CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

CREDIT CARD FRAUD DETECTION

Programming Language: R

Steps are given as follows.

1. Import the dataset
2. Exploration of data
3. Data manipulation
4. Data modeling
5. Application of machine learning techniques

Machine learning techniques used:

- **Logistic Regression**
- **Decision tree**
- **Artificial Neural Networks**
- **Gradient Boosting**

1. Import the dataset

The data set (creditcard.csv) import process was done with the R codes given below.

```
library(ranger)
library(caret)
library(data.table)
creditcard_data <- read.csv("creditcard.csv")
```

2. Exploration of data

In this section, the data within the scope of the data set were explored. For this purpose, head() and tail() functions are used. After that, other components of the data set were examined.

```
dim(creditcard_data)
head(creditcard_data,6)
```

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

```
##   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
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            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
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##           V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
```

Screenshot for head() function

```
##          Time          V1          V2         V3         V4          V5
## 284802 172785   0.1203164  0.93100513 -0.5460121 -0.7450968  1.13031398
## 284803 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278
## 284804 172787  -0.7327887 -0.05508049  2.0350297 -0.7385886  0.86822940
## 284805 172788   1.9195650 -0.30125385 -3.2496398 -0.5578281  2.63051512
## 284806 172788  -0.2404400  0.53048251  0.7025102  0.6897992 -0.37796113
## 284807 172792  -0.5334125 -0.18973334  0.7033374 -0.5062712 -0.01254568
##                V6         V7         V8         V9        V10        V11
## 284802 -0.2359732  0.8127221  0.1150929 -0.2040635 -0.6574221  0.6448373
## 284803 -2.6068373 -4.9182154  7.3053340  1.9144283  4.3561704 -1.5931053
## 284804  1.0584153  0.0243297  0.2948687  0.5848000 -0.9759261 -0.1501888
## 284805  3.0312601 -0.2968265  0.7084172  0.4324540 -0.4847818  0.4116137
## 284806  0.6237077 -0.6861800  0.6791455  0.3920867 -0.3991257 -1.9338488
## 284807 -0.6496167  1.5770063 -0.4146504  0.4861795 -0.9154266 -1.0404583
##                V12        V13         V14         V15        V16
## 284802  0.19091623 -0.5463289 -0.73170658 -0.80803553  0.5996281
## 284803  2.71194079 -0.6892556  4.62694203 -0.92445871  1.1076406
## 284804  0.91580191  1.2147558 -0.67514296  1.16493091 -0.7117573
## 284805  0.06311886 -0.1836987 -0.51060184  1.32928351  0.1407160
## 284806 -0.96288614 -1.0420817  0.44962444  1.96256312 -0.6085771
## 284807 -0.03151305 -0.1880929 -0.08431647  0.04133346 -0.3026201
```

Screenshot for tail() function

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

After this stage, the R codes applied on the data set and the relevant screenshots are included.

table(creditcard_data$Class)

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

summary(creditcard_data$Amount)

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     5.60    22.00    88.35    77.17 25691.16
```

names(creditcard_data)

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

var(creditcard_data$Amount)

```
## [1] 62560.07
```

sd(creditcard_data$Amount)

```
sd(creditcard_data$Amount)
```

```
## [1] 250.1201
```

3. Data Manipulation

In this section, the data is scaled using the scale() function. Scaling is also known as feature standardization. With the help of scaling, the data is structured according to a certain range. In this way, there will be no extreme values in our data set that may affect the operation of the models.

head(creditcard_data)

```
head(creditcard_data)
```

```
##   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
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            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
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767
```

This screenshot shows the state of the data before data manipulation. The state of the data after the data manipulation phase is shown below.

creditcard_data$Amount=scale(creditcard_data$Amount)

NewData=creditcard_data[,-c(1)]

head(NewData)

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

```
creditcard_data$Amount=scale(creditcard_data$Amount)
NewData=creditcard_data[,-c(1)]
head(NewData)
```

