Bike Sharing System – Data Analysis

Data Preprocessing

The first step in any data analysis is to look at the dataset and find basic information such as the number of observations, variables, rows, columns, variable types, numeric summary, graphical summary, column and row names. It is performed using the following R commands

```
> bike <- read.csv("Bike_data.csv")
> str(bike)
'data.frame':   711 obs. of  11 variables:
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ year      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ month     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ weekday   : int  6 0 1 2 3 4 5 6 0 2 ...
 $ weathersit: int  2 2 1 1 1 1 2 2 1 2 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ count     : int  331 131 120 108 82 88 148 68 54 43 ...
> sum(is.na(bike))
[1] 0
```

As the output of is.na() function is 0 there are no missing/null values in the data. From the summary we could notice that some variables which are meant to be factors are loaded as integers by R which is rectified by encoding those variables.

Encoding categorical variables: From the analysis it is identified that the variables 'season', 'year', 'month', 'holiday', 'weekday', 'weathersit' are categorical whereas they are represented as integers in the dataset. Those variables are encoded as follows

```
> bike$season <- factor(bike$season, levels=c(1,2,3,4), labels=c("spring","summer","fall","winter"))
> bike$year <- factor(bike$year, levels=c(0,1), labels=c("2011","2012"))
> bike$month <- factor(bike$month, levels=c(1,2,3,4,5,6,7,8,9,10,11,12), labels=c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"))
> bike$holiday <- factor(bike$holiday, levels=c(1,0), labels=c("holiday","not"))
> bike$weekday <- factor(bike$weekday, levels=c(0,1,2,3,4,5,6), labels=c("Sun","Mon","Tue","Wed","Thu","Fri","Sat"))
> bike$weathersit <- factor(bike$weathersit, levels=c(1,2,3,4), labels=c("Clear","Mist","Light Snow","Heavy Rain"))
```

Multi collinearity analysis: Multi collinearity masks the real effect of a predictor variable on the response due to interaction among predictor variables. Thus, dataset should be analyzed for significant multicollinearity among predictor variables.

```
> cor(bike[,c(7:10)])
                temp      atemp        hum  windspeed
temp       1.0000000  0.9915141  0.1224348 -0.1558074
atemp      0.9915141  1.0000000  0.1356574 -0.1818180
hum        0.1224348  0.1356574  1.0000000 -0.2519732
windspeed -0.1558074 -0.1818180 -0.2519732  1.0000000
```

The result of cor() function implies that there is significant correlation among the predictor variables temp and atemp. Thus, in general dropping one of the collinear variables improves model performance. The above output also indicates the direction of relationship between predictor variables (negative sign indicates inverse relation and positive sign indicates direct relation among each pair).

Lakshmi Jeyakumar

Converting continuous response into categorical

As the response variable count is numeric (continuous) a new categorical variable 'count01' is created to do classification analysis on the dataset.

```
> count01 <- rep("Low", 711)
> count01[count>median(count)] <- "High"
```

'count01' has two levels 'High' and 'Low' based on the median of 'count' variable in the original dataset. High indicates that the corresponding value is greater than the median while low indicates that it is less than the median value. A data frame is created for further analysis by eliminating 'count' variable and adding the categorical response variable 'count01'.

```
> df.bike <- data.frame(bike[,-11], count01)
```

The dataset is split into training set of 600 observations and test set of 111 observations as shown

```
> set.seed(1)
> train <- sample(1:dim(df.bike)[1],600)
> train.set <- df.bike[train,]
> test.set <- df.bike[-train,]
```

A new variable containing the response variable for test data set is created as follows

```
> count01.test <- count01[-train]
```

