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
```

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

```r
library(caTools)
```

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

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

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

```r
library(randomForest)
```

```
## 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
```

```r
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
```

```r
# NEURAL NETWORK

library(ElemStatLearn)
library(neuralnet)
m<- marketing[complete.cases(marketing),]
dim(m)
```

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

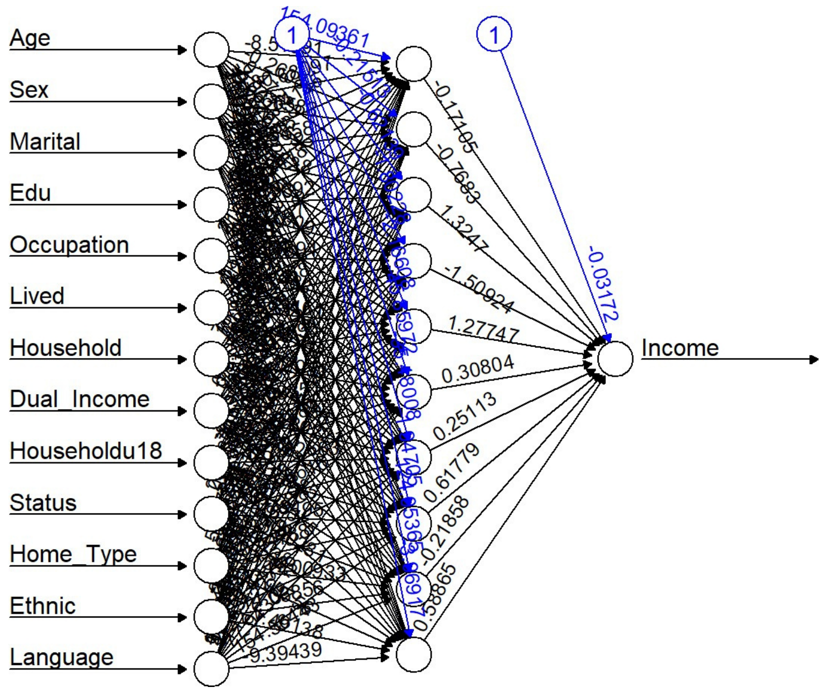
```r
set.seed(9999)
str(m)
```

```
## 'data.frame':    6876 obs. of 14 variables:
## $ Income      : int  9 9 1 1  8 1 6 2  4 1 ...
## $ Sex         : int  1 2 2 2  1 1 1 1  1 1 ...
## $ Marital     : int  1 1 5 5  1 5 3 1  1 5 ...
## $ Age         : int  5 3 1 1  6 2 3 6  7 2 ...
## $ Edu         : int  5 5 2 2  4 3 4 3  4 4 ...
## $ Occupation  : int  5 1 6 6  8 9 3 8  8 9 ...
## $ Lived       : int  5 5 5 3  5 4 5 5  4 5 ...
## $ Dual_Income : int  3 2 1 1  3 1 1 3  3 1 ...
## $ Household   : int  5 3 4 4  2 3 1 3  2 1 ...
## $ Householdu18: int  2 1 2 2  0 1 0 0  0 0 ...
## $ Status      : int  1 2 3 3  1 2 2 2  2 2 ...
## $ Home_Type   : int  1 3 1 1  1 3 3 3  3 3 ...
## $ Ethnic      : int  7 7 7 7  7 7 7 7  7 7 ...
## $ Language    : int  1 1 1 1  1 1 1 1  1 1 ...
```

```r
m$Income=(m$Income-min(m$Income))/(max(m$Income)-min(m$Income))
m$Age=(m$Age-min(m$Age))/(max(m$Age)-min(m$Age))
m$Edu=(m$Edu-min(m$Edu))/(max(m$Edu)-min(m$Edu))
m$Sex=(m$Sex-min(m$Sex))/(max(m$Sex)-min(m$Sex))
m$Marital=(m$Marital-min(m$Marital))/(max(m$Marital)-min(m$Marital))
m$Occupation=(m$Occupation-min(m$Occupation))/(max(m$Occupation)-min(m$Occupation))
m$Lived=(m$Lived-min(m$Lived))/(max(m$Lived)-min(m$Lived))
m$Dual_Income=(m$Dual_Income-min(m$Dual_Income))/(max(m$Dual_Income)-min(m$Dual_Income))
m$Household=(m$Household-min(m$Household))/(max(m$Household)-min(m$Household))
m$Householdu18=(m$Householdu18-min(m$Householdu18))/(max(m$Householdu18)-min(m$Householdu18))
m$Status=(m$Status-min(m$Status))/(max(m$Status)-min(m$Status))
m$Home_Type=(m$Home_Type-min(m$Home_Type))/(max(m$Home_Type)-min(m$Home_Type))
m$Ethnic=(m$Ethnic-min(m$Ethnic))/(max(m$Ethnic)-min(m$Ethnic))
m$Language=(m$Language-min(m$Language))/(max(m$Language)-min(m$Language))
str(m)
```

