Fraud Claim Detection for Auto Insurance

Shaozun Chen; Wenpeng Hao; Dohyung Lee; Mohammad Ridwan (Iwan); Long Zhang

1. Introduction

Insurance fraud can be defined as "Intentionally making a false claim, inflating a claim, adding extra items to a claim, or being dishonest in any way with the intention to gain more than legitimate entitlement" (Bologa, Bologa, & Florea, 2013). Specifically, claims fraud and excessive loss amounts have been increasing as a major concern among automobile insurance companies (Sharon, Pau, 2002). According to the FBI reports, the total cost of insurance fraud (non-health insurance) is more than 40 billion dollars per year. In addition, the Insurance Fraud costs the average U.S. family are between 400 and 700 dollars per year in the form of increased premiums. The trends of fraudulently added to paid claims for auto insurance bodily injury payments has been increasing between 5.6 billion dollars and 7.7 billion dollars in 2012, compared with a range of 4.3 billion dollars to 5.8 billion dollars in 2002 (Insurance Research Council, 2015). Auto insurance fraud claims might cause wasting a tremendous number of assets and bringing much potential financial risk to insurance companies.

The goal of this project is applying predictive analytics and data mining methods to classify abnormal claims and guide the business decision-making with practical results under these circumstances. To accomplish our aims, we will analyze the Auto Insurance Claims Data, which originated from Kaggle (Bunty Shah, 2018). This dataset has one binary response variable and 37 predictors, including 3 auto features, 8 customer features, 16 incident features, and ten policy features.

To this end, we focus on a classification problem; "How to predict whether an auto insurance claim is a fraud or not?" To accomplish the problem, we might create a classifier with high performance, find out the most essential factors for auto insurance fraud claim detection, and finally provide decision-makers with valuable suggestions against the question.

The process of analysis will proceed as: identify all predictors and prepare the entire dataset with data preprocessing; analyze datasets (i.e., original dataset, up-sampling dataset) with three different categories of classification models, including linear classification models, nonlinear classification models, and tree-based models; evaluate each model's performance; conclude a practical model which has the best performance.

2. Data Preparation

2.1. Dataset Introduction

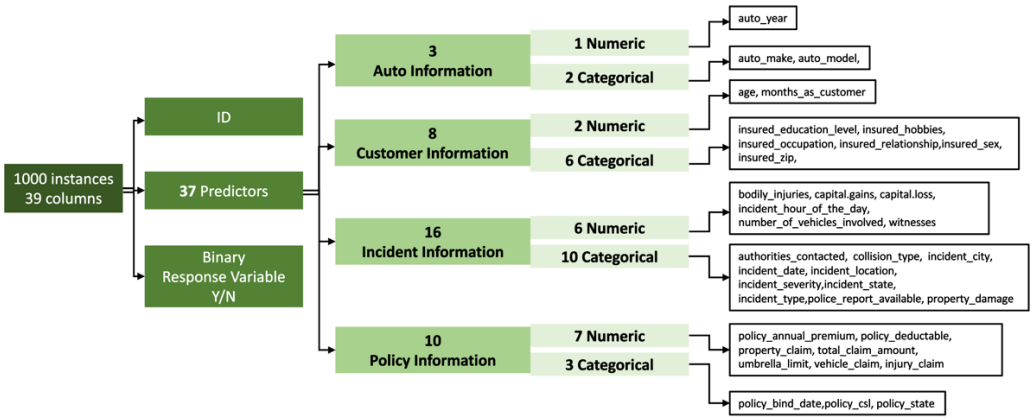


Fig 2.1. Raw Data Information

The dataset “Auto Insurance Claims Data” is collected from Kaggle (Shah, 2018). The data structure is shown in Fig 2.1. which includes the specific names of 37 predictors of the raw data. And the binary response variable is whether a claim is fraud (Y) or not (N).

2.2. Selection of Predictors

The first step is to generate new variables from raw data based on the industry knowledge. The results are shown in Table 2.1. And then, we dropped categorical variables that have too many levels, like **policy bind date**, **insured zip**, **incident location**, and **policy number**. They have almost 1000 levels. Besides, as for variable **incident date**, it has 60 levels including the dates of the first two month in 2015. We dropped it because this interval cannot cover the dates of the whole year. It cannot make predictions for incidents in the other months. Finally, we got **36 predictors, including 19 numerical variables and 17 categorical variables.**

Name	Description	Derived From
days_after_bind	the number of days between policy bind date and incident date	• **incident_date**: the incident dates YYYY-MM-DD, 60 levels (two month) • **policy_bind_date**: the date when the policy is bound, 951 levels.
years_after_auto_year	the number of years between the production year of an auto and the incident year.	• **auto_year**: the year of an auto produced. • **incident_date**: the date of the incident
policy_bind_year	the year when the policy is bound	• **policy_bind_date:** the date when the policy is bound, 951/1000 levels.

Table 2.1. New Variables

2.3. Data Cleaning

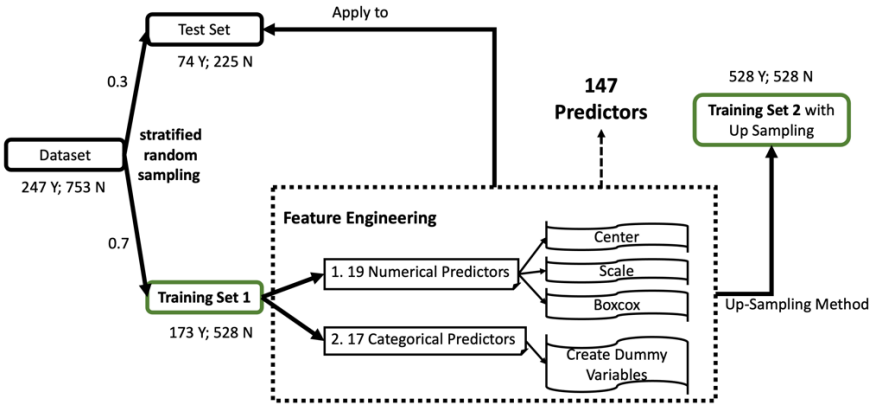


Fig 2.2. Data Cleaning Process

2.3.1. Missing Values

In this dataset, missing values are stored as a question mark. Table 2.2. displays the count of missing values by a descending order. We can see that 360 instances don't have the information about whether the property is damaged, 343 instances don't have the information about Whether the police report is available, and 178 instances don't have the information about collision type. In this project we keep the question mark as a level for the three categorical variables. This means that the answer for the above three questions is "not sure".

Predictors	Missing Values
property_damage	360
police_report_available	343
collision_type	178

Table 2.2. Missing Values

2.3.2. Near 0 Variance

In this dataset, there are not predictors with near 0 variance.

