frauddetection

Pritesh Contractor

8/09/2020

Introduction

In this project, we are asked to choose publicly available datasets which are not from the set of well known datasets used during the course and provide us an opportunity to explore those, analyze it by applying machine learning techniques and going beyond standard linear regressions models.

Objective

The main aim for this project is to explore publicly available dataset and apply machine learning techniques to analyse the dataset. With increase in the online shoppings and wide use of plastic money. It is also important to keep an eye on any transaction which are fraud and detect those. Hence for this project we are going to use publicly available dataset from kaggle which I have uploaded on the google drive and provided access to download the use the same throughout the predictions.The datasets contains the transactions made by credit cards in late 2013 and due to highly imbalance nature of data, many observations could be predicted as False Negative. Also within datasets the prediction which is always 0 (legal transaction) achieves an Accuracy of 99.8. Due to which we are measureing the score for Area Under The Precision-Recall Curve (AUCPR) instead of traditional AUC Curve. Moreover, the acceptable or desirable result is when AUCPR is greater than 0.83. For achieving those results we are trainig several algorithms like Naive Bayes Classfier, KNN, SVM, Random Forest, GBM and XGBoost.

During the analysis the most desirable AUCPR is achieved through the model > 0.83

```
# install all libraries which are not present and required

if(!require(tidyverse)) install.packages("tidyverse")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Loading required package: kableExtra
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
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")
```

```
## Loading required package: gbm
```

```
## Loaded gbm 2.1.8
```

```
if(!require(dplyr)) install.packages("dplyr")
if(!require(caret)) install.packages("caret")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(xgboost)) install.packages("xgboost")
```

```
## Loading required package: xgboost
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
if(!require(e1071)) install.packages("e1071")
```

```
## Loading required package: e1071
```

```r
if(!require(class)) install.packages("class")
```

```
## Loading required package: class
```

```r
if(!require(ROCR)) install.packages("ROCR")
```

```
## Loading required package: ROCR
```

```r
if(!require(randomForest)) install.packages("randomForest")
```

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
if(!require(PRROC)) install.packages("PRROC")
```

```
## Loading required package: PRROC
```

```r
if(!require(reshape2)) install.packages("reshape2")
```

```
## Loading required package: reshape2

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
if(!require(googledrive)) install.packages("googledrive")
```

```
## Loading required package: googledrive
```

```r
if(!require(purrr)) install.packages("purrr")

# loading required libraries

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(googledrive)
library(purrr)
```

Process

Below are the methods and analysis steps which we will be performing as a part of project:

1. Data Preparation
2. Data Exploration
3. Data Preprocessing
4. Data Analysis
5. Communicate

As a part of this project, we have use publicly available dataset called creditcard from kaggle

Issues Came acroos:

I have uploaded to the Google Drive because while uploading on GitHub it was giving me warning and error that I cannot upload file more than 25 MB.

Hence, I have downloaded the csv file and uploaded to newly created Google account "datawithpritesh@gmail.com" Drive and also provided permission to download the same when the code is ran within R Studio. Please note while running the code sometime the authorization part is creating an problem. However, I have tested and it seems to be working fine. But I have seen that while you have mutliple GMAIL Account open at your end it may ask you to choose which one you want to go with. You can select the option which as "datawithpritesh@gmail.com" in case there are any issues please get back to me or post in the discussion box.

Once data set is downloaded after that we read the csv file and perform the analysis on the same.

Method and Analysis

Part 1 - Data Preparation

In this section we will download and prepare dataset for analysis. We will download it from Google Drive and save it locally and then read the csv file from the location it is saved.

```
# restricting google drive to not attempt for getting or sending token
drive_auth_config(active = FALSE)
```

```
## Warning:   `drive_auth_config()` has been deprecated.
##    Use `drive_auth_configure()` to configure your own OAuth app or API key.
##    Use `drive_deauth()` to go into a de-authorized state.
##    Use `drive_oauth_app()` to retrieve a user-configured app, if it exists.
##    Use `drive_api_key()` to retrieve a user-configured API key, if it exists.
```

```
# downloading csv file from google drive and saving into temp folder

folder_url <- "https://drive.google.com/file/d/1DnIYCJBlqOpatH-juUgJDN2JXVYQZwb9/view?usp=sharing"
folder <- drive_get(as_id(folder_url))
```

```
## Suitable tokens found in the cache, associated with these emails:
##   * datawithpritesh@gmail.com
##   * priteshcontractor@gmail.com
## The first will be used.

## Using an auto-discovered, cached token.
## To suppress this message, modify your code or options to clearly consent to the use of a cached toke
## See gargle's "Non-interactive auth" vignette for more details:
## https://gargle.r-lib.org/articles/non-interactive-auth.html

## The googledrive package is using a cached token for datawithpritesh@gmail.com.
```

```
#if (dir.exists("C:\\Users\\nisha\\AppData\\Local\\Temp\\Rtmpq8dIel\\")){
  downloadpath <- tempfile(fileext="creditcard.csv")
  drive_download(as_id(folder_url),path=downloadpath, overwrite = TRUE)#} else {#
```

```
## File downloaded:
##   * creditcard.csv
## Saved locally as:
##   * C:\Users\nisha\AppData\Local\Temp\RtmpGAx9h5\file12c70743d7a4bcreditcard.csv
```

```
#  dir.create("C:\\Users\\nisha\\AppData\\Local\\Temp\\Rtmpq8dIel\\")
  downloadpath <- tempfile(fileext="creditcard.csv")
  drive_download(as_id(folder_url),path=downloadpath, overwrite = TRUE)
```

```
## File downloaded:
##   * creditcard.csv
## Saved locally as:
##   * C:\Users\nisha\AppData\Local\Temp\RtmpGAx9h5\file12c7078956428creditcard.csv
```

```
#}

# closing all unwanted connections
# on.exit(closeAllConnections())

# loading dataset
readcsv <- read.csv(downloadpath, header = TRUE, sep=",")
```

Part 2 - Data Exploration

During data exploration we will get the length and column for the dataset. Also we will visualize the class proportion as a part of Data Visualization. Identifying the missing values to check for data cleansing. Plot the histogram over amount and time distribution and identify the correlation matrix for ploting the Pearson Correlation on the dataset.

This datasets presents transactions which were occured in two days and it has 492 frauds out of 284,807 transactions. It is highly inbalanced, also the dataset contain only numerical input variables which are result of a PCA transformation. Unfortunetly, due to confidentiality issues, we cannot provide the original features. Features V1, V2, ... v28 are the principle componets obtained with PCA.

Source

https://www.kaggle.com/mlg-ulb/creditcardfraud

It is imbalanced data which means there are few rows that represent a class. Also while exploring the Data we will also identify missing values to check and fix it if any available

Get length and columns of the dataset

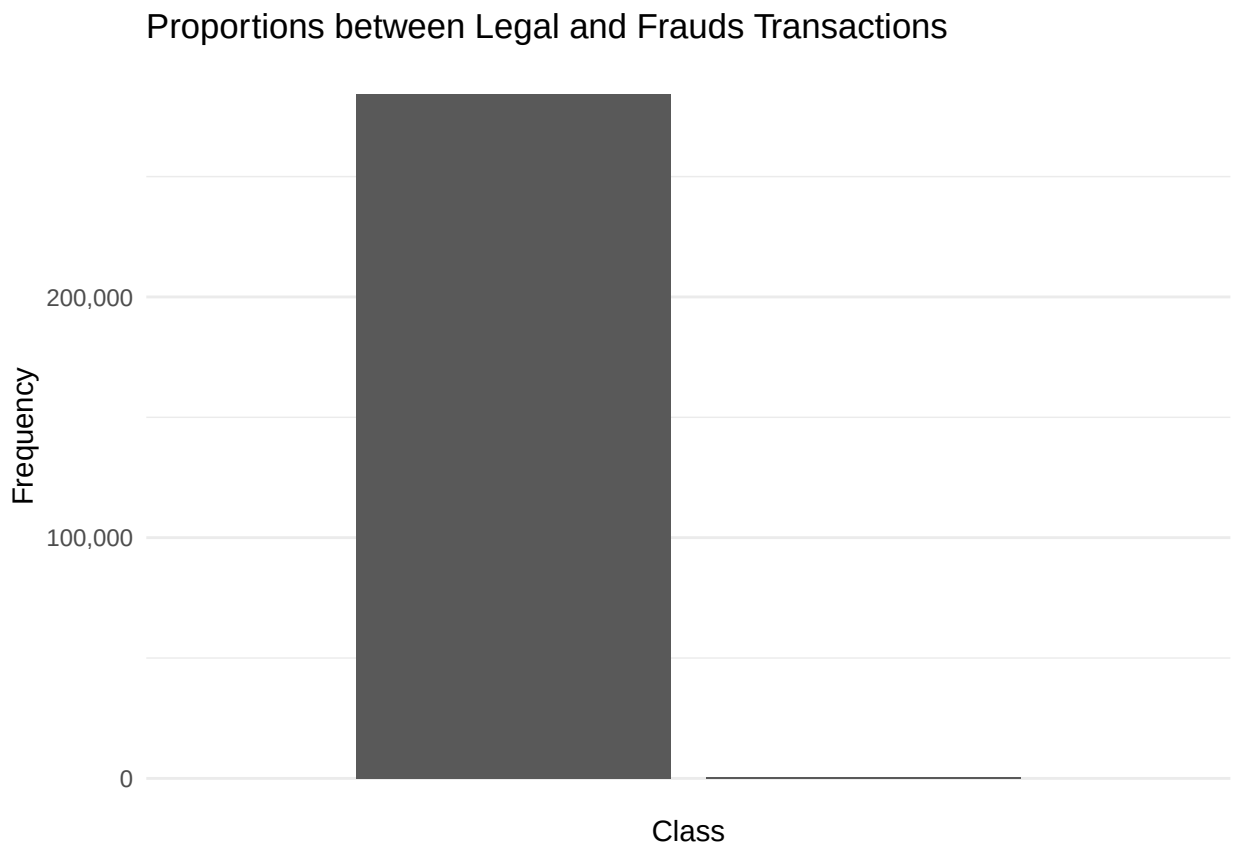
```
# Get length and columns details from the dataset

data.frame("Length"=nrow(readcsv), "Columns"=ncol(readcsv)) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped","hover","condensed","responsive"),
                position="center",
                font_size = 12,
                full_width = FALSE)
```

Length	Columns
284807	31

```
dim <- data.frame(readcsv)
dim$class = ifelse(readcsv$Class==0,'Legal Transaction','Fraud Transaction') %>% as.factor()
```

Visualize class proportion

```r
# Visualize class proportion

dim %>%
  ggplot(aes(Class)) +
  theme_minimal() +
  geom_bar() +
  scale_x_discrete() +
  scale_y_continuous(labels = scales::comma) +
  labs(title = "Proportions between Legal and Frauds Transactions",
       x = "Class",
       y = "Frequency")
```



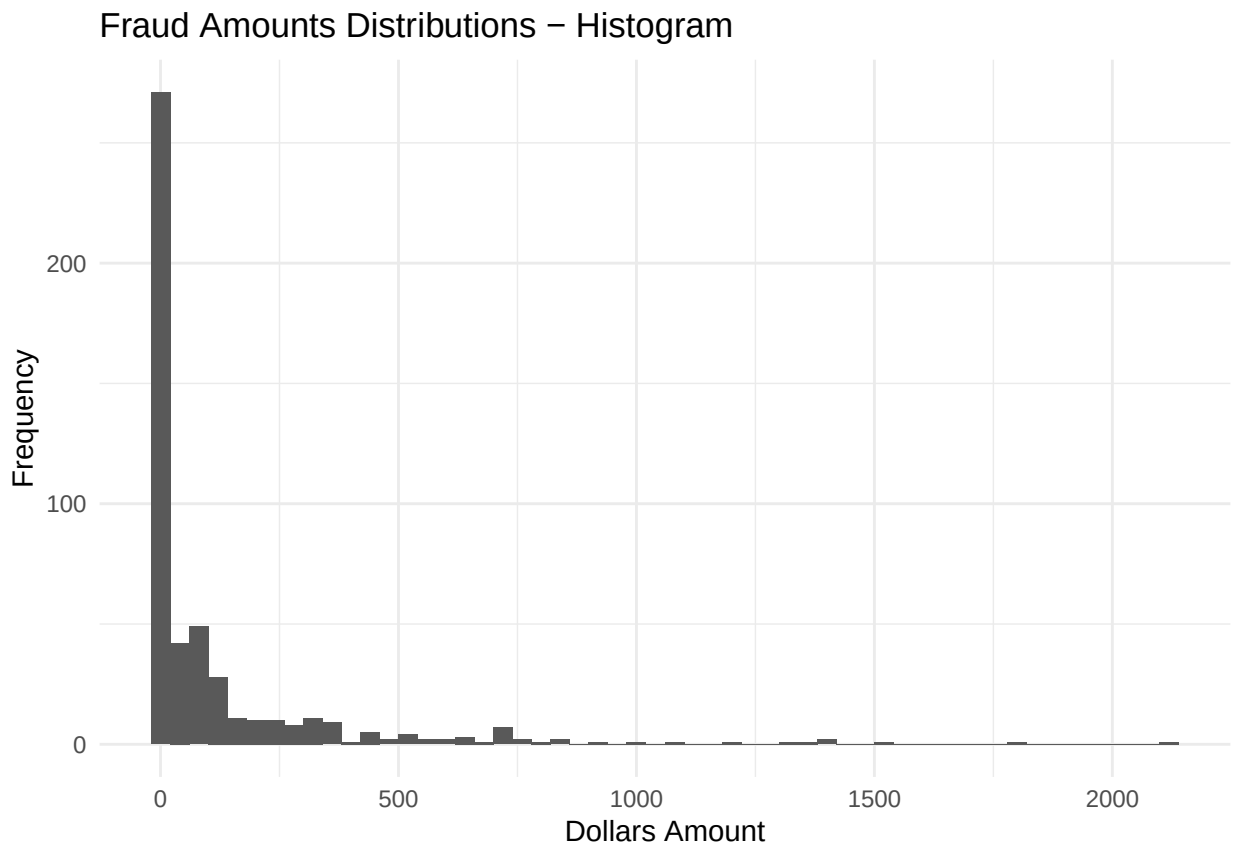
Identifying missing values from imbalancced dataset

