“Fraudolent Credit Card” Project

written by Samuel Beqiri

Contents

Introduction and Overview

The dataset contains transactions made by credit cards in September 2013 by card-holders in two-day period. Of 284,807 valid transactions, 492 are listed as fraudulent. The variable ‘Time’ contains the seconds elapsed between each transaction and the first transaction in the dataset. The variable ‘Amount’ is the transaction value. The variable ‘Class’ is the response variable where 1 is a case of fraud and 0 is a valid transaction.

Dataset and Eploratory Analysis

For the initial set up the following packages will be installed.

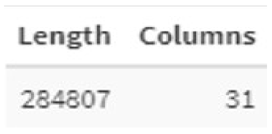
```
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(kableExtra)) install.packages("kableExtra")
if(!require(tidyr)) install.packages("tidyr")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(stringr)) install.packages("stringr")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(gbm)) install.packages("gbm")
if(!require(dplyr)) install.packages("dplyr")
if(!require(caret)) install.packages("caret")
if(!require(xgboost)) install.packages("xgboost")
if(!require(e1071)) install.packages("e1071")
if(!require(class)) install.packages("class")
if(!require(ROCR)) install.packages("ROCR")
if(!require(randomForest)) install.packages("randomForest")
if(!require(PRROC)) install.packages("PRROC")
if(!require(reshape2)) install.packages("reshape2")
if(!require(corrplot)) install.packages("corrplot")
```

Load the following packages using the library() function.

```
library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(ggplot2)
library(gbm)
library(caret)
library(xgboost)
library(e1071)
library(class)
library(ROCR)
library(randomForest)
library(PRROC)
library(reshape2)
library(corrplot)
```

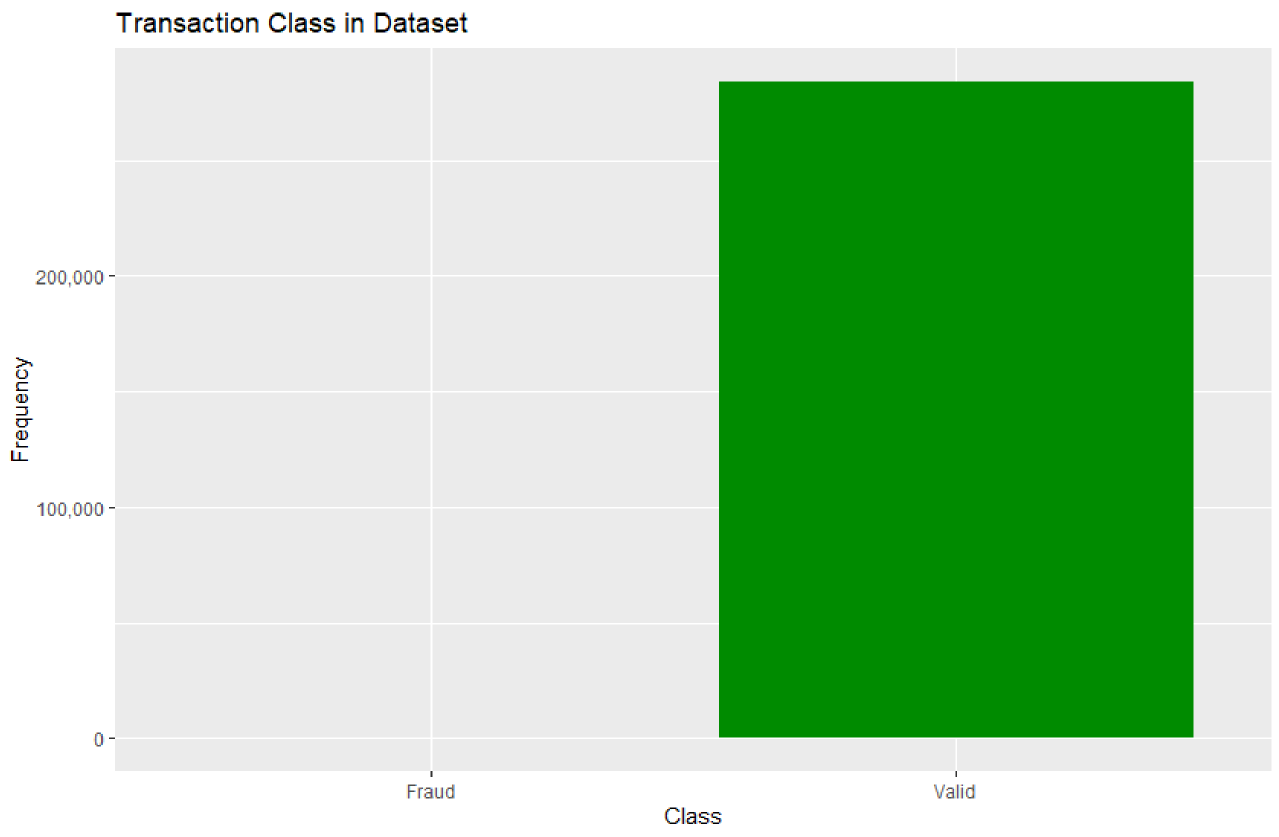
To better understand the data we present a data dictionary of the 31 variables in the dataset.

- Time - the number of seconds elapsed between this transaction and the first transaction in the dataset
- V1-V28 result of a PCA Dimensionality reduction to protect user identities and sensitive features
- Amount - the dollar value of the transaction
- Class - 1 for fraudulent transactions, 0 for Normal transactions



Length	Columns
284807	31

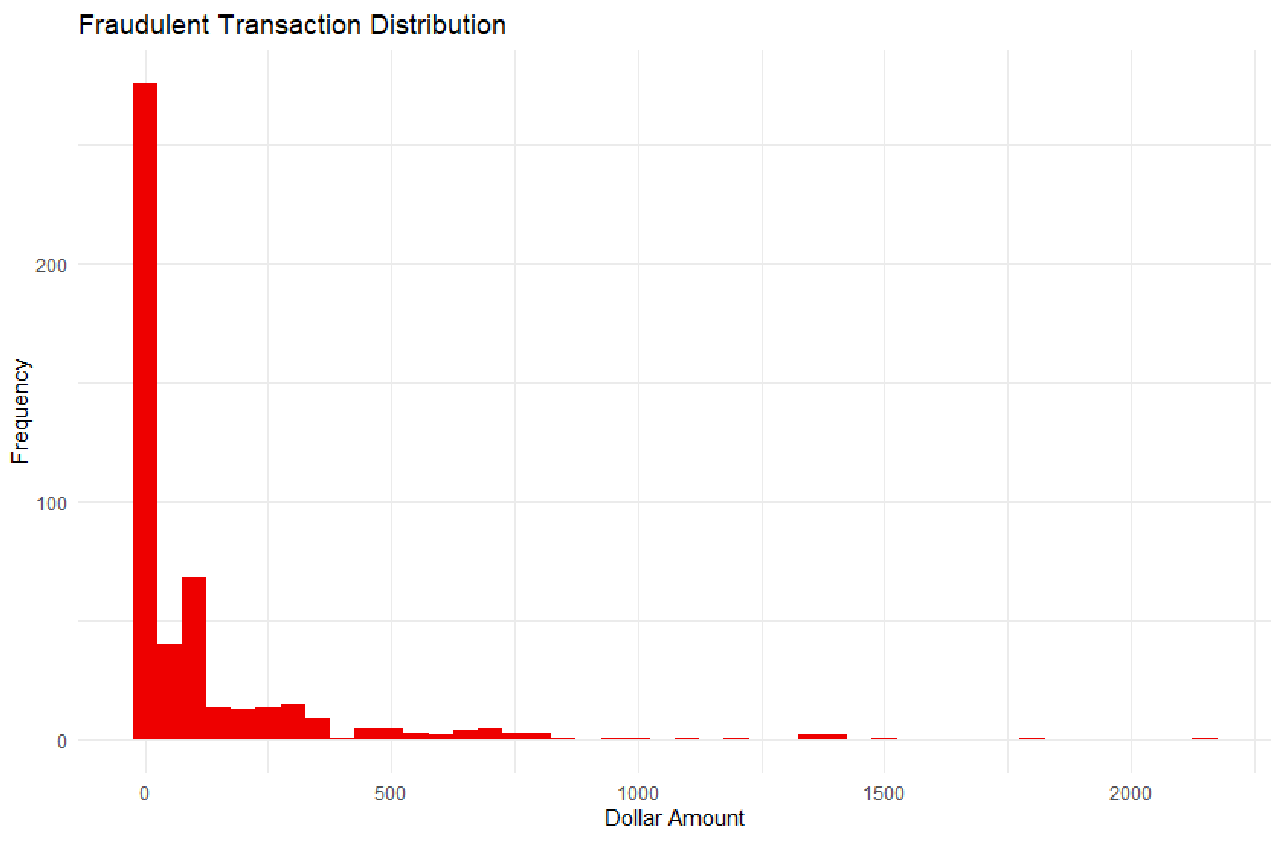
We want to see how many transactions are fraudulent compared to how many are Normal. 0 is defined as a Normal transaction, and 1 is defined as a fraudulent transaction. To see the data, we plot a bar graph of the frequency of fraud versus valid credit card transactions.



