İSTANBUL ŞEHİR ÜNİVERSİTESİ

ISE 534: Data Mining

"Final Project"

Submitted by:

Kanza Haider

Qurat ul Aain

Noran Islam

Nuha Hashem

Submitted to:

Prof. Selim Zaim

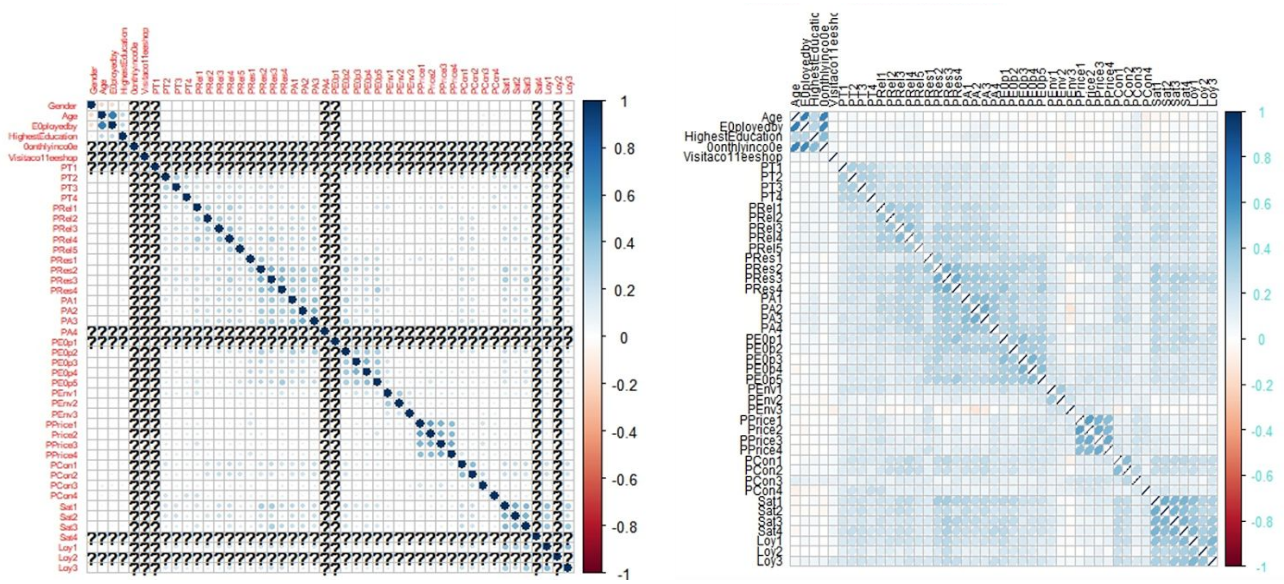
PART A

Using Principal component analysis, with the varimax rotation procedure to determine the service quality variables as input variables.

Step 1: Delete the missing values and do the data scaling

First we prepare the data by deleting the missing values that are interfering with our results. The observations reduced from 2086 to 2077 observations with 49 variables.

```
coffeeall_Perceptions <- na.omit(coffeeall_Perceptions)
```



Step 2: Create a new data set with service quality variables only

```
#Generate a newdata of only service variables
servicedata <- coffeeall_Perceptions[c(8:40)]
```

The data is reduced to 34 variables now from a total of 44 variables

Step 3: Check the available variables in the new dataset

```
#check available variables
colnames(servicedata)
```

```
> colnames(servicedata)
 [1] "PT1"     "PT2"     "PT3"     "PT4"     "PRel1"   "PRel2"
 [7] "PRel3"   "PRel4"   "PRel5"   "PRes1"   "PRes2"   "PRes3"
[13] "PRes4"   "PA1"     "PA2"     "PA3"     "PA4"     "PEmp1"
[19] "PEmp2"   "PEmp3"   "PEmp4"   "PEmp5"   "PEnv1"   "PEnv2"
[25] "PEnv3"   "PPrice1" "Price2"  "PPrice3" "PPrice4" "PCon1"
[31] "PCon2"   "PCon3"   "PCon4"
```

As you could see, the variables of gender, age, employed by, highest education, monthly income etc has been filtered out which do not have any relation with the service quality variables**.**

Step 4: Check the variable class of each variable to be all numeric

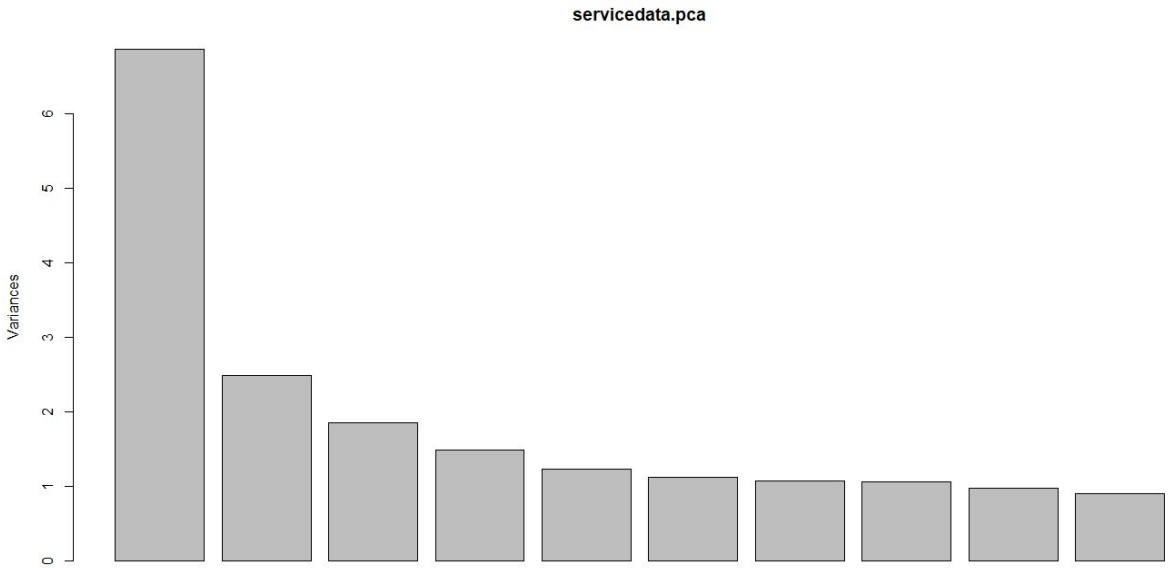
This is essential as any non-numeric or categorical variables have to be changed into numeric. In our case, all variables are numeric as we could see. Therefore, we do not require any work here.

```
#check variable class
str(servicedata)
```

```
> str(servicedata)
Classes 'tbl_df', 'tbl' and 'data.frame':       2077 obs. of  33 variables:
 $ PT1    : num  5 4 4 4 4 3 4 4 4 4 ...
 $ PT2    : num  5 4 3 3 4 4 4 4 5 4 ...
 $ PT3    : num  4 4 4 4 4 4 4 4 5 4 ...
 $ PT4    : num  5 4 4 4 4 5 4 4 3 4 ...
 $ PRel1  : num  5 5 5 4 5 3 3 4 3 4 ...
 $ PRel2  : num  5 4 3 3 5 3 4 3 3 4 ...
 $ PRel3  : num  5 5 5 3 4 3 4 4 3 4 ...
 $ PRel4  : num  4 5 5 4 4 3 4 4 5 4 ...
 $ PRel5  : num  4 4 4 4 4 3 3 4 5 4 ...
 $ PRes1  : num  4 3 5 5 3 4 5 4 4 4 ...
 $ PRes2  : num  5 4 4 5 3 4 4 3 5 4 ...
 $ PRes3  : num  5 5 5 5 3 3 4 3 5 4 ...
 $ PRes4  : num  4 5 3 4 3 5 4 4 5 4 ...
 $ PA1    : num  5 4 3 4 3 3 3 4 4 4 ...
 $ PA2    : num  5 4 4 4 3 4 3 4 4 4 ...
 $ PA3    : num  5 4 5 4 3 3 3 4 4 4 ...
 $ PA4    : num  5 5 4 4 4 3 4 4 4 4 ...
 $ PEmp1  : num  5 4 4 5 4 3 3 3 4 4 ...
 $ PEmp2  : num  5 4 3 4 4 3 4 3 4 4 ...
 $ PEmp3  : num  4 3 4 3 4 3 4 3 3 4 ...
 $ PEmp4  : num  5 4 4 4 4 3 4 3 3 4 ...
 $ PEmp5  : num  5 4 4 4 4 3 4 3 3 4 ...
 $ PEnv1  : num  4 4 4 4 4 3 4 4 3 4 ...
 $ PEnv2  : num  4 4 4 4 4 4 4 4 3 4 ...
 $ PEnv3  : num  3 3 4 4 4 3 3 4 4 4 ...
 $ PPrice1: num  5 3 4 3 4 4 3 3 5 5 ...
 $ Price2 : num  5 4 3 4 4 3 4 3 5 5 ...
 $ PPrice3: num  5 4 4 3 4 3 4 3 5 5 ...
 $ PPrice4: num  5 4 4 3 4 3 4 3 5 4 ...
 $ PCon1  : num  5 5 4 3 4 3 4 3 4 4 ...
 $ PCon2  : num  5 4 3 3 4 3 4 3 4 4 ...
 $ PCon3  : num  4 3 5 3 3 3 3 3 4 4 ...
 $ PCon4  : num  5 5 5 4 4 3 3 4 3 4 ...
```

