Question-1 Practicum 3 DA5030

Shivam Verma

Problem 1

Part 1. Download the data set Bank Marketing Data Set. Note that the data file does not contain header names; you may wish to add those. The description of each column can be found in the data set explanation.

```
data <- read.csv("bank-additional-full.csv", header = T, sep = ";", stringsAsFactors = TRUE)
summary(data)
```

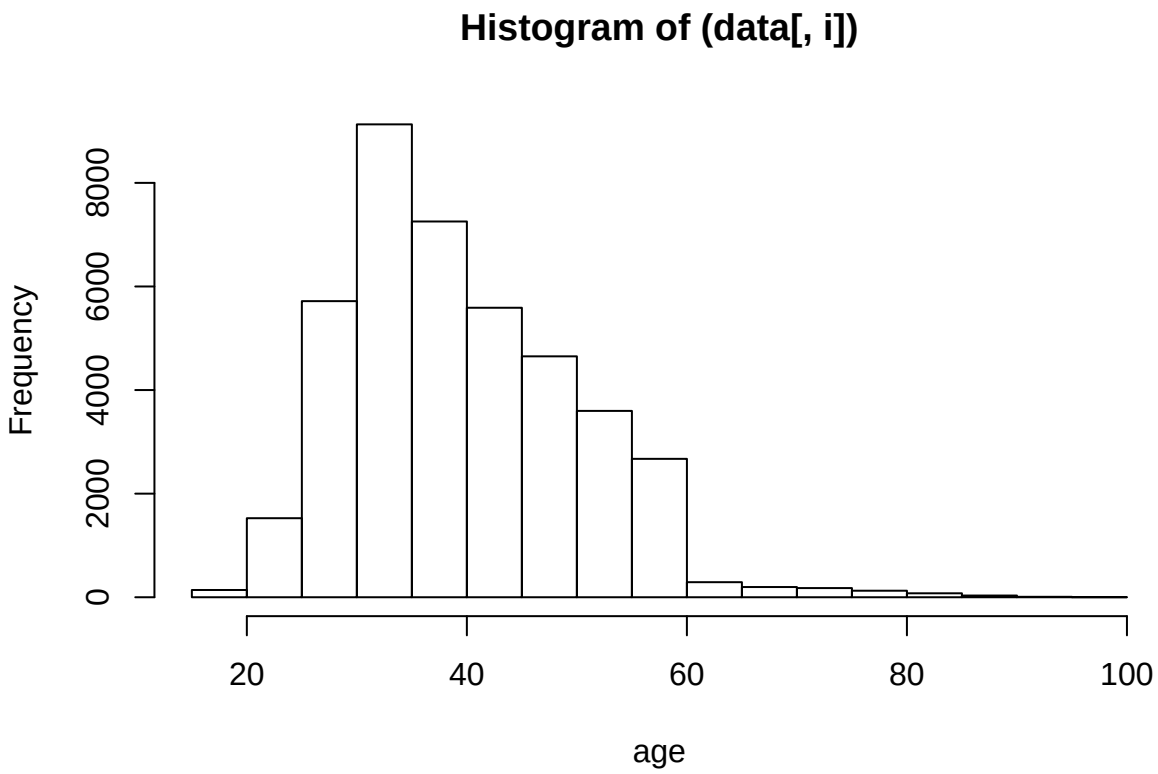
```
##       age                 job           marital
##  Min.   :17.00   admin.     :10422   divorced: 4612
##  1st Qu.:32.00   blue-collar: 9254   married :24928
##  Median :38.00   technician : 6743   single  :11568
##  Mean   :40.02   services   : 3969   unknown :   80
##  3rd Qu.:47.00   management : 2924
##  Max.   :98.00   retired    : 1720
##                  (Other)    : 6156
##                education        default         housing           loan
##  university.degree  :12168   no     :32588   no     :18622   no     :33950
##  high.school        : 9515   unknown: 8597   unknown:  990   unknown:  990
##  basic.9y           : 6045   yes    :    3   yes    :21576   yes    : 6248
##  professional.course: 5243
##  basic.4y           : 4176
##  basic.6y           : 2292
##  (Other)            : 1749
##       contact           month       day_of_week    duration
##  cellular :26144   may    :13769   fri:7827    Min.   :   0.0
##  telephone:15044   jul    : 7174   mon:8514    1st Qu.: 102.0
##                    aug    : 6178   thu:8623    Median : 180.0
##                    jun    : 5318   tue:8090    Mean   : 258.3
##                    nov    : 4101   wed:8134    3rd Qu.: 319.0
##                    apr    : 2632               Max.   :4918.0
##                    (Other): 2016
##     campaign          pdays          previous            poutcome
##  Min.   : 1.000   Min.   :  0.0   Min.   :0.000   failure    : 4252
##  1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000   nonexistent:35563
##  Median : 2.000   Median :999.0   Median :0.000   success    : 1373
##  Mean   : 2.568   Mean   :962.5   Mean   :0.173
##  3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000
##  Max.   :56.000   Max.   :999.0   Max.   :7.000
##
##   emp.var.rate      cons.price.idx  cons.conf.idx     euribor3m
##  Min.   :-3.40000   Min.   :92.20   Min.   :-50.8   Min.   :0.634
##  1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7   1st Qu.:1.344
```

