PROJECT REPORT

ISEN 613 - ENGINEERING DATA ANALYSIS

BY

ARINDHAM BAISYA-527001456

SAURABH KUMAR JAIN-527002462

NAGA VENUGOPAL GANGARAJU-627008152

YASH ATULBHAI MEHTA-228002156

THE TEXAS A&M UNIVERSITY

INDEX

Executive Summary

One of the major aims of this project with a complex dataset was to provide a detailed analytical insight whether a car insurance claim would be made for a particular car or not. It included with what accuracy we could predict including all the necessary parameters in a model, which we deemed, fit.

The dataset had a lot of missing values and if one takes an image of dataset having the complete data under attributes columns, it results in roughly 25% of data in hand, which brings in very less data to work on. So, the first step was to replace the missing data and fill it in with approximations based on the present values. Once we had the new dataset, we were ready to run different models on this. The data manipulation helped us in running the model in a very less time i.e. by helping the model to run those predictors on important levels.

After the data filtering was accomplished, we fitted several models on this revised data. But after running various models we found out that there were no attributes in the given data that explained the variability in response significantly. That is, it was difficult to determine by just looking at the attributes whether that observation corresponded to claim being made or not. But due to the computational power of the software used, when we used all the attributes to predict the class of response, we got a few somewhat satisfactory models that determined the class of response.

The best model that we obtained to predict the class of response gave out a reasonable accuracy when compared to the kind of dataset provided. However, here a tradeoff must be made between getting overall high accuracy rate, i.e., determining as truly as possible the class of each observation. In doing so the model would often fail to predict the class of observations where claim was made and still have a high accuracy because of the nature of the provided dataset. On the other hand, the model can more correctly predict the class of observations corresponding to claim being made but on the cost of a lower accuracy, i.e., even when the likelihood of claim being made was low the model would still assign the class of the observation as claim being made to be on the safer side. We decided to side with predicting the observations corresponding to claim being made more accurately at the cost of achieving overall lower accuracy.

Based on the model selected and the decision made, we can now predict the class of observations with an accuracy of 73% and the observations corresponding to claim being made with accuracy of about 36%. These parameters can be changed according to what trade-off we decide to make.

Contributions - Venugopal did data preprocessing, which includes undersampling the data so that our models could run efficiently on the revised dataset with a drastic reduction in computation time. He fitted the random forest model on this dataset. Yash helped in the data preprocessing too using several complex packages like Rose, Mice and many more.Arindham and Yash ran several models on the revised data set with Arindham contributing towards the models SVM, SVC, LDA and QDA. Yash contributed towards the logistic model and the analysis of these models using Cross Validation and finding out Area under the Roc curve by the ROC plots. Saurabh completed the initial and final project report draft, which included the executive summary, technical summary, selection of the three best models, their comparison, reporting the test results and model improvements along with posting the final modified code for our model.

Technical Analysis

The dataset has a lot of missing values and if one takes an image of dataset having the complete data under predictor columns, it results in roughly 25% of data in hand, which brings in very less data to work on. Therefore, the first step was to impute the missing data and we used predictive mean matching in the mice package. Once we had the new dataset, we were ready to run different models on this. However, the crude reality hit us in terms of computation time, which was so long that it took several hours for some. The next issue we dealt with was reducing the computation time. When we observed the levels of factor type predictors, there were too many and if removed from, the main model resulted in model running in a few minutes. We reduced the levels of predictors Blind_Make, Blind_Model and Blind_Submodel by first sorting the levels based on frequency of response and frequency of the level being observed in the dataset separately. Both the frequencies columns were ranked in ascending order and a threshold of 1% was used such that any frequency upto 1% had a unique rank and below 1% ranking was done based on a range of 0.25% i.e. (1%,0.75%]->common rank, (0.75%,0.5]->common rank etc. Finally, both these frequency columns were ranked together by using frequency of response as a pivot in the descending order and converting ranks to categorical data to be used a factor level. This resulted in a drastic reduction in levels. The data manipulation helped us in running the model in a very less time i.e. by helping the model to run those predictors on important levels.