2.3.3. Data Splitting

We divided the data into training set and test set with a proportion of 0.7 using stratified random sampling. This is because the dataset is extremely imbalanced with 247 positive instances and 753 negative instances. After splitting, training set has 173 positive instances and 528 negative instances; test set has 24 positive instances and 225 negative instances.

2.3.4. Feature Engineering

As for the training set, we applied the feature engineering for 19 numerical predictors first, including center, scale, and BoxCox. And then we created dummy variable for 17 categorical predictors. After feature engineering, we got **147** predictors totally. The feature engineering method on training set is also applied to the test set.

2.3.5. Up-Sampling Dataset

In this project, to further Increase the prediction accuracy for the minority class samples, we applied up-sampling method on training set. Specifically, cases from the minority classes are sampled with replacement until each class has approximately the same number. After sampling, in the training set 2, both positive class and negative class have 528 instances.

2.3.6. Cleaned-Up Dataset Description

Finally, we obtained two training datasets. One is preprocessed with up-sampling method, and the other one is not transformed. They both have a binary response variable to indicate whether a claim is fraud (Y) or not (N). And they both have the following 147 predictors:

147 Predictors in the Cleaned-Up Datasets:

months_as_customer, age, policy_state.IN, policy_state.OH, policy_csl.250.500, policy_csl.500.1000, policy_deductable, policy_annual_premium, umbrella_limit, insured_sex.MALE, insured_education_level.College, insured_education_level.High.School, insured_education_level.JD, insured_education_level.Masters, insured_education_level.MD, insured_education_level.PhD, insured_occupation.armed.forces, insured_occupation.craft.repair, insured_occupation.exec.managerial, insured_occupation.farming.fishing, insured_occupation.handlers.cleaners, insured_occupation.machine.op.inspct, insured_occupation.other.service, insured_occupation.priv.house.serv, insured_occupation.prof.specialty, insured_occupation.protective.serv, insured_occupation.sales, insured_occupation.tech.support, insured_occupation.transport.moving, insured_hobbies.basketball, insured_hobbies.board.games, insured_hobbies.bungie.jumping, insured_hobbies.camping, insured_hobbies.chess, insured_hobbies.cross.fit, insured_hobbies.dancing, insured_hobbies.exercise, insured_hobbies.golf, insured_hobbies.hiking, insured_hobbies.kayaking, insured_hobbies.movies, insured_hobbies.paintball, insured_hobbies.polo, insured_hobbies.reading, insured_hobbies.skydiving, insured_hobbies.sleeping, insured_hobbies.video.games, insured_hobbies.yachting, insured_relationship.not.in.family, insured_relationship.other.relative, insured_relationship.own.child, insured_relationship.unmarried, insured_relationship.wife, capital.gains, capital.loss, incident_type.Parked.Car, incident_type.Single.Vehicle.Collision, incident_type.Vehicle.Theft, collision_type.Front.Collision, collision_type.Rear.Collision, collision_type.Side.Collision, incident_severity.Minor.Damage, incident_severity.Total.Loss, incident_severity.Trivial.Damage, authorities_contacted.Fire, authorities_contacted.None, authorities_contacted.Other, authorities_contacted.Police, incident_state.NY, incident_state.OH, incident_state.PA, incident_state.SC, incident_state.VA, incident_state.WV, incident_city.Columbus, incident_city.Hillsdale, incident_city.Northbend, incident_city.Northbrook, incident_city.Riverwood, incident_city.Springfield, incident_hour_of_the_day, number_of_vehicles_involved, property_damage.NO, property_damage.YES, bodily_injuries, witnesses, police_report_available.NO, police_report_available.YES, total_claim_amount, injury_claim, property_claim, vehicle_claim, auto_make.Audi, auto_make.BMW, auto_make.Chevrolet, auto_make.Dodge, auto_make.Ford, auto_make.Honda, auto_make.Jeep, auto_make.Mercedes, auto_make.Nissan, auto_make.Saab, auto_make.Suburu, auto_make.Toyota, auto_make.Volkswagen, auto_model.92x, auto_model.93, auto_model.95, auto_model.A3, auto_model.A5, auto_model.Accord, auto_model.C300, auto_model.Camry, auto_model.Civic, auto_model.Corolla, auto_model.CRV, auto_model.E400, auto_model.Escape, auto_model.F150, auto_model.Forrestor, auto_model.Fusion, auto_model.Grand.Cherokee, auto_model.Highlander, auto_model.Impreza, auto_model.Jetta, auto_model.Legacy, auto_model.M5, auto_model.Malibu, auto_model.Maxima, auto_model.MDX, auto_model.ML350, auto_model.Neon, auto_model.Passat, auto_model.Pathfinder, auto_model.RAM, auto_model.RSX, auto_model.Silverado, auto_model.Tahoe, auto_model.TL, auto_model.Ultima, auto_model.Wrangler, auto_model.X5, auto_model.X6, auto_year, days_after_bind, years_after_auto_year, policy_bind_year

3. Modeling

The modeling comprises of 3 types, which are linear models, nonlinear models, and tree-based models. In each modeling type, we present the sub method which classified based on their types. In the linear models, we do the Linear Discriminant Analysis (LDA), Sparse LDA, Partial Least Square (PLS) LDA, Logistic Regression, and Penalized Logistic Regression. In the nonlinear models, we present the neural network, average neural network, support vector machine, and key nearest neighbor. In the tree-based models, we do the regression tree, random forest, and boosting tree.

Besides that, we implement those models into two different datasets. First, we build the model on the original dataset. Second, we do the modeling on the up-sampling dataset. Other than that, we also do the modeling by using the alternate cutoff on each dataset. The purpose by doing it on different dataset and different method is to compare the performance and finally decides which model and method works best for predicting the insurance fraud. Figure 4.1 presents the hierarchy of how the models and types of datasets used in our implementation.

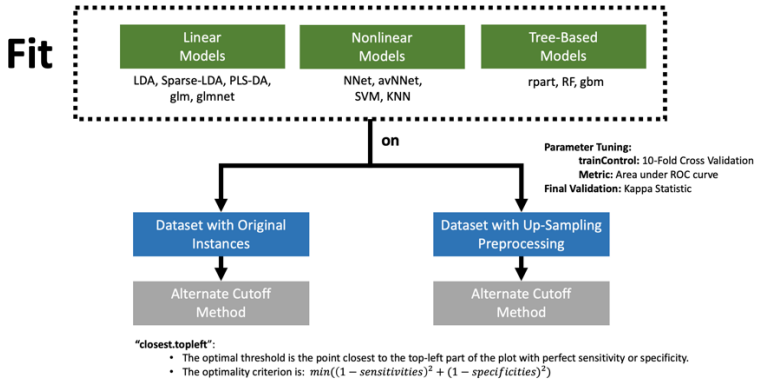


Fig 4.1. Modeling Procedures

In addition, all the model will be done through the 10-fold cross validation on the train set. Finally, the kappa values will be used to determine the best model.