Step 5: Apply Principal Component Analysis, PCA

```
#PCA
library(psych)
servicedata.pca <- prcomp(servicedata, scale. = T)
summary(servicedata.pca)
plot(servicedata.pca)
```



As we can see from this summary of the principal component analysis, there are 10 main components with a variance score higher than 1.

```
> servicedata.pca$center
       PT1        PT2        PT3        PT4       PRel1       PRel2
3.757081133 3.635621699 3.843014882 3.734997600 3.840134422 3.826212194
      PRel3       PRel4       PRel5       PRes1       PRes2       PRes3
3.898223716 3.893903024 3.717234758 3.462313970 3.784445511 3.857417187
      PRes4         PA1         PA2         PA3         PA4       PEmp1
3.649063850 3.855496880 3.946231397 3.976956313 3.726836294 3.649063850
      PEmp2       PEmp3       PEmp4       PEmp5       PEnv1       PEnv2
3.803648584 3.627460394 3.650024004 3.638502160 3.544407105 3.426308209
      PEnv3     PPrice1      Price2     PPrice3     PPrice4       PCon1
2.923667787 3.467114738 3.548247720 3.561209794 3.540566491 3.938070091
      PCon2       PCon3       PCon4
3.873259722 3.410465675 3.582813250
>
```

Center and scale refers to respective mean and standard deviation of the variables that are used for normalization prior to implementing PCA

Outputs the standard deviation of variables
servicedata.pca$scale

```
> servicedata.pca$scale
         PT1          PT2          PT3          PT4         PRel1
0.9935147666 1.0661224754 0.9955872539 1.0413447937 1.0519102888
       PRel2        PRel3        PRel4        PRel5         PRes1
1.0379653891 1.0661194474 1.0956977627 1.1497924618 1.2570247657
       PRes2        PRes3        PRes4          PA1           PA2
1.1333339057 1.0906670936 1.1426719438 1.1065920199 1.0984177592
         PA3          PA4        PEmp1        PEmp2        PEmp3
1.0657748515 1.1370052367 1.1359266414 1.1223629079 1.1185513793
       PEmp4        PEmp5        PEnv1        PEnv2        PEnv3
1.0754146038 1.1249489532 1.1081399858 1.0873118352 1.3044782061
     PPrice1       Price2      PPrice3      PPrice4        PCon1
1.1705022772 1.0749753933 1.1038631299 1.1428519300 1.0761163678
       PCon2        PCon3        PCon4
1.0782409463 1.2962022130 1.2285556749
> |
```

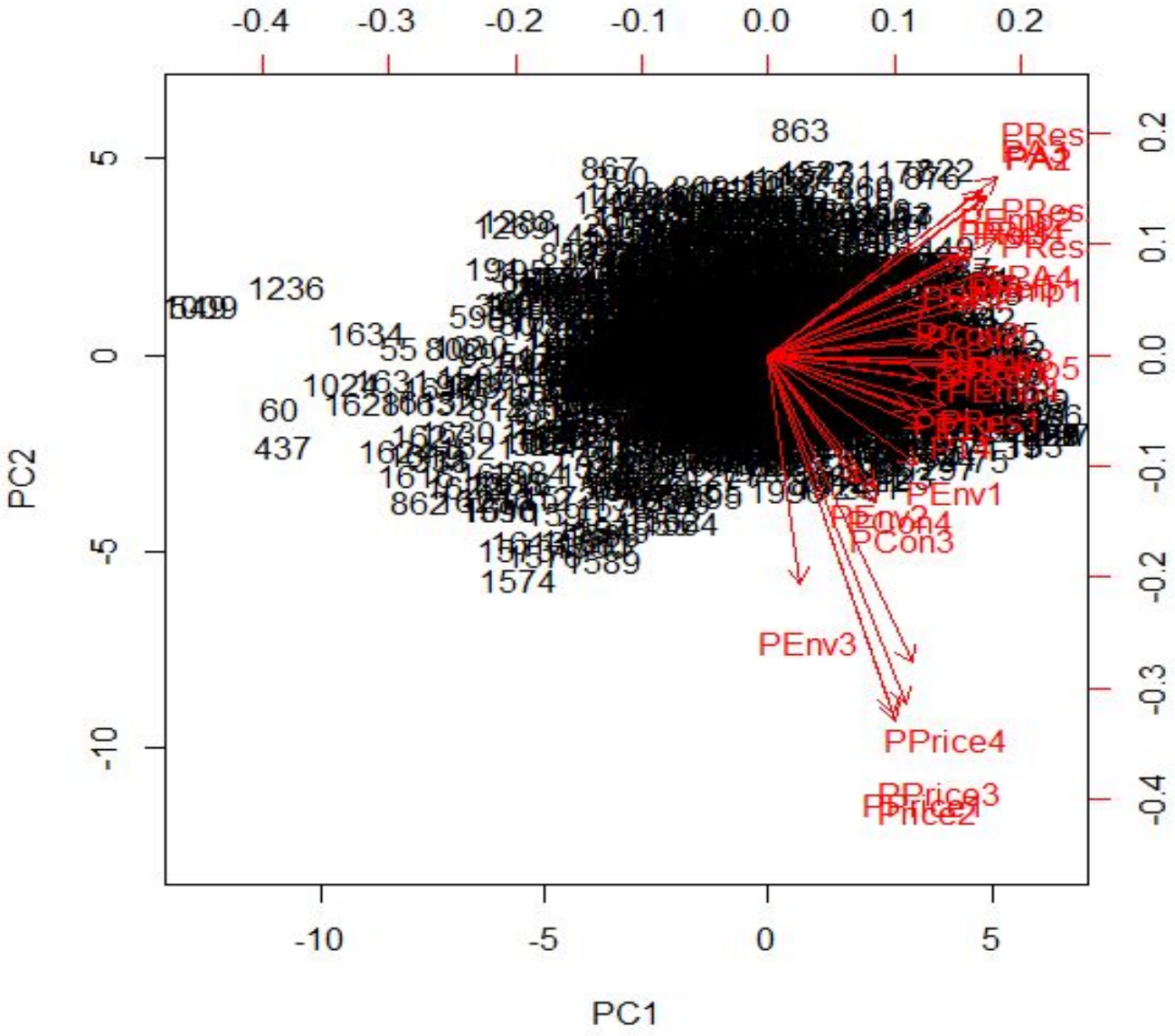
Rotation
servicedata.pca$rotation

The rotation measure provides the principal component loading. Each column of rotation matrix contains the principal component loading vector. This is the important measure we are interested in.

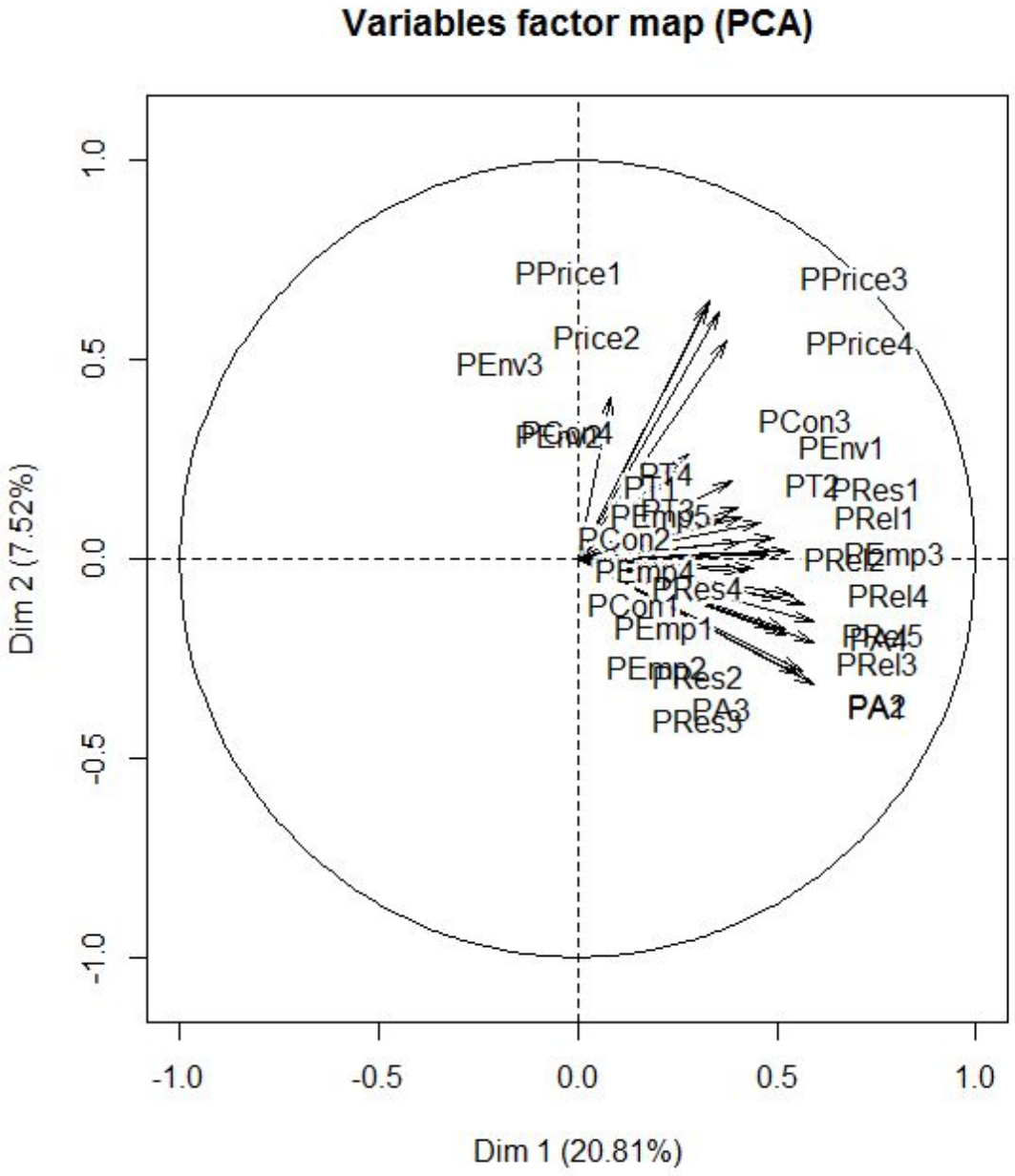
This returns 33 principal components loadings. In a data set, the maximum number of principal component loadings is a minimum of (n-1, p). Let's look at first 4 principal components and first 5 rows.

```
> servicedata.pca$rotation[1:5,1:4]
             PC1          PC2       PC3          PC4
PT1    0.1396260 -0.06183631 0.1748582 -0.320662390
PT2    0.1575635 -0.06545589 0.2040199 -0.293002588
PT3    0.1568114 -0.02558078 0.2136531 -0.281615873
PT4    0.1541018 -0.08171313 0.1932217 -0.277067216
PRel1  0.1863756 -0.01324805 0.2350397 -0.002750822
```