Logistic Regression Model

Once the data was imputed and the levels reduced to help reduce the computation time, the other main issue facing the logistic model was the imbalance in the dataset, which was in the form of only 720 of the 100000 observations being identified as claimed. This imbalance caused the models to give results where almost no claims were identified. To tackle this problem, we use the undersampling and oversampling techniques that are incorporated in the ROSE package. Using the rose function, we undersampled the observation where no claim was made and oversampled the observations where claims were made. This led to almost a 50-50 split in the obtained dataset in terms of response.

Moving ahead with fitting the model, the first thing that we did was to split the original imputed data with reduced levels into training and test data. The split ratio was 80:20, where 80% of the observations were allotted as training data and the rest 20% as testdata. This was done to ensure that for validating the results, the data used was as close to the original dataset provided and not the one tempered by using the rose function.

We then applied the rose function to the train data part to remove the imbalance and fit the logistic model on that data and checked for important variables. After fitting the model and looking at the p-values, we saw that 4 variables household id, vehicle no., var7 and nvar4 were insignificant. However, since the p-value was still significant in our viewpoint, we kept the predictors in our model. We got a confusion matrix, which gave us the training error rate as 33.815% and a training accuracy of 66.185%. The sensitivity obtained here was 53.73%.

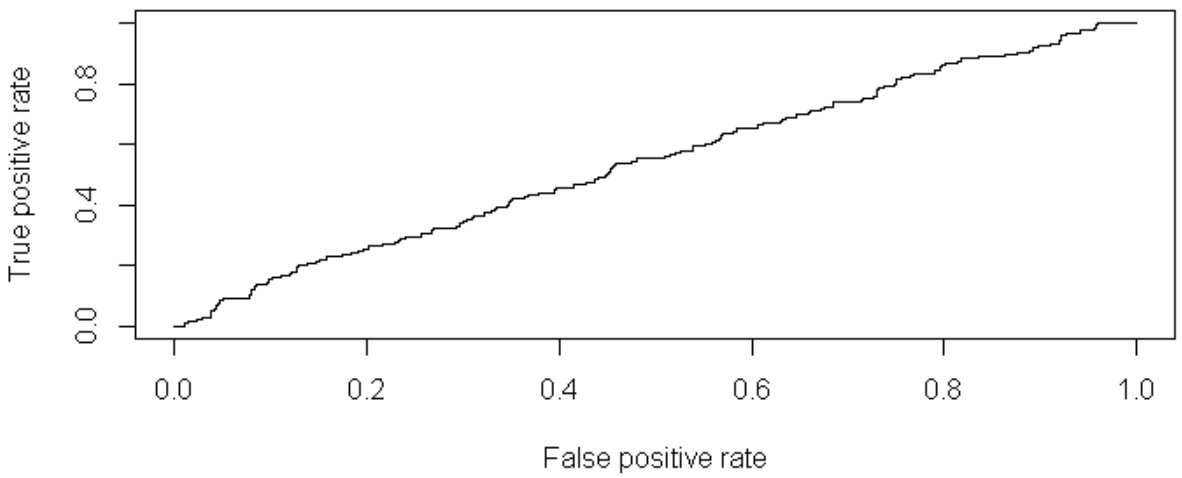
glm.pred	0	1
0	52639	266
1	26786	309

To validate our model we used the validation set approach. We did this by fitting our model on the 20% of the original imputed data with reduced levels and comparing the results against the actual values. Here, we got an error rate of 33.15% and an accuracy of 66.86%. The sensitivity obtained is about 37.93%. The confusion matrix obtained here is

```
> table(glm.pred,testdata$Claim_Amount)

glm.pred     0     1
       0 13280    90
       1  6575    55
>
```

Here the Roc curve, which was obtained, is:



The area under the curve obtained here is 54.44%.

