Project Report

Transitional Capstone
BAN 693T
Spring 2019-Fall 2019

Predicting Car Insurance Fraud

Submitted by: Leena Hemant Damle

NetID: ch2879

Introduction

Car Insurance fraud consists of filing false insurance claims with the fraudulent intention towards an insurance provider. These frauds can be committed by Insurance company employees or by the insurance claimants.

According to the United States Federal Bureau of Investigation, the most common schemes include: premium diversion, fee churning, asset diversion, and workers compensation fraud. The total cost of insurance fraud (non-health insurance) is estimated to be more than $40 billion per year. That means Insurance Fraud costs the average U.S. family between $400 and $700 per year in the form of increased premiums. Hence it is very essential to identify and stop this fraud.

Objective

The aim of this project is **to identify whether a car insurance claim is fraudulent or not** using Logistic Regression and Random Forest models and to compare the performances of these models. I will also be implementing feature selection, 10-fold cross validation and upsampling techniques to overcome the data imbalance to improve model accuracy.

Data Description

Source: This Dataset is used in an example in the book: Machine Learning in Java by Boštjan Kaluža. (This book is also available in the CSUEB online library database). I chose this data because the book only gave a basic introduction to fraud detection models, explained possible ways of overcoming data imbalance and then left it up to the user to explore them.

The dataset describes insurance vehicle incident claims for an undisclosed insurance company. It contains **15,430 claims**; each claim comprises **33 attributes** describing the following components:

- Customer demographic details (like Age, Sex, MartialStatus)
- Purchased policy (PolicyType, VehicleCategory, number of supplements, agent type, and so on)

- Claim circumstances (day/month/week claimed, policy report filed, witness present, past days between incident-policy report, incident-claim, etc)
- Other customer data (number of cars, previous claims, DriverRating, etc)
- Fraud found (yes and no)

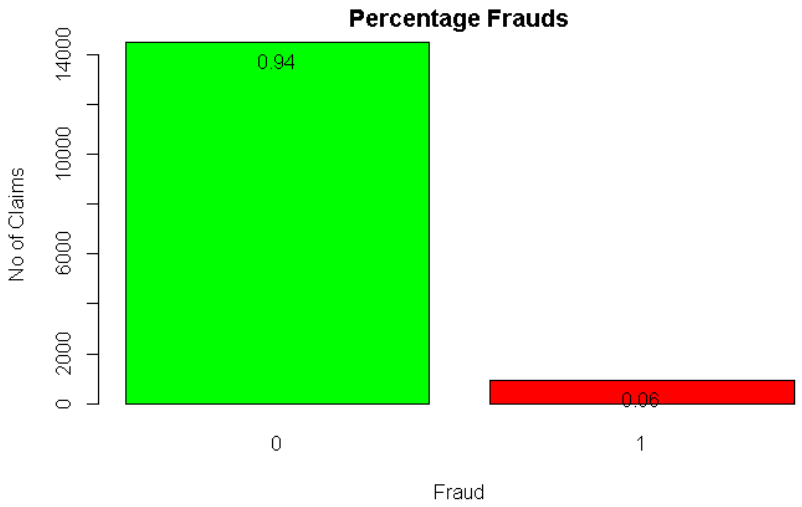
Overall, the dataset is **imbalanced.** There were **missing values** in only one variable in the dataset.

Data Cleaning, Pre-processing and Exploratory Analysis

Cleaning and Preprocessing:

- Certain attributes that were numeric by default, like Year, Rep Number, Week of the Month, Week Of the Month Claimed were encoded to categorical form by converting them to factors in R.
- This is done to avoid treating them as numbers during model training. For example, Rep Number ‘2’ does not indicate that the Representative is twice as good as Rep Number ‘1’.
- Attribute Age contained missing data. A binned version of Age called ‘AgeOfPolicyHolder’ was also present, which did not have missing values. Hence, I selected it and dropped attribute ‘Age’.
- Attribute ‘PolicyNumber’ was also dropped to avoid overfitting the models.
- No other missing values were found in the data set.