```
## 'data.frame':    6876 obs. of 14 variables:
##  $ Income      : num  1 1 0 0 0.875 0 0.625 0.125 0.375 0 ...
##  $ Sex         : num  0 1 1 1 0 0 0 0 0 0 ...
##  $ Marital     : num  0 0 1 1 0 1 0.5 0 0 1 ...
##  $ Age         : num  0.667 0.333 0 0 0.833 ...
##  $ Edu         : num  0.8 0.8 0.2 0.2 0.6 0.4 0.6 0.4 0.6 0.6 ...
##  $ Occupation  : num  0.5 0 0.625 0.625 0.875 1 0.25 0.875 0.875 1 ...
##  $ Lived       : num  1 1 1 0.5 1 0.75 1 1 0.75 1 ...
##  $ Dual_Income : num  1 0.5 0 0 1 0 0 1 1 0 ...
##  $ Household   : num  0.5 0.25 0.375 0.375 0.125 0.25 0 0.25 0.125 0 ...
##  $ Householdu18: num  0.222 0.111 0.222 0.222 0 ...
##  $ Status      : num  0 0.5 1 1 0 0.5 0.5 0.5 0.5 0.5 ...
##  $ Home_Type   : num  0 0.5 0 0 0 0.5 0.5 0.5 0.5 0.5 ...
##  $ Ethnic      : num  0.857 0.857 0.857 0.857 0.857 ...
##  $ Language    : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
training=head(m,n=1500)
testing=tail(m,n=5376)
 n <-neuralnet(Income~Age+Sex+Marital+Edu
                +Occupation+Lived+Household
               +Dual_Income+Householdu18+Status
               +Home_Type+Ethnic+Language,data = training,hidden = 10)
plot(n,rep="best")
```




```
# Confusion Matrix & Misclassification Error - training data
output <- compute(n, training[,-1])
p1 <- output$net.result
pred1 <- ifelse(p1<0.12, 1,ifelse((p1<0.23) & (p1>0.12),2,ifelse((p1<0.36) & (p1>0.23),3,ifelse((p1<0.5) & (
p1>0.36),4,ifelse((p1<0.61) & (p1>0.5),5,ifelse((p1<0.73) & (p1>0.61),6,ifelse((p1<0.85) & (p1>0.73),7,ifels
e((p1>0.85),8,9))))))))
tab1 <- table(pred1, training$Income)
tab1
```

```
##
## pred1  0 0.125 0.25 0.375 0.5 0.625 0.75 0.875  1
##     1 63    21   14     9  17    12   14    11  5
##     2 54    16   22    21  13    10    9    14  3
##     3 68    32   28    38  24    28   20    24 14
##     4 56    36   19    36  23    34   37    33 17
##     5 24    10   21    15  13    30   18    30 15
##     6  9     9   13    17  13    24   16    43 16
##     7  6     4    5    13  11    15   23    24 21
##     8  2     1    7     7   6    17   25    53 29
```

```
sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.1706667
```

```
# Confusion Matrix & Misclassification Error - testing data
output <- compute(n, testing[,-1])
p2 <- output$net.result
pred2 <- ifelse(p2<0.12, 1,ifelse((p2<0.23) & (p2>0.12),2,ifelse((p2<0.36) & (p2>0.23),3,ifelse((p2<0.5) & (
p2>0.36),4,ifelse((p2<0.61) & (p2>0.5),5,ifelse((p2<0.73) & (p2>0.61),6,ifelse((p2<0.85) & (p2>0.73),7,ifels
e((p2>0.85),8,9))))))))
tab2 <- table(pred2, testing$Income)
tab2
```