We see that the vast majority of transactions are valid (99.828%).

Fraudulent Transaction Distribution

We want to investigate the amount of fraud transactions in dollars. Here we plot all the fraudulent transaction by amount. This plot shows a massive skew toward transactions under $100.



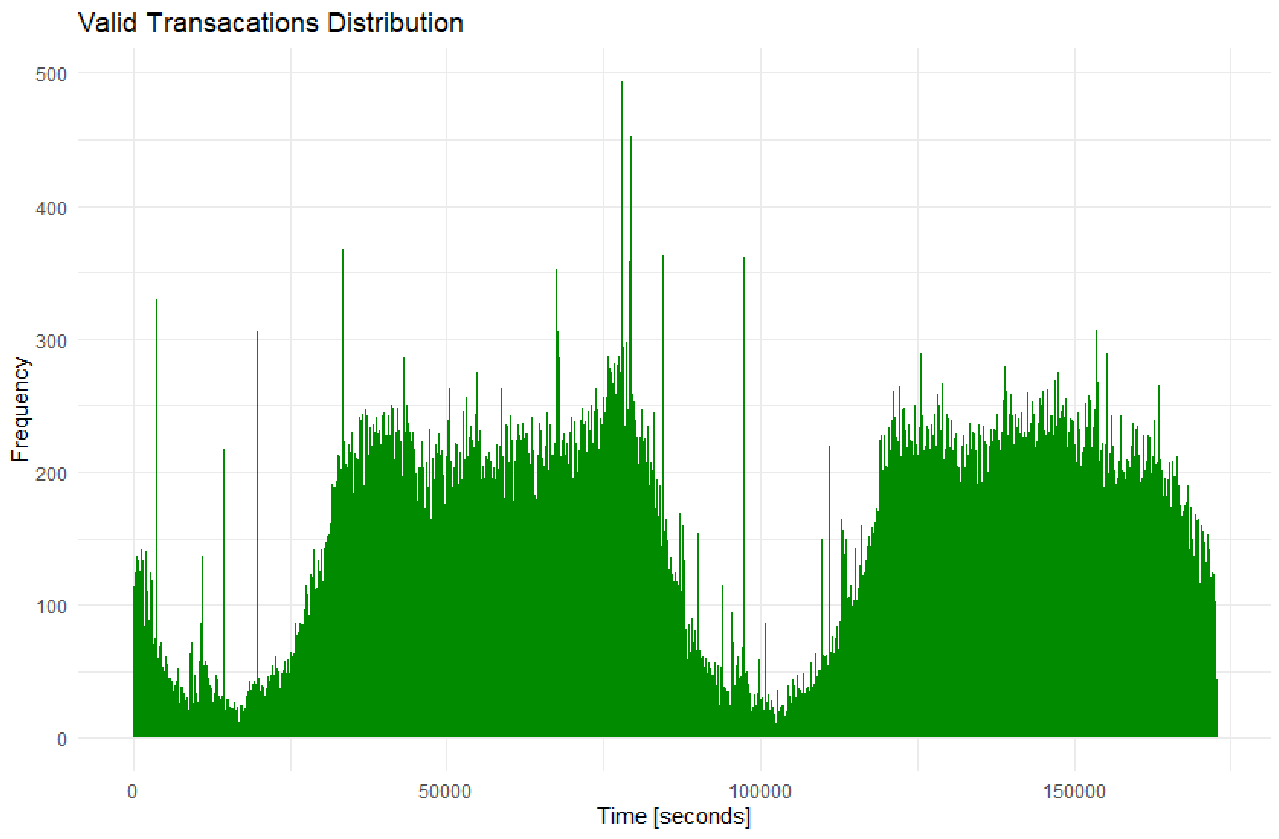
We can also investigate what are the most common valid transactions in the dataset.

An interesting observation is that $1 is the most common fraudulent and valid transaction. In fact the chance of a transaction of $1 being fraud is almost five times higher than other transactions in the data set. Another very interesting observations is that a transactions of $99.99 is the 98th most common valid transaction with 303 transactions, but it is also the second most common fraudulent transaction with 27 times count. This means that ~9% of $99.99 transactions in the data set are fraudulent!

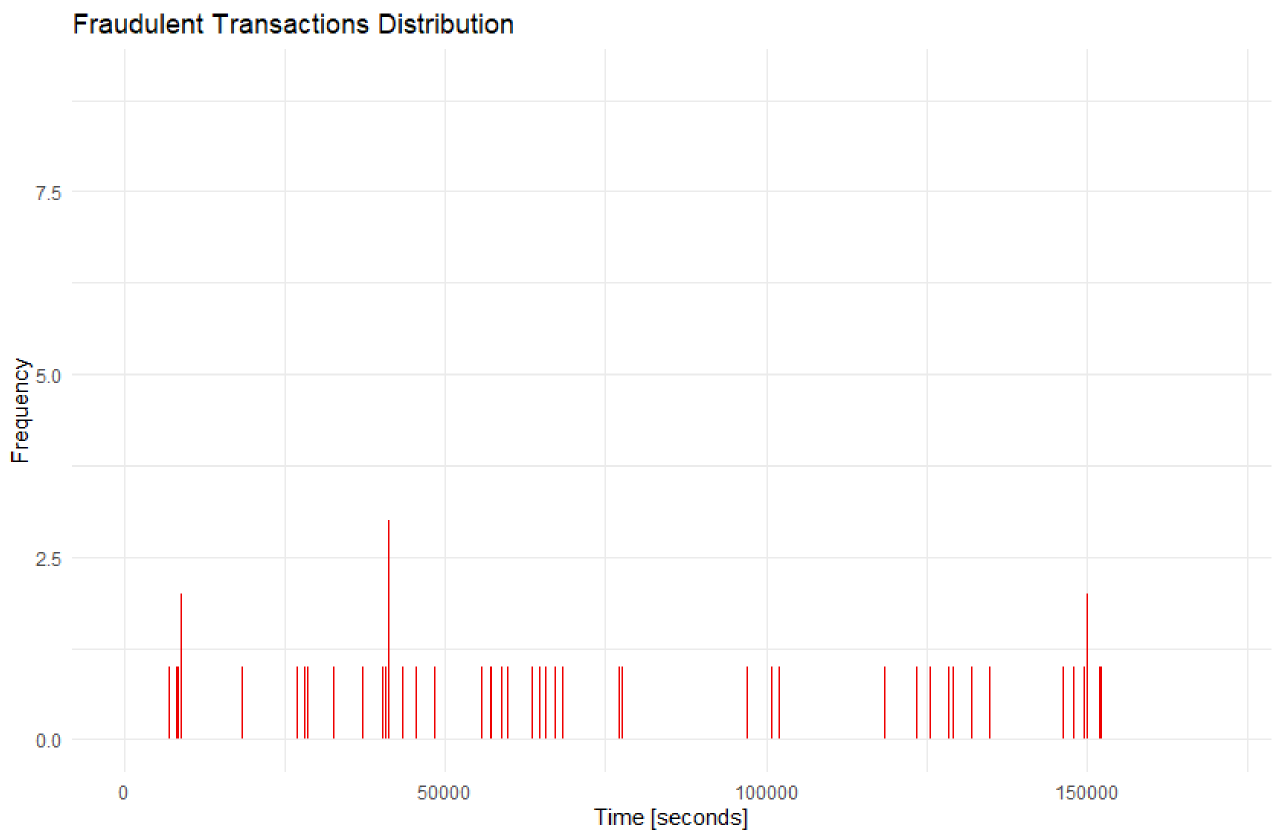
Amount	count
1.00	113
0.00	27
99.99	27
0.76	17
0.77	10
0.01	5
2.00	4
3.79	4
0.68	3
1.10	3

Transactions Distribution Over Time

We can plot a distribution of valid transactions over time. This plot has a clear episodic distribution. This makes sense since a day has 86,400 seconds, which is the approximate period of this distribution.
The punchline is that most transactions occur during the day, while fewer transactions occur at night. There is a clear spike of outlier transactions near the trough of the graph. We surmise that these spikes correlate to automated transactions that are processed a little before the close of midnight or shortly after midnight. An example of automated transactions would be monthly recurring bills set to autopay.



