MLA REVIEW 2

```r
#BY
#SHRENI AGRAWAL    18BLC1012
#S HARSHAAVARDHINI 18BLC1053
#RITIKA          18BLC1027
library(ElemStatLearn)
```

```
## Warning: package 'ElemStatLearn' was built under R version 3.6.2
```

```r
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 3.6.2
```

```r
library(gdata)
```

```
## Warning: package 'gdata' was built under R version 3.6.2
```

```
## gdata: Unable to locate valid perl interpreter
## gdata:
## gdata: read.xls() will be unable to read Excel XLS and XLSX files
## gdata: unless the 'perl=' argument is used to specify the location of a
## gdata: valid perl intrpreter.
## gdata:
## gdata: (To avoid display of this message in the future, please ensure
## gdata: perl is installed and available on the executable search path.)
```

```
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLX' (Excel 97-2004) files.
```

```
##
```

```
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.
```

```
##
```

```
## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
## gdata: libaries needed to support Excel XLS and XLSX formats.
```

```
##
## Attaching package: 'gdata'
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

```
## The following object is masked from 'package:utils':
##
##     object.size
```

```r
library(MASS)
library(splines)
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.6.2
```

```r
library (randomForest )
```

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```r
library (caTools )
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```
## Warning: package 'randomForest' was built under R version 3.6.2
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gdata':
##
##     combine
```

```r
library (gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.2
```

```
## Loaded gbm 2.1.5
```

```r
library (e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

```r
fix(ElemStatLearn)
names(marketing)
```

```
##  [1] "Income"      "Sex"         "Marital"     "Age"         "Edu"
##  [6] "Occupation"  "Lived"       "Dual_Income" "Household"   "Householdu18"
## [11] "Status"      "Home_Type"   "Ethnic"      "Language"
```

```r
?marketing
```

```
## starting httpd help server ...
```

```
##  done
```

```
##
## Call:
## svm(formula = Income ~ ., data = train_marketing, kernal = "linear",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.07692308
##     epsilon:  0.1
##
##
## Number of Support Vectors:  6286
```

```
market=marketing[ ! marketing$Marital % in% c( NA), ]
market=market[ ! market$Edu % in% c( NA), ]
market=market[ ! market$Occupation % in% c( NA), ]
market=market[ ! market$Lived % in% c( NA), ]
market=market[ ! market$Household % in% c( NA), ]
market=market[ ! market$Status % in% c( NA), ]
market=market[ ! market$Home_Type % in% c( NA), ]
market=market[ ! market$Ethnic % in% c( NA), ]
market=market[ ! market$Language % in% c( NA), ]
dim(market)
```

```
## [1] 6876   14
```

```
#SUPPORT VECTOR MACHINES
attach (market)
n <- nrow(market)
ntrain <- round(n* 0.999)
set.seed( 1110)
tindex <- sample(n, ntrain)
train_marketing <- market[tindex,]
test_marketing <- market[-tindex,]
svm1 <- svm(Income~., data=train_marketing, kernal= "linear", cost= 10,scale= FALSE )
summary(svm1)
```

```
plot(svm1, test_marketing, Income ~ Edu)
test.svm1<-predict(svm1,test_marketing)
table(predict=test.svm1,truth=test_marketing$Income)
```

```
##                   truth
## predict            1 2 3 6 7 8 9
##   1.17644058475668 1 0 0 0 0 0 0
##   1.27153900451675 0 1 0 0 0 0 0
##   2.80871367276248 0 0 1 0 0 0 0
##   5.61626628893989 0 0 1 0 0 0 0
##   5.80972194327446 0 0 0 1 0 0 0
##   6.34970524633368 0 0 0 0 0 1 0
##   6.90008389724613 0 0 0 0 0 0 1
```

```
(8000+14000+18000+73000+80000)/(8000+14000+30000+40000+18000+73000+80000)
```

```
## [1] 0.7338403
```

```
#SPLINES
agelims=range(Edu)
Age.grid=seq(from=agelims[1],to=agelims[2])
fit=lm(Income~bs(Edu,knots=c(25,40,60)),data=market)
pred=predict(fit,newdata=list(Edu=Age.grid),se=T)
```

```
## Warning in predict.lm(fit, newdata = list(Edu = Age.grid), se = T): prediction
## from a rank-deficient fit may be misleading
```

```
plot(Edu,Income,col="gray")
lines(Age.grid,pred$fit,lwd=2)
lines(Age.grid,pred$fit+2*pred$se,lty="dashed")
lines(Age.grid,pred$fit-2*pred$se,lty="dashed")
dim(bs(Edu,knots=c(25,40,60)))
```