```
##           V1          V2        V3         V4          V5          V6
## 1 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            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
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767
```

4. Data modeling

In the previous section, the data set was standardized. In this section, we will divide the data set into a training set and a test set. Therefore, 80% of the data is allocated as training set (train_data) and 20% as test data. After that, the dimensions were found using the dim() function.

library(caTools)

set.seed(123)

data_sample = sample.split(NewData$Class,SplitRatio=0.80)

train_data = subset(NewData,data_sample==TRUE)

test_data = subset(NewData,data_sample==FALSE)

dim(train_data)

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

dim(test_data)

```
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
```

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

```
dim(test_data)
```

```
## [1] 56961    30
```

```
  Logistic_Model=glm(Class~.,test_data,family=binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
  summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(), data = test_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9019  -0.0254  -0.0156  -0.0078   4.0877
```

FINDINGS

In this part of the study, the following machine learning techniques were applied to the data set that was prepared for modeling in the previous part. Relevant codes and findings are included for each technique.

Machine learning techniques used:

- **Logistic Regression**
- **Decision tree**
- **Artificial Neural Networks**
- **Gradient Boosting**

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

- **Logistic Regression**

Logistic regression was applied to detect credit card fraud. Logistic regression is used to model the probability of outcome for a class such as pass/fail or positive/negative. In this case, it was used to detect credit card fraud, not fraud/fraud.

Logistic_Model=glm(Class~.,test_data,family=binomial())

summary(Logistic_Model)

```
Logistic_Model=glm(Class~.,test_data,family=binomial())
```

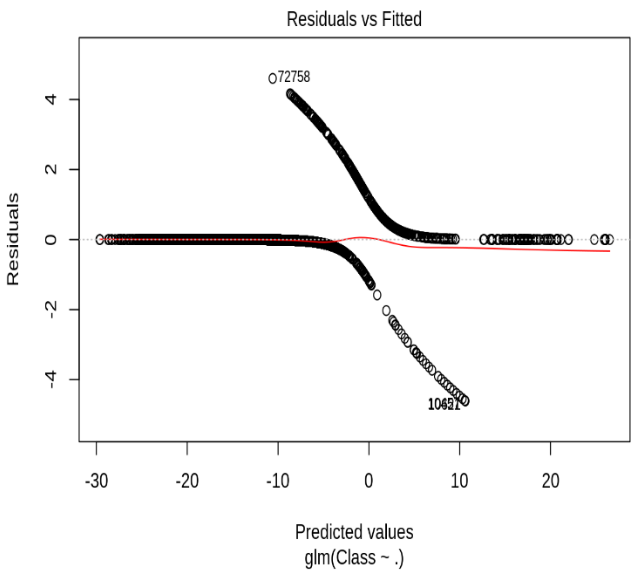
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(), data = test_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9019  -0.0254  -0.0156  -0.0078   4.0877
```

