10.1 Final Project Step 3: Final Project Submission

Kalkar, Manish

August 6, 2020

Overall, write a coherent narrative that tells a story with the data as you complete this section.

Introduction

The data set is retrieved from Kaggle - https://www.kaggle.com/mlg-ulb/creditcardfraud

The data set contains transactions made by credit cards in September 2013 by European cardholders. According to creditcards.com, in UK alone, there was over £300m in fraudulent credit card transactions in the first half of 2016, with banks preventing over £470m of fraud in the same period. The data shows that credit card fraud is rising, so there was an urgent need to develop new, and improve current, fraud detection methods.

The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group of ULB (University of Libre de Bruxelles) on big data mining and fraud detection. To preserve anonymity, these data have been transformed using principal components analysis.

Data Description

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. It contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, there are not provided the original features and more background information about the data.

- Variables V1 to V28 are the principal components obtained with PCA.
- The only variables which have not been transformed with PCA are Time and Amount. Variable Time contains the time in seconds elapsed between each transaction and the first transaction in the dataset.
- The variable Amount is the transaction Amount.
- Variable Class is the response variable and it has the values as below:
- 1 - Fraudulent Transaction
- 0 - Non-fraudulent Transaction

Summarize the problem statement you addressed.

Credit card fraud is a major concern in the financial industry nowadays. With the growth of e-commerce websites, people and financial companies rely on online services to carry out their transactions that have led to an exponential increase in the credit card frauds. The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. The design of an effective fraud detection system is necessary to reduce the losses incurred by the customers and financial companies.

Summarize how you addressed this problem statement (the data used and the methodology employed).

Following methodologies were employed to address the problem statement:

Machine Learning Technique

Using this dataset, we used machine learning technique to develop models that attempt to predict whether a transaction is fraudulent. Following models have been developed and applied to the data set:

- Logistic Regression
- ROC Curve for Logistic Regression
- K Nearest Neighbor Algorithm

We observed the accuracy of above models to see which one fits the best to predict the fraudulent transaction.

Anomaly Detection Technique

Detecting anomalies is nothing but to identify irregularities in data is to flag the data points that deviate from common statistical properties of a distribution. We relied on visual representation by plotting below:

- Residuals vs Predicted Values
- Normal Q-Q Plot
- Standardized Residuals vs Predicted Value
- Residuals vs Leverage

Summarize the interesting insights that your analysis provided.

Analyze the need for Data Cleaning

Sensitive data

The data set consists of numerical values from the 28 'Principal Component Analysis (PCA)' transformed features. Due to confidentiality issues, the original features and more background information about the data could not be provided.

Consistency in Variables Names

```
## Display Variable Names

names(creditcard_df)
```

```
##  [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"
##  [9] "V8"     "V9"     "V10"    "V11"    "V12"    "V13"    "V14"    "V15"
## [17] "V16"    "V17"    "V18"    "V19"    "V20"    "V21"    "V22"    "V23"
## [25] "V24"    "V25"    "V26"    "V27"    "V28"    "Amount" "Class"
```

All variable names are precise and as expected.

Missing / Null observations in the data set

```
## Check missing values in each variable

colMeans(is.na(creditcard_df))
```

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

There are no missing values in the data set.

Data Cleaning

The data set is already fairly clean. No need to do any further cleaning.

Data Exploration / Data Analysis

Display Data set

```
## Display first 5 rows of the data set
head(creditcard_df,5)
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
##            V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
##          V13        V14        V15        V16        V17         V18
V19
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.2079712  0.02579058
0.4039930
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.1148047 -0.18336127 -
0.1457830
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.1099694 -0.12135931 -
2.2618571
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.6840928  1.96577500 -
1.2326220
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.2370332 -0.03819479
0.8034869
##           V20          V21          V22        V23         V24        V25
## 1  0.25141210 -0.018306778  0.277837576 -0.1104739  0.06692807  0.1285394
## 2 -0.06908314 -0.225775248 -0.638671953  0.1012880 -0.33984648  0.1671704
## 3  0.52497973  0.247998153  0.771679402  0.9094123 -0.68928096 -0.3276418
## 4 -0.20803778 -0.108300452  0.005273597 -0.1903205 -1.17557533  0.6473760
## 5  0.40854236 -0.009430697  0.798278495 -0.1374581  0.14126698 -0.2060096
```

```
##          V26          V27         V28 Amount Class
## 1 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4 -0.2219288  0.062722849  0.06145763 123.50     0
## 5  0.5022922  0.219422230  0.21515315  69.99     0
```