3.1. Linear Classification Modeling

The linear classification models' purpose is to categorize samples into group based on the predictor characteristic (Kuhn & Johnson, 2013). In this project, we do the LDA, Sparse LDA, and PLS DA to do the linear classification modeling.

3.1.1. Linear Discriminant Analysis (LDA)

LDA is a model than can be used while the dataset contains more sample than the predictors (Kuhn & Johnson, 2013). Other than that, the predictors should be independent (Kuhn & Johnson, 2013). This model is also known as a P-dimensional vector where the values are directly paired to the original predictors (Kuhn & Johnson, 2013). Such values can be used to see the contribution of each predictor while doing the prediction (Kuhn & Johnson, 2013).

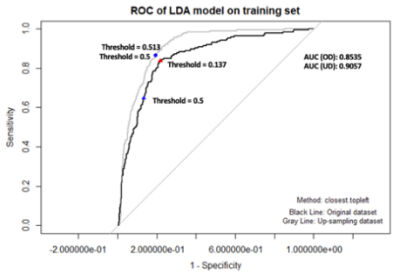


Fig 4.2. The ROC of LDA on the train set.

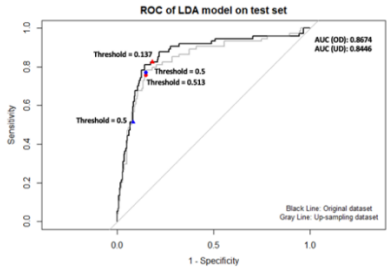


Fig 4.3. The ROC of LDA on the test set.

As mentioned in the previous section, we implement the LDA on the two different datasets. The figure 4.2 and 4.3 show the performance of the LDA model while it used on the training and the test set. The performances are shown through the ROC curve. Specifically, on each picture, we can see that there are two different lines. The black line represents the performance of the model in the original dataset, while the gray line is the performance of the model in the up-sampling dataset.

In detail, according to those two pictures, we can see that the AUC value of the train set while it is modeled by using the original dataset is 0.8535 (in figure 4.2), and the AUC of the test set is 0.8674 (in figure 4.3). the comparison of these AUC value, we conclude that the LDA model does not overfit the train set on while the original dataset is used as the AUC value of the train set is lower than the test, and the difference of the value is not significant. However, the up-sampling dataset tends to overfit the model as the AUC value of the train set is higher than the test set.

Other than that, we also can see that the new threshold has been applied as well on those two datasets. Specifically, on figure 4.2, we can see that the new threshold after the cutoff method has been applied on the model is 0.137 for the original dataset, and the new threshold for the up-sampling dataset is 0.513.

3.1.2. Sparse LDA

This model is a penalized model of LDA which use penalties or regularization to improve the fit to the data (Kuhn & Johnson, 2013). By adding the penalties, such as ridge regression, the model can overcome the problem of the high correlated predictors if they are existed the dataset (Kuhn & Johnson, 2013). This model tunes the parameters to get the best performance. There are two parameters that can be tuned in this model. The first one is λ, which controls the total amount of penalization (Kuhn & Johnson, 2013). The second one is the number of variables (Kuhn & Johnson, 2013).

The best tuning parameter of the sparse LDA model on the original dataset and on the up-sampling dataset are shown on figure 4.4 and 4.5. According to the pictures, we can see that the best lambda values and number of predictors while the model reach the best performance on the original dataset are 0.1 and 50 (on fig 4.4).

The ROC curve which tells the performance of the sparse LDA model is shown by figure 4.6 and 4.7. According to all methods that have been applied on the sparse LDA model, we obtain that the best performance is while the model uses the original dataset within the new threshold (0.0270). Such performance has the highest kappa values among the other methods which are implemented in this model, which is, 0.6764.

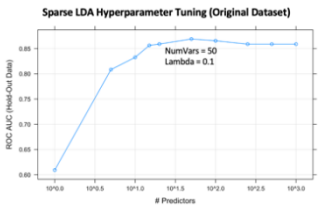


Fig 4.4.

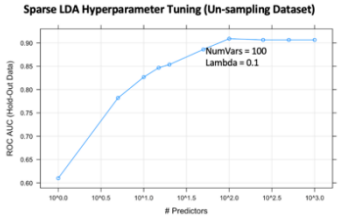


Fig 4.5.

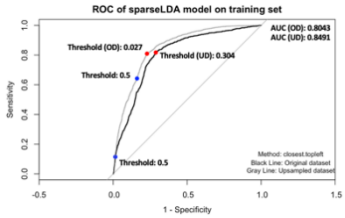


Fig 4.6.

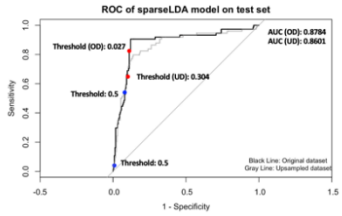


Fig 4.7.

3.1.3. Partial Least Squares Discriminant Analysis (PLS-DA)

PLS-DA is the model which works by simultaneously reduce the dimension and maximize the correlation of the variables to minimize the classification error (Kuhn & Johnson, 2013). In this model there is only parameter that can be tuned that is the numVars. The numVars is the number of latent variables to be retained (Kuhn & Johnson, 2013).

In this model, the best tuning parameter while the model reaches the best performance on the original dataset and on the up-sampling dataset are shown on figure 4.8 and 4.9. According to such picture, we can see that the PLS-DA reach the best performance on the original dataset when the number of latent variables to be retained is 6; while it gets 10 on the up-sampling dataset. Besides that, the model performance on each dataset is shown by the ROC curve on figure 4.10 and 4.11.

According to all methods that have been applied on the PLS DA model, we obtain that the best performance is while the model uses the original dataset within the new threshold (0.3989). Such performance has the highest kappa values among the other methods which are implemented in this model, which is, 0.5704.

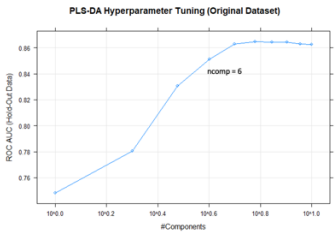


Fig 4.8.

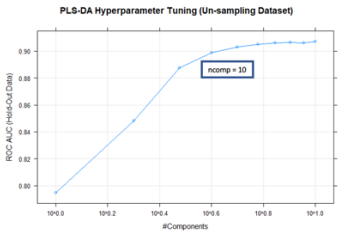


Fig 4.9.

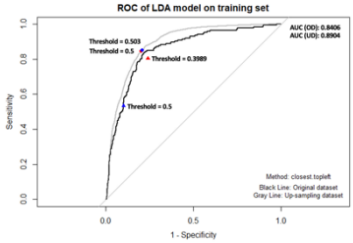


Fig 4.10.