```r
# identifying missing values

sapply(readcsv, function(x) sum(is.na(x))) %>%
  kable(col.names = c("Missing Values")) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

	Missing Values
Time	0
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0
V19	0
V20	0
V21	0
V22	0
V23	0
V24	0
V25	0
V26	0
V27	0
V28	0
Amount	0
Class	0

Fraud Transaction over amount distribution - Histogram

```
# Histogram - frauds transaction amounts distribution

readcsv[readcsv$Class == 1,] %>%
  ggplot(aes(Amount)) +
  theme_minimal()  +
  geom_histogram(binwidth = 40) +
  labs(title = "Fraud Amounts Distributions - Histogram",
       x = "Dollars Amount",
       y = "Frequency")
```



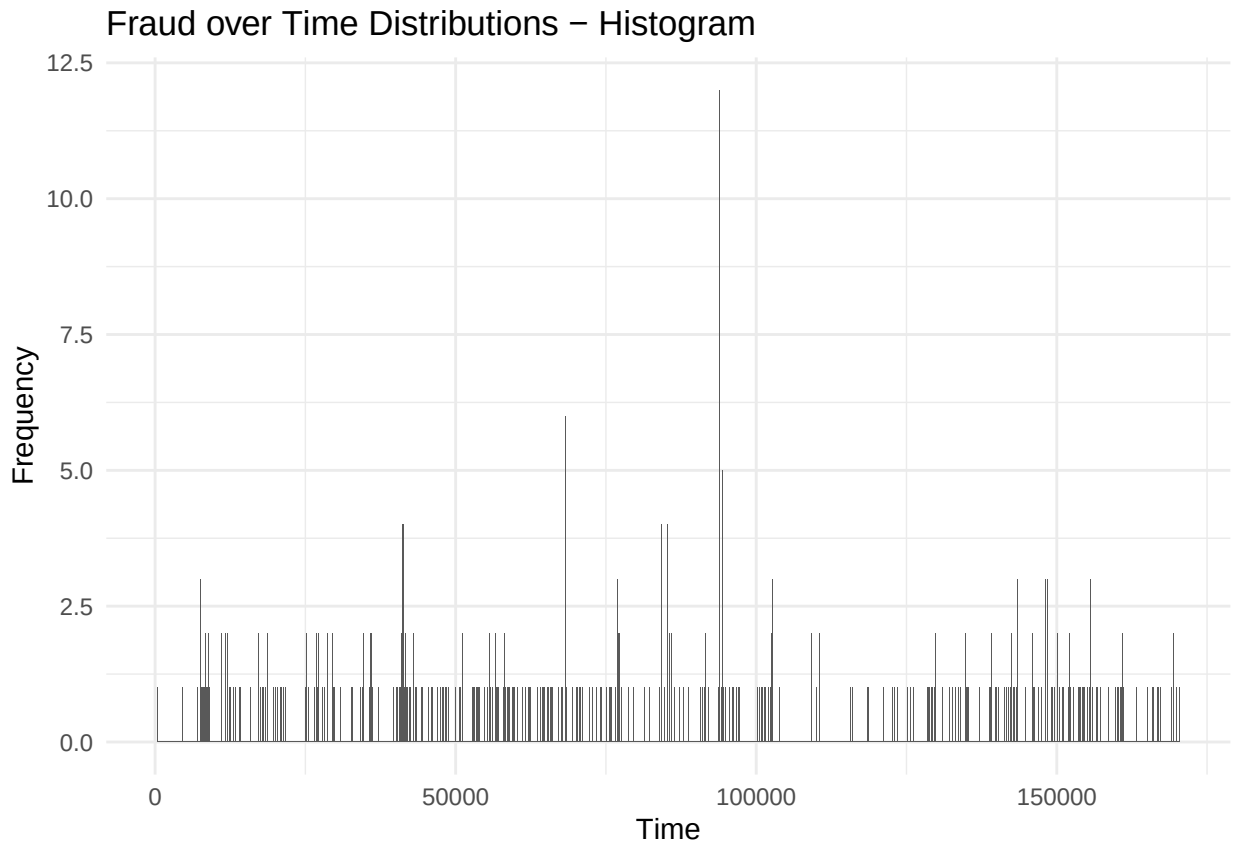
Extracting counts per amount

```
# Extracting counts per amount

readcsv[readcsv$Class == 1,] %>%
  group_by(Amount) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(n=10) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

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

Fraud transaction over time distribution through Histogram

```
# Histogram - Fraud transaction over time distribution

readcsv[readcsv$Class == 1,] %>%
  ggplot(aes(Time)) +
  theme_minimal()  +
  geom_histogram(binwidth = 40) +
  labs(title = "Fraud over Time Distributions - Histogram",
       x = "Time",
       y = "Frequency")
```



Identifyying count per time

```
# identifying count per time

readcsv[readcsv$Class == 1,] %>%
  group_by(Time) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(n=10) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Time	count
68207	6
84204	4
85285	4
93853	4
93860	4
93879	4
94362	4
148053	2
406	1
472	1

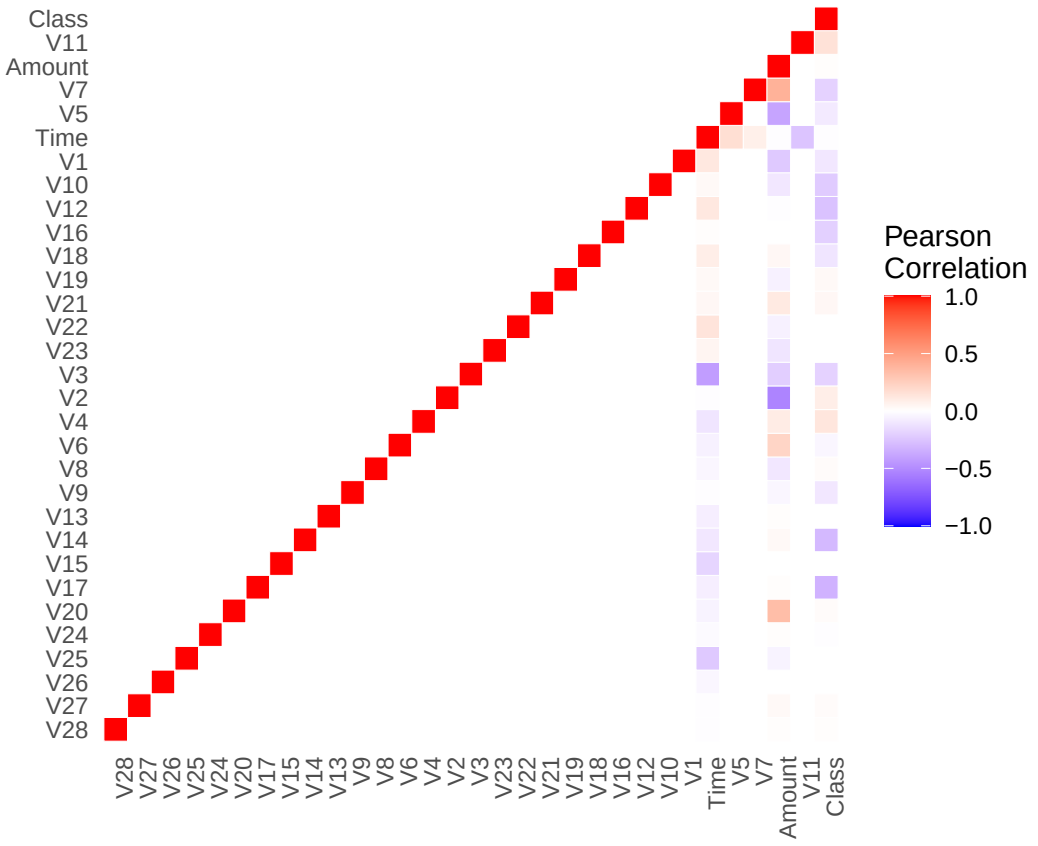
Calculating Correlation Matrix

```
# identifying correlation matrix - lower traig

get_lower_triag <- function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

# identifying correlation matrix - upper traig

get_upper_triag <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

# creating function reorder cormat - using correlation between variables as distance

reorder_cormat <- function(cormat){
  # Use correlation between variables as distance
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
```

```
}

corr_matrix <- round(cor(readcsv),2)
corr_matrix <- reorder_cormat(corr_matrix)

upper_traig <- get_upper_triag(corr_matrix)

melted_corr_matrix <- melt(upper_traig, na.rm = TRUE)
```

Plotting for Pearson Correlation Matrix

```
# ggplot for pearson correlation of the dataset

ggplot(melted_corr_matrix, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 1,
                                   size = 9, hjust = 1), axis.text.y = element_text(size = 9),axis.titl
        axis.title.y = element_blank(),
        panel.grid.major = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.ticks = element_blank()) +
  coord_fixed()
```



Part 3 - Preprocessing of data

Before we start analysing and build models. There is a requirement to do some data preprocessing i.e. removing unwanted or not useful column like the "Time"column from the dataset and split the dataset into train, test and cv dataset as it will be required in later stage for training variouse models and getting desirable AUCPR

```
# removing the time column from the dataset

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

# split the dataset into train, test and cross validation dataset

train_index <- createDataPartition(
  y = readcsv$Class,
  p = .6,
  list = F
)

train <- readcsv[train_index,]

test_cv <- readcsv[-train_index,]
```

```
test_index <- createDataPartition(
  y = test_cv$Class,
  p = .5,
  list = F)

test <- test_cv[test_index,]
cv <- test_cv[-test_index,]

rm(train_index, test_index, test_cv)
```

Part 4 - Data Analysis

Building various models and comparing the result to get desirable outcome of AUCPR

Naive Baseline Algorithm - Predicting Always Legal Transactions

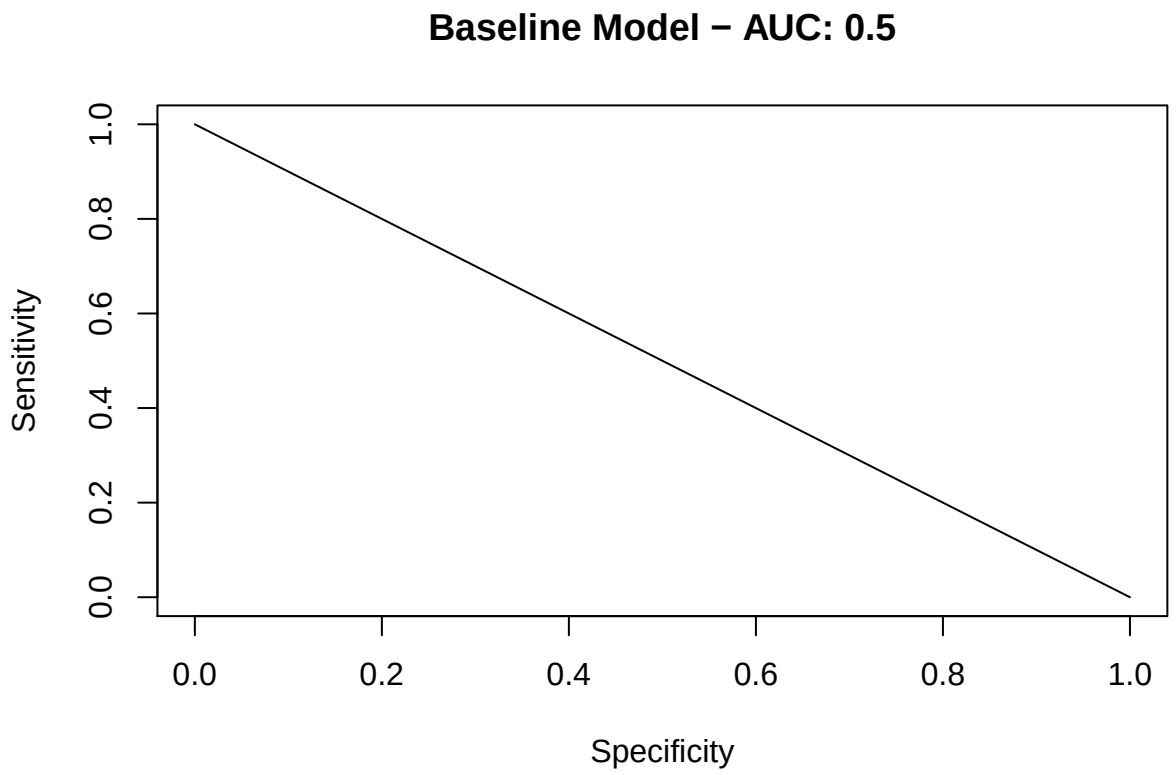
Naive Bayes is supervised machine learning which is based on the Bayes Theorem that is used to solve classification problems by following a probabilistic approach. The model is build Predicting always Legal transaction which achieves an impressive accuracy 99.8 and an AUC 0.5. Recalling and precision are 0, it is impossible to compute the AUCPR so that is 0

```
# Baseline model creation which predicts Legal Transaction only with 0 and compute all metrics

# cloning the data frame

baseline_model <- data.frame(readcsv)

# Setting up the class al to Legal (0)

baseline_model$Class = factor(0, c(0,1))

# Predicting using baseline model

baseline_pred <- prediction(
  as.numeric(as.character(baseline_model$Class)),as.numeric(as.character(readcsv$Class))
)

# Computing AUC and AUCPR

auc_val_baseline <- performance(baseline_pred, "auc")
auc_plot_baseline <- performance(baseline_pred, 'sens', 'spec')
aucpr_plot_baseline <- performance(baseline_pred, "prec", "rec")
```