```
## [1] 6876    6
```

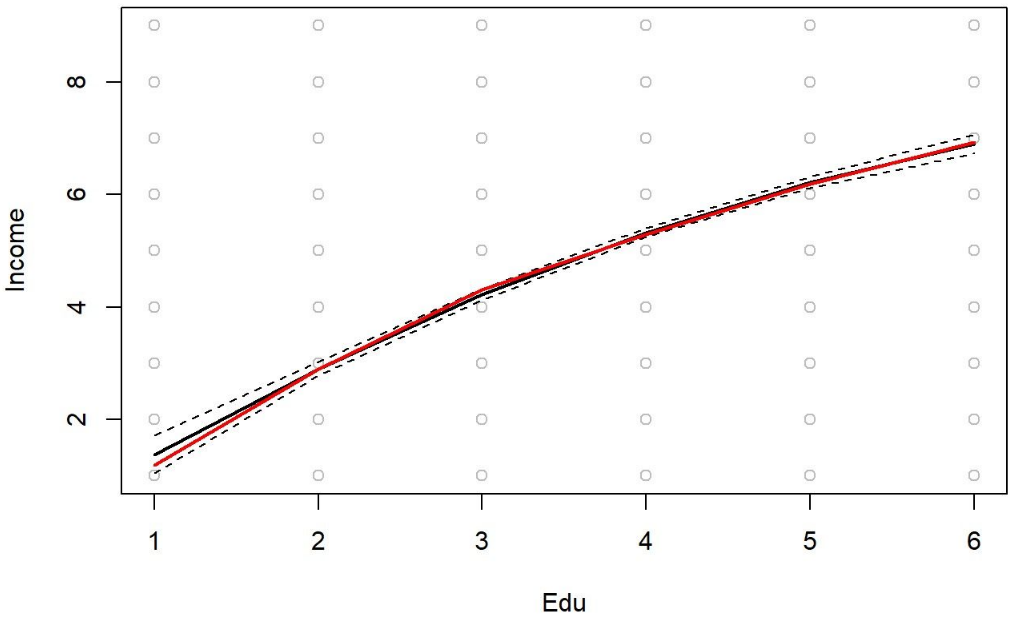
```
dim(bs(Edu,df=6))
```

```
## [1] 6876    6
```

```
attr(bs(Edu,df=6),"knots")
```

```
## 25% 50% 75%
##   3   4   5
```

```
fit2=lm(Income~ns(Edu,df=4),data=market)
pred2=predict(fit2,newdata=list(Edu=Age.grid),se=T)
lines(Age.grid, pred2$fit,col="red",lwd=2)
```




