1)

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> View(Bike_data)
> attach(Bike_data)
> fix(Bike_data)
> set.seed(1)
> train=sample(711,600)
> count01=rep("High",nrow(Bike_data))
> count01[count<median(count)]="Low"
> Bike_data$count01=count01
> Bike_data$season=as.factor(Bike_data$season)
> Bike_data$year=as.factor(Bike_data$year)
> Bike_data$month=as.factor(Bike_data$month)
> Bike_data$holiday=as.factor(Bike_data$holiday)
> Bike_data$weekday=as.factor(Bike_data$weekday)
> Bike_data$weathersit=as.factor(Bike_data$weathersit)
> Bike_data$count01=as.factor(Bike_data$count01)
> count01=as.factor(count01)
> bike_test=Bike_data[-train,]
> count_resp=count01[-train]
> contrasts(bike_test$count01)
     Low
High   0
Low    1
> 
```

The contrast here suggests that Low is taken as '1' and High is taken as '0'.

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> Bike_data=Bike_data[,-11]
> log1.fit=glm(count01~.,data=Bike_data,family=binomial,subset=train)
```

```
> summary(log1.fit)

Call:
glm(formula = count01 ~ ., family = binomial, data = Bike_data,
    subset = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.39008  -0.28009  -0.00804   0.13082   2.88026


Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   5.5328     1.8837   2.937 0.003312 **
season2      -0.9936     0.9196  -1.081 0.279914
season3      -3.4427     1.4928  -2.306 0.021097 *
season4      -3.9308     1.6971  -2.316 0.020545 *
year1        -2.8961     0.4018  -7.209 5.65e-13 ***
month2       -0.9351     1.7586  -0.532 0.594898
month3       -6.1155     1.5046  -4.064 4.82e-05 ***
month4       -6.2123     1.7678  -3.514 0.000441 ***
month5       -5.5537     1.8438  -3.012 0.002594 **
month6       -5.1444     1.8968  -2.712 0.006685 **
month7       -1.6007     2.2246  -0.720 0.471788
month8       -3.1405     2.2150  -1.418 0.156233
month9       -2.7350     2.1251  -1.287 0.198098
month10      -3.3519     2.2021  -1.522 0.127974
month11      -1.2015     2.1595  -0.556 0.577956
month12       0.5490     2.1518   0.255 0.798635
holiday1     -1.9271     1.0927  -1.764 0.077784 .
weekday1      4.7023     0.7977   5.895 3.75e-09 ***
weekday2      5.7954     0.8592   6.745 1.53e-11 ***
weekday3      6.3662     0.8999   7.074 1.50e-12 ***
weekday4      5.6281     0.8449   6.661 2.71e-11 ***
weekday5      3.8248     0.7583   5.044 4.56e-07 ***
weekday6     -1.1014     0.7663  -1.437 0.150647
weathersit2   1.3542     0.4479   3.023 0.002499 **
weathersit3  16.0805   717.2471   0.022 0.982113
temp        -12.1763    10.8907  -1.118 0.263550
atemp        -2.6124    11.9534  -0.219 0.827002
hum           5.1799     1.8405   2.814 0.004886 **
windspeed     8.0661     2.5342   3.183 0.001458 **
---
```

According to the logistic model arrived at, it can be inferred that the variables season, year, month, holiday, weekday, weathersit, hum, windspeed seem to be having a significant effect on the outcome "count01".

Inferences/Interpretations (of the statistically significant predictors):

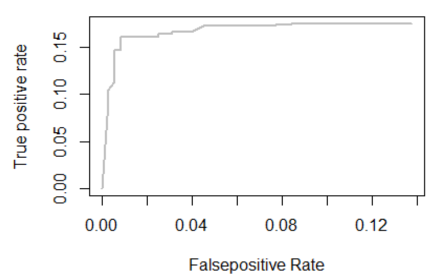
- Compared to the season "Spring", season "Fall" and "winter" seem to have a change in the log odds of 'count01' by -3.47 and -3.93 respectively and hence more people ride the bikes during that season.

