Credit Fraud Detection Analysis

Arjun Sekhar

2023-12-26

Introduction

The Credit Fraud Detection data is derived from a project in Kaggle (see link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data) with the intention of decoding and modelling the transactions from September 2013 by European cardholders. The data itself is drawn from transactions across two calendar days, comprising of 492 fraudulent transactions from 284,807 transactions. Note that this data contains only numeric variables, which are a resultant of a Principal Component Analysis (PCA). This means the original raw data features are unknown for the purposes of this exercise.

Data Preparation

As preparation we establish some of the packages that will come into use in this analysis. This is outlined in the below chunk of R.

```r
library(dplyr)
library(tidyverse)
library(pander)
library(corrplot)
library(caTools)
library(DHARMa)
library(caret)
library(pROC)
library(PRROC)
library(glmnet)
```

The next step involves introducing the `ratings_for_upload.csv` data set using the `read.csv()` function. Our aim here is to analyse and understand the data, as well as the variables that are at our disposal. This will segway into the exploratory data analysis (EDA), allowing us to pursue a model building strategy.

```r
# Set the working directory
setwd("/Users/arjunsekhar/OneDrive/Knowledge/Courses/Kaggle/credit-fraud-
analysis")

# Input the data
credit <- read.csv('creditcard.csv', header = TRUE)
pander(head(credit))
```

Table continues below

Time	V1	V2	V3	V4	V5	V6	V7
0	-1.36	-0.07278	2.536	1.378	-0.3383	0.4624	0.2396
0	1.192	0.2662	0.1665	0.4482	0.06002	-0.08236	-0.0788
1	-1.358	-1.34	1.773	0.3798	-0.5032	1.8	0.7915
1	-0.9663	-0.1852	1.793	-0.8633	-0.01031	1.247	0.2376
2	-1.158	0.8777	1.549	0.403	-0.4072	0.09592	0.5929
2	-0.426	0.9605	1.141	-0.1683	0.421	-0.02973	0.4762

Table continues below

V8	V9	V10	V11	V12	V13	V14	V15
0.0987	0.3638	0.09079	-0.5516	-0.6178	-0.9914	-0.3112	1.468
0.0851	-0.2554	-0.167	1.613	1.065	0.4891	-0.1438	0.6356
0.2477	-1.515	0.2076	0.6245	0.06608	0.7173	-0.1659	2.346
0.3774	-1.387	-0.05495	-0.2265	0.1782	0.5078	-0.2879	-0.6314
-0.2705	0.8177	0.7531	-0.8228	0.5382	1.346	-1.12	0.1751
0.2603	-0.5687	-0.3714	1.341	0.3599	-0.3581	-0.1371	0.5176

Table continues below

V16	V17	V18	V19	V20	V21	V22
-0.4704	0.208	0.02579	0.404	0.2514	-0.01831	0.2778
0.4639	-0.1148	-0.1834	-0.1458	-0.06908	-0.2258	-0.6387
-2.89	1.11	-0.1214	-2.262	0.525	0.248	0.7717
-1.06	-0.6841	1.966	-1.233	-0.208	-0.1083	0.005274
-0.4514	-0.237	-0.03819	0.8035	0.4085	-0.009431	0.7983
0.4017	-0.05813	0.06865	-0.03319	0.08497	-0.2083	-0.5598

V23	V24	V25	V26	V27	V28	Amount	Class
-0.1105	0.06693	0.1285	-0.1891	0.1336	-0.02105	149.6	0
0.1013	-0.3398	0.1672	0.1259	-0.008983	0.01472	2.69	0
0.9094	-0.6893	-0.3276	-0.1391	-0.05535	-0.05975	378.7	0
-0.1903	-1.176	0.6474	-0.2219	0.06272	0.06146	123.5	0
-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	69.99	0
-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.08108	3.67	0

From the data presented, a total of 284807 transactions of 31 variables are recorded, which is a representation of credit card transactions in September 2013 by European credit card holders. As mentioned in the context of this task, the data has been dealt with Principal Component Analysis (PCA) transformations extensively, this limits the extent of information available.