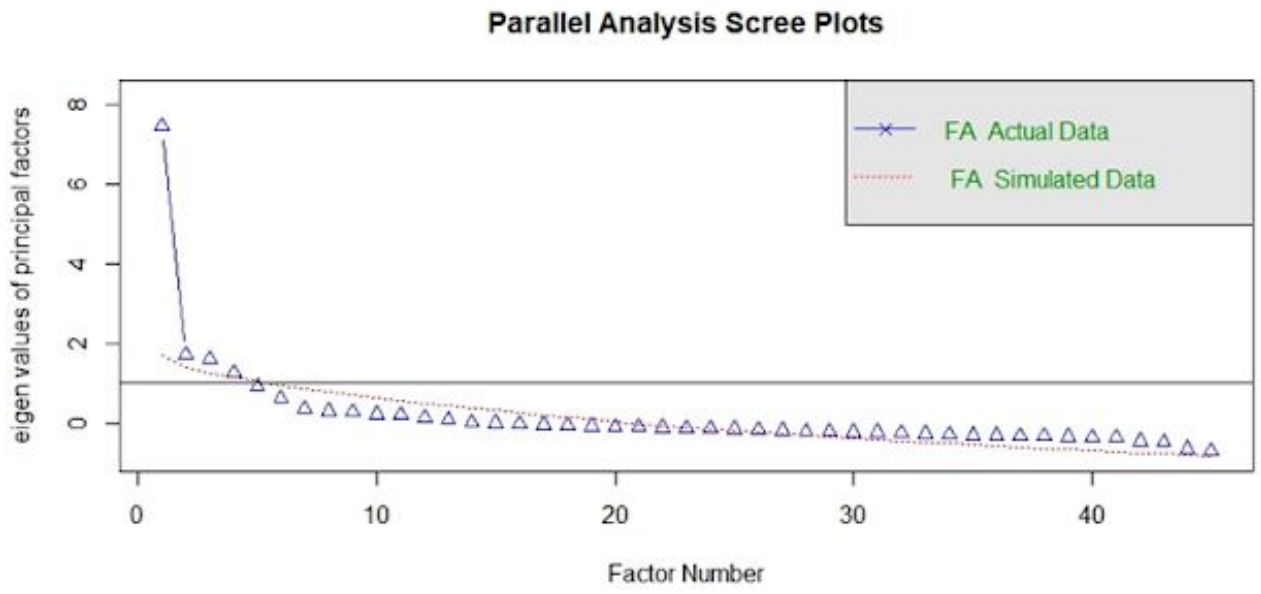
#Biplot
Biplot (servicedata.pca, scale = 0)



#Factor Analysis
Entering raw data and extracting PCs from the correlation matrix.



#Parallel Analysis
This tells us about to find the acceptable number of factors and generate a scree plot



# Compute standard deviation of each principal component
Std_dev <- servicedata.pca$dev
Std_dev

```
> std_dev
 [1] 2.6203840570 1.5759333754 1.3601703058 1.2205024699 1.1082153158
 [6] 1.0613375002 1.0358163256 1.0276495417 0.9907177238 0.9515705787
[11] 0.8948104262 0.8809420269 0.8760327481 0.8642996035 0.8523980263
[16] 0.8474373333 0.8414436137 0.8263116376 0.8081552601 0.7979300338
[21] 0.7819285287 0.7776157087 0.7639099140 0.7475255049 0.7401218311
[26] 0.7367941259 0.7280731488 0.7105989627 0.6980743968 0.6781196081
[31] 0.6643115116 0.6593731359 0.6448575160
>
```

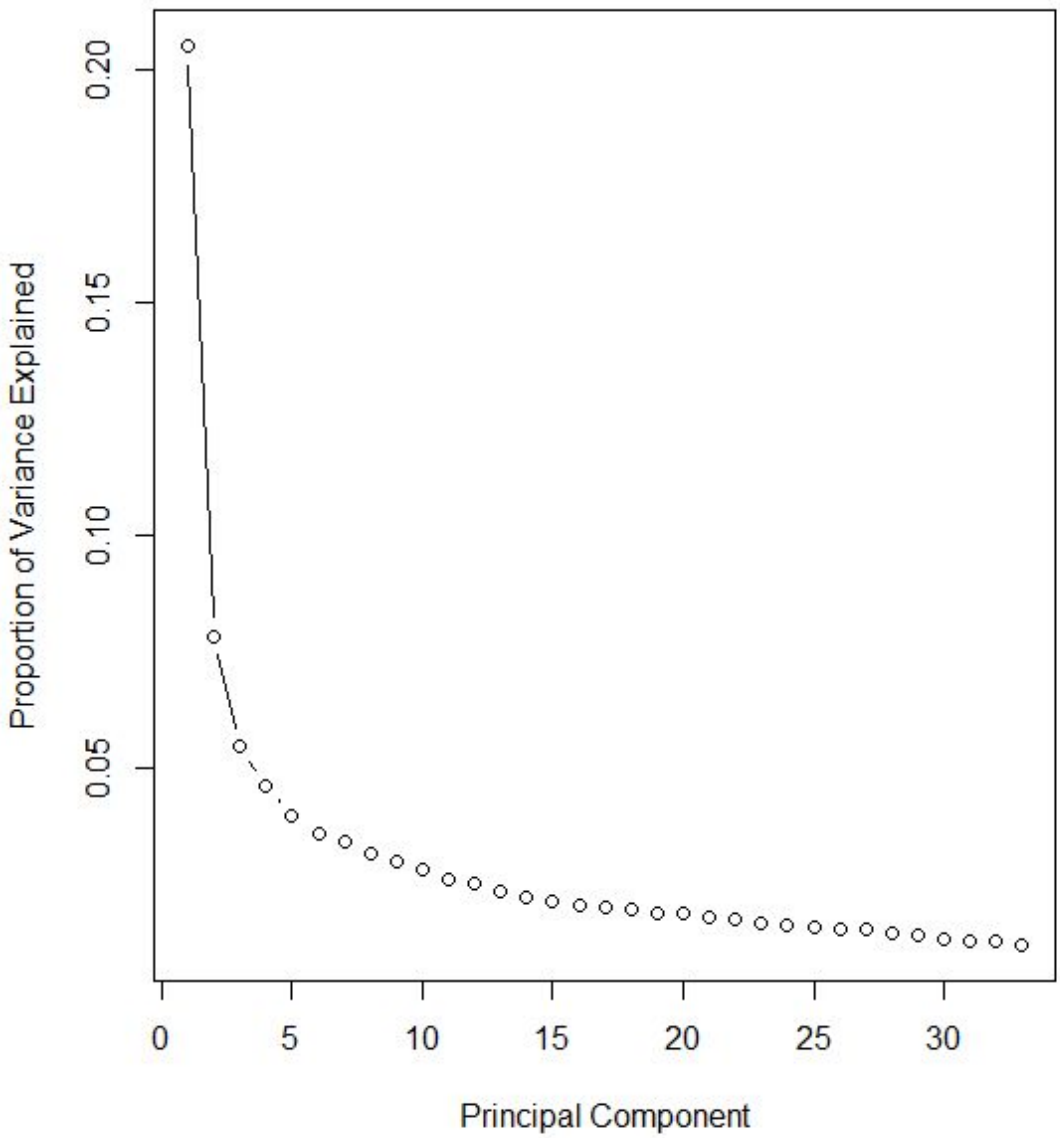
# Compute variance
Pr_var <- std_dev^2
Pr_var

```
> pr_var
 [1] 6.8664126061 2.4835660038 1.8500632609 1.4896262790 1.2281411862
 [6] 1.1264372893 1.0729154604 1.0560635806 0.9815216082 0.9054865662
[11] 0.8006856988 0.7760588548 0.7674333758 0.7470138045 0.7265823953
[16] 0.7181500339 0.7080273551 0.6827909224 0.6531149245 0.6366923389
[21] 0.6114122241 0.6046861904 0.5835583567 0.5587943804 0.5477803249
[26] 0.5428655839 0.5300905100 0.5049508858 0.4873078634 0.4598462028
[31] 0.4413097845 0.4347729323 0.4158412160
>
```

