Homework 5

Naga Kartheek Peddisetty, 50538422

12/10/2023

Question 3) Fit a single layer neural network to Weekly data in the ISLR2 package. Predict the direction (Up or Down) of return on a given week. Use cross-validation or the hold out method to determine the number of neurons to use in the layer. Compare your results to those for the logistic regression model. When making the comparison, consider both the classification performance and interpretability of the final model.

```
library(neuralnet)
library(nnet)
library(ISLR2)

data("Weekly")
dim(Weekly)
```

```
## [1] 1089    9
```

```
head(Weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```
table(Weekly$Direction)
```

```
##
## Down   Up
##  484  605
```

```
weekly <- Weekly

weekly$Direction <- as.numeric(weekly$Direction) - 1

head(weekly)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270         0
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576         0
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514         1
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712         1
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178         1
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372         0
```

```
table(weekly$Direction)
```

```
##
##   0   1
## 484 605
```

```
set.seed(123)

indis <- sample(1:nrow(weekly),size = round(0.65 * nrow(weekly)), replace = FALSE)

train_data <- weekly[indis, ]
test_data <- weekly[-indis, ]

X_train <- train_data[, -9]
Y_train <- train_data[, 9]

X_test <- test_data[, -9]
Y_test <- test_data[, 9]

train_err_store <- c()
test_err_store <- c()

for (i in 1:10){
    # fitting neural network with "i" neurons
    nn <- neuralnet(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = train_data, hidden = i, err.fct = "ce", st
epmax = 10^9, linear.output = FALSE)

    pred_train <- predict(nn, newdata = train_data)
    y_hat_train <- ifelse(pred_train > 0.5 , 1 , 0)
    train_err <- sum(Y_train != y_hat_train) / nrow(train_data)
    train_err_store <- c(train_err_store, train_err)

    pred_test <- predict(nn, newdata = test_data)
    y_hat_test <- ifelse(pred_test > 0.5, 1, 0)
    test_err <- sum(Y_test != y_hat_test) / nrow(test_data)
    test_err_store <- c(test_err_store, test_err)

    if (i == 5) {
      plot(nn)
      print(table(Actual = Y_test, Predicted = y_hat_test))
    }
}
```

```
##       Predicted
## Actual   0   1
##      0  66  92
##      1  80 143
```

```
train_err_store
```

```
##  [1] 0.4124294 0.4053672 0.3855932 0.3559322 0.3403955 0.3262712 0.2711864
##  [8] 0.3036723 0.2923729 0.2923729
```

```
test_err_store
```

```
##  [1] 0.4829396 0.4619423 0.4566929 0.4619423 0.4514436 0.4724409 0.4750656
##  [8] 0.4566929 0.4803150 0.4829396
```

```
test_err_store[which.min(test_err_store)]
```

```
## [1] 0.4514436
```

```
paste(which.min(test_err_store),"Neurons optimal single layer neural network model with minimum test error : " , round(test_
err_store[which.min(test_err_store)],3))
```

```
## [1] "5 Neurons optimal single layer neural network model with minimum test error :  0.451"
```

Logistic Regression

```
set.seed(123)

glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = "binomial", data = train_data)

pred_test <- predict(glm.fit , newdata = test_data, type="response")
y_hat_test <- ifelse(pred_test > 0.5, 1, 0)

test_err_glm <- sum(y_hat_test != Y_test) / nrow(test_data)
table(Actual = Y_test,Predicted = y_hat_test)
```

```
##       Predicted
## Actual   0   1
##      0  32 126
##      1  35 188
```

```
test_err_glm
```

```
## [1] 0.4225722
```

Comparision between Neural Network and Logistic Regression

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:neuralnet':
##
##     compute
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
a <- data.frame(Methods = c("single layer neural network model with five neurons", "Logistic Regression"), Test_MSE = c(test
_err_store[which.min(test_err_store)],test_err_glm))

arrange(a,Test_MSE)
```

```
##                                               Methods  Test_MSE
## 1                                 Logistic Regression 0.4225722
## 2 single layer neural network model with five neurons 0.4514436
```

On the Weekly data logistic regression is performing better than single layer neural network when compared with test_mse.

The interpretability of logistic regression is also simple when compared to the neural networks even with single layer has more complex model and difficult to interpret the weights and biases.

Question 4) Take any classification data set and divide it up into a learning set and a test set. Change the value of one observation on one input variable in the learning set so that the value is now a univariate outlier. Fit separate single-hidden layer neural networks to the original learning-set data and to the learning-set data with the outlier. Use cross-validation or the hold out method to determine the number of neurons to use in the layer. Comment on the effect of the outlier on the fit and on its effect on classifying the test set. Shrink the value of that outlier toward its original value and evaluate when the effect of the outlier on the fit vanishes. How far away must the outlier move from its original value that significant changes to the network coefficient estimates occur?