- In the year 2012, the log odds decreases by 2.896 as the coefficient correspondingly has a negative coefficient. Hence the probability of ‘Low’ decreases, and so more people tend to ride the bikes in 2012 than in 2011.
- Compared to ‘January’, the months “March, April, May, June’ seem to have a lower log odds or as the coefficient correspondingly for months 3,4,5,6 are negative, the probability of ‘low’ decreases and hence those months have more bike share riders comparatively. Moreover, these months are not as cold as January.
- The probability of ‘low’ of ‘count01’ is higher in weekdays than on Sunday, a weekend day. Weekdays seem to have lesser bike share riders compared to Sunday as the corresponding coefficients of the weekdays are positive. This might be because people don’t use the bikes to offices, schools etc on weekdays, but only for recreation in the Sundays.
- Compared to a day which is clear/cloudy partly (weathersit=1), a day with mist/cloud(weathersit=2), the probability of ‘low’ of ‘count01’ is higher and as the coefficients are correspondingly positive, more people tend to ride bikes in “weathersit=1” than in “weathersit=2”.
- A day with high humidity and windspeed has a higher probability of ‘low’ of ‘count01’.Hence as the coefficient correspondingly is positive, the log odds of the normalized values is higher by 5.18 and 8.06 respectively and hence the bike riders are low in a humid and/or windy days.

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 831.45  on 599  degrees of freedom
Residual deviance: 265.18  on 571  degrees of freedom
AIC: 323.18

Number of Fisher Scoring iterations: 16

> log1.prob=predict(log1.fit,bike_test,type="response")
> log1.pred=rep("Low",length(log1.prob))
> log1.pred[log1.prob<0.5]="High"
> table(log1.pred,count_resp)
         count_resp
log1.pred High Low
     High   43   5
     Low     6  57
> mean(log1.pred==count_resp)
[1] 0.9009009
```

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> roc.curve=function(s,print=FALSE)
+ {
+   Ps=(log1.pred>s)*1
+   FP=sum((Ps==1)*(bike_test$count01=="High"))/sum(Bike_data$count01=="High")
+   TP=sum((Ps==1)*(bike_test$count01=="Low"))/sum(Bike_data$count01=="Low")
+   if(print==TRUE){
+     print(table(Observed=bike_test$count01,Predicted=Ps))
+   }
+   vect=c(FP,TP)
+   names(vect)=c("FPR","TPR")
+   return(vect)
+ }
> threshold=0.5
> roc.curve(threshold,print=TRUE)
        Predicted
Observed  0  1
    High 43  6
    Low   5 57
       FPR        TPR
0.01685393 0.16056338
> ROC.curve=Vectorize(roc.curve)
> M.ROC=ROC.curve(seq(0,1,by=0.01))
> plot(M.ROC[1,],M.ROC[2,],col="grey",lwd=2,type="l",xlab="Falsepositive Rate",ylab="True positive rate")
> 
```

<u>ROC CURVE</u>



Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(57+43)/(62+49)=90.09%
Error Rate=1-0.9009=9.91%
Sensitivity = True Positive/Positive=57/62=91.94%
Specificity=True Negative/Negative=43/49=87.76%

2)

If high prediction performance was the only criteria, for this question, boosting/random forest/ bagging could be used. But as interpretability is also considered, classification trees are used. As pruning increases the prediction accuracy, pruning of the basic unpruned tree is done using cross validation and pruning.

UNPRUNED TREE:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> tree.car=tree(count01~.,Bike_data,subset=train)
> summary(tree.car)