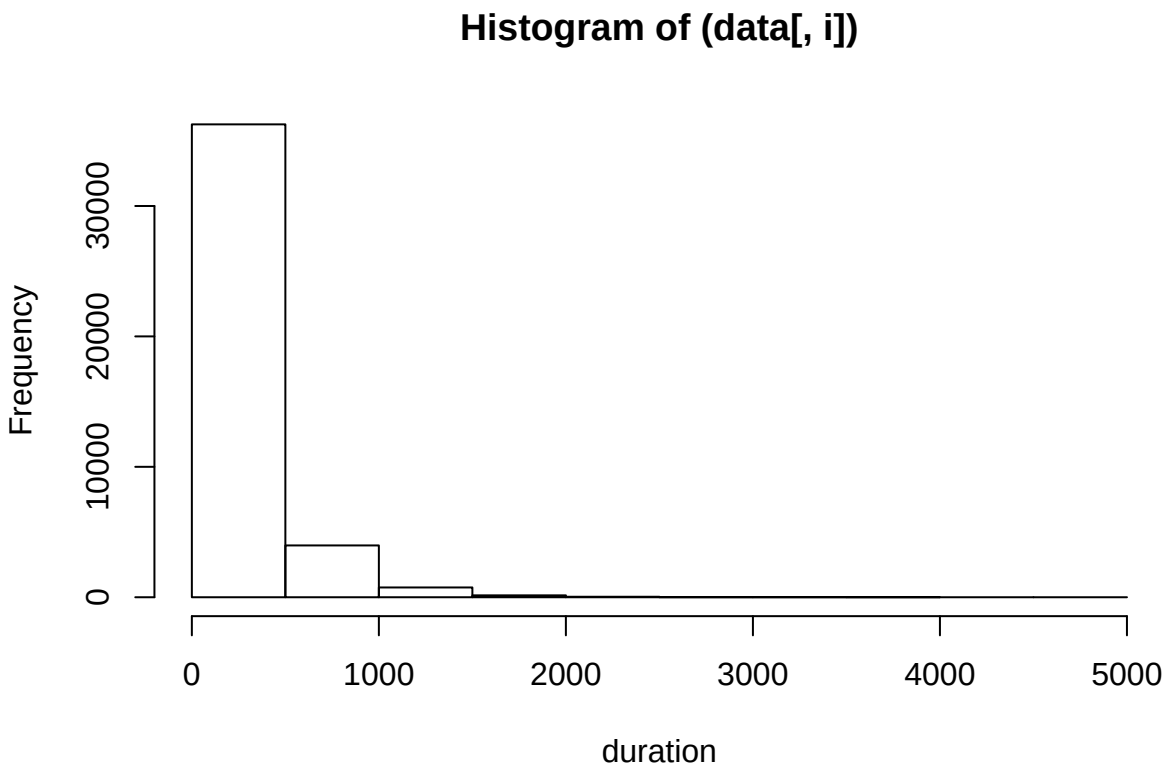
```
##  Median : 1.10000   Median :93.75   Median :-41.8   Median :4.857
##  Mean   : 0.08189   Mean   :93.58   Mean   :-40.5   Mean   :3.621
##  3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4   3rd Qu.:4.961
##  Max.   : 1.40000   Max.   :94.77   Max.   :-26.9   Max.   :5.045
##
##   nr.employed     y
##  Min.   :4964   no :36548
##  1st Qu.:5099   yes: 4640
##  Median :5191
##  Mean   :5167
##  3rd Qu.:5228
##  Max.   :5228
##
```

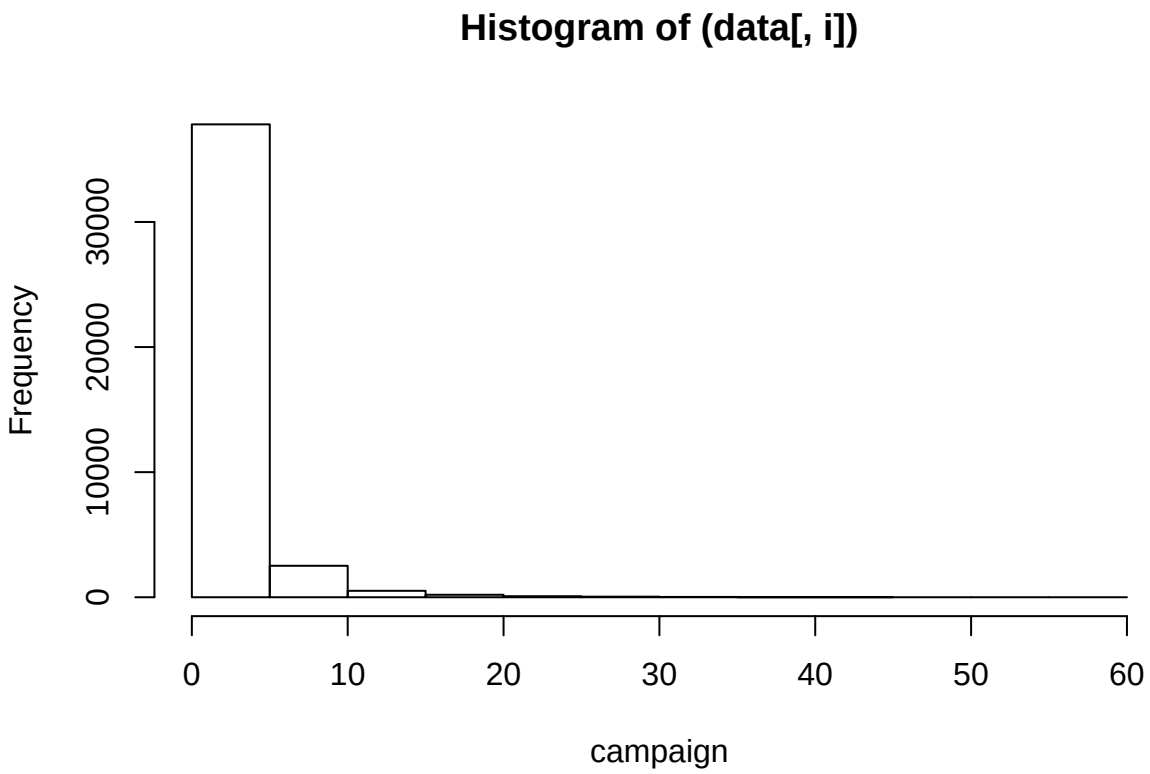
Part 2. Explore the data set as you see fit and that allows you to get a sense of the data and get comfortable with it. Is there distributional skew in any of the features? Is there a need to apply a transform?