Exploratory Data Analysis (EDA)

A) Spread of values in 'Class' column

Firstly with the `Class` column, we can alter the summary by identifying the factors as `Non Fraudulent` and `Fraudulent` transactions. Upon doing so, this summary can be presented as a table using `pander`, which is a neater way of displaying the information in LaTeX format.

```r
credit$Class <- as.factor(credit$Class)
levels(credit$Class) <- c('Non Fraudulent', 'Fraudulent')

credit_class <- credit %>%
  group_by(Class) %>%
  summarise(Total = n()) %>%
  mutate(Frequency = round(Total/sum(Total), 5)) %>%
  arrange(desc(Frequency))

pander(credit_class)
```

Class	Total	Frequency
Non Fraudulent	284315	0.9983
Fraudulent	492	0.00173

From the above we can see how the data provided is unbalanced. Despite the context provided acting as a foreshadow to this exploratory revelation, it can be foreshadowed from an everyday perspective since we would anticipate most transactions to be non fraudulent. Given the intention is to measure the accuracy, the Area Under the Curve (AUC) concept will be applied as the accuracy measure.

B) Missing Values

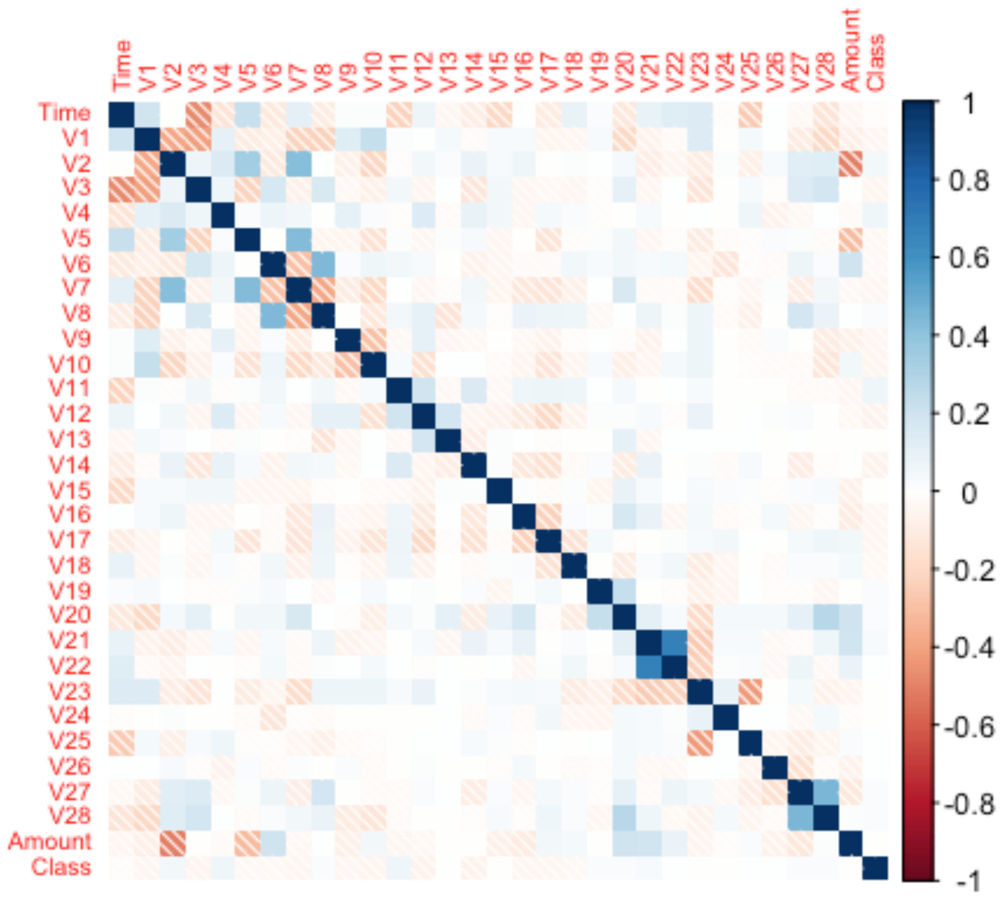
The next step is to analyse the proportion of missing values from the context of this data set.

```r
# Quantity of Missing Values
credit %>%
  is.na() %>%
  sum()

## [1] 0
```