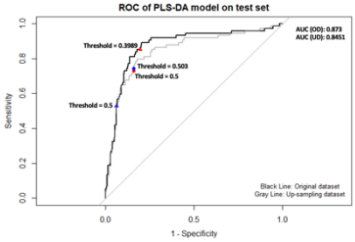


Fig 4.11.

3.1.4. Logistic Regression

The logistic regression is based on assuming each observation of response variable follows a Bernoulli distribution (Kuhn & Johnson, 2013). Theoretically, we can calculate the probability by using the logistic function and classify the response variable into positive event or negative event with default threshold. In addition, this model has no hyperparameter to tune.

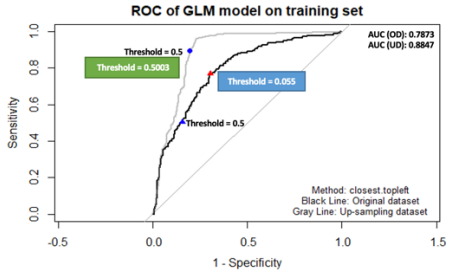


Fig 4.12

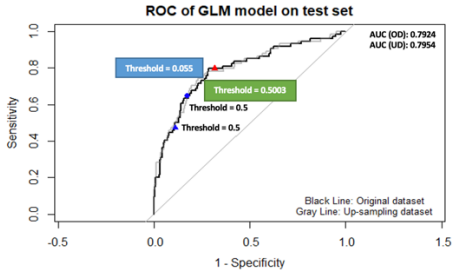


Fig 4.13

According to Fig 4.12 and Fig 4.13, we can see area under the curve of each dataset. First, in the original dataset, we can see the AUC of training set is 0.7873, and the test set is 0.7924. This means that the model with original dataset is not overfitted on the training set. However, in the up-sampling dataset, we can see the AUC of training set is 0.8847, and the test set is 0.7954. This means that the model with up-sampling dataset tends to be overfitted on the training set. In addition, the ROC curve plots of training set at left upper corner, we selected the alternative threshold using closest.topleft method. New thresholds of up-sampling dataset and original dataset are 0.5004 and 0.055, respectively. We applied them to the prediction on test set. As a result, we might conclude that up-sampling method cannot improve the model performance at all.

3.1.5. Penalized Logistic Regression

The penalized logistic regression is used the ridge and lasso penalties simultaneously to penalize regression coefficients (Kuhn & Johnson, 2013). We can select important predictors against highly correlated predictors by utilizing the hyperparameter, including alpha and lambda (Kuhn & Johnson, 2013). Alpha is an elastic net mixing parameter, with $0 \leq \alpha \leq 1$ (Kuhn & Johnson, 2013). When alpha is 1, it means the lasso penalty, and alpha is 0, it means the ridge penalty (Kuhn & Johnson, 2013). Lambda is a regularization parameter, when lambda is 0, it does not be penalized on regression coefficients (Kuhn & Johnson, 2013). If lambda is too large, the model is more likely to be underfitted (Kuhn & Johnson, 2013).

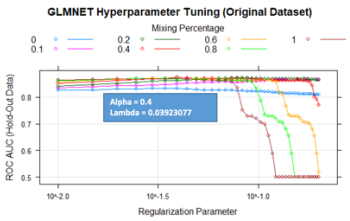


Fig 4.14

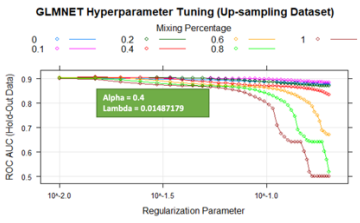


Fig 4.15

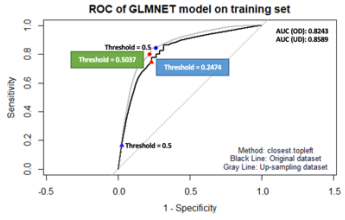


Fig 4.16

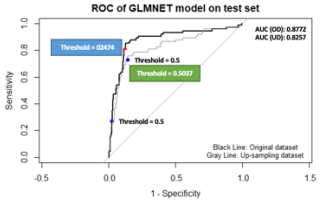


Fig 4.17

According to Fig 4.16 and Fig 4.17, in original dataset, we can see the AUC of training set is 0.8243, and the test set is 0.8772. This means that the model with original dataset is not overfitted on the training set. Similarly, in up-sampling dataset, we can see the AUC of training set is 0.8589, and the test set is 0.8257. This means that the model with up-sampling dataset is likely to overfit on the training set. Even though we implemented the alternative cutoff method, the new threshold of up-sampling dataset is almost same with default threshold 0.5, so we obtain the same result from each threshold. The bottom line is that in the penalized logistic regression model, we can conclude that we might obtain a better performance with original dataset by using alternative cutoff method.

3.2. Nonlinear Classification Modeling

3.2.1. Neural Network

Similar to PLS, there is an intermediate layer (hidden) to generate new unknown variables in neural networks (Kuhn & Johnson, 2013). But in mural networks, the linear combination of original predictors is transformed by an activation function, like logistic function (Kuhn & Johnson, 2013). Parameter size is the number of hidden units and decay is to penalize the weights between units. The tuning results are shown in Fig 4.18. and 4.19.

In Fig 4.18., when the number of units is 5 and decay is 0.1, the model has best performance on original training set. In Fig 4.19., when the number of units is 10 and the decay is 0.01, the model has best performance on up-sampling training set.

According to the ROC curve plots of training set in Fig 4.20, we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.8422 and the new threshold for original dataset is 0.0512. We applied them to the prediction on test set.

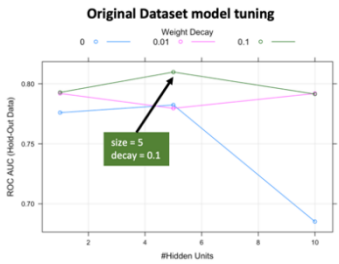


Fig 4.18.

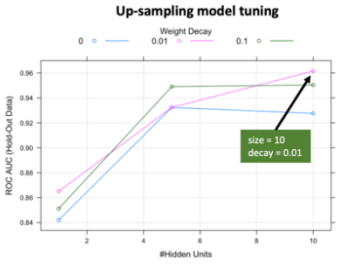


Fig 4.19.

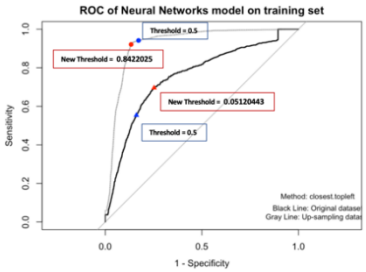


Fig 4.20.