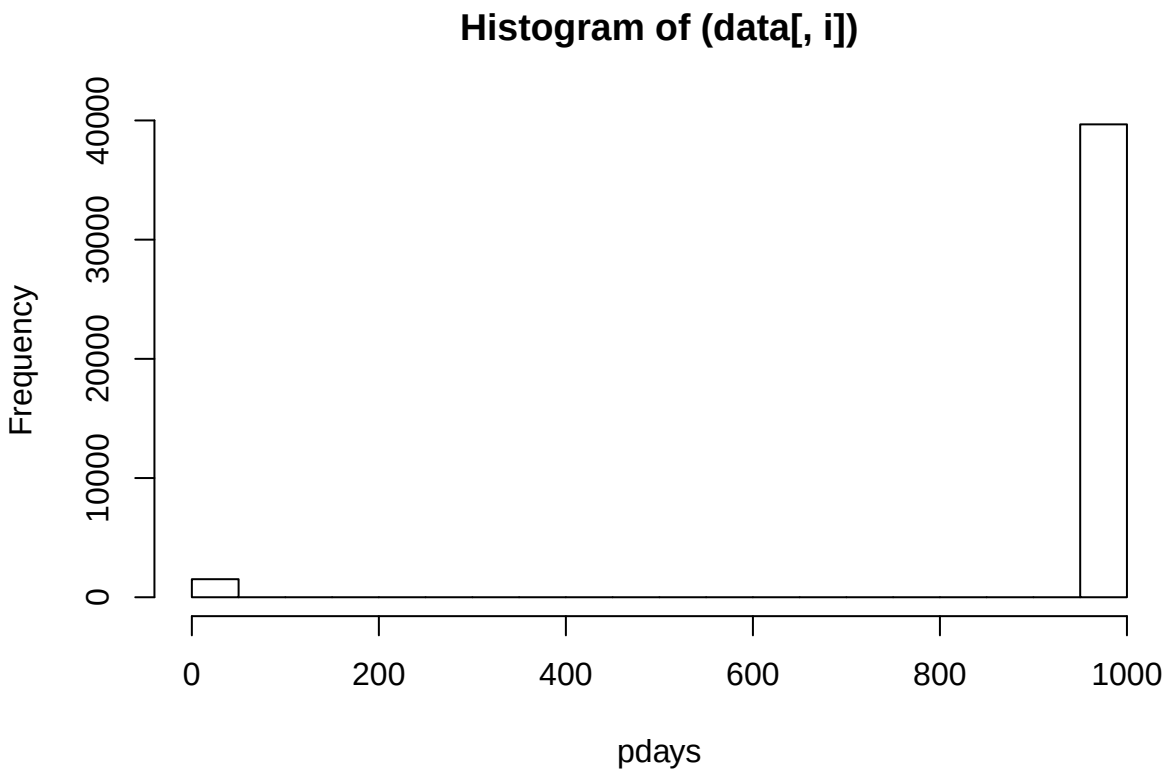
Generated histograms of features to check the skewness and the need for normalization. As most of the features are categorical values we can ignore them for scaling however, it is oserved some numeric features have skewness thus I applied normalization to them.

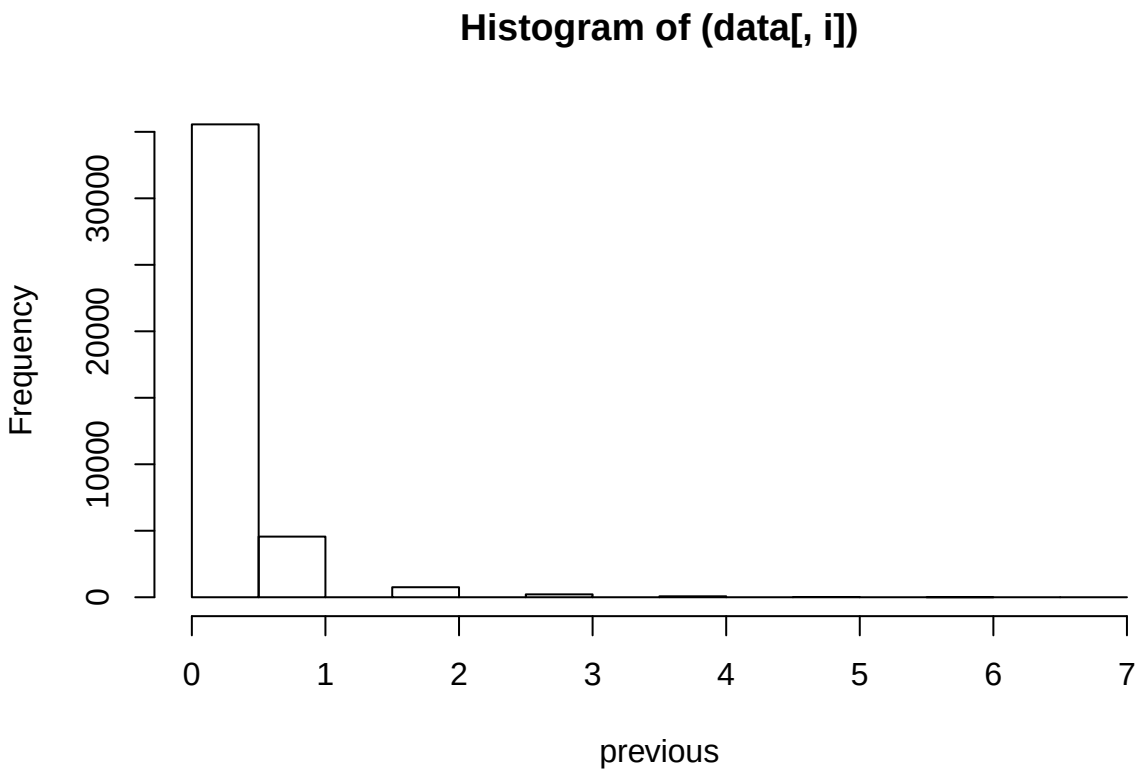
```
# loop to generate histogram
for(i in 1:ncol(data))
{
  # checking for the class of feature
  if (class(data[,i]) =="integer"| class(data[,i]) =="numeric"){
    paste0("Histogram for : ", colnames(data[i]))
  # generating histogram
    hist((data[,i]), xlab = colnames(data[i]))
  }
    else{}
  }
```