```
library(neuralnet)
library(nnet)



data(infert)
dim(infert)  ### 248 * 8
```

```
## [1] 248   8
```

```
head(infert)
```

```
##   education age parity induced case spontaneous stratum pooled.stratum
## 1    0-5yrs  26      6       1    1           2       1              3
## 2    0-5yrs  42      1       1    1           0       2              1
## 3    0-5yrs  39      6       2    1           0       3              4
## 4    0-5yrs  34      4       2    1           0       4              2
## 5   6-11yrs  35      3       1    1           1       5             32
## 6   6-11yrs  36      4       2    1           1       6             36
```

```
table(infert$case)
```

```
## 
##   0   1 
## 165  83
```

```
inf <- infert

set.seed(123)

indis <- sample(1:nrow(inf), round(2/3*nrow(inf)), replace = FALSE)
train <- inf[indis, ]
test <- inf[-indis, ]

### Adding outlier to age variable in the first row
train_outlier <- train
train_outlier$age[1] <- 100

train_err_store <- c()
test_err_store <- c()
for (i in 1:5){

    # fit neural network with "i" neurons
    nn <- neuralnet(case ~ age + parity + induced + spontaneous, data = train,
    hidden = i, stepmax = 10^9, err.fct = "ce", linear.output = FALSE)


    pred <- predict(nn, newdata = train)
    y_hat_train <- round(pred)
    train_err <- mean(train$case != y_hat_train)
    train_err_store <- c(train_err_store, train_err)

    pred <- predict(nn, newdata = test)
    y_hat_test <- round(pred)
    test_err <- mean(test$case != y_hat_test)
    test_err_store <- c(test_err_store, test_err)
}
train_err_store
```

```
## [1] 0.2606061 0.2181818 0.2484848 0.2121212 0.2000000
```

```
test_err_store
```

```
## [1] 0.2650602 0.2168675 0.2409639 0.2771084 0.3012048
```

Neural network model with the outlier

```
set.seed(123)

train_err_store1 <- c()
test_err_store1 <- c()
head(train_outlier)
```

```
##     education age parity induced case spontaneous stratum pooled.stratum
## 159   12+ yrs 100      2       2    0           0      77             53
## 207   6-11yrs  35      1       0    0           0      42             11
## 179   6-11yrs  29      3       0    0           2      14             29
## 14    6-11yrs  29      3       2    1           0      14             29
## 195   6-11yrs  30      4       1    0           1      30             35
## 170   6-11yrs  35      3       0    0           0       5             32
```

```
for (i in 1:5){

    # fit neural network with "i" neurons
    nn1 <- neuralnet(case ~ age + parity + induced + spontaneous, data = train_outlier,
    hidden = i, stepmax = 10^9, err.fct = "ce", linear.output = FALSE)


    pred1 <- predict(nn1, newdata = train_outlier)
    y_hat_train1 <- round(pred1)
    train_err1 <- mean(train_outlier$case != y_hat_train1)
    train_err_store1 <- c(train_err_store1, train_err1)

    pred1 <- predict(nn1, newdata = test)
    y_hat_test1 <- round(pred1)
    test_err1 <- mean(test$case != y_hat_test1)
    test_err_store1 <- c(test_err_store1, test_err1)
}
train_err_store1
```

```
## [1] 0.2848485 0.2545455 0.2303030 0.2060606 0.1818182
```

```
test_err_store1
```

```
## [1] 0.2409639 0.2168675 0.3132530 0.2771084 0.2650602
```

```
library(knitr)

data <- data.frame(neurons = 1:5,train_error_without_outlier = train_err_store, train_error_with_outlier = train_err_store1,
test_error_without_outlier = test_err_store, test_error_with_outlier = test_err_store1)

kable(data, caption="Model Errors", row.names = FALSE)
```

Model Errors

neurons	train_error_without_outlier	train_error_with_outlier	test_error_without_outlier	test_error_with_outlier
1	0.2606061	0.2848485	0.2650602	0.2409639
2	0.2181818	0.2545455	0.2168675	0.2168675
3	0.2484848	0.2303030	0.2409639	0.3132530
4	0.2121212	0.2060606	0.2771084	0.2771084
5	0.2000000	0.1818182	0.3012048	0.2650602

```
set.seed(123)

# Shrink the outlier towards its original value and retrain the model
shrink_values <- seq(100, 31, length.out = 10)
shrinked_mse <- numeric(length(shrink_values))


for (i in 1:length(shrink_values)) {

  train_outlier$age[1] <- shrink_values[i]

  nn_shrinked <- neuralnet(case ~ age + parity + induced + spontaneous, data = train_outlier, hidden = 2, stepmax = 10^9, er
r.fct = "ce", linear.output = FALSE)

  # Evaluate model performance on the test set
  pred_shrinked <- predict(nn_shrinked, newdata = test)
  y_hat_shrinked <- round(pred_shrinked)
  shrinked_mse[i] <- mean(test$case != y_hat_shrinked)

}

library(knitr)
d <- data.frame(Shrinkage_values = shrink_values, Shrinkage_MSE = shrinked_mse)
kable(d, row.names = FALSE)
```

Shrinkage_values	Shrinkage_MSE
100.00000	0.2650602
92.33333	0.2530120
84.66667	0.2168675
77.00000	0.2891566
69.33333	0.2168675
61.66667	0.3012048

	Shrinkage_values	Shrinkage_MSE
	54.00000	0.2168675
	46.33333	0.2289157
	38.66667	0.2289157
	31.00000	0.2289157

As shown in the above table the shrink value first stabilizes at 84.66667, mse = 0.2168675

Distance between shrink stabilization and original age value: 85 - 31 = 54.

Significant changes to network coefficient estimates occur when outlier moves beyond 54 units from original age value(31).

Question 5) Apply a nonlinear SVM to a binary classification data set of your choice. Make up a table of values of Cost and tuning parameter (e.g., gamma). For each cell in the table compute the misclassification rate using cross-validation or a hold out approach. Find the optimal cost and tuning parameter combination from the table. Compare this rate with the results obtained using LDA, logistic regression and a classification tree.