```
##
## pred2   0 0.125 0.25 0.375 0.5 0.625 0.75 0.875   1
##     1 203    68   38    55  48    59   47    50  35
##     2 232    70   60    56  55    68   62    67  31
##     3 269    91   90    84  74    98   94    72  59
##     4 177    92   95   121  95   148   90   130  72
##     5  45    37   38    49  39    86   83   116  70
##     6  30    18   24    52  35   100   95   143 111
##     7  12    18   18    23  38    62   71    90  94
##     8   5     6   13    22  23    55   80   169 151
```

```
sum(diag(tab2))/sum(tab2)
```

```
## [1] 0.1605283
```

```r
#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)
```

```
##
## 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
```

```r
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 0 0 0 0 0
##   5.80972194327446 0 0 0 1 0 0 0
##   6.34970524633368 0 0 0 0 1 1 0
##   6.90008389724613 0 0 0 0 0 0 1
```

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

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

```r
#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
```

```r
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
```

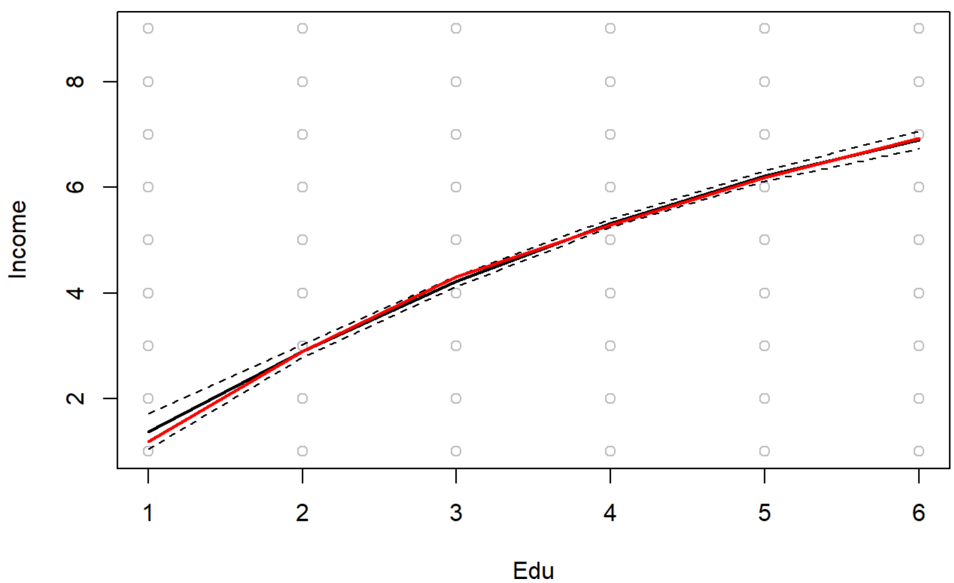
```r
dim(bs(Edu,df=6))
```

```
## [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)
```



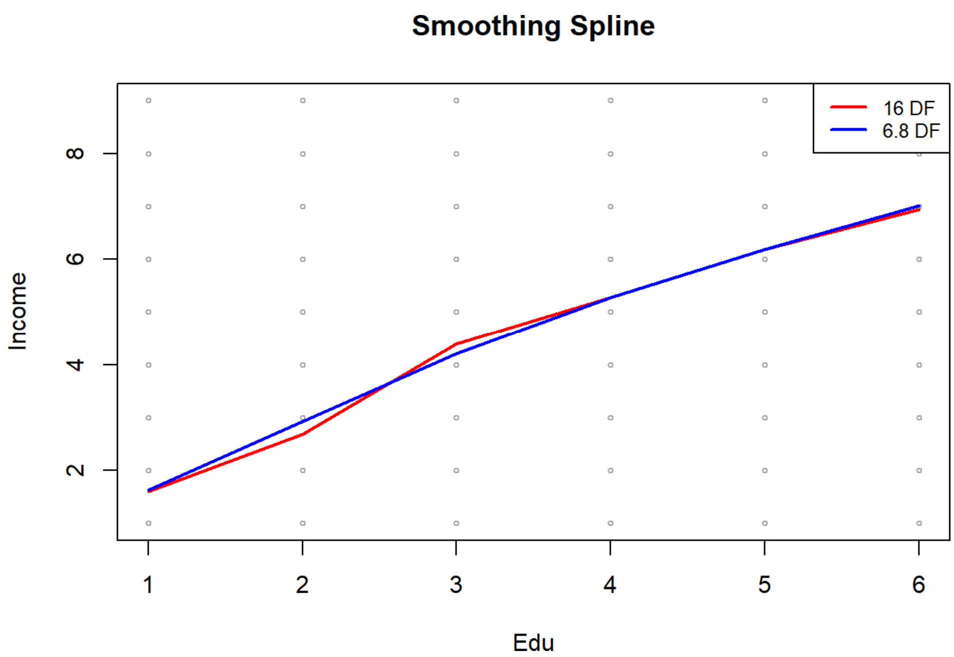