To conclude here we would like to say that the logistic model did not perform satisfactorily on our dataset. The values obtained for the accuracy and sensitivity were poor. We saw major improvements in our next two methods SVM and Random Forest, which we discuss in the subsequent part of this report.

Support Vector Machines

After performing logistic regression, we were not happy with the results we got for the test MSE along with the area under the roc curve, which was just 54.5%. In this method, our major aim was to reduce the Test MSE and ensure that the results, which were obtained, were better than the logistic model. In this model we started by dividing the data in the 80% training and 20% test data. Now since our dataset was imbalanced where only 720 actual claims were filed out of 100000 data entries, we then under-sampled the data by a factor of 1.1.

To tackle the imbalance in the data, we created a new dataset by undersampling the observations corresponding to the response as no claim being made. This resulted in a dataset where the split in terms of response was 55:45 where 55% of the observations corresponded to no claim being made and 45% of the observations correspond to claim being made. Without doing this our model would not predict claims being made accurately and will give overall poor results where it will predict majority of responses as no claim being made. The only drawback we felt here was that the under-sampling could cause was that removing observations might cause the training dataset to lose important information pertaining to the majority class. However, the results, which we obtained, were quite satisfactory which overshadowed our drawbacks.

The prime reason for under-sampling was that most of the algorithms were struggling due to the unequal distribution in various dependent variables. Also because of the imbalance, it was causing most of the existing classifiers to be biased towards the majority class in that column. In addition, most of the methods assume that the errors obtained from different classes have the same cost and they assume a balanced class distribution. Most of the models and methods aim for higher accuracy where they aim to minimize the overall error to which the minority classes contributes very little.

We later on moved to various different models like the SVC, LDA and the QDA models all of which performed poorly. However, the model, which performed slightly better than the logistic regression model here, was the Support Vector Machine Method.

We fitted a SVM model on the dataset where we took gamma as 0.1, (we had tried different values of 0.5,1,2,3 and 4) and our cost as five (we had tried different values of (0.1, 1, 10 and 100) and the kernel as polynomial. We tried various values for the gamma as we had tried a radial kernel to start with in the SVM Modelling. The radial kernel was not fitting the data properly because of which we opted for the polynomial kernel. The polynomial kernel was fitting the data in the model much better than the radial kernel.

After fitting the model on the training data, we tested on the remaining 20% test data to get a confusion matrix, which gives us:

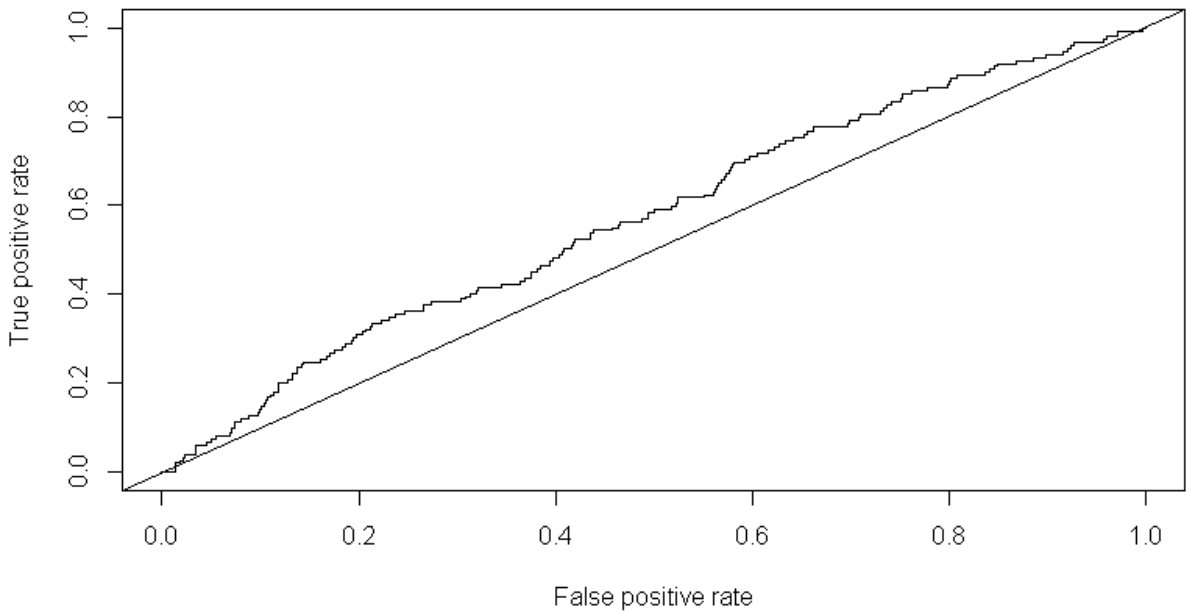
Accuracy = 61.475%

Error rate = 38.52%

Sensitivity = 45.63%

```
> table(predict =ypred , truth=test$C_Claim )
       truth
predict    No   Yes
    No  12227    81
    Yes  7624    68
```

We than plotted these results in the form of a roc curve. The roc curve is given below.



The area under the curve for this roc plot is obtained as 57.34%.

We successfully ran the 10-fold k cross validation on this SVM model where we got an error rate of 41.160%. Hence an accuracy of 58.84%.

To conclude here we would say that the results obtained for the SVM method were slightly better than those obtained from the Logistic model. The accuracy and sensitivity values along with the area under the curve was slightly better for the SVM model than the Logistic, LDA and QDA models.

Random Forest

After performing SVM, we started with KNN and other methods, which performed poorly. We were in the search of a method, which would give us the best possible accuracy, and we found just that while performing random forest on our dataset.

One of the major issues or challenges we faced while making the random forest model was the highly unbalanced dataset. It is due to this unbalance in the dataset, the model that we would have made, would make a prediction with a high bias towards the major class. For example, without balancing the data when we ran the random Forest model on the data, it did not predict the response as claimed at all but gave a good overall accuracy. Since our main aim here is to reduce the number of instances where insurance is claimed we balanced the data, which although reduced the overall accuracy, did predict the instances where insurance was claimed somewhat satisfactorily.

To overcome this issue, we used the down sampling method with a ratio of 5:1. Here, this resulted in a dataset where the split in terms of response was 5:1 where 83.33% of the observations corresponded to no claim being made and 16.67% of the observations correspond to claim being made. This was much better than the original ratio where only 720 out of a 100000 observations corresponded to the claim being made. Furthermore, we used this image of the data to train the model by taking stratified sampling.

We went about this model by first converting the Claim_Amount to factors and then we went on to convert the data into the testdata and train data. It was an 80:20 split. The proximity is kept true in the random forest model to reduce the intensive computation load on the computers. We have used the strata factor in the random forest model to account for the stratified sampling and the sampsize as the sizes of the sample to draw from the dataset.

Here since we are doing classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata. Here they are drawn from the undersampled data. We have used stratified sampling here, as Stratified-sampling aims at splitting one data set so that each split are similar with respect to something. In our classification, setting it is chosen to ensure that the train and test sets have approximately the same percentage of samples of each class as the complete set. Now since there are various major classes in our dataset, then stratified sampling may yield a different and better class distribution in the train and test sets than what random sampling may yield.

After fitting the model, we ran the model on the 20% held out data to get a confusion matrix with the respective values of accuracy and the sensitivity.