# Proportion of variance explained
Prop_varex <- pr_var/sum(pr_var)
Prop_varex[1:20]

```
> prop_varex[1:20]
 [1] 0.20807310928 0.07525957587 0.05606252306 0.04514019027
 [5] 0.03721639958 0.03413446331 0.03251258971 0.03200192669
 [9] 0.02974307904 0.02743898685 0.02426320299 0.02351693499
[13] 0.02325555684 0.02263678196 0.02201764834 0.02176212224
[17] 0.02145537440 0.02069063401 0.01979136135 0.01929370724
>
```

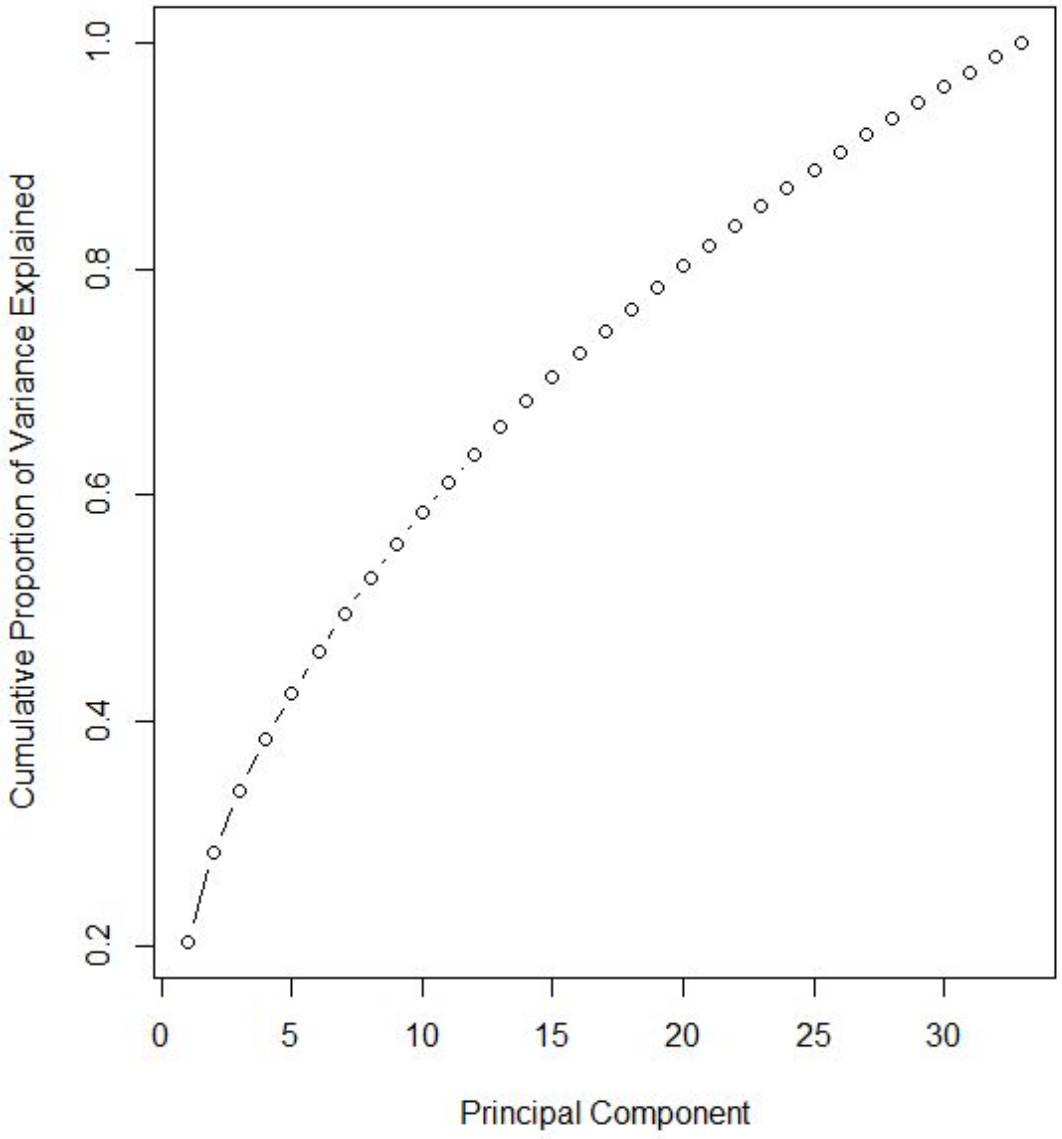
# Scree plot
plot(prop_varex, xlab = “principal component”, ylab = “Proportion of variance explained”, type = “b”)



The plot above shows that ~ 30 components explains around **98.4% variance** in the data set. In order words, using PCA we have reduced predictors to 30 without compromising on explained variance. This is the power of PCA> Let's do a confirmation check, by plotting a cumulative variance plot. This will give us a clear picture of number of components.

Following is the Cumulative scree plot:

Plot (cumsum(prop_varex), xlab = “Principal Component”, ylab = “Cumulative Proportion of variance explained”, type = “b”)



The plot shows that around **30** components result in variance close to **~ 95%.**

PART B

In the following steps you'll see below, we compute the average value of satisfaction variables and then create the binary ***"SatisfactionVariable"*** as our *output variable*, and use principal components of the Service Data as our *input variables*.

'1'in the SatisfactionVariable refers to the high satisfied customers and '0' refers to low satisfied customers.

Working Steps

1. Omit the missing values from the data set.
2. Compute the `rowMeans` of the 4 satisfaction variables (sat1, sat2, sat3 and sat4) and store the results in a new variable ***"Mean"***
3. Take the average of the Mean column and fix it as the threshold value. (The mean in this case is 3.84 for all 2077 observations in the column)
4. Create a new column of "binSat",use the threshold value to assign binary values into the column. Assign 1 if the Mean value of the observation is greater than the threshold and 0 if Mean value is less
5. View the first few observations to confirm that our variable is successfully converted to binary
6. Combine both input(the PCs) and output variable (binSat) into one new dataset, `data` . Give name to the binSat variable as `SatisfactionVariable`

Working Code

```
coffeeall_Perceptions <- na.omit(coffeeall_Perceptions)
coffeeall_Perceptions$Mean <- rowMeans(coffeeall_Perceptions[41:44])
mean(coffeeall_Perceptions$Mean) #Mean is 3.84

#Converting into binary , 0 if less than mean and 1 if more than mean
coffeeall_Perceptions$binsat <- 1
coffeeall_Perceptions$binsat[coffeeall_Perceptions$Mean<=mean(coffeeall_Perceptions$Mean)]<-0

#add a data with principal components
data <- data.frame(SatisfactionVariable = coffeeall_Perceptions$binsat, servicedata.pca$x)
```

Preview of the Data

Following is the preview of the `data` with first 10 PCs shown:

	SatisfactionVariable	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
1	1	5.023047988	-1.35396563	0.33009056	0.37148414	0.277040304	-0.3464166991	0.862282138	0.41700950	0.357208336	-0.57779529
2	0	2.207265432	0.14747883	0.85924097	-0.16508701	-0.033039160	0.6223053768	-0.251697673	-0.04423664	-0.088197367	-1.44018206
3	1	1.467768660	-0.72542923	0.13229923	-0.66914057	0.014963348	0.4395921353	-1.359077676	-0.52736412	0.772143171	-0.19146077
4	0	0.699524022	0.31779515	-1.50657595	-1.34507333	0.831223715	0.8226472407	-0.626023575	-1.30478225	-0.024636831	-0.15868911
5	0	0.268138275	-1.66686373	0.51088682	-0.76816339	-0.287683138	0.2056066346	-0.600495052	1.88043174	0.011879833	0.03740518
6	0	-1.958430136	-0.35207767	-0.38853701	-1.09743990	0.852429266	0.2235569010	0.025175571	-1.35833921	-0.263566547	-0.30865835
7	1	0.078299404	-0.84025065	-0.59214143	-0.74250303	-0.033787381	-0.5007761046	-0.328679929	-0.11484687	-1.404225952	-0.15482260
8	1	-0.937620368	0.02288448	0.42552723	-1.51445372	0.300896814	1.2492517097	-0.112607168	-0.67783312	0.647646687	0.57746341
9	1	1.609798642	-1.58024819	0.23992485	0.72793007	0.924693648	-0.2710479031	0.205414128	-2.40070304	-1.326089524	0.73045701
10	1	1.696986278	-2.08003367	-0.28341898	0.12934537	0.082491836	0.2605933084	-0.177778321	-0.20336314	-0.013495816	0.08262431
11	0	0.027627815	1.21669035	-0.60244683	0.05713161	0.844265454	0.4565897554	-0.467926935	0.61970680	1.291236858	0.04094573

Data Partitioning

Once our data is prepared of total 2077 observation and consisting of the binary outcome `SatisfactionVariable` along with the Principle components together, we shall now partition the data into Training and Test set to be used for training and testing of our prediction models.