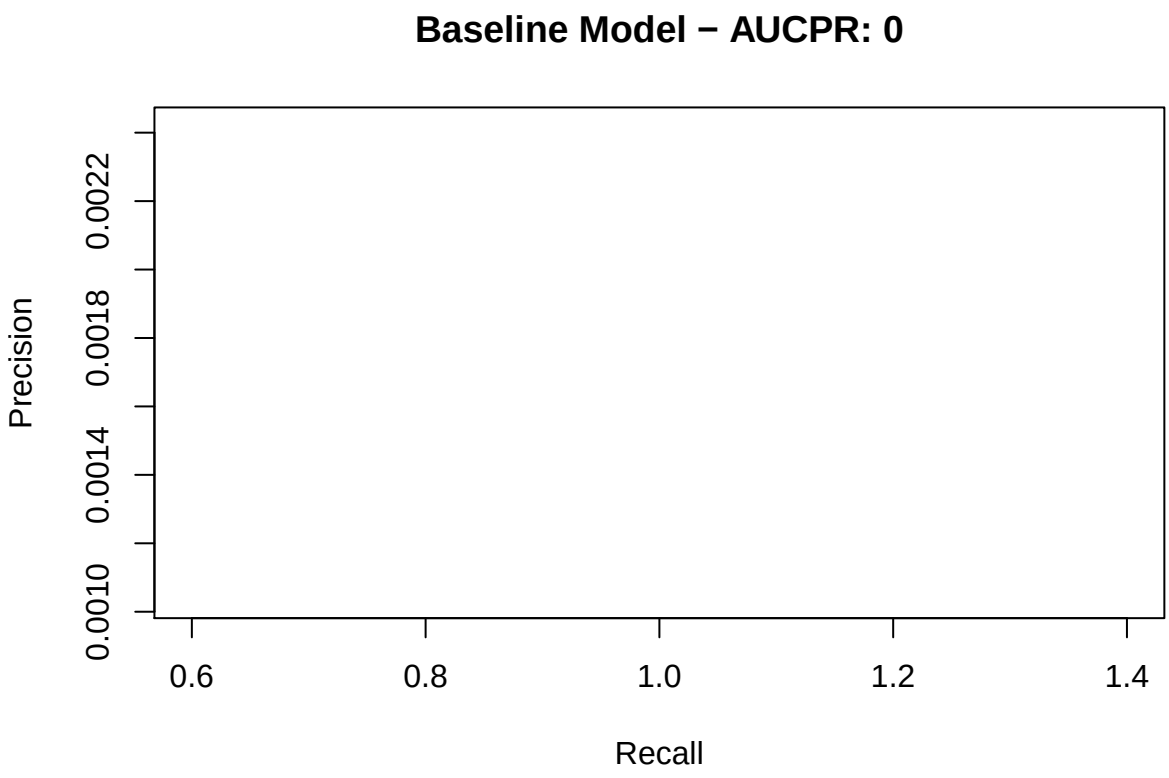
Create relative plot for baseline model

```
# creating relative plot

plot(auc_plot_baseline,
     main=paste("Baseline Model - AUC:",
```

```
                auc_val_baseline@y.values[[1]])
)
```




```
plot(aucpr_plot_baseline, main="Baseline Model - AUCPR: 0")
```



Create Data Frame Outcome to enter the results obtained through Based Models and will be using same going forward for adding the metrics to the same

```
# Create data frame 'outcome' that contains all metrics obtained from the Base models

outcome <- data.frame(
  Model = "Baseline - Predict Always Legal",
  AUC = auc_val_baseline@y.values[[1]],
  AUCPR = 0
)

# display outcome in a table

outcome %>%
  kable() %>%
  kable_styling(
    bootstrap_options =
      c("striped", "hover", "condensed", "responsive"),
    position = "center",
    font_size = 10,
    full_width = FALSE
  )
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5	0

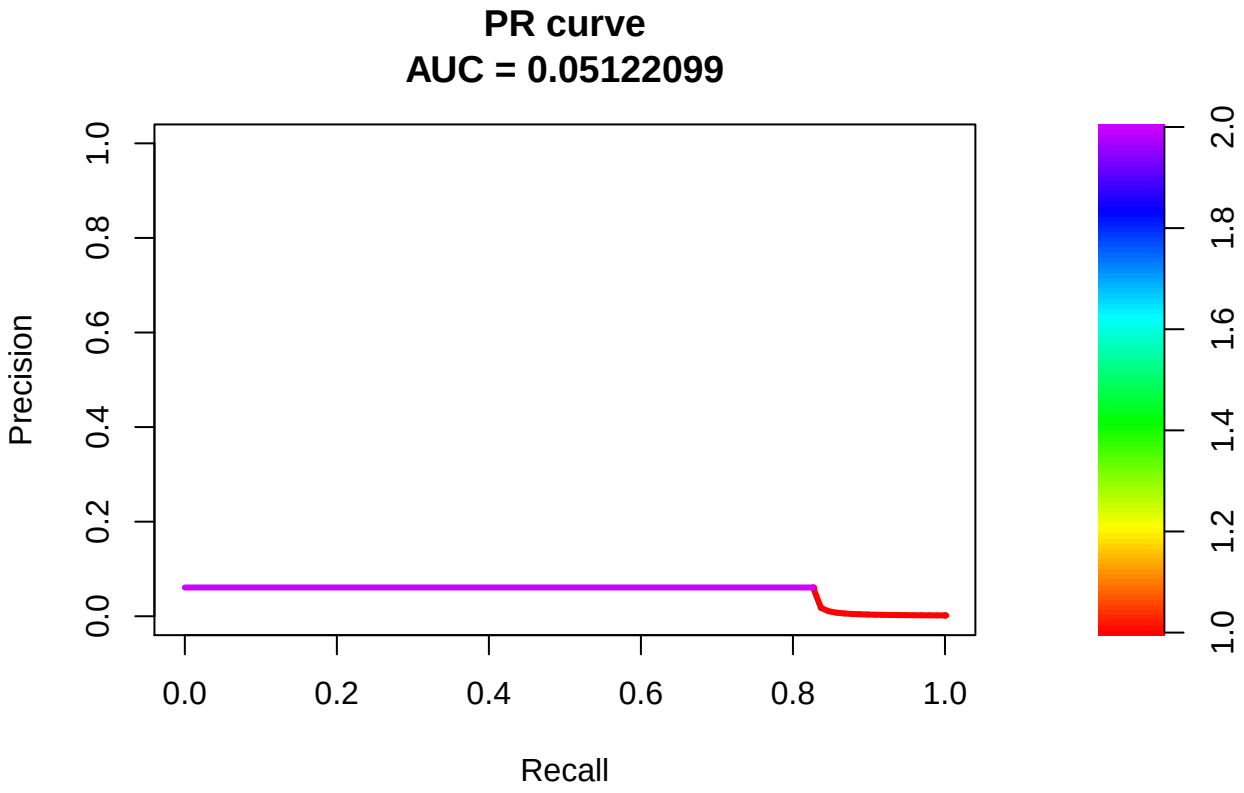
Naive Bayes Classifier

It is a simple probailistic classifier which is based on Bayes Theorem but it is with strong assumption regarding independence. Stepping forward the performance improve a little bit with regards to AUCPR prediction. It is not what we desired according to the metric of interest and we will still improve it with other models

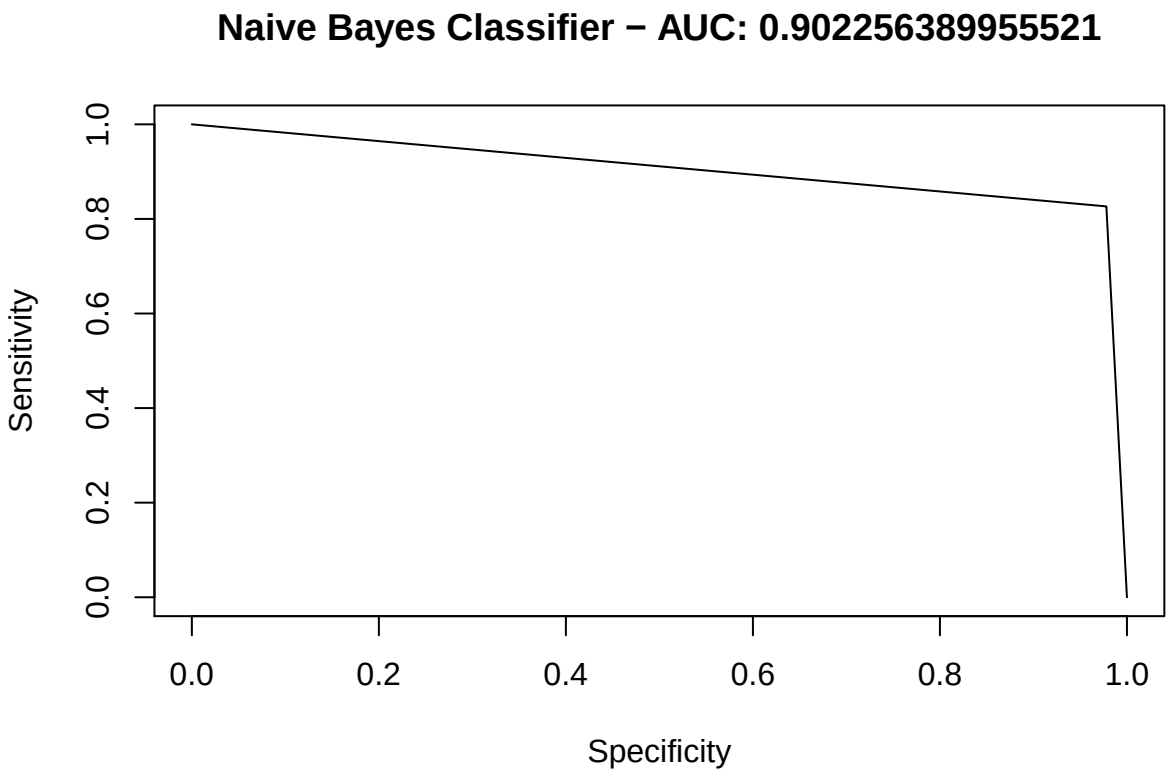
```
# Naive Bayes Model creation which will improve the outcome in AUC and AUCPR

# Building Naive Bayes Classifier Model  with Class as a target and all other variables as predictor


naive_model <- naiveBayes(Class ~ ., data = train, laplace=1)

# Prediction outcome

naive_model_predictions <- predict(naive_model, newdata=test)

# Computing AUC and AUCPR for the Naive Model

naive_pred <- prediction(as.numeric(naive_model_predictions) , test$Class)

auc_val_naive <- performance(naive_pred, "auc")

auc_plot_naive <- performance(naive_pred, 'sens', 'spec')
aucpr_plot_naive <- performance(naive_pred, "prec", "rec")

aucpr_val_naive <- pr.curve(
  scores.class0 = naive_model_predictions[test$Class == 1],
  scores.class1 = naive_model_predictions[test$Class == 0],
  curve = T,
  dg.compute = T
)
```

Plot the relatives for aupr

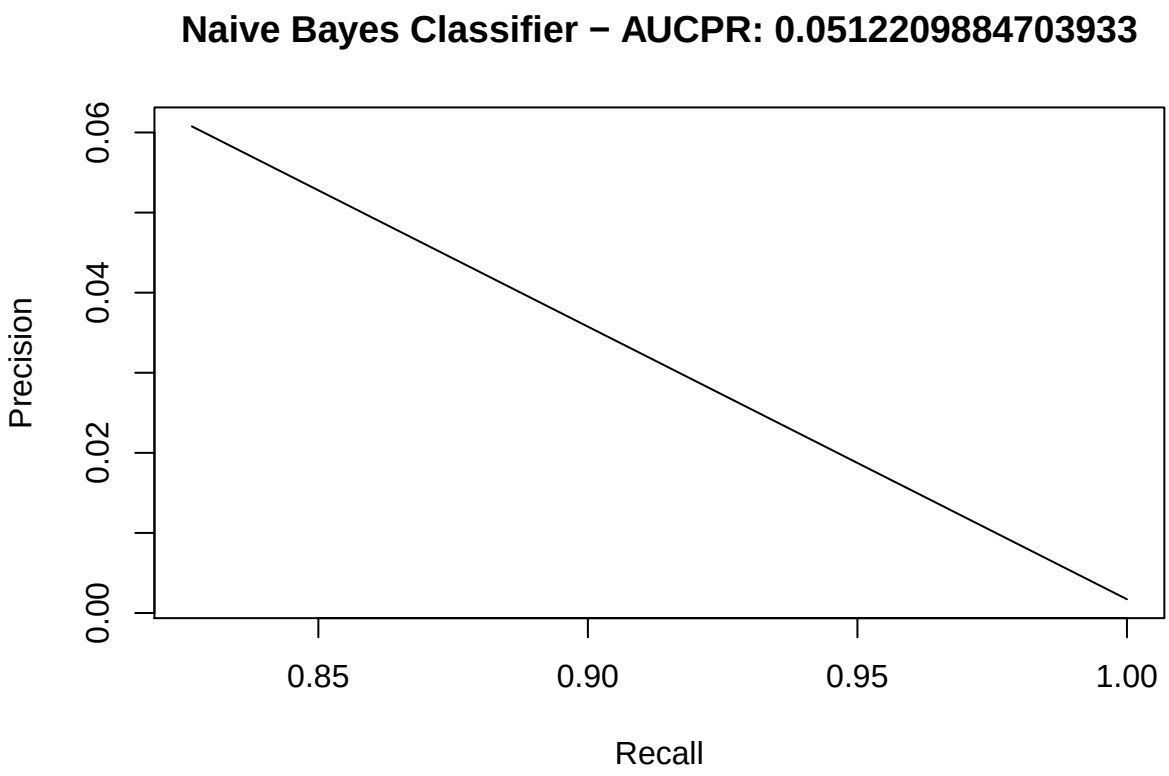
```
# Creating the relative plots

plot(aucpr_val_naive)
```




```
plot(auc_plot_naive, main=paste("Naive Bayes Classifier - AUC:", auc_val_naive@y.values[[1]]))
```




```
plot(aucpr_plot_naive, main=paste("Naive Bayes Classifier - AUCPR:", aucpr_val_naive$auc.integral))
```



Adding the respective metrics to the outcome dataset