```
plot(Edu,Income,xlim=agelims,cex= .5,col= "darkgrey" )
title("Smoothing Spline" )
fit=smooth.spline(Edu,Income,df= 5)
fit2=smooth.spline(Edu,Income,cv= TRUE )
```

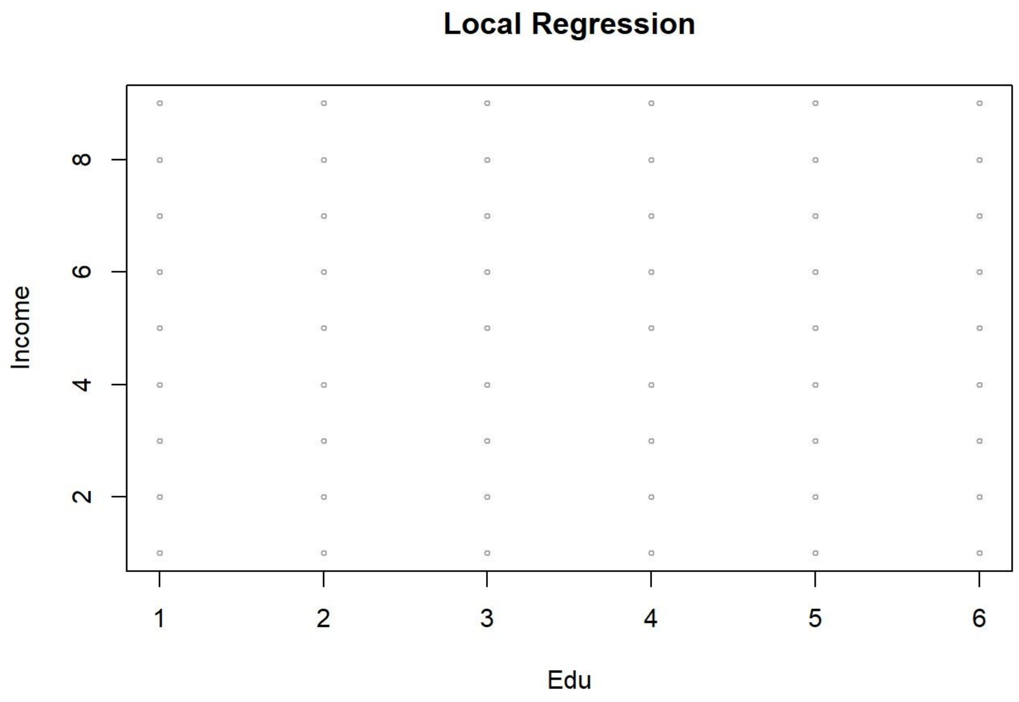
```
## Warning in smooth.spline(Edu, Income, cv = TRUE): cross-validation with non-
## unique 'x' values seems doubtful
```

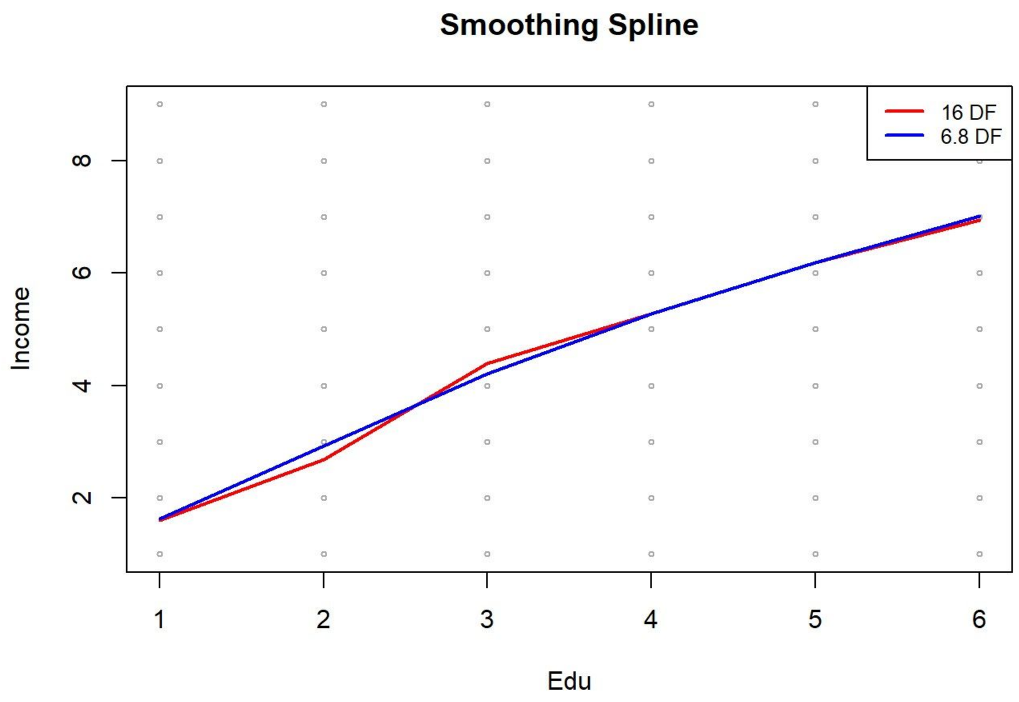
```
fit2$df
```

```
## [1] 2.997699
```

```
lines(fit,col= "red",lwd=2)
lines(fit2,col= "blue",lwd=2)
legend( "topright",legend=c( "16 DF" ,"6.8 DF" ),col=c ("red","blue"),lty=1,lwd=2,cex= .8)
```

```
plot(Edu,Income,xlim=agelims,cex= .5,col= "darkgrey" )
title("Local Regression" )
```






```
#TREES


#Skeleton of tree
High=ifelse(Income<=   2,"No","Yes" )
market=data.frame(market,High)
tree.market=tree(High~.-Income,market)
summary(tree.market)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Income, data = market)
## Variables actually used in tree construction:
## [1] "Age"        "Occupation" "Marital"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7186 = 4934 / 6867
## Misclassification error rate: 0.1878 = 1291 / 6876
```

```
fit=loess(Income~Edu,span=    .2,data=market)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 0.975
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 2.025
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 6.8095e-015
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1.0506
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger
```