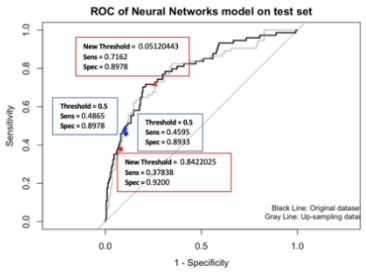


Fig 4.21.

3.2.2. Average Neural Network

Average Neural Network is to train multiple neural networks by assigning different initial values for weights, and then take average of the results from these models as the final prediction value (Kuhn & Johnson, 2013). It can overcome the problem of local optimization (Kuhn & Johnson, 2013). Besides the parameter in a single neural network mdoel, parameter bag here is the number of neural network models to train.

In Fig 4.22. and fig 4.23., models with decay of 0 have the worst performance because they don't penalize weights. So, these models overfit training sets. Models with decay of 0.01 and 0.1 had similar performances. As for best tuning results, the number of hidden units of 10 and decay of 0.01 are the best hyperparameters for both of the models. According to the ROC curve plots of training set in Fig 4.24, we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.5286 and the new threshold for original dataset is 0.1460. We applied them to the prediction on test set.

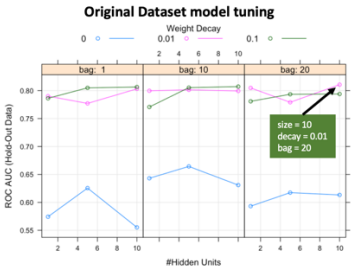


Fig 4.22.

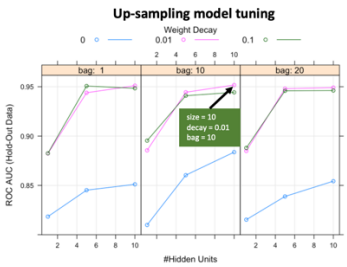


Fig 4.23.

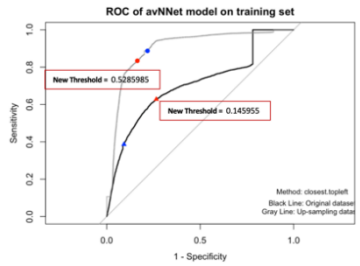


Fig 4.24.

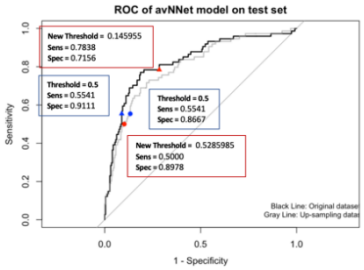


Fig 4.25.

3.2.3. Support Vector Machine

SVM is to find a hyperplane in some kernel-induced feature space that "best" separates the two classes (Kuhn & Johnson, 2013). In our project, Radial Kernel is applied.

Parameter C is the regularization parameter. The higher the c is, the less error can be tolerated, and it is easy to overfit. The smaller C is, the easier it is to underfit. We used 10-Fold Cross Validation to tune c in a search grid. Fig 4.26 shows the model tuning on original dataset and as C increases, the value of AUC first increases and then decreases. The best one is 4. Fig 4.27 shows the model tuning on up-sampling dataset and AUC increases as C. The best one is 16.

According to the ROC curve plots of training set in Fig 4.28, we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.554 and the new threshold for original dataset is 0.263. We applied them to the prediction on test set. The validation results are show in Fig 4.29.

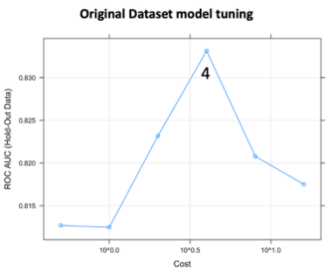


Fig 4.26.

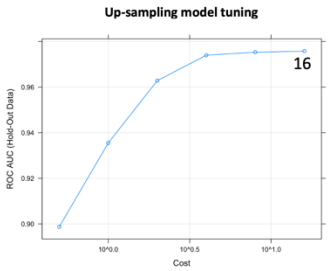


Fig 4.27.

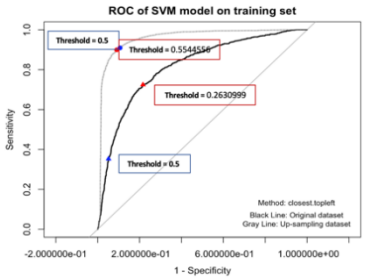


Fig 4.28.

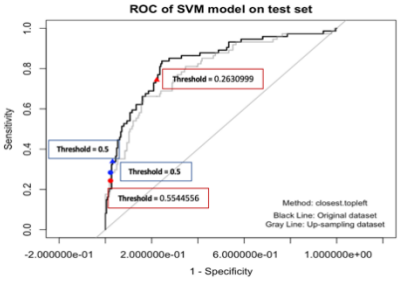


Fig 4.29.

3.2.4. K-Nearest Neighbors (KNN)

KNN model uses k-closed neighbors to predict an instance's class (Kuhn & Johnson, 2013). K is the number of neighbors. When K is too large, the model tends to be underfitting. We used 10-Fold Cross Validation to tune K in a search grid. In Fig 4.30. and Fig 4.31., we can see that the cross-validated AUC values quickly decrease with K because the fitted models become underfitting and have higher bias with the increase of cp. The best K for original dataset and up-sampling dataset are both 1. This may result in overfitting on the test set.

According to the ROC curve plots of training set in Fig.4.32, we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.607 and the new threshold for original dataset is 0.236. We applied them to the prediction on test set. The validation results are show in Fig.4.33. The result of new threshold is overlapped with the default one, which means that alternate cutoff method doesn't improve the performance.

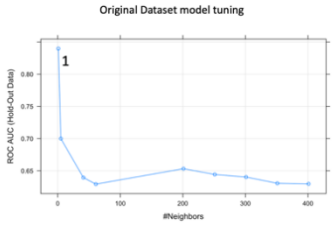


Fig 4.30.

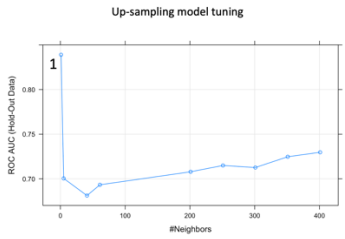


Fig 4.31.

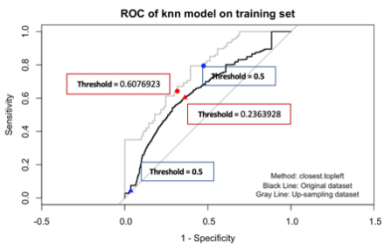


Fig 4.32.

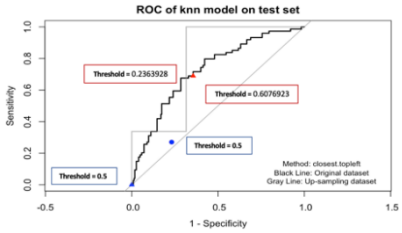


Fig 4.33.