```
# Adding the respective metrics to the outcome dataset

outcome <- outcome %>% add_row(
  Model = "Naive Bayes",
  AUC = auc_val_naive@y.values[[1]],
  AUCPR = aucpr_val_naive$auc.integral
)
```

Dipalying the outcome in a table

Display outcome in a table

```
outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed","responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.000000
Naive Bayes	0.9022564	0.051221

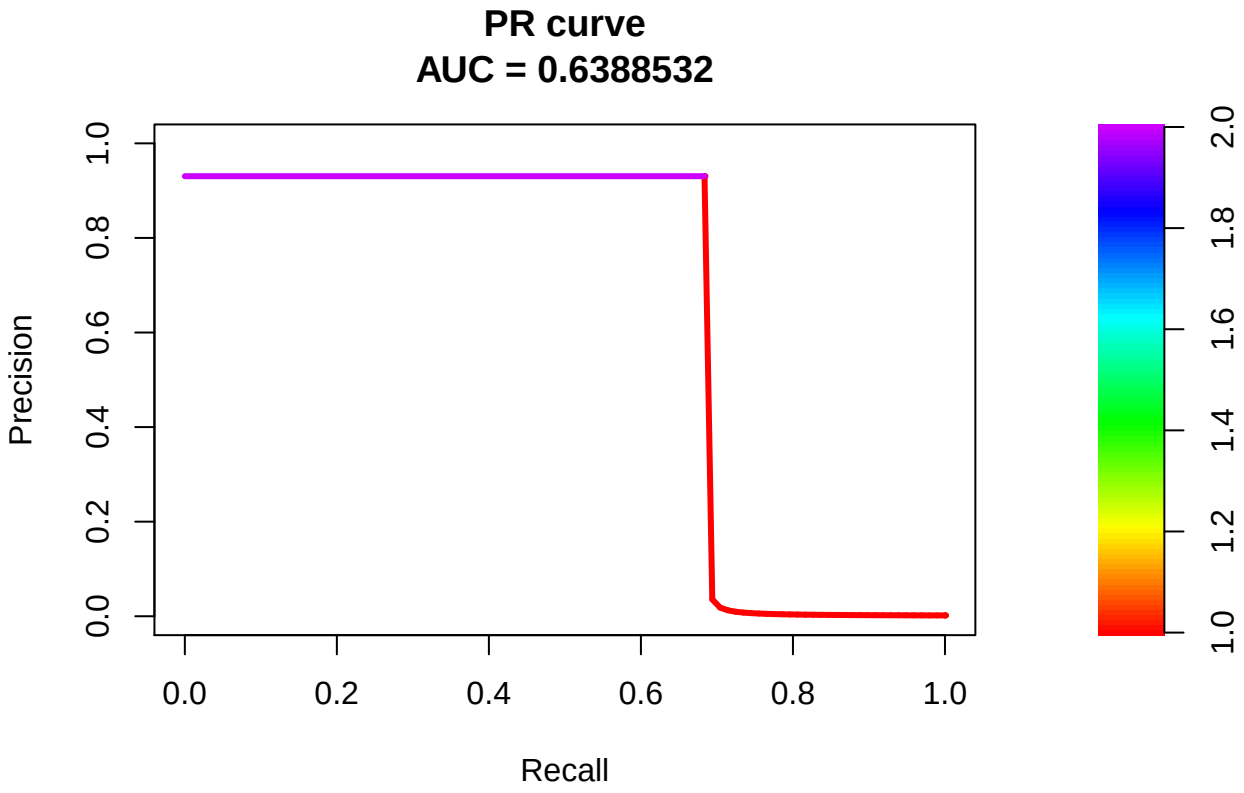
KNN - K - Nearest Neighbours

Using KNN Model, set k = 5 which achieves a significant improvement with respect to previous model. KNN is Supervised Learning Algorithm that use labeled input data set to predict the output of the data point. It is one of the most simple machine learning algorithm. It is mainly based on feature similarity. KNN is use here to check how similar a data point will be to its neighbour and classifies the data point. AUCPR and AUC prediction calculated by KNN is not what we are looking for so we will train another model

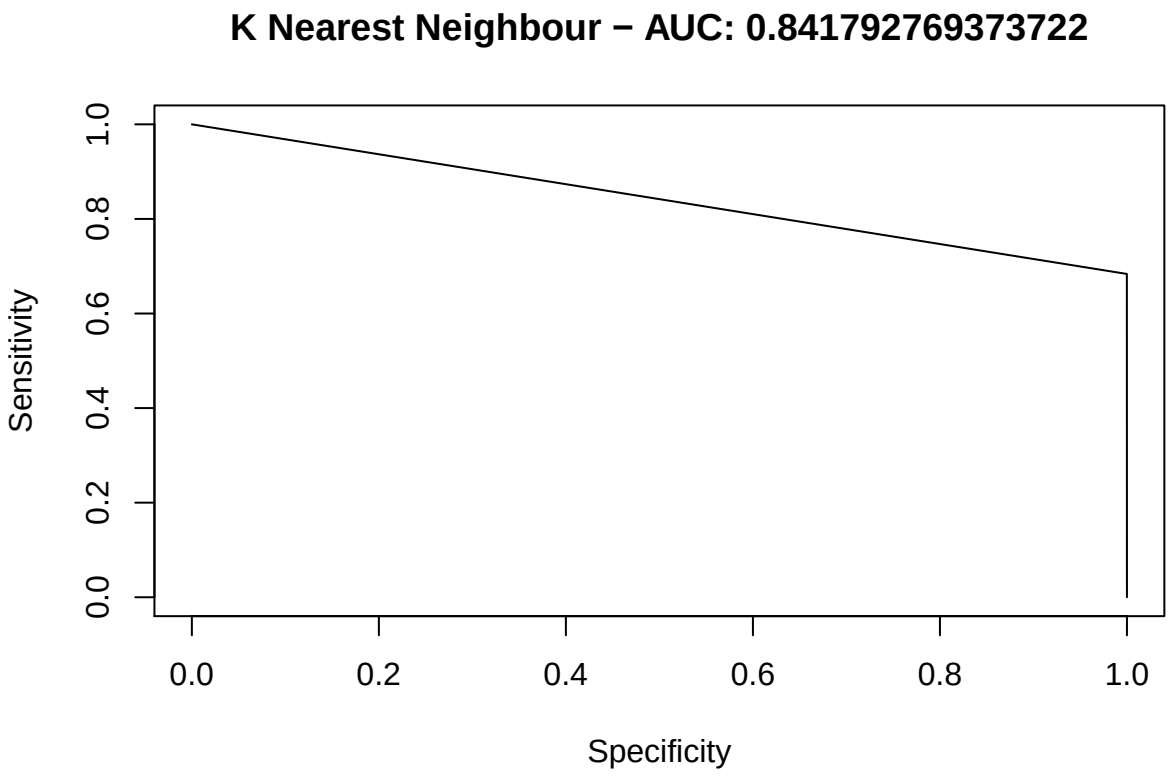
```
# KNN Model with class as target and all other variables as predictors where k is set to 5

knn_model <- knn(train[,-30], test[,-30], train$Class, k=5, prob = TRUE)

# computing AUC and AUCPR for KNN Model

knn_pred <- prediction(
  as.numeric(as.character(knn_model)),as.numeric(as.character(test$Class))
)

auc_val_knn <- performance(knn_pred, "auc")

auc_plot_knn <- performance(knn_pred, 'sens', 'spec')
aucpr_plot_knn <- performance(knn_pred, "prec", "rec")

aucpr_val_knn <- pr.curve(
  scores.class0 = knn_model[test$Class == 1],
  scores.class1 = knn_model[test$Class == 0],
  curve = T,
  dg.compute = T
)
```

Creating the relative plot for KNN Model

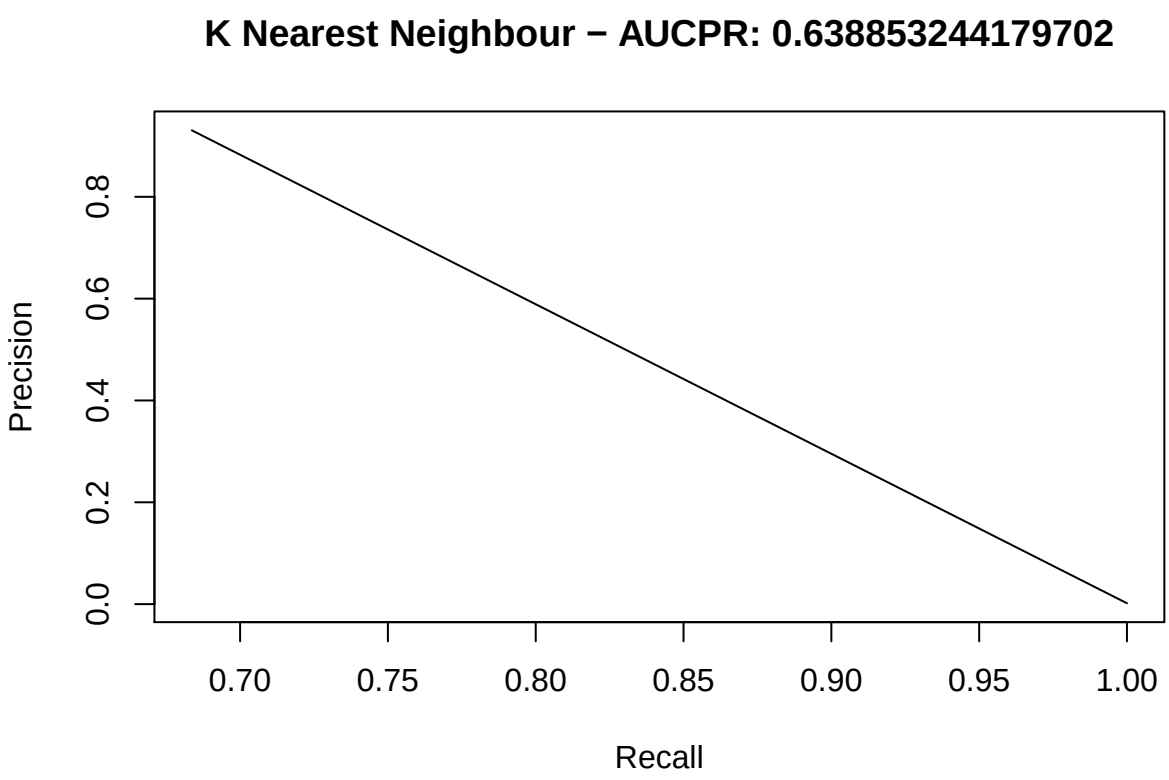
```
# Creating relative plot

plot(aucpr_val_knn)
```




```
plot(auc_plot_knn, main=paste("K Nearest Neighbour - AUC:", auc_val_knn@y.values[[1]]))
```




```
plot(aucpr_plot_knn, main=paste("K Nearest Neighbour - AUCPR:", aucpr_val_knn$auc.integral))
```



Add Metrics to the outcome dataset

```
# Adding metrics to the outcome dataset

outcome <- outcome %>% add_row(
  Model = "K-Nearest Neighbors k=5",
  AUC = auc_val_knn@y.values[[1]],
  AUCPR = aucpr_val_knn$auc.integral
)
```