```
plot(Edu,Income,xlim=agelims,cex=.5,col="darkgrey")
title("Smoothing Spline")
fit=smooth.spline(Edu,Income,df=5)
fit2=smooth.spline(Edu,Income,cv=TRUE)
```

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

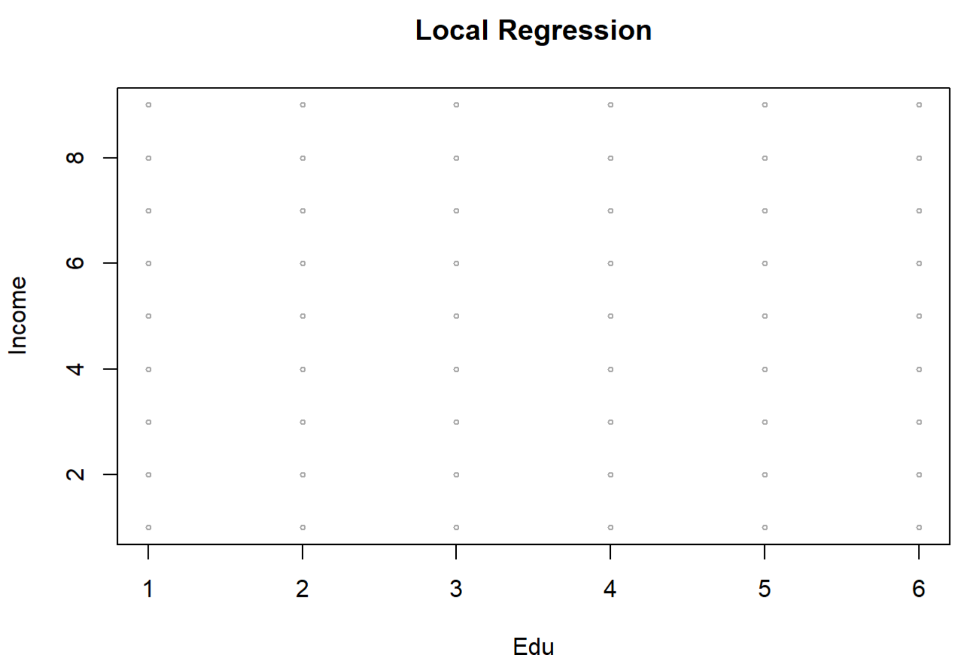
```
fit2$df
```

```
## [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")
```




```
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
```

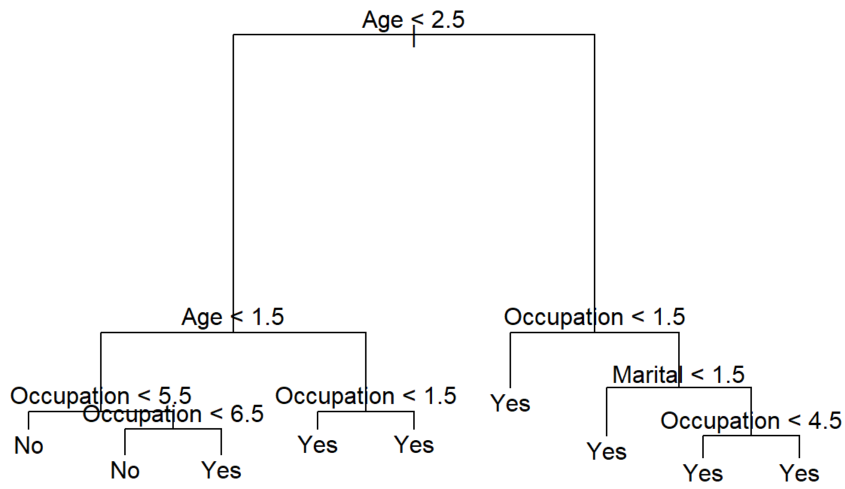
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1
```

```
#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
```