Data Exploration:



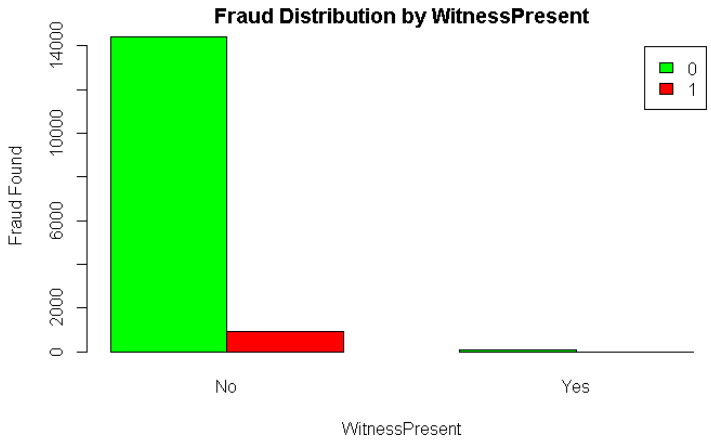
923 frauds were found out of 15420 samples. Thus, 94% claims were not fraudulent. The Data is highly imbalanced.

Gender and Frauds:

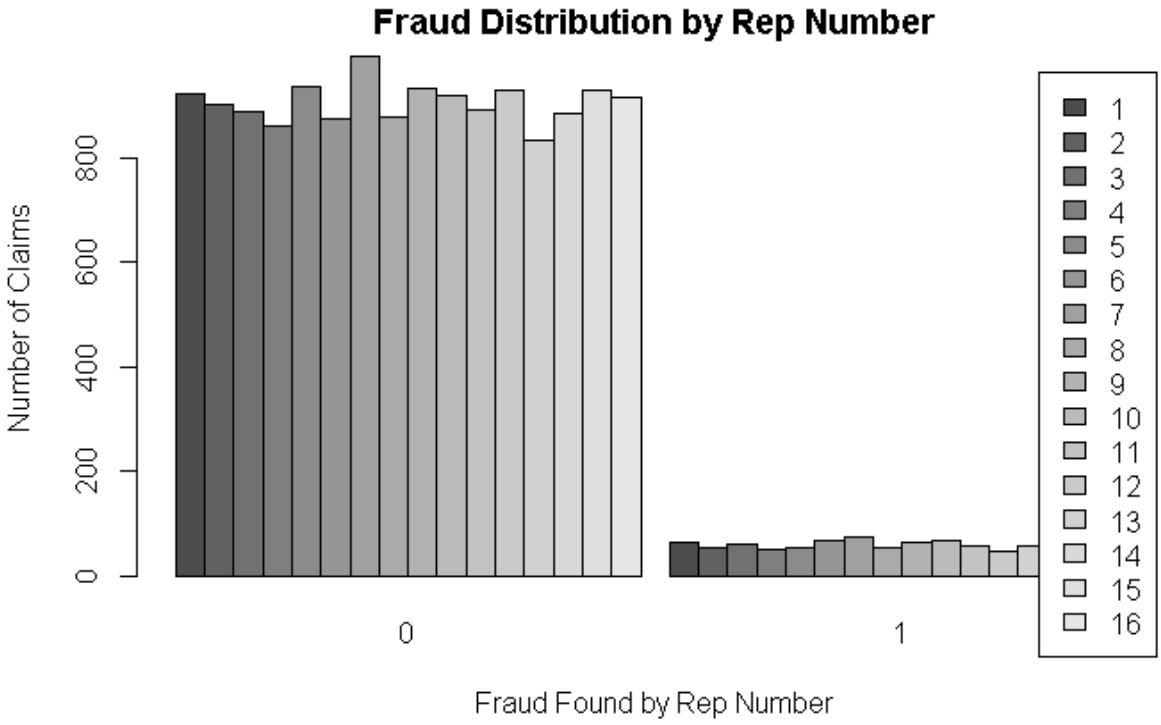
In general, our data contains more male claimants, hence they have more fraudulent claims.