- ➔ The `training` set is divided such that it consists of 80% of the entire data set with 1678 observations
- ➔ The `testing` set is divided such that it consists of 20% of the remaining data set with 399 observations

Working code:

```
set.seed(111)
ind <- sample(2, nrow(data),
              replace = TRUE,
              prob = c(0.8, 0.2))
training <- data[ind==1,]
testing <- data[ind==2,]
```

PART C

In this part, we employed our inputs and output variables to determine the best prediction model using support vector machine, neural network and logistic regression analysis.

LOGISTIC REGRESSION

Selection of Number of PC's

For Logistics Regression, we selected 20 PC's out from 33 total PCs to work with. Overall performance of Support Vector with 20 carefully selected PC's came out to be **76.19%**

13 PC's were removed that are: PC3, PC4, PC8, PC9, PC11, PC12, PC13, PC14, PC17, PC22, PC25, PC31, and PC32 because of either of the two reasons:

1. they were diminishing the accuracy of the overall model or
2. had no contribution in improving the results

The selected PC's were PC1, PC2, PC5, PC6, PC7, PC10, PC15, PC16, PC18, PC19, PC20, PC21, PC23, PC24, PC26, PC27, PC28, PC29, PC30, and PC33

Working Steps

1. We fit a Logistics Regression model onto 20 PCs against the binary `satisfactionVariable` as our dependent/outcome variable
2. We make prediction on our `training` set
3. We tabulate the confusion Matrix on predictions on our `training` set
4. We make prediction on our `testing` set
5. We tabulate the confusion Matrix on predictions on our `testing` set
6. We calculate the Misclassification Error of the Logistics Regression model.
7. We calculate the accuracy of the Logistics Regression model

Working Code

```
#Training Logistic Regression
Fit_LG <- glm(SatisfactionVariable ~  PC1 + PC2 + PC5 + PC6 + PC7 + PC10
              + PC15 + PC16 + PC18 + PC19 + PC20 + PC21
              + PC23 + PC24 + PC26 + PC27 + PC28 + PC29 + PC30 + PC33
             , data = training, family = 'binomial')

summary(Fit_LG)

#Predictions on Logistic Regresssion and accuracy check
LG.prediction <- predict(Fit_LG, training, type = "response")
LG.prediction <- ifelse(LG.prediction > 0.5, 1,0)

tab1 <- table(predicted=LG.prediction, Actual= training$SatisfactionVariable)
tab1

LG.prediction2 <- predict(Fit_LG, testing, type = "response")
LG.prediction2 <- ifelse(LG.prediction2 > 0.5, 1,0)
tab2 <- table(predicted=LG.prediction2, Actual= testing$SatisfactionVariable)
tab2

#Misclassification error
Misc1asserror = 1 - sum(diag(tab2))/sum(tab2) #28.3% misclassification error is very high
Misc1asserror
#accuracy of Logistic Regression
Accuracy = 1 - Misc1asserror
Accuracy #Only 71.6% ... very low
```

Output of the Code

```
Call:
glm(formula = SatisfactionVariable ~ PC1 + PC2 + PC5 + PC6 +
    PC7 + PC10 + PC15 + PC16 + PC18 + PC19 + PC20 + PC21 + PC23 +
    PC24 + PC26 + PC27 + PC28 + PC29 + PC30 + PC33, family = "binomial",
    data = training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3359  -1.0201   0.5197   0.9028   2.1960

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.30228    0.05552   5.445 5.19e-08 ***
PC1          0.40582    0.02597  15.626  < 2e-16 ***
PC2          0.09029    0.03580   2.522 0.011669 *
PC5         -0.06669    0.04954  -1.346 0.178272
PC6         -0.19572    0.05257  -3.723 0.000197 ***
PC7          0.15137    0.05349   2.830 0.004656 **
PC10        -0.15362    0.05822  -2.639 0.008319 **
PC15         0.07366    0.06392   1.152 0.249206
PC16        -0.01588    0.06545  -0.243 0.808357
PC18         0.08060    0.06741   1.196 0.231829
PC19        -0.01883    0.06808  -0.277 0.782128
PC20        -0.03344    0.06977  -0.479 0.631729
PC21         0.04332    0.06888   0.629 0.529364
PC23         0.15932    0.07105   2.242 0.024936 *
PC24        -0.12644    0.07310  -1.730 0.083709 .
PC26         0.07444    0.07489   0.994 0.320261
PC27         0.08970    0.07506   1.195 0.232025
PC28         0.08981    0.07687   1.168 0.242656
PC29        -0.08053    0.07865  -1.024 0.305884
PC30        -0.02579    0.08025  -0.321 0.747908
PC33         0.07978    0.08495   0.939 0.347650
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2297.3  on 1677  degrees of freedom
Residual deviance: 1924.6  on 1657  degrees of freedom
AIC: 1966.6

Number of Fisher Scoring iterations: 4
```

The chi-square difference of the model is **372.7** (2297.3-1924.6) which means that it is a good fit.

Confusion Matrix

Confusion Matrix plotted for the Training set (1678 observations) is shown below (tab1).
According to it :

- ➔ 430 are correctly classified predictions as Satisfaction =0
- ➔ 747 are correctly classified predictions as Satisfaction =1
- ➔ However,
- ➔ 299 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 202 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab1
         Actual
predicted   0   1
        0 430 202
        1 299 747
```

Confusion Matrix plotted for the Testing set (399 observations) is shown below (tab2).
According to it :

- ➔ 115 are correctly classified predictions as Satisfaction =0
- ➔ 189 are correctly classified predictions as Satisfaction =1
- ➔ However,
- ➔ 57 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 38 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab2
         Actual
predicted   0   1
        0 115  38
        1  57 189
```

Misclassification Error

Misclassification error for the Logistics Regression model using the testing set came out to be **23.8%**

Following formula was used to calculate the misclassification error, whereby tab2 represents the confusion matrix of the testing data

```
Misclasserror = 1 - sum(diag(tab2))/sum(tab2)
```