From the above we can see how there are no missing values. Although such a reading is rare, in this context it can be attributed to the data preparation when the data was provided.

C) Correlation Matrix

```
credit %>% duplicated() %>% sum()

## [1] 1081

credit2 <- credit
credit2$Class <- as.numeric(credit2$Class)


credit_correlation <- cor(credit2[], method = 'spearman')
corrplot(credit_correlation, method = 'shade', tl.cex = 0.65)
```



Due to the the author applying the Principal Component Analysis prior to the publishing of this data, it can be seen from the above that most variables are not correlated with one another.

Data Modelling

A) Splitting the data set into train and test sets

The next step is useful for the purpose of our model building and subsequent analysis. The first goal is to standardise the data, so as to ensure that the values are scaled according to a specific range, negating the existence of extreme values to our analysis. Note that the `set.seed()` is used to ensure data reproducibility using random numbers. Upon doing this, we split the data into train and test data sets, with 80% of values used in the former and 20% in the latter.

```r
# Standardising the data
credit$Amount <- scale(credit$Amount)
credit3 <- credit[,c(-1)]
pander(head(credit3))
```

Table continues below

V1	V2	V3	V4	V5	V6	V7
-1.36	-0.07278	2.536	1.378	-0.3383	0.4624	0.2396
1.192	0.2662	0.1665	0.4482	0.06002	-0.08236	-0.0788
-1.358	-1.34	1.773	0.3798	-0.5032	1.8	0.7915
-0.9663	-0.1852	1.793	-0.8633	-0.01031	1.247	0.2376
-1.158	0.8777	1.549	0.403	-0.4072	0.09592	0.5929
-0.426	0.9605	1.141	-0.1683	0.421	-0.02973	0.4762

Table continues below

V8	V9	V10	V11	V12	V13	V14	V15
0.0987	0.3638	0.09079	-0.5516	-0.6178	-0.9914	-0.3112	1.468
0.0851	-0.2554	-0.167	1.613	1.065	0.4891	-0.1438	0.6356
0.2477	-1.515	0.2076	0.6245	0.06608	0.7173	-0.1659	2.346
0.3774	-1.387	-0.05495	-0.2265	0.1782	0.5078	-0.2879	-0.6314
-0.2705	0.8177	0.7531	-0.8228	0.5382	1.346	-1.12	0.1751
0.2603	-0.5687	-0.3714	1.341	0.3599	-0.3581	-0.1371	0.5176

Table continues below

V16	V17	V18	V19	V20	V21	V22
-0.4704	0.208	0.02579	0.404	0.2514	-0.01831	0.2778
0.4639	-0.1148	-0.1834	-0.1458	-0.06908	-0.2258	-0.6387
-2.89	1.11	-0.1214	-2.262	0.525	0.248	0.7717
-1.06	-0.6841	1.966	-1.233	-0.208	-0.1083	0.005274
-0.4514	-0.237	-0.03819	0.8035	0.4085	-0.009431	0.7983
0.4017	-0.05813	0.06865	-0.03319	0.08497	-0.2083	-0.5598

Table continues below

V23	V24	V25	V26	V27	V28	Amount
-0.1105	0.06693	0.1285	-0.1891	0.1336	-0.02105	0.245
0.1013	-0.3398	0.1672	0.1259	-0.008983	0.01472	-0.3425
0.9094	-0.6893	-0.3276	-0.1391	-0.05535	-0.05975	1.161
-0.1903	-1.176	0.6474	-0.2219	0.06272	0.06146	0.1405
-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	-0.0734
-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.08108	-0.3386