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




```
tree.market
```

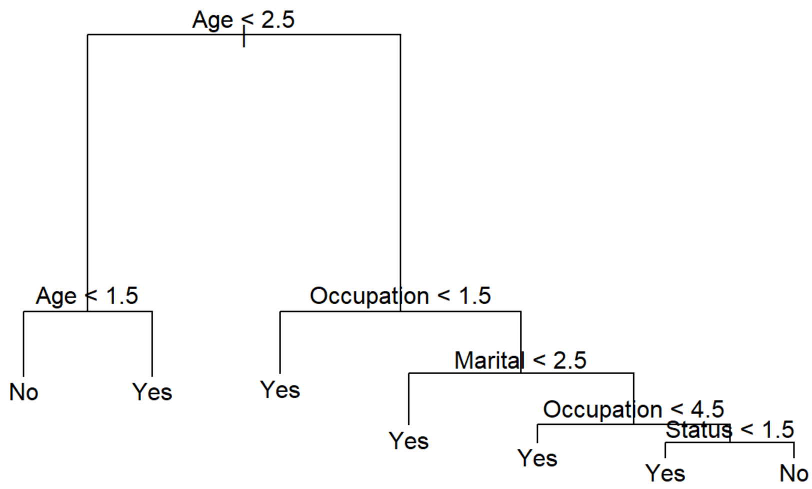
```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 6876 7873.00 Yes ( 0.259453 0.740547 )
##    2) Age < 2.5 2257 3093.00 No ( 0.562694 0.437306 )
##      4) Age < 1.5 647 476.20 No ( 0.879444 0.120556 )
##        8) Occupation < 5.5 110 150.70 No ( 0.563636 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)
```

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

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



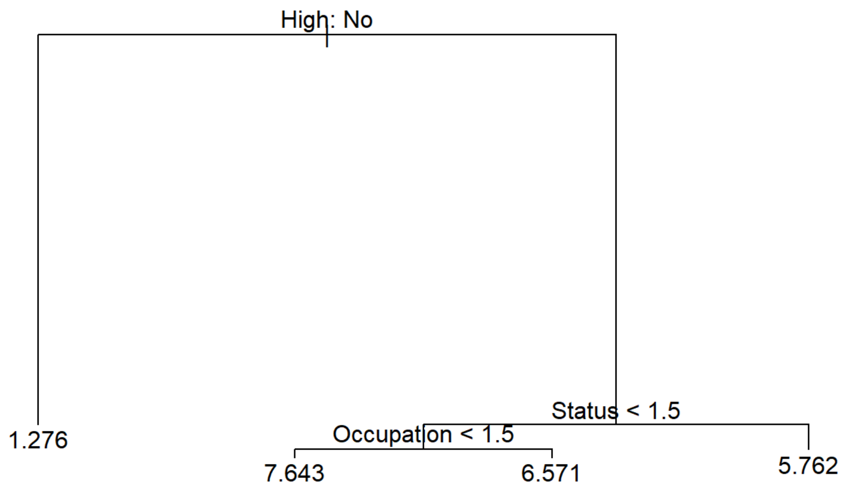

```
tree.pred=predict(prune.market,market.test,type="class")
table(tree.pred,High.test)
```

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

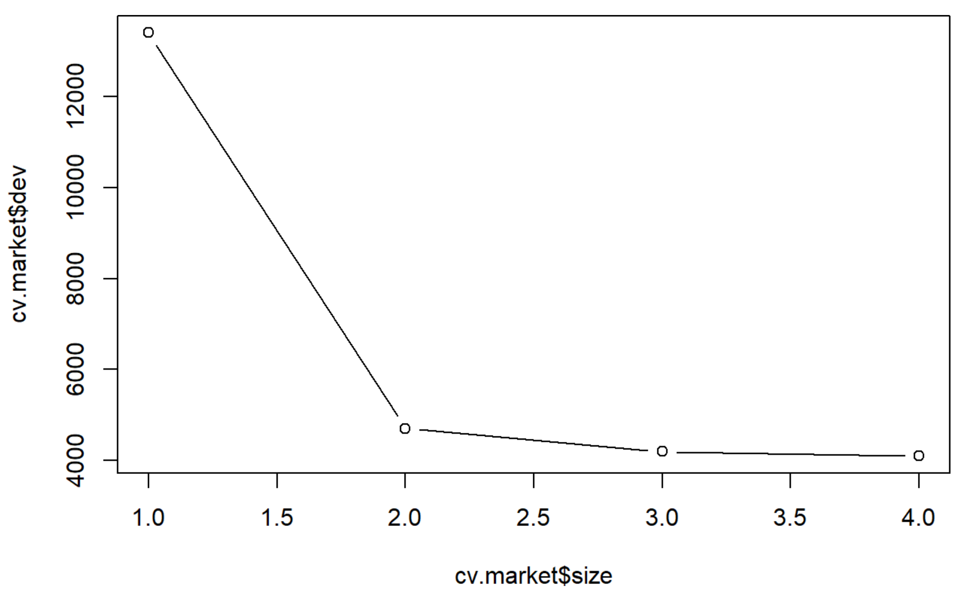
```
#Fitting Regression tree
set.seed(1)
train = sample(1:nrow(market),nrow(market)/4)
tree.market=tree(Income~.,market,subset=train)
summary(tree.market)
```

```
##
## 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
```

