.

The success of the model will solve the following problems for the insurance industry in screening insurance fraud.

1)   Accuracy of Detecting

We will select the optimal algorithm for predicting the likely insurance fraud claims. Reducing the probability of fraud claims will not only reduce the non-operating expenses of insurance companies but also reduce the amount of consumer co-payments for that expense and reduce the overhead of insurance rates.

2)   Data Abuse

By analyzing the coefficients of each indicator in the algorithm, we will explore the indicators that are under-correlated with the model predictions. Reducing the collection of data on this type of indicator will allow us to determine the reasonableness and authenticity of claims with minimal data, and maximize the privacy of customers

3)   Data Complementation

We will use the algorithm to find the indicators with the highest correlation with insurance fraud and perform cluster analysis. As a single category of data contains a very large amount of data, the analysis of a certain type of data can improve the accuracy and efficiency of data collection

4. Data Preparation and Exploration

1)   Basic Process

Data preparation is the first step to processing data in order to run the model successfully. Firstly, we use str() the structure of each variable, we found the majority of features are character type, and we change all character features to factor type since those variables have different levels, like Fault, Sex, PolicyType, and factor type is more suitable for running models. Then, we check the missing value and invalid data.

```
> colMeans(is.na(df))
            Month         WeekOfMonth          DayOfWeek               Make         AccidentArea   DayOfWeekClaimed
                0                   0                  0                  0                    0                  0
      MonthClaimed    WeekOfMonthClaimed                Sex       MaritalStatus                Age              Fault
                0                   0                  0                  0                    0                  0
        PolicyType      VehicleCategory       VehiclePrice        PolicyNumber          RepNumber         Deductible
                0                   0                  0                  0                    0                  0
      DriverRating  Days_Policy_Accident    Days_Policy_Claim   PastNumberOfClaims        AgeOfVehicle   AgeOfPolicyHolder
                0                   0                  0                  0                    0                  0
  PoliceReportFiled       WitnessPresent          AgentType    NumberOfSuppliments  AddressChange_Claim       NumberOfCars
                0                   0                  0                  0                    0                  0
              Year           BasePolicy        FraudFound_P
                0                   0                  0
```

Fortunately, there is no missing value in our dataset, but there are some 0 values in features, DayOfWeekClaimed, MonthClaimed, and Age. We delete the DayOfWeekClaimed = 0 and MonthClaimed = 0 since there is only 1 record and occurs at the same record. As for the Age, there are 320 records indicating the Age = 0, we notice all 320 records are assigned to group '16 to 17' of 'AgeOfPolicyHolder', based on this information, we replace Age 0 to 16.5.

```
The number of records is 0 in DayOfWeekClaimed: 1
The number of records is 0 in MonthClaimed: 1
The number of records is 0 in Age: 320
```

```
The number of records is 0 in DayOfWeekClaimed: 0
The number of records is 0 in MonthClaimed: 0
The number of records is 0 in Age: 0
```

2)   Imbalanced Dataset Handling

The target variable is FraudFound_P with two classes: 1 and 0 represent the claim was fraudulent (1) or not (0), respectively. Plotting a class distribution to check whether the data present an imbalance. There are only 14496/15420 are not fraud records (94%), and only 6% are fraudulent. If we use imbalanced data to build models directly, we could get extremely high accuracy, but it does not represent that is a good result since we still need to consider the other performance metrics like Specificity and Sensitivity for imbalanced data. We could not use raw data to run machine learning models, which will cause bias toward the majority class. This is because most machine learning is designed for balanced data. Before doing this change, we need to split the dataset into train and test, this change is only for training purposes. For testing data, it keeps the imbalanced data since balanced data is rare when we get data in the real world, using imbalanced data as testing data is better to test model performance.