Accuracy % = 74.73%

Sensitivity % = 33.55%

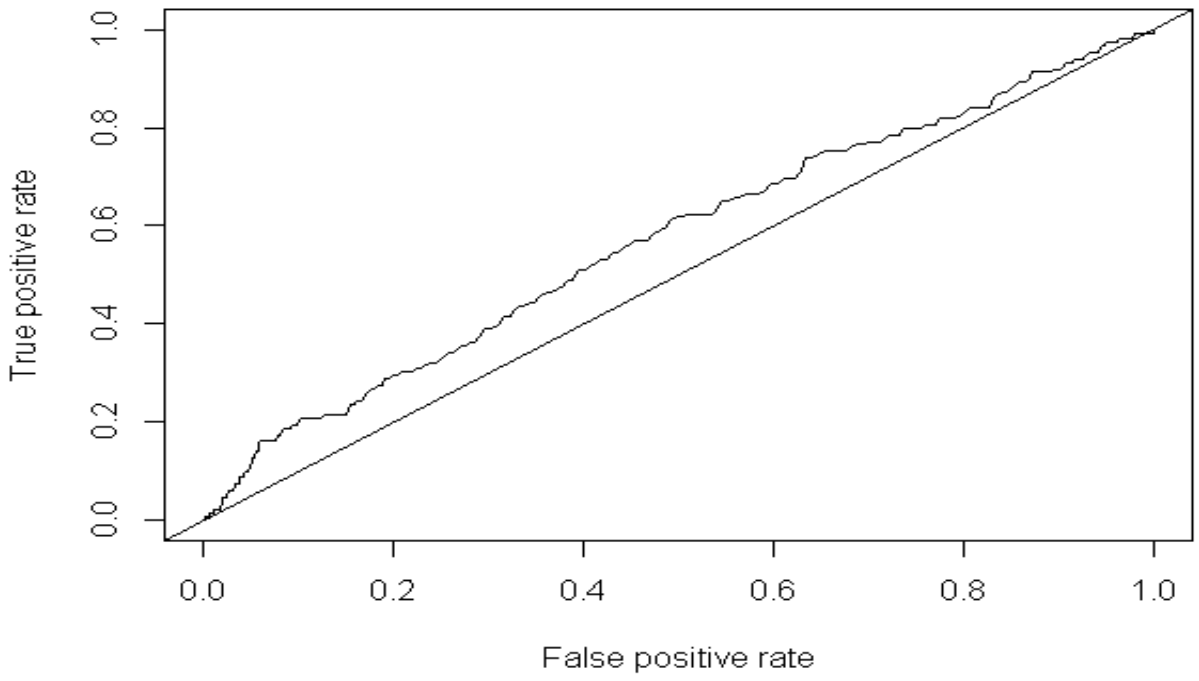
```
> confusionMatrix
         observed
predicted    No   Yes
      No  14896    99
      Yes  4955    50
> confusionMatrix[2,2]/sum(confusionMatrix[,2]) #sensitivity
[1] 0.3355705
> (confusionMatrix[1,1]+confusionMatrix[2,2])/(sum(confusionMatrix[,1])+sum(confusionMatrix[,2])) #accuracy
[1] 0.7473
```

We then performed the 5-fold cross validation with the model on the stratified sampled dataset to get an error rate of

Error rate % = 19.80%

```
> rf.cv=rfcv(ids[-33],ids$Claim_Amount,cv.fold=5,strata=ids$Claim_Amount)
> rf.cv$error.cv
       32        16         8         4         2         1
0.1727963 0.2227087 0.2221249 0.2352598 0.1684180 0.1666667
```

The roc curve obtained for the model is given by



The area under the curve is obtained is 57.13%

A Brief Comparison

The three best models chosen from the pool of models, which we used to fit the train data with, in chronological order, are:

1) Random Forest

2) Support Vector Machine

3) Logistic model

	The Test MSE for the Validation Set Approach	The Test MSE for the Cross-Validation Approach	Area under the ROC Curve
Random forest	25.27%	19.80%	57.13%
Support Vector machine	38.52%	41.160%	57.34%
Logistic Model	33.15%	39.78%	54.44%

Hence, we see here that the Random Forest performs much better than the Support Vector Machine Model and the Logistic Model. The results obtained for the Support Vector Machine model and the Logistic Model are quite comparable but far worse than that obtained for the Random Forest Model. Thus, we have chosen the random forest model as our model to fit the dataset provided.