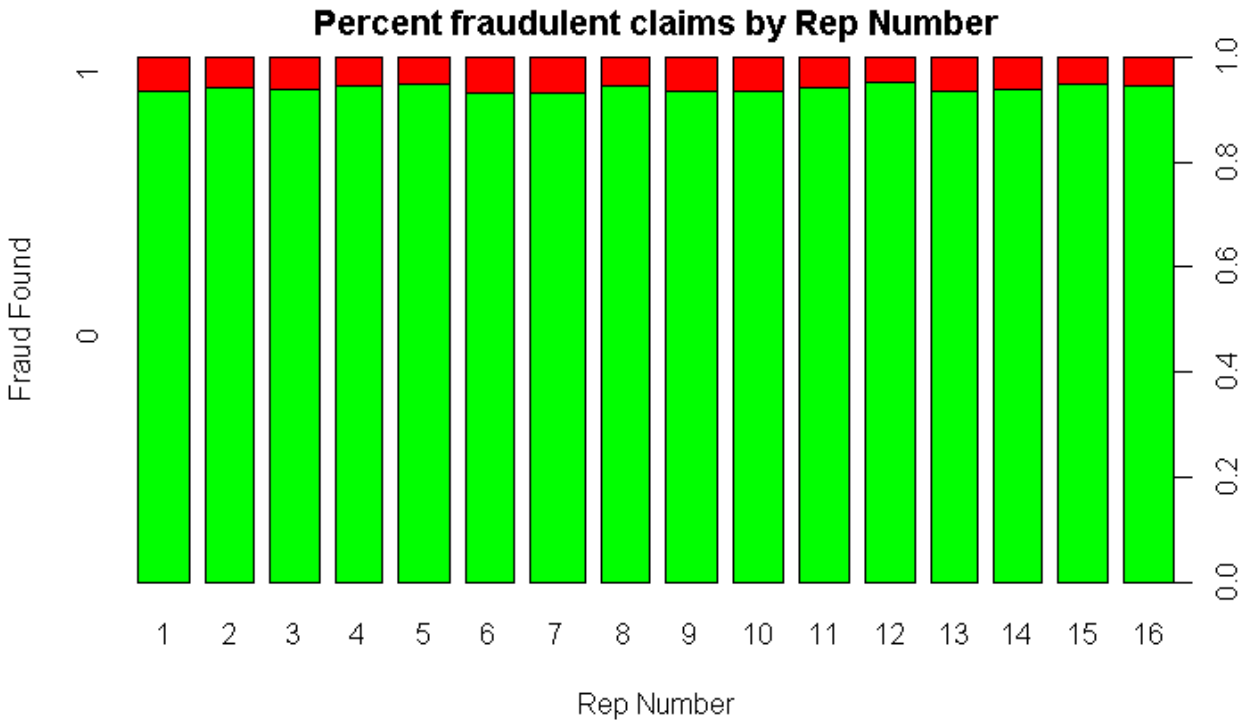
Witness Present and Frauds:

Like Gender, most insurance claims do not have witnesses present. Mere visualization is insufficient to conclude whether most fraudulent claims have witnesses involved or not.



Rep Number (Representative Agent number) and Fraud:





Rep Numbers 6 and 7 seem to have more percent of Fraudulent claims than other Reps.

Division into training and test data sets:
65% data is randomly selected as training set and the remaining 35% is used as test data.

Evaluation Methods

The main metric used to evaluate the model performances is **Specificity,** rather than just Accuracy.
In this case, the Negative class is ‘Fraud found’ and Positive Class as ‘Fraud not found’.
Since we are interested in maximizing the identification of Frauds or True Negatives, we will evaluate the models based on their Specificity.
Specificity identifies what percentage of negative tuples did the model actually classify as negative.

$$\text{Specificity} = \frac{\text{True negative}}{\text{True negative + false positive}}$$

Application of Machine learning algorithms

1. Logistic Regression:

- Logistic regression is used to explain the relationship between one binary dependent variable (Fraud Found or Not in our case) and one or more nominal, ordinal, interval or ratio-level independent variables.
- Without cross validation and upsampling, logistic regression gave an accuracy of 94.42%, balanced accuracy of 50.59% and Specificity of only 1.35%.
- This model is not ideal as we would be unsuccessful in identifying only 1.35% frauds. The model needs to be improved.

Basic Multiple Logistic Regression model accuracy for test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5092  292
##          1    9    4
##
##                Accuracy : 0.9442
##                  95% CI : (0.9378, 0.9502)
##     No Information Rate : 0.9452
##     P-Value [Acc > NIR] : 0.6318
##
##                   Kappa : 0.0214
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99824
##             Specificity : 0.01351
##          Pos Pred Value : 0.94577
##          Neg Pred Value : 0.30769
##              Prevalence : 0.94515
##          Detection Rate : 0.94349
```

