Predicting Whether The Neighborhood Will Be At Risk For High Crime Levels Using Logistics Regression

Umer Farooq

2023-10-28

Introduction:

In this homework assignment, you will explore, analyze and model a data set containing information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or not (0).

Your objective is to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels. You will provide classifications and probabilities for the evaluation data set using your binary logistic regression model. You can only use the variables given to you (or variables that you derive from the variables provided).

1. DATA EXPLORATION:

Loading The Training Data set:

The following code chunk load the training data set from the github repository where the data file is stored:

```
training <- read_csv('https://raw.githubusercontent.com/Umerfarooq122/predicting-whether-the-neighborho
```

Let's display the first few rows of data set to check if we have the data set loaded correctly:

```
knitr::kable(head(training))
```

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	19.58	0	0.605	7.929	96.2	2.0459	5	403	14.7	3.70	50.0	1
0	19.58	1	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4	1
0	18.10	0	0.740	6.485	100.0	1.9784	24	666	20.2	18.85	15.4	1
30	4.93	0	0.428	6.393	7.8	7.0355	6	300	16.6	5.19	23.7	0
0	2.46	0	0.488	7.155	92.2	2.7006	3	193	17.8	4.82	37.9	0
0	8.56	0	0.520	6.781	71.3	2.8561	5	384	20.9	7.67	26.5	0

Let's quickly peek into the number of observation and variable we have available in the training data set.

```
dim(training)
```

```
## [1] 466  13
```

We have in total 13 columns available out of which 12 columns are independent variable and one column by the name of `target` is the dependent or the target variable. The data set contains 466 observations in its raw form. Before moving on to processing the data set let's get and in overview of all the column in a descriptive summary section.

Descriptive Summary Statistics:

Let's dive into thew summary statistics of the all the varaible we have in the data set. Below code chunk shows us the summary of each column.

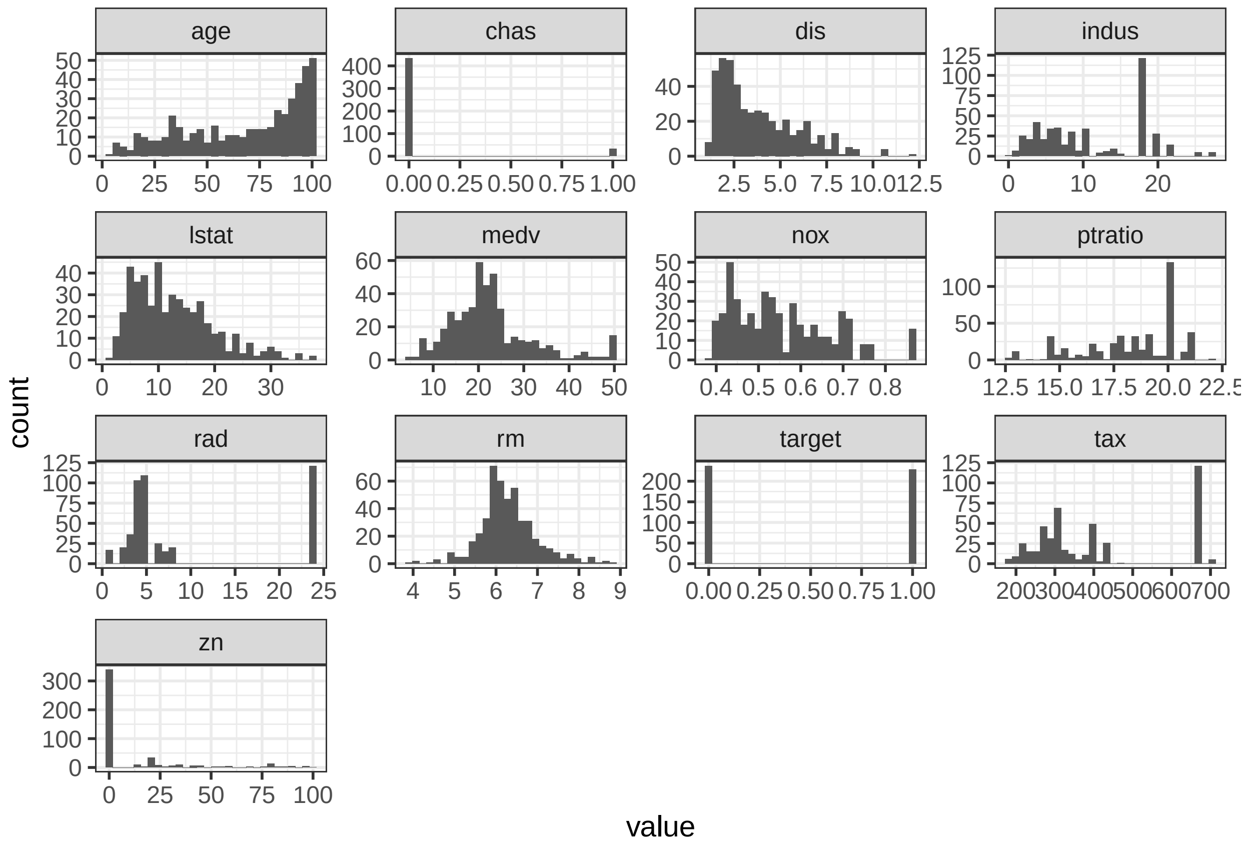
```r
knitr::kable(describe(training))
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
zn	1	466	11.5772532	23.3646511	0.00000	5.3542781	0.0000000	0.0000	100.0000	100.0000	2.1768152	3.8135765	1.0823466
indus	2	466	11.1050215	6.8458549	9.69000	10.9082353	9.3403800	0.4600	27.7400	27.2800	0.2885450	-1.2432132	0.3171281
chas	3	466	0.0708155	0.2567920	0.00000	0.0000000	0.0000000	0.0000	1.0000	1.0000	3.3354899	9.1451310	0.0118957
nox	4	466	0.5543105	0.1166667	0.53800	0.5442684	0.1334340	0.3890	0.8710	0.4820	0.7463281	-0.0357736	0.0054045
rm	5	466	6.2906738	0.7048513	6.21000	6.2570615	0.5166861	3.8630	8.7800	4.9170	0.4793202	1.5424378	0.0326516
age	6	466	68.3675966	28.3213784	77.15000	70.9553476	30.0226500	2.9000	100.0000	97.1000	-0.5777075	-1.0098814	1.3119625
dis	7	466	3.7956929	2.1069496	3.19095	3.5443647	1.9144814	1.1296	12.1265	10.9969	0.9988926	0.4719679	0.0976026
rad	8	466	9.5300429	8.6859272	5.00000	8.6978610	1.4826000	1.0000	24.0000	23.0000	1.0102788	-0.8619110	0.4023678
tax	9	466	409.5021459	167.9000887	334.50000	401.5080214	104.5233000	187.0000	711.0000	524.0000	0.6593136	-1.1480456	7.7778214
ptratio	10	466	18.3984979	2.1968447	18.90000	18.5970588	1.9273800	12.6000	22.0000	9.4000	-0.7542681	-0.4003627	0.1017669
lstat	11	466	12.6314592	7.1018907	11.35000	11.8809626	6.0720020	1.7300	37.9700	36.2400	0.9055864	0.5033688	0.3289887
medv	12	466	22.5892704	9.2396814	21.20000	21.6304813	6.0045300	5.0000	50.0000	45.0000	1.0766920	1.3737825	0.4280200
target	13	466	0.4914163	0.5004636	0.00000	0.4893048	0.0000000	0.0000	1.0000	1.0000	0.0342293	-2.0031131	0.0231835

From the summary above with the help of `describe()` function we can see the min, max, median, standard deviation and skewness of each column. Alongside that we also get a generic idea about the type of data. If we look at columns like `target` and `chas` we clearly see that they are represented as int type but in actual those are factor data type which needs to be corrected later at some stage.

Now let's look at the distribution of each variable in the data set using histogram. Since we got 12 independent columns/variables so it will be tedious to look at each one of them separately so I will try to plot all the histogram together and get a holistic view of entire data set.