Test Results

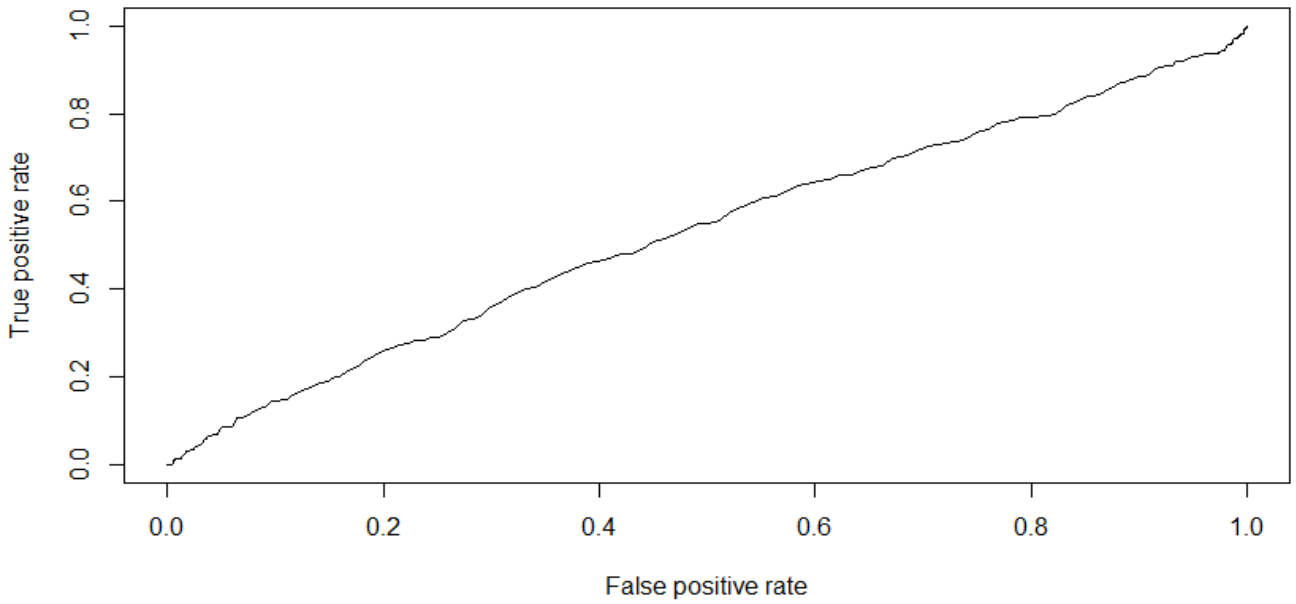
After running the trained model on the test data, we got the following results:

```
> confusionMatrix
          observed
predicted    No   Yes
      No  30148   197
      Yes 19485   170
> confusionMatrix[2,2]/sum(confusionMatrix[,2]) #sensitivity
[1] 0.4632153
> (confusionMatrix[1,1]+confusionMatrix[2,2])/(sum(confusionMatrix[,1])+sum(confusionMatrix[,2]))
uracy
[1] 0.60636
```

Accuracy% = 60.636%

Sensitivity% = 46.32%

The Roc curve for the trained model fitted on the test data is



The area under the curve obtained is 52.98%.

We could have achieved an accuracy rate of more than 99% by simply fitting a simple model that made the same prediction of No claim being made every time we made a prediction. We decided here that even sensitivity is important here as in the case of the car insurances the amount claimed is generally much higher than the amount of premium that a customer would pay so it is a classic case of trade-of between sensitivity and accuracy.