Displaying the outcome in a table

```
# display outcome in table

outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed","responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.0000000
Naive Bayes	0.9022564	0.0512210
K-Nearest Neighbors k=5	0.8417928	0.6388532

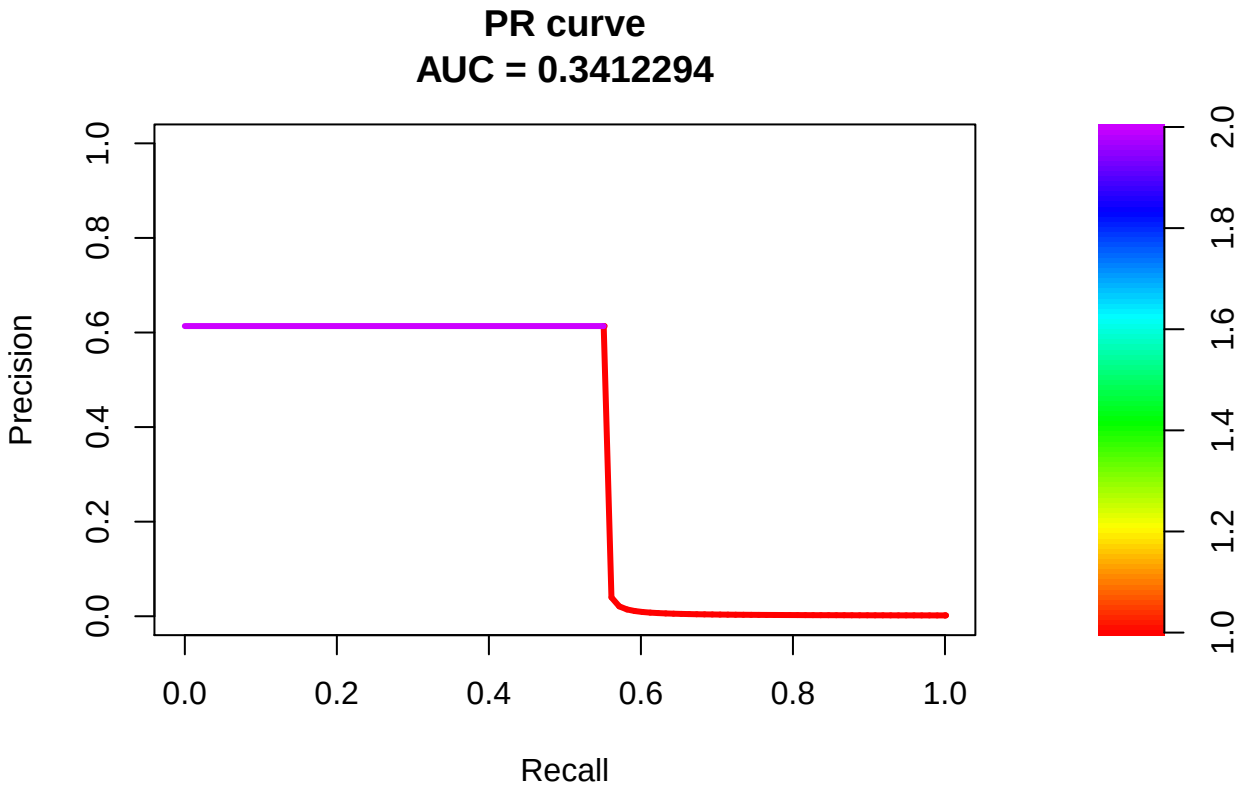
SVM Model - Support Vector Machine

Using SVM model with signmoid kernel will represents a step back on all fronts and does not give desirable outcome

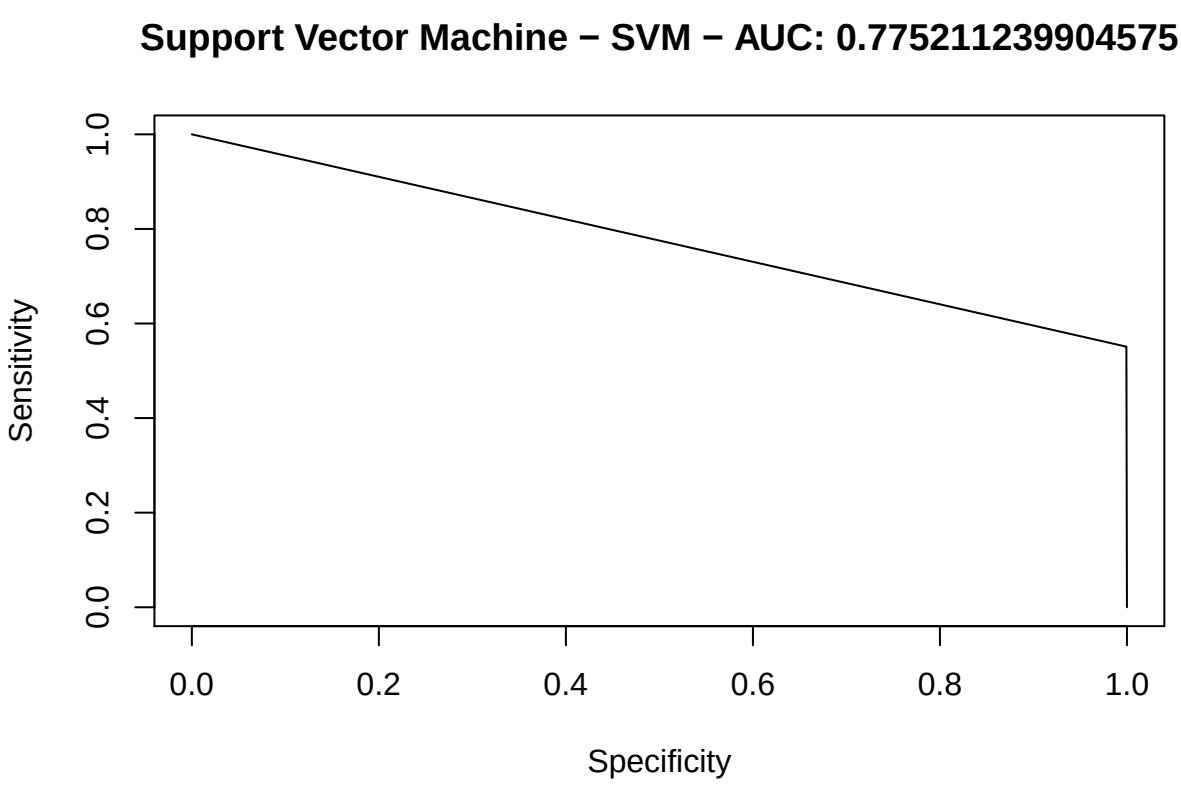
```
# Predicting through SVM Model - Support Vector Model with class as target and all other variables as p

svm_model <- svm(Class ~ ., data = train, kernel='sigmoid')

# Predicting bassed on this model

svm_model_predictions <- predict(svm_model, newdata=test)

# computing AUC and AUCPR through SVM Model predictions

svm_pred <- prediction(
  as.numeric(as.character(svm_model_predictions)), as.numeric(as.character(test$Class))
)

auc_val_svm <- performance(svm_pred, "auc")

auc_plot_svm <- performance(svm_pred, 'sens', 'spec')
aucpr_plot_svm <- performance(svm_pred, "prec", "rec")

aucpr_val_svm <- pr.curve(
  scores.class0 = svm_model_predictions[test$Class == 1],
  scores.class1 = svm_model_predictions[test$Class == 0],
  curve = T,
  dg.compute = T
)
```

Creating relative path for SVM Model

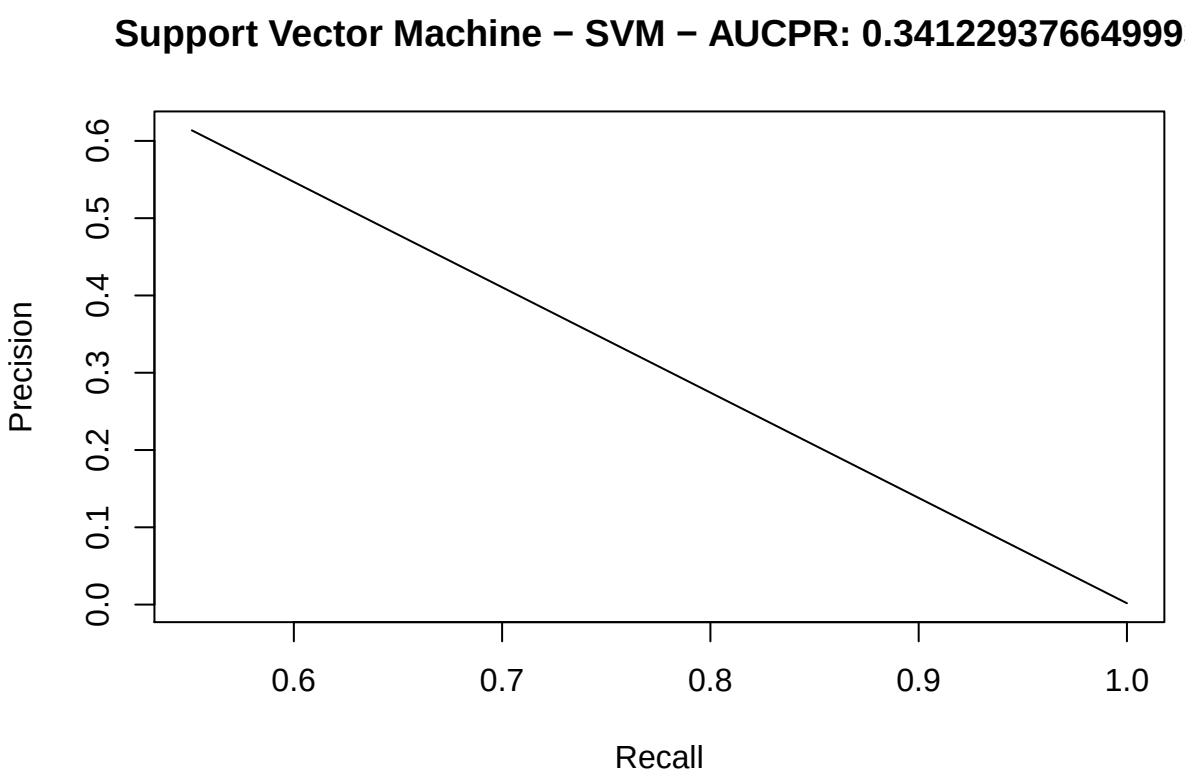
```
# Creating relative path for SVM Model

plot(aucpr_val_svm)
```




```
plot(auc_plot_svm, main=paste("Support Vector Machine - SVM - AUC:", auc_val_svm@y.values[[1]]))
```




```
plot(aucpr_plot_svm, main=paste("Support Vector Machine - SVM - AUCPR:", aucpr_val_svm$auc.integral))
```



Adding the metrics calculate to the outcome dataset

```
# Adding the metrics to the outcome dataset

outcome <- outcome %>% add_row(
  Model = "SVM - Support Vector Machine",
  AUC = auc_val_svm@y.values[[1]],
  AUCPR = aucpr_val_svm$auc.integral)
```

Display output in a table for SVM Model

```
# display output in a table

outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.0000000
Naive Bayes	0.9022564	0.0512210
K-Nearest Neighbors k=5	0.8417928	0.6388532
SVM - Support Vector Machine	0.7752112	0.3412294

Random Forest

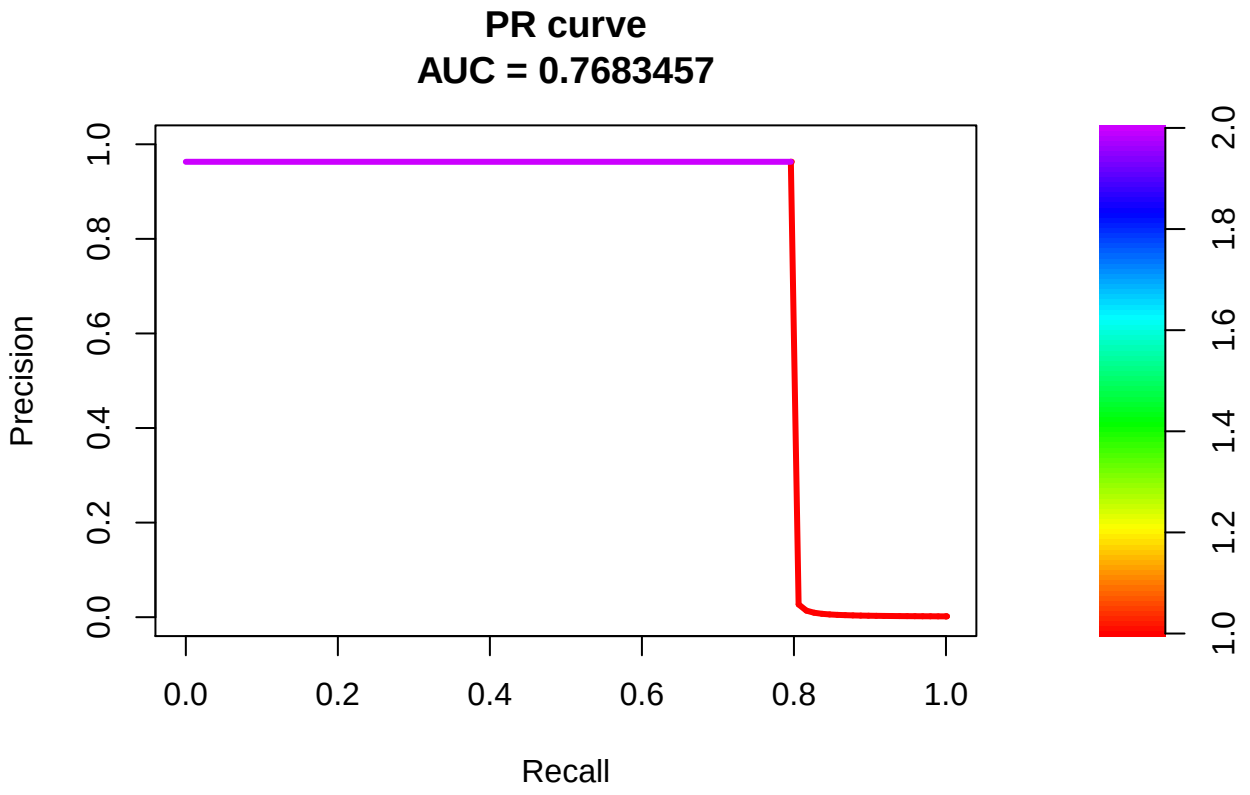
Random forest are a very popular machine learning approach that addresses that addresses shortcoming of decision trees using clever idea. The goal is to improve the prediction performance and reduce instability by averaging multiple decision trees (a forest of trees consutructed with randomness) Here we build the model with keeping Class as a target and all other variables as predictors and have set the number of trees to 500 for improving the prediction performance and reducing the instability. It uses two features that help in accomplishing this first step is boot strapper where the general idea is to generate many predictors each using regression or classification trees and then forming final prediction based on the average prediction of all the trees.