```r
training_long <- training %>%                          # Apply pivot_longer function
  pivot_longer(colnames(training)) %>%
  as.data.frame()
ggp1 <- ggplot(training_long, aes(x = value)) +     # Draw each column as histogram
  geom_histogram() +
  facet_wrap(~ name, scales = "free")+theme_bw()
ggp1
```

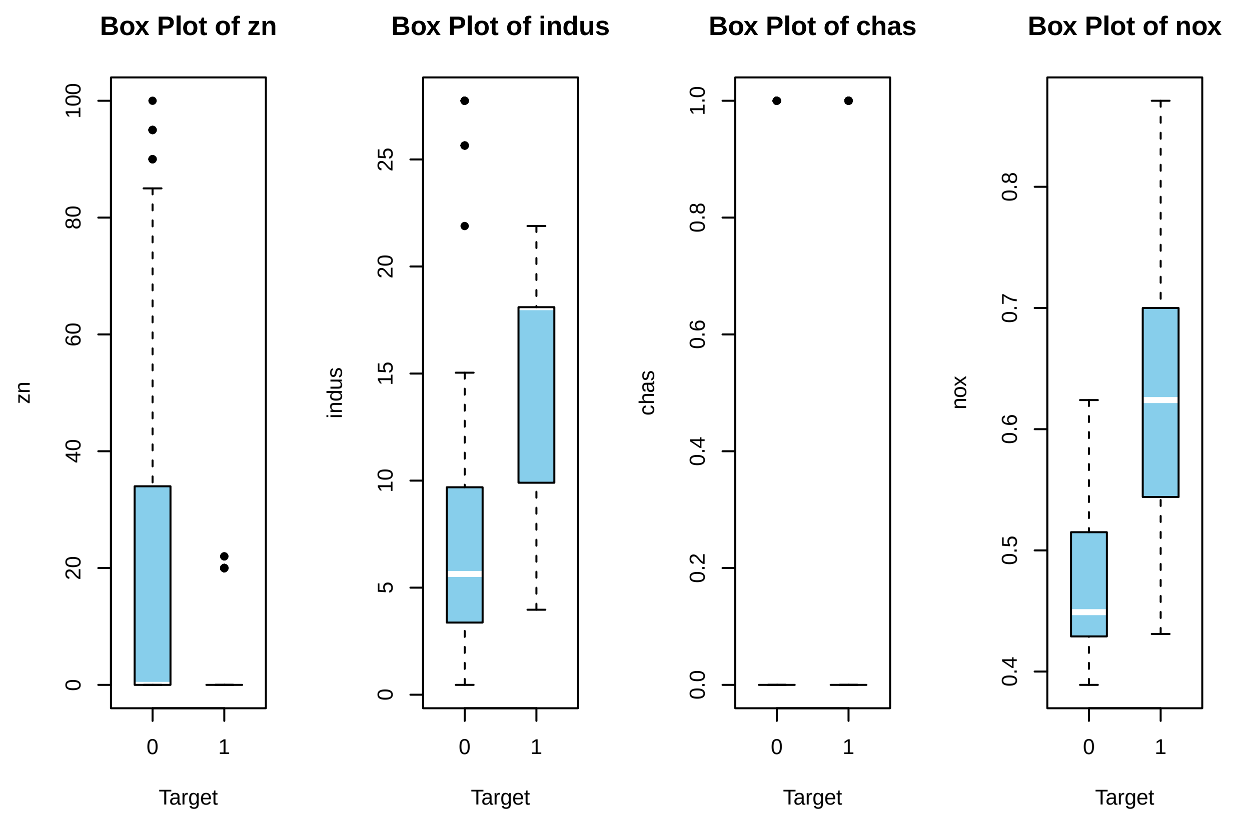


The distribution does not look very ideal. Most of the variable have skewed distribution with outliers. Let confirm the outliers with the box plot of all the variable. I will try to plot all the variable against our `target` variable and see if the variable would be a good predictor or no. Let's look at the first four variables:

```r
# Define the variables for the box plots
variables <-c("zn","indus","chas","nox")