3.3. Tree-Based Classification Modeling

3.3.1. Basic Classification Tree

The basic classification tree model constructs a top-down tree by selecting optimal attributes through minimizing Gini index (Kuhn & Johnson, 2013). The hyperparameter **cp** is to restrict the complexity of a tree. And the higher the cp is, the simpler the tree will be. We used 10-Fold Cross Validation to tune **cp** in a search grid with a tuneLength of 30. In Fig 4.34. and Fig 4.35., we can see that the cross-validated AUC values gradually decrease with cp because the fitted models become underfitting and have higher bias with the increase of cp. As for the model fitted on original dataset, cp of 0.025 has the highest cross-validated

AUC value. However, as for the model fitted on up-sampling dataset, the best cp is 0, which means the tree is not pruned. It is more likely to overfit the training set.

According to the ROC curve plots of training set in Fig 4.36., we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.51 and the new threshold for original dataset is 0.22. We applied them to the prediction on test set. However, the roc curves on test set in Fig 4.37. shows, the result of new threshold is overlapped with the default one, which means that alternate cutoff method does not improve the performance.

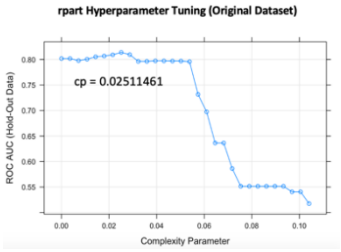


Fig 4.34.

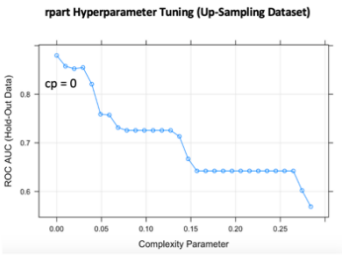


Fig 4.35.

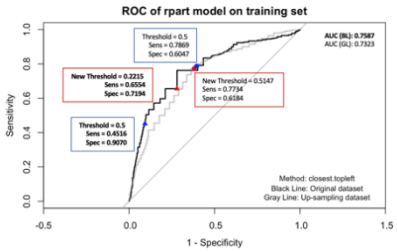


Fig 4.36.

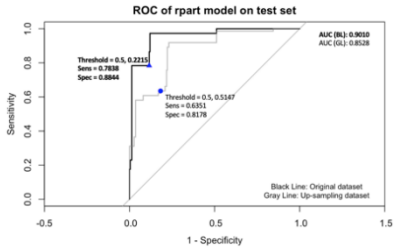


Fig 4.37.

3.3.2. Random Forest

Random forest models use bootstrap samples and randomly chosen predictors to train multiple single trees. The final prediction value is voted by all single trees (Kuhn & Johnson, 2013).

The hyperparameter **ntree** is the number of trees to grow. We set it to 500 here.

The hyperparameter **mtry** is the number of variables randomly sampled as candidates at each split. We used 10-Fold Cross Validation to tune it in search grids of {30,40,50,60,70,100} (original dataset) and {10,20,30,40,50,60} (up-sampling dataset). In Fig 4.38., as for the model fitted on original dataset, mtry of 60 has the highest cross-validated auc value. In Fig 4.39., as for the model fitted on up-sampling dataset, the best mtry is 20.

According to the ROC curve plots of training set in Fig 4.40., we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.687 and the new threshold for original dataset is 0.279. We applied them to the prediction on test set. The results are shown in Fig 4.41.

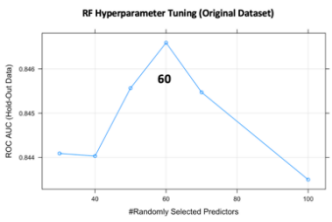


Fig 4.38.

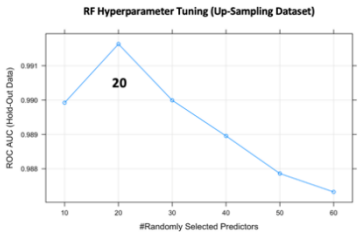


Fig 4.39.

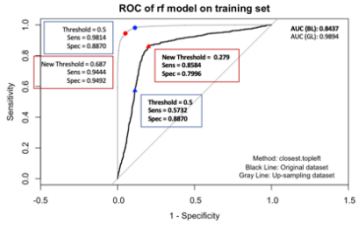


Fig 4.40.

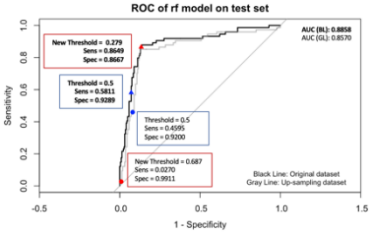


Fig 4.41.

3.3.3. Gradient Boosting Tree

Gradient Boosted Tree sequentially generates a set of weak classifiers by assigning and updating weights for each data point (Kuhn & Johnson, 2013). And then the results from all Classifiers are weighted to make the final prediction (Kuhn & Johnson, 2013).

The hyperparameter **ntree** is Number of trees to grow. We used 10-Fold Cross Validation to tune it with AUC metric in a search grid of {50,100,500}. The hyperparameter **interaction.depth** is the maximum depth of each tree. We used 10-Fold Cross Validation to tune it with AUC metric in a search grid of {1,3,5}. The hyperparameter **n.Minobsinnode** is the minimum number of observations in the terminal nodes of the trees. We set it to 10. The hyperparameter **Shrinkage** is the learning rate. We used 10-Fold Cross Validation to tune it with AUC metric in a search grid of {0.01,0.1,0.5}.

The final tuning results for original dataset and up-sampling dataset are respectively shown in Fig 4.42. and 4.43. We can see that on the original dataset, models with smaller **shrinkage** and **ntrees** tend to have better performance. However, it is contrary to the result of up-sampling dataset. Besides, on the up-sampling dataset, the models with larger **interaction.depth** have better performance. In detail, as for models fitted on original dataset, n.trees = 500, interaction.depth = 1, shrinkage = 0.01, and n.minobsinnode = 10. As for models fitted on up-sampling dataset, n.trees = 500, interaction.depth = 5, shrinkage = 0.1, and n.minobsinnode = 10.

According to the ROC curve plots of training set in Fig 4.44., we selected the alternate threshold using closest.topleft method. New threshold for up-sampling dataset is 0.53 and the new threshold for original dataset is 0.249. We applied them to the prediction on test set. The results are shown in Fig 4.41.

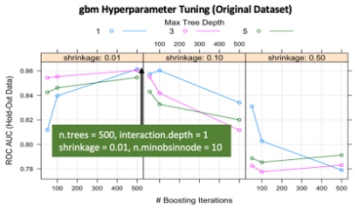


Fig 4.42.