```
##    Detection Prevalence : 0.99759
##       Balanced Accuracy : 0.50587
##
##        'Positive' Class : 0
##
```

K-fold cross validation:

This is a sampling technique in which:

1. Data is randomly split into k-subsets (or k-fold)
2. One subset is reserved for testing the model and the remaining subsets are used to train the model. Prediction error on the test subset is recorded.
3. This is repeated until each of the k subsets has served as the test set.
4. An average of the k recorded errors, called the cross-validation error is calculated and used as the performance metric for the model.

An advantage of using K-fold cross-validation (CV) is that it is a robust method for estimating the accuracy of a model.
Usually, we use k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that do not suffer from excessively high bias or very high variance.

- On using 10-fold cross validation, specificity improved only slightly to 2.37%
- Thus, it is necessary to overcome class imbalance in the data to get improved results.

Logistic Regression with 10-fold cross validation

```
Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5098  289
##          1    3    7
##
##                Accuracy : 0.9459
##                  95% CI : (0.9395, 0.9518)
##     No Information Rate : 0.9452
##     P-Value [Acc > NIR] : 0.4204
##
```

```
##                   Kappa : 0.0423
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99941
##             Specificity : 0.02365
##          Pos Pred Value : 0.94635
##          Neg Pred Value : 0.70000
##              Prevalence : 0.94515
##          Detection Rate : 0.94460
##    Detection Prevalence : 0.99815
##       Balanced Accuracy : 0.51153
##
##        'Positive' Class : 0
```

Up-sampling:

To resolve the class imbalance and its effect of model specificity, the training data can be sampled in such a way as to try to balance out this imbalance. The minority class (Fraud Found = Yes) is randomly sampled (with replacement) to be the same size as the majority class (Fraud Found =No).

Using a combination of 10-fold cross validation and upsampling makes our model more robust.

- On using upsampling with 10-fold cross validation, specificity improved greatly to 91.89%. Balanced accuracy also improves to 78%.
- However, model accuracy is now 65.63% since more non-fraudulent claims are classified as fraudulent claims. This is a tradeoff we have to make since our main evaluation metric is Specificity rather than accuracy.

Logistic Regression with 10-fold cross validation and upsampling

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3270   24
##          1 1831  272
##
##                Accuracy : 0.6563
```

```
##                  95% CI : (0.6434, 0.669)
##     No Information Rate : 0.9452
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1445
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6411
##             Specificity : 0.9189
##          Pos Pred Value : 0.9927
##          Neg Pred Value : 0.1293
##              Prevalence : 0.9452
##          Detection Rate : 0.6059
##    Detection Prevalence : 0.6103
##       Balanced Accuracy : 0.7800
##
##        'Positive' Class : 0
```

2. Random Forest:

- Random forests or random decision forests are an ensemble learning method for classification, where multiple decision trees are constructed and trained using a random selection of features for each tree. The output class is the mode of the classes or mean prediction of the individual trees.
- In decision trees, data is classified into partitions. The partitions are created to minimize the entropy or non-uniformity between attributes belonging to the same class partition.
- A major advantage of using Random Forest is that it is one of the most accurate classifiers.
- Without cross validation and upsampling, random forest gave an accuracy of 94.52%, balanced accuracy of 50.32% and Specificity of only 0.68%.

Random Forest model accuracy for test set

```
confusionMatrix(data=rf1s_pred, y_test)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 5099  294
##          1    2    2
##
##                Accuracy : 0.9452
##                  95% CI : (0.9387, 0.9511)
##     No Information Rate : 0.9452
##     P-Value [Acc > NIR] : 0.5155
##
##                   Kappa : 0.0119
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999608
##             Specificity : 0.006757
##          Pos Pred Value : 0.945485
##          Neg Pred Value : 0.500000
##              Prevalence : 0.945155
##          Detection Rate : 0.944784
##    Detection Prevalence : 0.999259
##       Balanced Accuracy : 0.503182
##
##        'Positive' Class : 0
```

- Since merely using k-fold cross validation did not give much improved results in case of Logistic Regression model, I proceeded to use upsampling along with k-fold cross validation for Random Forest Model.
- Initially, I tried using 10-fold cross validation. However, this was taking very long to process on my laptop, and I had to use 5-fold cross validation instead to lower the computational time.