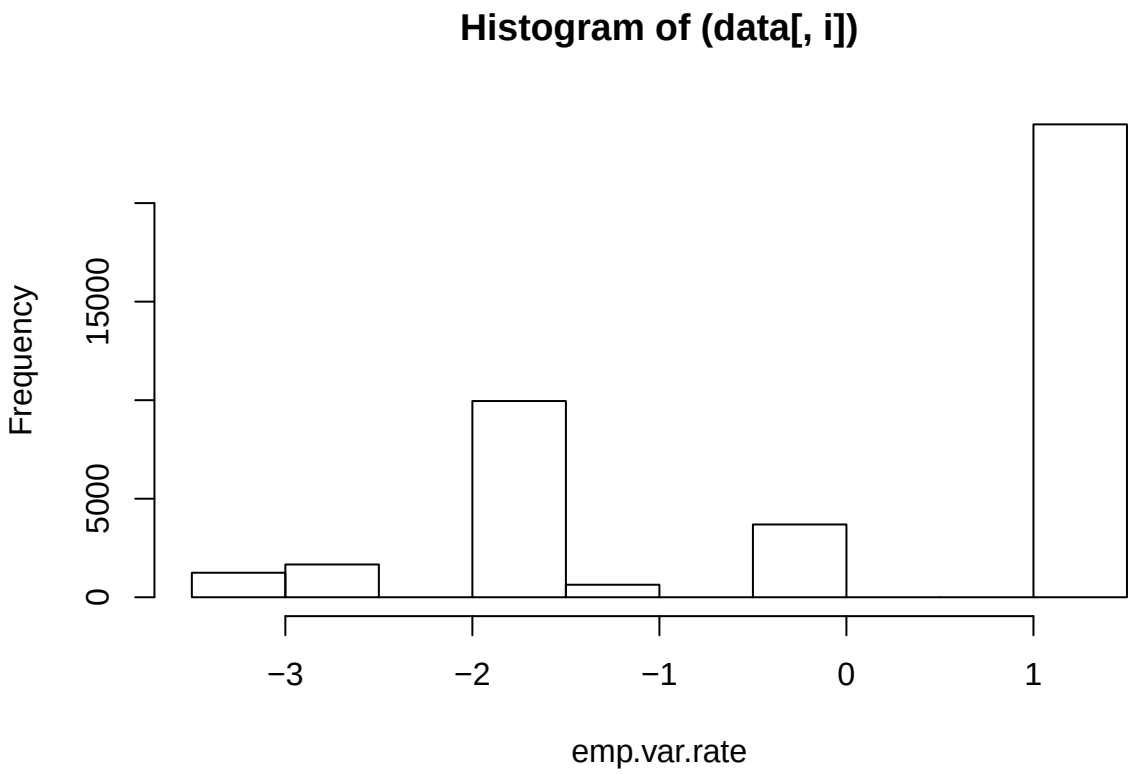

Histogram of (data[, i])
Frequency
0
2000
4000
6000
8000
20
40
60
80
100
age

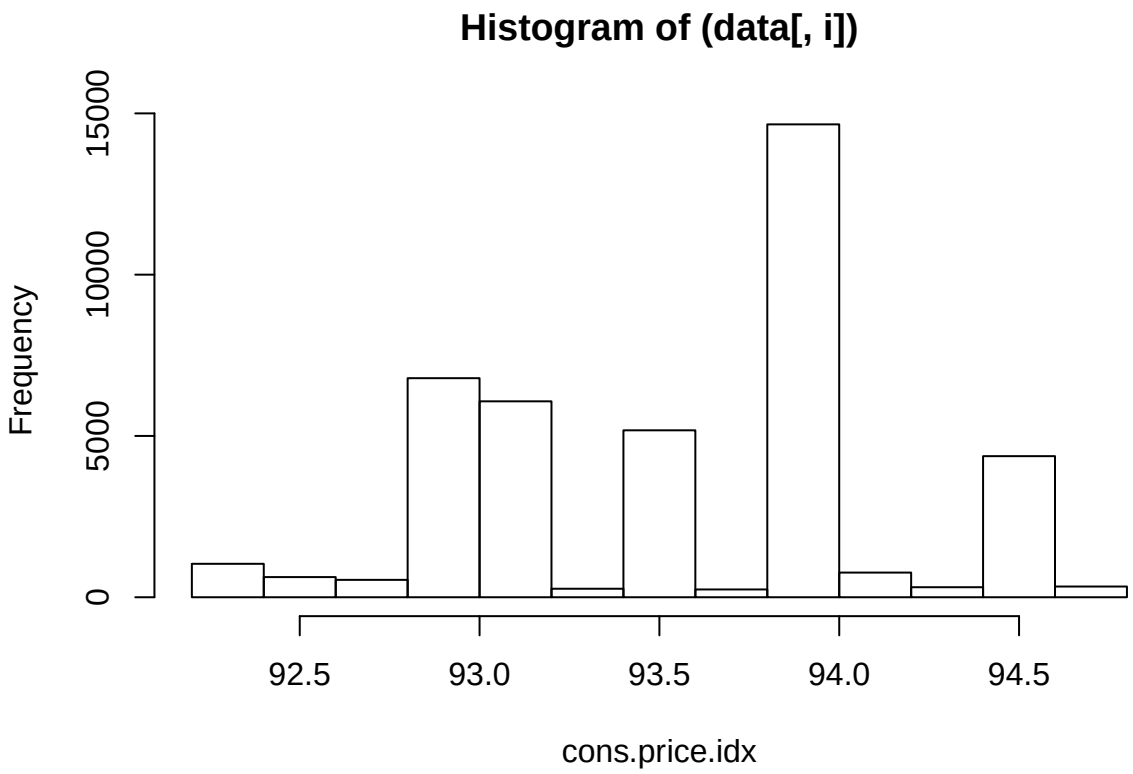

Histogram of (data[, i])
Frequency
0
10000
20000
30000
0
1000
2000
3000
4000
5000
duration