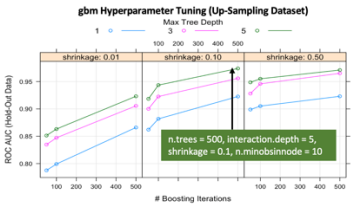


Fig 4.43.

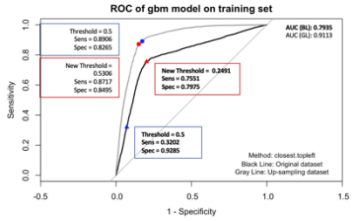


Fig 4.44.

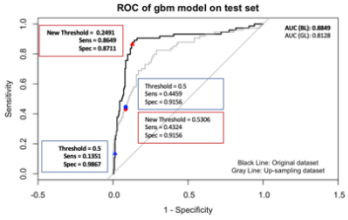


Fig 4.45

4. Evaluation

4.1. Model Performance

This section explains the performance of all method that we have done. The performance is shown through the AUC values on table 5.1, and the Kappa Statistics on table 5.2.

In table 5.1, almost the whole models overfit the train set as almost the whole AUC values of the train set are higher than the test set. Other than that, we conclude that the up-sampling method which is applied to the dataset does not guarantee to improve the model performance as there is almost no improvement on the AUC value on the test set of the up-sampling dataset.

In table 5.2, the alternate cutoff does improve the model performances of some method in the original dataset. Specifically, the alternate cutoff improves almost the whole model on the linear models, half of the nonlinear models, and almost the whole tree-based models. In contrary, the alternate cutoff does not yield the same thing on the up-sampling dataset. Instead, the model tends to overfit the train set after the alternate cutoff are applied as the kappa values most likely to decrease or remain the same on each model after the new threshold is applied.

Therefore, according to the performance on table 5.1 and 5.2, we conclude that applying the alternate cutoff on the original dataset does improve the model performance even though there is some imbalance class on the model.

AUC	Models	Original Dataset			Up-sampling Dataset		
		AUC (training)		AUC (test)	AUC (training)		AUC (test)
Linear Models	LDA	0.8535	<	0.8674	0.9057	>	0.8446
	Sparse LDA	0.8043	<	0.8784	0.8491	<	0.8601
	PLS-DA	0.8406	<	0.873	0.8904	>	0.8451
	glm	0.7873	<	0.7924	0.8847	>	0.7954
	glmnet	0.8243	<	0.8772	0.8589	>	0.8257
Nonlinear Models	NNet	0.7521	<	0.7957	0.9252	>	0.7772
	avNNet	0.6886	<	0.8285	0.8996	>	0.8017
	SVM	0.8169	<	0.8329	0.9591	>	0.7969
	KNN	0.6356	<	0.7171	0.7365	>	0.5196
Tree-Based Models	rpart	0.7587	<	0.901	0.7323	<	0.8528
	RF	0.8437	<	0.8858	0.9894	>	0.857
	gbm	0.7935	<	0.8849	0.9113	>	0.8128

Table 5.1. AUC Metric

Kappa Statistic	Models	Original Dataset			Up-sampling Dataset		
		Default Threshold		Alternate Cutoff	Default Threshold		Alternate Cutoff
Linear Models	LDA	0.472	<	0.5698	0.5814	>	0.5712
	Sparse LDA	0.0461	<	0.6764	0.4966	<	0.554
	PLS-DA	0.5137	<	0.5704	0.5415	>	0.5312
	glm	0.4316	>	0.3842	0.4489	=	0.4489
	glmnet	0.0792	<	0.6524	0.5572	=	0.5572
Nonlinear Models	NNet	0.4123	>	0.3926	0.3804	>	0.3418
	avNNet	04943	>	0.4101	0.4265	>	0.4247
	SVM	0.3789	<	0.4568	0.3789	>	0.2875
	KNN	0	<	0.2651	0.0395	=	0.0395
Tree-Based Models	rpart	0.6392	=	0.6392	0.4259	=	0.4259
	RF	0.5472	<	0.6707	0.4215	>	0.0266
	gbm	0.1684	<	0.6776	0.4016	>	0.3885

Table 5.2. Kappa Statistic Metric

4.2. Best Model

According to table 5.2, gbm model with an alternative cutoff (Threshold = 0.2490539) has the best performance with a Kappa statistic of 0.6676 and AUC of 0.8849. However, we would like to utilize the Sparse LDA model (Threshold = 0.0270) to classify whether a claim is fraud or not even though it has slightly poorer performance than gbm's. This is because gbm model is exceedingly difficult to interpret the result and taking a great deal of executing time. According to positive predictive value (ppv) and negative predictive value (npv), the trained model (Sparse LDA) can provide insurance companies with a confidence of 70.93% to believe a claim that is classified fraud is real fraud and a confidence of 93.9% to believe a claim that is classified non-fraud is real non-fraud.

	Dataset	Method	Model	Kappa	AUC training	AUC test	Accuracy	Sensitivity	Specificity	ppv	npv	Hyperparameter
Best Model	Original Dataset	Alternate Cutoff	gbm	0.6776	0.7935	0.8849	0.8696	0.8649	0.8711	0.6882	0.9515	n.trees = 500, interaction.depth = 1 shrinkage = 0.01, n.minobsinnode = 10. Threshold = 0.2490539
	Dataset	**Method**	**Model**	**Kappa**	**AUC training**	**AUC test**	**Accuracy**	**Sensitivity**	**Specificity**	**ppv**	**npv**	**Hyperparameter**
Model to Use	Original Dataset	Alternate Cutoff	**Sparse LDA**	0.6764	0.8043	0.8784	0.8729	0.8243	0.8889	0.7093	0.9390	NumVars = 50 , Lambda = 0.1 Threshold = 0.0270

Table 5.3. Best Models

4.3. Important Variables

The variable importance plots for the best model in each type are shown in Fig 5.1., Fig 5.2., and Fig 5.3. The results for sparseLDA and svmRadual are same. This is because we used "varImp" function in r and there are not corresponding variable importance methods for them. The importance of each predictor in the two models is evaluated individually using a "filter" approach. So, they have the same results.