```
library(ISLR2)
library(e1071)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
data("OJ")
dim(OJ)
```

```
## [1] 1070   18
```

```
head(OJ)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       CH            237       1    1.75    1.99   0.00    0.0         0
## 2       CH            239       1    1.75    1.99   0.00    0.3         0
## 3       CH            245       1    1.86    2.09   0.17    0.0         0
## 4       MM            227       1    1.69    1.69   0.00    0.0         0
## 5       CH            228       7    1.69    1.69   0.00    0.0         0
## 6       CH            230       7    1.69    1.99   0.00    0.0         0
##   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.500000        1.99        1.75      0.24     No  0.000000
## 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
## 3         0 0.680000        2.09        1.69      0.40     No  0.000000
## 4         0 0.400000        1.69        1.69      0.00     No  0.000000
## 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
## 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
##   PctDiscCH ListPriceDiff STORE
## 1  0.000000          0.24     1
## 2  0.000000          0.24     1
## 3  0.091398          0.23     1
## 4  0.000000          0.00     1
## 5  0.000000          0.00     0
## 6  0.000000          0.30     0
```

```
unique(OJ$Purchase)
```

```
## [1] CH MM
## Levels: CH MM
```

```
oj <- OJ
head(oj)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       CH            237       1    1.75    1.99   0.00    0.0         0
## 2       CH            239       1    1.75    1.99   0.00    0.3         0
## 3       CH            245       1    1.86    2.09   0.17    0.0         0
## 4       MM            227       1    1.69    1.69   0.00    0.0         0
## 5       CH            228       7    1.69    1.69   0.00    0.0         0
## 6       CH            230       7    1.69    1.99   0.00    0.0         0
##   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.500000        1.99        1.75      0.24     No  0.000000
## 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
## 3         0 0.680000        2.09        1.69      0.40     No  0.000000
## 4         0 0.400000        1.69        1.69      0.00     No  0.000000
## 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
## 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
##   PctDiscCH ListPriceDiff STORE
## 1  0.000000          0.24     1
## 2  0.000000          0.24     1
## 3  0.091398          0.23     1
## 4  0.000000          0.00     1
## 5  0.000000          0.00     0
## 6  0.000000          0.30     0
```

```
set.seed(123)

indis <- sample(1:nrow(oj),size = round(0.65 * nrow(oj)), replace =  FALSE)

train_data <- oj[indis, ]
test_data <- oj[-indis, ]

X_train <- train_data[, -1]
Y_train <- train_data[, 1]

X_test <- test_data[, -1]
Y_test <- test_data[, 1]

cost_values <- c(0.01,0.1,1,5,10)
gamma_values <- c(0.01,0.1,0.2,0.5,1)

result_table <- matrix(NA, nrow = length(cost_values), ncol = length(gamma_values))

rownames(result_table) <- as.character(cost_values)
colnames(result_table) <- as.character(gamma_values)

for (cost in cost_values) {
  for (gamma in gamma_values) {

    svm_model <- svm(Purchase ~ ., data = train_data, cost = cost, gamma = gamma)

    pred <- predict(svm_model, newdata = test_data)

    misclassification_rate <- mean(Y_test != pred)

    result_table[as.character(cost), as.character(gamma)] <- misclassification_rate
  }
}

min_misclassification_rate <- min(result_table)
optimal_params <- which(result_table == min_misclassification_rate, arr.ind = TRUE)
optimal_cost <- as.numeric(rownames(result_table)[optimal_params[1]])
optimal_gamma <- as.numeric(colnames(result_table)[optimal_params[2]])
```

```
result_table
```

```
##           0.01       0.1       0.2       0.5         1
## 0.01 0.3903743 0.3903743 0.3903743 0.3903743 0.3903743
## 0.1  0.2139037 0.2112299 0.2566845 0.3021390 0.3556150
## 1    0.1684492 0.1925134 0.2058824 0.2192513 0.2459893
## 5    0.1791444 0.2005348 0.2112299 0.2326203 0.2379679
## 10   0.1764706 0.2005348 0.2165775 0.2352941 0.2433155
```

```
cat("Optimal Cost:", optimal_cost)
```

```
## Optimal Cost: 1
```

```
cat("Optimal Gamma:", optimal_gamma)
```

```
## Optimal Gamma: 0.01
```

```
cat("Min Misclassification Rate:", min_misclassification_rate)
```

```
## Min Misclassification Rate: 0.1684492
```

LDA

```
set.seed(123)

lda.fit <- lda(Purchase ~., data = train_data)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda.fit
```