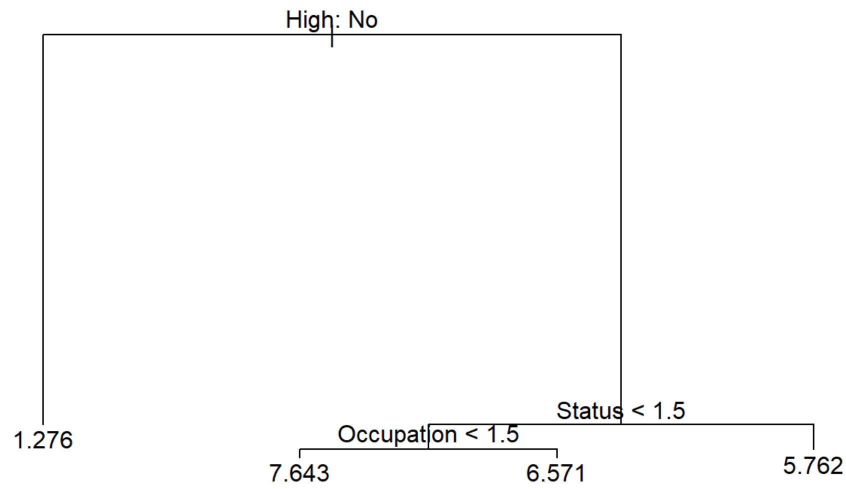
```
plot(tree.market)
text(tree.market,pretty=0)
```



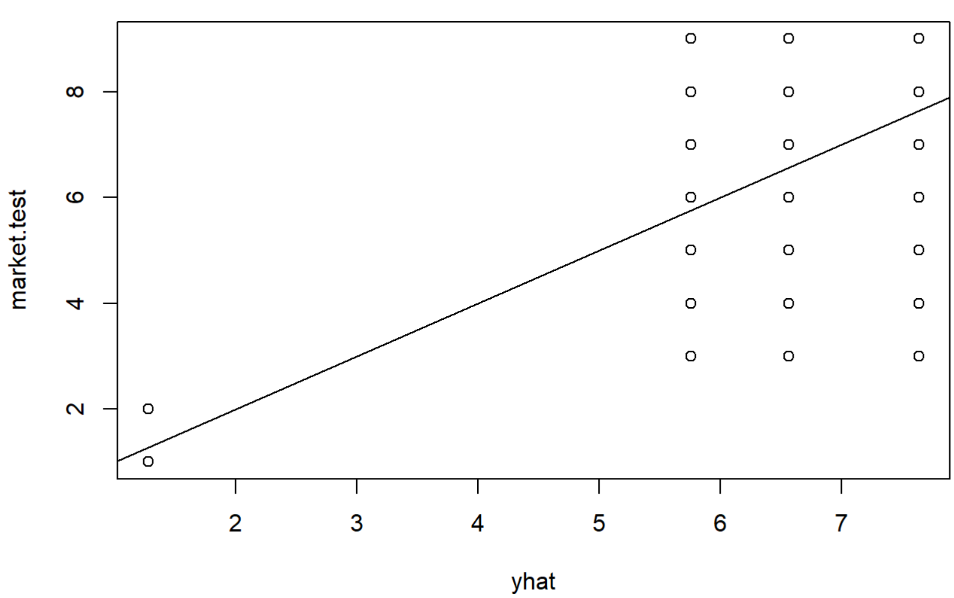

```
cv.market=cv.tree(tree.market)
plot(cv.market$size,cv.market$dev,type='b')
```