Classification tree:
tree(formula = count01 ~ ., data = Bike_data, subset = train)
Variables actually used in tree construction:
[1] "atemp"      "weekday"    "temp"       "month"      "windspeed"  "holiday"    "year"
[8] "hum"        "weathersit"
Number of terminal nodes:  24
Residual mean deviance:  0.3213 = 185.1 / 576
Misclassification error rate: 0.075 = 45 / 600
> plot(tree.car)
> text(tree.car ,pretty =0)
> tree.car.pred=predict(tree.car,bike_test,type="class")
> table(tree.car.pred,count_resp)
             count_resp
tree.car.pred High Low
         High   41  11
         Low     8  51
> mean(tree.car.pred==count_resp)
[1] 0.8288288
```

Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(51+41)/(62+49)=82.88%

Error Rate=1-0.8288=17.12%

Sensitivity = True Positive/Positive=51/62=82.26%

Specificity=True Negative/Negative=41/49=83.67%

PRUNED TREE:

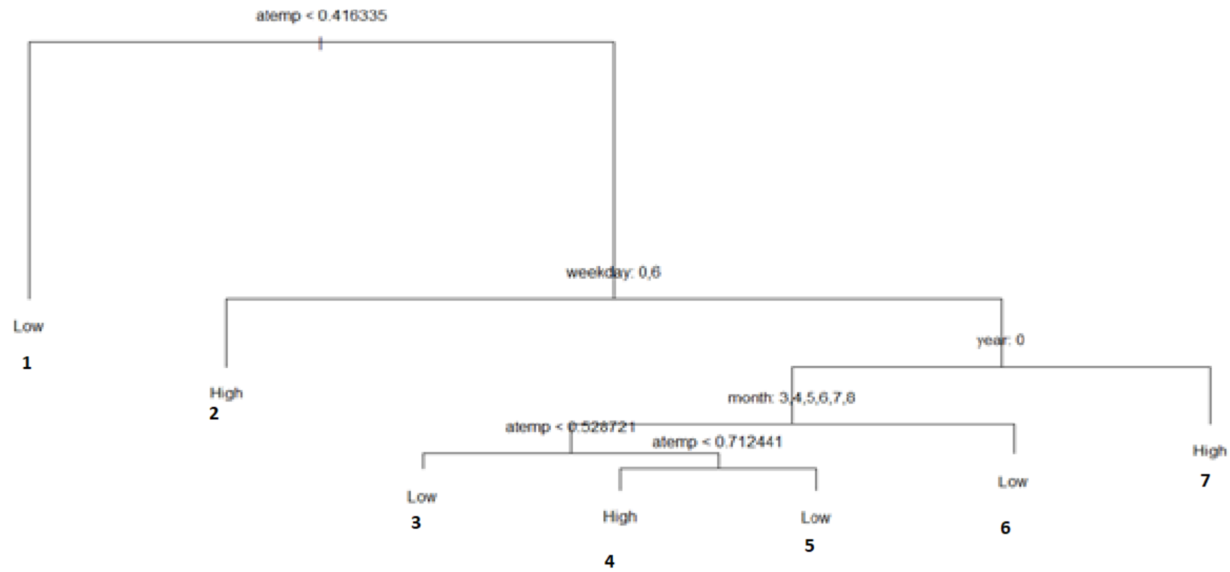
```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> set.seed(10)
> cv.carseats <- cv.tree(tree.car,FUN=prune.misclass)
> (cv.carseats)
$size
[1] 24 21 19 12 11  7  6  2  1