Similarly, to the distribution of valid transactions, we can plot the distribution of fraudulent transactions over time. The lack of any clear episodic distribution indicates that fraud can occur at any time.



To note: Without performing Fourier analysis (such as a Fast Fourier Transform) on this data, we do not know with certainty that fraudulent transactions are non-episodic. This analysis is beyond the scope of this project, and the frequency distribution plotted above will suffice to show that fraudulent transactions are not episodic and can occur at any point in time.

Correlation Matrix

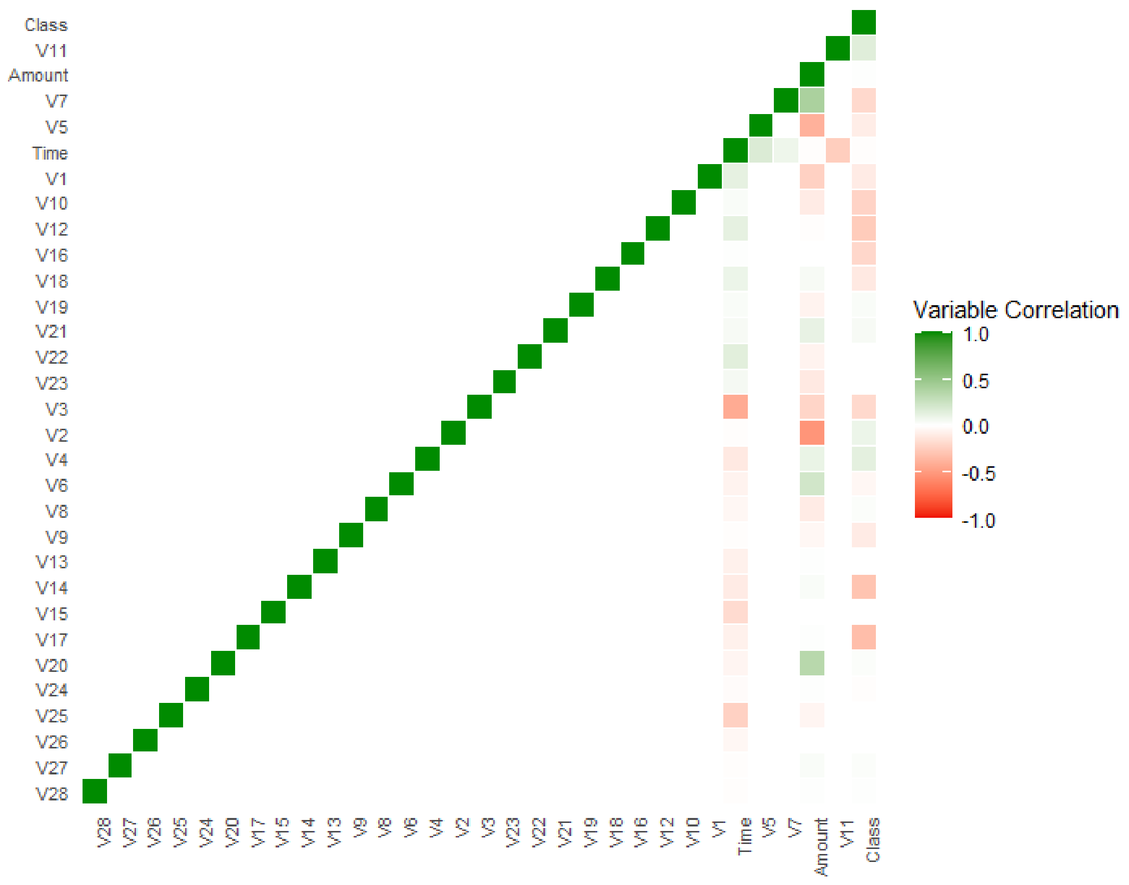
We want to calculate the correlation between the variables and graph them. We first design a correlation matrix.

```
get_lower_triangle<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}
We obtain the upper triangle of the correlation matrix.
get_upper_triangle <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}
reorder_cormat <- function(cormat){
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}
corr_matrix <- round(cor(mydataset ),2)
corr_matrix <- reorder_cormat(corr_matrix)
```

Here is a matrix of the correlation between the 31 distinct variables.

	V28	V27	V26	V25	V24	V20	V17	V15	V14	V13	V9	V8	V6	V4	V2	V3	V23	V22	V21	V19	V18	V16	V12	V10	V1	Time	V5	V7	Amount	V11	Class
V28	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00	0.00	0.01	0.00	0.01
V27	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00	0.00	0.03	0.00	0.02
V26	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.04	0.00	0.00	0.00	0.00	0.00
V25	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.23	0.00	0.00	-0.05	0.00	0.00
V24	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.02	0.00	0.00	0.01	0.00	-0.01
V20	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.05	0.00	0.00	0.34	0.00	0.02
V17	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.07	0.00	0.00	0.01	0.00	-0.33
V15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.18	0.00	0.00	0.00	0.00	0.00
V14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.10	0.00	0.00	0.03	0.00	-0.30
V13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.07	0.00	0.00	0.01	0.00	0.00
V9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00	0.00	-0.04	0.00	-0.10
V8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.04	0.00	0.00	-0.10	0.00	0.02
V6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.06	0.00	0.00	0.22	0.00	-0.04
V4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.11	0.00	0.00	0.10	0.00	0.13
V2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00	0.00	-0.53	0.00	0.09
V3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.42	0.00	0.00	-0.21	0.00	-0.19
V23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	-0.11	0.00	0.00
V22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	-0.06	0.00	0.00
V21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.11	0.00	0.04
V19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	-0.06	0.00	0.03
V18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.04	0.00	-0.11
V16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	-0.20
V12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.12	0.00	0.00	-0.01	0.00	-0.26
V10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.03	0.00	0.00	-0.10	0.00	-0.22
V1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.12	0.00	0.00	-0.23	0.00	-0.10
Time	-0.01	-0.01	-0.04	-0.23	-0.02	-0.05	-0.07	-0.18	-0.10	-0.07	-0.01	-0.04	-0.06	-0.11	-0.01	-0.42	0.05	0.14	0.04	0.03	0.09	0.01	0.12	0.03	0.12	1.00	0.17	0.08	-0.01	-0.25	-0.01
V5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	1.00	0.00	-0.39	0.00	-0.09
V7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	1.00	0.40	0.00	-0.19
Amount	0.01	0.03	0.00	-0.05	0.01	0.34	0.01	0.00	0.03	0.01	-0.04	-0.10	0.22	0.10	-0.53	-0.21	-0.11	-0.06	0.11	-0.06	0.04	0.00	-0.01	-0.10	-0.23	-0.01	-0.39	0.40	1.00	0.00	0.01
V11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.25	0.00	0.00	0.00	1.00	0.15
Class	0.01	0.02	0.00	0.00	-0.01	0.02	-0.33	0.00	-0.30	0.00	-0.10	0.02	-0.04	0.13	0.09	-0.19	0.00	0.00	0.04	0.03	-0.11	-0.20	-0.26	-0.22	-0.10	-0.01	-0.09	-0.19	0.01	0.15	1.00

Further, we can plot the correlation. Notice how all the variables V1-V28 have very low correlation coefficients among each other, and especially low correlation with the ‘Class’ feature. This was already expected since the data was processed using PCA.