```
fit2=loess(Income~Edu,span=    .5,data=market)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 6.025
```

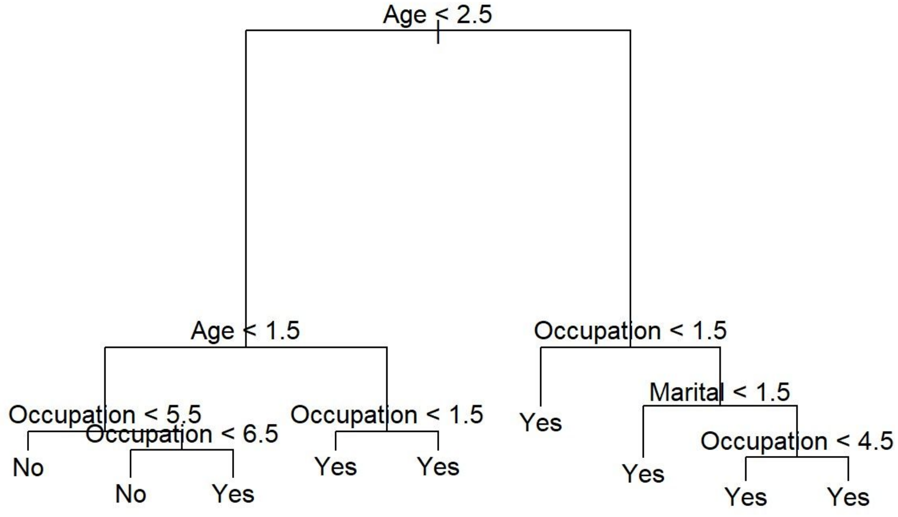
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 2.025
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.656e-014
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1
```

```
plot(tree.market)
text(tree.market,pretty= 0)
```

```
##         High.test
## tree.pred   No  Yes
##       No  1076  680
##       Yes  319 3301
```



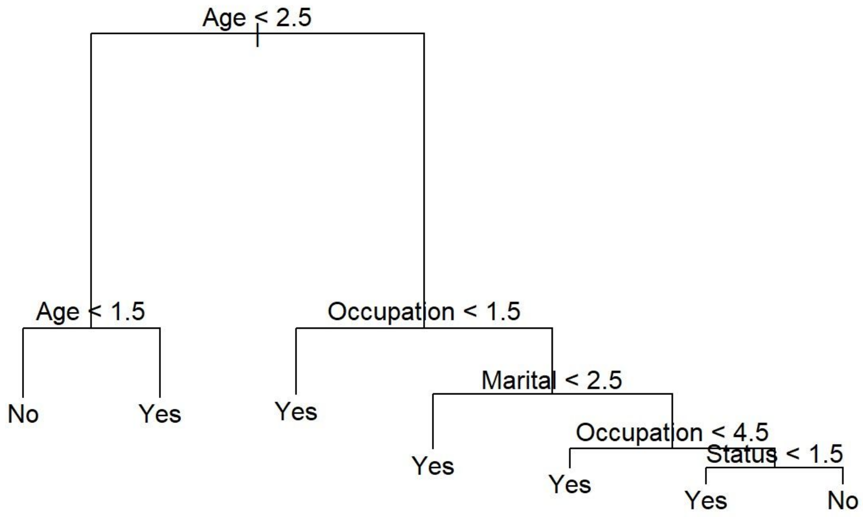

```
tree.market
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 6876 7873.00 Yes ( 0.259453 0.740547 )
##    2) Age < 2.5 2257 3093.00 No ( 0.562694 0.437306 )
##      4) Age < 1.5 647  476.20 No ( 0.879444 0.120556 )
##        8) Occupation < 5.5 110  150.70 No ( 0.563636 0.436364 ) *
##        9) Occupation > 5.5 537  231.40 No ( 0.944134 0.055866 )
##         18) Occupation < 6.5 483   25.94 No ( 0.995859 0.004141 ) *
##         19) Occupation > 6.5 54   74.79 Yes ( 0.481481 0.518519 ) *
##      5) Age > 1.5 1610 2205.00 Yes ( 0.435404 0.564596 )
##       10) Occupation < 1.5 277  264.60 Yes ( 0.184116 0.815884 ) *
##       11) Occupation > 1.5 1333 1847.00 Yes ( 0.487622 0.512378 ) *
##    3) Age > 2.5 4619 3226.00 Yes ( 0.111279 0.888721 )
##      6) Occupation < 1.5 2050  542.00 Yes ( 0.029268 0.970732 ) *
##      7) Occupation > 1.5 2569 2396.00 Yes ( 0.176722 0.823278 )
##       14) Marital < 1.5 1413  717.20 Yes ( 0.070064 0.929936 ) *
##       15) Marital > 1.5 1156 1426.00 Yes ( 0.307093 0.692907 )
##         30) Occupation < 4.5 679  651.50 Yes ( 0.185567 0.814433 ) *
##         31) Occupation > 4.5 477  660.50 Yes ( 0.480084 0.519916 ) *
```

```
#tree

train=sample( 1:nrow(market), 1500 )
market.test=market[-train,]
High.test=High[-train]
tree.market=tree(High~.-Income,market,subset=train)
tree.pred=predict(tree.market,market.test,type= "class" )
table(tree.pred,High.test)
```

```
#pruned tree
prune.market=prune.misclass(tree.market,best= 5)
plot(prune.market)
text(prune.market,pretty= 0)
```




```
tree.pred=predict(prune.market,market.test,type= "class" )
table(tree.pred,High.test)
```