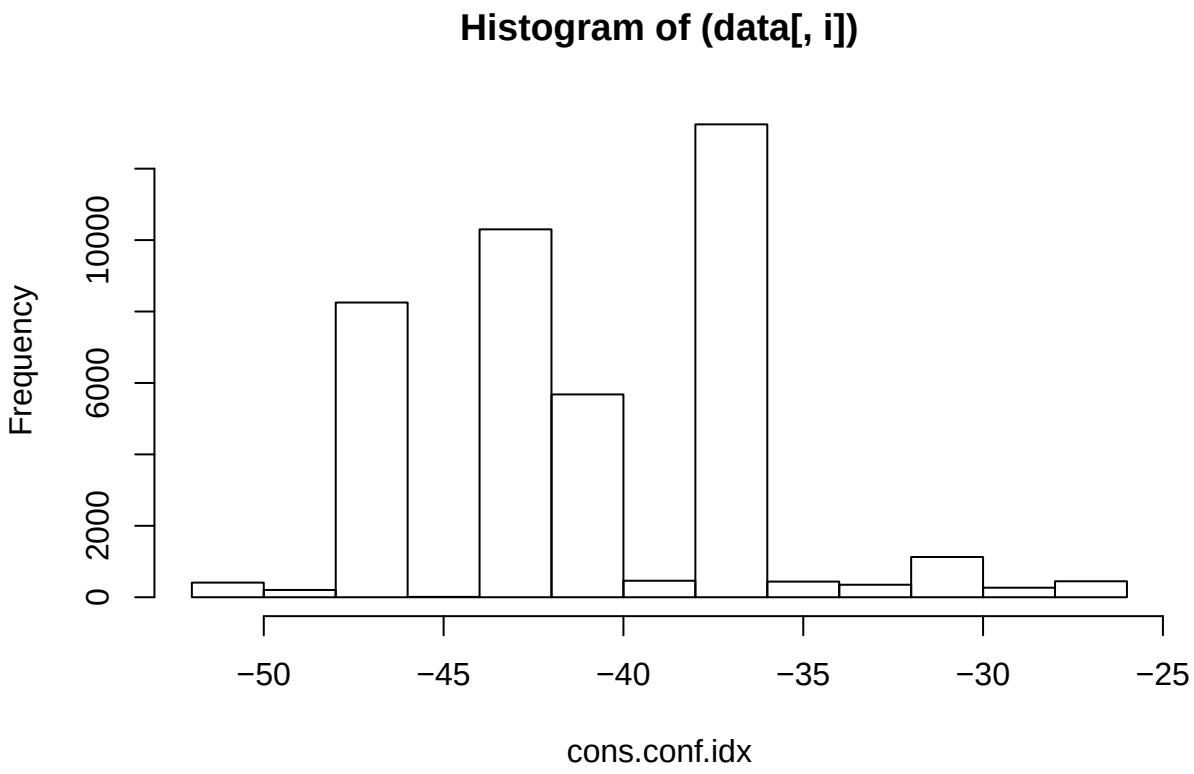

Histogram of (data[, i])
Frequency
0
10000
20000
30000
0
10
20
30
40
50
60
campaign

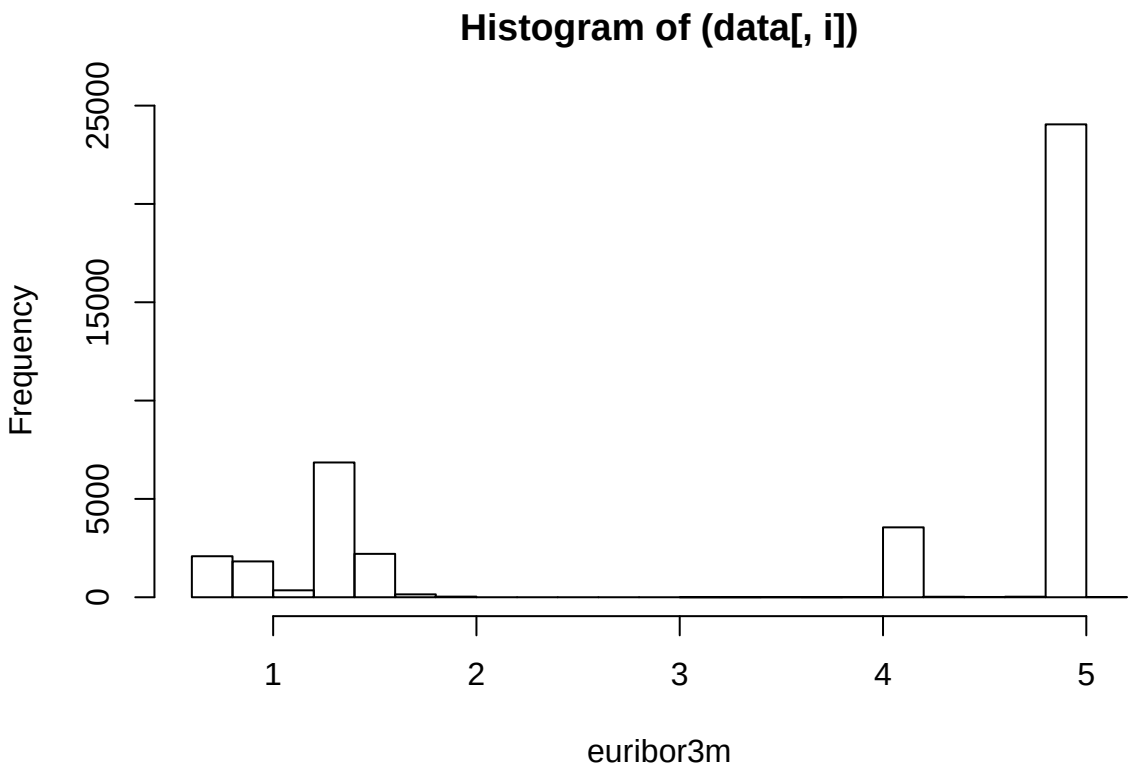

Histogram of (data[, i])
Frequency
0
10000
20000
30000
40000
0
200
400
600
800
1000
pdays