With little drop off in terms of AUC with respect to the Naive Bayes Model there is huge step forward in terms of AUCPR. This model does not reached desired performance. As the plot and table below suggest there are few predictors like V17, V12 and V14 that are particularly useful for classifying a fraud

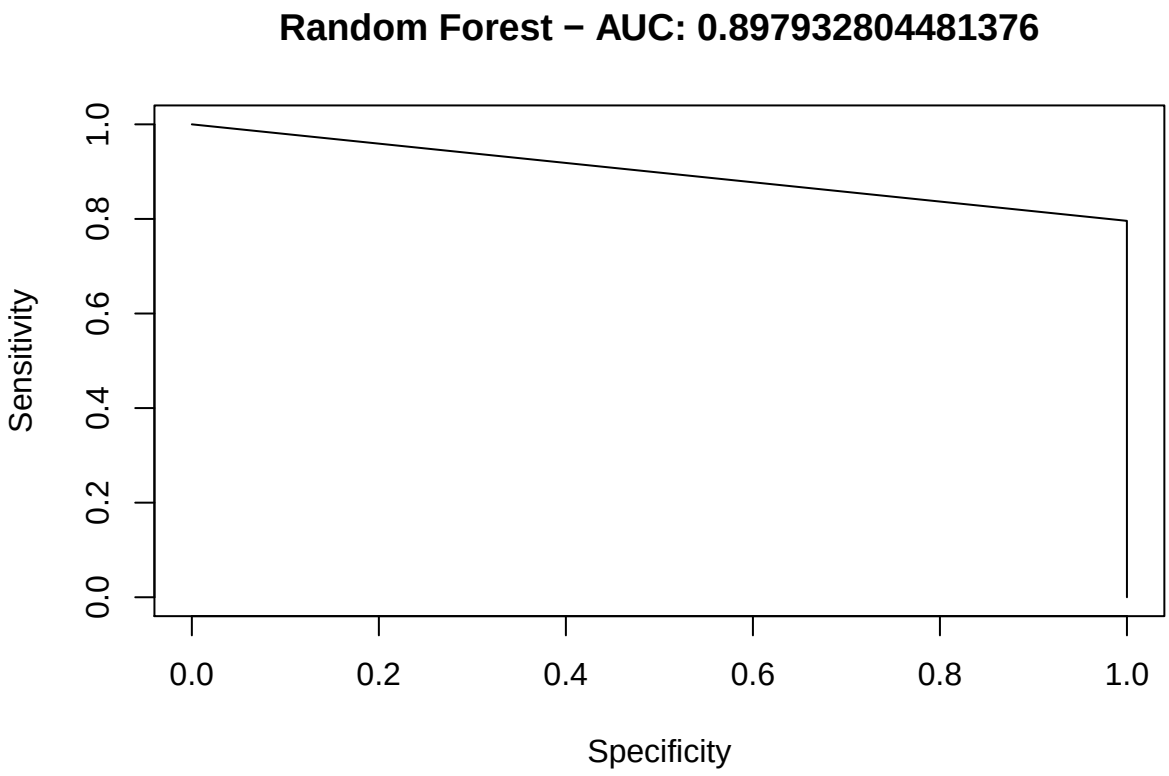
```
#Random forest Model creation with class as target and all other variables as predictors. For the model

random_forest_model <- randomForest(Class ~ ., data = train, ntree = 500)

# extracting the feature importance

feature_imp <- data.frame(importance(random_forest_model))

# Making prediction based on the model

random_predictions <- predict(random_forest_model, newdata=test)

# computing the AUC and AUCPR

random_pred <- prediction(
  as.numeric(as.character(random_predictions)),as.numeric(as.character(test$Class))
)

auc_val_rf <- performance(random_pred, "auc")

auc_plot_rf <- performance(random_pred, 'sens', 'spec')

aucpr_plot_rf <- performance(random_pred, "prec", "rec", curve = T,  dg.compute = T)

aucpr_val_rf <- pr.curve(scores.class0 = random_predictions[test$Class == 1], scores.class1 = random_pr
```

Creating relative path for the predictions generated through the random forest model

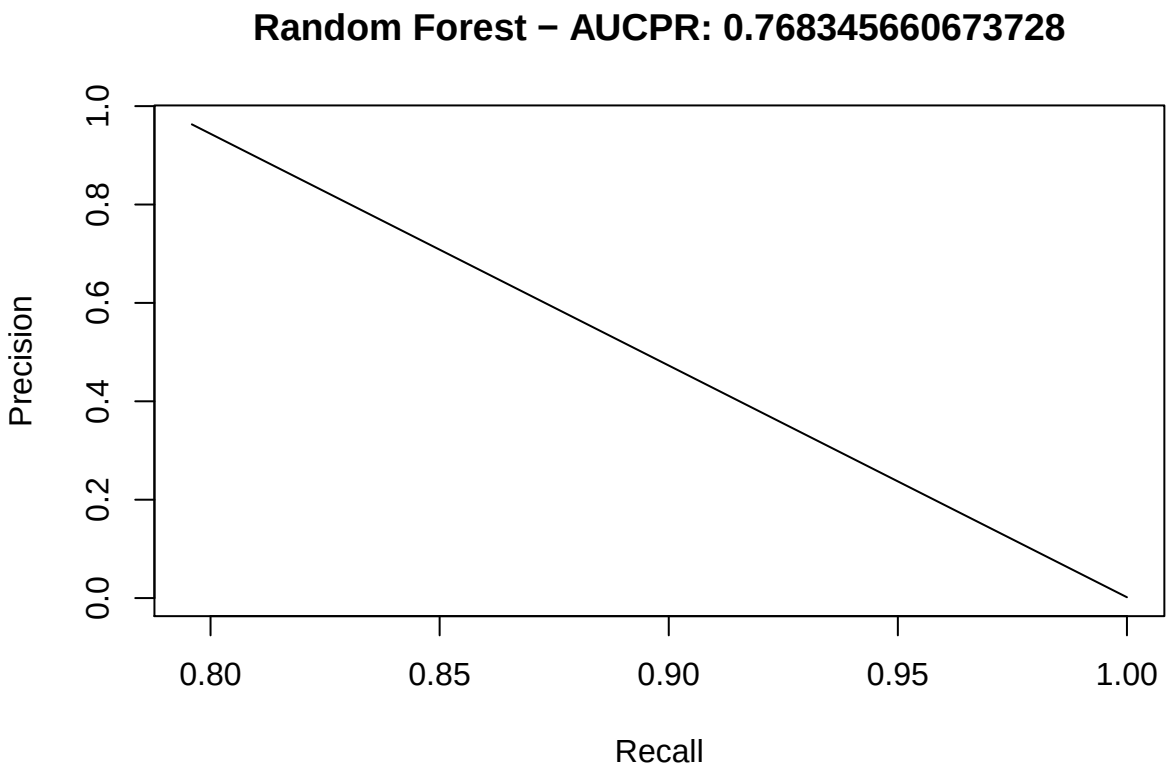
```r
# creating relative path for predictions generated through random forest model

plot(aucpr_val_rf)
```




```r
plot(auc_plot_rf, main=paste("Random Forest - AUC:", auc_val_rf@y.values[[1]]))
```




```
plot(aucpr_plot_rf, main=paste("Random Forest - AUCPR:", aucpr_val_rf$auc.integral))
```



Adding the metrics calculate to the outcome data set

```
# adding respective metrics generate to outcome data set

outcome <- outcome %>% add_row(
  Model = "Random Forest",
  AUC = auc_val_rf@y.values[[1]],
  AUCPR = aucpr_val_rf$auc.integral)
```

Display the output for random forest model to the table

```
# display output in a table

outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.0000000
Naive Bayes	0.9022564	0.0512210
K-Nearest Neighbors k=5	0.8417928	0.6388532
SVM - Support Vector Machine	0.7752112	0.3412294
Random Forest	0.8979328	0.7683457

Display the feature important in a table to identify fraud

```
# display feature importance in a table

feature_imp %>% kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",            "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

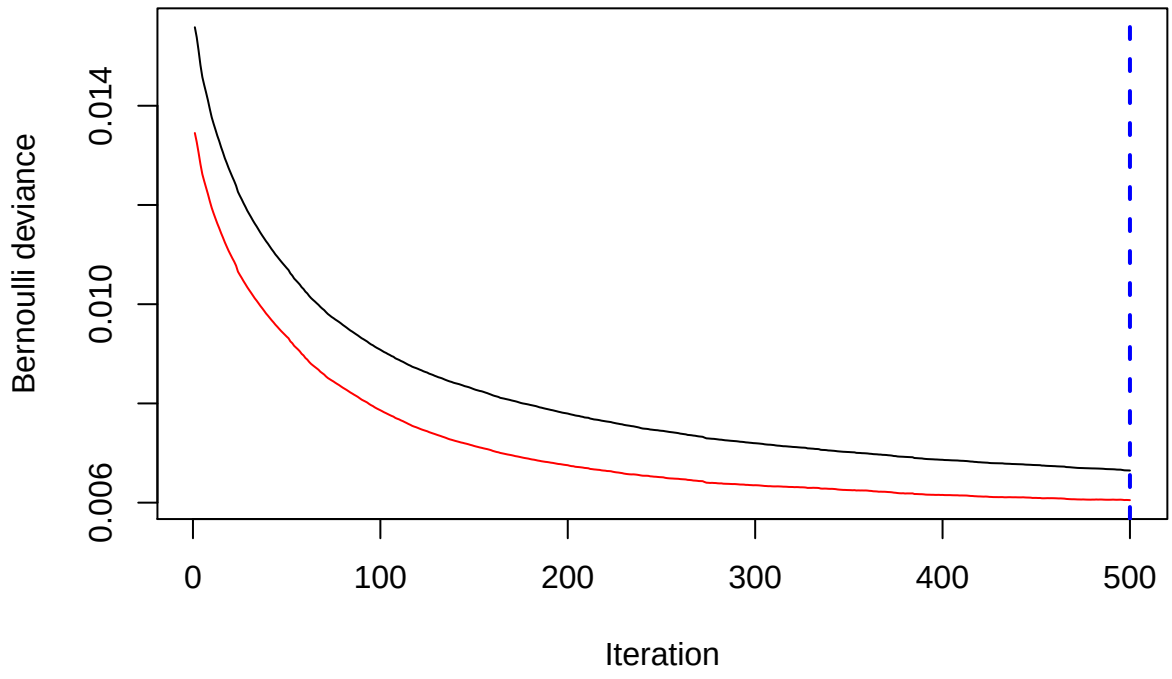
	MeanDecreaseGini
V1	9.377475
V2	7.866811
V3	10.882784
V4	18.728626
V5	6.996411
V6	8.439337
V7	14.278879
V8	7.526667
V9	22.408538
V10	43.630539
V11	41.841211
V12	67.475648
V13	6.423091
V14	65.938084
V15	7.203841
V16	44.158615
V17	104.873488
V18	18.041132
V19	7.626641
V20	7.966848
V21	8.688590
V22	7.686116
V23	5.167771
V24	6.386524
V25	5.320740
V26	12.511135
V27	8.836414
V28	7.351728
Amount	6.616101

GBM Model - Generalised Boosted Regression

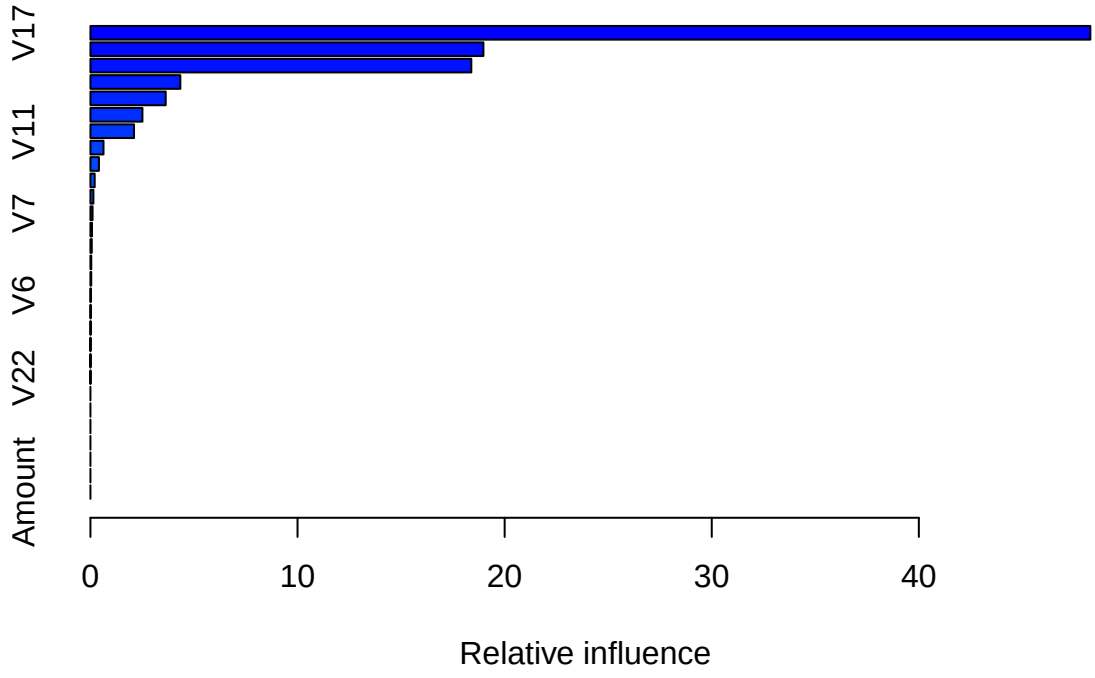
The accuracy of a predictive model can be boosted in two ways either through embracing or applying boosting algorithm right away. Hence, In order to improve the prediction further we will be using the Generalized Boosted Regression building based on Class as a Target and all other variable as predictors and we have set the distribution to bernoully and number of tree are set to 500

The GMB performance has came out really good. It does not achieve the desired value as the Random Forest Model shows with V17 and V14 which are still relevant to predict a fraud.

```
# GBM Modle - Generalized Boosted Model - building with class as traget and all other variables as pred
# distribution used is bernoully and number of tree set is 500

gbm_model <- gbm(as.character(Class) ~ .,
                 distribution = "bernoulli",
                 data = rbind(train, test),
                 n.trees = 500,
                 interaction.depth = 3,
                 n.minobsinnode = 100,
                 shrinkage = 0.01,
                 train.fraction = 0.7,
)

# identifying the best iteration using the test data

best_iteration <- gbm.perf(gbm_model, method = "test")
```



```r
# Calculating predictions based on the GBM Model

gbm_predictions = predict.gbm(
  gbm_model,
  newdata = test,
  n.trees = best_iteration,
  type="response"
)

# calculating feature importance

feature_imp_gbm_model = summary(gbm_model, n.trees = best_iteration)
```