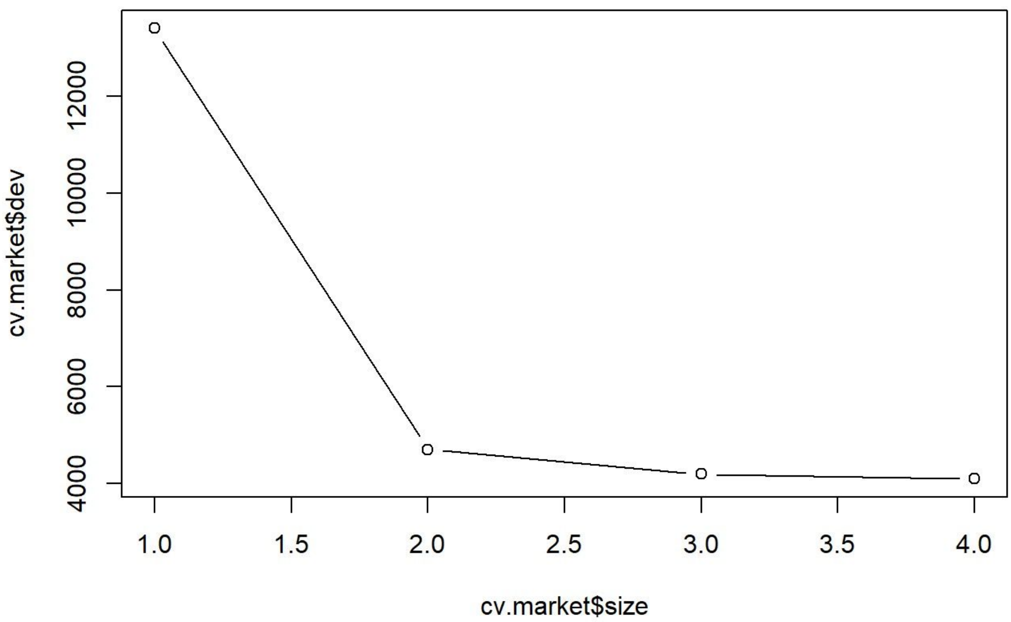
```
##         High.test
## tree.pred   No  Yes
##       No   582  149
##       Yes  813 3832
```

```
#Fitting Regression tree
set.seed( 1)
train = sample( 1:nrow(market), nrow(market)/ 4)
tree.market=tree(Income~.,market,subset=train)
summary(tree.market)
```

```
##
## Regression tree:
## tree(formula = Income ~ ., data = market, subset = train)
## Variables actually used in tree construction:
## [1] "High"       "Status"     "Occupation"
## Number of terminal nodes:  4
## Residual mean deviance:  2.304 = 3951 / 1715
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.6430 -0.7615 -0.2760  0.0000  1.2380  3.2380
```

```
plot(tree.market)
text(tree.market,pretty= 0)
```