Accuracy

Accuracy of SVM model came out to be **76.19%**

Following formula was used to calculate the accuracy

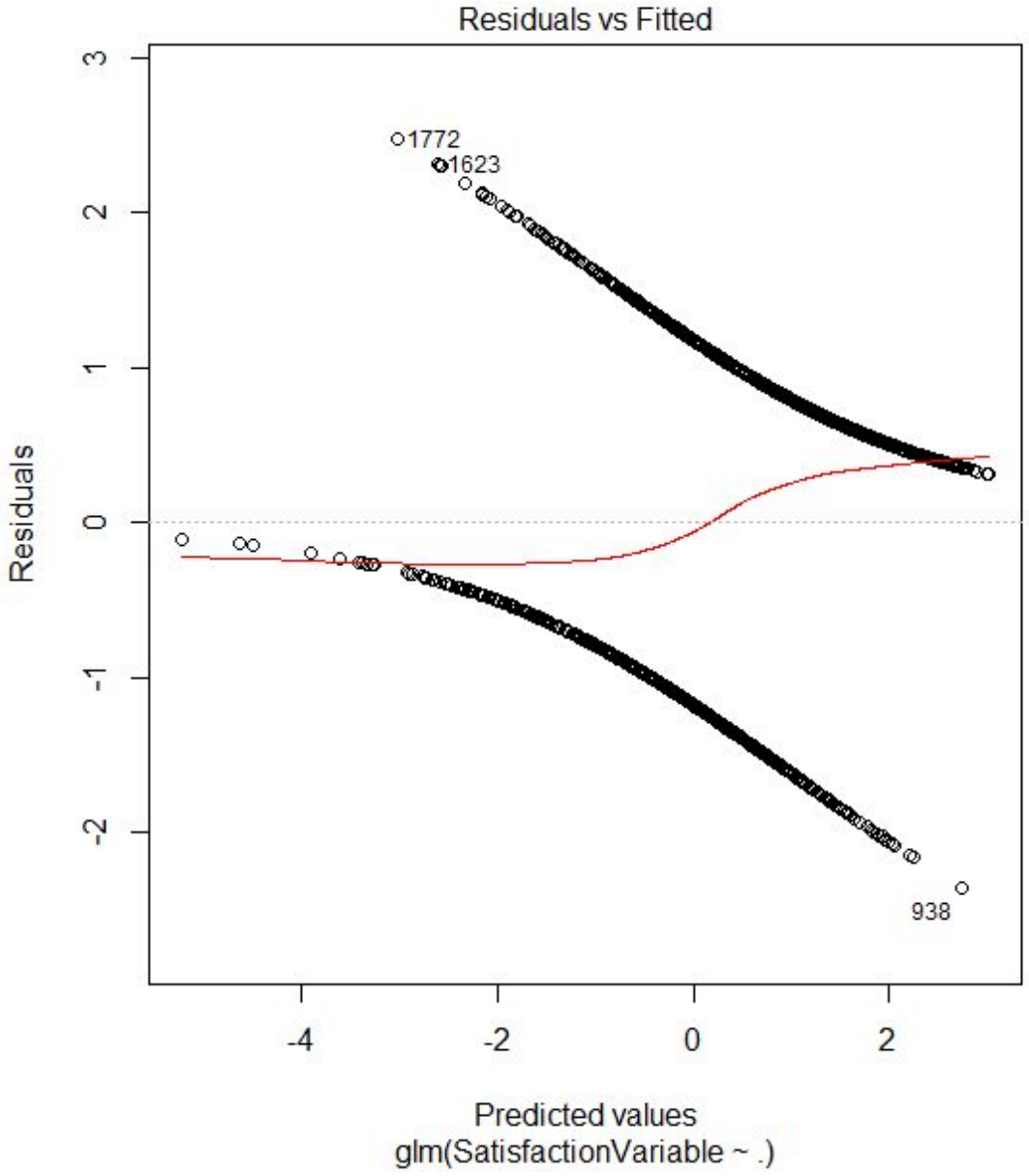
```
Accuracy = 1 - Misclasserror
```

Logistic Regression Plots

We shall make use of some plots to analyse Logistic Regression model fit

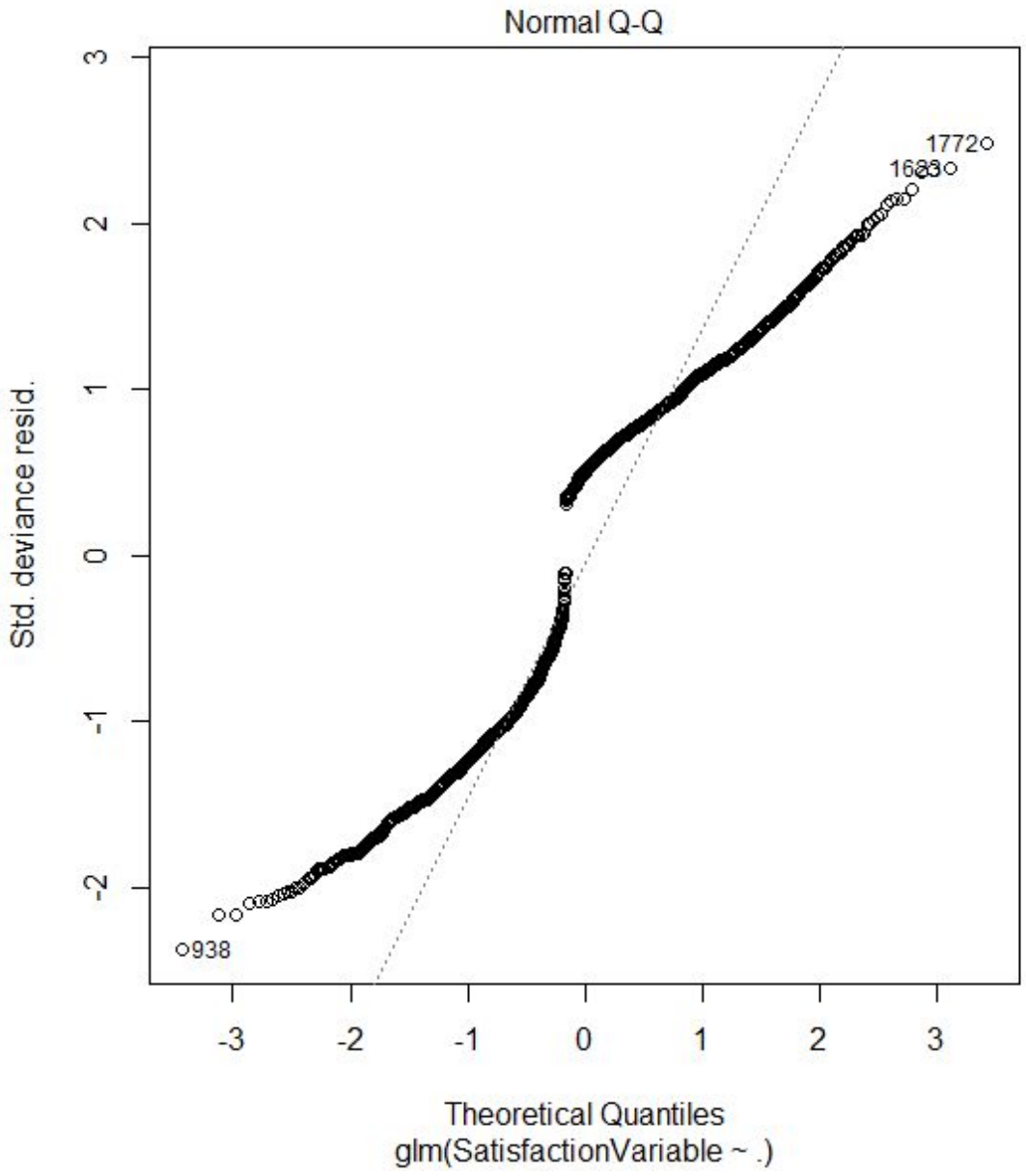
```
Fit_LG <- glm(SatisfactionVariable ~., data = training, family = 'binomial')
summary(Fit_LG)
#>>>>>>> #PLOT
plot(Fit_LG)
```

Residuals vs Fitted plot
The Residuals vs Fitted plot can help you see, for example, if there are curvilinear trends that you missed. But the fit of a logistic regression is curvilinear by nature, so you can have odd looking trends in the residuals with nothing amiss.


Residuals vs Fitted
Residuals
Predicted values
glm(SatisfactionVariable ~ .)
1772
1623
938
3
2
1
0
-1
-2
-4
-2
0
2

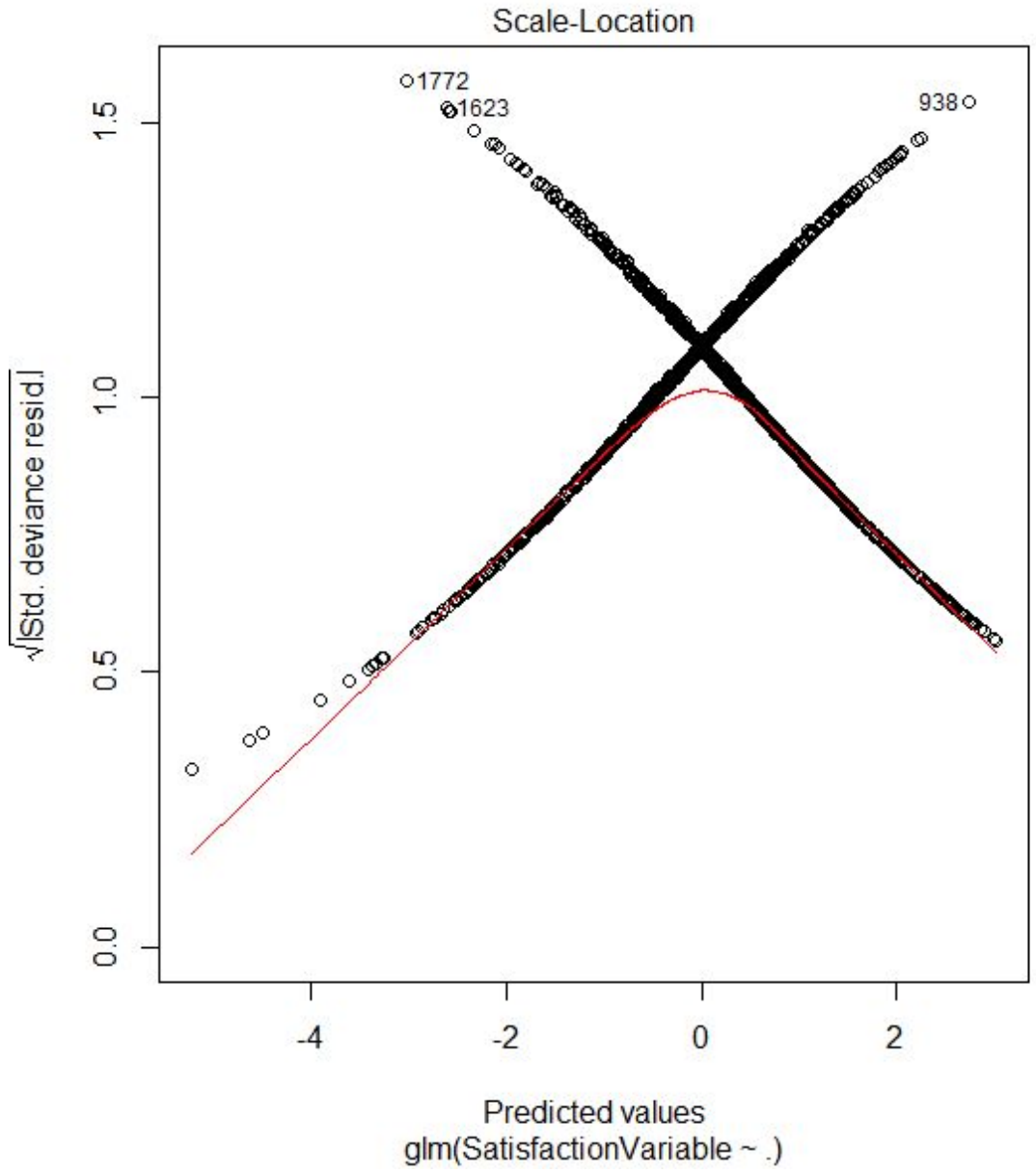
Normal Q-Q plot

The Normal Q-Q plot helps to detect if our residuals are normally distributed. But the deviance residuals don't have to be normally distributed for the model to be valid, so the normality / non-normality of the residuals doesn't necessarily tell us anything.