```
## Call:
## lda(Purchase ~ ., data = train_data)
##
## Prior probabilities of groups:
##        CH        MM
## 0.6106322 0.3893678
##
## Group means:
##    WeekofPurchase  StoreID  PriceCH  PriceMM     DiscCH    DiscMM SpecialCH
## CH       256.4706 4.383529 1.873318 2.099741 0.06811765 0.1000471 0.1835294
## MM       252.2841 3.254613 1.865683 2.063985 0.02671587 0.1674539 0.0996310
##    SpecialMM   LoyalCH SalePriceMM SalePriceCH  PriceDiff Store7Yes  PctDiscMM
## CH 0.1176471 0.7202877    1.999694    1.805200 0.19449412 0.4141176 0.04839651
## MM 0.2287823 0.3119065    1.896531    1.838967 0.05756458 0.1992620 0.08013515
##     PctDiscCH ListPriceDiff    STORE
## CH 0.03593773     0.2264235 1.484706
## MM 0.01407481     0.1983026 1.859779
##
## Coefficients of linear discriminants:
##                         LD1
## WeekofPurchase  -0.005596065
## StoreID         -0.028376467
## PriceCH          5.411571785
## PriceMM          5.633107001
## DiscCH          11.039055129
## DiscMM          11.568586519
## SpecialCH        0.292305380
## SpecialMM        0.299973848
## LoyalCH         -3.919138210
## SalePriceMM     -6.678058301
## SalePriceCH     -4.406448590
## PriceDiff       -4.678101073
## Store7Yes       -0.094019783
## PctDiscMM      -45.612963409
## PctDiscCH      -23.423874410
## ListPriceDiff    4.002042780
## STORE           -0.002298879
```

```
test_pred_lda <- predict(lda.fit, newdata = test_data)

table(predicted = test_pred_lda$class , Actual = Y_test)
```

```
##          Actual
## predicted  CH  MM
##        CH 200  42
##        MM  28 104
```

```
test_error_lda <- (1/length(test_pred_lda$class))*length(which(Y_test != test_pred_lda$class))
test_error_lda
```

```
## [1] 0.1871658
```

Logistic regression

```
set.seed(123)

glm.fit <- glm(Purchase ~., data = train_data, family = "binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Purchase ~ ., family = "binomial", data = train_data)
##
## Coefficients: (5 not defined because of singularities)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       5.610558   2.480195   2.262  0.02369 *
## WeekofPurchase   -0.008533   0.013257  -0.644  0.51979
## StoreID          -0.015117   0.168593  -0.090  0.92855
## PriceCH           2.519095   2.165983   1.163  0.24482
## PriceMM          -2.628080   1.079881  -2.434  0.01495 *
## DiscCH           13.822294  22.343772   0.619  0.53617
## DiscMM           37.858041  11.244269   3.367  0.00076 ***
## SpecialCH         0.560970   0.422357   1.328  0.18412
## SpecialMM         0.439598   0.341008   1.289  0.19736
## LoyalCH          -6.222700   0.480793 -12.943  < 2e-16 ***
## SalePriceMM             NA         NA      NA       NA
## SalePriceCH             NA         NA      NA       NA
## PriceDiff               NA         NA      NA       NA
## Store7Yes        -0.447960   0.870105  -0.515  0.60667
## PctDiscMM       -74.383869  23.524986  -3.162  0.00157 **
## PctDiscCH       -32.854890  42.006894  -0.782  0.43414
## ListPriceDiff           NA         NA      NA       NA
## STORE                   NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 930.50  on 695  degrees of freedom
## Residual deviance: 538.85  on 683  degrees of freedom
## AIC: 564.85
##
## Number of Fisher Scoring iterations: 5
```

```
names(glm.fit)
```

```
##  [1] "coefficients"      "residuals"         "fitted.values"    
##  [4] "effects"           "R"                 "rank"             
##  [7] "qr"                "family"            "linear.predictors"
## [10] "deviance"          "aic"               "null.deviance"    
## [13] "iter"              "weights"           "prior.weights"    
## [16] "df.residual"       "df.null"           "y"                
## [19] "converged"         "boundary"          "model"            
## [22] "call"              "formula"           "terms"            
## [25] "data"              "offset"            "control"          
## [28] "method"            "contrasts"         "xlevels"
```

```
glm_test <- predict(glm.fit, newdata = test_data, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
y_hat_test <- ifelse(glm_test > 0.5, "MM", "CH")

table(predicted = y_hat_test,Actual = Y_test)
```

```
##          Actual
## predicted  CH  MM
##        CH 203  42
##        MM  25 104
```