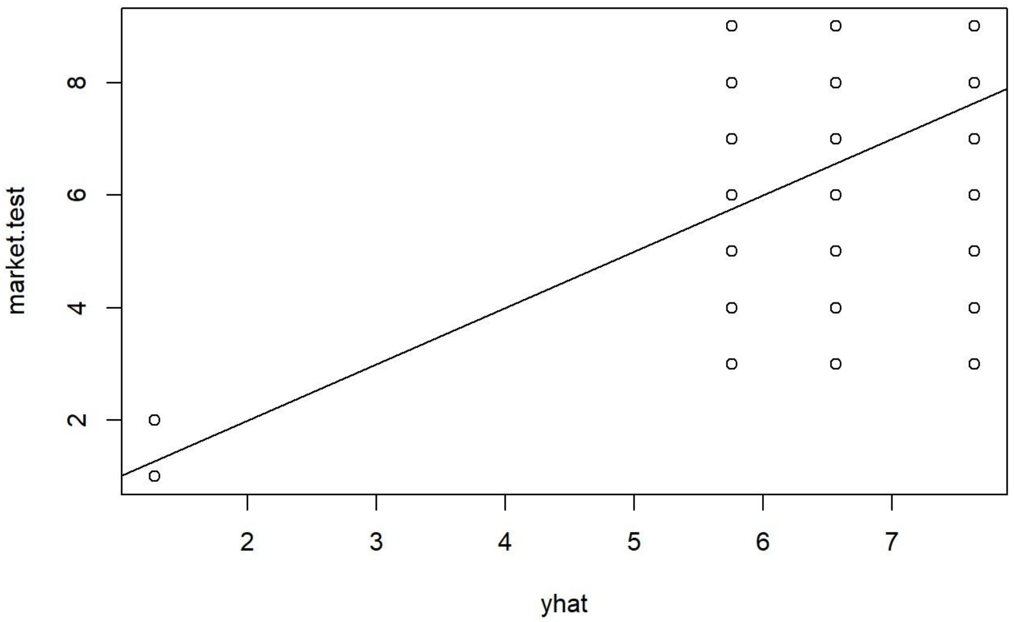

```
cv.market=cv.tree(tree.market)
plot(cv.market$size,cv.market$dev,type='b')
```




```
prune.market=prune.tree(tree.market,best=4)
plot(prune.market)
text(prune.market,pretty=0)
```

```
##
## Call:
##  randomForest(formula = Income ~ ., data = market, mtry = 5, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 1.906525
##                     % Var explained: 75.51
```

```
yhat=predict(tree.market,newdata=market[-train,])
market.test=market[-train, "Income"]
plot(yhat,market.test)
abline(0,1)
```




```
mean((yhat-market.test)^ 2)
```

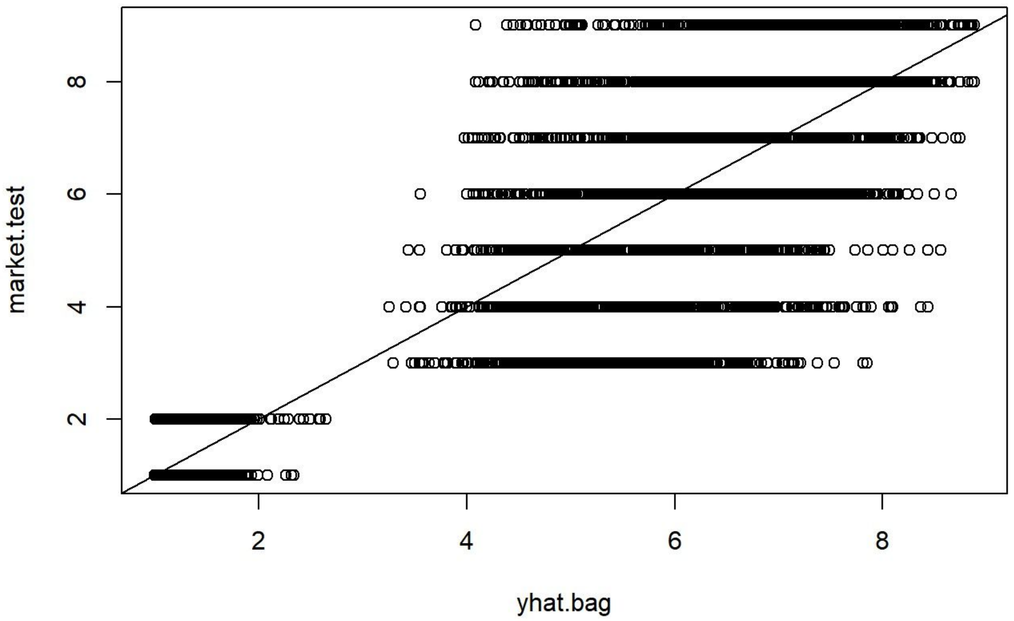
```
## [1] 2.256684
```

```
#Bagging and random forest


bag.market=randomForest(Income~.,data=market,subset=train,mtry=      5,importance= TRUE  )
bag.market
```

```
yhat.bag = predict(bag.market,newdata=market[-train,])
plot(yhat.bag, market.test)
abline(0,1)
```

```
##            %IncMSE IncNodePurity
## Sex         5.015431     118.09708
## Marital     27.693208    1033.08852
## Age         27.214169    1050.79776
## Edu         23.787378     822.12668
## Occupation  31.496948     965.73112
## Lived        2.474272     187.80125
## Dual_Income 22.604311     758.48066
## Household   16.678051     344.22150
## Householdu18 12.654730    189.01943
## Status      24.570615     780.20730
## Home_Type   22.034722     374.66261
## Ethnic       5.960532     249.40511
## Language     2.528773      86.04535
## High        73.990797    4163.13558
```




```
mean((yhat.bag-market.test)^ 2)
```

```
## [1] 1.856304
```