# Set up the plotting layout
par(mfrow = c(1, length(variables)))

# Create the box plots
for (var in variables) {
  boxplot(get(var) ~ target, data = training,
          main = paste("Box Plot of", var),
          xlab = "Target",
          ylab = var,
          col = "skyblue",
          border = "black",
          notch = FALSE,
          notchwidth = 0.5,
          medcol = "white",
          whiskcol = "black",
          boxwex = 0.5,
          outpch = 19,
          outcol = "black")
}
```



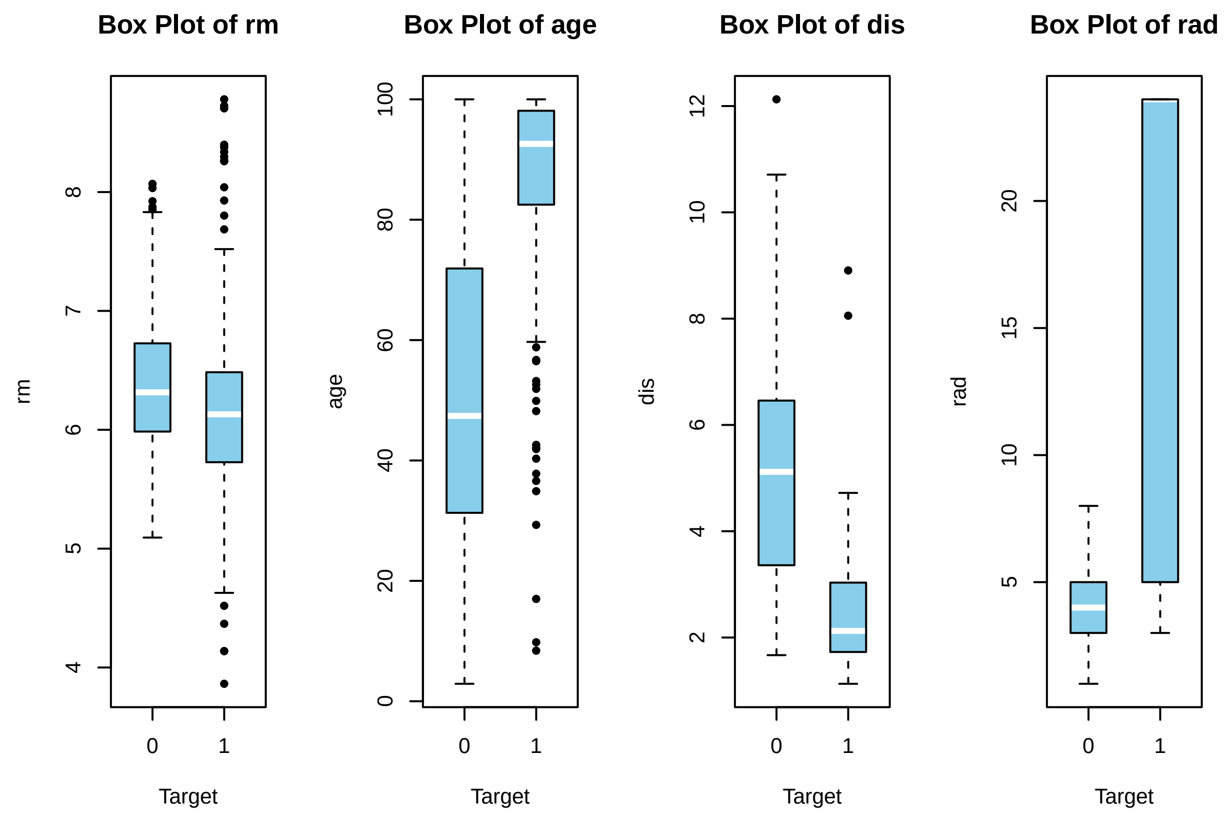
```r
# Reset the plotting layout
#par(mfrow = c(1, 1))
```

As we can see that variable `chas` might not be that useful in terms of predicting the `target` so we can easily ignore that in the upcoming models.Let's check the next four columns

```r
variables <-c('rm','age','dis','rad')

# Set up the plotting layout
par(mfrow = c(1, length(variables)))

# Create the box plots
for (var in variables) {
  boxplot(get(var) ~ target, data = training,
          main = paste("Box Plot of", var),
          xlab = "Target",
          ylab = var,
          col = "skyblue",
          border = "black",
          notch = FALSE,
          notchwidth = 0.5,
          medcol = "white",
          whiskcol = "black",
          boxwex = 0.5,
          outpch = 19,
          outcol = "black")
}
```



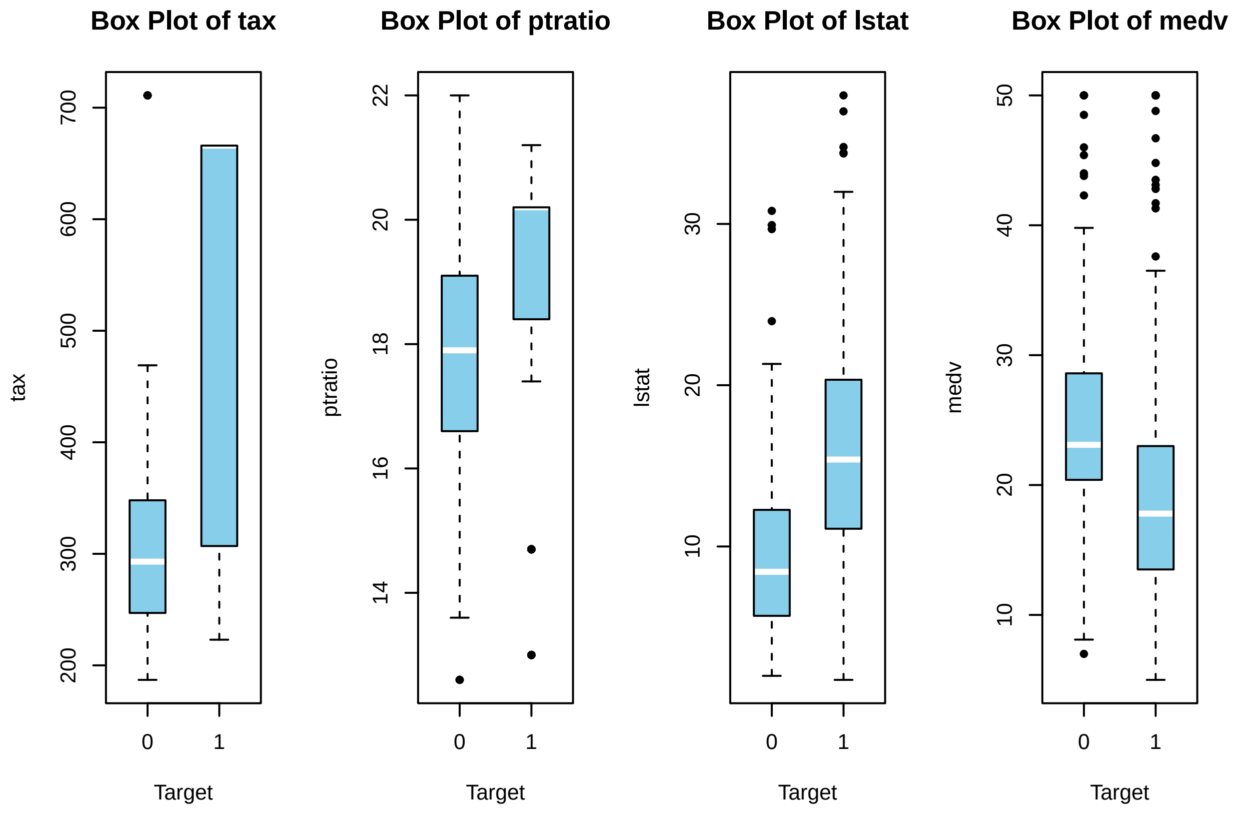
```
# Reset the plotting layout
par(mfrow = c(1, 1))
```

Apart from a lot of outlier is `dis`, `age`, and `rm` everything looks okay-ish. We already had an idea about outliers and skewness when we were dealing with histograms. let do the plotof final four columns:

```
variables <-c('tax','ptratio','lstat','medv')

# Set up the plotting layout
par(mfrow = c(1, length(variables)))

# Create the box plots
for (var in variables) {
  boxplot(get(var) ~ target, data = training,
          main = paste("Box Plot of", var),
          xlab = "Target",
          ylab = var,
          col = "skyblue",
          border = "black",
          notch = FALSE,
          notchwidth = 0.5,
          medcol = "white",
          whiskcol = "black",
          boxwex = 0.5,
          outpch = 19,
          outcol = "black")
}
```



```
# Reset the plotting layout
par(mfrow = c(1, 1))
```

So the plots look good. let's check out on if there are any missing values in the data set that needs to be imputed.

```
knitr::kable(colSums(is.na(training)))
```

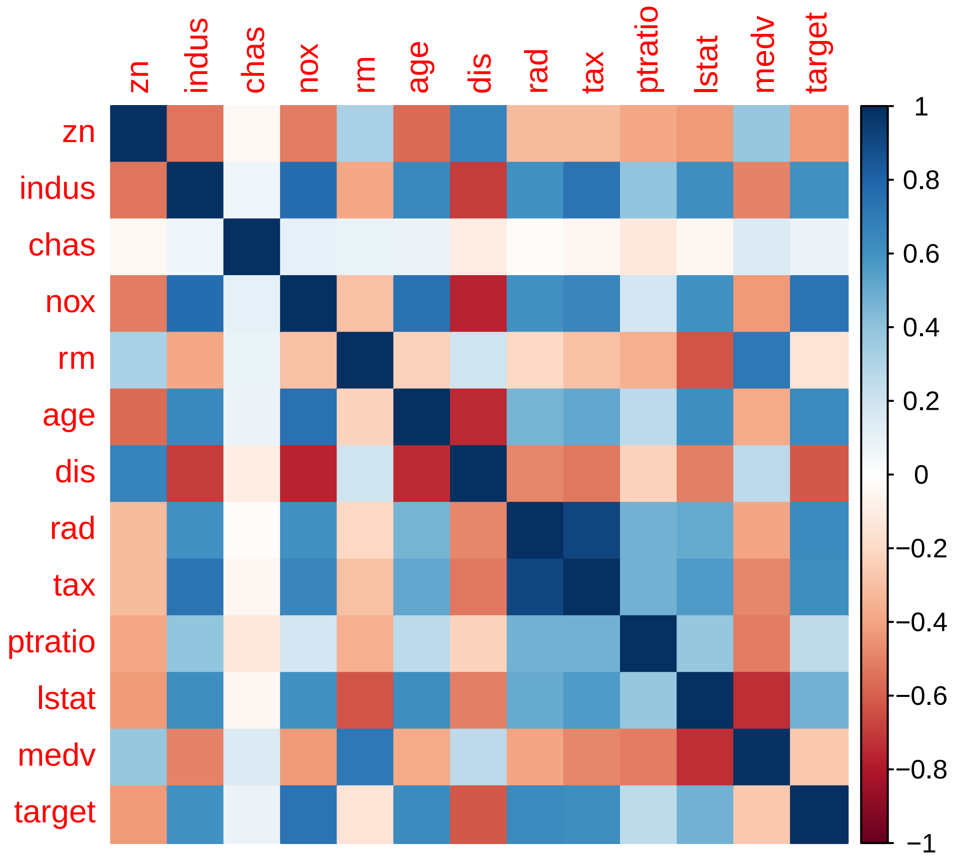
	x
zn	0
indus	0
chas	0
nox	0
rm	0
age	0
dis	0
rad	0

	x
tax	0
ptratio	0
lstat	0
medv	0
target	0

Thankfully there are no missing values in the data set which is in itself a big relief. Before we jump into the data preparation let's check one last thing and that is multi-colinearity. Let's check the colinearity among the variables:

```r
cor_matrix <- round(cor(training),2)
```

```r
corrplot(cor_matrix, method = "color")
```



As we can see that a lot of variables have correlation between them, for instance, `indus` has strong correlation with `nox`,`age`,`rad`,`tax` e.t.c which may cause multi co-linearity so we have to fix that too in our data preparation section.

1. DATA PREPARATION:

Fixing Outliers:

In our data exploration we saw that columns like `indus` and `dis` has outliers so let's fix that first:

```r
attach(training)
training <- training[-c(3)]
training <- training[-which(target==0 & indus > 20),]
training <- training[-which(target==0 & dis > 11),]
training <- training[-which(target==1 & dis > 7.5),]
#detach(train)
```

In the process of imputing outliers we did lose some observations and we can confirm it by:

```r
dim(training)
```

```
## [1] 452  12
```

We can also fix other columns too but I'm afraid we might lose valuable observations in the process so I will leave those columns as is.

The second problem that we found in the data set during our data exploration is that there was multi co-linearity among the variable and that could easily effect our final outcome using logistics regression models. In order to deal with mutli co-linearity we will carry out Principal Components Analysis (PCA). ### **Principal Component Analysis:**

Principal Component Analysis (PCA) is a dimensionality reduction technique and a powerful tool used in various fields, including statistics, machine learning, and data analysis.It will not only help us to mitigate multi co-linearity but will also helps us in reducing the number of features used in the models too (feature engineering). Below code chunk will create the principal components for the columns

```r
pc <- prcomp(training[-12],
             center = TRUE,
             scale. = TRUE)
```

we can check all the principal components by simply printing them:

```r
print(pc)
```

```
## Standard deviations (1, .., p=11):
##  [1] 2.4637711 1.2524101 1.0378675 0.8924944 0.6380241 0.5181958 0.5126598
##  [8] 0.4434342 0.4217091 0.3650029 0.2036286
##
## Rotation (n x k) = (11 x 11):
##                PC1         PC2         PC3          PC4         PC5
## zn       0.2679288  0.10759280 -0.39900450  0.566765361 -0.28251909
## indus   -0.3580688 -0.12009734 -0.07439193  0.068431314  0.35473767
## nox     -0.3387193 -0.26253302  0.09730582  0.260634026  0.10588232
## rm       0.2123627 -0.53122459 -0.29664523 -0.177666642 -0.41188888
## age     -0.3124997 -0.27148277  0.23402310  0.007355941 -0.60976971
## dis      0.3127042  0.37869362 -0.19267732  0.113147169 -0.09021755
## rad     -0.3177259 -0.05820221 -0.53166548  0.051083593  0.04942761
## tax     -0.3385611 -0.03616035 -0.48030555  0.129585825  0.10914559
## ptratio -0.2171129  0.33851690 -0.29942946 -0.672738775 -0.23046722
## lstat   -0.3253668  0.23119138  0.17923843  0.280351478 -0.37402362
## medv     0.2759661 -0.48272902 -0.10371622 -0.105668485  0.16173331
##                  PC6         PC7         PC8         PC9        PC10
## zn       0.247766590 -0.46951720  0.18470326 -0.10497843 -0.144996058
## indus   -0.160279312 -0.41177826  0.13019840  0.67175482 -0.089694158
## nox     -0.250942821 -0.30291997  0.01239104 -0.45644587  0.602965732
## rm      -0.509760687  0.10995399  0.31433830  0.11165363  0.005364143
## age      0.153400294 -0.18196635 -0.55367610  0.13587791 -0.139438163
## dis     -0.327086882  0.02654934 -0.48025808  0.34975978  0.489831126
## rad      0.101256320  0.41450931 -0.18880904 -0.15348665  0.020047006
## tax      0.006340914  0.13313005 -0.15088837  0.03101624 -0.145602571
## ptratio  0.157512791 -0.36981460  0.16381208 -0.08361263  0.217139514
## lstat    0.170099332  0.38110386  0.47588418  0.29980171  0.309855875
## medv     0.628919241  0.02794089 -0.03239793  0.23394148  0.428012730
##                PC11
## zn      -0.09164616
## indus   -0.22317149
## nox      0.03064654
## rm       0.01537939
## age     -0.02990973
## dis      0.01167255
## rad     -0.60519605
## tax      0.74965121
## ptratio  0.01922305
## lstat    0.05948584
## medv     0.08626481
```

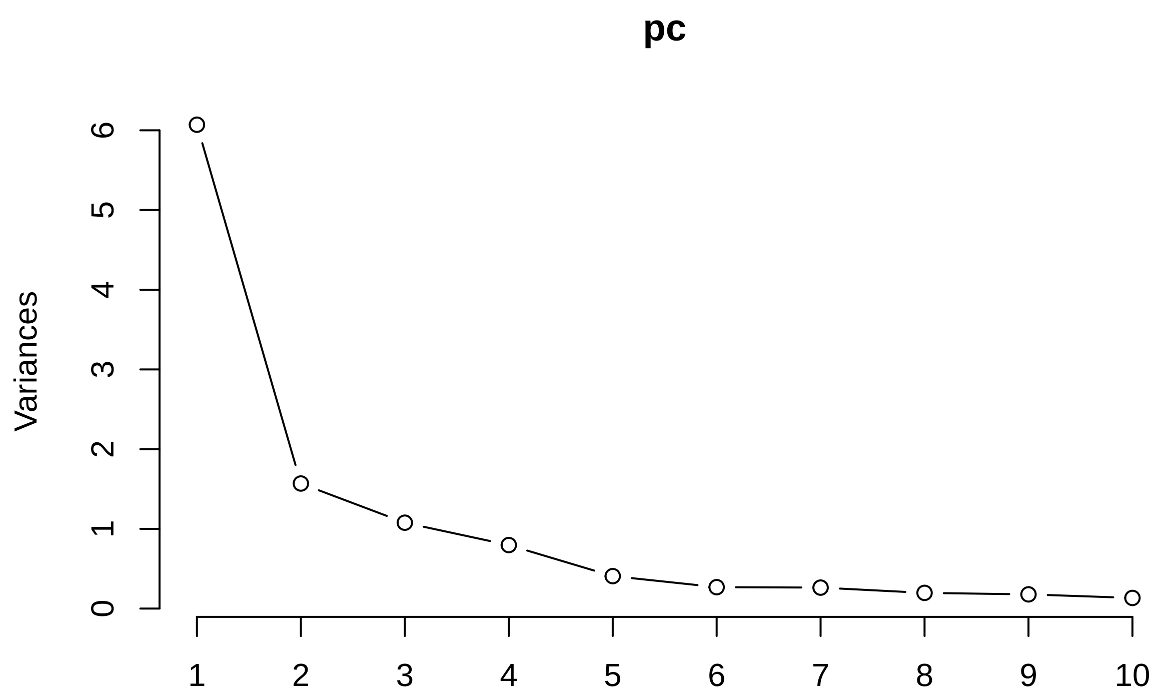
We can check the summary of all the principal components too.