```
test_err <- mean(Y_test != y_hat_test)
test_err
```

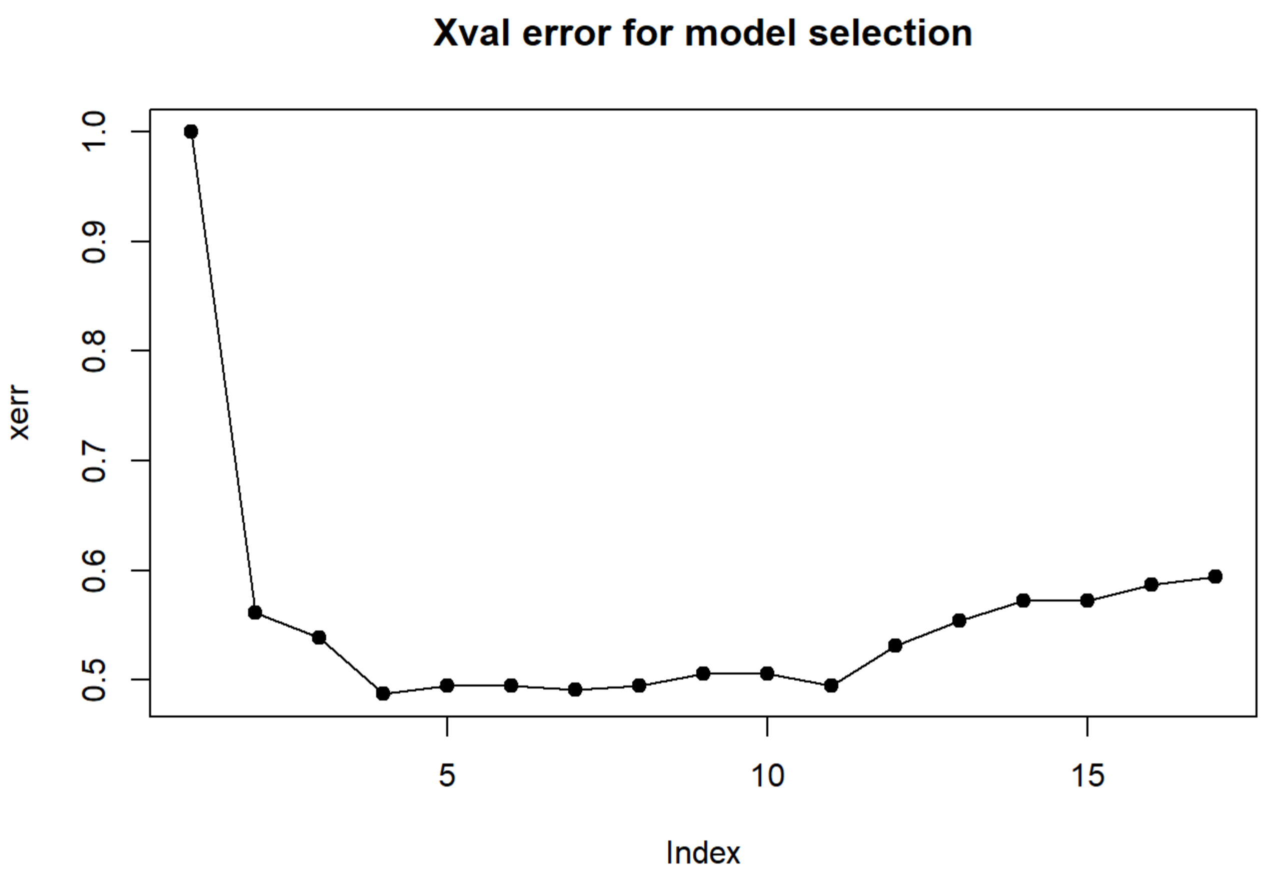
```
## [1] 0.1791444
```

Classification Tree

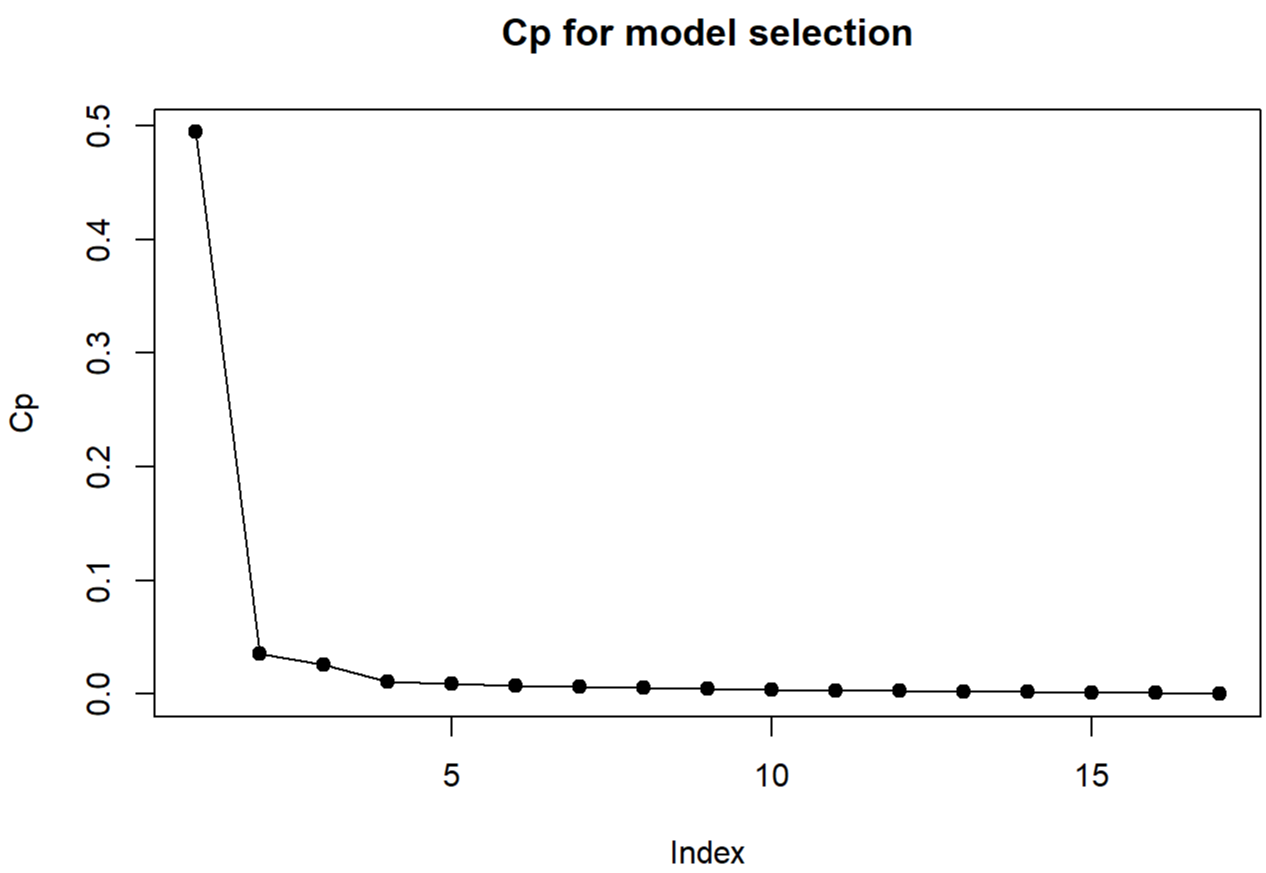
```
library(rpart)
library(rpart.plot)