Summary of Logistic Regression

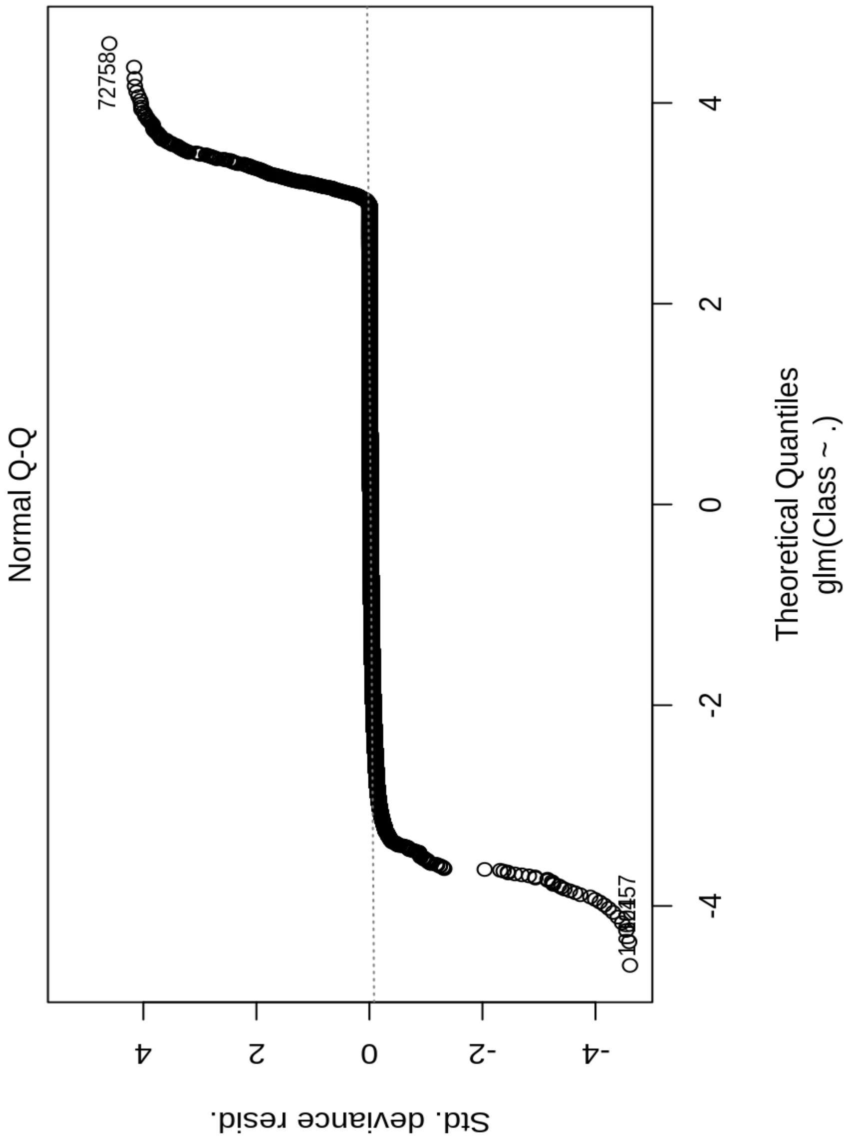
plot(Logistic_Model)