$dev
[1] 115 115 112 113 110 107 131 135 293

$k
[1] -Inf    0    1    2    3    4    9   10  164

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
> plot(cv.carseats$size, cv.carseats$dev, type = "b")
> prune.car=prune.misclass(tree.car,best=7)
> plot(prune.car)
> text(prune.car,pretty=0)
> prune.car.pred=predict(prune.car,bike_test,type="class")
> table(prune.car.pred,count_resp)
              count_resp
prune.car.pred High Low
          High   42   9
          Low     7  53
> mean(prune.car.pred==count_resp)
[1] 0.8558559
> 
```

Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(53+44)/(62+49)=85.59%

Error Rate=1-0.8559=14.41%

Sensitivity = True Positive/Positive=53/62=85.48%

Specificity=True Negative/Negative=44/49=85.71%



- The final model arrived at has 7 terminal nodes.

- Totally only 4 predictors/features (1 continuous and 3 categorical variables) were used in the model (atemp, weekday, month, year).

-The most important or the first and foremost and many more split happens on the 'atemp' variable. In the first split, It has a decision point such that whether atemp is lesser than 0.416 or not (if lesser move on the left side of the branch else right) .Here, as the normalization of the variables is used, 0.416 is interpreted as :

0.416*standard deviation (of original atemp data value before normalization) +mean(of original atemp data value before normalization) to get the actual sense.

<u>General Interpretations:</u>

<u>Terminal Node 1</u>: When the temperature is lower, i.e., if the real feel temperature is less than the temperature corresponding to the normalized value of 0.413, people don't use the bike much due to cold weather.

<u>Terminal Node 2</u>: At the temperature corresponding to atemp>0.416, people prefer prefer using the bikes during the weekends probably for recreation as during the weekdays, the offices and schools might be far and they prefer other modes of transportation

<u>Terminal Node 3,4,5</u>: During the period of March to August in 2011,people don't use the bikes much if the temperature is less than the one corresponding to atemp=0.528 and when the temperature above the atemp corresponding to 0.712, but do so if the real feel s nominal between 0.528 and 0.712 because, the temperature is too cold during below atemp=0.528 and above atemp=0.712. This is not the case for other months as the temperature then is always cold with atemp<0.528

<u>Terminal 6</u>: As the months from September to February are cold, during the weekdays of 2011 people don't use the bikes much.

<u>Terminal 7</u>: It can be seen that people became health conscious with the progress of time from 2011 to 2012 even during the weekdays to offices and schools if the temperature is bearable/not colder when atemp is above 0.416.

3)

Prediction Accuracy= (True Positive+ True Negative/Positive +Negative)

1. **LDA**:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> library(MASS)
> lda.fit=lda(count01~.,data=Bike_data,family=binomial,subset=train)
> summary(lda.fit)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   56     -none- numeric
scaling 28     -none- numeric
lev      2     -none- character
svd      1     -none- numeric
N        1     -none- numeric
call     5     -none- call
terms    3     terms  call
xlevels  6     -none- list
> log1.prob=predict(lda.fit,bike_test,type="response")
> lda1.class=log1.prob$class
> table(lda1.class,count_resp)
          count_resp
lda1.class High Low
      High   46   8
      Low     3  54
> mean(lda1.class==count_resp)
[1] 0.9009009
```

Prediction Accuracy=90.09%

Error Rate=1-0.9009=9.91%

Sensitivity = True Positive/Positive =54/62=87.1%

Specificity=True Negative/Negative=46/49=93.88%

2. **QDA**:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> qda.fit=qda(count01~.-weathersit,data=Bike_data,family=binomial,subset=train)
> summary(qda.fit)
        Length Class  Mode
prior      2   -none- numeric
counts     2   -none- numeric
means     52   -none- numeric
scaling 1352   -none- numeric
ldet       2   -none- numeric
lev        2   -none- character
N          1   -none- numeric
call       5   -none- call
terms      3   terms  call
xlevels    6   -none- list
> qda.prob=predict(qda.fit,bike_test,type="response")
> qda.class=qda.prob$class
>
> table(qda.class,count_resp)
         count_resp
qda.class High Low
     High   48  17
     Low     1  45
> mean(qda.class==count_resp)
[1] 0.8378378
```

Prediction Accuracy=83.78%

Error Rate=1-0.8378=16.22%

Sensitivity = True Positive/Positive=45/62=72.58%

Specificity=True Negative/Negative=48/49=97.96%

3. **KNN**:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> library(class)
> train.X=cbind(season,year,month,weekday,weathersit,temp,holiday,atemp,hum,windspeed)[train,]
> test.X=cbind(season,year,month,weekday,weathersit,temp,holiday,atemp,hum,windspeed)[-train,]
> train.Direction=count01[train]
> maxk=0
> max_K=1
> mean_K=0
> for (i in 1:100)
+ {
+   set.seed(1)
+   knn.pred=knn(data.frame(train.X),data.frame(test.X),train.Direction,k=i)
+   mean_K=mean(knn.pred==count_resp)
+   if(mean_K>maxk)
+   {   maxk=mean_K
+   max_K=i
+   }
+ }
> print(maxk)
[1] 0.8738739
> print(max_K)
[1] 6
> set.seed(1)
> knn.pred=knn(data.frame(train.X),data.frame(test.X),train.Direction,k=6)
> table(knn.pred,count_resp)
        count_resp
knn.pred High Low
    High   43   8
    Low     6  54
> mean(knn.pred==count_resp)
[1] 0.8738739
```