Random Forest with 5-fold cross validation and upsampling

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 72.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5099    0
##          1    2  296
```

```
##
##                Accuracy : 0.9996
##                  95% CI : (0.9987, 1)
##     No Information Rate : 0.9452
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9964
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9996
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9933
##              Prevalence : 0.9452
##          Detection Rate : 0.9448
##    Detection Prevalence : 0.9448
##       Balanced Accuracy : 0.9998
##
##        'Positive' Class : 0
```

- Random Forest model gives very good results on using upsampling with 5-fold cross validation. Specificity improved greatly to 100%. Balanced accuracy also improved to 99.98%.
- Model accuracy also improved to 99.96%, crossing our baseline accuracy of 94%.

Model Performance Comparison Table:

Model	Accuracy	Balanced Accuracy	Specificity
Logistic Regression	94.42%	50.59%	1.35%
Logistic Regression with 10-fold CV	94.59%	51.15%	2.37%
Logistic Regression with 10-fold CV and upsampling	65.63%	78%	91.89%
Random Forest	94.52%	50.32%	6.77%
Random Forest with 5-fold CV and upsampling	99.96%	99.98%	100%

Recommendations & Conclusion

To conclude, Random Forest Algorithm used with 5-fold cross validation and upsampling detected the fraudulent insurance claims with 100% Specificity. The exciting part of the project was dealing with the imbalance data. This model also gave a very high accuracy of 99.96% (greater than our baseline accuracy of 94%).

It would be interesting to further explore this model with higher ‘k’ folds or number of trees with the availability of better computational power, and to test the model performance on more data.

References:

1) Machine Learning in Java by Boštjan Kaluža.
2) https://rpubs.com/phamdinhkhanh/389752
3) http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/
4) https://topepo.github.io/caret/subsampling-for-class-imbalances.html

Code:

Predicting Insurance Fraud

```
claims<-read.csv("D:\\CSUEB_MSBA\\Capstone\\claims.csv")

dim(claims)
summary(claims)

#convert numeric to factors
claims$Year <- as.factor(claims$Year)
claims$RepNumber <- as.factor(claims$RepNumber)
claims$WeekOfMonth <- as.factor(claims$WeekOfMonth)
claims$WeekOfMonthClaimed <- as.factor(claims$WeekOfMonthClaimed)
claims$FraudFound_P <- as.factor(claims$FraudFound_P)

summary(claims$AgeOfPolicyHolder[which(claims$Age == 0)])
#Since Age and AgeOfPolicyHolder variables both are the same and as Age varia
ble has missing data, we will drop it. Moreover, variable AgeOfPolicyHolder i
s already divided into bins
claims <-claims[-11]
```

check if there are any missing values in the dataset

```
sum(is.na(claims))
```

no missing values found

```
#Policy number will also cause over-fitting, hence remove it

claims<-claims[,-c(16)]

claims$FraudFound_P <-as.factor(claims$FraudFound_P)#encode as factor
table(claims$FraudFound_P)#see distribution of Response variable- Fraud_Found
_P
contrasts(claims$FraudFound_P) #see how factor is encoded
```

```
count <- table(claims$FraudFound_P)
prop <- round(count/length(claims$FraudFound_P),2)
b <- plot(claims$FraudFound_P, xlab='Fraud', ylab='No of Claims', main='Perce
ntage Frauds', legend = rownames(count),beside=TRUE, col=c("green","red"))
text(b, count, prop, pos=1)
```

923 frauds were found out of 15420 samples

94% claims were not fraudulent

Data is highly imbalanced

Exploratory ANalysis

```
b <- plot(claims$Sex,claims$FraudFound_P , ylab='Fraud Found', xlab='Sex', ma
in='Percentage Frauds by Gender', col=c("green","red"))

counts <- table(claims$FraudFound_P,claims$Sex)
barplot(counts, main="Fraud Distribution by Gender",
  ylab="Fraud Found", xlab= "Gender",
  legend = rownames(counts),beside=TRUE, col=c("green","red"))
text(b, counts, prop, pos=1)

#b <- plot(claims$WitnessPresent ~claims$FraudFound_P , xlab='Fraud Found', y
lab='Witness Present', main='Distribution of Frauds')

b <- plot(claims$WitnessPresent,claims$FraudFound_P , ylab='Fraud Found', xla
b='WitnessPresent', main='Percentage Frauds by WitnessPresent', col=c("green"
,"red"))

counts <- table(claims$FraudFound_P,claims$WitnessPresent)
barplot(counts, main="Fraud Distribution by WitnessPresent",
  ylab="Fraud Found", xlab= "WitnessPresent",
  legend = rownames(counts),beside=TRUE, col=c("green","red"))
text(b, counts, prop, pos=1)