Class Distribution
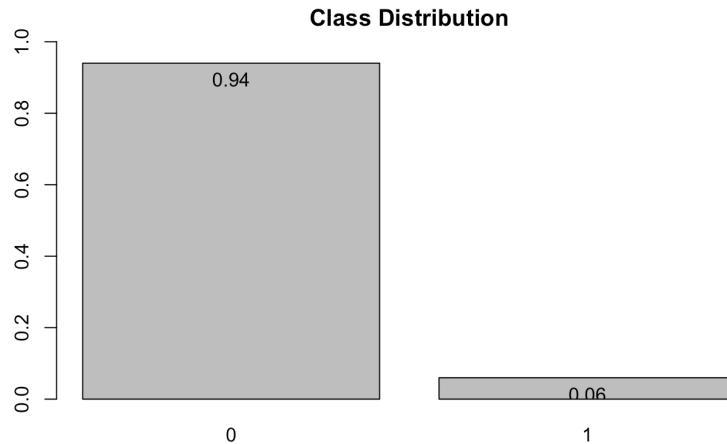
Therefore, we choose to use an upsampling method to deal with imbalanced data, through duplicating the minority class many times to achieve the counts of both labels are almost the same. After balancing the training dataset, the below picture shows that labels 0 and 1 almost have equal weight in the training dataset. Then, our dataset is ready to build different models.

Class Distribution
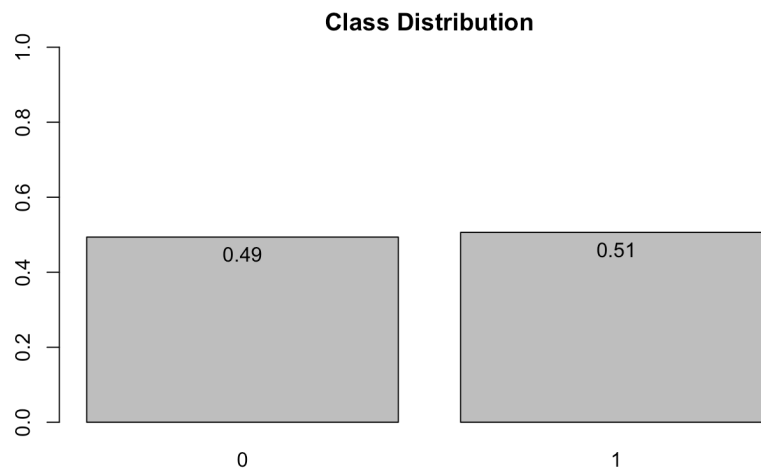
5. Methodology

The algorithms we use are:

logistic regression

Instead of using Y (or p) as the dependent variable, we use a function of it, which is called the logit. We will use the logistic function instead of a linear probability function to estimate the probability of fraud.

Classification Trees

Classification trees, a method for classification and profiling. It is simple interpretation and can create understandable rules. Overall impurity is the weighted average of the impurity of sub-nodes created by the split. The algorithm picks a split to minimize overall impurity

Random Forest

Ensemble methods combine the results from multiple models with the goal of improving prediction accuracy. The random forest algorithm averages many classification trees, where each is constructed using a random subset of the variables for each split in the tree.

Boosting

Boosting works on the same dataset (does not resample like Bagging), instead it "upweights" misclassified data points. Boosting has been called the "best off-the-shelf classifier in the world". Boosting usually gives zero training error, but rarely overfits.

Xg Boost

A variant of boosting popular these days is gradient boosting. XGBoost uses the second-order gradient as an approximation to the residuals, closer to the true value, faster convergence.

6. Results and Finding (varies considerably in length depending on study)

Logistic Regression

The logistic regression model is one of the simplest classification model, and sometimes it's easy for it to get distracted when the data is not preprocessed well. As mentioned earlier, the data sample for this problem is quite unbalanced, containing too much observation in class 0 and too little in class 1, while class 1 is the most important class we want to focus on. So we tried the upsample method, and apparently it worked well for the logistic regression model, and worked way better than some "more intelligent" models such as random forest and XGBoost.

Concerning this, we tried out three upsampling ways to process the data for logistic model: (a) to let it remain unbalanced, (b) to upsample the class 1 observations until both classes have pretty much the same proportion in all observations, and (c) to upsample the class 1 observations until

they're as many as half of the number of class 0. The reason we did in the last way in to prevent both the unbalance issue and over-weighting some repeated features.