```
summary(pc)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4638 1.2524 1.03787 0.89249 0.63802 0.51820 0.51266
## Proportion of Variance 0.5518 0.1426 0.09792 0.07241 0.03701 0.02441 0.02389
## Cumulative Proportion  0.5518 0.6944 0.79235 0.86476 0.90177 0.92618 0.95008
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.44343 0.42171 0.36500 0.20363
## Proportion of Variance 0.01788 0.01617 0.01211 0.00377
## Cumulative Proportion  0.96795 0.98412 0.99623 1.00000
```

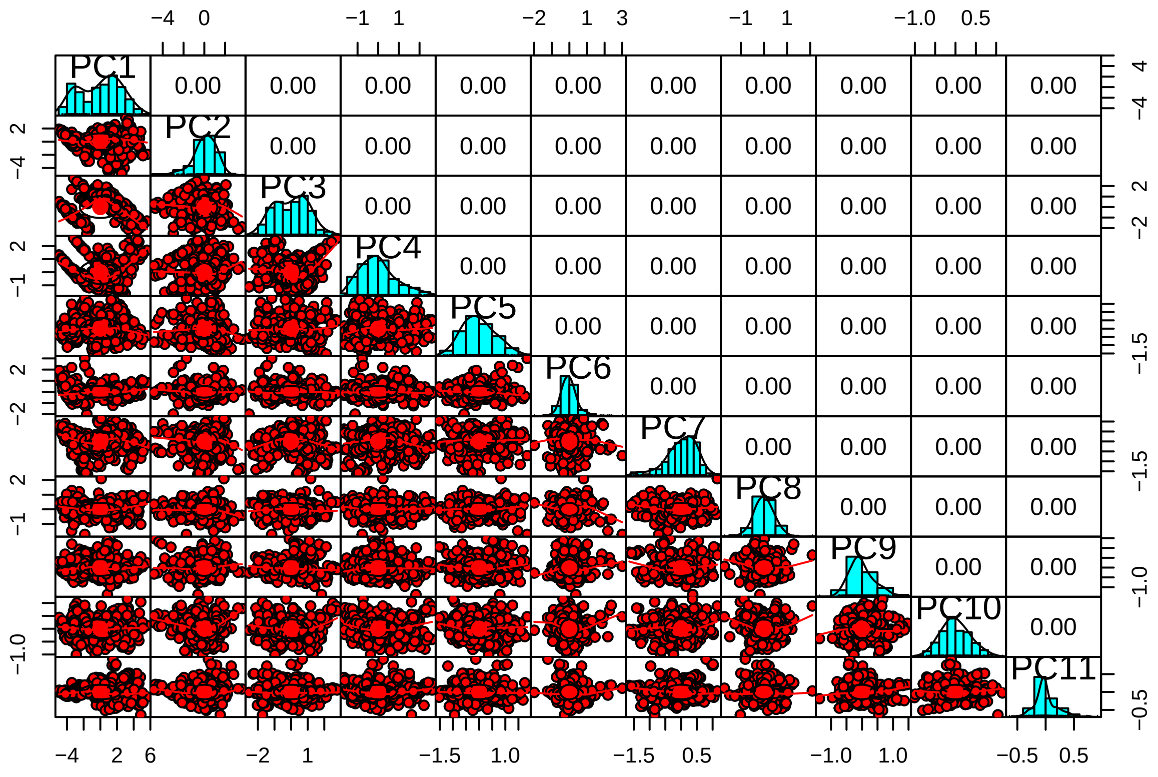
The most significant thing to look at the om the summary of principal components is the `Proportion of Variance` which shows the total variance in the data explained or represented by each component. For example Proportion of variance for PC1 is .5518 which means that PC1 accounts for 55.18% of variance in the data. Components up-to PC5 accounts for 90% of variance in the data which should be more than enough to create a model. We can also create scree plot to see how PCs are:

```r
plot(pc, type = 'lines')
```



We can also check the correlayion among the PCs now:

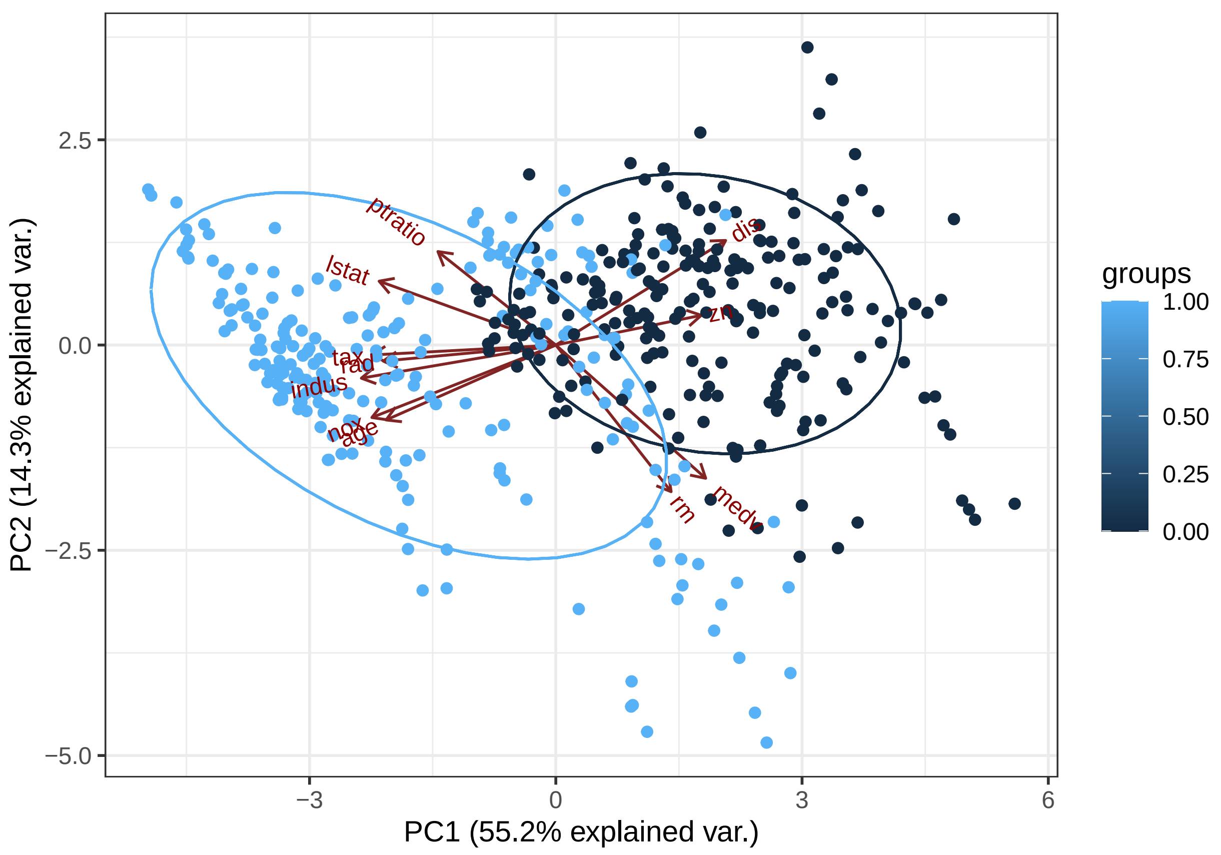
```r
pairs.panels(pc$x,
             gap = 0,
             bg = c('red','yellow','blue')[training$target],
             pch = 21)
```



As expected there is zero correlation among PCs which accounts for multi co-linearity. We can also check out the biplot as shown below

```
g <- ggbiplot(pc,
              obs.scale = 1,
              var.scale = 1,
              groups = training$target,
              ellipse = TRUE,
              ellipse.prob = .75)

g+theme_bw()
```



At this point with PCs and imputing some outliers we are confident enough to create models and evaluate their performance:

3. BUILD MODELS:

Before creating models let predict all the PCs values.

