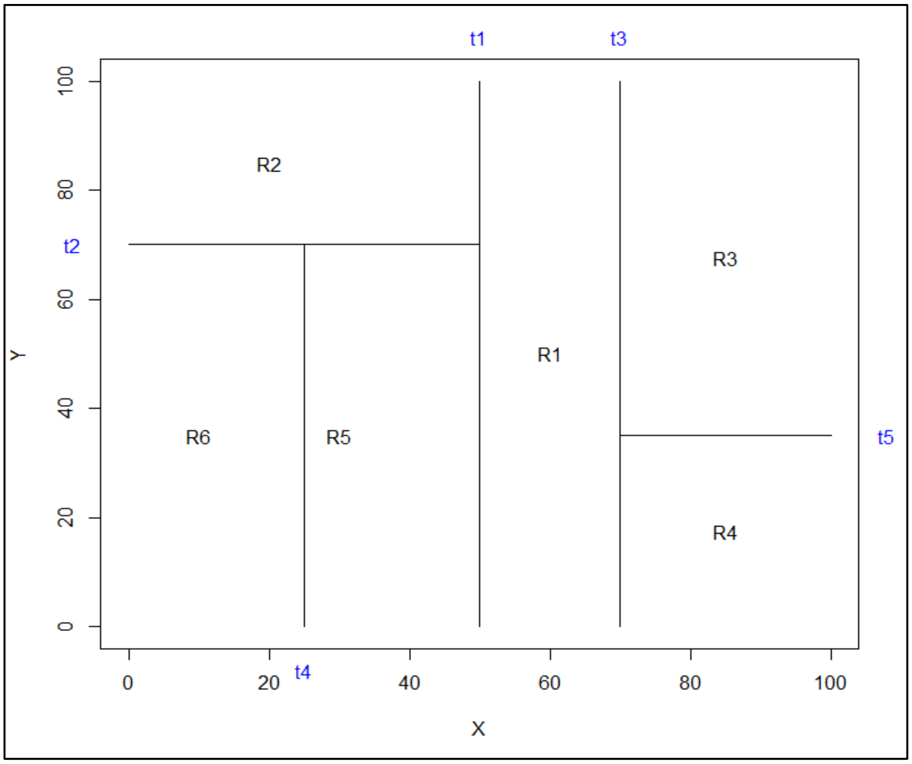
ASSIGNMENT 3

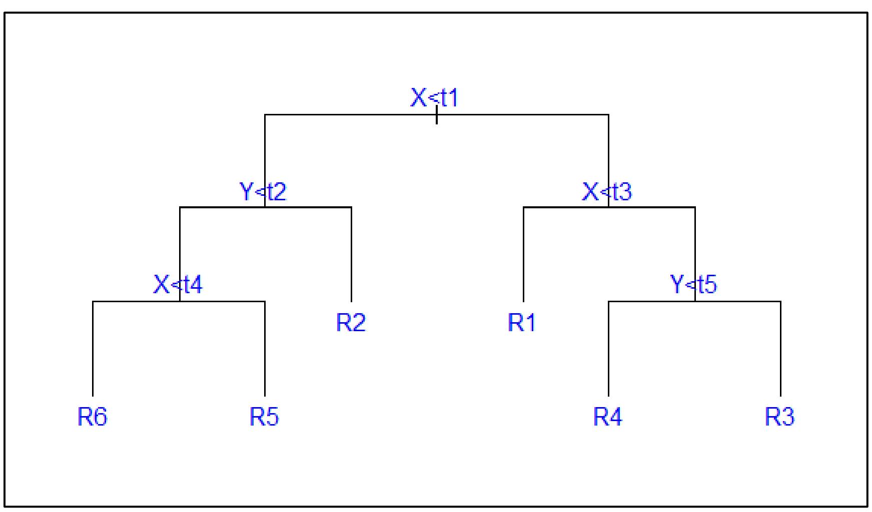
Ghorpade, Mrunal Mahesh
Mghorp2@uic.edu
UIN:677441117

1) Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions $R1, R2, \ldots$, the cutpoints $t1, t2, \ldots$, and so forth.

<u>Answer:</u> Following is the partition of 2-dimensional feature space that could result from recursive binary splitting:

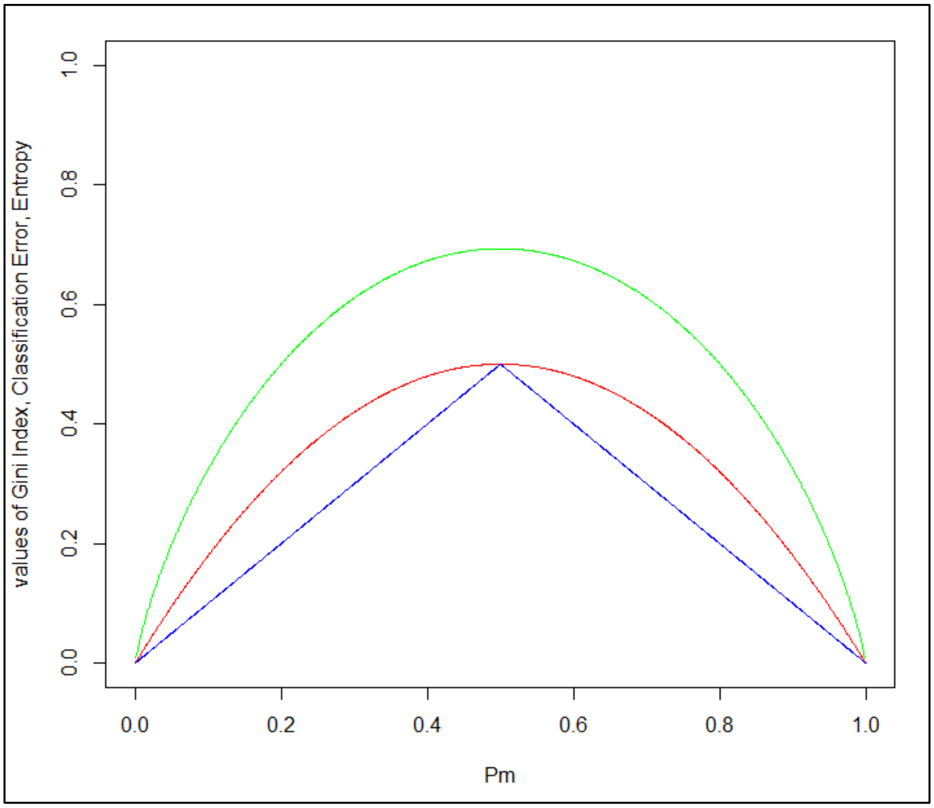


Decision tree corresponding to the above partition is given below;



2) Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of ˆ*pm*1. The *x* axis should display ˆ*pm*1, ranging from 0 to 1, and the *y*-axis should display the value of the Gini index, classification error, and entropy.

<u>Answer:</u> Below is the plot of Pm Vs Values of Gini index, classification error, and entropy



3) In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

(a) Split the data set into a training set and a test set. (Please Refer to “Problem 8” R code)

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

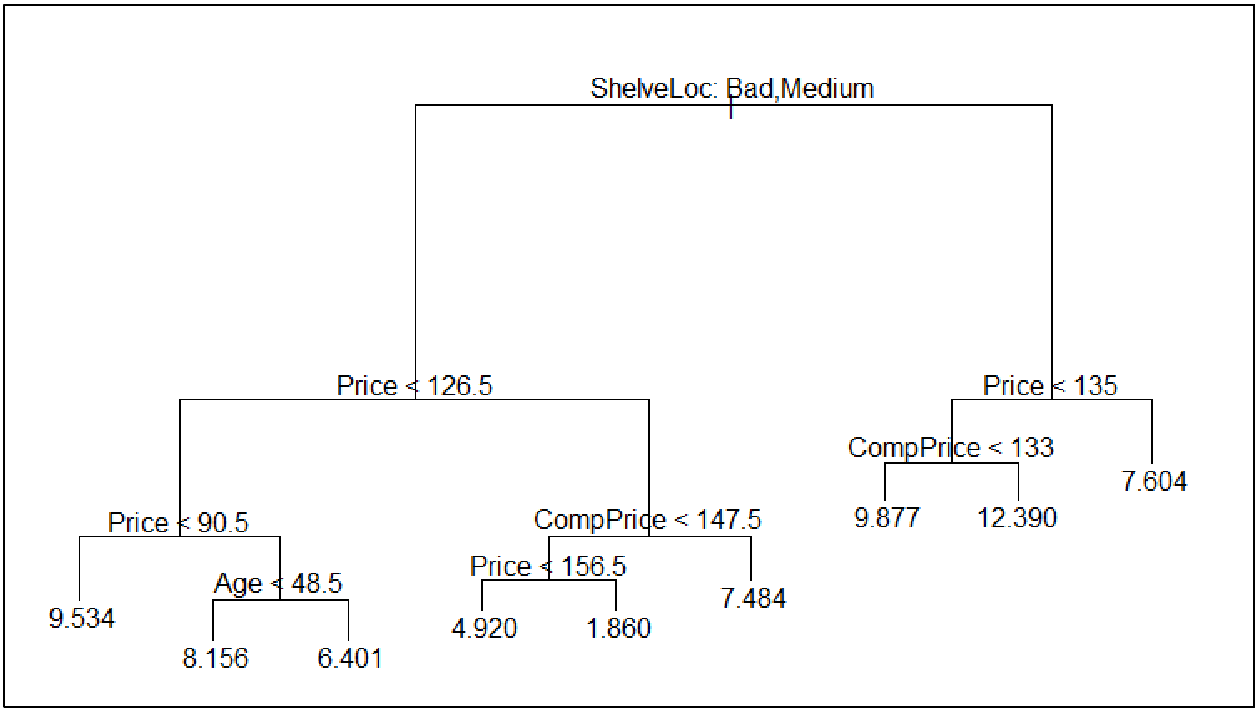
<u>Observation:</u>

Following is the summary of Regression Tree on Training Data;

```
> summary(tree.cs)

Regression tree:
tree(formula = Sales ~ ., data = cs_train)
Variables actually used in tree construction:
[1] "ShelveLoc"    "Price"        "Age"          "Advertising" "CompPrice"
Number of terminal nodes:  15
Residual mean deviance:  2.049 = 379 / 185
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.0240 -0.9870 -0.1052  0.0000  0.8988  3.8840
```

Below in the Regression tree obtained:



For Carseats Data, a regression tree for predicting the log Sales of car seats at different locations, based on number of parameters i.e. predictor variables in the data e.g. Shelveloc: the quality of the shelving location for the car seats at each site, Price: Price charged by the company, Age: Average age of local population, CompPrice: price charged by competitors etc. At a given node the lable indicates the left-hand branch emanating from the split e.g. Split on the first node is on variable “ShelveLoc”, where ShelveLoc: Bad, Medium means the left-hand branch contaings the observations corresponding to ShelveLoc = Bad/Medium whereas the right-hand branch contains the data containing ShelveLoc = Good. The tree has 8 internal nodes and 9 leaf nodes. The number in each leaf is the mean of the Sales for the observations that fall there.

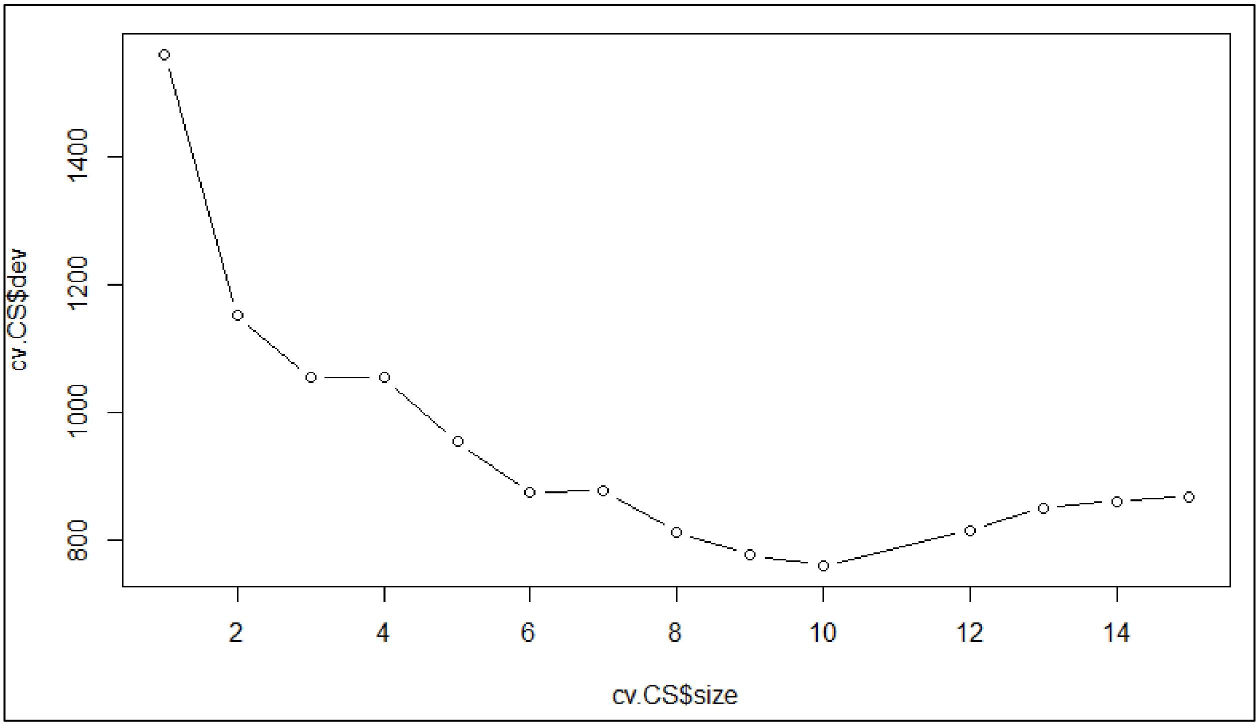
The Test MSE obtained is 4.325436

```
> MSE<-mean((yhat - cs_test$Sales)^2)
> MSE
[1] 4.325436
```

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

<u>Observations:</u>

After applying Cross Validation we get that optimal size of the tree should be 10 as shown in the below plot.



After applying pruning to get 10 node tree. We get MSE as 5.077. So, pruning increased the Test MSE.

```
> prune_MSE
[1] 5.07767
```

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

<u>Observations</u>:

Bagging decreases Test MSE to 3.0008 . The two most important variables are “Price” and “ShelveLoc” as shown below;

```
> mean((yhat.Bag - CS_test$Sales)^2)
[1] 3.000802
> importance(Bag.CS)
                %IncMSE IncNodePurity
CompPrice    22.0063590    143.685939
Income        0.8152944     64.309405
Advertising 15.8275932    108.053536
Population  -1.4634587     63.445421
Price       55.8979195    470.763139
ShelveLoc   58.3810079    454.585322
Age          9.3077279    112.626206
Education   -3.1737379     38.775418
Urban        1.1014470      6.856806
US           5.5143471     14.523398
```

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of *m*, the number of variables considered at each split, on the error rate obtained.

Random Forest decreases Test MSE to 3.35 but bagging decreases more. Here as well the two most important variables are “Price” and “ShelveLoc”.

Below results obtained are with m=sqrt(# of variables) which is approximately 3

```
> MSE
[1] 3.351339
> importance(RF.CS)
                %IncMSE IncNodePurity
CompPrice   10.8484860     138.96741
Income       3.3534745     105.91710
Advertising 11.6210690     137.13468
Population  -0.4625651     103.28424
Price       32.9653600     337.12382
ShelveLoc   40.7983769     353.61898
Age          8.9127052     140.30246
Education   -0.3947651      60.43430
Urban       -0.8865703      10.67028
US           6.7901998      30.04198
```

As m is increased the test error rate decreases till certain limit and it increases again.

m	MSE
2	3.835014
3	3.35133
5	3.00397
8	2.932019
9	2.940827
10	2.96737

4) We now use boosting to predict Salary in the Hitters data set.

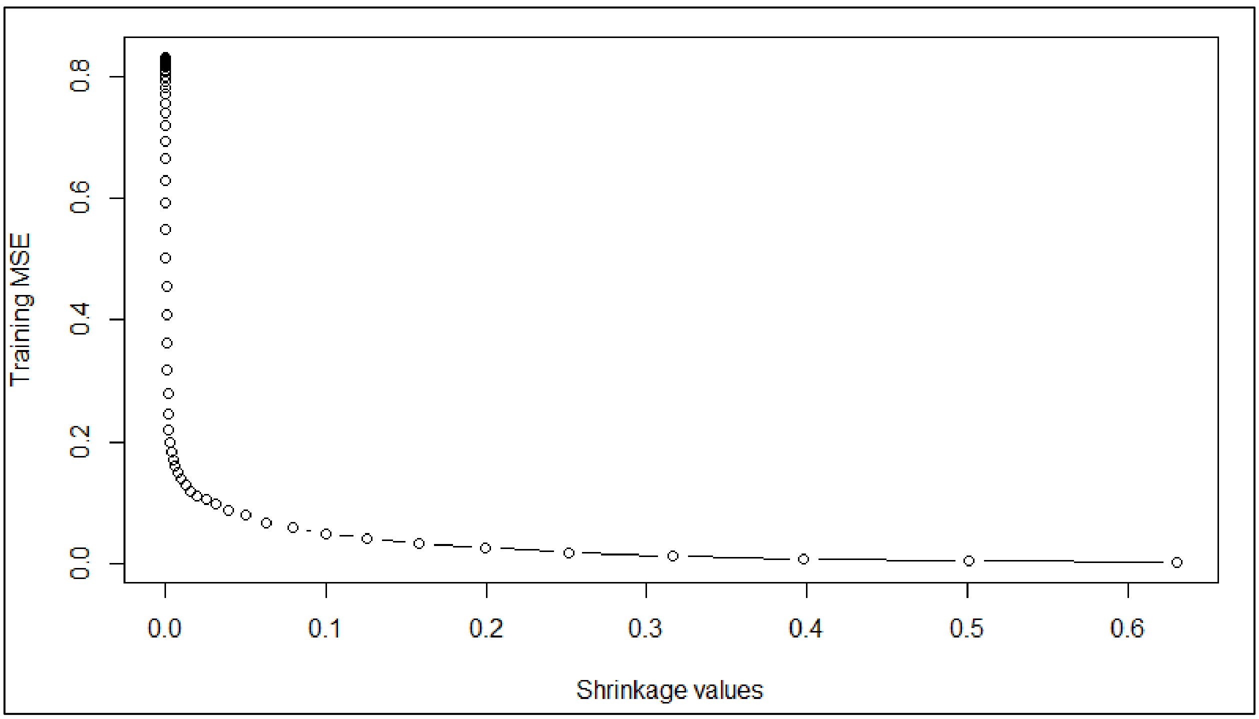
(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries. (Please Refer to "Problem 10" R code)

(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations. (Please Refer to "Problem 10" R code)

(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

<u>Observations</u>:

Below is the Plot of Different Shrinkage values Vs Training MSE;



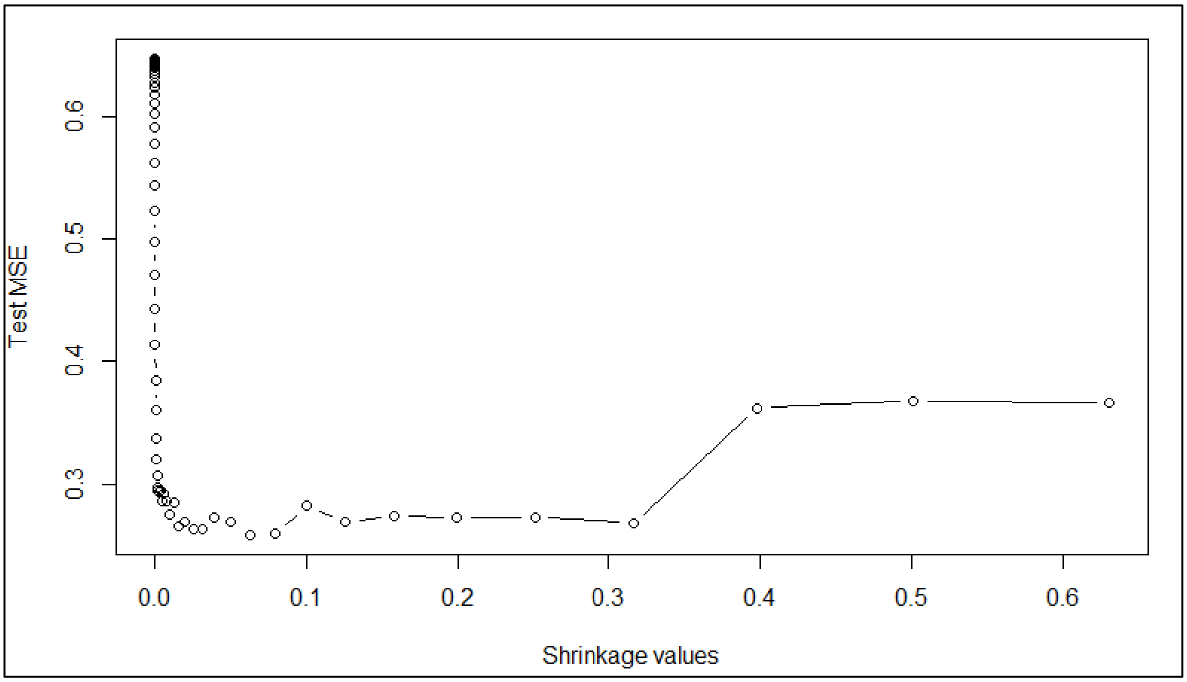
We get minimum Training Error 0.00208 at lambda = 0.63 as shown below;

```
> Min_Train
[1] 0.002085409
> Train_lambda<-lambda[which.min(as.numeric(Train_MSR))]
> Train_lambda
[1] 0.6309573
```

(d) Produce a plot with different shrinkage values on the *x*-axis and the corresponding test set MSE on the *y*-axis.

<u>Observations</u>:

Below is the Plot of Different Shrinkage values Vs Test MSE;



We get minimum Test Error 0.258 at lambda = 0.063 as shown below;

```
> Min_Test
[1] 0.2583995
> Test_lambda<-lambda[which.min(as.numeric(Test_MSR))]
> Test_lambda
[1] 0.06309573
```

(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

<u>Observations</u>:

Applying Ridge Regression on the Test data we get MSE as follows;

```
> ridge_MSE_test
[1] 0.4567626
```

Applying Linear Regression on the Test data we get MSE as follows;

```
> lm_MSE
[1] 0.4917959
```

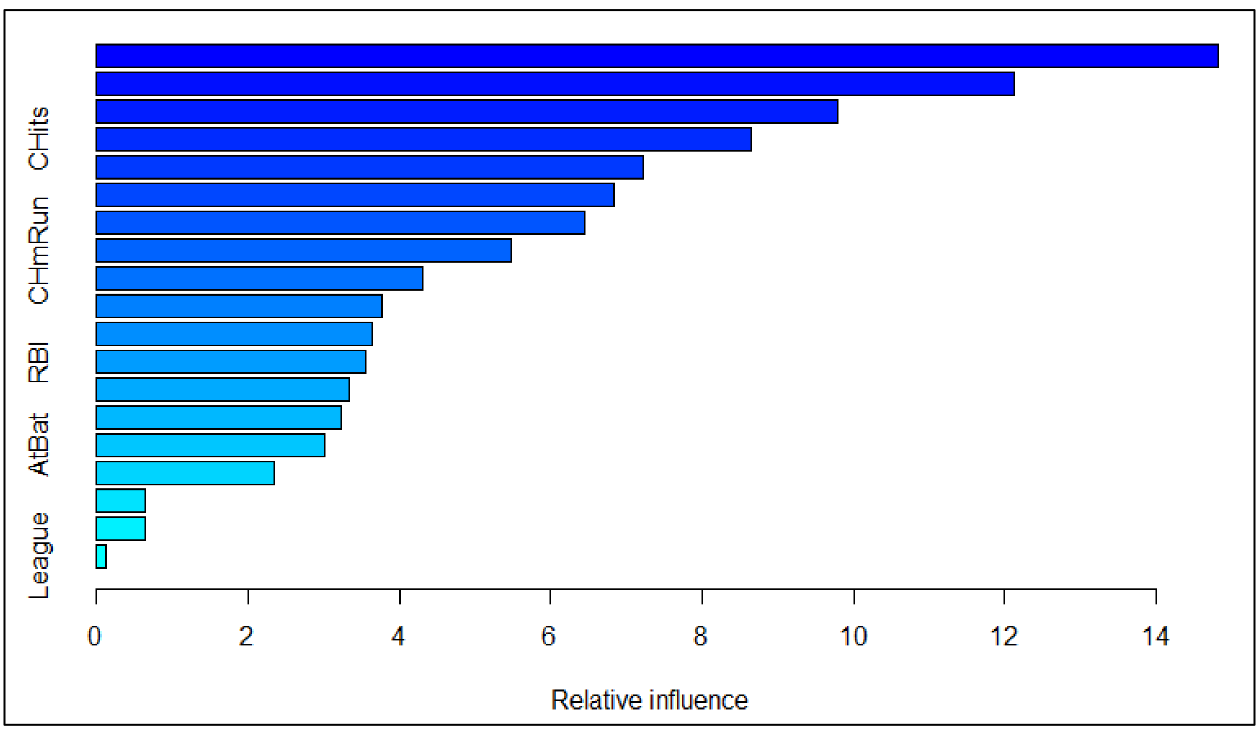
Both the approaches give higher MSE on Test Data as compared with Boosting

(f) Which variables appear to be the most important predictors in the boosted model?

<u>Observation</u>:

From the below summary we can see that CAtBat, CRBI, CWalks, Chits are the most important variables in the boosted model

```
> summary(boost.hit)  #to see which variables are important
                var    rel.inf
CAtBat       CAtBat 14.8197925
CRBI           CRBI 12.1186597
CWalks       CWalks  9.7998044
CHits         CHits  8.6560652
PutOuts     PutOuts  7.2138695
Walks         Walks  6.8368300
Years         Years  6.4412453
CHmRun       CHmRun  5.4742375
Assists     Assists  4.3071056
Hits           Hits  3.7625714
Runs           Runs  3.6334822
RBI             RBI  3.5479456
HmRun         HmRun  3.3512231
CRuns         CRuns  3.2227444
AtBat         AtBat  3.0281593
Errors       Errors  2.3590083
Division   Division  0.6475954
NewLeague NewLeague  0.6438269
League       League  0.1358334
```



(g) Now apply bagging to the training set. What is the test set MSE for this approach?

Observation:

After Bagging is applied we get following MSE;

```
> MSE_bag
[1] 0.2318283
```

Bagging gives lesser MSE than Boosting.