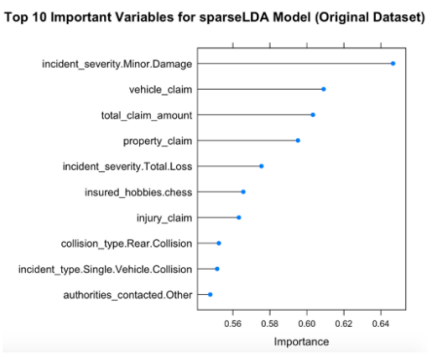


Fig 5.1. Linear Model

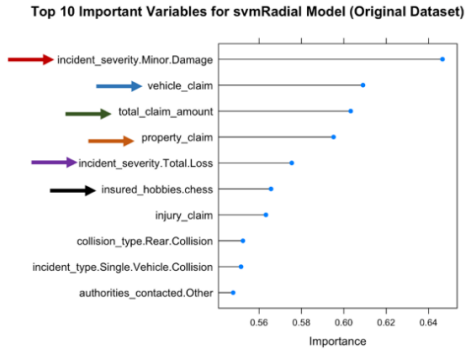


Fig 5.2. Nonlinear Model

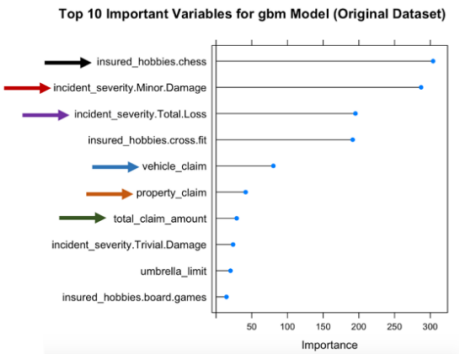
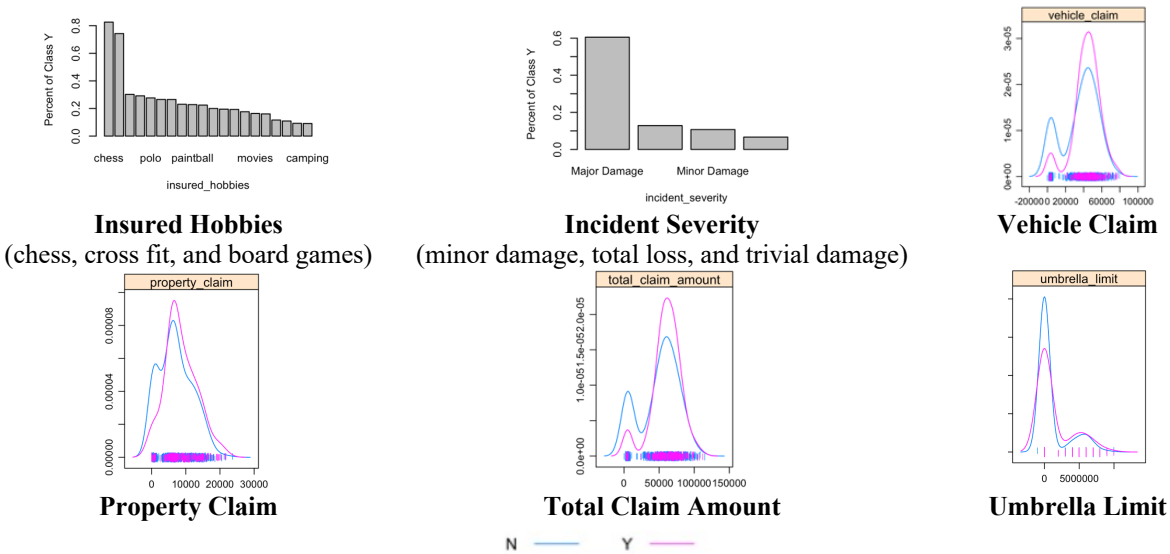


Fig 5.3. Tree-Based Model

According to Fig 5.2. and Fig 5.3., the common important predictors in the three models are whether the incident severity is minor damage, the amount of vehicle claim, the total claim amount, property claim, whether the incident severity is total loss, and whether the insured hobbies include chess. As for whether the incident severity is total loss and whether the incident severity is minor damage are two levels of variable incident severity.

According to the variable importance plot for the best model (gradient boosted tree) in Fig 5.3., we can see that insured hobbies (chess, cross fit, and board games), incident severity (minor damage, total loss, and trivial damage), vehicle claim, property claim, total claim amount, and umbrella limit are most important variables. The detailed density plots and bar plots are shown in Fig 5.4.

In Fig 5.4., each level in the variable of insured hobbies has extremely imbalanced distribution between class Y and N. It means this predictor can greatly classify whether the claim is fraud or not. This conclusion can be applied to the variable of incident severity as well. For example, if the insured person likes a chess game, the probability that the claim is fraud is more than 80%. If the incident severity is minor damage, the probability that the claim is non-fraud is almost 90%. As for numerical predictors, when the vehicle claim is around 50000, or the property claim is around 10000, or total claim amount is around 70000, the claim is more likely to be fraud. However, when umbrella limit is near 0, the claim is more likely to be non-fraud.



Insured Hobbies
(chess, cross fit, and board games)

Incident Severity
(minor damage, total loss, and trivial damage)

Vehicle Claim

Property Claim

Total Claim Amount

Umbrella Limit

Fig 5.4. Top 10 important Variables for Gradient Boosted Tree

5. Conclusion

Finally, we applied 5 linear classification models, 4 nonlinear classification models, and 3 tree-based models to predict whether a claim is fraud or not. Besides, we additionally implemented alternate cutoff method and up-sampling method to increase the prediction accuracy of the minority class samples. In most cases, up-sampling method does not increase the prediction accuracy of the minority class samples and tends to overfit the training set. Alternate cutoff method has considerable improvements for this imbalanced dataset.

The result shows gradient boosted tree (gbm) fitted on original dataset with an alternate cutoff (Threshold = 0.2490539) has the best performance with a Kappa statistic of 0.6676 and AUC of 0.8849. However, we tend to utilize the Sparse LDA model (Threshold = 0.0270) to classify whether a claim is fraud or not even though it has slightly poorer performance than gbm's. This is because gbm is difficult to interpret and taking a great deal of executing time.

As for recommendations for insurance companies, they should focus more on the validation of those important predictors, like whether an incident severity is minor damage, amount of a vehicle claim, and types of hobbies for each insured person. Besides, the trained model (Sparse LDA) can provide insurance companies with a confidence of 70.93% to believe a claim that is classified fraud is real fraud and a confidence of 93.9% to believe a claim that is classified non-fraud is real non-fraud.

Bibliography

Bologa, A., Bologa, R., & Florea, A. (2013). Big Data and Specific Analysis Methods for Insurance Fraud Detection. *Database Systems Journal*, 30-39.

FBI. (2010, May 17). *Insurance Fraud*. Retrieved from FBI: https://www.fbi.gov/stats-services/publications/insurance-fraud

Insurance Research Council. (2015, February 4). *Fraud Adds Up to 17% to Auto Insurance Injury Claims: IRC*. Retrieved from Insurance Journal: https://www.insurancejournal.com/news/national/2015/02/04/356392.htm#:~:text=A%20new%20study%20estimates%20that,paid%20in%20the%20United%20States.&text=States%20with%20highest%20rates%20of,Florida%20(31%20percent)

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* New York: Springer.

Shah, B. (2018, August 20). *Auto Insurance Claims Data*. Retrieved from Kaggle: https://www.kaggle.com/buntyshah/auto-insurance-claims-data