set.seed(123)

model.control <- rpart.control(minsplit = 3, xval = 10, cp = 0)
fit.X <- rpart(Purchase ~., data = train_data, method = "class", control = model.control)
names(fit.X)
```

```
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "csplit"
## [13] "variable.importance" "y"                   "ordered"
```

```
plot(fit.X$cptable[,4], main = "Xval error for model selection", ylab = "xerr",pch=19,type
='o')
```



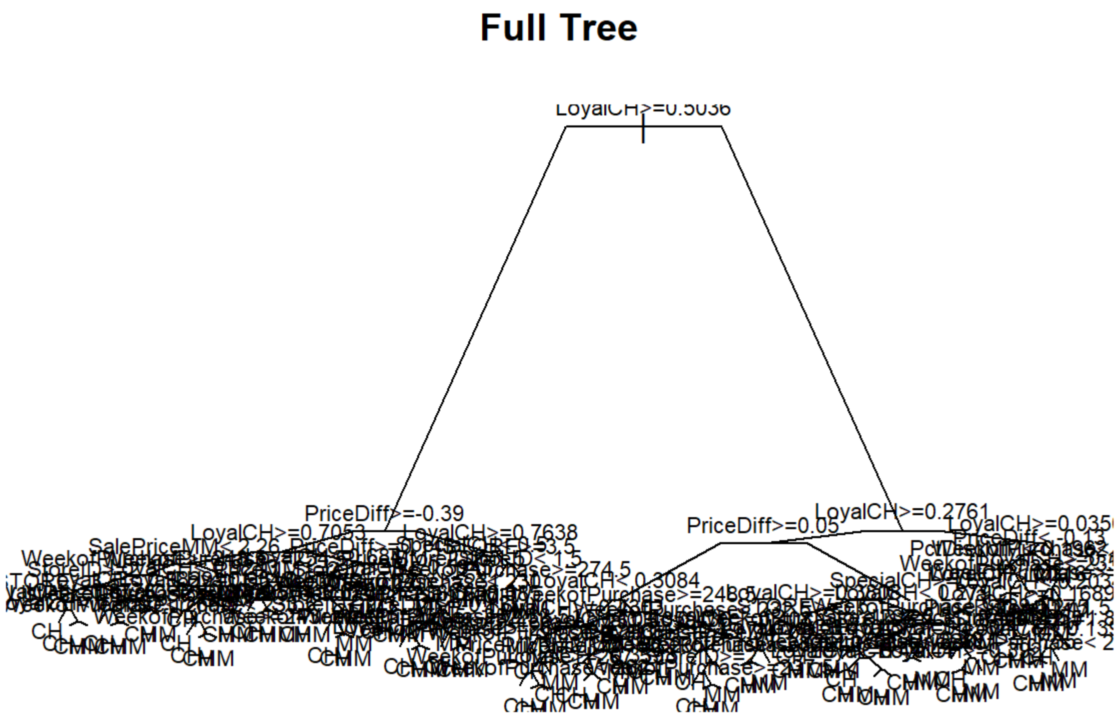

```
plot(fit.X$cptable[,1], main = "Cp for model selection", ylab = "Cp",pch=19,type='o')
```




```
min_cp <- which.min(fit.X$cptable[,4])
min_cp
```

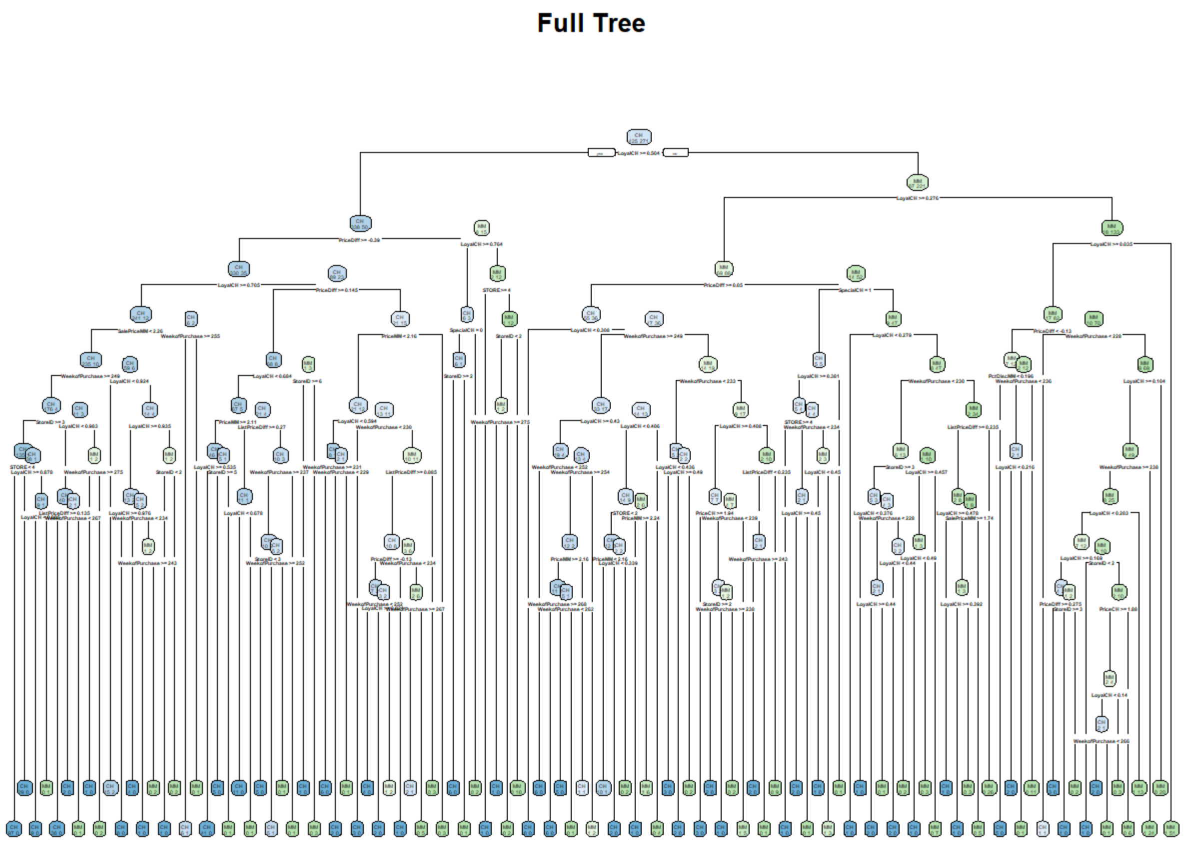
```
## 4
## 4
```

```
pruned_fit_X <- prune(fit.X, cp = fit.X$cptable[min_cp, 1])

plot(fit.X, branch = .3, compress = T, main = "Full Tree")
text(fit.X, cex = .7)
```

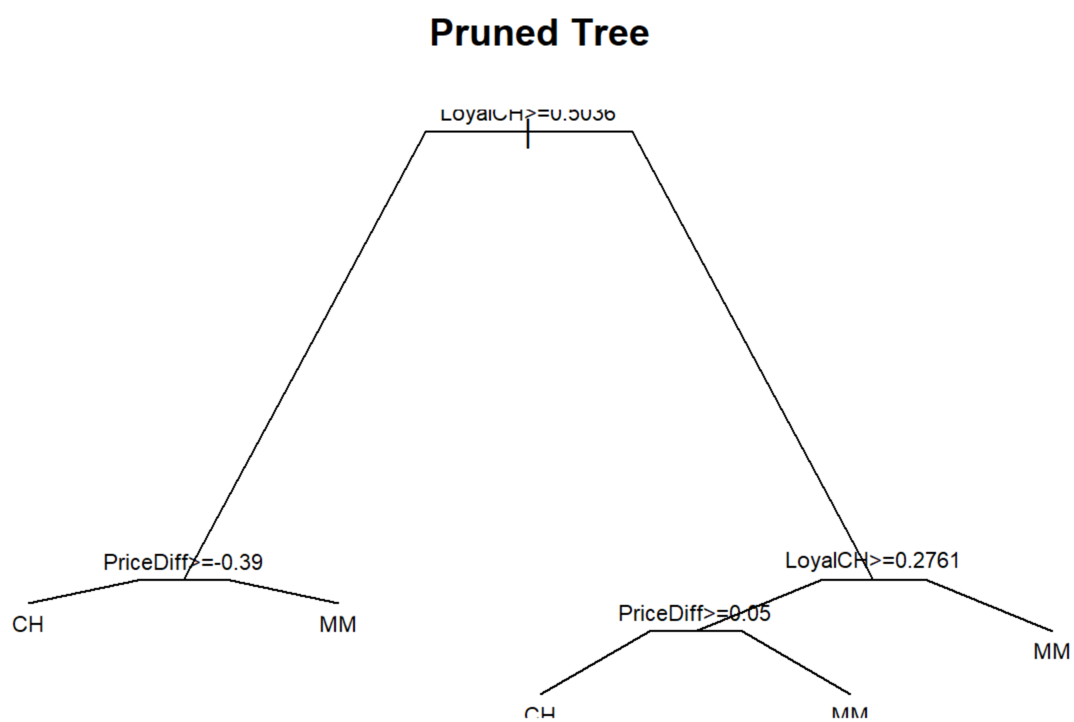



```
rpart.plot(fit.X,digits = 3, extra = 1,main = "Full Tree")
```

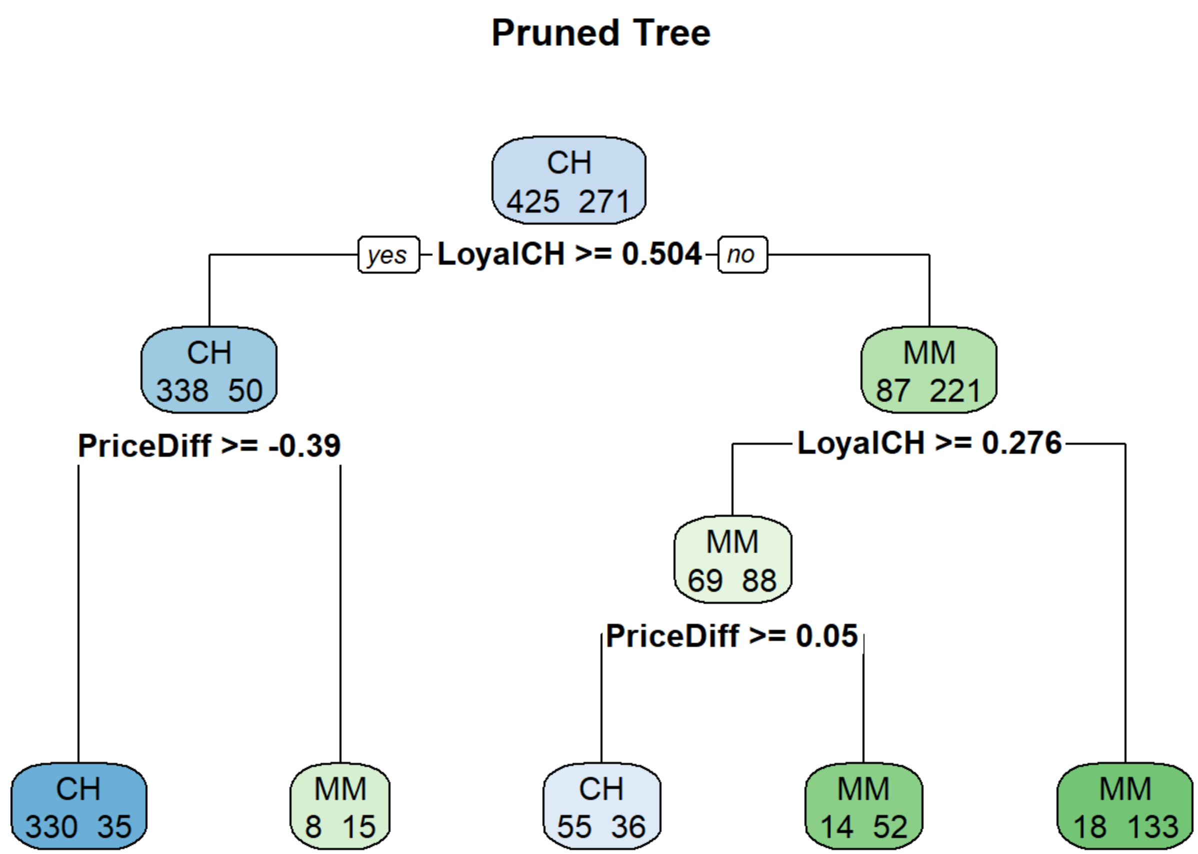
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```




```
plot(pruned_fit_X, branch = .3, compress = T, main = "Pruned Tree")
text(pruned_fit_X, cex = .7)
```




```
rpart.plot(pruned_fit_X,digits = 3, extra = 1,main = "Pruned Tree")
```




```
pred_test <- predict(pruned_fit_X, newdata = test_data, type = 'class')
table(predicted = pred_test,Actual = Y_test)
```

```
##          Actual
## predicted  CH  MM
##        CH 207  50
##        MM  21  96
```

```
err <- sum(pred_test != Y_test) / nrow(test_data)
cat("Misclassification rate for a Classification Tree :", err * 100, "\n")
```

```
## Misclassification rate for a Classification Tree : 18.98396
```

Comparision between the models

```
library(dplyr)

data <- data.frame(Methods = c("Non linear SVM", "LDA", "Logistic Regression", "Classification Tree"), misclassification_rat
e = c(min_misclassification_rate * 100, test_error_lda * 100, test_err * 100, err * 100))

arrange(data,misclassification_rate)
```

```
##               Methods misclassification_rate
## 1      Non linear SVM              16.84492
## 2 Logistic Regression              17.91444
## 3                 LDA              18.71658
## 4 Classification Tree              18.98396
```

On the OJ data set Non-linear SVM is performing well with low test mse = 16.844 when compared with the other models.

Next logistic regression performing better with test mse = 17.914

LDA model is performing with test mse = 18.716

And Classification Tree performance is poor compared to the other models on this OJ data set.