Model Visualization- Residuals

CREDIT CARD FRAUD DETECTION

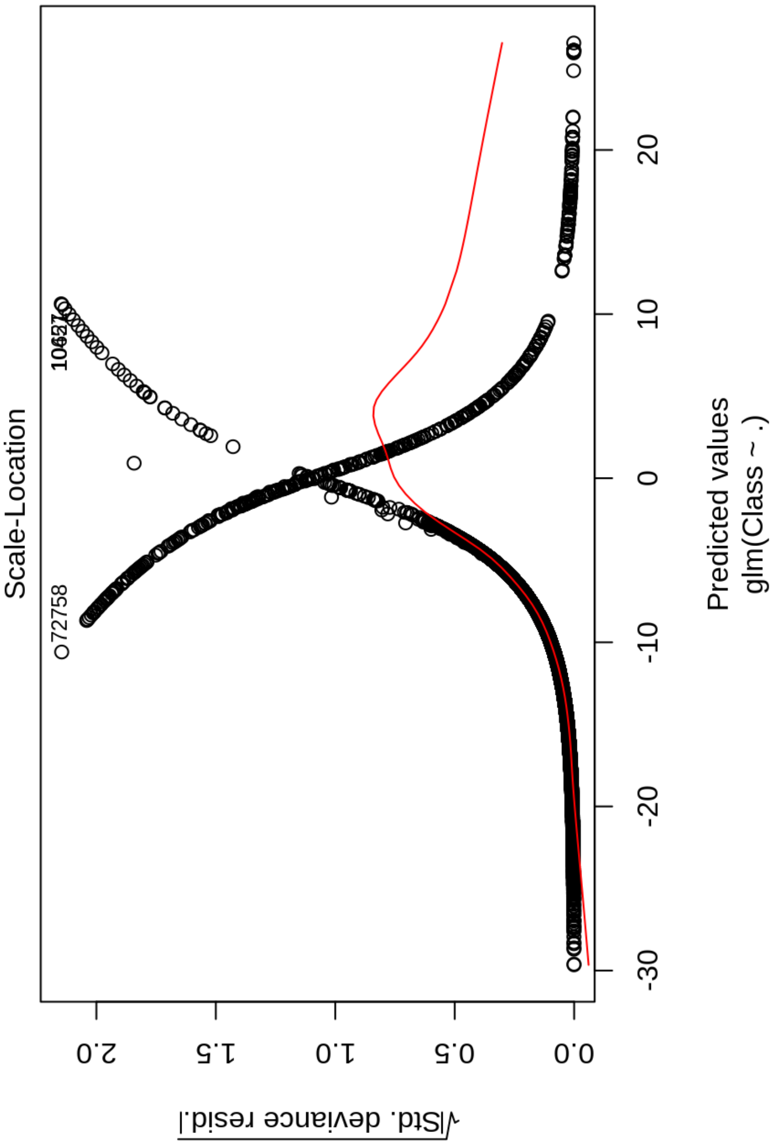
By Ozlem Kilickaya



Model Visualization- Standard Deviation Residuals

CREDIT CARD FRAUD DETECTION

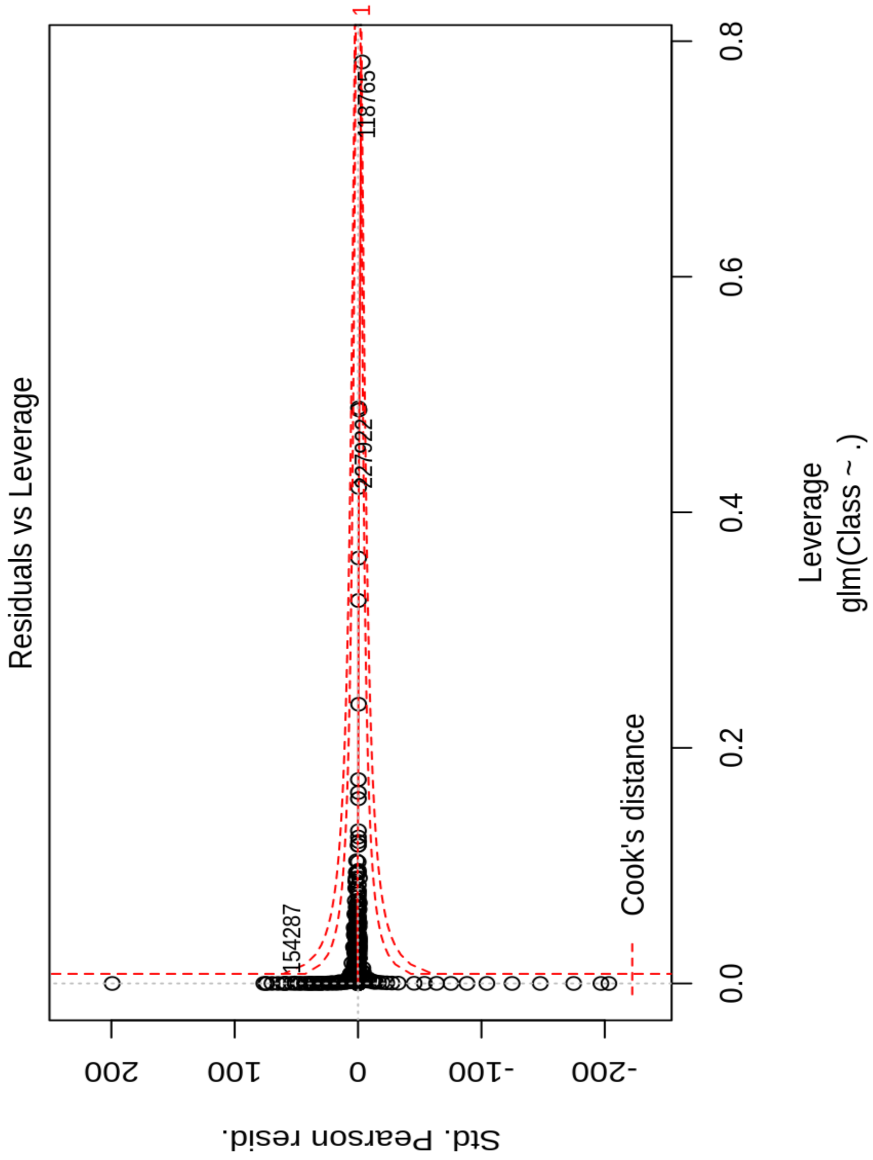
By Ozlem Kilickaya



Model Visualization-Predicted Values

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya



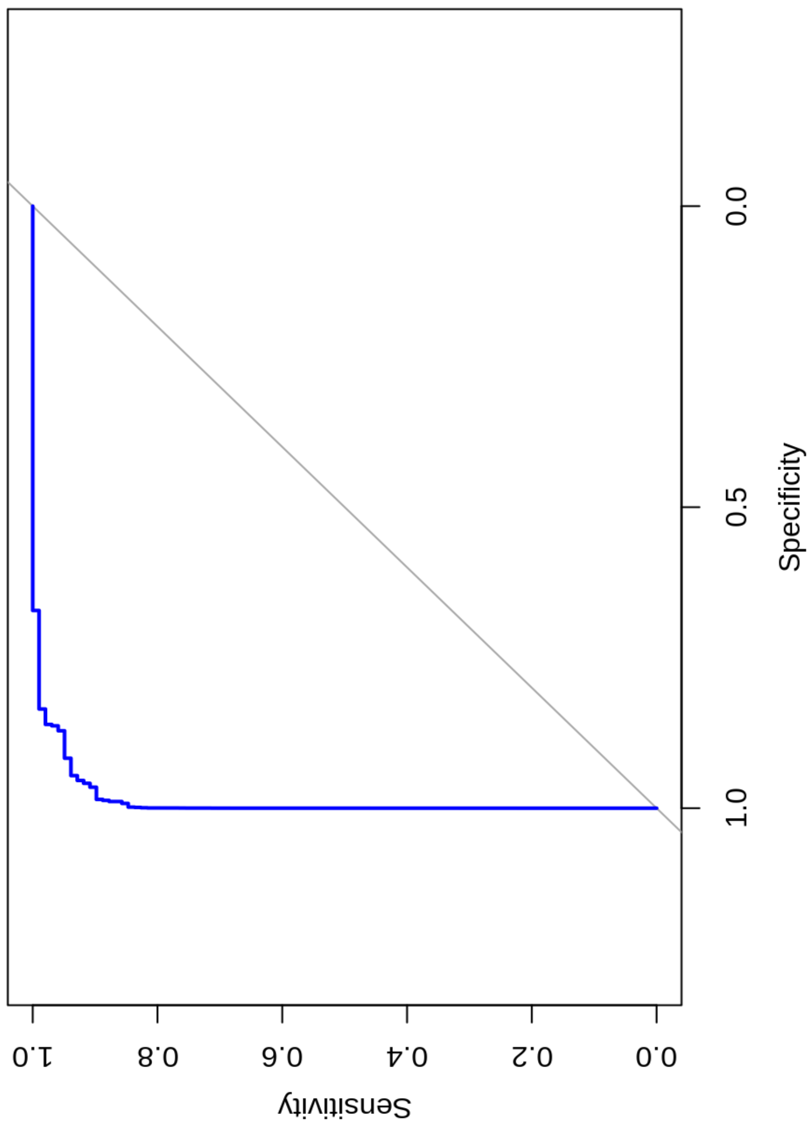
Model Visualization-Residuals vs Leverage

After visualizing the model, a ROC curve was drawn to evaluate the performance of the logistic regression model.

