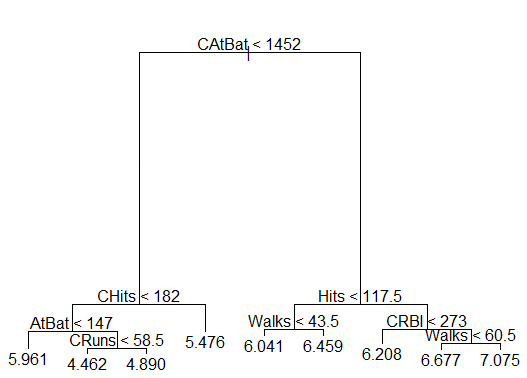
**STAT 4360 (Introduction to Statistical Learning, Spring 2023)**

**Mini Project 6  
 Name: Ann Biju**

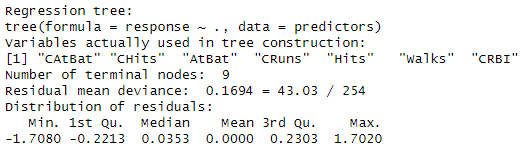
1a.

Unpruned Tree:



The first split occurs to check if CStBat is below 1452 at the root. The tree is split such that there are 9 total nodes and 3 levels not including the root.

Summary:

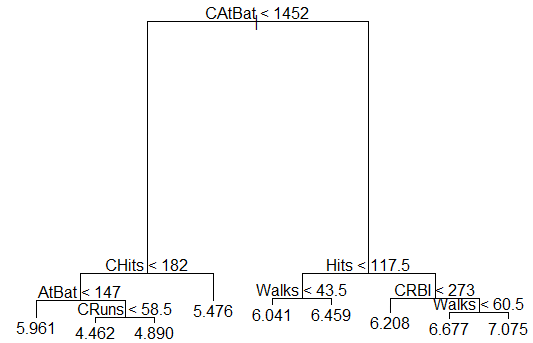


Test mse for unpruned tree:

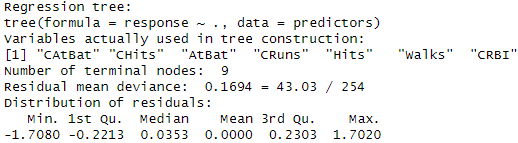


1b.

Pruned Tree:



Summary:



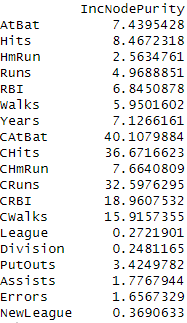


All predictors were used in the pruned tree and the test MSE between the unpruned and pruned trees did not change as they were both 0.164.

1c. Bagging:

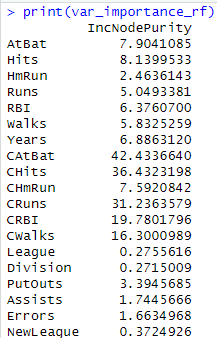


As shown by the importance function below, CAtBat, CHits, CRuns, CRBI, CWalks are the most important predictors, since they have the highest node impurity scores of 40.107, 36.671, 32.597, 18.96, 15.92.



1d. Random Forest

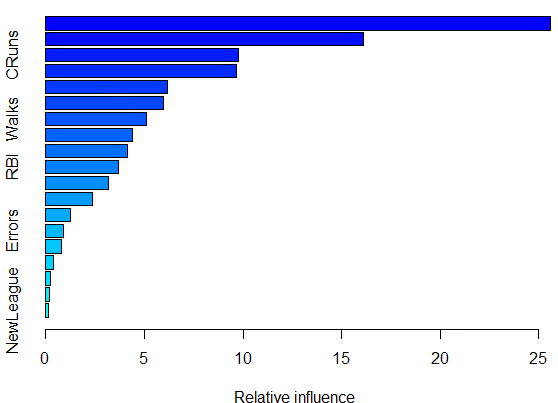
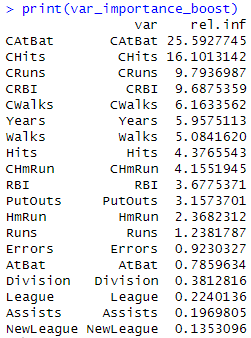




Most important predictors are CAtBat, CHits, CRuns, and CRBI with Node purity scores of 42.4, 36.4, 36.4, and 19.8 respectively.

1e. Boosting

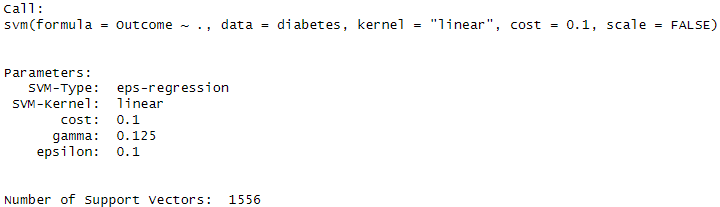




CAtBat, CHits, and CRuns are the most important predictors with the highest relative influences (25.6, 16.1, and CRuns) on the response variable as shown by the chart and table.