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




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




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

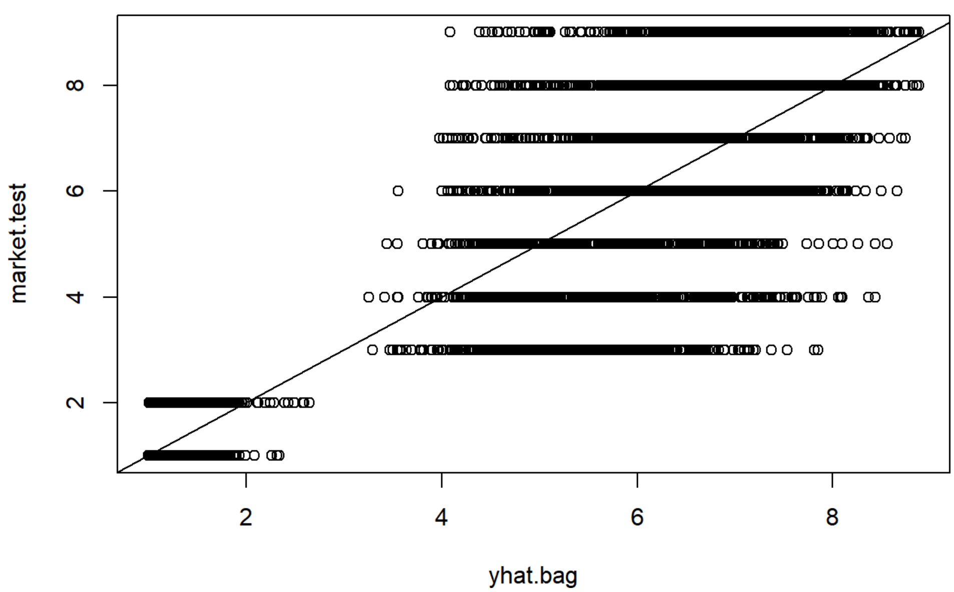
```
## [1] 2.256684
```

```
#Bagging and random forest


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

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




```
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
```

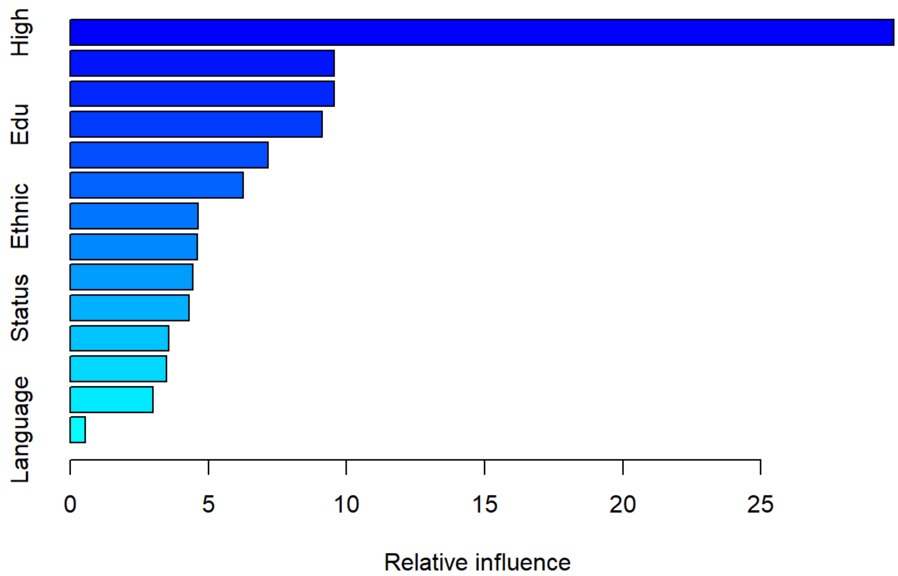
```
importance(rf.market)
```