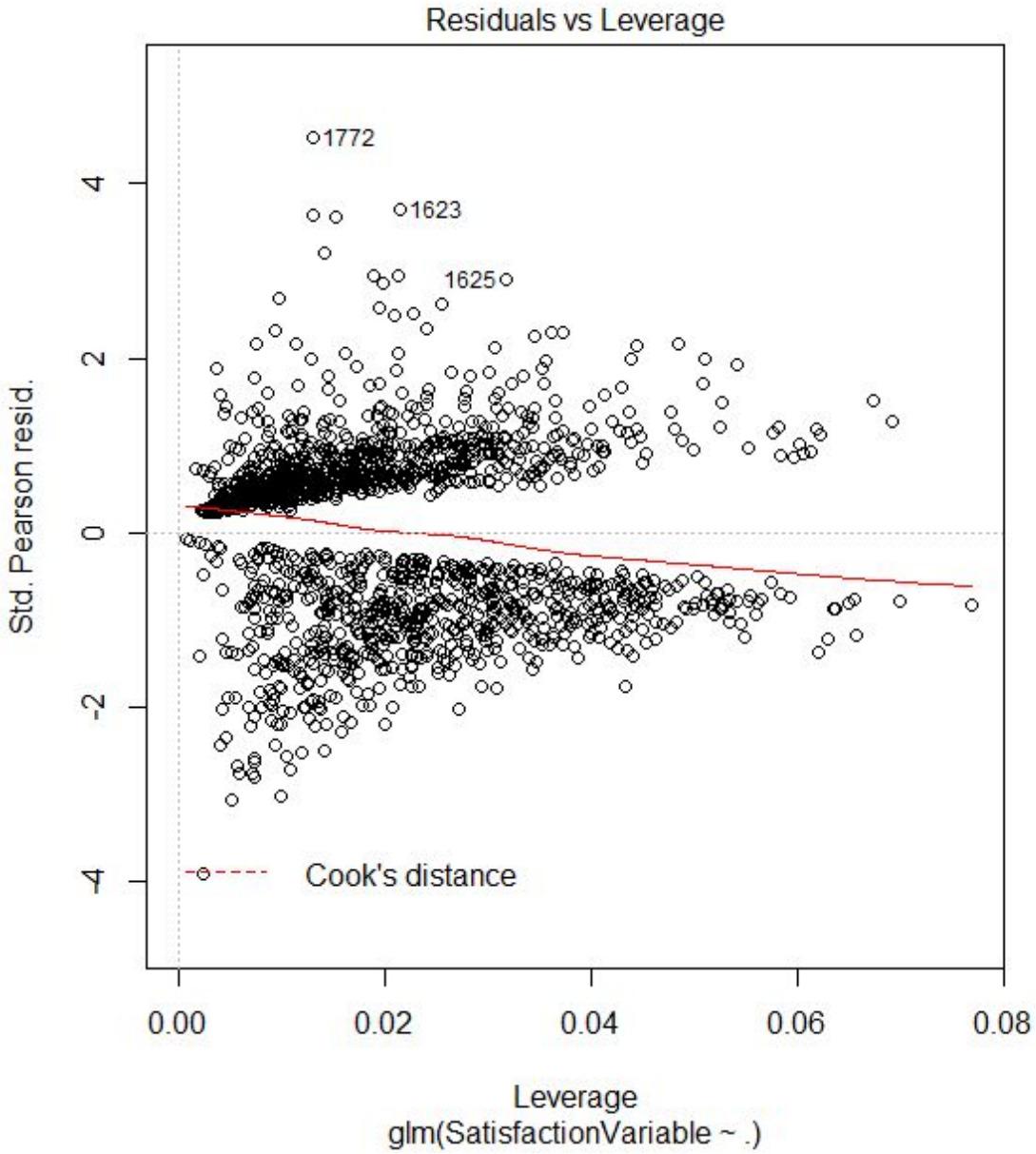
Scale-Location plot

The Scale-Location plot helps us identify heteroscedasticity. But logistic regression models are pretty much heteroscedastic by nature.



Residuals vs Leverage

The Residuals vs Leverage helps in identifying possible outliers in our case. But outliers in logistic regression don't necessarily manifest in the same way as in linear regression, so this plot may or may not be helpful in identifying them.



SUPPORT VECTOR MACHINE

Selection of Number of PC's

For Support Vector Machines, we selected 22 PC's out from 33 total PCs to work with. Overall performance of Support Vector with 22 carefully selected PC's came out to be **76.99%**

11 PC's were removed that are: PC3,PC16,PC21,PC6,PC4,PC13,PC17,PC23,PC25,PC27,PC32 because of either of the two reasons:

1. they were diminishing the accuracy of the overall model or
2. had no contribution in improving the results

The selected PC's were
PC1,PC2,PC5,PC7,PC8,PC9,PC10,PC11,PC12,PC14,PC15,PC18,PC19,PC20,PC22,PC24,PC26,PC28,PC29,PC30,PC31 and PC33

Kernel Selection

Out from the three Kernel offered by the r package of SVM i.e "Radial", "Linear" and "Polynomial", for our model the best results were offered by "**Radial"** kernel with the accuracy of 76.99%

The accuracy of all three Kernels are as below:

Polynomial: 64.1%
Linear: 71.6%
Radial: 76.99%

Working Steps

1. First we included the library of SVM in r which is `library(e1071)`
2. We fit SVM model onto 22 PCs against the binary `SatisfactionVariable` as our dependent/outcome variable
3. We make prediction on our `training` set
4. We tabulate the confusion Matrix on predictions on our `training` set
5. We make prediction on our `testing` set

6. We tabulate the confusion Matrix on predictions on our `testing` set
7. We calculate the Misclassification Error of the SVM model.
8. We calculate the accuracy of the SVM model

Confusion Matrix

Confusion Matrix plotted for the Training set (1678 observations) is shown below(tab1).
According to it :

- ➔ 528 are correctly classified predictions as Satisfaction =0
- ➔ 886 are correctly classified predictions as Satisfaction =1
- ➔ However,
- ➔ 201 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 63 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab1
         Actual
predicted   0   1
        0 528  63
        1 201 886
```

Confusion Matrix plotted for the Testing set (399 observations) is shown below (tab2).
According to it :

- ➔ 124 are correctly classified predictions as Satisfaction =0
- ➔ 183 are correctly classified predictions as Satisfaction =1
- ➔ However,
- ➔ 48 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 44 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab2
         Actual
predicted   0   1
        0 124  44
        1  48 183
```

Misclassification Error

Misclassification error for the SVM model using the testing set came out to be **23.05%**

Following formula was used to calculate the misclassification error, whereby tab2 represents the confusion matrix of the testing data

```
Misc1asserror = 1 - sum(diag(tab2))/sum(tab2)
```

Accuracy

Accuracy of SVM model came out to be 76.9% ~ **77%**

Following formula was used to calculate the accuracy

```
Accuracy = 1 - Misc1asserror
```

Working Code

```
#Training SVM
library(e1071)
Fit_SVM <- svm(SatisfactionVariable ~PC1+PC2+PC5+PC7+PC8+PC9+PC10+PC11+PC12+PC14+PC15+PC18
               +PC19+PC20+PC22+PC24+PC26+PC28+PC29+PC30+PC31+PC33,
               data = training, kernel = "radial", type = "C-classification")
summary(Fit_SVM)

#Prediction on traing set
SVM.prediction <- predict(Fit_SVM, training)

#Confusion matrix on training set
tab1 <- table(predicted=SVM.prediction, Actual= training$SatisfactionVariable)
tab1

#Prediction on testing set
SVM.prediction <- predict(Fit_SVM, testing)

#Confusion matrix on testing set
tab2 <- table(predicted=SVM.prediction, Actual= testing$SatisfactionVariable)
tab2

#Misclassification error
Misc1asserror = 1 - sum(diag(tab2))/sum(tab2)
Misc1asserror #23.05%

#accuracy of SVM
Accuracy = 1 - Misc1asserror
Accuracy  #76.9%
```