```
trg <- predict(pc, training[-12])
```

```
trg <-data.frame(trg, training[12])
```

```
knitr::kable(head(trg))
```

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	target
0.9174044	-4.4039304	0.1168961	-0.1567290	0.4768271	0.1306879	-0.4090493	-0.1793159	1.4021849	0.1864947	0.2021928	1
-2.9164842	-0.5824598	2.4974667	2.3659076	0.2842808	-0.4087548	-0.3886715	0.4902973	-0.0810863	0.6517563	0.0597166	1
-3.3744733	-0.6705932	-0.9950598	0.1285105	-0.4135539	-0.4148660	-0.0119231	-0.0030654	-0.2238829	0.2606384	-0.0211746	1
2.9595311	1.0383039	-0.5952193	0.3611073	0.9523646	-0.5726610	0.4142218	0.0512915	-0.1096659	0.0106480	-0.1220620	0

PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	target
1.8867360	-1.8825536	0.9245413	-1.3006277	-0.7931214	0.6308600	0.1297391	-0.3705188	-0.3266626	-0.0375316	-0.1380647	0
0.3614515	-0.4459198	-0.0419462	-1.6064798	-0.3048431	0.0887110	-0.4119418	0.2050227	-0.2397484	-0.0849197	0.3452703	0

After getting all the PCs now we can change the data type of `target` column from numeric to factor as shown below:

```r
trg$target <- as.factor(training$target)
trg$target <- relevel(trg$target, ref = "0")
```

Model 1:

In our first model we use first 6 PCs which accounts for approx 92% variation in the data. We will also use cross validation since we are using the training data in its entirety to create model so its better to go with techniques like cross validation.

```r
set.seed(121)
split <- createDataPartition(trg$target, p=0.80, list=FALSE)
partial_train <- trg[split, ]
validation <- trg[ -split, ]
modcv6 <- train(target ~PC1+PC2+PC3+PC4+PC5+PC6, data = partial_train,
              method = "glm", family = "binomial",
              trControl = trainControl(
                  method = "cv", number = 10,
                  savePredictions = TRUE),
              tuneLength = 5,
              preProcess = c("center", "scale"))
```

Model 2:

Let’s create another model only first 4 PCs

```r
modcv4 <- train(target ~PC1+PC2+PC3+PC4, data = partial_train,
              method = "glm", family = "binomial",
              trControl = trainControl(
                  method = "cv", number = 10,
                  savePredictions = TRUE),
              tuneLength = 5,
              preProcess = c("center", "scale"))
```

Model 3:

Model with only first 2 PCs

```r
modcv2 <- train(target ~PC1+PC2, data = partial_train,
              method = "glm", family = "binomial",
              trControl = trainControl(
                  method = "cv", number = 10,
                  savePredictions = TRUE),
              tuneLength = 5,
              preProcess = c("center", "scale"))
```

Model 4:

Model with all PCs

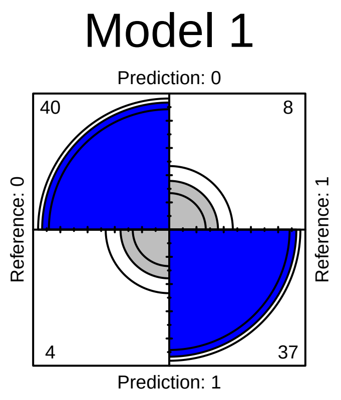
```
modcv <- train(target ~., data = partial_train,
              method = "glm", family = "binomial",
              trControl = trainControl(
                  method = "cv", number = 10,
                  savePredictions = TRUE),
              tuneLength = 5,
              preProcess = c("center", "scale"))
```

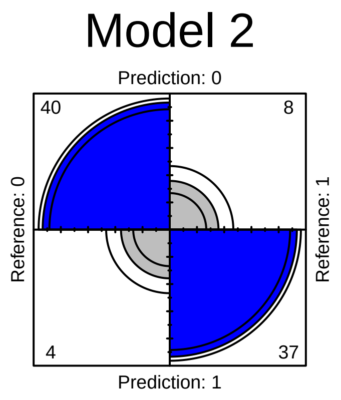
MODEL SELECTION:

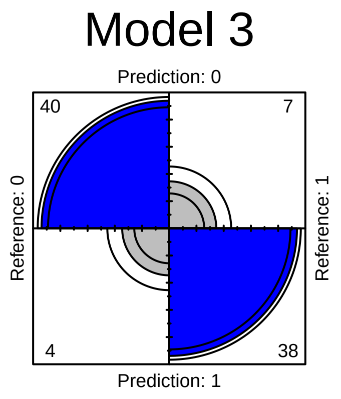
Selecting Models Based on Classification Metrics:

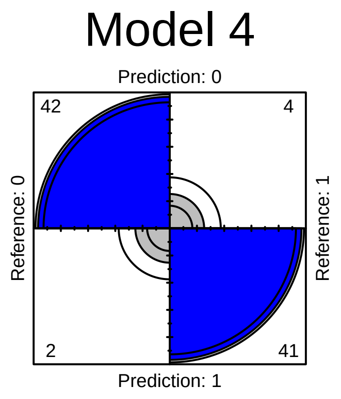
Now our models are and it is the time to pick the best one. We will tryp to look at different metrics like accuracy, classification error rate, precision, sensitivity, specificity, F1 score, AUC, and confusion matrix