Library(pROC)

lr.predict <- predict(Logistic_Model,train_data, probability = TRUE)

auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = “blue”)



ROC curve for logistic regression

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

➢ **Decision Tree**

In this section, the results of the decision tree algorithm are included. A decision tree is used to plot the results of a decision. Recursive splitting is used to plot the decision tree.

library(rpart)

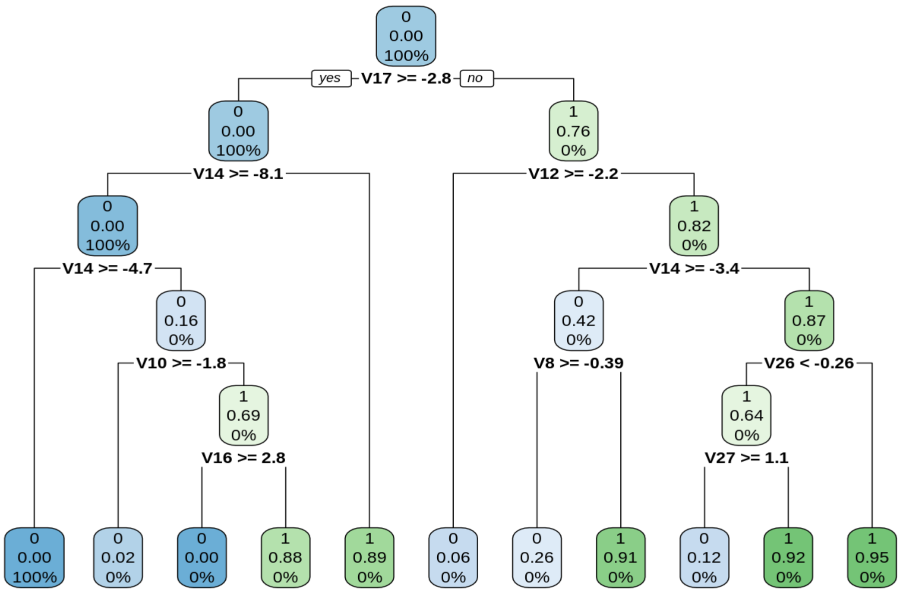
library(rpart.plot)

decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')

predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')

probability <- predict(decisionTree_model, creditcard_data, type = 'prob')

rpart.plot(decisionTree_model)



Decision Tree

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

- **Artificial Neural Networks**

Neural network models can learn certain patterns and classify on input models using historical data. The relevant package has been imported for the artificial neural networks application. Then the model is drawn using the plot() function. Neural networks have a value range from 1 to 0. Here, our threshold value is 0.5. So values above 0.5 will correspond to 1 and the remainder will be 0.

library(neuralnet)

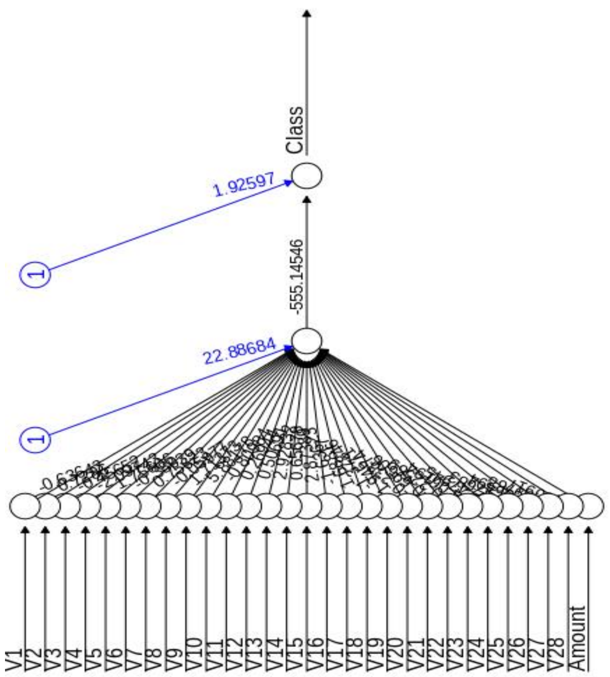
ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)

plot(ANN_model)

predANN=compute(ANN_model,test_data)

resultANN=predANN$net.result

resultANN=ifelse(resultANN>0.5,1,0)