Fraudulent vs Non-fraudulent Transactions

```
table(creditcard_df$Class)
```

```
##
##      0      1
## 284315    492
```

There are only 492 fraudulent transactions vs 284315 non-fraudulent transactions.

% Fraudulent Transactions

```
TransactionCount <- as.numeric(table(creditcard_df$Class))
TransactionCount
```

```
## [1] 284315    492
```

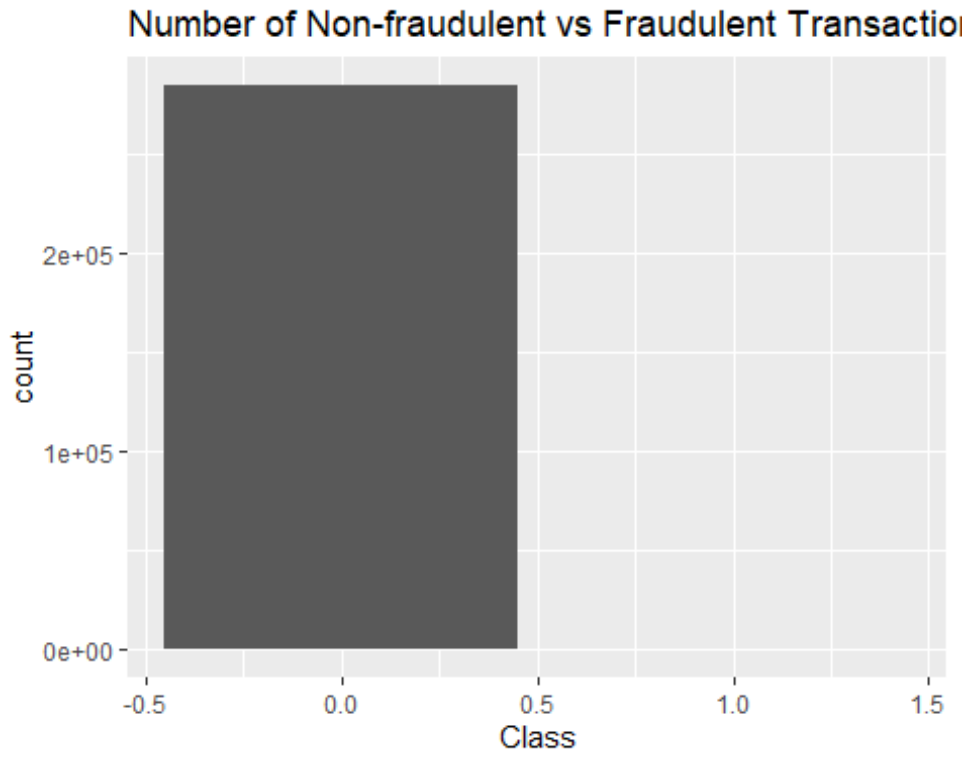
```
percentTransactions <- TransactionCount / sum(TransactionCount)
percentTransactions
```

```
## [1] 0.998272514 0.001727486
```

There are only 0.172% Fraudulent Transactions vs 99.82% Non-fraudulent Transactions. Only 0.172% Fraudulent Transaction shows that the data is highly Unbalanced.

Plot - Number of Non-fraudulent vs Fraudulent Transactions

```
ggplot(creditcard_df, aes(x = Class)) + geom_bar() + ggtitle("Number of Non-fraudulent vs Fraudulent Transactions")
```



Above plot shows that the data set is highly unbalanced.