1. Fit a logistic regression model for the training data. Interpret the fitted model. Find its prediction performance (prediction accuracy, sensitivity, specificity) on the test data. **(Note: Show formulas and calculations on each performance measure.)**

```
> logistic.fit = glm(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike,family=binomial)
> summary(logistic.fit)

Call:
glm(formula = count01 ~ atemp + weekday + month + windspeed +
    year + hum, family = binomial, data = df.bike)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.18326  -0.32210   0.00077   0.18377   3.08186

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    3.3776     1.4029   2.408 0.016060 *
atemp        -15.9505     2.4509  -6.508 7.61e-11 ***
weekdayMon     3.8643     0.6277   6.156 7.46e-10 ***
weekdayTue     5.0399     0.6828   7.381 1.57e-13 ***
weekdayWed     5.4890     0.7133   7.695 1.41e-14 ***
weekdayThu     4.5734     0.6578   6.952 3.59e-12 ***
weekdayFri     3.1211     0.6131   5.091 3.56e-07 ***
weekdaySat    -1.2019     0.6268  -1.918 0.055167 .
monthFeb       0.2630     1.4836   0.177 0.859290
monthMar      -4.4826     1.0955  -4.092 4.28e-05 ***
monthApr      -5.1507     1.1247  -4.579 4.66e-06 ***
monthMay      -5.2793     1.2090  -4.367 1.26e-05 ***
monthJun      -4.6889     1.2699  -3.692 0.000222 ***
monthJul      -3.5185     1.3316  -2.642 0.008237 **
monthAug      -4.9526     1.3042  -3.798 0.000146 ***
monthSep      -4.8041     1.2377  -3.881 0.000104 ***
monthOct      -5.3823     1.1558  -4.657 3.21e-06 ***
monthNov      -3.9702     1.0801  -3.676 0.000237 ***
monthDec      -1.2428     1.1432  -1.087 0.276989
windspeed      9.9213     2.1129   4.696 2.66e-06 ***
year2012      -2.5728     0.3303  -7.790 6.73e-15 ***
hum            7.5322     1.2409   6.070 1.28e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 985.65  on 710  degrees of freedom
Residual deviance: 348.19  on 689  degrees of freedom
AIC: 392.19

Number of Fisher Scoring iterations: 7
> logistic.prob=predict(logistic.fit,test.set,type="response")
> logistic.pred=rep("Low",111)
> logistic.pred[logistic.prob>0.5]="High"
> table(logistic.pred,count01.test)
             count01.test
logistic.pred High Low
         High    6  56
         Low    42   7
```

Logistic regression model with 6 significant predictors is fit on the training data and the fitted model is used to predict the test set results.

Accuracy rate: Accuracy rate = [(No. of correct predictions) / (Total number of predictions)]*100 =(TN+TP)/N+P = [(6+7)/111] * 100 = 11.7117%
Error rate: Error rate = 100-Accuracy rate = 100-11.7117= 88.2883%
Sensitivity: Sensitivity refers to the proportion of positives that are correctly identified
Sensitivity = (True Positive) / (True Positive + False Negative) = 6 / (48) = 12.5%
Specificity: Specificity refers to the proportion of negatives that are identified correctly
Specificity = True Negative / (True Negative + False Positive) = 7 / (63) = 11.1111%
Thus, logistic regression **performs poorly** on this dataset. This indicates the possibility of non-linearity in the dataset.

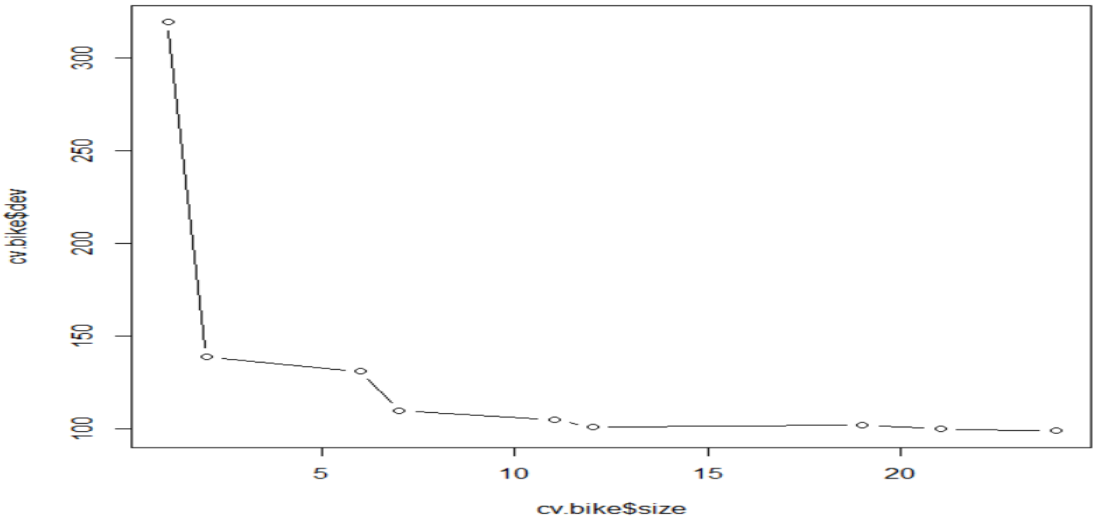
2. Fit a tree model for the training data considering both prediction performance and interpretability (that is, your model must be good in prediction and easy to interpret). Interpret the fitted model. Find its prediction performance (prediction accuracy, sensitivity, specificity) on the test data. **(Note: Justify your choices, if any, in the model building process.)**

Tree model (unpruned tree) is fit as follows

```
> library(tree)
> tree.bike <- tree(count01~., df.bike, subset=train)
> tree.pred <- predict(tree.bike, test.set,type="class")
> table(tree.pred, count01.test)
          count01.test
tree.pred High Low
     High   40  12
     Low     8  51
```

As unpruned tree has high variance and could lead to overfitting of test dataset the tree should be pruned to reduce variance and hence improve prediction accuracy. Cross validation is performed to find the optimal tree size for pruning

```
> set.seed(1)
> cv.bike <- cv.tree(tree.bike,FUN=prune.misclass)
> plot(cv.bike$size,cv.bike$dev,type="b")
```

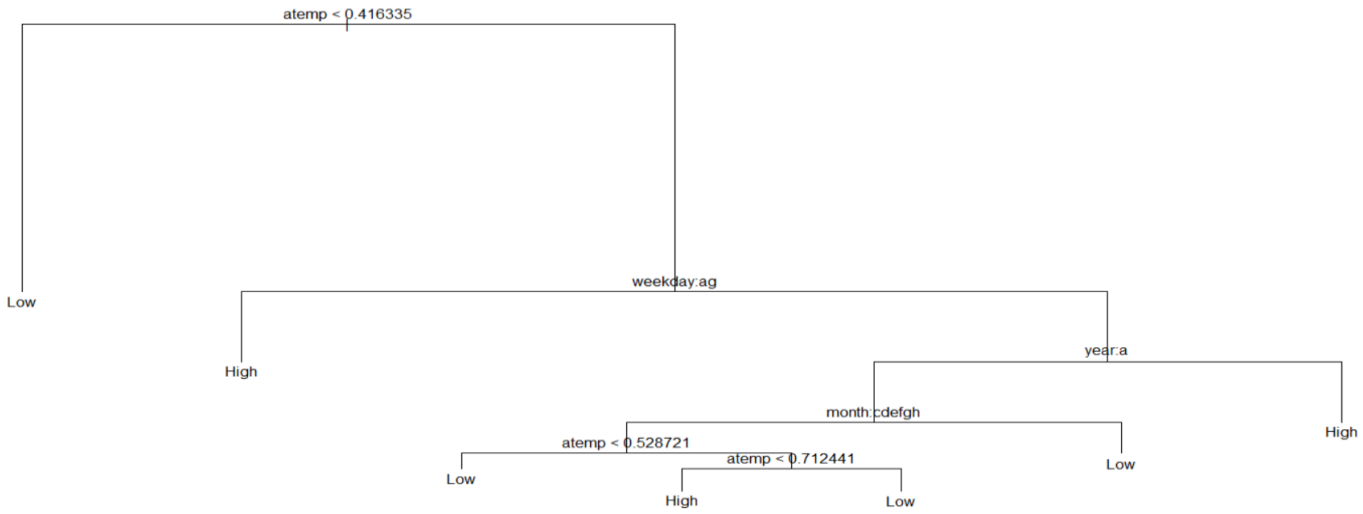


The plot indicates that tree sizes from 7 onwards have approximately similar deviance. Thus, tree size is chosen as 7.

```
> prune.bike1=prune.misclass(tree.bike,best=7)
> prune.pred=predict(prune.bike1,test.set,type="class")
> table(prune.pred,count01.test)
           count01.test
prune.pred High Low
      High   42   9
      Low     6  54
```

```
> summary(prune.bike1)

Classification tree:
snip.tree(tree = tree.bike, nodes = c(6L, 29L, 56L, 15L, 114L,
2L))
Variables actually used in tree construction:
[1] "atemp"   "weekday" "year"    "month"
Number of terminal nodes:  7
Residual mean deviance:  0.7582 = 449.6 / 593
Misclassification error rate: 0.1333 = 80 / 600
```



The confusion matrices obtained from pruned and unpruned trees indicate that there is a considerable reduction in false positives and false negatives after pruning. The performance assessment measures for pruned tree are as follows

Accuracy rate: This indicates the number of correctly classified observations and given by =(TN+TP)/N+P

The accuracy of pruned tree is 86.4865%

Error rate: Error rate = 100-Accuracy rate = 13.5135%

Sensitivity: Sensitivity refers to the proportion of positives that are correctly identified

Sensitivity = (True Positive) / (True Positive + False Negative) = 91.6667%

Specificity: Specificity refers to the proportion of negatives that are identified correctly

Specificity = True Negative / (True Negative + False Positive) = 84.127%

The summary() of pruned tree indicates that only 4 predictors atemp, weekday, year and month are used in node splitting. As the tree size is chosen as 7 there are 7 terminal nodes. The tree diagram indicates that the categorical variables are well handled by tree() model. The various levels are coded starting from alphabet a and continues ttill the last level of the variable. For example, 'weekday' has seven levels namely Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday they are encoded as a-Monday, b-Tuesday and so on.

The above analysis of performance measures and tree size and structure indicates that tree model performs better than logistic regression on this dataset. Hence, there is non-linear relationship between response and the predictors.

3. List all the other methods you have learned in this course that can be used for this dataset. For each of those methods, apply it on the training data and then find its prediction performance (prediction accuracy, sensitivity, specificity) on the test data.

Apart from Logistic regression and tree the other classification models learnt from this class are fitted as follows

Linear Discriminant Analysis(LDA)

```
> library(MASS)
> lda.fit=lda(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike,subset=train)
> lda.pred=predict(lda.fit,test.set)
> table(lda.pred$class,count01.test)
      count01.test
       High Low
  High   44  10
  Low     4  53
```

Quadratic Discriminant Analysis(QDA)

```
> qda.fit=qda(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike,subset=train)
> qda.pred=predict(qda.fit,test.set)
> table(qda.pred$class,count01.test)
       count01.test
        High Low
  High    47  19
  Low      1  44
```

K-Nearest Neighbors(KNN)

```
> library(class)
> train.x=cbind(atemp,weekday,month,windspeed,year,hum)[train,]
> test.x=cbind(atemp,weekday,month,windspeed,year,hum)[-train,]
> train.y=count01[train]
> set.seed(1)
> knn.pred=knn(train.x,test.x,train.y,k=5)
> table(knn.pred,count01.test)
         count01.test
knn.pred High Low
     High    42  12
     Low      6  51
```

Maximum Margin Classifier

```
> set.seed(1)
> mm.fit <- svm(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike,subset=train,kernel="linear")
> mm.pred=predict(mm.fit,test.set)
> table(mm.pred,count01.test)
       count01.test
mm.pred High Low
   High    44   9
   Low      4  54
```

Support Vector Classifier (Linear Kernel)

```
> library(e1071)
> set.seed(1)
> svc.tune <- tune(svm,count01~atemp+weekday+month+windspeed+year+hum,data=train.set, kernel="linear", ranges=list(cost=c(0.01,0.1,1,5,10,100,1000)))
> summary(svc.tune)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.1933333

- Detailed performance results:
   cost     error dispersion
1 1e-02 0.1983333 0.06258328
2 1e-01 0.1950000 0.06433420
3 1e+00 0.1950000 0.06851602
4 5e+00 0.1933333 0.06992059
5 1e+01 0.1933333 0.06992059
6 1e+02 0.1933333 0.06992059
7 1e+03 0.1933333 0.06992059

> svc.fit <- svm(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike, subset=train, kernel="linear", cost=5)
> test.pred = predict(svc.fit,test.set)
> table(test.pred, count01.test)
         count01.test
test.pred High Low
     High    42   9
     Low      6  54
```

Support Vector Classifier (Polynomial Kernel)

```
> svmpoly.tune <- tune(svm,count01~atemp+weekday+month+windspeed+year+hum,data=train.set, kernel="polynomial", ranges=list(cost=c(0.01,0.1,1,5,10,100,1000), degree=(1:5)))
> svmpoly.tune$best.parameters
   cost degree
14 1000      2
> svmpoly.fit <- svm(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike, subset=train, kernel="polynomial", cost=1000, degree=2)
> poly.pred = predict(svmpoly.fit,test.set)
> table(poly.pred, count01.test)
         count01.test
poly.pred High Low
     High   41   6
     Low     7  57
```

Support Vector Classifier (Radial Kernel)

```
> svmrad.tune <- tune(svm,count01~atemp+weekday+month+windspeed+year+hum,data=train.set, kernel="radial", ranges=list(cost=c(0.01,0.1,1,5,10,100,1000), gamma=c(0.001,0.01,0.1,0.5,1,2)))
> svmrad.tune$best.parameters
   cost gamma
14 1000  0.01
> svmrad.fit <- svm(count01~atemp+weekday+month+windspeed+year+hum,data=df.bike, subset=train, kernel="radial", cost=1000, gamma=0.01)
> rad.pred = predict(svmrad.fit,test.set)
> table(rad.pred, count01.test)
       count01.test
rad.pred High Low
    High   43   6
    Low     5  57
```

Ensemble learning methods

Bagging

```
> library(randomForest)
> set.seed(1)
> bag.bike=randomForest(count01~., data=df.bike, subset=train, mtry=10, ntree=300, importance=TRUE)
> bag.pred=predict(bag.bike,newdata=test.set)
> table(bag.pred,count01.test)
         count01.test
bag.pred High Low
    High   44   9
    Low     4  54
```

Random Forest

```
> set.seed(1)
> rf.bike=randomForest(count01~., data=df.bike, subset=train, mtry=3, ntree=500, importance=TRUE)
> rf.pred=predict(rf.bike,newdata=test.set)
> table(rf.pred,count01.test)
       count01.test
rf.pred High Low
   High   45  10
   Low     3  53
```

Boosting

```
> library(gbm)
> set.seed(1)
> boost.Bike = gbm((unclass(count01)-1)~.,data=df.bike[train,],distribution="bernoulli",n.trees=5000, interaction.depth=4)
> boost.pred=predict(boost.Bike, newdata=test.set, n.trees=5000)
> pred.class = as.factor(ifelse(boost.pred < 0.5,'High','Low'))
> table(pred.class,count01.test)
           count01.test
pred.class High Low
      High   45  12
      Low     3  51
```

4. Summarize the prediction performance of all methods in a table. Which method is the best? Why?

Method	Prediction accuracy (%)	Sensitivity (%)	Specificity (%)	Error Rate (%)
Logistic regression	11.7117	12.5	11.1111	88.2883
LDA	87.3873	91.6667	84.127	13.5135
QDA	81.982	97.92	69.8413	18.018
KNN	83.7838	87.5	80.9524	16.2162
Maximum Margin	88.288	91.6667	85.7143	11.712
SVM – Linear kernel	86.4865	87.5	85.7143	13.5135
SVM – Polynomial kernel	88.288	85.4167	90.476	11.712
SVM – Radial kernel	90.09	89.5833	90.476	9.91
Bagging	88.288	91.6667	85.7143	11.712
Random Forest	88.288	93.75	84.127	11.712
Boosting	86.4865	93.75	80.9524	13.5135
Tree (Unpruned)	81.982	83.333	80.9524	18.018
Tree (Pruned)	86.4865	87.5	85.7143	13.5135

Best model in terms of Prediction accuracy

From the performance assessment measures tabulated above, Support Vector Machine with Radial Kernel turns out to be the best in terms of prediction accuracy for the following as the error rate of 9.91 is the minimum test error rate obtained among all the other models. Thus, this model has the highest prediction accuracy.
It also has comparatively better sensitivity and specificity.

The SVM models with various kernels are approximately similar in prediction performance. SVM models are closely followed by ensemble learning methods such as bagging, random forest and boosting. All three of them are good in prediction.

Best model in terms of Interpretability

Pruned tree is the best model in terms of interpretability as the tree structure can be easily explained to anyone. It also clearly says the important/significant predictors that plays a major role in bike rentals. The prediction accuracy, sensitivity and specificity are also at reasonably high levels to make predictions.
LDA, QDA, KNN have similar error rates compared to pruned tree. But, pruned tree tops all of them in terms of interpretability.

Conclusion:

As model interpretability is as important as predicting results in this case and the error rates are not significantly different between SVM radial kernel and pruned tree I would conclude that ***pruned tree*** is the best model in terms of prediction accuracy as well as model interpretability.