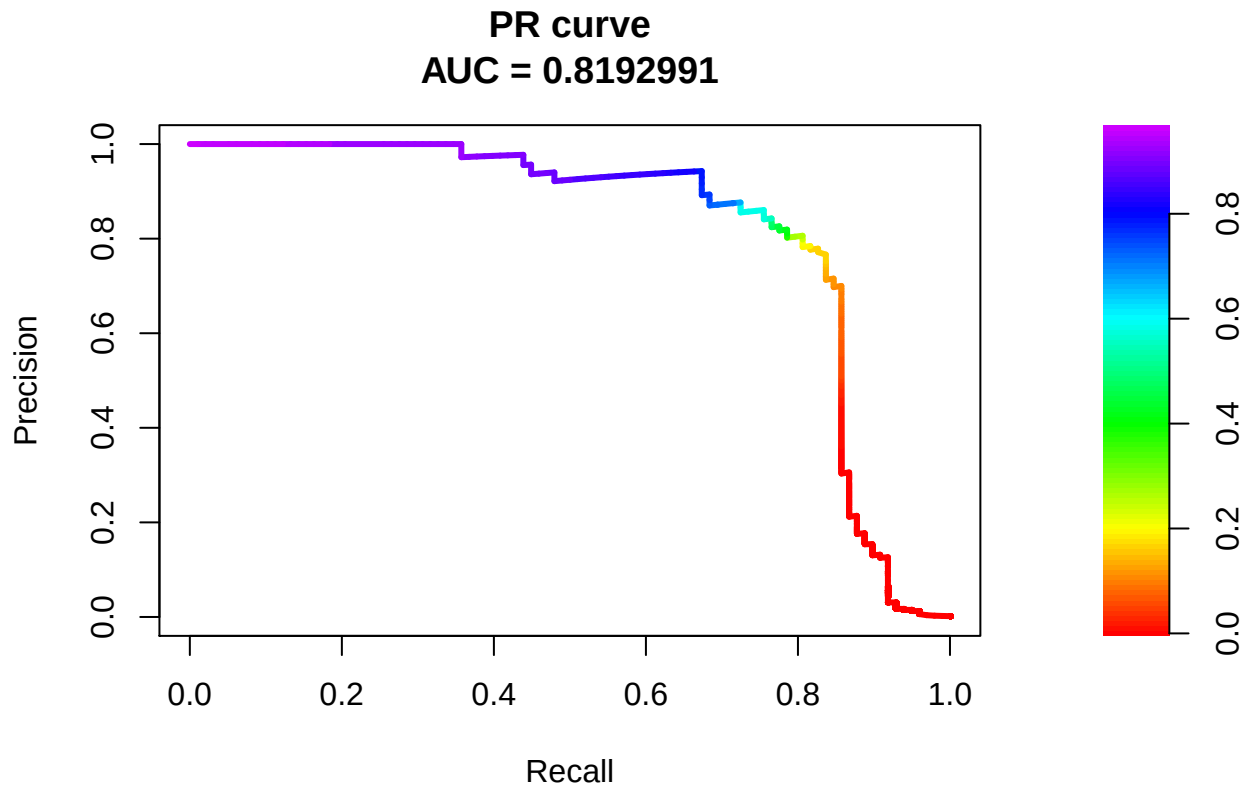



```r
# Calculating AUC and AUCPR through the prediction generated from GBM Model

gbm_pred <- prediction(
  as.numeric(as.character(gbm_predictions)), as.numeric(as.character(test$Class))
)

auc_val_gbm <- performance(gbm_pred, "auc")

auc_plot_gbm <- performance(gbm_pred, 'sens', 'spec')
aucpr_plot_gbm <- performance(gbm_pred, "prec", "rec")

aucpr_val_gbm <- pr.curve(
```

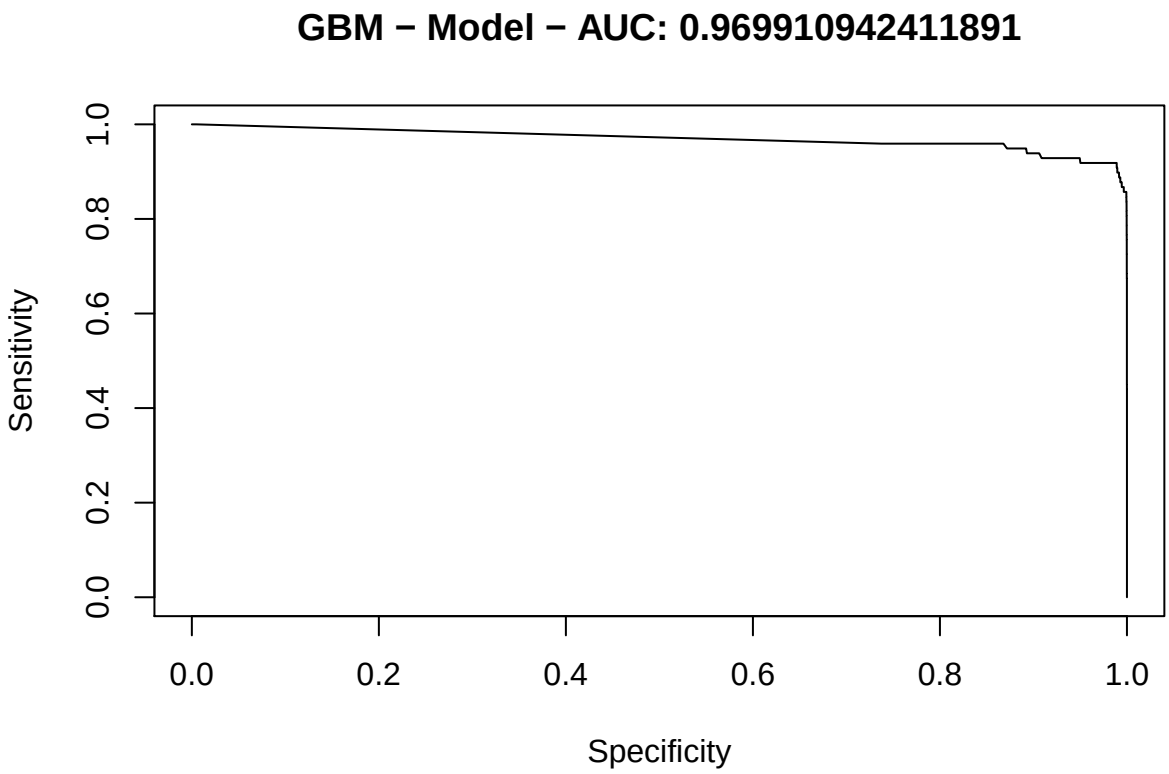
```
  scores.class0 = gbm_predictions[test$Class == 1],
  scores.class1 = gbm_predictions[test$Class == 0],
  curve = T,
  dg.compute = T
)
```

Generating relative path for GBM Model

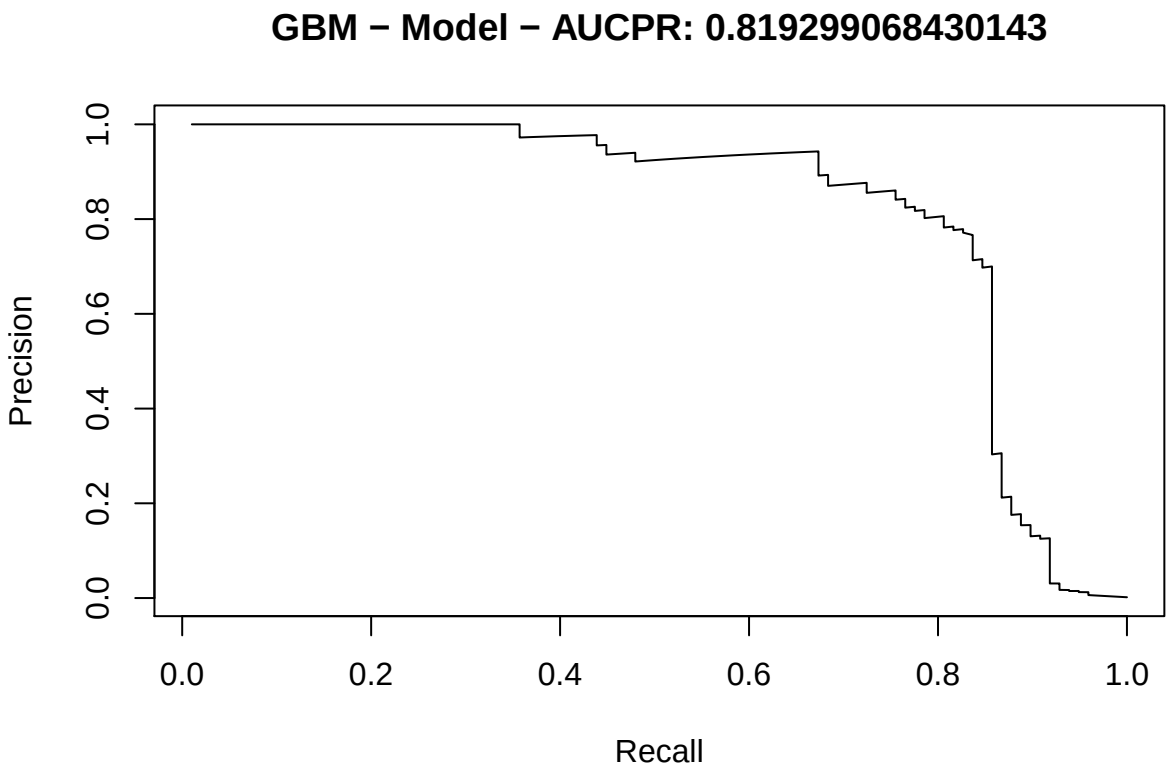
```
# generating relative path plot

plot(aucpr_val_gbm)
```




```
plot(auc_plot_gbm, main=paste("GBM - Model - AUC:", auc_val_gbm@y.values[[1]]))
```




```
plot(aucpr_plot_gbm, main=paste("GBM - Model - AUCPR:", aucpr_val_gbm$auc.integral))
```



Adding respective metrics to the outcome datasets

```
# Adding respective metrics to the outcome datasets

outcome <- outcome %>% add_row(
  Model = "GBM - Generalized Boosted Regression",
  AUC = auc_val_gbm@y.values[[1]],
  AUCPR = aucpr_val_gbm$auc.integral)
```

Display the outcome in a table with GBM Model Calculation

```
# display outcome in a table

outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.0000000
Naive Bayes	0.9022564	0.0512210
K-Nearest Neighbors k=5	0.8417928	0.6388532
SVM - Support Vector Machine	0.7752112	0.3412294
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9699109	0.8192991

Display the feature importance generated through the GBM Model

```
# dispaly the feature importance in a table

feature_imp_gbm_model %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

	var	rel.inf
V17	V17	48.2815625
V14	V14	18.9756902
V12	V12	18.3904223
V10	V10	4.3411764
V9	V9	3.6350484
V20	V20	2.5113555
V11	V11	2.1036100
V4	V4	0.6303354
V3	V3	0.4107263
V18	V18	0.2052700
V26	V26	0.1423641
V7	V7	0.1023950
V28	V28	0.0751979
V16	V16	0.0645817
V24	V24	0.0399136
V8	V8	0.0390014
V6	V6	0.0167732
V1	V1	0.0129378
V23	V23	0.0121842
V19	V19	0.0039095
V27	V27	0.0032196
V22	V22	0.0019869
V21	V21	0.0003380
V2	V2	0.0000000
V5	V5	0.0000000
V13	V13	0.0000000
V15	V15	0.0000000
V25	V25	0.0000000
Amount	Amount	0.0000000

XGBoost

eXtreme Gradient Boosting is an especially efficient implimentation of gradient boosting. It is powerful tool for classifcation and regression.

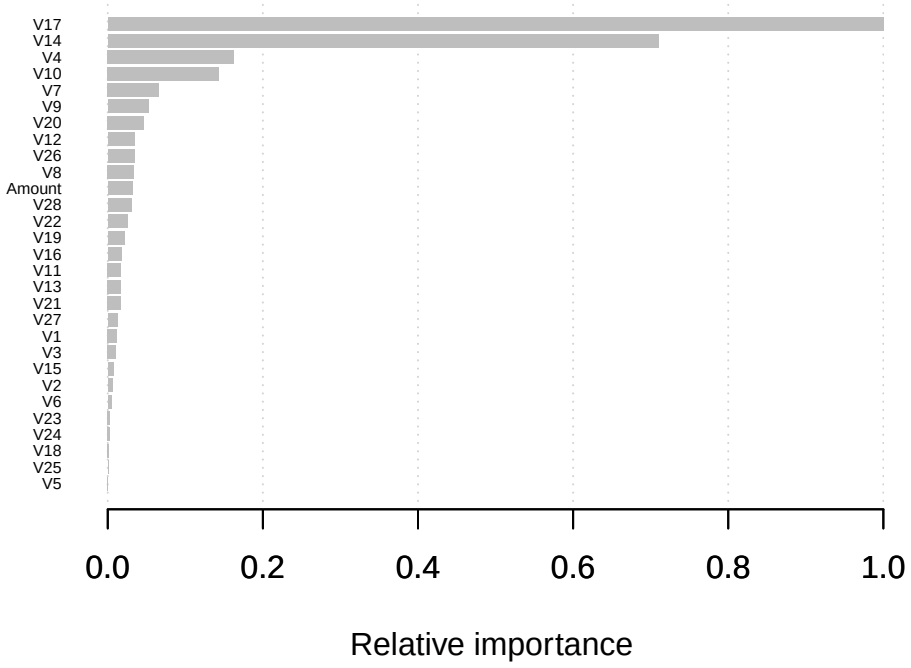
XGB is top class model and always stays on TOP5 or wins them in every competitions on Kaggles and even in this case too. It is very fast to train and performance are good. it reach the desired preformance as expected

```
# Building / Preparing the training dataset for XGB Model


xgb_model_train <- xgb.DMatrix(
  as.matrix(train[, colnames(train) != "Class"]),
  label = as.numeric(as.character(train$Class))
)

# building / Preparing the test data set for XGB Model - XG Boost Model

xgb_model_test <- xgb.DMatrix(
  as.matrix(test[, colnames(test) != "Class"]),
  label = as.numeric(as.character(test$Class))
)

# building and preparing the cv data set for XGB Model

xgb_model_cv <- xgb.DMatrix(
  as.matrix(cv[, colnames(cv) != "Class"]),
  label = as.numeric(as.character(cv$Class))
)

# Parameter list preparation

xgb_model_params<- list(
  objective = "binary:logistic",
  eta = 0.1,
  max.depth = 3,
  nthread = 6,
  eval_metric = "aucpr"
)

# Train the XGBoost Model

xgb_model <- xgb.train(
  data = xgb_model_train,
  params = xgb_model_params,
  watchlist = list(test = xgb_model_test, cv = xgb_model_cv),
  nrounds = 500,
  early_stopping_rounds = 40,
  print_every_n = 20
)
```

```
## [1]  test-aucpr:0.725214 cv-aucpr:0.711143
## Multiple eval metrics are present. Will use cv_aucpr for early stopping.
```

```
## Will train until cv_aucpr hasn't improved in 40 rounds.
##
## [21] test-aucpr:0.800625 cv-aucpr:0.801756
## [41] test-aucpr:0.837654 cv-aucpr:0.847612
## [61] test-aucpr:0.848483 cv-aucpr:0.850451
## [81] test-aucpr:0.852719 cv-aucpr:0.847156
## Stopping. Best iteration:
## [51] test-aucpr:0.843432 cv-aucpr:0.853041
```

```r
# Calculating feature importance

feature_xgb_imp <- xgb.importance(colnames(train), model = xgb_model)

# plot relative importance

xgb.plot.importance(feature_xgb_imp, rel_to_first = TRUE, xlab = "Relative importance")
```