Summary of the Data set

```
# Display summary of the data set
summary(creditcard_df)
```

```
##       Time              V1                   V2                   V3
##  Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256
##  1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904
##  Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799
##  Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000
##  3rd Qu.:139321   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272
##  Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826
##        V4                   V5                    V6                   V7
##  Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605   Min.   :-
43.5572
##  1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -
0.5541
##  Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :
0.0401
##  Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :
0.0000
```

```
##  3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:
0.5704
##  Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.
:120.5895
##        V8                 V9                 V10                 V11
##  Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-
4.79747
##  1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-
0.76249
##  Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-
0.03276
##  Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   :
0.00000
##  3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.:
0.73959
##  Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.
:12.01891
##       V12                V13                V14                V15
##  Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894
##  1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288
##  Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807
##  Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000
##  3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882
##  Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774
##       V16                V17                V18
##  Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746
##  1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850
##  Median :  0.06641   Median : -0.06568   Median :-0.003636
##  Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000
##  3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807
##  Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069
##       V19                V20                V21
##  Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038
##  1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839
##  Median : 0.003735   Median : -0.06248   Median : -0.02945
##  Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000
##  3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638
##  Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284
##       V22                V23                V24
##  Min.   :-10.933144   Min.   :-44.80774   Min.   :-2.83663
##  1st Qu.: -0.542350   1st Qu.: -0.16185   1st Qu.:-0.35459
##  Median :  0.006782   Median : -0.01119   Median : 0.04098
##  Mean   :  0.000000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.:  0.528554   3rd Qu.:  0.14764   3rd Qu.: 0.43953
##  Max.   : 10.503090   Max.   : 22.52841   Max.   : 4.58455
##       V25                V26                V27
##  Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679
##  1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840
##  Median :  0.01659   Median :-0.05214   Median :  0.001342
##  Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000
```

```
##  3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045
##  Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198
##       V28                 Amount             Class
##  Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000
##  1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000
##  Median :  0.01124   Median :   22.00   Median :0.000000
##  Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728
##  3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000
##  Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000
```

Summary of the data set shows all the PCA transformed variables V1 to V28 have already been normalized with mean 0.

Plot - Distribution of Transaction Amount by Class

```
ggplot(creditcard_df, aes(x = Class, y = Amount)) + xlim("-0.5", "1.5") +
geom_boxplot() + ggtitle("Distribution of Transaction Amount by Class")
```

Distribution of Transaction Amount by Class above shows lot more variability in the transaction values for non-fraudulent transactions than the fraudulent ones. In addition, Amount variable is important in predicting whether a transaction was fraudulent.

Determine mean for fraudulent transactions vs Non-fraudulent transactions

```
aggregate(creditcard_df$Amount, by=list(creditcard_df$Class), FUN=mean)
```

```
##   Group.1         x
## 1       0  88.29102
## 2       1 122.21132
```

Above matrix shows the mean for fraudulent transactions is more and that the problem is critical and need to be resolved.

Data Split

The data set has been divided into two main groups. One for training the model and the other for Testing our trained model's performance. The data set has been split with the split ratio of 0.80. This means that 80% of the data was attributed to the train data whereas 20% was attributed to the test data.

The train and test data sets were then utilized in building the data models.

```
## Data Split

set.seed(100)

# Split data set into train and test data set and separate out predictor from
cl
random_df <- sample(1:nrow(creditcard_df), 0.8 * nrow(creditcard_df))

# Train Data set
train_df <- creditcard_df[random_df,]

# Test Data set
test_df <- creditcard_df[-random_df,]

# Original Data set
table(creditcard_df$Class)
```

```
##
##      0      1
## 284315    492
```

```
## Train Data set - 80%
table(train_df$Class)
```

```
##
##      0      1
## 227447    398
```

```
## Test Data set - 20%
table(test_df$Class)
```

```
##
##     0     1
## 56868    94
```