We established that fraud does not appear to coincide with a specific time of day, so the ‘Time’ variable will be removed from the dataset.

```
mydataset $Class <- as.factor(mydataset $Class)
mydataset  <- mydataset  %>% select(-Time)
```

To verify the variable ‘Time’ has been removed, we can view the first six entries with the head() function

```
head(mydataset )
```

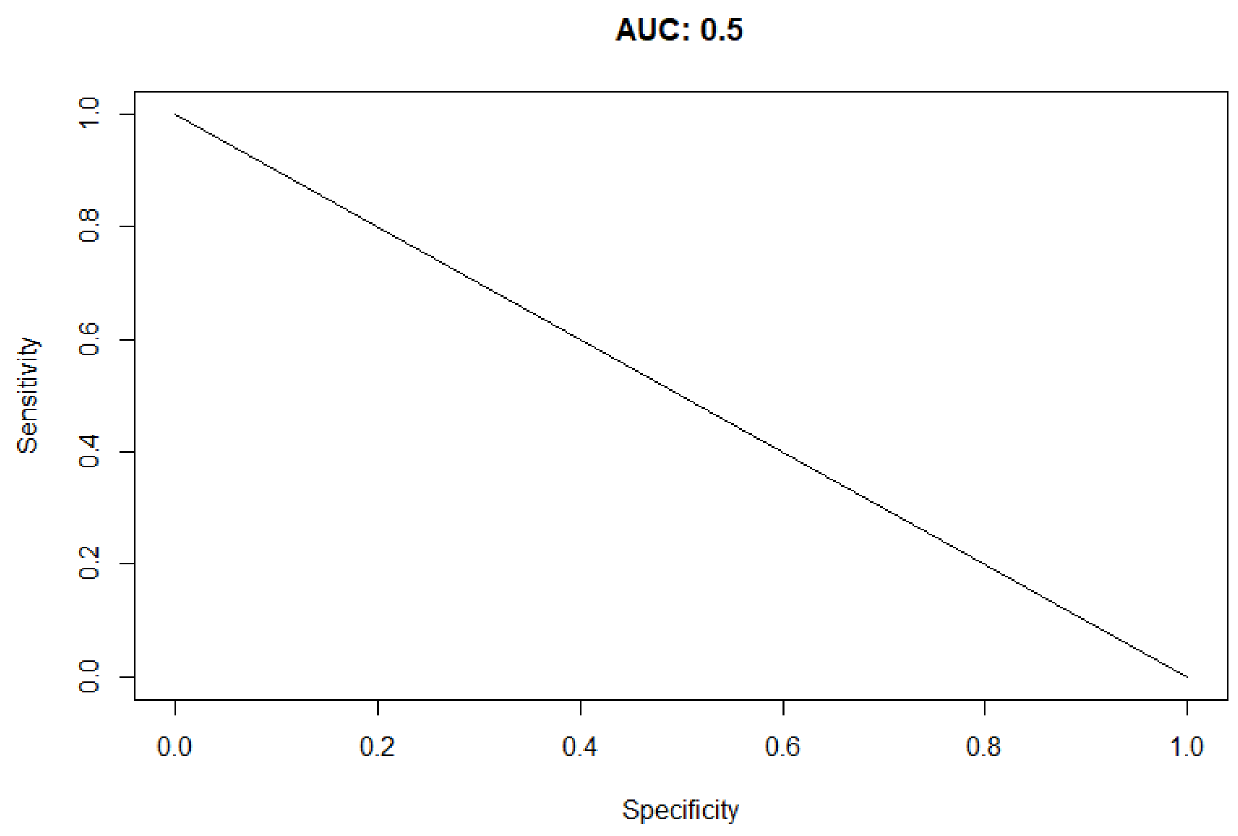
Methods and Analysis

For this report we will investigate four models: The Naive Model, the Naive Bayes Model, the K-Nearest Neighbor Model, and the Random Forest Model

Naive Model

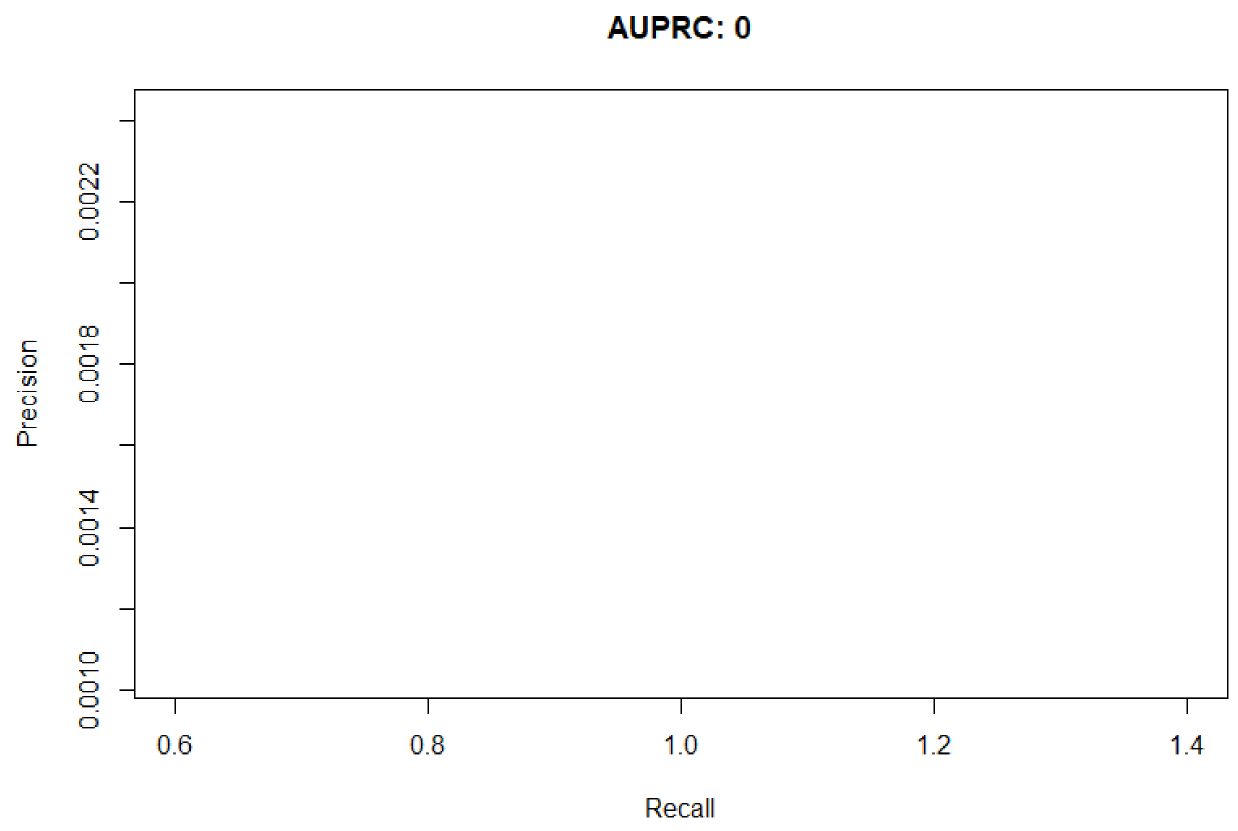
The first model we design is the Naive Model. This model makes the simple prediction that every transaction is a valid transaction and that there are no fraudulent transactions. This will serve as our first attempt in trying to better the model.

```r
set.seed(13)
#We need to create a training data set, a test dataset, and a cross validation data set.
train_index <- createDataPartition(
  y = mydataset $Class,
  p = .6,
  list = F)
#Our training set.
train <- mydataset [train_index,]
#Temporary test dataset.
test_cv <- mydataset [-train_index,]
#We partition the test dataset.
test_index <- createDataPartition(
  y = test_cv$Class,
  p = .5,
  list = F)
#The partitioned test dataset is split between a test set and a cross validation set.
test <- test_cv[test_index,]
cv <- test_cv[-test_index,]
#We remove the temporary files to create our datasets.
rm(train_index, test_index, test_cv)
#Copy the mydataset  dataframe to make the necessary changes for the baseline model.
naive_model <- data.frame(mydataset )
#We now define all transactions as valid by defining all entries in the class set as valid.
naive_model$Class = factor(0, c(0,1))
#We then make the prediction of that all entries are valid transactions.
pred <- prediction(
  as.numeric(as.character(naive_model$Class)),
  as.numeric(as.character(mydataset $Class)))
#We need to compute the Area Under the Curve (AUC) and the Area Under the Precision-Recall Curve (AUPRC).
auc_val_naive <- performance(pred, "auc")
auc_plot_naive <- performance(pred, 'sens', 'spec')
auprc_plot_naive <- performance(pred, "prec", "rec")
#We plot the AUC for the Naive Model.
#As expected, we obtain an area under the curve of 0.5.
plot(auc_plot_naive,
     main=paste("AUC:",
          auc_val_naive@y.values[[1]]))
```



No line is generated for the Area Under the Precision Recall Curve (AUPRC) since these values are zero.

```
plot(auprc_plot_naive, main="AUPRC: 0")
```



We save our results from our first model in a data frame and display them at the end.
Although this model has an accuracy ~99.8%, it has a AUPRC of 0, and therefore is comepletely useless for our task at hand.