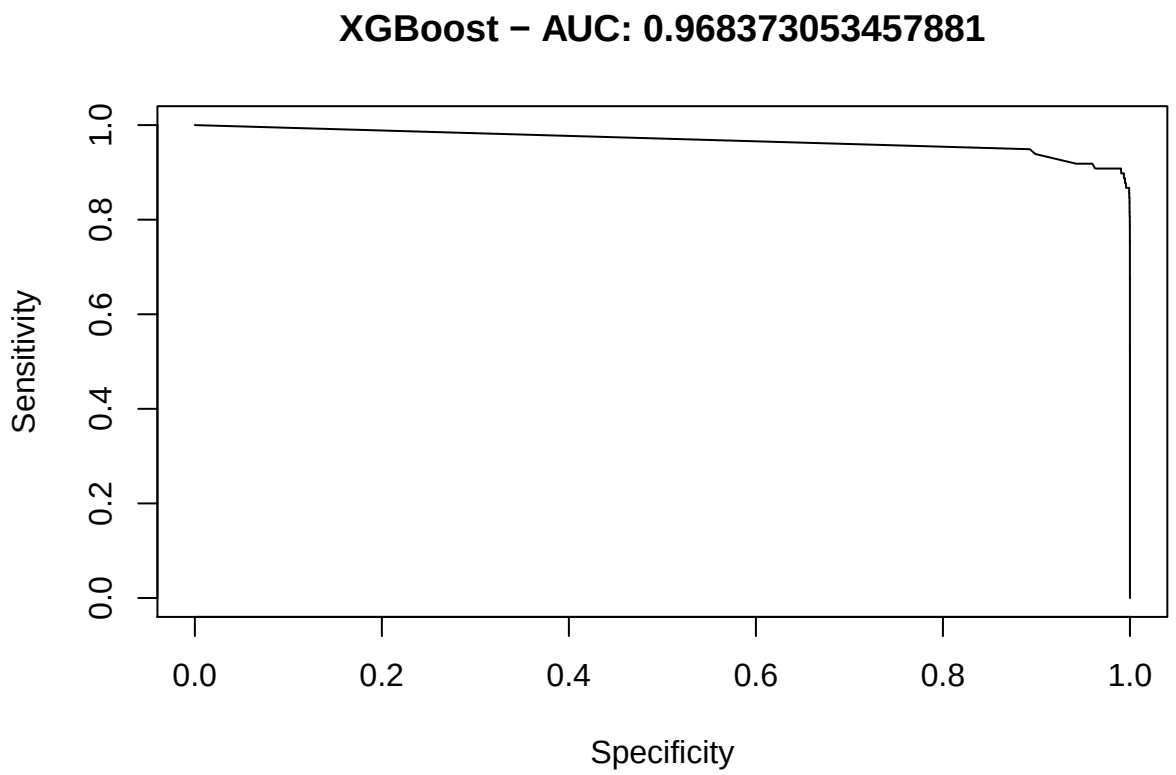



```r
# Calculate predictions based on the XGB Model

xgb_predictions = predict(
  xgb_model,
  newdata = as.matrix(test[, colnames(test) != "Class"]),
  ntreelimit = xgb_model$bestInd
)
```

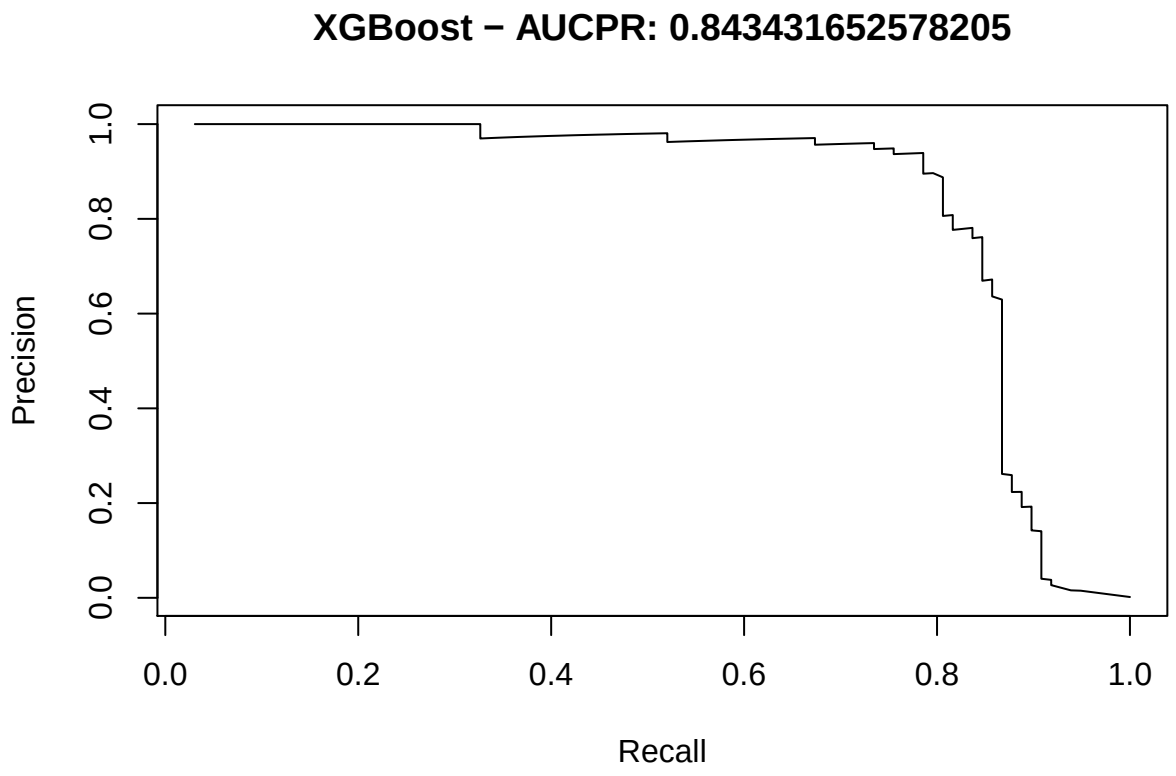
```r
# calculating the AUC and AUCPR

xgb_pred <- prediction(
  as.numeric(as.character(xgb_predictions)),as.numeric(as.character(test$Class))
)

auc_val_xgb <- performance(xgb_pred, "auc")

auc_plot_xgb <- performance(xgb_pred, 'sens', 'spec')
aucpr_plot_xgb <- performance(xgb_pred, "prec", "rec")

aucpr_val_xgb <- pr.curve(
  scores.class0 = xgb_predictions[test$Class == 1],
  scores.class1 = xgb_predictions[test$Class == 0],
  curve = T,
  dg.compute = T
)
```

Generating the relative plot for XGBoost

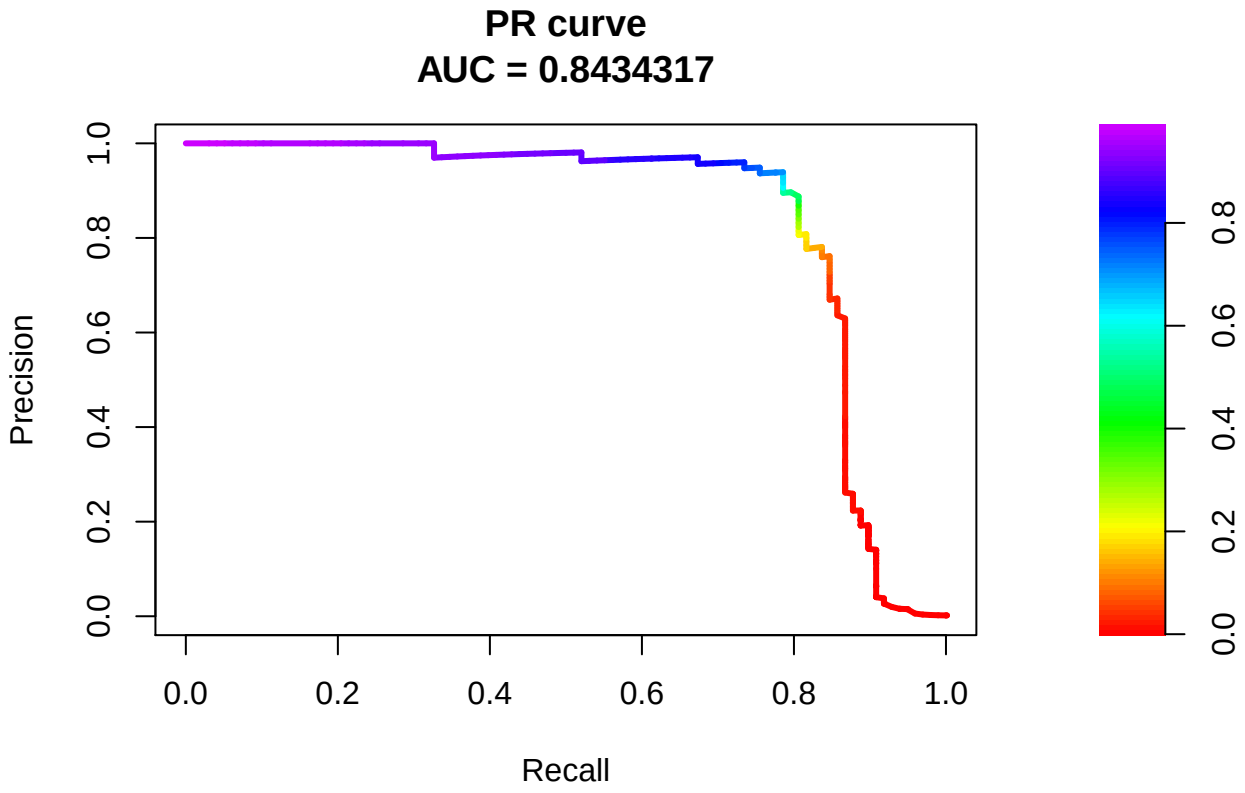
```r
# Relative Plot

plot(auc_plot_xgb, main=paste("XGBoost - AUC:", auc_val_xgb@y.values[[1]]))
```



```r
plot(aucpr_plot_xgb, main=paste("XGBoost - AUCPR:", aucpr_val_xgb$auc.integral))
```




```r
plot(aucpr_val_xgb)
```



Display Feature importance in a table

```
# display feature importance in a table

feature_xgb_imp %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Feature	Gain	Cover	Frequency	Importance
V17	0.3959880	0.3941438	0.0964052	0.3959880
V14	0.2810706	0.5349960	0.1813725	0.2810706
V4	0.0644815	0.0098961	0.1176471	0.0644815
V10	0.0568661	0.0008738	0.0637255	0.0568661
V7	0.0261027	0.0012996	0.0490196	0.0261027
V9	0.0208205	0.0005325	0.0392157	0.0208205
V20	0.0184294	0.0026758	0.0457516	0.0184294
V12	0.0136861	0.0357165	0.0310458	0.0136861
V26	0.0136401	0.0002758	0.0212418	0.0136401
V8	0.0134695	0.0005570	0.0408497	0.0134695
Amount	0.0127395	0.0005508	0.0375817	0.0127395
V28	0.0120635	0.0002257	0.0261438	0.0120635
V22	0.0101375	0.0049439	0.0310458	0.0101375
V19	0.0085145	0.0006109	0.0359477	0.0085145
V16	0.0069648	0.0002021	0.0294118	0.0069648
V11	0.0068815	0.0101251	0.0261438	0.0068815
V13	0.0068449	0.0003521	0.0179739	0.0068449
V21	0.0067239	0.0012151	0.0196078	0.0067239
V27	0.0049513	0.0001101	0.0130719	0.0049513
V1	0.0048349	0.0000670	0.0147059	0.0048349
V3	0.0043286	0.0002623	0.0163399	0.0043286
V15	0.0028782	0.0001186	0.0098039	0.0028782
V2	0.0023495	0.0000555	0.0081699	0.0023495
V6	0.0020027	0.0000856	0.0130719	0.0020027
V23	0.0010980	0.0000470	0.0049020	0.0010980
V24	0.0010280	0.0000322	0.0032680	0.0010280
V18	0.0006578	0.0000072	0.0016340	0.0006578
V25	0.0003875	0.0000162	0.0016340	0.0003875
V5	0.0000589	0.0000058	0.0032680	0.0000589

Results

Below is the summary results for all the models which are build, trained and validated:

```
# adding the respective metrics to the outcome datasets

outcome <- outcome %>% add_row(
  Model = "XGBoost",
  AUC = auc_val_xgb@y.values[[1]],
  AUCPR = aucpr_val_xgb$auc.integral)

# display outcome in a table

outcome %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",           "responsive"),
                position = "center",
                font_size = 10,
                full_width = FALSE)
```

Model	AUC	AUCPR
Baseline - Predict Always Legal	0.5000000	0.0000000
Naive Bayes	0.9022564	0.0512210
K-Nearest Neighbors k=5	0.8417928	0.6388532
SVM - Support Vector Machine	0.7752112	0.3412294
Random Forest	0.8979328	0.7683457
GBM - Generalized Boosted Regression	0.9699109	0.8192991
XGBoost	0.9683731	0.8434317

Conclusion

In this task with the help of XGBoost model we were able to achieve desierable results or outcome for our dataset and predict the frauds within the dataset. Further other model build, train and validate can also give more accurate results and can helpful for further fraud detections.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.