- 0 represents Non-fraudulent Transactions
- 1 represents Fraudulent Transactions

Data Model

Logistic Regression

Fit the logistic regression model to the data set that predicts the fraudulent transaction

Logistic regression is used for modeling the outcome probability of a class – fraud vs not fraud.

```
creditcard_glm <- glm(Class ~ Time + V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +
V9 + V10 +
                      V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20
+ V21 +
                      V22 + V23 + V24 + V25 + V26 + V27 + V28 + Amount, data =
train_df, family = "binomial")

## Include a summary using the summary() function in your results
summary(creditcard_glm)
```

```
##
## Call:
## glm(formula = Class ~ Time + V1 + V2 + V3 + V4 + V5 + V6 + V7 +
##     V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 +
##     V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 +
##     V28 + Amount, family = "binomial", data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.0426  -0.0293  -0.0182  -0.0110   4.5790
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.431e+00  2.911e-01 -28.957  < 2e-16 ***
## Time        -4.609e-06  2.562e-06  -1.799 0.072002 .
## V1           6.441e-02  4.855e-02   1.327 0.184627
## V2           5.394e-02  7.004e-02   0.770 0.441204
## V3          -1.226e-02  6.058e-02  -0.202 0.839687
## V4           7.340e-01  9.320e-02   7.875 3.40e-15 ***
## V5           1.642e-01  8.104e-02   2.026 0.042787 *
## V6          -1.479e-01  8.692e-02  -1.702 0.088843 .
## V7          -2.853e-02  7.895e-02  -0.361 0.717853
## V8          -1.968e-01  3.609e-02  -5.452 4.99e-08 ***
## V9          -2.248e-01  1.429e-01  -1.573 0.115746
## V10         -8.268e-01  1.228e-01  -6.733 1.66e-11 ***
## V11         -5.599e-02  9.330e-02  -0.600 0.548408
## V12          9.404e-02  1.018e-01   0.924 0.355669
## V13         -4.451e-01  9.332e-02  -4.770 1.84e-06 ***
## V14         -5.658e-01  7.299e-02  -7.752 9.02e-15 ***
## V15         -4.080e-02  9.725e-02  -0.420 0.674837
## V16         -1.307e-01  1.668e-01  -0.783 0.433337
## V17         -6.450e-02  8.112e-02  -0.795 0.426518
## V18         -6.192e-02  1.651e-01  -0.375 0.707662
## V19          1.566e-01  1.187e-01   1.319 0.187107
## V20         -3.728e-01  1.027e-01  -3.630 0.000283 ***
## V21          4.199e-01  7.305e-02   5.749 8.98e-09 ***
## V22          7.572e-01  1.571e-01   4.820 1.44e-06 ***
## V23         -1.532e-01  6.867e-02  -2.231 0.025650 *
## V24          1.213e-01  1.704e-01   0.712 0.476670
## V25         -4.038e-02  1.459e-01  -0.277 0.781980
## V26         -2.138e-01  2.283e-01  -0.937 0.348986
## V27         -6.906e-01  1.566e-01  -4.411 1.03e-05 ***
## V28         -2.165e-01  1.003e-01  -2.159 0.030815 *
## Amount       8.233e-04  4.001e-04   2.058 0.039635 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5849.9  on 227844  degrees of freedom
## Residual deviance: 1742.3  on 227814  degrees of freedom
## AIC: 1804.3
##
## Number of Fisher Scoring iterations: 12

## Calculate Predicted Probability
creditcard_predicted_probability = predict(creditcard_glm, test_df, type =
"response")

## Calculate and build the Confusion Matrix
confusion_matrix <- table(Actual_Value = test_df$Class, Predicted_Value =
```

```
creditcard_predicted_probability > 0.5)

## Calculate the Accuracy of the model
accuracy = (confusion_matrix[[1,1]] +
confusion_matrix[[2,2]])/sum(confusion_matrix) * 100
accuracy

## [1] 99.92276
```

The accuracy of fitting the Logistic Regression Model is 99.92%.

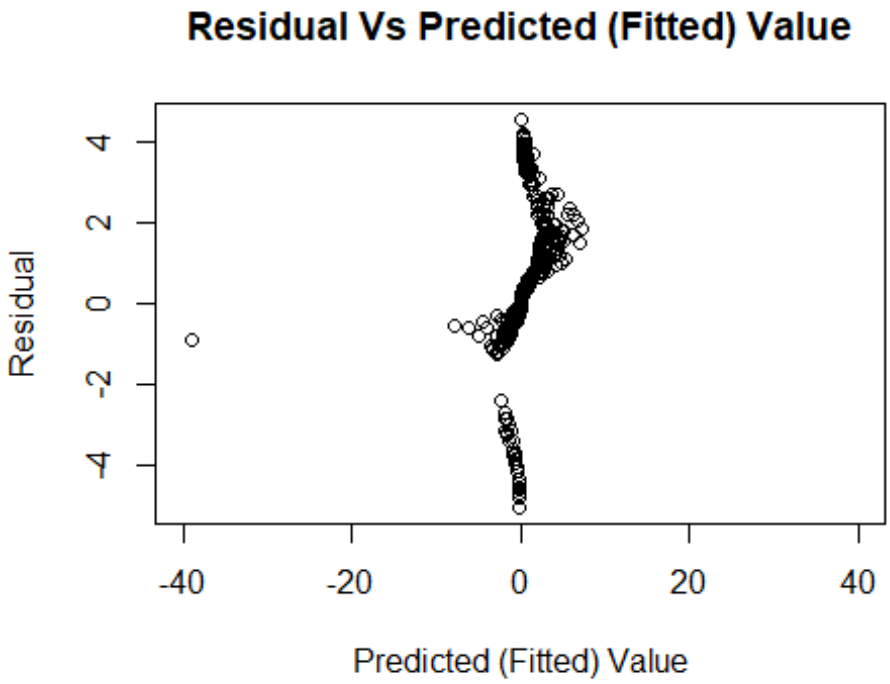
Visual representation - Plotting the Logistic Regression

Compute the Casewise diagnostics to identify outliers and/or influential cases

```
## casewise diagnostics to identify outliers and/or influential cases
creditcard_DataFrame <- as.data.frame(resid(creditcard_glm))
creditcard_DataFrame$residuals <- resid(creditcard_glm)
creditcard_DataFrame$standardized.residuals <- rstandard(creditcard_glm)
creditcard_DataFrame$studentized.residuals <- rstudent(creditcard_glm)
creditcard_DataFrame$cooks.distance <- cooks.distance(creditcard_glm)
creditcard_DataFrame$dfbeta <- dfbeta(creditcard_glm)
creditcard_DataFrame$dffit <- dffits(creditcard_glm)
creditcard_DataFrame$leverage <- hatvalues(creditcard_glm)
creditcard_DataFrame$covariance.ratios <- covratio(creditcard_glm)
```

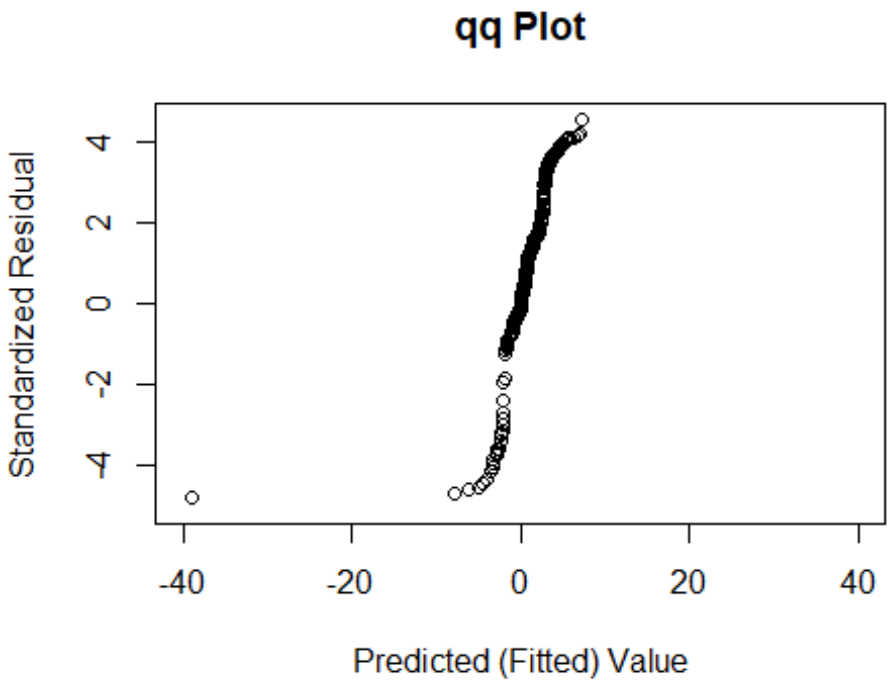
Plot - Residual Vs Predicted (Fitted) Value

```
plot(dffits(creditcard_glm), resid(creditcard_glm), xlim=c(-40, 40), xlab =
"Predicted (Fitted) Value", ylab = "Residual", main="Residual Vs Predicted
(Fitted) Value")
```



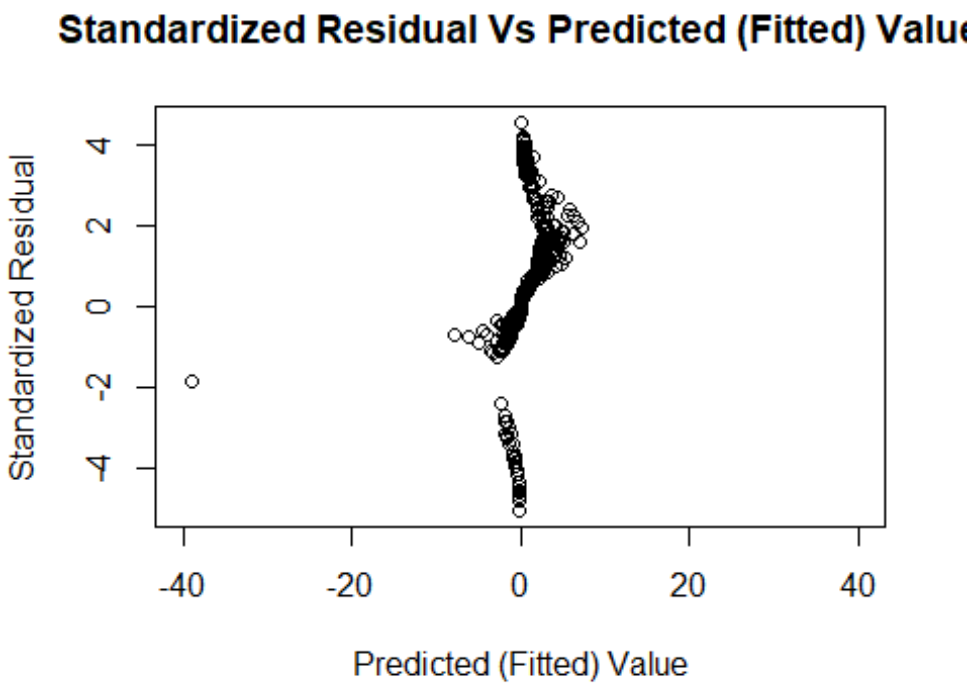
Plot - qq Plot

```
qqplot(dffits(creditcard_glm), rstandard(creditcard_glm), xlim=c(-40, 40),
xlab = "Predicted (Fitted) Value", ylab = "Standardized Residual", main="qq
Plot")
```