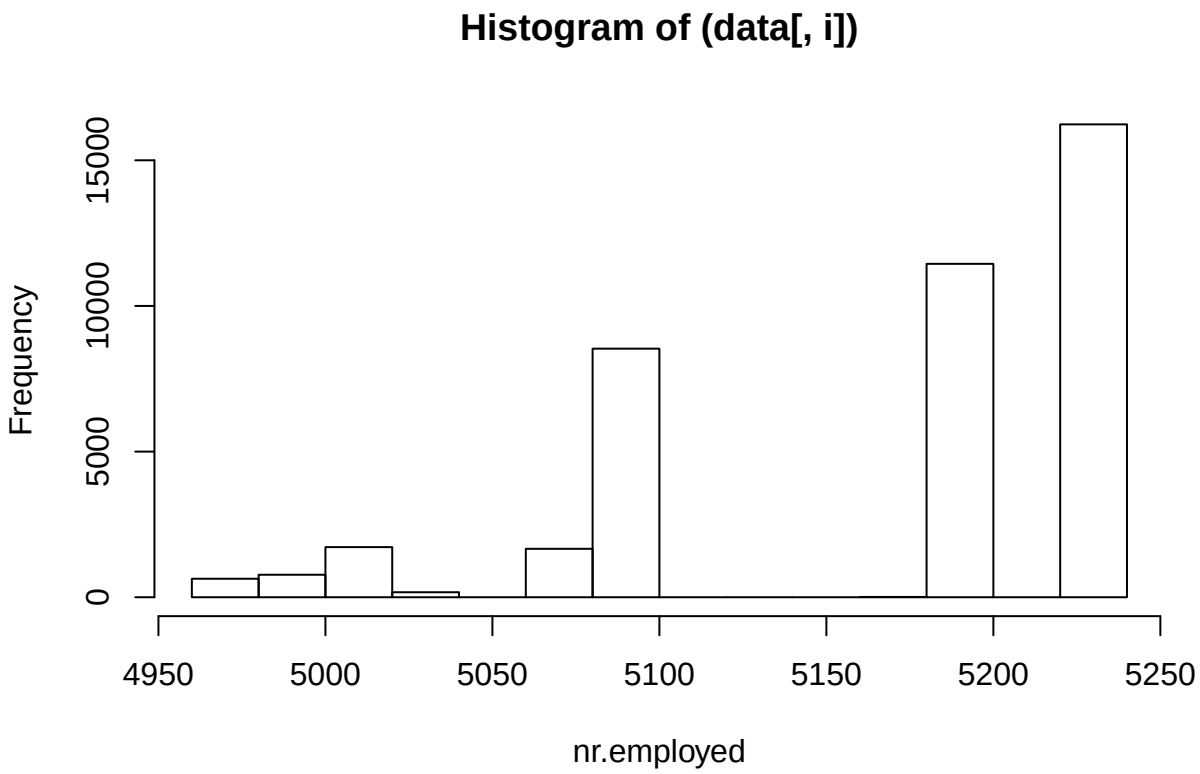

Histogram of (data[, i])
Frequency
0
10000
20000
30000
0
1
2
3
4
5
6
7
previous


Histogram of (data[, i])
Frequency
0
5000
15000
−3
−2
−1
0
1
emp.var.rate


Histogram of (data[, i])
Frequency
0
5000
10000
15000
92.5
93.0
93.5
94.0
94.5
cons.price.idx


Histogram of (data[, i])
Frequency
0
2000
6000
10000
−50
−45
−40
−35
−30
−25
cons.conf.idx


Histogram of (data[, i])
Frequency
0
5000
15000
25000
1
2
3
4
5
euribor3m




```
# loop for applying normalization to feature values
for(i in 1:ncol(data))
{
  # checking for the class of feature
  if (class(data[,i]) =="integer" | class(data[,i]) =="numeric"){
  # applying normalization to feature values
    data[,i] = (data[,i]-min(data[,i]))/(max(data[,i])-min(data[,i]))
  }
  else{}
}

# converting the labels to factors
data$y <- as.factor(data$y)
```

Splitting the data to Training and Testing

```
set.seed(123)
# creating a sample of 75% for train and 25% for test
sample <- sample.int(n = nrow(data), size = floor(.75*nrow(data)), replace = F)
train <- data[sample, ]
test  <- data[-sample, ]
```

Part 3. Build a classification model using a support vector machine that predicts if a bank customer will open a term deposit account.

Using kernlab library to apply svm model using ksvm function

```r
library(kernlab)
# Generating svm classifier using RBF kernel.
svm_classifier <- ksvm(y ~. , data= train, kernel= "rbfdot", kpar=list(sigma=0.015), C=70, cross=4,prob
# summarizing the classifier
svm_classifier
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 70
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.015
##
## Number of Support Vectors : 6121
##
## Objective Function Value : -346712.7
## Training error : 0.065585
## Cross validation error : 0.092648
## Probability model included.
```

Predicting values using a support vector machine classifier.

It is observed that we got an accuracy of 91.4% for the SVM classifier.

```r
# Predicting the class of test data
prediction <- predict(svm_classifier, test[,-21])
# Getting class probabilities for the test data
p <- predict(svm_classifier, test[,-21], type = "probabilities")
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```r
# Generating the confusion matrix for the model
confusionMatrix(test$y, prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  8921  264
##        yes  622  490
##
##                Accuracy : 0.914
##                  95% CI : (0.9084, 0.9193)
##     No Information Rate : 0.9268
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4798
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9348
##             Specificity : 0.6499
##          Pos Pred Value : 0.9713
##          Neg Pred Value : 0.4406
##              Prevalence : 0.9268
##          Detection Rate : 0.8664
##    Detection Prevalence : 0.8920
##       Balanced Accuracy : 0.7923
##
##        'Positive' Class : no
##
```

Part 4. Build another classification model using a neural network that also predicts if a bank customer will open a term deposit account.

Neural Network works with numerical features to give optimal results hence, I converted the non-categorical feature values to numeric format. To reduce the computational time of the model, I also altered some values, like hidden node = 3, threshold = 0.5, stepmax = 1e+08 and linear output = False, so this algorithm with big data can work in Polynomial time instead of Non-Polynomial time.

```
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 3.6.3
```