```
##                %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
```

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

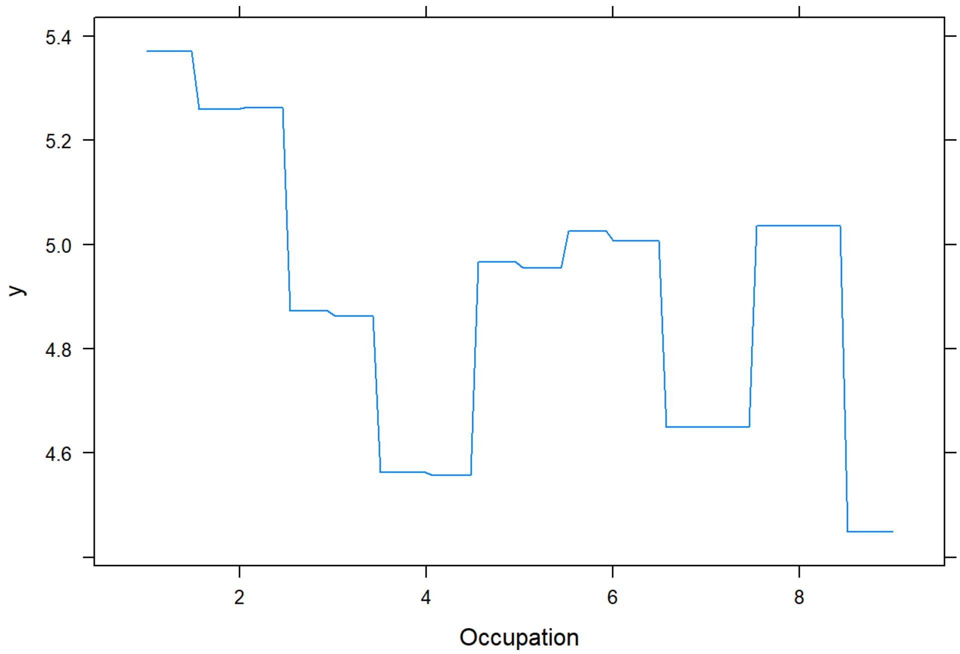

```
#Boosting


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

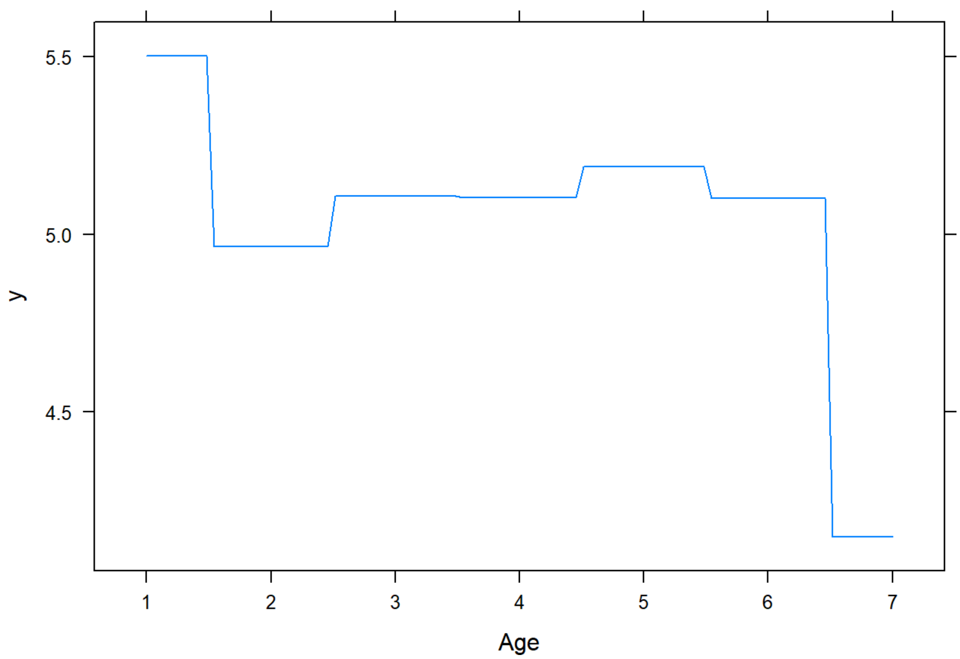



```
##                        var    rel.inf
## High                  High 29.7997106
## Age                    Age  9.5540772
## Occupation      Occupation  9.5524459
## Edu                    Edu  9.1215449
## Household        Household  7.1557530
## Marital            Marital  6.2572653
## Ethnic              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")
```




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




```
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,shrin
kage=0.2,verbose=F)
yhat.boost=predict(boost.market,newdata=market[-train,],n.trees=5000)
mean((yhat.boost-market.test)^2)
```

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