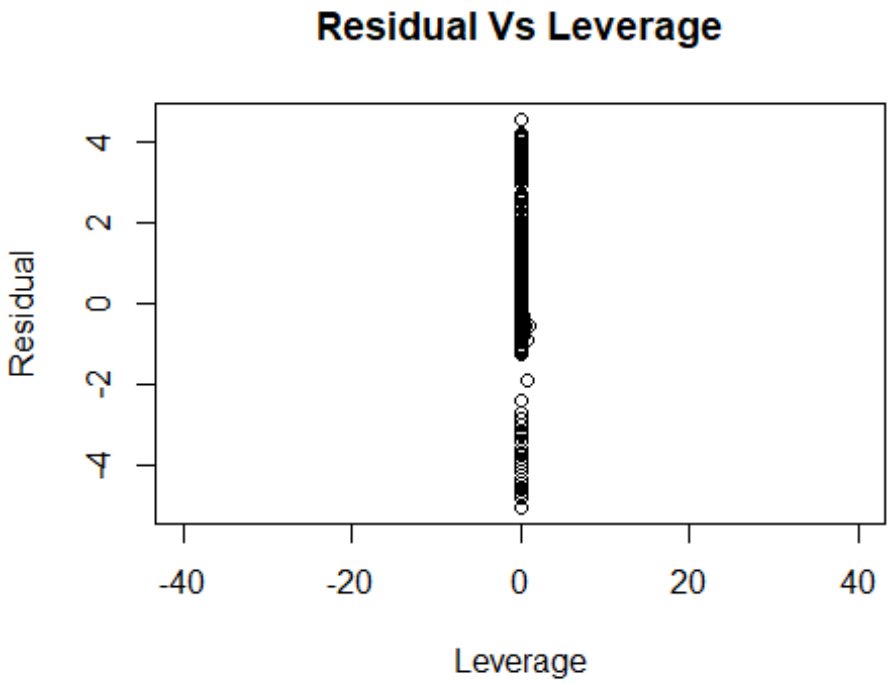
Plot - Standardized Residual Vs Predicted (Fitted) Value

```
plot(dffits(creditcard_glm), rstandard(creditcard_glm), xlim=c(-40, 40), xlab
= "Predicted (Fitted) Value", ylab = "Standardized Residual",
main="Standardized Residual Vs Predicted (Fitted) Value")
```



Plot - Residual Vs Leverage

```
plot(hatvalues(creditcard_glm), resid(creditcard_glm), xlim=c(-40, 40), xlab
= "Leverage", ylab = "Residual", main="Residual Vs Leverage")
```



We can clearly notice the outliers / anomalies representing the Fraudulent Transaction.