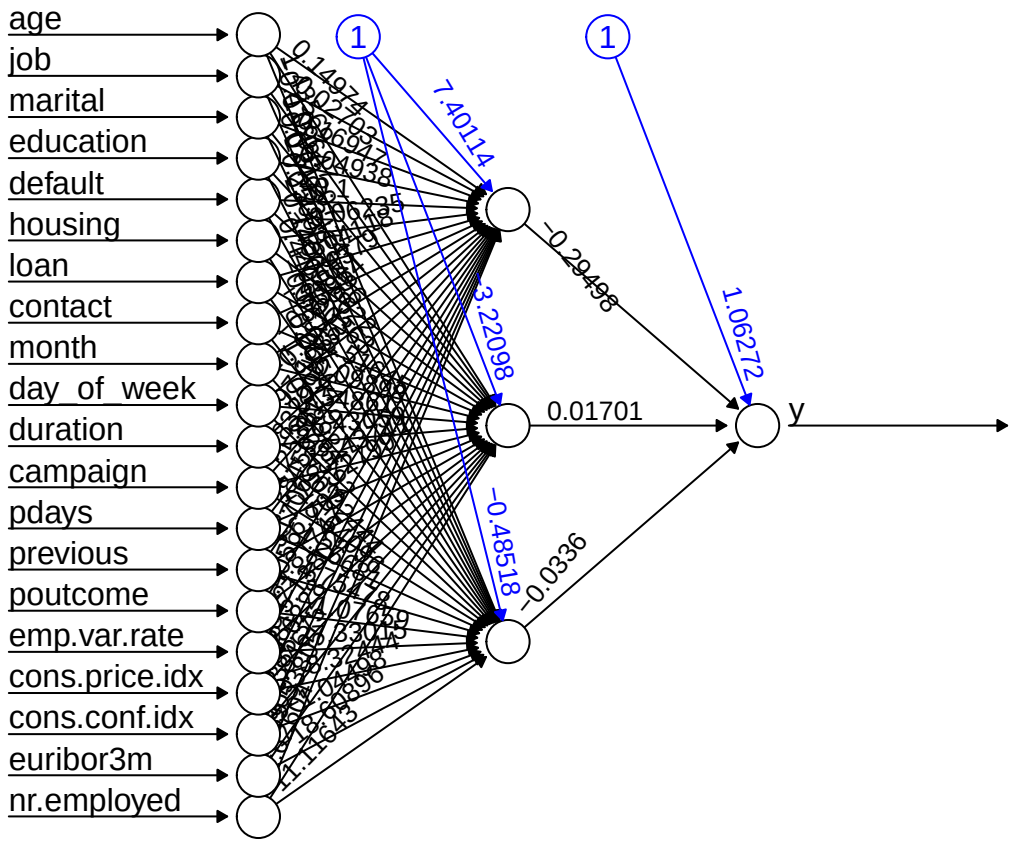
```
# creating a copy of data
data_neural_net <- data
# loop for checking the data type
for(i in 1:ncol(data_neural_net))
{
  if (class(data_neural_net[,i]) !="integer" | class(data_neural_net[,i]) !="numeric")
  {
    # converting non-categorical feature values
    data_neural_net[,i] = as.numeric(data_neural_net[,i])
}
else{}
}
```

```r
# Soothning function for neural network
softplus <- function(x) {
  log(1+exp(x))
}
# Creating training and testing samples
train_NN <- data_neural_net[sample, ]
test_NN  <- data_neural_net[-sample, ]

# Generating the Neural Net model
neural_net_model <- neuralnet(y ~. , data = train_NN, act.fct=softplus, hidden = 3, threshold=0.5, rep=
print("NeuralNet plot")
```

```
## [1] "NeuralNet plot"
```

```r
# Plotting the neural net model
plot(neural_net_model, rep="best")
```



Getting Predictions for test data

It is observed the correlation between values comes out to be 0.62

```r
# computing predictions for test data
model_results <- neuralnet::compute(neural_net_model, test_NN[,-21])

# storing result in variable
```

```r
predicted_strength <- model_results$net.result

# calculating correlation
print("Correlation between default values and predicted values by Neural Net")
```

```
## [1] "Correlation between default values and predicted values by Neural Net"
```

```r
cor(predicted_strength, test_NN$y)
```

```
##           [,1]
## [1,] 0.6263507
```

Part 5. Compare the accuracy of the two models based on AUC.

For AUC plot we need the probability of predicted classes, which we take in terms of by plotting a graph by True Positive vs False positive, we build a curve graph with the each predicted probability for these two variables.There is a 50% line in AUC curve which serves as the base line, if the area under the curve is close to that line it means the model is not doing well. For plotting AUC curve for neural net is simple we have probability of each correlation within $net argument, but for SVM, I specifically calculated and stored values in variable p for each class.

To compare models based on this graph it is observed that the Neural network clearly have better accuracy with ~ 94%. Although if we compare the same models on the basis of accuracy from confusion matrix it is observed SVM perform better with accuracy of about 91.5% whereas the correlation for neural net is ~ 0.60. This justifies the fact that although SVM has better prediction accuracy in comparison with Neural Network, but the rate of false positive over True positive is higher in Neural Net.

```r
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.6.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

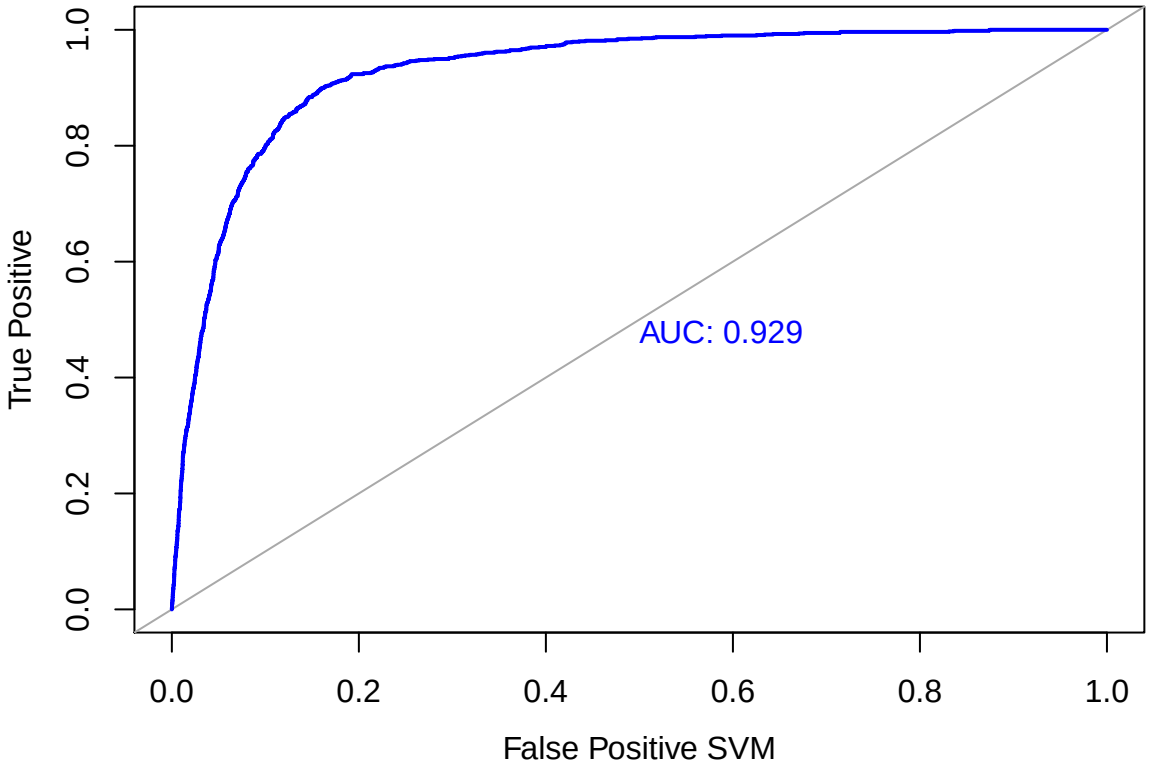
```r
# plotting Auc graph for SVM classifier
print("AUC graph for SVM")
```