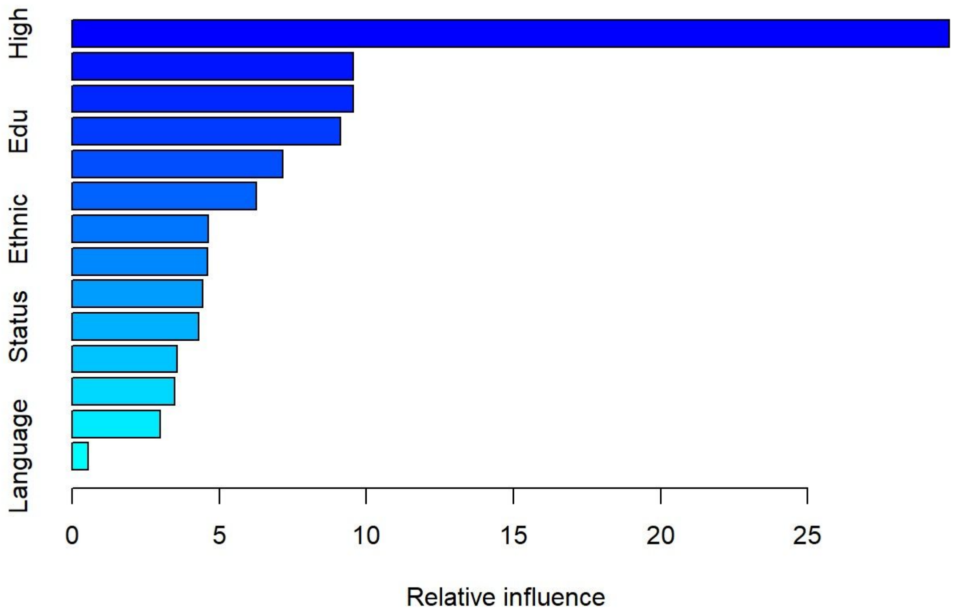
```
bag.market=randomForest(Income~.,data=market,subset=train,mtry= 5,ntree= 25)
yhat.bag = predict(bag.market,newdata=market[-train,])
mean((yhat.bag-market.test)^ 2)
```

```
## [1] 1.909137
```

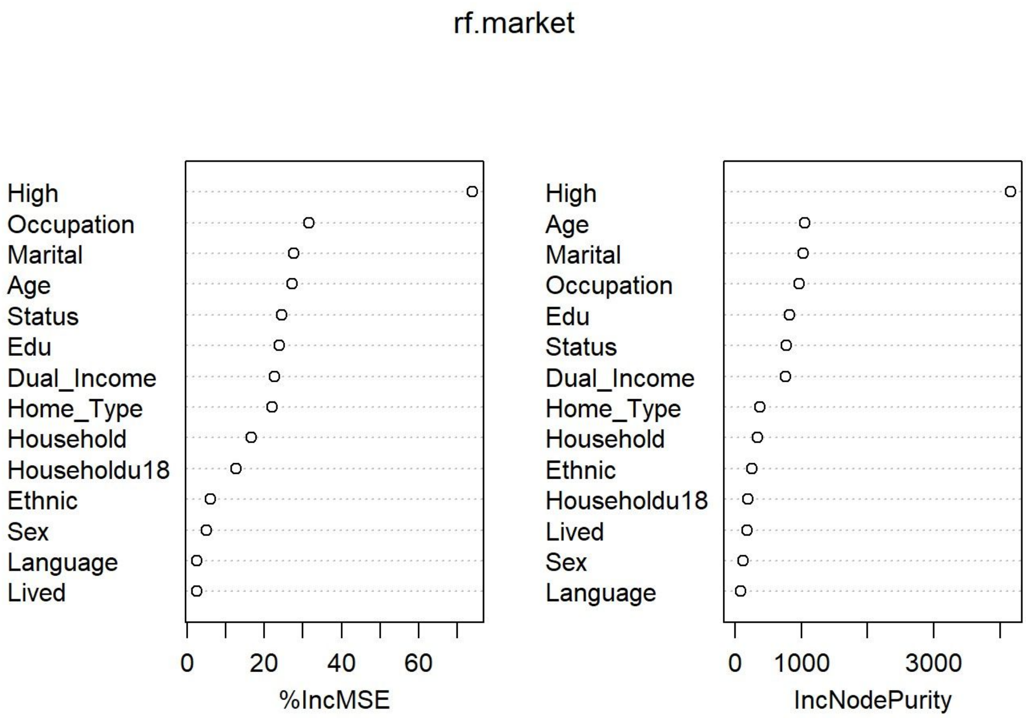
```
rf.market=randomForest(Income~.,data=market,subset=train,mtry= 2,importance= TRUE )
yhat.rf = predict(rf.market,newdata=market[-train,])
mean((yhat.rf-market.test)^ 2)
```

```
## [1] 1.939694
```

```
importance(rf.market)
```




```
varImpPlot(rf.market)
```