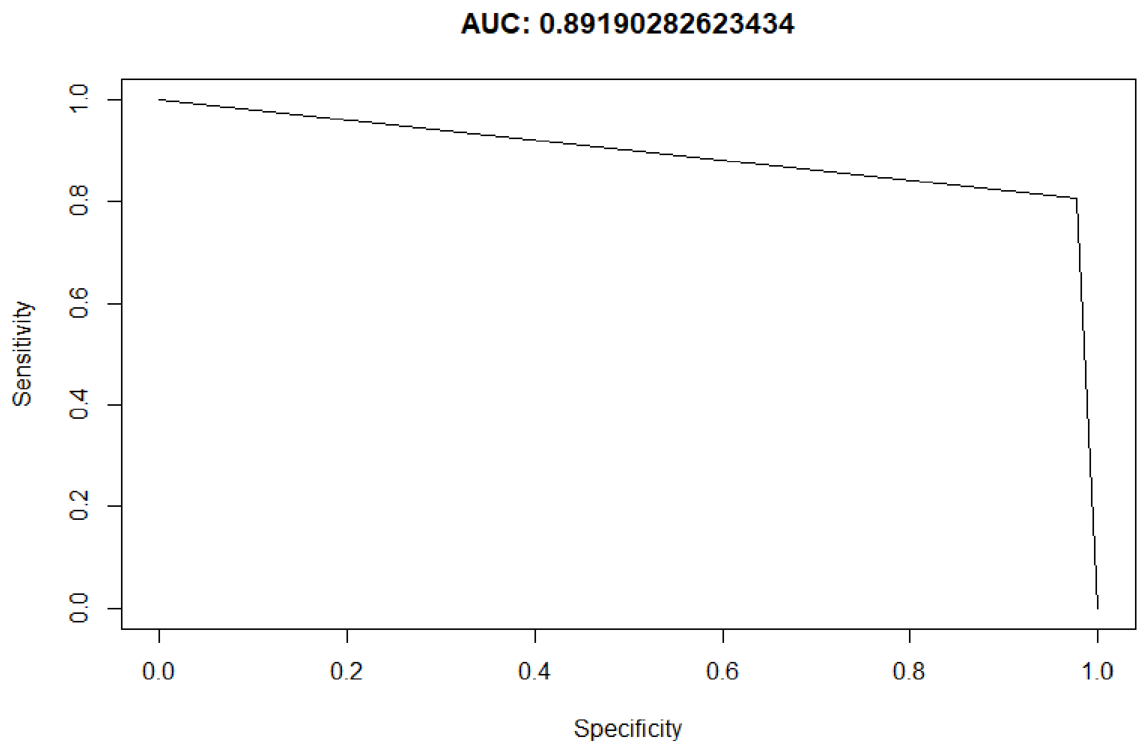
Naive Bayes

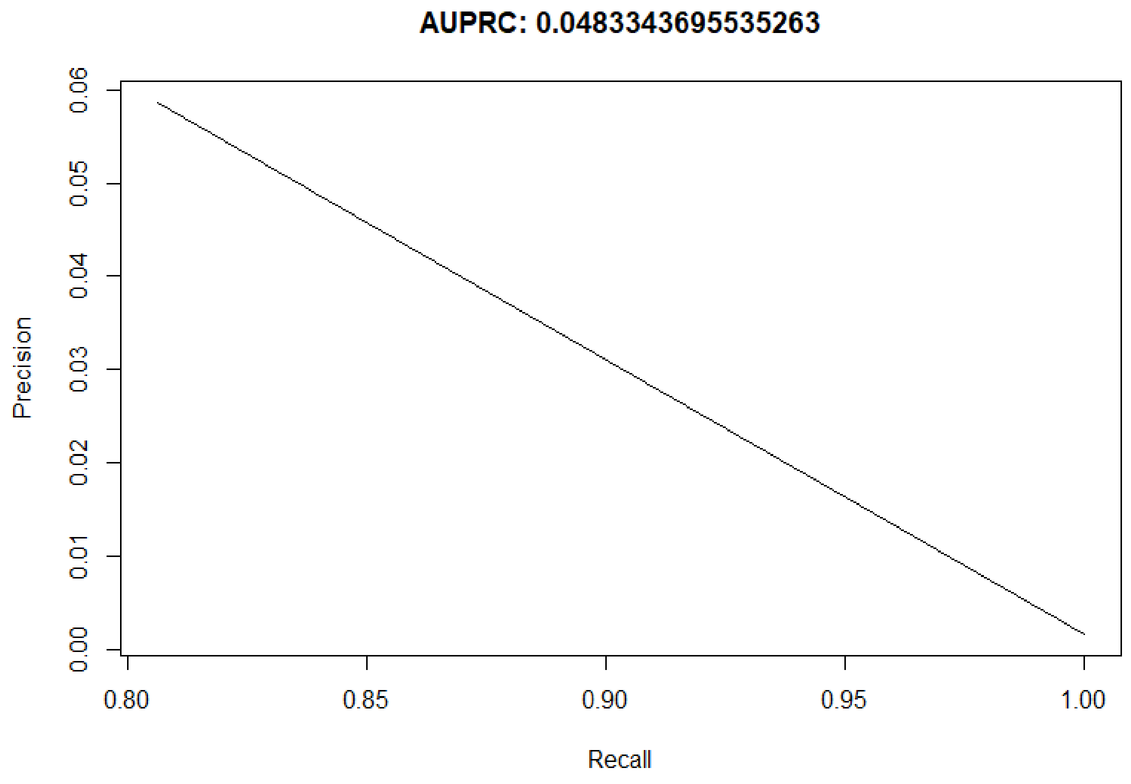
Model The Naive Bayes Model is a model that applies Bayes' theorem with strong (naive) independence assumptions between the features. We build the model with the 'Class' (i.e. whether the transaction is valid or fraud) as the target and with the remaining variables are predictors.

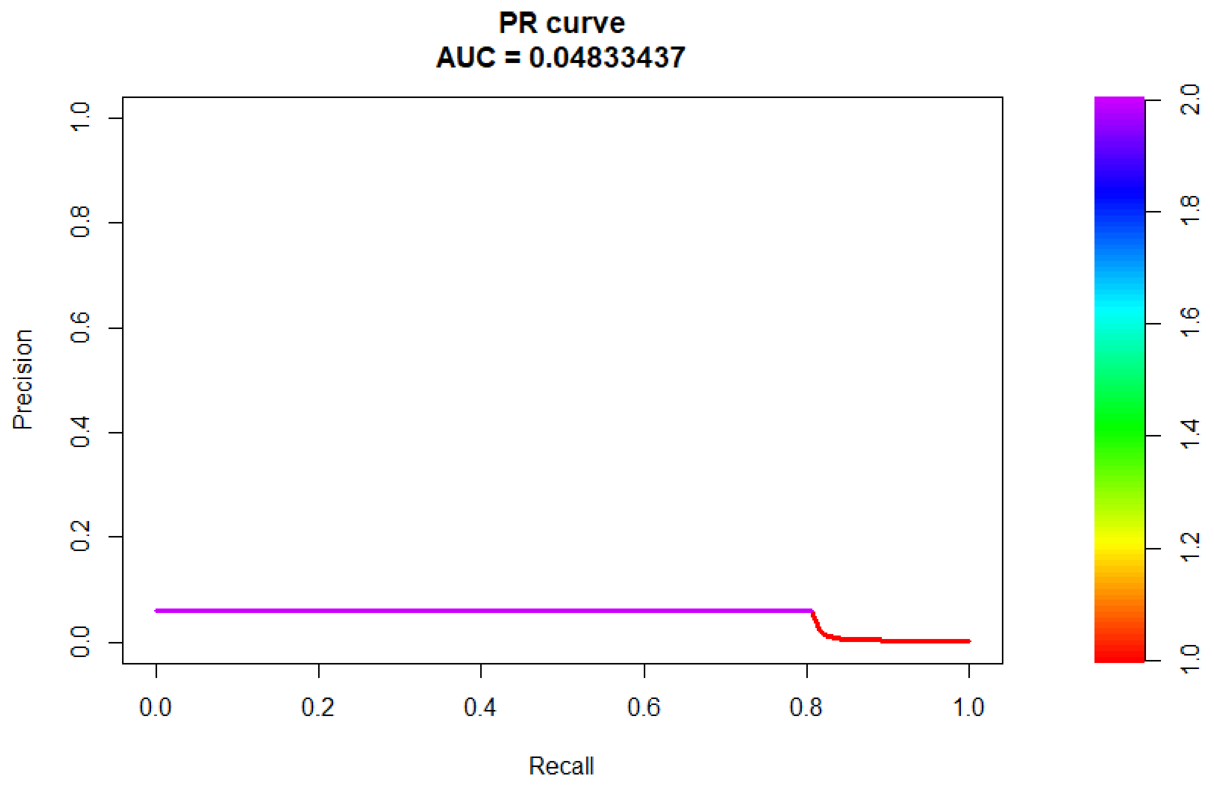
```r
set.seed(13)
#For the Naive Bayes Model, we build the model with the class as the target and with the remaining variables are
predictors.
#We start with our naive model and define the target and the predictors.
naive_model <- naiveBayes(Class ~ ., data = train, laplace=1)
#We then make the prediction based on our modified dataset.
predictions <- predict(naive_model, newdata=test)
#We need to compute the Area Under the Curve (AUC) and the Area Under the Precision-Recall Curve (AUPRC).
pred <- prediction(as.numeric(predictions), test$Class)
auc_val_naive <- performance(pred, "auc")
auc_plot_naive <- performance(pred, 'sens', 'spec')
auprc_plot_naive <- performance(pred, "prec", "rec")
#We apply the model to our test set.
auprc_val_naive <- pr.curve(
 scores.class0 = predictions[test$Class == 1],
 scores.class1 = predictions[test$Class == 0],
 curve = T,
 dg.compute = T)
```

The AUC for sensitivity versus specificity for the Naive Bayes Model is greatly improved compared to the Naive Model alone. Additionally, the AUPRC improves (albeit marginally) to just 0.05. We can improve on this with the following two models.

```
#We plot our curves for the Naive Bayes Model.
plot(auc_plot_naive, main=paste("AUC:", auc_val_naive@y.values[[1]]))
plot(auprc_plot_naive, main=paste("AUPRC:", auprc_val_naive$auc.integral))
plot(auprc_val_naive)
```







We save our results from our Naive Bayes Model in the previous data frame.