In R, we used the glm() function to fit the model, and used all features as predictors. Take an example of way c as stated above, the accuracy of logistic regression turned out to be 0.65 and the specificity is 0.64, which are relatively low compared to other models. But when we looked at the sensitivity which is 0.83, we realized that the upsampling did help a lot in balancing the data and improving the model prediction in class 1 which is what we're interested in.

Classification Tree

Classification tree is one of the most commonly used and the easiest to understand classification models. I utilized the concept of impurity to classify the observations level by level, which means variable by variable. In this problem, we fit this model by using tree() function in "tree" library in R, used all features as classifier for the model itself to pick and set all parameters as default.

The model didn't stop classifying until it reached the end node, in which every single observation has been classified into a group that contains only itself. This is also known as a "full tree". While a full tree is more than accurate for a training set, it's not usually that case in a test set as it probably included random noise into it. To avoid this problem, we pruned the tree into a size in which the model has a relatively low standard deviation, and in which even if increasing the size, the standard deviation wouldn't decrease much. We did the pruning by using cv.tree() function in R, and got the tree plot as following:
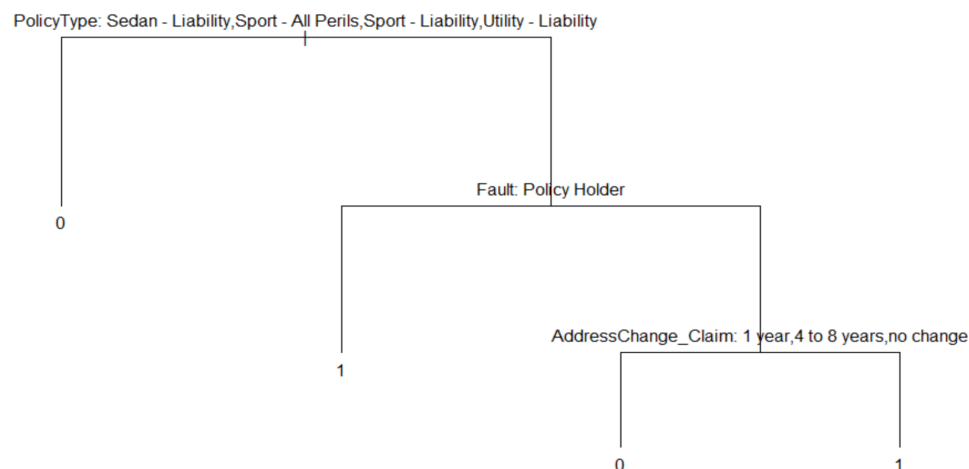
In terms of the tree performance, we got a 0.6 accuracy and 0.58 specificity, which is okay, while the sensitivity turned out to be 0.96 which is quite impressive and the highest sensitivity we got so far among all models. The results are similar to that of logistic regression above. Therefore, in general, if our client decides that to identify the fraud is their most important objective and they're willing to sacrifice some accuracy in identifying class 0, this model is one of the best.

Random Forest

Random forest is one of the most "intelligent" ensemble algorithms, it can automatically avoid the little issues the data might have and consider all related features while maintaining randomness, which means it does a great job in avoiding overfitting. So for fitting random forest algorithm, we used randomForest() function in randomForest library in R, used all features and set the mtry as 6, which defines the number of variables randomly sampled as candidates at each split.

Different from the two models mentioned above, random forest has a great performance overall. In terms of overall performance metrics, it reached 0.94 accuracy and 0.998 specificity. It indicates that for most samples, especially those samples that are actually in class 0 (Not a fraud), this model did a great job. But when coming to identifying those actual class 1, it did terrible as the sensitivity metric is 0.05, meaning that 95% of samples that are actually class 1 (Fraud) are classified as class 0 (Not a fraud). We assumed that it probably has something to do with the upsampling method we used, which is duplicating the class 1 observations to increase its proportion. This method is probably so simple that it couldn't work well for random forest algorithm.