Class
Non Fraudulent
Non Fraudulent
Non Fraudulent
Non Fraudulent
Non Fraudulent
Non Fraudulent

```r
# Random Number Generator
set.seed(1234)

# Split the data
credit_data <- sample.split(credit3$Class, SplitRatio = 0.8)
train_set <- subset(credit3, credit_data == TRUE)
test_set <- subset(credit3, credit_data == FALSE)

# Dimensions
dim(train_set)
```

```
## [1] 227846     30
```

```
dim(test_set)

## [1] 56961    30
```

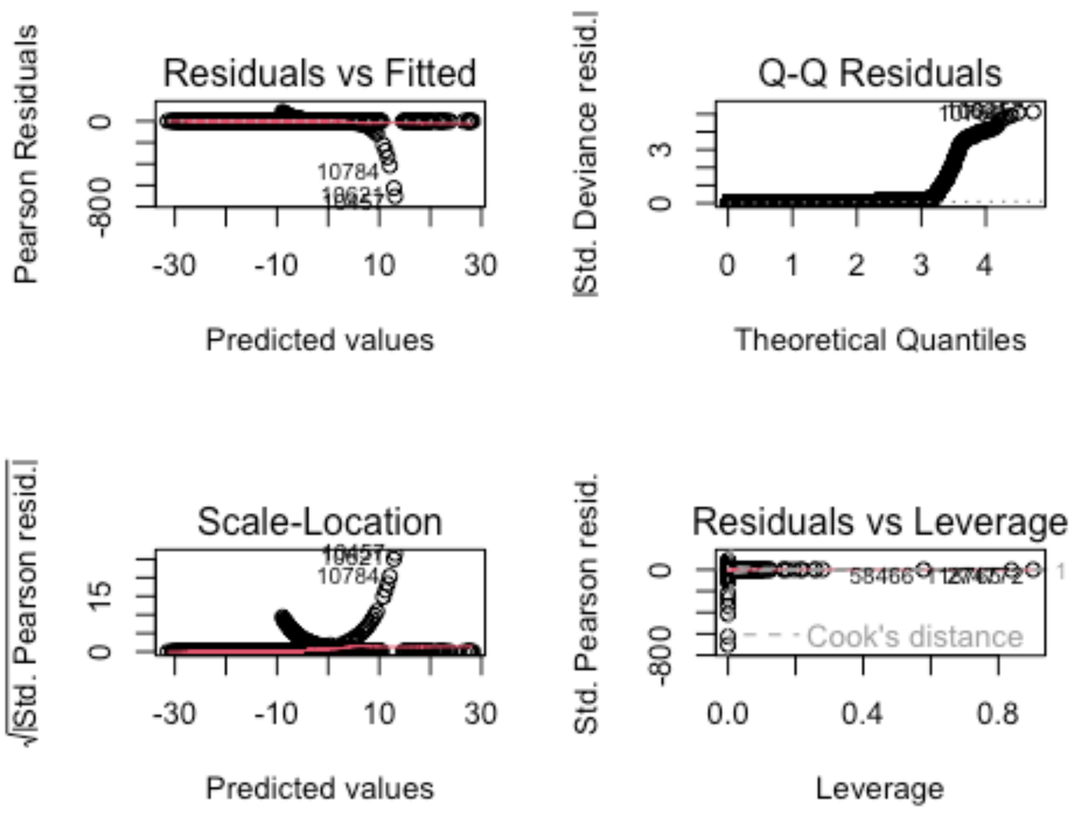
B) Fit Logistic Regression Model

The next step involves the fitting of a logistic model using the test data. This is our initial model building method because traditional fraud analytics involves the usage of a binary system to detect card fraud. A Receiver Operating Characteristic (ROC) Curve will be employed in order to investigate the Area Under the Curve (AUC).

```
# Logistic Regression using Train Data
credit_log_train <- glm(Class ~ ., data = train_set, family = 'binomial')
summary(credit_log_train)

##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = train_set)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.6551161  0.1615404 -53.579  < 2e-16 ***
## V1           0.1331322  0.0474949   2.803  0.00506 **
## V2          -0.0007304  0.0695566  -0.011  0.99162
## V3           0.0641292  0.0532110   1.205  0.22813
## V4           0.7380053  0.0815459   9.050  < 2e-16 ***
## V5           0.0723711  0.0785744   0.921  0.35702
## V6          -0.1695213  0.0945237  -1.793  0.07290 .
## V7          -0.1191590  0.0789220  -1.510  0.13109
## V8          -0.1900745  0.0389307  -4.882 1.05e-06 ***
## V9          -0.3447030  0.1199843  -2.873  0.00407 **
## V10         -0.8640536  0.1088894  -7.935 2.10e-15 ***
## V11         -0.0466138  0.0850883  -0.548  0.58381