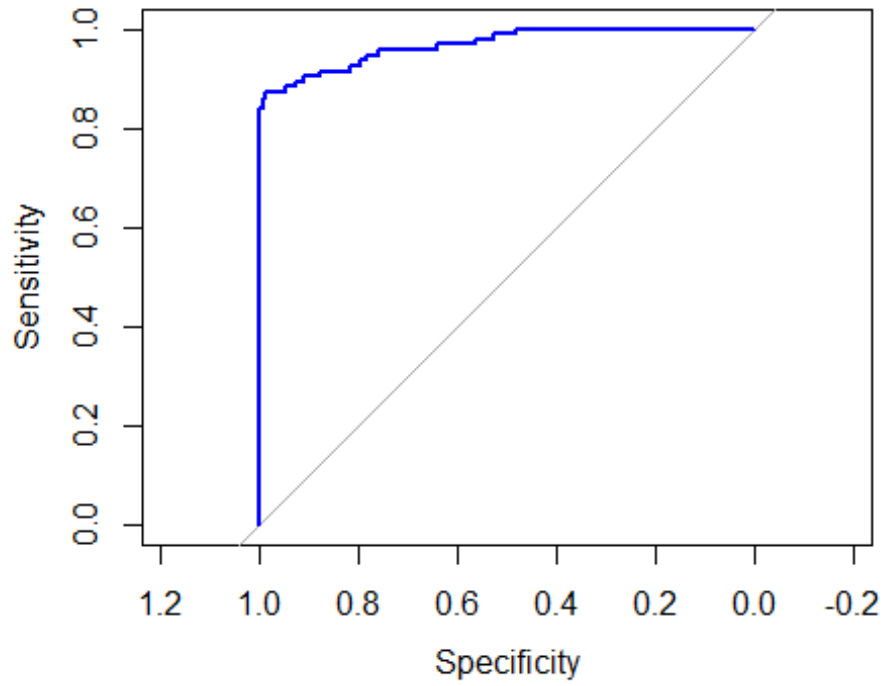
Plot ROC Curve

Next, we will plot our ROC curve to further assess the performance of our model.

```
## Plot ROC Curve
creditcard_predict_ROC <- predict(creditcard_glm, test_df, probability =
TRUE)
creditcard_auc_ROC = roc(test_df$Class, creditcard_predict_ROC, plot = TRUE,
col = "blue")

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```




```
creditcard_auc_ROC

##
## Call:
## roc.default(response = test_df$Class, predictor = creditcard_predict_ROC,
plot = TRUE, col = "blue")
##
## Data: creditcard_predict_ROC in 56868 controls (test_df$Class 0) < 94
cases (test_df$Class 1).
## Area under the curve: 0.9679
```

The accuracy of the model by plotting the ROC is 96.79%. The accuracy seem much more desirable, especially when the data set is highly unbalanced.

K Nearest Neighbor Algorithm

```
## Target data set
target_df<- creditcard_df[random_df,31]
target_test_df <- creditcard_df[-random_df,31]
```

Build nearest neighbors model

```
# Let's build the nearest neighbor model based on k value = 1
prediction_model_k1 <- knn(train_df, test_df, cl= target_df, k = 1)

# confusion Matrix for k=1
knn_confusion_matrix_k1 <- table(prediction_model_k1, target_test_df)

## Calculate the Accuracy of the model for k=1
accuracy_k1 = (knn_confusion_matrix_k1[[1,1]])/sum(knn_confusion_matrix_k1) *
100
accuracy_k1
```

```
## [1] 99.78758
```

The accuracy of fitting the K Nearest Neighbor Algorithm is 99.78%.

Conclusion

We used various Machine Learning Techniques as well as Anomaly Detection Techniques through visual representation by plotting graphs. Based on all the models applied to the data set, below is the accuracy of each model:

- Logistic Regression - 99.92%
- ROC Curve applied to Logistic Regression - 96.79%
- K Nearest Neighbor (KNN) Algorithm - 99.78%

It has been observed that shown Logistic Regression model provide good results. K Nearest Neighbors algorithm provides higher accuracy as well. However, KNN is relatively sluggish in obtaining the desired results. Applying ROC CUrve to the Logistic Regression model provided the most desired result, especially looking at highly unbalanced data set.

Summarize the implications to the consumer (target audience) of your analysis.

Based on this analysis, various data models have been developed. Ultimately by utilizing the model that best fit the data set, credit card companies should be able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. The goal here was to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

Discuss the limitations of your analysis and how you, or someone else, could improve or build on it.

Somebody could have developed additional types of models such as Decision Tree and / or Random Forest Models by analyzing the data further in order to achieve more realistic accuracy of the model for such highly unbalanced data set.

References

- Data set retrieved from Kaggle - https://www.kaggle.com/mlg-ulb/creditcardfraud
- Anomaly Detection - Credit Card Fraud Analysis By Pavan Sanagapati. https://www.kaggle.com/pavansanagapati/anomaly-detection-credit-card-fraud-analysis
- Predicting fraudulent credit card transactions. By Chris Bow. https://www.kaggle.com/chrisbow/credit-card-fraud-prediction-in-r
- Credit Card Fraud Detection Predictive Models. By Gabriel Preda. https://www.kaggle.com/gpreda/credit-card-fraud-detection-predictive-models
- Discovering Statistics Using R. Sage Publications, 2012.