Artificial Neural Networks

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

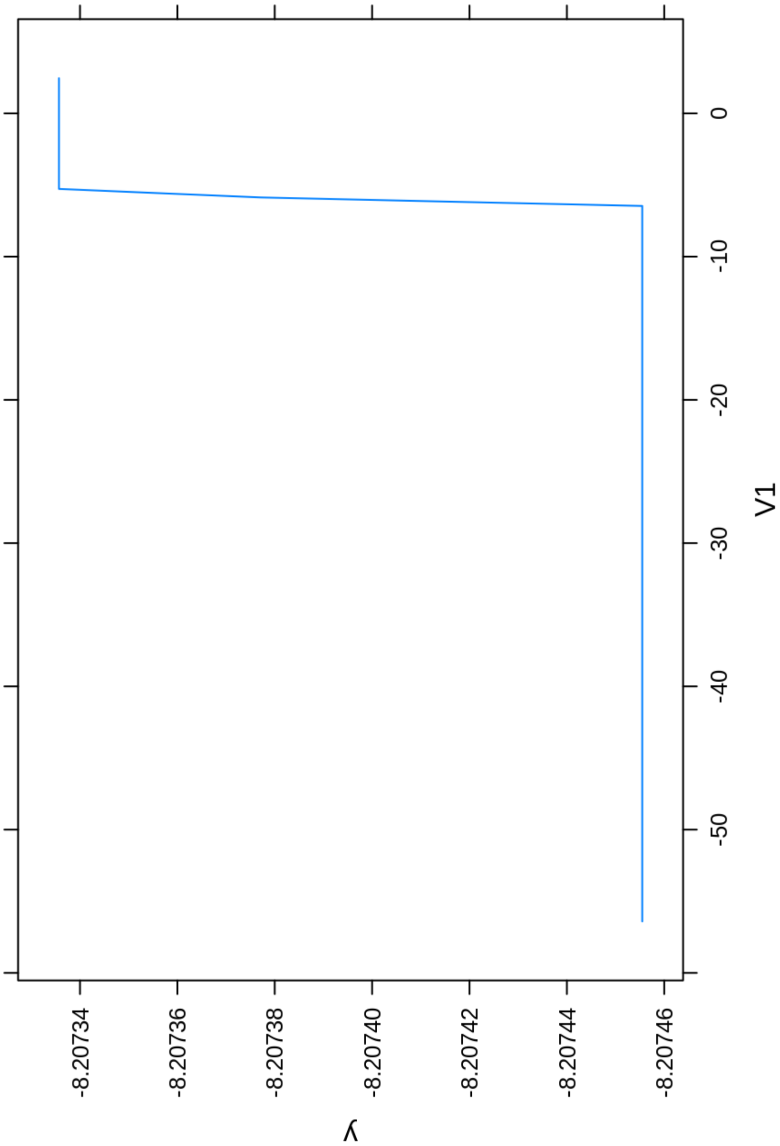
- **Gradient Boosting**

Gradient boosting is a machine learning algorithm used for classification and regression. This model consists of several basic ensemble models such as weak decision trees. These decision trees come together to form a gradient reinforcement model. The gradient boosting algorithm has been applied to our model as given below.

```
library(gbm, quietly=TRUE)

# Get the time to train the GBM model

system.time(

    model_gbm <- gbm(Class ~ .

        , distribution = "bernoulli"

        , data = rbind(train_data, test_data)

        , n.trees = 500

        , interaction.depth = 3

        , n.minobsinnode = 100

        , shrinkage = 0.01

        , bag.fraction = 0.5

        , train.fraction = nrow(train_data) / (nrow(train_data) + nrow(test_data))

)

)

# Determine best iteration based on test data

gbm.iter = gbm.perf(model_gbm, method = "test")

model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)

#Plot the gbm model

plot(model_gbm)
```