## V12          0.0766177  0.0965561   0.794  0.42748
## V13         -0.2854009  0.0903333  -3.159  0.00158 **
## V14         -0.5437923  0.0691614  -7.863 3.76e-15 ***
## V15         -0.1116702  0.0956097  -1.168  0.24282
## V16         -0.2059172  0.1340339  -1.536  0.12446
## V17         -0.0470808  0.0787853  -0.598  0.55012
## V18         -0.0228321  0.1396322  -0.164  0.87011
## V19          0.0789513  0.1066455   0.740  0.45911
## V20         -0.4998013  0.0971133  -5.147 2.65e-07 ***
## V21          0.3121892  0.0690020   4.524 6.06e-06 ***
## V22          0.4085491  0.1421835   2.873  0.00406 **
## V23         -0.0866283  0.0738566  -1.173  0.24083
## V24          0.1772063  0.1764181   1.004  0.31515
## V25          0.0231801  0.1464762   0.158  0.87426
## V26         -0.1247253  0.2167122  -0.576  0.56493
## V27         -0.8912882  0.1402574  -6.355 2.09e-10 ***
## V28         -0.2634860  0.1105700  -2.383  0.01717 *
## Amount       0.2438620  0.1257413   1.939  0.05245 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5799.1  on 227845  degrees of freedom
## Residual deviance: 1778.7  on 227816  degrees of freedom
## AIC: 1838.7
##
## Number of Fisher Scoring iterations: 12
```

Viewing the plots of this logistic model, we can see the following:

```r
## Plots of the Train Data
par(mfrow=c(2,2))
plot(credit_log_train)
```



We can see from the above that both the QQ and the Residuals Plots have the majority of points following a linear trend (although there are values that deviate from normality). Given there are no substantial outliers (from the context of this exercise), we will move into the Test Set in order to transition into the analysis of the Area Under the Curve (AUC) using the Receiver Operating Characteristic (ROC).

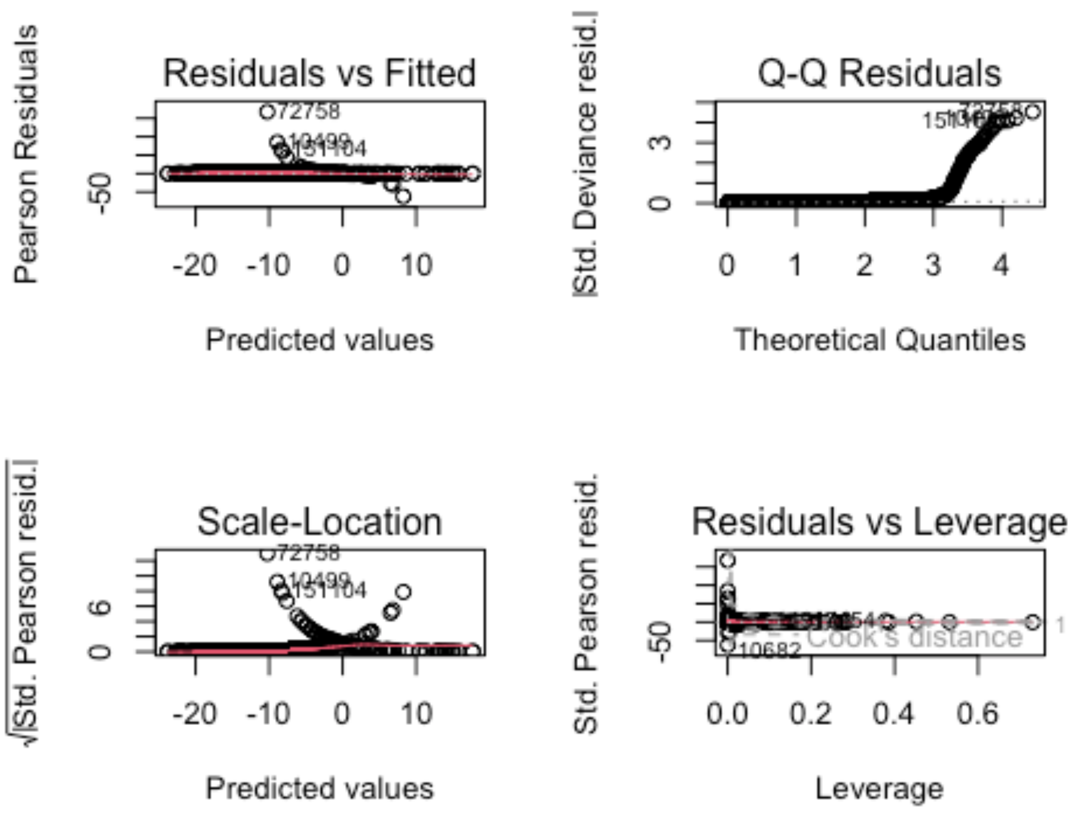
```
# Logistic Regression using Train Data
credit_log_test <- glm(Class ~ ., data = test_set, family = 'binomial')
summary(credit_log_test)

##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = test_set)
##
## Coefficients:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.836694   3.067548  -3.207  0.00134 **
## V1          -0.193396   0.305763  -0.633  0.52706
## V2           0.396543   1.314889   0.302  0.76297
## V3          -0.046397   0.124569  -0.372  0.70955
## V4           0.983296   1.790120   0.549  0.58281
## V5           0.500318   1.308540   0.382  0.70220
## V6          -0.051331   0.198162  -0.259  0.79561
## V7           0.206221   1.089224   0.189  0.84984
## V8          -0.120117   0.075839  -1.584  0.11323
## V9           0.687675   2.784236   0.247  0.80492
## V10         -1.138700   2.136130  -0.533  0.59399
## V11          0.140328   0.629735   0.223  0.82366
## V12          0.248144   0.805345   0.308  0.75799
## V13         -0.636200   0.806605  -0.789  0.43027
## V14         -0.596586   0.432623  -1.379  0.16790
## V15          0.148489   0.444498   0.334  0.73833
## V16          0.827692   3.970408   0.208  0.83487
## V17         -0.080828   0.690485  -0.117  0.90681
## V18         -0.865786   3.546247  -0.244  0.80712
## V19          0.553622   2.017946   0.274  0.78382
## V20         -0.044449   1.072053  -0.041  0.96693
## V21          0.758026   0.956660   0.792  0.42815
## V22          1.659655   1.542085   1.076  0.28182
## V23         -0.153897   0.251541  -0.612  0.54066
## V24         -0.023284   0.319200  -0.073  0.94185
## V25         -0.088803   0.418194  -0.212  0.83184
## V26          0.421959   0.402707   1.048  0.29473
## V27         -0.545951   0.253734  -2.152  0.03142 *
## V28         -0.470157   0.182660  -2.574  0.01005 *
## Amount       0.006524   0.415834   0.016  0.98748
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1443.40  on 56960  degrees of freedom
## Residual deviance:  415.85  on 56931  degrees of freedom
## AIC: 475.85
##
## Number of Fisher Scoring iterations: 17
```

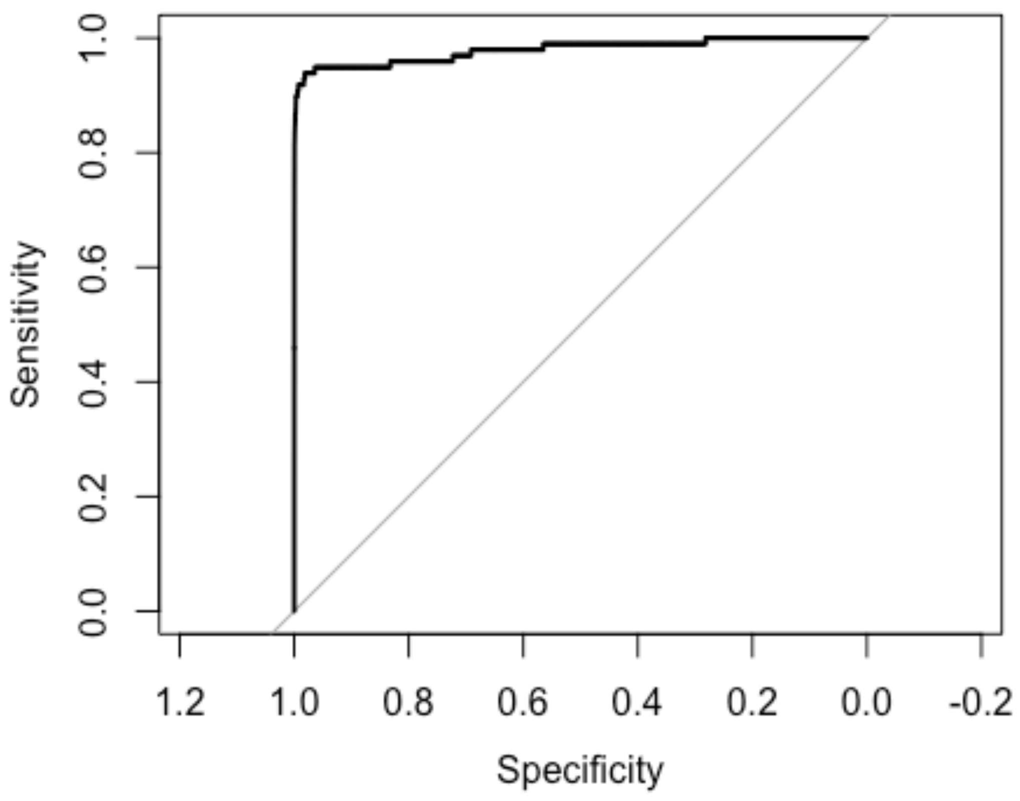
Having now established the model of the test data, we can proceed with visualising the residuals and the normality of this data. Upon doing this step, we can proceed with creating the Receiver Operating Characteristic (ROC) curve. Note that we want this reading to be high, but in order to be able to compare against other algorithms and investigate the model suitability.

```
## Plots of the Test Data
par(mfrow=c(2,2))
plot(credit_log_test)
```



Similar to the train data, the test data shows near similar results. It can be seen that the majority of points follow a linear trend in the tests of normality, but given we are assessing the model against the binary response variable `Class`, we will proceed with visualising with an Area Under the Curve plot.

```
credit_log_prediction <- predict(credit_log_test, newdata = test_set,
probability = TRUE)
roc(test_set$Class, credit_log_prediction, plot = TRUE)

## Setting levels: control = Non Fraudulent, case = Fraudulent

## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = test_set$Class, predictor = credit_log_prediction,
plot = TRUE)
##
## Data: credit_log_prediction in 56863 controls (test_set$Class Non
Fraudulent) < 98 cases (test_set$Class Fraudulent).
## Area under the curve: 0.9794
```