Model Improvement

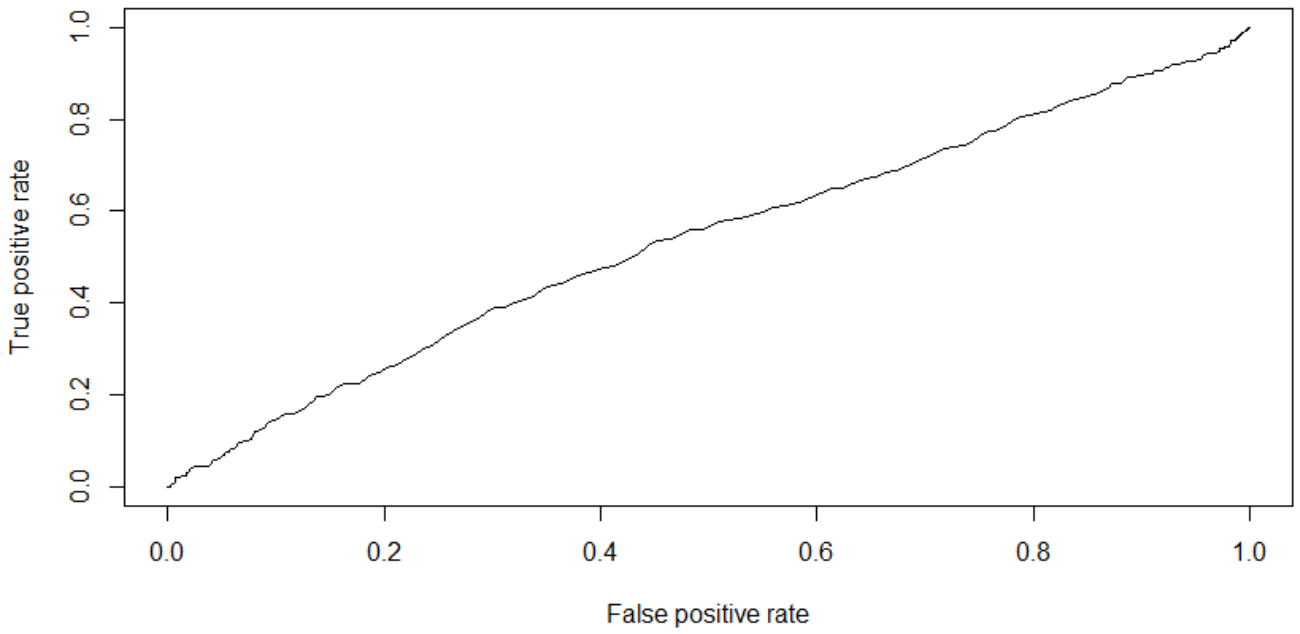
In the training model, we got an accuracy rate of 74.7% and a sensitivity rate of 33.5% by using an undersampling ratio of 5:1. After running this fitted model on the provided test data our accuracy decreased but our sensitivity of the model increased. Now the model improvement, which we tried here, was to increase the undersampling ratio to 20:1 and increase our accuracy rate of our model. Using this undersampling ratio and running the trained model on the test data we got the following results:

```
> confusionMatrix
          observed
predicted    No   Yes
      No  36081   238
      Yes 13552   129
> confusionMatrix[2,2]/sum(confusionMatrix[,2]) #sensitivity
[1] 0.3514986
> (confusionMatrix[1,1]+confusionMatrix[2,2])/(sum(confusionMatrix[,1])+sum(confusionMatrix[,2])) #accuracy
[1] 0.7242
```

Accuracy = 72.42%

Sensitivity = 35.149%

The ROC curve for the improved model with a higher undersampling ratio is given below



The area under the curve is given by 53.73%.

Here, we have tried to improve the accuracy of the trained model on the test data by increasing the undersampling ratio. This does increase the accuracy but also lowers down the sensitivity of the model. Thus, we have tried to achieve an optimal trade-off between the two in this improved model.