```
preds1 <- predict(modcv6, newdata = validation)
preds2 <- predict(modcv4, newdata = validation)
preds3 <- predict(modcv2, newdata = validation)
preds4 <- predict(modcv, newdata = validation)
m1<- confusionMatrix(preds1, validation$target,
                        mode = "everything")
m2<- confusionMatrix(preds2, validation$target,
                        mode = "everything")
m3 <- confusionMatrix(preds3, validation$target,
                        mode = "everything")
m4 <- confusionMatrix(preds4, validation$target,
                        mode = "everything")
par(mfrow=c(2,2))
fourfoldplot(m1$table, color = c("gray", "blue"), main="Model 1")
fourfoldplot(m2$table, color = c("gray", "blue"), main="Model 2")
fourfoldplot(m3$table, color = c("gray", "blue"), main="Model 3")
fourfoldplot(m4$table, color = c("gray", "blue"), main="Model 4")
```









Surprisingly Model 3 which is based on only two PCs performs better than Model 1 and Model 2 when comes to confusion matrix. Let's look at other metrics too.

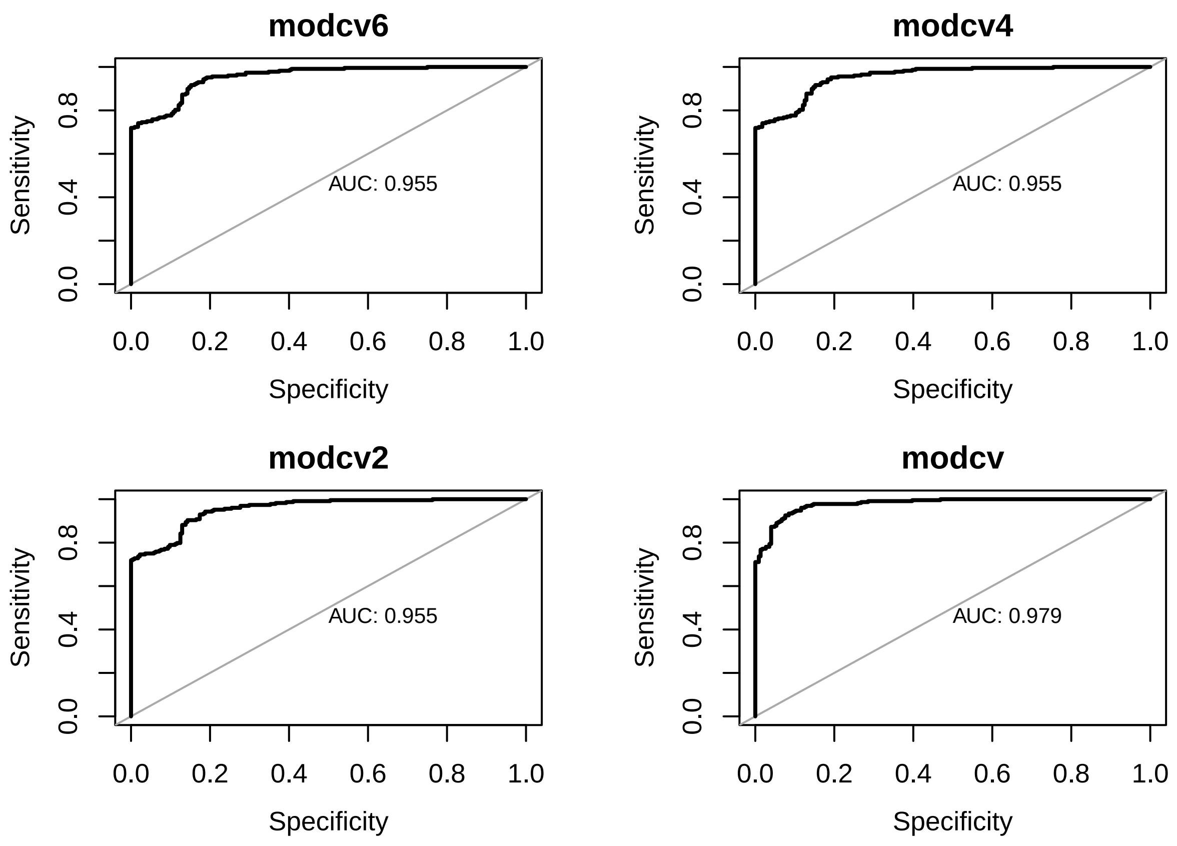
```r
eval <- data.frame(m1$byClass,
                   m2$byClass,
                   m3$byClass,
                   m4$byClass)
eval <- data.frame(t(eval))

eval <- dplyr::select(eval, Sensitivity, Specificity, Precision, Recall, F1)
row.names(eval) <- c("Model 1", "Model 2", "Model 3", "Model 4")
knitr::kable(eval)
```

	Sensitivity	Specificity	Precision	Recall	F1
Model 1	0.9090909	0.8222222	0.8333333	0.9090909	0.8695652
Model 2	0.9090909	0.8222222	0.8333333	0.9090909	0.8695652
Model 3	0.9090909	0.8444444	0.8510638	0.9090909	0.8791209
Model 4	0.9545455	0.9111111	0.9130435	0.9545455	0.9333333

Again Model 3 which is based on only two PCs out performs Model 1 and Model 2. Similarly looking at the ROC/AUC curves

```r
getROC <- function(model) {
    name <- deparse(substitute(model))
    pred.prob1 <- predict(model, newdata = trg, type="prob")
    p1 <- data.frame(pred = trg$target, prob = pred.prob1[[1]])
    p1 <- p1[order(p1$prob),]
    rocobj <- roc(p1$pred, p1$prob)
    plot(rocobj, asp=NA, legacy.axes = TRUE, print.auc=TRUE,
         xlab="Specificity", main = name)
}
par(mfrow=c(2,2))
getROC(modcv6)
getROC(modcv4)
getROC(modcv2)
getROC(modcv)
```



Similarly, `modcv2` which is our Model 3 has a good ROC curve where AUC is .955 which is very close to 1. The reason why we are skipping Model 4 is that it utilizes all of the PCs to give prediction which could be costly in terms of computing power and time. Looking at all the circumstances we will go with Model 3 as our final model to predict.

Making Predictions:

Finally, we can make our final predictions. We can see from the head of our final dataframe and the table output of our predicted variable class that the prediction distribution looks very similar to that of our initial test distribution. Let's load the testing data set:

```
testing <- read_csv("https://raw.githubusercontent.com/Umerfarooq122/predicting-whether-the-neighborhoo
```

```
test <- predict(pc, testing)
```

0	1	prediction
0.8563292	0.1436708	0
0.6500690	0.3499310	0
0.5354625	0.4645375	0
0.5208585	0.4791415	0
0.8854474	0.1145526	0
0.9663826	0.0336174	0

Var1	Freq
0	21
1	19

Conclusion:

In this particular we applied a binary logistic regression model to predict the `target` variable for the testing data. We trained our model using training data. First we explored the data in training data set and we imputed some outliers. After that we encounter multi co-linearity among the independent variable using principal component analysis (PCA). When the data was ready with all the PCs then we formulated a few models followed by section of models based on classification model metrics. We ended up picking the model with only two PCAs which performed well on training data set. When model was finalized then it was applied to testing data to predict the `target`.

Appendix: