R Notebook

Code

read the predicted data

```
predicted_df=read.csv("Predicted_Data.csv",header=TRUE,stringsAsFactors=F)
cat(paste('\nTotal Number of rows in the predicted data',nrow(predicted_df)))
```

```

Total Number of rows in the predicted data 50
```

```
str(predicted_df)
```

```
'data.frame':   50 obs. of  2 variables:
 $ Predicted.Â..Â.: num  0.3151 0.2471 0.0199 0.6352 0.8165 ...
 $ Y              : int  1 0 0 1 0 0 0 1 1 0 ...
```

```
head(predicted_df)
```

Rename the column name

```
newcolnames<-c('Predicted','y')
colnames(predicted_df)<-newcolnames
#check the structure of the data frame
str(predicted_df)
```

```
'data.frame':   50 obs. of  2 variables:
 $ Predicted: num  0.3151 0.2471 0.0199 0.6352 0.8165 ...
 $ y        : int  1 0 0 1 0 0 0 1 1 0 ...
```

now looks fine as the column name has changed

to check the

install.packages(“ROCR”)

```
library(ROCR)
```

```
Loading required package: gplots

Attaching package: <U+393C><U+3E31>gplots<U+393C><U+3E32>

The following object is masked from <U+393C><U+3E31>package:stats<U+393C><U+3E3
2>:

    lowess
```

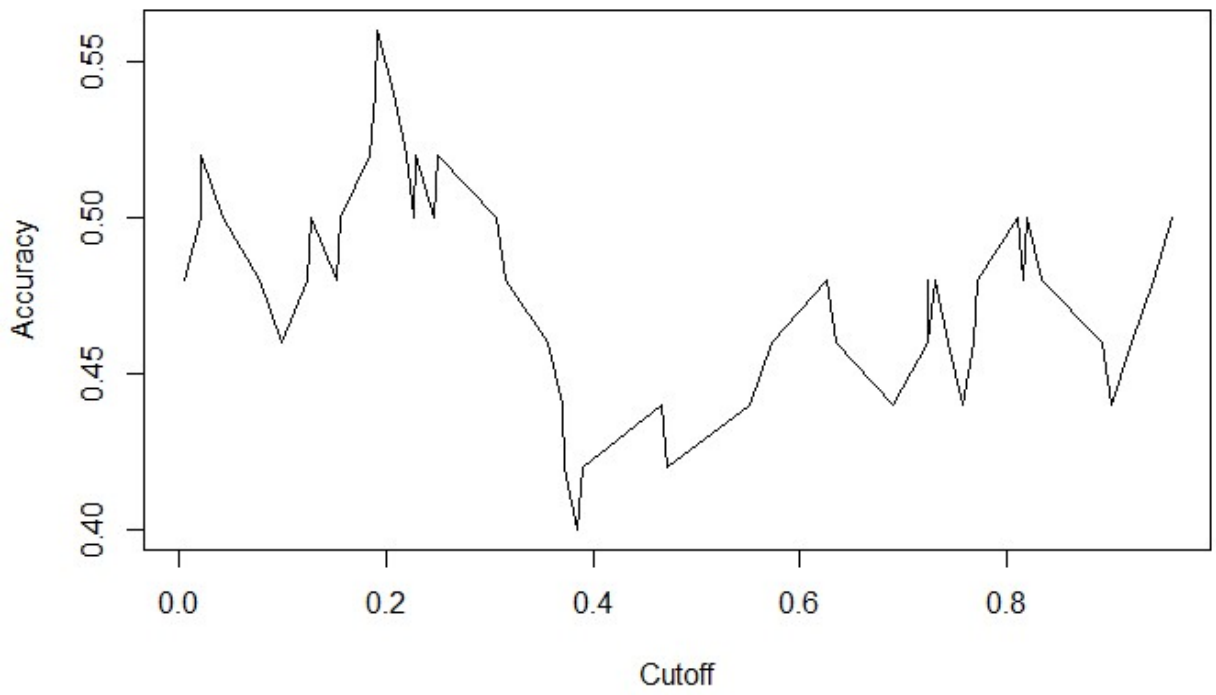
```
#create the object of prediction with the predicted probablilty  and original p
robablity
pred<-prediction(predicted_df$Predicted,predicted_df$y)
pred
```

```
An object of class "prediction"
Slot "predictions":
[[1]]
 [1] 0.315075300 0.247116583 0.019938541 0.635223676 0.816512256 0.768671539 0.
901123086
 [8] 0.207731809 0.626827129 0.724754964 0.004070023 0.833692794 0.123358244 0.
249252743
[15] 0.573036542 0.190291112 0.356869033 0.811016988 0.306974869 0.041476616 0.
920796024
[22] 0.960045044 0.099158739 0.467322766 0.190094205 0.151809214 0.820857232 0.
077273261
[29] 0.552612980 0.732081003 0.758074351 0.390286229 0.724388645 0.020296189 0.
942697427
[36] 0.893232176 0.743319952 0.773019863 0.229053364 0.219980348 0.156213842 0.
371065335
[43] 0.690554685 0.127486623 0.372439317 0.183769105 0.472558978 0.724079378 0.
384598448
[50] 0.226369171


Slot "labels":
[[1]]
 [1] 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 1 1
0 1 1 0 1 0 1 1 0 0
[48] 0 0 0
Levels: 0 < 1


Slot "cutoffs":
[[1]]
 [1]         Inf 0.960045044 0.942697427 0.920796024 0.901123086 0.893232176 0.
833692794
 [8] 0.820857232 0.816512256 0.811016988 0.773019863 0.768671539 0.758074351 0.
743319952
[15] 0.732081003 0.724754964 0.724388645 0.724079378 0.690554685 0.635223676 0.
626827129
[22] 0.573036542 0.552612980 0.472558978 0.467322766 0.390286229 0.384598448 0.
372439317
[29] 0.371065335 0.356869033 0.315075300 0.306974869 0.249252743 0.247116583 0.
229053364
[36] 0.226369171 0.219980348 0.207731809 0.190291112 0.190094205 0.183769105 0.
156213842
[43] 0.151809214 0.127486623 0.123358244 0.099158739 0.077273261 0.041476616 0.
020296189
[50] 0.019938541 0.004070023


Slot "fp":
```

```
[[1]]
 [1]  0  1  2  3  4  4  4  4  5  5  6  7  8  8  8  9  9 10 11 11 11 12 13 14 1
4 15 16 16 16 16 16
[32] 16 16 17 17 18 18 18 18 19 20 21 22 22 23 24 24 24 24 25 26


Slot "tp":
[[1]]
 [1]  0  0  0  0  0  1  2  3  3  4  4  4  4  5  6  6  7  7  7  8  9  9  9  9 1
0 10 10 11 12 13 14
[32] 15 16 16 17 17 18 19 20 20 20 20 20 21 21 21 22 23 24 24 24


Slot "tn":
[[1]]
 [1] 26 25 24 23 22 22 22 22 21 21 20 19 18 18 18 17 17 16 15 15 15 14 13 12 1
2 11 10 10 10 10 10
[32] 10 10  9  9  8  8  8  8  7  6  5  4  4  3  2  2  2  2  1  0


Slot "fn":
[[1]]
 [1] 24 24 24 24 24 23 22 21 21 20 20 20 20 19 18 18 17 17 17 16 15 15 15 15 1
4 14 14 13 12 11 10
[32]  9  8  8  7  7  6  5  4  4  4  4  4  3  3  3  2  1  0  0  0


Slot "n.pos":
[[1]]
[1] 24


Slot "n.neg":
[[1]]
[1] 26


Slot "n.pos.pred":
[[1]]
 [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 2
4 25 26 27 28 29 30
[32] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50


Slot "n.neg.pred":
[[1]]
 [1] 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 2
6 25 24 23 22 21 20
[32] 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
```

Creating the performance object

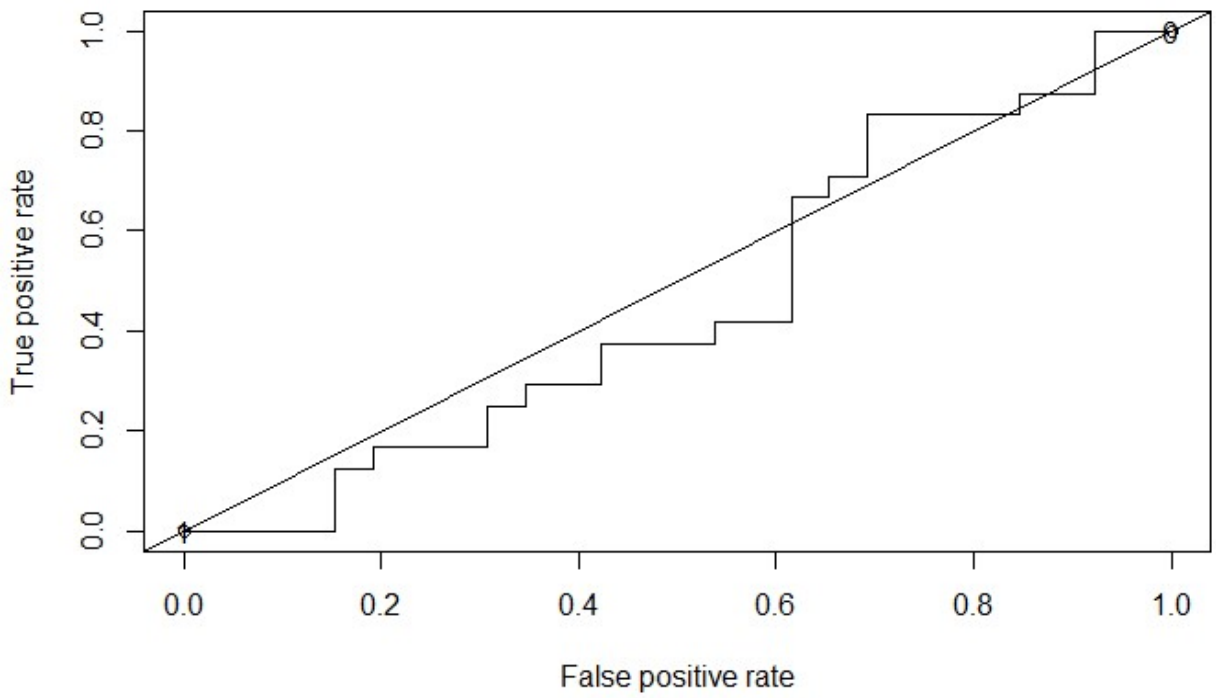
```
#create the performance based on accuracy
perf_acc<-performance(pred,"acc")
#plot the performance based on accuracy
plot(perf_acc)
```



From the above plot we can see if probablity is 0.2 its giving the maximum level accuracy (which is >.5)

Now we will check cut off value based on TPR(True positive Rate) and FPR(False positive Rate)

```
#performance object based on tpr and fpr
perf_TPR_FPR<-performance(pred,measure = "tpr", x.measure = "fpr")
#plot the performance object to see the cut of value
plot(perf_TPR_FPR,print.cutoffs.at=seq(0,1,1))
abline(a=0, b= 1)
```



from the above chart tpr and fpr are proportional

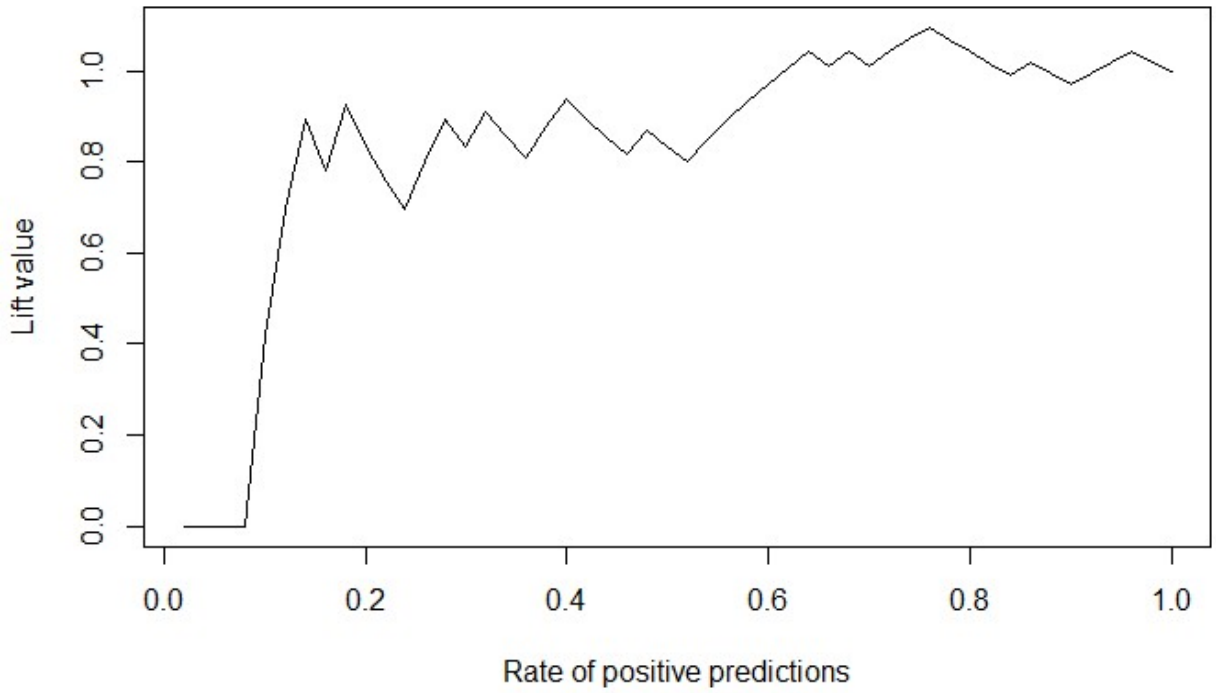
performance based on lift

```
#create the performance object based on lift andrpp
perf_lift<-performance(pred,"lift", "rpp")
perf_lift
```

```
An object of class "performance"
Slot "x.name":
[1] "Rate of positive predictions"

Slot "y.name":
[1] "Lift value"

Slot "alpha.name":
[1] "Cutoff"

Slot "x.values":
[[1]]
 [1] 0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24 0.26 0.2
8 0.30 0.32 0.34 0.36
[20] 0.38 0.40 0.42 0.44 0.46 0.48 0.50 0.52 0.54 0.56 0.58 0.60 0.62 0.64 0.6
6 0.68 0.70 0.72 0.74
[39] 0.76 0.78 0.80 0.82 0.84 0.86 0.88 0.90 0.92 0.94 0.96 0.98 1.00


Slot "y.values":
[[1]]
 [1]       NaN 0.0000000 0.0000000 0.0000000 0.0000000 0.4166667 0.6944444 0.89
28571 0.7812500
[10] 0.9259259 0.8333333 0.7575758 0.6944444 0.8012821 0.8928571 0.8333333 0.91
14583 0.8578431
[19] 0.8101852 0.8771930 0.9375000 0.8928571 0.8522727 0.8152174 0.8680556 0.83
33333 0.8012821
[28] 0.8487654 0.8928571 0.9339080 0.9722222 1.0080645 1.0416667 1.0101010 1.04
16667 1.0119048
[37] 1.0416667 1.0698198 1.0964912 1.0683761 1.0416667 1.0162602 0.9920635 1.01
74419 0.9943182
[46] 0.9722222 0.9963768 1.0195035 1.0416667 1.0204082 1.0000000


Slot "alpha.values":
[[1]]
 [1]         Inf 0.960045044 0.942697427 0.920796024 0.901123086 0.893232176 0.
833692794
 [8] 0.820857232 0.816512256 0.811016988 0.773019863 0.768671539 0.758074351 0.
743319952
[15] 0.732081003 0.724754964 0.724388645 0.724079378 0.690554685 0.635223676 0.
626827129
[22] 0.573036542 0.552612980 0.472558978 0.467322766 0.390286229 0.384598448 0.
372439317
[29] 0.371065335 0.356869033 0.315075300 0.306974869 0.249252743 0.247116583 0.
229053364
[36] 0.226369171 0.219980348 0.207731809 0.190291112 0.190094205 0.183769105 0.
156213842
```

```
[43] 0.151809214 0.127486623 0.123358244 0.099158739 0.077273261 0.041476616 0.
020296189
[50] 0.019938541 0.004070023
```

```
plot(perf_lift)
```



As we know that lift chart shows how much more likely we are to receive positive responses than if we contact a random sample of customers. For example, by contacting only 20% of customers based on the predictive model we will reach >8 times as many respondents, as if we use no model.