Additionally, in randomForest() function, we also set the parameter "importance" as true, in this way we can get a plot that shows the importance levels of all variables back. The plot shows as following:

bag.fraud
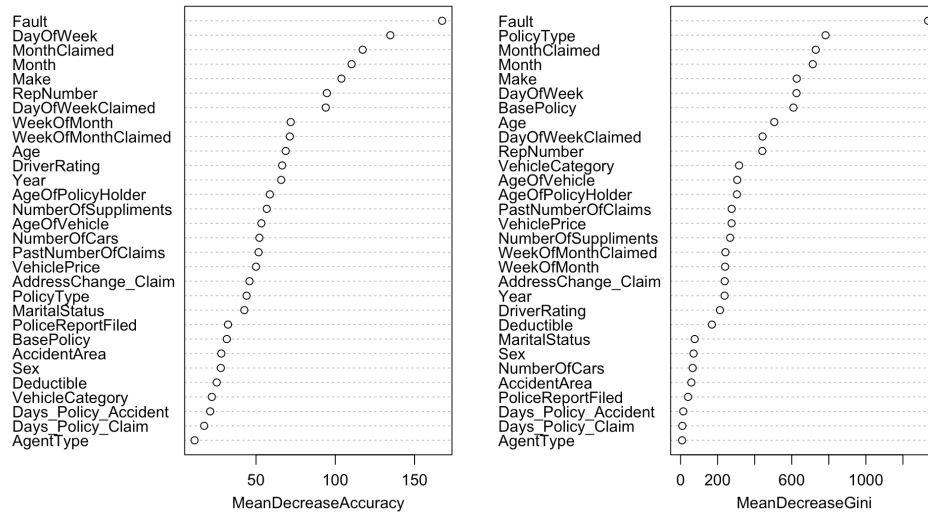
From the plot above, we can see that there are a few variables that are obviously more important than others. In other words, those features, such as "Fault", "DayOfWeek", and "Month Claimed", most likely determine whether the data sample is a fraud or not. This plot gave us basic ideas of a whole picture, and more importantly, it helped when we do the subset selection, trying to remove those features that is sort of "useless" out of our model.

Boosting

The boost model's test accuracy rate is 0.9258, and the test specificity is 0.9710. But the test sensitivity is only 0.2051, which means the true positive rate is low. The model's ability to correctly generate fraud for the people who have the condition is poor. In reality, if the benefit of selling insurance is more than the loss in fraud reports, the boosting model is still useful because the true negative is high. However, if the goal of the insurance company is decreasing risks, the boosting model fails.

According to the importance of variables report, PolicyType, Fault(who was deemed at fault), Month, Monthclaimed, and Make are the top five important variables, whereas the Days_Policy_Claim, WitnessPrensent, and AgentType are the least important variables. In this case, the insurance company can focus on investigating which policy type has more fraud claim reports, and which months when the fraud claim reports easily happen. Then the next step is to

adopt some methods, like inspecting the application more carefully in the certain policy type and increasing the fees in certain car Make.
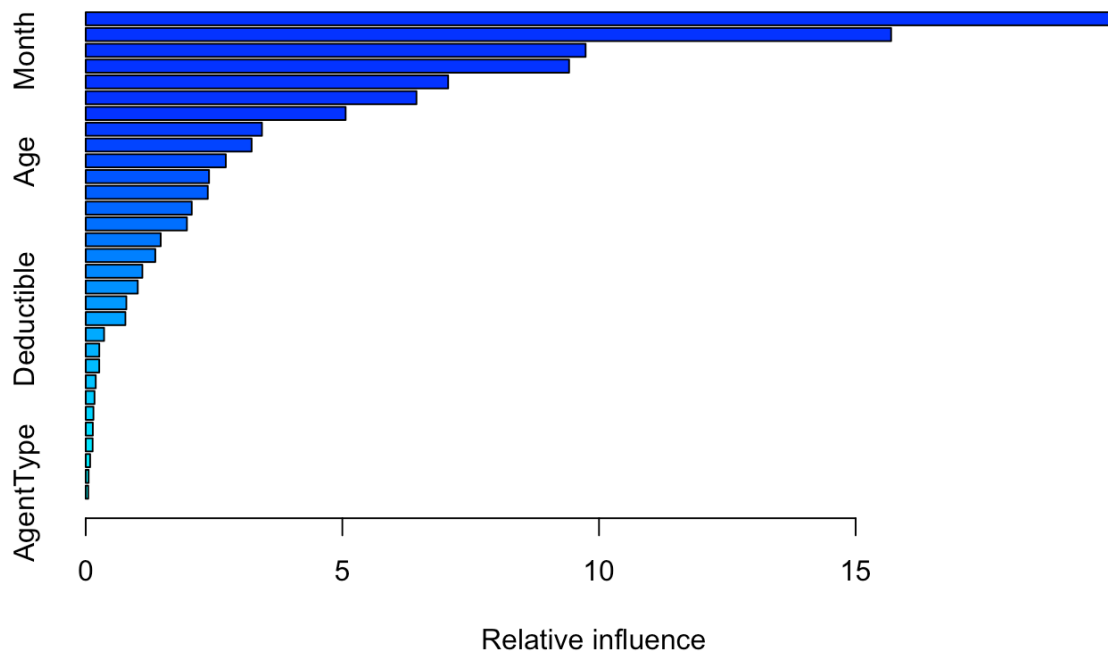
XGBoost

The binary:logistic is used in the xgboost. The xgboost model's test accuracy rate is 0.6595, and the test specificity is 0.6461. The test sensitivity is 0.8718. The specificity is relatively low, which means the model's ability to correctly generate a negative result for people who do not have the condition that is being tested for is relatively poor. However, if the insurance company is at risk aversion, the xgboost model is helpful because the sensitivity is high and it could predict the true fraud well.

According to the bar plot that shows the importance of variables, Fault_Policy Holder, Basepolicy_Liability, AddressChange_Claim_2 to3 years, Age and Year are the top five important variables. In this case, the company should put more effort to check the driver in the accident is the policyholder, the different coverage of different policies, address status, and ages of individuals making claims, if the insurance company focuses on avoiding risks.

XG Boosted Feature Importance (Top 5)
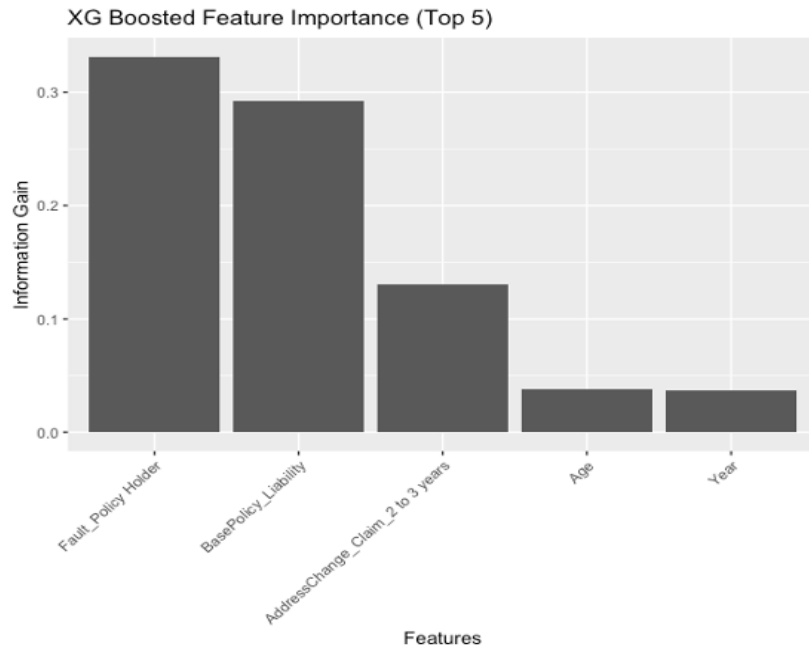
Model selection:

Model	Upsample Ratio: 1 : 1 (class 0 vs class 1)			
	Accuracy	**Specificity**	**Sensitivity**	**AUC**
Logistic	0.6484	0.6369	0.8315	0.779
Class Full tree	0.6024	0.5797	0.9634	
Pruned tree	0.6024	0.5797	**0.9634**	0.772
Random Forest	**0.9418**	0.9977	0.0512	0.524
Boosting	0.9258	0.9710	0.2051	0.815
XGboost	0.6595	0.6461	0.8718	0.820
Sensitivity best:	Classification tree			
Accuracybest:	Random Forest			
Overall:	XGBoost			

Accuracy, Specificity, Sensitivity, and Auc are important measurement indicators to indicate model performance, which we mentioned above. Then, we make a matrix for integrating all those measurement values of each model to compare which model performs best. Let us give those values into business interpretation. Accuracy is the overall performance of the model, representing the proportion of fraud in all classes. Sensitivity indicates the ability of a model to detect the vehicle Insurance Fraud correctly. Specificity is the ability to rule out legal vehicle insurance claims correctly. Pruned Classification tree has the highest Sensitivity 96.34%, Random Forest has the highest accuracy 94.18%, the overall performance of the XGBoost modeL is better, with high sensitivity and average performance on accuracy and specificity. AUC is another key criterion for selecting a classifier, higher AUC indicates better performance, we could see Boosting and XGBoost have good performance. We also visualize the ROC curve to illustrate AUC value.

AUC Range	Classification
0.9 < AUC < 1.0	Excellent
0.8 < AUC < 0.9	Good
0.7 < AUC < 0.8	Worthless
0.6 < AUC < 0.7	Not good

ROC chart
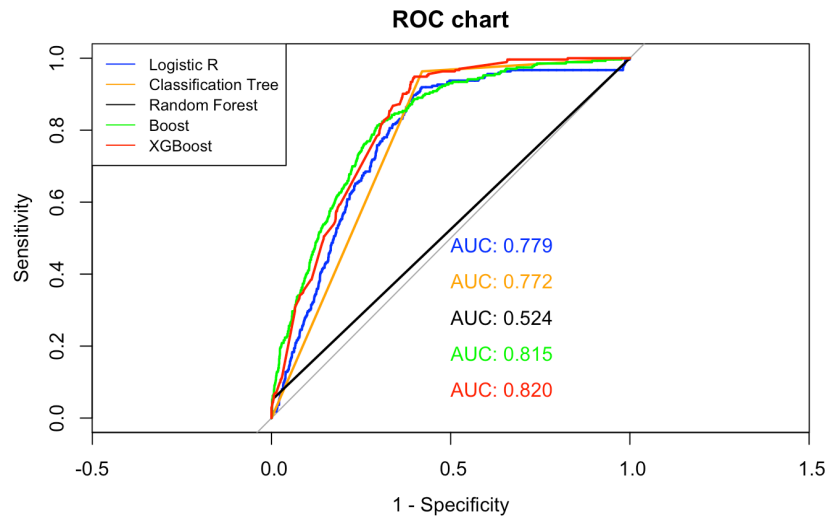
7. Conclusion

In conclusion, through building the predictive models and analyzing the results, we did get some outcomes and new insights regarding helping identify the auto insurance fraud for our client.

First of all, in order to pick the "best" predictive model, we first need to know for certain our client's need, whether it's to identify the most real fraud (i.e. the highest sensitivity), or it's to classify the most cases correctly in general (i.e. the highest accuracy), or to find the trade-off between these two. To be more specific, if the client wants the best fraud identifier, we recommend the classification tree model with upsample ratio of 1:1; if the client wants the best overall model (regardless of fraud or not), we recommend the random forest model with upsample ratio of 1:1; and if they wants both, we'd say the XGBoost model is the best.