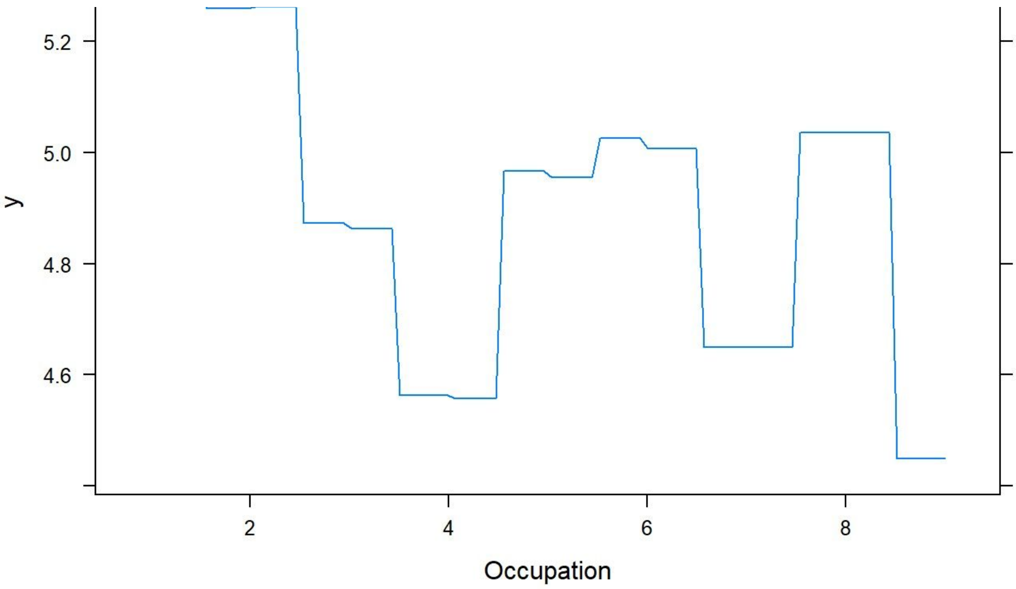

```
#Boosting

boost.market=gbm(Income~.,data=market[train,],distribution=  "gaussian" ,n.trees= 5000 ,interaction.depth= 4)
summary(boost.market)
```

```
plot(boost.market,i= "Age")
```

```
##                    var    rel.inf
## High              High 29.7997106
## Age                Age  9.5540772
## Occupation  Occupation  9.5524459
## Edu                Edu  9.1215449
## Household    Household  7.1557530
## Marital        Marital  6.2572653
## Ethnic          Ethnic  4.6346858
## Lived            Lived  4.6012460
## Home_Type    Home_Type  4.4309989
## Status          Status  4.2904512
## Sex                Sex  3.5765837
## Dual_Income Dual_Income  3.4734714
## Householdu18 Householdu18  2.9908868
## Language      Language  0.5608793
```

```
par(mfrow=c( 1,2))
plot(boost.market,i= "Occupation" )
```



```
yhat.boost=predict(boost.market,newdata=market[-train,],n.trees= 5000)
mean((yhat.boost-market.test)^ 2)
```

```
## [1] 2.3222
```

```
boost.market=gbm(Income~.,data=market[train,],distribution= "gaussian" ,n.trees= 5000,interaction.depth= 4,shrinkage= 0.2,verbose= F)
yhat.boost=predict(boost.market,newdata=market[-train,],n.trees= 5000)
mean((yhat.boost-market.test)^ 2)
```

```
## [1] 2.714409
```