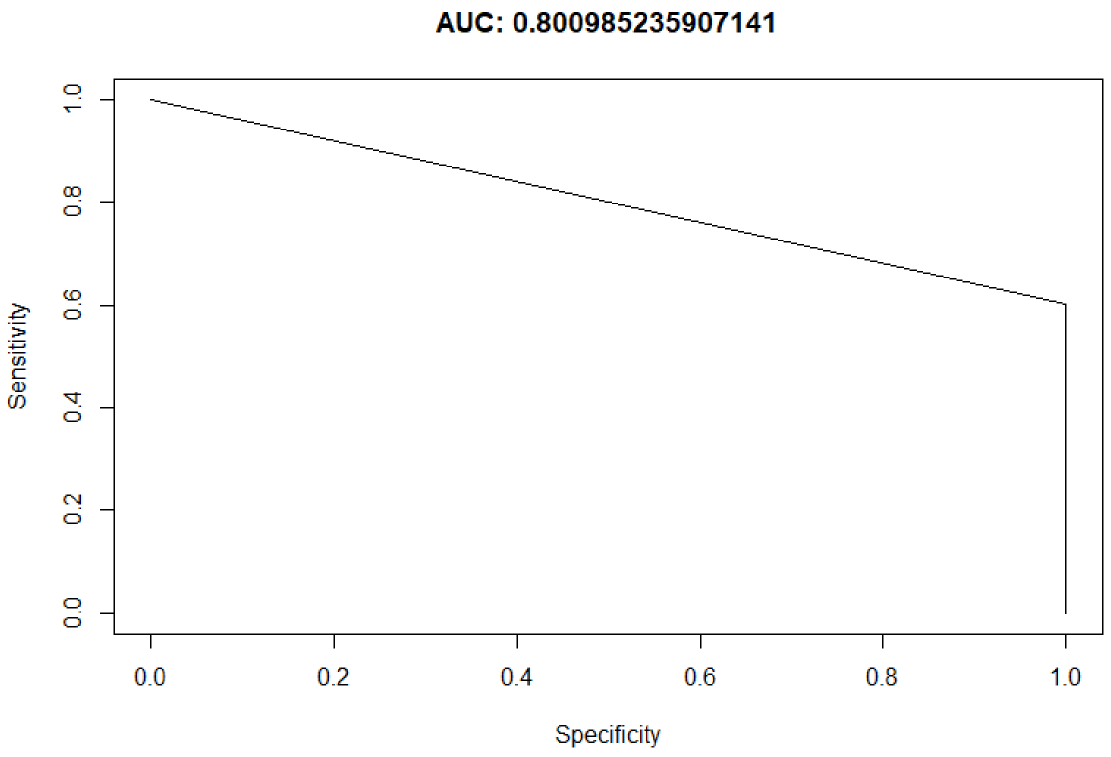
K-Nearest Neighbor

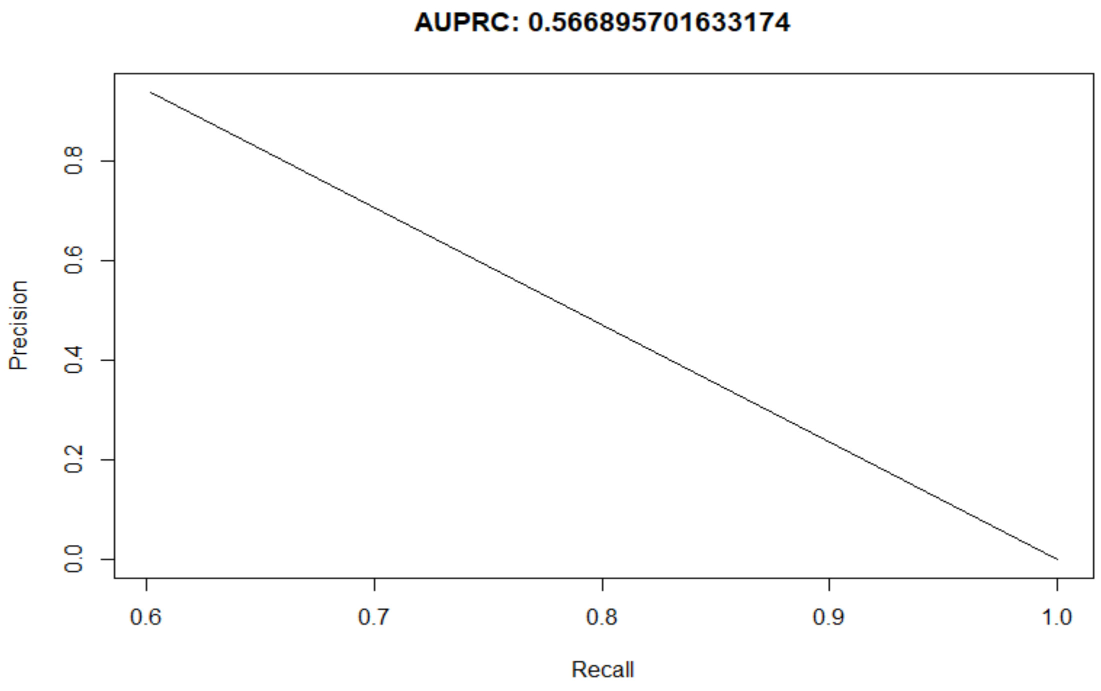
The K-Nearest Neighbors algorithm (KNN) is a non-parametric method used for classification where the input consists of the k closest training examples in the feature space. In KNN classification (determining if the transaction was valid or fraud), the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Several values of k were tested and 5 was chosen as a value that provided the best results. In this model, 'Class' is the target and all other variables are predictors.

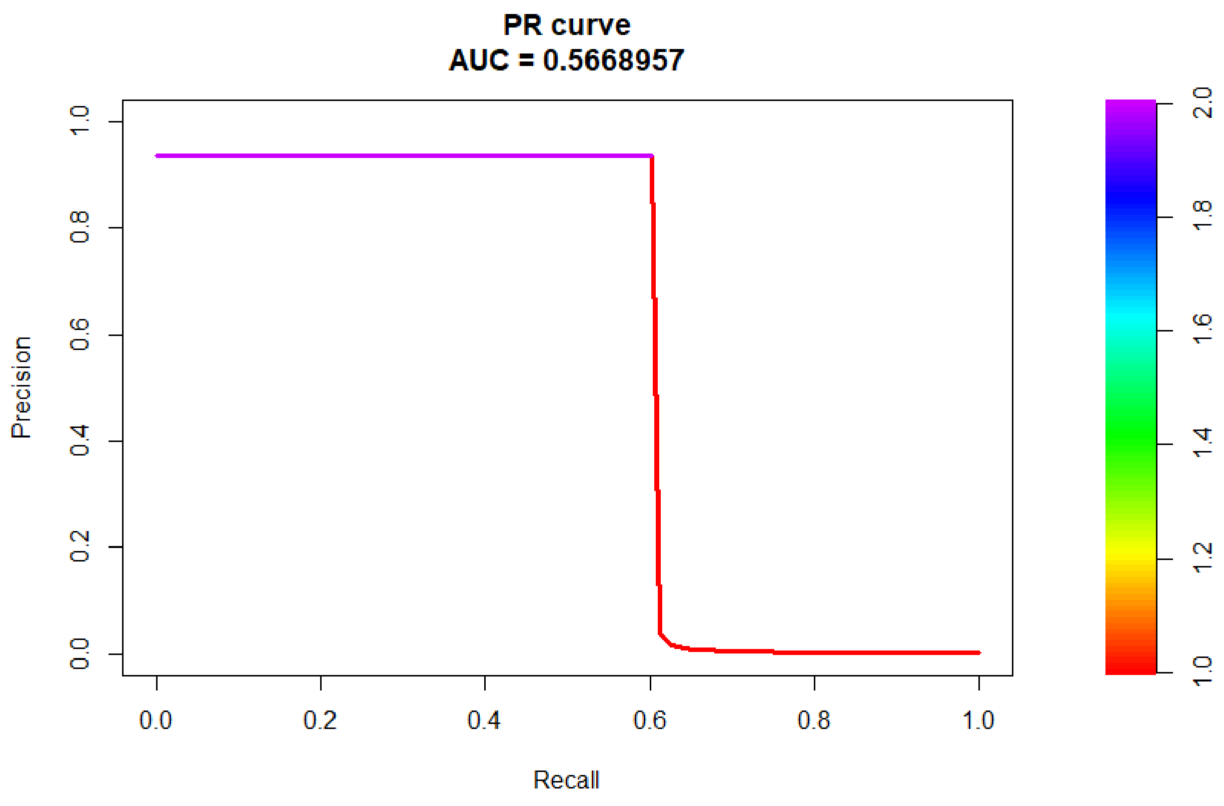
```
set.seed(13)
#Our next approach will be the K-Nearest Neighbor Model.
#This is building off of our previous model, the Naive Bayes Model, where we specify that Class is the target and all
#other variables are predictors. For this model we set k=5.
#Training this model takes a little bit of time.
knn_model <- knn(train[,-30],
          test[,-30],
          train$Class,
          k=5,
          prob = TRUE)
#We then make the prediction based on our modified dataset.
pred <- prediction(
  as.numeric(as.character(knn_model)),
  as.numeric(as.character(test$Class)))
#We need to compute the Area Under the Curve (AUC) and the Area Under the Precision-Recall Curve (AUPRC).
auc_val_knn <- performance(pred, "auc")
auc_plot_knn <- performance(pred, 'sens', 'spec')
auprc_plot_knn <- performance(pred, "prec", "rec")
#We apply the model to our test set.
auprc_val_knn <- pr.curve(
  scores.class0 = knn_model[test$Class == 1],
  scores.class1 = knn_model[test$Class == 0],
  curve = T,
  dg.compute = T)
```

For the K Nearest Neighbors, we have a small reduction for our AUC when looking at sensitivity versus specificity compared to the Naive Bayes Model, but a substantial improvement on precision versus recall in our AUPRC. This value of 0.56 is still low. We would like to achieve an AUC for precision versus recall close to 0.8.

```
#We plot our curves for the KNN Model.
plot(auc_plot_knn, main=paste("AUC:", auc_val_knn@y.values[[1]]))
plot(auprc_plot_knn, main=paste("AUPRC:", auprc_val_knn$auc.integral))
plot(auprc_val_knn)
```







We save our results from our K-Nearest Neighbor Model in a data frame and display them with previous results.

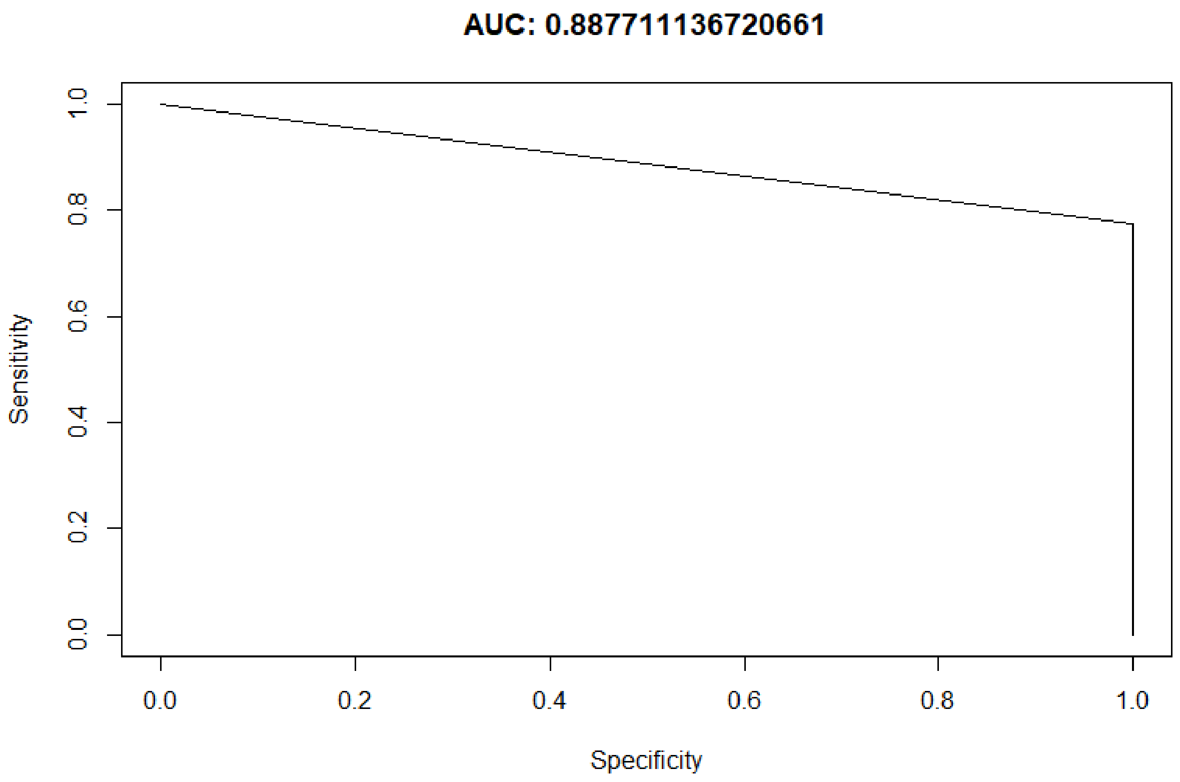
Random Forest

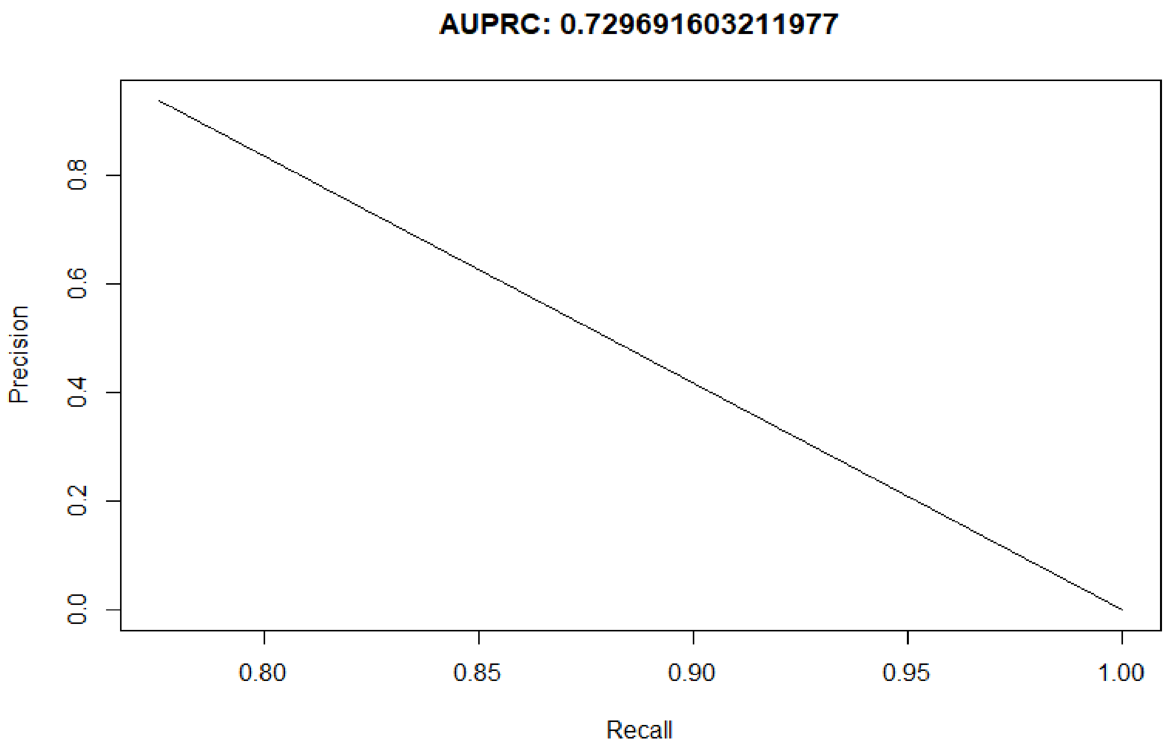
The Random Forest algorithm (sometimes called Random Decision Forests) is an algorithm of machine learning were an ensemble learning method for classification operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classification of the individual trees. These trees are a decision tree that goes from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). For 11 this model, 'Class' (whether a transaction is valid or fraud) is the target, and all other variables are predictors. In this model we define the number of trees to be 500.

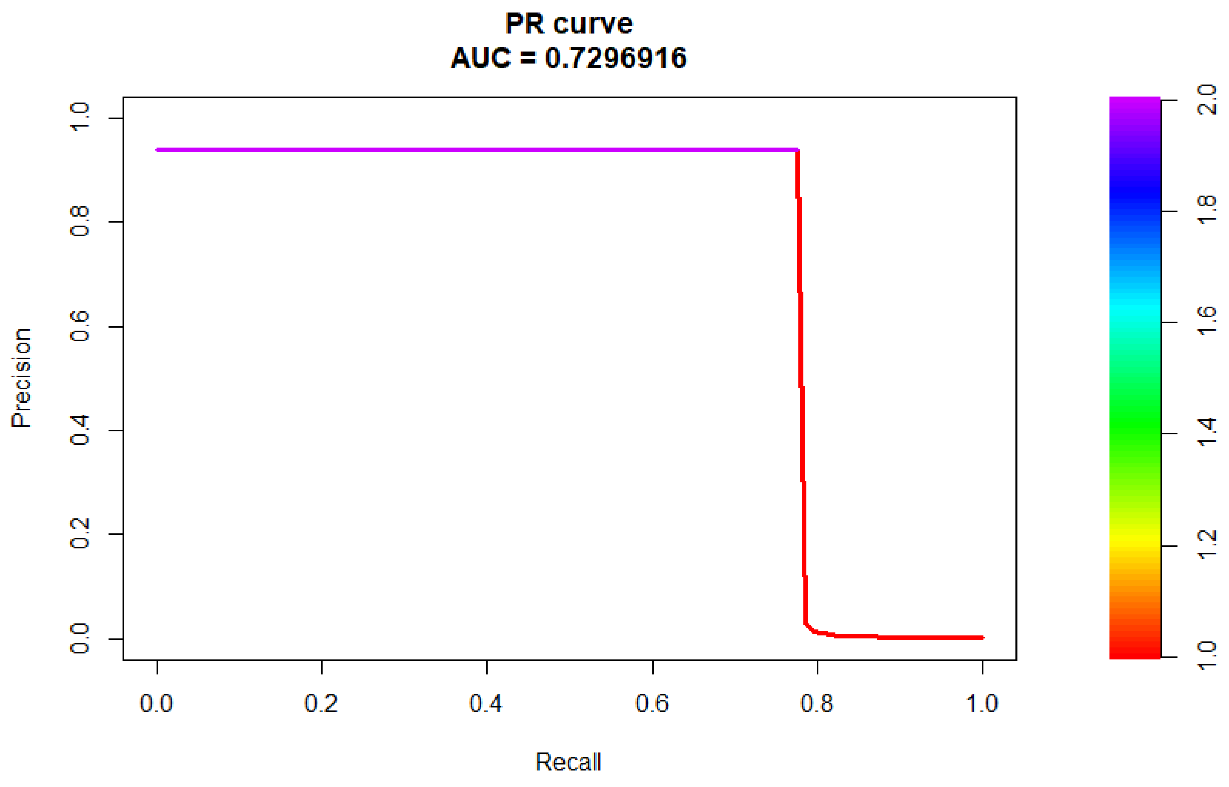
```
set.seed(13)
#Our next approach will be the Random Forest Model.
#As with the two previous models, we specify that Class is the target and all other variables being predictors.
#For the Random Forest Model, we will define the number of trees to be 500.
#This takes a while to train the model.
rf_model <- randomForest(Class ~ ., data = train, ntree = 500)
#We then make the prediction based on our modified dataset.
predictions <- predict(rf_model, newdata=test)
pred <- prediction(
  as.numeric(as.character(predictions)),
  as.numeric(as.character(test$Class)))
#We need to compute the Area Under the Curve (AUC) and the Area Under the Precision-Recall Curve (AUPRC).
auc_val_rf <- performance(pred, "auc")
auc_plot_rf <- performance(pred, 'sens', 'spec')
auprc_plot_rf <- performance(pred, "prec", "rec",
                             curve = T,
                             dg.compute = T)
auprc_val_rf <- pr.curve(scores.class0 = predictions[test$Class == 1],
                         scores.class1 = predictions[test$Class == 0],
                         curve = T,
                         dg.compute = T)
```

For our Random Forest Model, we not only obtain the best AUC for sensitivity versus specificity (0.91), but we also obtain the best AUC for precision versus recall (0.78). Out of the models developed and trained, this model is the most accurate for our task at hand. The use of 500 trees for this algorithm worked well.

```r
#We plot our curves for the Random Forest Model.
plot(auc_plot_rf, main=paste("AUC:", auc_val_rf@y.values[[1]]))
plot(auprc_plot_rf, main=paste("AUPRC:", auprc_val_rf$auc.integral))
plot(auprc_val_rf)
```







We save our results from our Random Forest Model in our data frame.

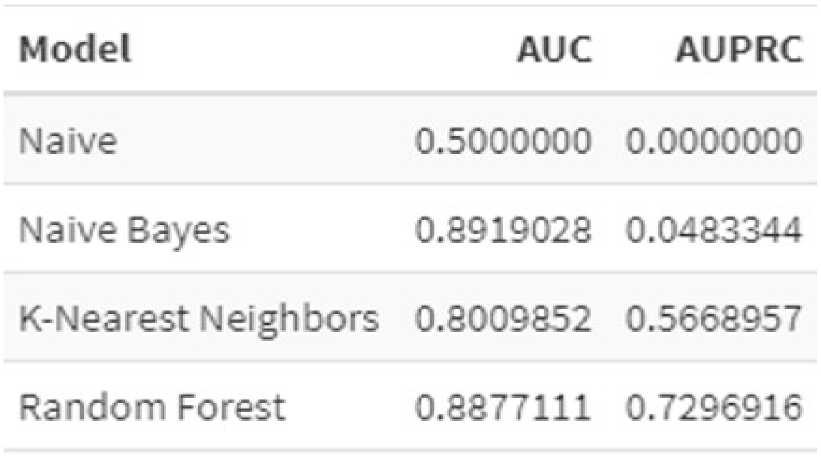
Results

Prior to our computations, we partition the dataset into a training set, a test set, and a cross validation set.

In this report we seek to address credit card fraud using a machine learning approach. Since credit card fraud is very rare compared to the volume of valid transactions, we are posed with a machine learning problem that utilizes the accuracy of the model by calculating the Area Under the Precision-Recall Curve as opposed to a more traditional method such as a confusion matrix.

Four models were developed and each was tested with a dataset of credit card transactions provided by Kaggle. Here we again present the findings from the four models utilized for this report.

```
#Our results are displayed in a table format with previous results.
results %>%
 kable() %>%
 kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
          position = "center",
          font_size = 14,
          full_width = FALSE)
```



Model	AUC	AUPRC
Naive	0.5000000	0.0000000
Naive Bayes	0.8919028	0.0483344
K-Nearest Neighbors	0.8009852	0.5668957
Random Forest	0.8877111	0.7296916

The model that best suited the needs of the task at hand was the Random Forest algorithm. This machine learning algorithm is an ensemble learning method for classification. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classification of the individual trees. In this algorithm, we defined the number of trees to be 500.

Our results from the Random Forest algorithm are impressive compared to three models previously tested on this dataset. We obtained an Area Under the Curve (AUC) for sensitivity versus specificity of 0.913, and an Area Under the Precision-Recall Curve (AUPRC) of 0.78. This model drastically improved the AUPRC of the K-Nearest Neighbors algorithm. Although an AUPRC of 0.8 was not achieved, our result is very close. More sophisticated models utilized in machine learning may be able to obtain a better result.

Conclusion

In this report we seek to address credit card fraud using a machine learning approach. Since credit card fraud is very rare compared to the volume of valid transactions, we are posed with a machine learning problem that utilizes the accuracy of the model by calculating the Area Under the Precision-Recall Curve as opposed to a more traditional method such as a confusion matrix. Four models were developed and each was tested with a dataset of credit card transactions provided by Kaggle (*https://www.kaggle.com/mlg-ulb/creditcardfraud*).