#extract numeric variables
claims_num <-claims[,sapply(claims,is.numeric)]
#claims_num <-claims[,colnames]

#see correlation between the numeric predictors
library(corrplot)
correlations <- cor(claims_num)
corrplot(correlations, method="circle")
# We do not get any additional insights from the two numeric variables

#pairs(claims_num, claims$FraudFound_P)
```

```r
# Stacked Bar Plot with Colors and Legend
counts <- table(claims$RepNumber, claims$FraudFound_P)
barplot(counts, main="Fraud Distribution by Rep Number",
  xlab="Fraud Found by Rep Number", ylab= "Number of Claims",
  legend = rownames(counts),beside=TRUE)

plot(claims$RepNumber, claims$FraudFound_P, main= "Percent fraudulent claims
by Rep Number", ylab="Fraud Found", xlab= "Rep Number",col=c("green","red"))

set.seed(123)
#split into training (65%) and test(35%) data by random partitioning
x <- claims[, -15]
y <- claims$FraudFound_P
n <- nrow(x)

train <- sample(1:n, floor(0.65 * n))

y_test <- y[-train]
y_train <- y[train]
x_test <- x[-train, ]
x_train <- x[train, ]
```

Multiple logistic regression model

```r
glm_full <- glm(y_train ~ ., data=x_train, family=binomial,maxit = 100)
summary(glm_full)
```

making predictions and evaluating accuracy of full model

```r
glm_full_pred <- predict(glm_full, newdata = x_test, type="response")
glm_full_pred <-ifelse(glm_full_pred >=0.5,1,0)

library(caret)
glm_full_pred <- as.factor(glm_full_pred)
confusionMatrix(data=glm_full_pred, y_test)
```

Evaluating full model performace using AUC

```r
library(ROCR)
p <- predict(glm_full, newdata=x_test, type="response")
pr2 <- prediction(p, y_test)
prf2 <- performance(pr2, measure = "tpr", x.measure = "fpr")
plot(prf2)
auc1 <- performance(pr2, measure = "auc")
auc1 <- auc1@y.values[[1]]
print(auc1)
```

Basic Random forest model

```r
library(randomForest)
table(claims$FraudFound_P)
```

```r
set.seed(999)

rf1 <- randomForest(FraudFound_P ~ ., data=claims,importance = TRUE, ntree=10
00, subset=train)
rf1
varImpPlot(rf1, type=1)
plot(c(1:1000), rf1$err.rate[,1], type='l')

rf1_pred <- predict(rf1, newdata = claims[-train, ])
```

RF model accuracy for test set

```r
confusionMatrix(data=rf1_pred, y_test)

#claims$FraudFound_P <- as.factor(claims$FraudFound_P)
library(caret)
#data(GermanCredit)
Train <- createDataPartition(claims$FraudFound_P, p=0.6, list=FALSE)
training <- claims[ Train, ]
testing <- claims[ -Train, ]
```

Logistic Regresion with 10 fold cross validation

```r
library(caret)
# Define training control
set.seed(123)
train.control <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 1)
# Train the model
model <- train(FraudFound_P ~., data = claims, method="glm", family="binomial
",
               trControl = train.control)
# Summarize the results
print(model)

pred = predict(model, newdata=x_test)
confusionMatrix(data=pred, y_test)
```

Logistic Regresion with 10 fold cross validation and upsampling

```r
# Define training control for upsampling
set.seed(123)
train.control2 <- trainControl(method = "repeatedcv",
                              number = 10, repeats = 1, sampling = "up")
# Train the model
model2 <- train(FraudFound_P ~., data = claims, method="glm", family="binomia
l",
               trControl = train.control2)
```

```r
# Summarize the results
print(model2)

pred = predict(model2, newdata=x_test)
confusionMatrix(data=pred, y_test)
```

Random Forest with 5 fold cross validation and upsampling

```r
set.seed(123)
train.control3 <- trainControl(method = "repeatedcv",
                               number = 5, repeats = 1, sampling = "up")
mtry <- sqrt(ncol(claims))
#tunegrid <- expand.grid(.mtry=mtry)
rf_default <- train(FraudFound_P~.,
                    data=claims,
                    method='rf',
                    #metric='Accuracy',
                    #tuneGrid=tunegrid,
                    ntree=10 ,
                    trControl=train.control3)
print(rf_default)

pred = predict(rf_default, newdata=x_test)
confusionMatrix(data=pred, y_test)
```