```
## [1] "AUC graph for SVM"
```

```r
plot.roc(roc(as.numeric(test$y), as.numeric(p[,1])), legacy.axes = TRUE, print.auc = TRUE, asp = NA, xl
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls > cases
```



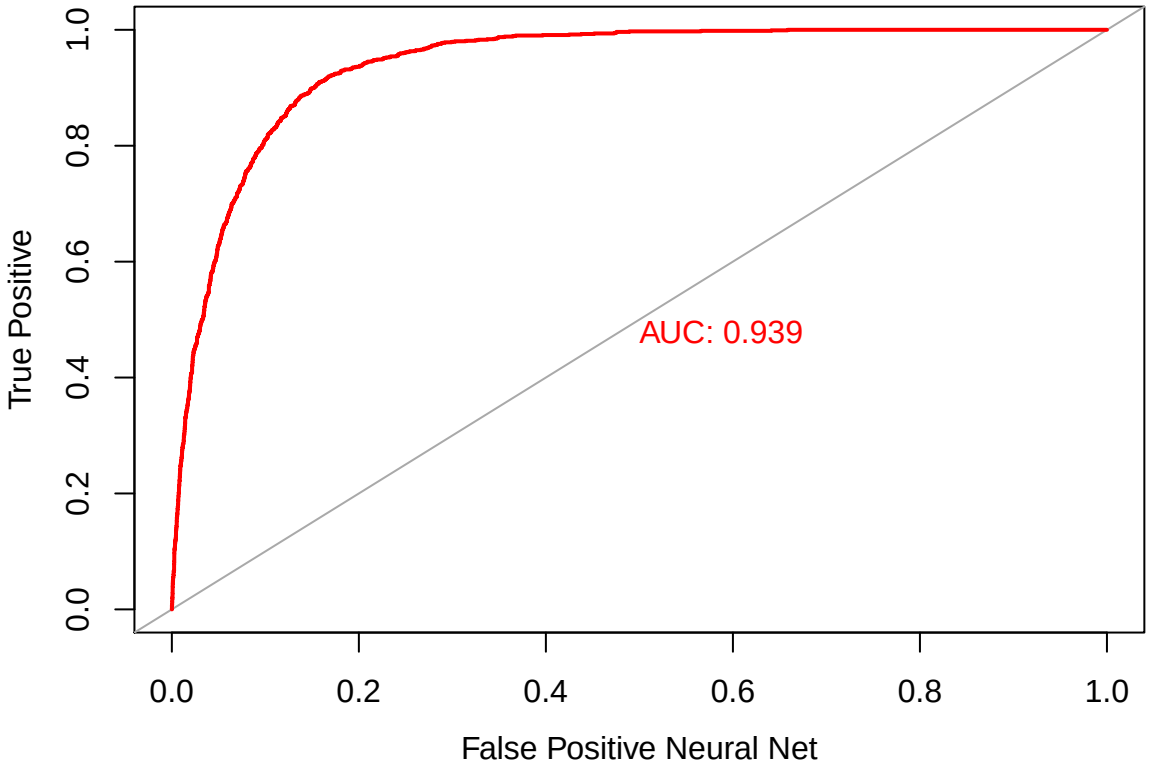

```
# plotting Auc graph for Neural Net
print("AUC graph for Neural Network")
```

```
## [1] "AUC graph for Neural Network"
```

```
plot.roc(roc(test_NN$y, as.numeric(predicted_strength)), legacy.axes = TRUE, print.auc = TRUE, asp = NA
```

```
## Setting levels: control = 1, case = 2
```

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



Part 6. Calculate precision and recall for both models. See this article to understand how to calculate these metrics.

Used caret package to calculate precision and recall for both the models.For calculating the precision and recall we need a matrix of false positve, false negative, true positve and true negative values predicted by the models, which is basically calculated on the bases of true values compared with predicted values.

For precision we need True positive divided by true positive plus false positve,

For recall we need True positive divided by true positive plus false negative.

```
# precision for svm
precision_svm <- posPredValue(prediction, test$y, positive="no")

# recall for svm
recall_svm <- sensitivity(prediction, test$y, positive="no")

paste0("Precision for SVM : ", precision_svm)
```

```
## [1] "Precision for SVM : 0.934821335009955"
```

```
paste0("Recall for SVM : ", recall_svm)
```

```
## [1] "Recall for SVM : 0.97125748502994"
```

```r
# storing values in different variable
predicted_strength1 <- predicted_strength

# changing the values on the basis of ratio of prediction to original factor values in dataset to get p
predicted_strength1 <- ifelse(predicted_strength1>1.5, 2, 1)

# precision for Neural net
precision_NN <- posPredValue(as.factor(predicted_strength1), as.factor(test_NN$y), positive="1")

# reacll for NeuralNet
recall_NN <- sensitivity(as.factor(predicted_strength1), as.factor(test_NN$y), positive="1")

paste0("Precision for NeuralNet : ", precision_NN)
```

```
## [1] "Precision for NeuralNet : 0.933763127794531"
```

```r
paste0("Recall for NeuralNet : ", recall_NN)
```

```
## [1] "Recall for NeuralNet : 0.977681001633097"
```