Second of all, in addition to finding the best model, we also figured out some most important features that probably affect whether a claim is fraud or not. We did this by making an importance level plot in randomForest package, and it shows that Fault (whether the driver who claimed is

deemed at fault), which day of week, which month it's claimed, the policy type, the make of car are some of those most important elements.

Last but not least, while we did solve the problems we're interested in, there is still something that we can keep improving in the future. For instance, when sampling the data to solve the unbalanced issue, maybe we can figure out some better ways to do it instead of simply duplicating the data. Besides, in this case, we have more than 30 features to deal with, and apparently some of them have some sort of correlation issue, in other words, they're not independent from one another, which might decrease the validity of the model. So we should try to decrease the number of variables in a better way or combine some of them, or decrease the levels of some categories. And at last, there's a more general question remained: is there a better way to trade off the accuracy and sensitivity in this case? We might need an even more efficient model to solve this.

Reference:

https://www.lexjansen.com/nesug/nesug10/hl/hl07.pdf

https://www.iii.org/article/background-on-insurance-fraud

https://clearsurance.com/blog/worst-states-for-auto-related-fraud

https://www.kaggle.com/code/jwilda3/classifying-fraud-by-decision-trees

Appendix

Name	Data Type	Description
Month	object	these are the months in which the accident occured
WeekOfMonth	int64	provides the week in the month the accident occured
DayOfWeek	object	these are the days of the week the accident occured on
Make	object	contains a list of 19 car manufacturers
AccidentArea	object	classifies area for accident as "Urban" or "Rural"
DayOfWeekClaimed	object	contains '0' - need to check how many of these there are and see about "fixing" - missing data
MonthClaimed	object	contains '0' - need to check how many there are and what they mean - missing data
WeekOfMonthClaimed	int64	contains weeks in the month that the claimed in field
Sex	object	gender of individual making claim- binary data, convert to 1 or 0
MaritalStatus	object	marital status of individual making claim
Age	int64	ages of individual making claim there is at least one individual with age 0 - missing data
Fault	object	categorization of who was deemed at fault. convert to binary, 1 or 0
PolicyType	object	contains two pieces of info - the type of insurance on the car - liability, all perils, collision category of the vehicle - sport, sedan, utility
VehicleCategory	object	contains the categorization of the vehicle (see PolicyType)
VehiclePrice	object	contains ranges for the value of the vehicle replace ranges with mean value of range and convert to float
FraudFound_P	int64	indicates whether the claim was fraudulent (1) or not (0) this is what we want to predict
PolicyNumber	int64	the masked policy number, appears to be the same as row number minus 1
RepNumber	int64	rep number is integer from 1 - 16

Deductible	int64	the deductible amount-integer values
DriverRating	int64	the scale is 1, 2, 3, 4 the name DriverRating implies the data is ordinal, but is it interval as well
Days_Policy_Accident	object	as a guess, this is the number of days between when the policy was purchased and the accident occured each value is again a range of values change these to be mean of the range and make float
Days_Policy_Claim	object	another guess, this is the number of days that pass between the policy was purchased and the claim was filed-each value is a range change these to be the mean of the ranges and make float
PastNumberOfClaims	object	previous number of claims filed by policy holder (or claimant?)
AgeOfVehicle	object	represents the age of vehicle at time of the accident? each value is a range of years change these to be the mean of the ranges and make float
AgeOfPolicyHolder	object	each value is a range of ages"-change these to be the mean of the ranges and make float
PoliceReportFiled	object	indicates whether a police report was filed for the accident - convert to binary
WitnessPresent	object	indicted whether a witness was present convert to binary
AgentType	object	this classifies an agent who is handling the claim as internal vs external. What does this mean? change to binary
Number Of Supplements	object	probably not the number of vitamins taken daily not sure what a supplement is in insurance
NumberOfCars	object	guess, number of cars involved in accident OR number of cars covered under policy. replace each interval with mean value of range
AddressChange_Claim	object	guess, time from claim was filled to when person moved (i.e. filed an address change) replace each interval with mean value of range
Year	int64	guess, year accident occured
BasePolicy	object	type of insurance coverage (see PolicyType)