CREDIT CARD FRAUD DETECTION

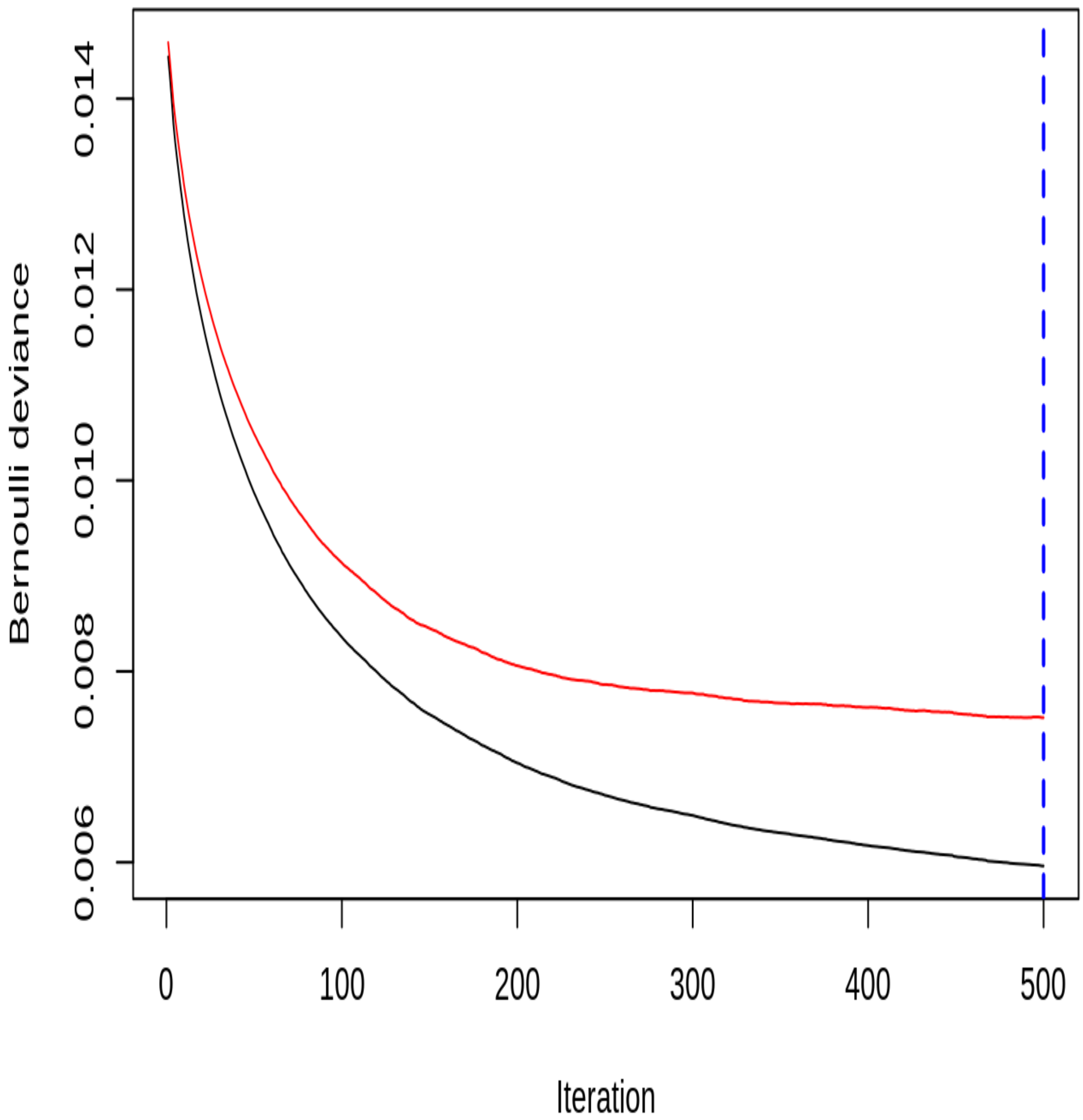
By Ozlem Kilickaya



Gradient Boosting

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya



Gradient Boosting- Bernoulli Deviance

CREDIT CARD FRAUD DETECTION

By Ozlem Kilickaya

Plot and calculate AUC on test data

gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)

gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")

```
# Plot and calculate AUC on test data
gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)
gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")
```

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

```
## Setting direction: controls < cases
```

print(gbm_auc)

```
print(gbm_auc)
```

```
##
## Call:
## roc.default(response = test_data$Class, predictor = gbm_test,     plot = TRUE, col = "red")
##
## Data: gbm_test in 56863 controls (test_data$Class 0) < 98 cases (test_data$Class 1).
## Area under the curve: 0.9555
```