Final Modified Code

```
set.seed(1)
id=read.csv("revdata.csv",header=T)[,2:35]
test_data=read.csv("test_data.csv",header=T)[,2:35]
str(id)
str(test_data)
id<-id[,-1]
test_data<-test_data[,-1]
str(id)
str(test_data)
id$Claim_Amount=as.factor(ifelse(id$Claim_Amount>0,"Yes","No"))
test_data$Claim_Amount=as.factor(ifelse(test_data$Claim_Amount>0,"Yes","No"))

levels(test_data$Blind_Make) <- levels(id$Blind_Make)
levels(test_data$Blind_Model) <- levels(id$Blind_Model)
levels(test_data$Blind_Submodel) <- levels(id$Blind_Submodel)
levels(test_data$Cat1) <- levels(id$Cat1)
levels(test_data$Cat2) <- levels(id$Cat2)
levels(test_data$Cat3) <- levels(id$Cat3)
levels(test_data$Cat4) <- levels(id$Cat4)
levels(test_data$Cat5) <- levels(id$Cat5)
levels(test_data$Cat6) <- levels(id$Cat6)
levels(test_data$Cat7) <- levels(id$Cat7)
levels(test_data$Cat8) <- levels(id$Cat8)
levels(test_data$Cat9) <- levels(id$Cat9)
levels(test_data$Cat10) <- levels(id$Cat10)
levels(test_data$Cat11) <- levels(id$Cat11)
levels(test_data$Cat12) <- levels(id$Cat12)
levels(test_data$NVCat) <- levels(id$NVCat)


str(id)
str(test_data)

b<-as.data.frame(table(id$Claim_Amount))

# subample the super majority class to reduce the ratio to 20:1 ~ undersampling
maj <- id[id$Claim_Amount=="No",]
subid <- sample(1:dim(maj)[1], size=b[2,2]*20, replace=FALSE)
ids <- rbind(id[id$Claim_Amount=="Yes",],maj[subid,])
table(ids$Claim_Amount)
summary(ids)
```

```
a<-as.data.frame(table(ids$Claim_Amount))

library(randomForest)
rf.df=randomForest(Claim_Amount~.,data=ids,importance=T, prox=TRUE,
strata=ids$Claim_Amount, sampsize=c(a[2,2],a[2,2]))

# make prediction on the test set
test.pred1 <- predict(rf.df,newdata=test_data)
# compare it to the actual outcome
confusionMatrix <- table(predicted=test.pred1,observed=test_data$Claim_Amount)
confusionMatrix
confusionMatrix[2,2]/sum(confusionMatrix[,2]) #sensitivity
(confusionMatrix[1,1]+confusionMatrix[2,2])/(sum(confusionMatrix[,1])+sum(confusionMatrix[
,2])) #accuracy


#ROC plot and AUC
library(ROCR)
library(pROC)
rocplot =function (pred , truth , ...){ predob = prediction (pred , truth)
perf = performance (predob , "tpr", "fpr")
plot(perf ,...)}
testpred2=predict(rf.df,newdata = test_data,type = "prob")
rocplot(testpred2[,2],test_data$Claim_Amount)
abline(0,1)
auc(test_data$Claim_Amount,testpred2[,2])
```

References

- https://stats.stackexchange.com/questions/168415/random-forest-in-r-using-unbalanced-data.
- https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf
- https://stats.stackexchange.com/questions/6067/does-an-unbalanced-sample-matter-when-doing-logistic-regression
- https://www.r-bloggers.com/handling-class-imbalance-with-r-and-caret-an-introduction/
- https://onlinelibrary.wiley.com/doi/full/10.1111/coin.12123
- http://www.cs.ox.ac.uk/people/vasile.palade/papers/Class-Imbalance-SVM.pdf
- https://pdfs.semanticscholar.org/f302/23db2330df3e03fd7be2b79d778fa4f5efff.pdf
- https://www.quora.com/Why-does-knn-get-effected-by-the-class-imbalance
- https://www.researchgate.net/post/Best_preprocessing_methods_for_imbalanced_data_in_classification_algorithms
- https://www.sciencedirect.com/science/article/pii/S0031320307005006
- www.socr.umich.edu/people/dinov/2017/Spring/.../20_PredictionCrossValidation.html
- https://www.jeremyjordan.me/imbalanced-data/
- An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) by Gareth James.