NEURAL NETWORKS

Selection of Number of PC's

For Neural Networks, we selected 17 PC's out from 33 total PCs to work with. Overall performance of Neural Networks with 17 carefully selected PC's came out to be **75.6%**

16 PC's were removed that are: PC16,PC21,PC6,PC4,PC13,PC17,PC23,PC25,PC27,PC32 because of either of the two reasons:

1. they were diminishing the accuracy of the overall model or
2. had no contribution in improving the results

The selected PC's were

PC1,PC2,PC7,PC8,PC9,PC12,PC15,PC18,PC19,PC20,PC22,PC24,PC26,PC28,PC29,PC33 and PC16

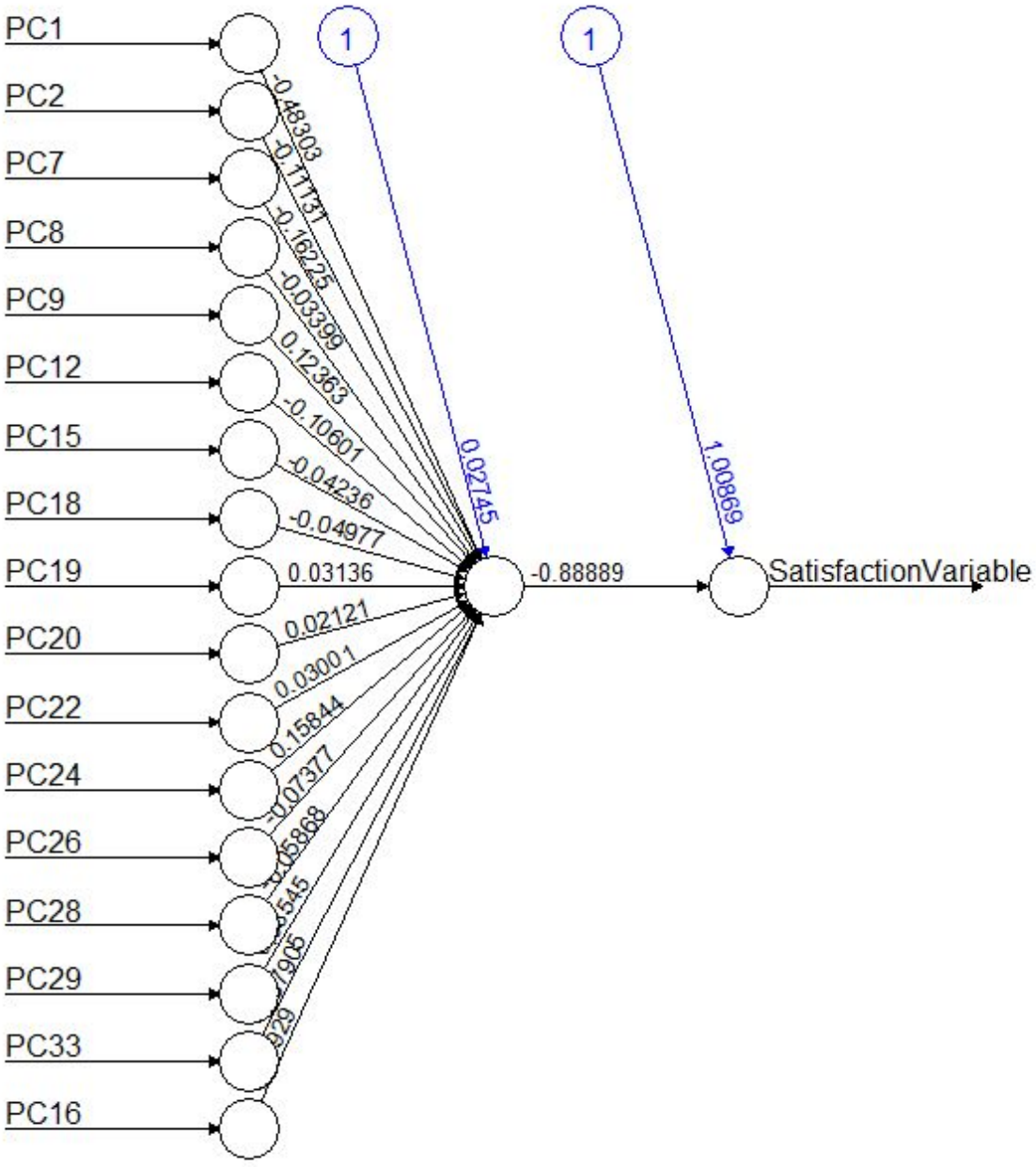
Selection of Hidden Layers

For neural networks, we selected **Hidden Layers=1** as this gave us the maximum accuracy of 75.6%

With increasing layers, the accuracy fall down as shown below:

Hidden Layers =1, Accuracy =75.6%
Hidden Layers =2, Accuracy = 70.4%
Hidden Layers = 3, Accuracy = 66.4%

Neural Network Plot



Working Steps

1. First we will include Neural Network Library in r `library(neuralnet)`

2. We fit NN model onto 17 PCs against the binary `SatisfactionVariable` as our dependent/outcome variable
3. We make prediction on our `training` set
4. The resultant prediction(probability) if greater than the threshold value of **0.5** is labelled as '1' otherwise '0'
5. We tabulate the confusion Matrix on predictions on our `training` set
6. We make prediction on our testing set
7. The resultant prediction(probability) if greater than the threshold value of **0.5** is labelled as '1' otherwise '0'
8. We make prediction on our `testing` set
9. We tabulate the confusion Matrix on predictions on our `testing` set
10. We calculate the Misclassification Error of the NN model.
11. We calculate the accuracy of the NN model

Confusion Matrix

Confusion Matrix plotted for the Training set (1678 observations) is shown below(tab1).
According to it :

- ➔ 446 are correctly classified predictions as Satisfaction =0
- ➔ 712 are correctly classified predictions as Satisfaction =1
- ➔ 283 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 237 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab1
         Actual
predicted   0   1
        0 446 237
        1 283 712
```

Confusion Matrix plotted for the Testing set (399 observations) is shown below (tab2).
According to it :

- ➔ 121 are correctly classified predictions as Satisfaction =0
- ➔ 181 are correctly classified predictions as Satisfaction =1
- ➔ 51 predictions are misclassified as Satisfaction= 1 which in actual were Satisfaction = 0 and,
- ➔ 46 predictions are misclassified as Satisfaction= 0 which in actual were Satisfaction = 1

```
> tab2
         Actual
predicted   0   1
        0 121  46
        1  51 181
```

Misclassification Error

Misclassification error for the NN model using the testing set came out to be **24.3%**

Following formula was used to calculate the misclassification error, whereby tab2 represents the confusion matrix of the testing data

```
Misc1asserror = 1 - sum(diag(tab2))/sum(tab2)
```

Accuracy

Accuracy of NN model came out to be **75.6%**

Following formula was used to calculate the accuracy

```
Accuracy = 1 - Misc1asserror
```

Working Code

```
#Training Neural Network
library(neuralnet)
Fit_NN <- neuralnet(SatisfactionVariable ~PC1+PC2+PC7+PC8+PC9+PC12+PC15+PC18
                    +PC19+PC20+PC22+PC24+PC26+PC28+PC29+PC33+PC16,
                    data = training, hidden = 1,err.fct = "sse",
                    linear.output = T)
summary(Fit_NN)
plot(Fit_NN)

#Prediction on Training set
train <- training[c("PC1","PC2","PC7","PC8","PC9","PC12","PC15","PC18",
                    "PC19","PC20","PC22","PC24","PC26","PC28","PC29","PC33","PC16")]
NN.prediction <- compute(Fit_NN, train)
p1 <-NN.prediction$net.result
pred1 <- ifelse(p1>0.5,1,0)

#Confusion Matrix of Training set
tab1 <- table(predicted=pred1, Actual= training$SatisfactionVariable)

#Prediction on Training set
test <- testing[c("PC1","PC2","PC7","PC8","PC9","PC12","PC15","PC18",
                  "PC19","PC20","PC22","PC24","PC26","PC28","PC29","PC33","PC16")]
NN.prediction <- compute(Fit_NN, test)
p2 <-NN.prediction$net.result
pred2 <- ifelse(p2>0.5,1,0)

#Confusion Matrix of Training set
tab2 <- table(predicted=pred2, Actual= testing$SatisfactionVariable)

#Misclassification error
Misc1asserror = 1 - sum(diag(tab2))/sum(tab2) #24.3%

#Accuracy of NN
Accuracy = 1 - Misc1asserror
Accuracy # 75.6%
```

PART D

Conclusion

The following table shows the accuracy of each prediction model:

Prediction Model	Misclassification Error%	Accuracy%
Logistics Regression	23.8	76.19
Support Vector Machine	23.05	76.99
Neural Network	24.3	75.60

We conclude from the above table that while the three prediction models have very similar accuracy percentages, we will choose **Support Vector Machine** as the best prediction model, since it has the **highest** accuracy(76.9%) and **lowest** Misclassification error (23.05%) compared to the other two models i.e Logistic Regression and Neural Network.

Grading

	Kanza Haider	Qura ul Aain	Noran Islam	Nuha Hashem
Kanza Haider		90	90	90
Qurat ul Aain				
Noran Islam				
Nuha Hashem				