Prediction Accuracy=87.38%

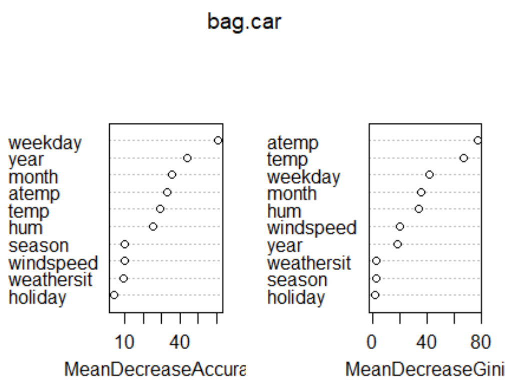
Error Rate=1-0.8738=16.22%

Sensitivity = True Positive/Positive=54/62=87.09%

Specificity=True Negative/Negative=43/49=87.75%

4. **BAGGING**:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> library(MASS)
> set.seed(1)
> bag.car=randomForest(count01~.,data=Bike_data,subset=train,mtry=10,ntree=100,importance=TRUE)
> bag.car

Call:
 randomForest(formula = count01 ~ ., data = Bike_data, mtry = 10,      ntree = 100, importance = TRUE, subset
 = train)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 10

        OOB estimate of  error rate: 12%
Confusion matrix:
     High Low class.error
High  277  30  0.09771987
Low    42 251  0.14334471
> bag.car.pred=predict(bag.car,newdata=Bike_data[-train,])
> table(bag.car.pred,count_resp)
            count_resp
bag.car.pred High Low
        High   44   6
        Low     5  56
> mean(bag.car.pred==count_resp)
[1] 0.9009009
```



Prediction Accuracy=90.09%

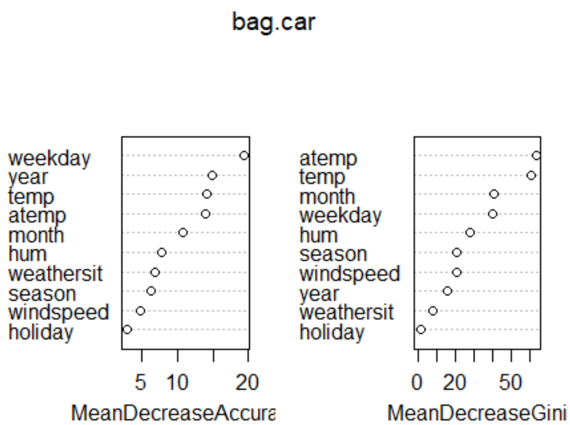
Error Rate=1-0.9009=9.91%

Sensitivity = True Positive/Positive=56/62=90.32%

Specificity=True Negative/Negative=44/49=89.8%

5. **RANDOM FOREST:**

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> library(randomForest)
> library(MASS)
> set.seed(1)
> bag.car=randomForest(count01~.,data=Bike_data,subset=train,mtry=3,ntree=100,importance=TRUE)
> bag.car

Call:
 randomForest(formula = count01 ~ ., data = Bike_data, mtry = 3,      ntree = 100, importance = TRUE, subset
= train)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 3

        OOB estimate of  error rate: 11.83%
Confusion matrix:
     High Low class.error
High  275  32   0.1042345
Low    39 254   0.1331058
> bag.car.pred=predict(bag.car,newdata=Bike_data[-train,])
> table(bag.car.pred,count_resp)
            count_resp
bag.car.pred High Low
        High   45   9
        Low     4  53
> mean(bag.car.pred==count_resp)
[1] 0.8828829
```



Prediction Accuracy=88.28%

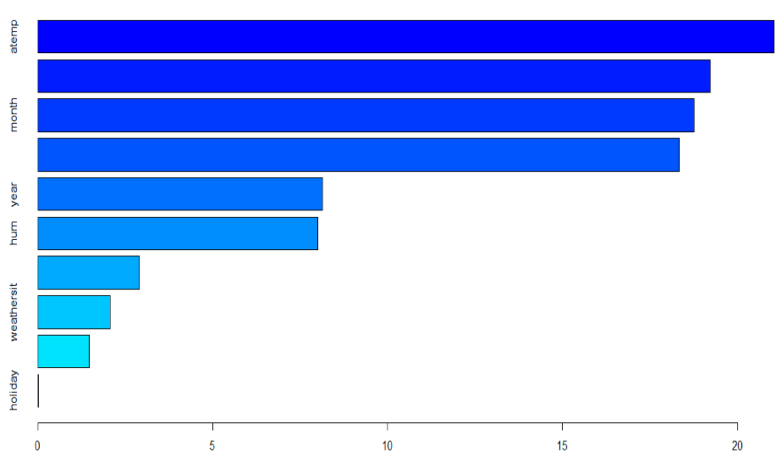
Error Rate=1-0.8828=11.72%

Sensitivity = True Positive/Positive=53/62=85.48%

Specificity=True Negative/Negative=45/49=91.84%

6. BOOSTING:

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> set.seed(1)
> boost.boston=gbm(unclass(count01)-1~.,data=Bike_data[train,],distribution="bernoulli",n.trees=5000,interact
ion.depth =4)
> summary(boost.boston)
                  var     rel.inf
atemp           atemp 21.04804942
temp             temp 19.21464483
month           month 18.76113978
weekday       weekday 18.33603568
year             year  8.13290255
hum               hum  8.00965924
windspeed   windspeed  2.90875618
weathersit weathersit  2.07724578
season         season  1.48142968
holiday       holiday  0.03013687
> par(mfrow=c(1,2))
> plot(boost.boston)
> plot(boost.boston)
> pred.boost=predict(boost.boston,Bike_data[-train,],n.trees=5000,type="response")
> log1.pred=rep("High",length(pred.boost))
> log1.pred[pred.boost>0.5]="Low"
> table(log1.pred,count_resp)
         count_resp
log1.pred High Low
     High   45   5
     Low     4  57
> mean(log1.pred==count_resp)
[1] 0.9189189
```



Prediction Accuracy=91.89%

Error Rate=1-0.9189=8.11%

Sensitivity = True Positive/Positive=57/62=91.94%

Specificity=True Negative/Negative=45/49=91.84%

7. SVM(Linear):

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> library(e1071)
> tune.out=tune(svm,count01~.,data=Bike_data[train,],kernel="linear",ranges=list(cost=c(0.01,0.1,1,5,10,100))
,scale=FALSE)
> summary(tune.out)#error min=cost(1)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.115

- Detailed performance results:
   cost     error dispersion
1 1e-02 0.2216667 0.04161107
2 1e-01 0.1650000 0.04934760
3 1e+00 0.1366667 0.04288946
4 5e+00 0.1150000 0.04190672
5 1e+01 0.1166667 0.03767961
6 1e+02 0.1183333 0.03963569

> svm.fit=svm(count01~.,data=Bike_data,subset=train,kernel="linear",cost=10,scale=FALSE)
> summary(svm.fit)
```

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> summary(svm.fit)

Call:
svm(formula = count01 ~ ., data = Bike_data, kernel = "linear", cost = 10, subset = train,
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.03448276

Number of Support Vectors:  170

 ( 86 84 )


Number of Classes:  2

Levels:
 High Low


> pred=predict(svm.fit,newdata=Bike_data[-train,])
> table(pred,count_resp)
      count_resp
pred   High Low
  High   45   5
  Low     4  57
> mean(pred==count_resp)
[1] 0.9189189
```

Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(57+45)/(62+49)=91.89%

Error Rate=1-0.8378=8.11%

Sensitivity = True Positive/Positive=57/62=91.94%

Specificity=True Negative/Negative=45/49=91.84%

8. SVM(Radial):

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
[1] 0.9189189
> tune.out=tune(svm,count01~.,data=Bike_data[train,],kernel="radial",ranges=list(cost=c(0.01,0.1,1,5,10,100),
gamma=c(0.5,1,2,3,4)),scale=FALSE)
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100   0.5

- best performance: 0.125
```

```
> svm.fit=svm(count01~.,data=Bike_data,subset=train,kernel="radial",gamma=0.5,cost=100,scale=FALSE)
> summary(svm.fit)

Call:
svm(formula = count01 ~ ., data = Bike_data, kernel = "radial", gamma = 0.5, cost = 100,
    subset = train, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  0.5

Number of Support Vectors:  226

 ( 115 111 )


Number of Classes:  2

Levels:
 High Low



> pred=predict(svm.fit,newdata=Bike_data[-train,])
> table(pred,count_resp)
      count_resp
pred   High Low
  High   41   9
  Low     8  53
> mean(pred==count_resp)
[1] 0.8468468
```

Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(53+41)/(62+49)=84.68%

Error Rate=1-0.8468=15.32%

Sensitivity = True Positive/Positive=53/62=85.48%

Specificity=True Negative/Negative=41/49=83.67%

9. SVM(Polynomial):

```
Console C:/Users/Karthik/Desktop/Sem 1/ISEN 613/
> tune.out=tune(svm,count01~.,data=Bike_data[train,],kernel="polynomial",ranges=list(cost=c(0.01,0.1,1,5,10,1
00),degree=c(2,3,4,5)),scale=FALSE)
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
  100      2

- best performance: 0.13
```

```
> svm.fit=svm(count01~.,data=Bike_data,subset=train,kernel="polynomial",cost=100, degree=2,scale=FALSE)
> pred=predict(svm.fit,newdata=Bike_data[-train,])
> table(pred,count_resp)
      count_resp
pred   High Low
  High   42  11
  Low     7  51
> mean(pred==count_resp)
[1] 0.8378378
```

Prediction Accuracy=(True Positive+ True Negative/Positive +Negative)=(51+42)/(62+49)=83.78%

Error Rate=1-0.8378=16.22%

Sensitivity = True Positive/Positive=51/62=82.26%

Specificity=True Negative/Negative=42/49=85.71%

4)

CLASSIFIER	Accuracy (in %)	Error Rate (in %)	Sensitivity (in %)	Specificity (in %)
Logistic Regression	90.09	9.91	91.94	87.76
Unpruned Classification Tree	82.88	17.12	82.26	83.67
Pruned Classification Tree	85.59	14.41	85.48	85.71
LDA	90.09	9.91	87.1	93.88
KNN	87.38	12.62	87.09	87.75
Bagging	90.09	9.91	90.32	89.8
Random Forest	88.28	11.72	85.48	91.84
Boosting	91.89	8.11	91.94	91.84
SVM(Linear)	91.89	8.11	91.94	91.84
SVM(Radial)	84.68	15.32	85.48	83.67
SVM(Polynomial)	83.78	16.22	82.26	85.71
QDA	83.78	16.22	72.58	97.96

Note: QDA had rank deficiency errors when the 'weathersit' variable was included, hence only that feature was not included in the QDA model.

[Best set of parameters after cross validation of the parameters and other methods were used as much as possible in all the classifiers]

- The "*Support Vector Machine Linear Kernel Model*" and the "*Boosting Model*" give similar and the best accuracy, least error rates, high sensitivity and specificity. This is because boosting a slow learning process and hence is having higher prediction accuracy than it's tree model counterparts as well as the other methods and the SVM linear kernel model performs better than all other models as the boundaries here seem to be linear and as it is an advanced model.
- It can be observed that the models like Logistic Regression, LDA, Support Vector Machine Linear Model perform better than methods like KNN, QDA and some other methods because the "decision boundary is seeming to be predominantly linear".
- Boosting outperforms random forest and bagging as it is a method that learns slowly from its residuals and hence is better.
- Pruned and the unpruned trees both do not comparatively perform better because they can outperform linear models only if the relation between the predictor and the response is non-linear and here the decision boundary seems to be linear.
- Support Vector Machine Linear kernel model outperforms the radial and the polynomial kernel models as the boundary type here seems to be linear. The decision surface is a hyperplane, hence the classification problem is linear.
- The pruned and the unpruned tree do not perform better than bagging, boosting and random forest as the variance is higher there and relatively more bias.