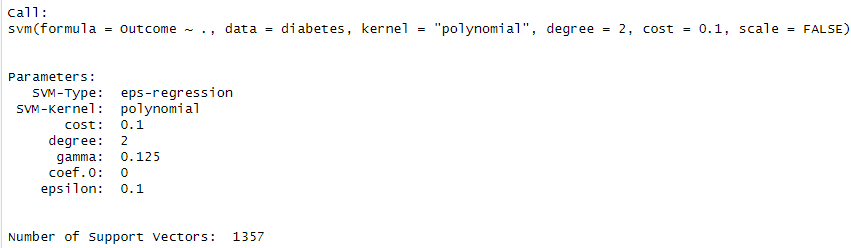
1f. While all methods displayed the same predictors as the most important, Random Forest had the smallest Test MSE at 0.032, so I would recommend this method. In the previous project I had recommended the PLS model.

2a.

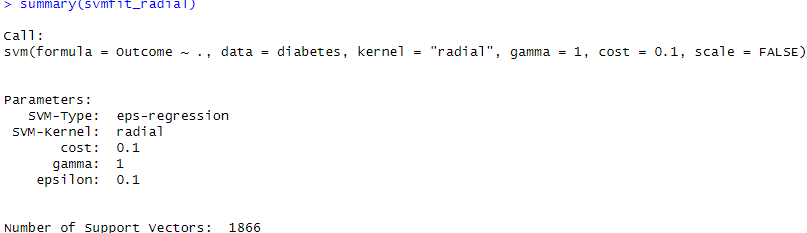


Test MSE: 1

2b.



2c.

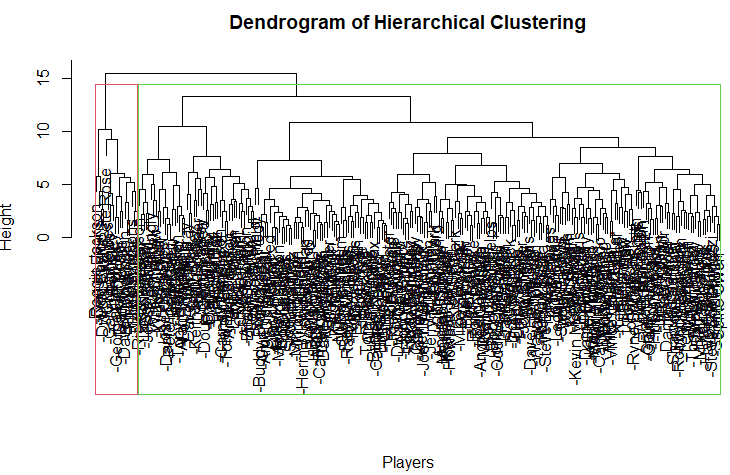


2d. I would choose the method used in 2c to be the best for 10-fold LOOCV because when cost is kept constant, using a radial kernel produced the most support vectors (1866).

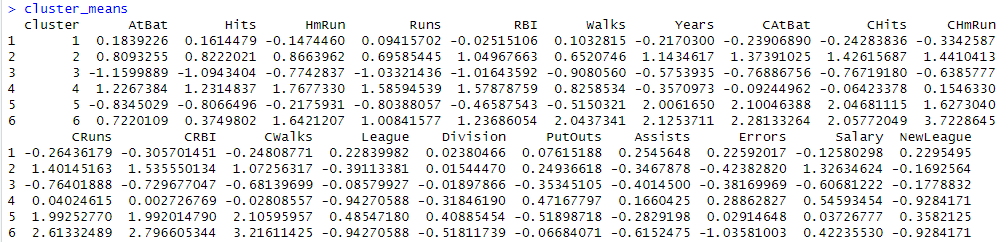
3a. Yes, I think standardizing the variables would be a good idea since this ensures the variables are scaled to eliminate as much bias as possible.

3b. Since I am standardizing the variables, using a metric based approach is best when clustering the players.

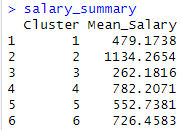
3c.



Cluster specific mean of variables:



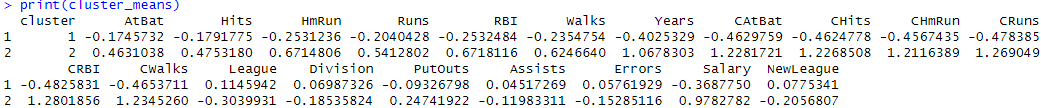
Mean of salaries of players:



Players belonging to cluster 2 seems to have the highest average salary, while cluster 3’s salaries are the lowest.

3d.

Cluster means:



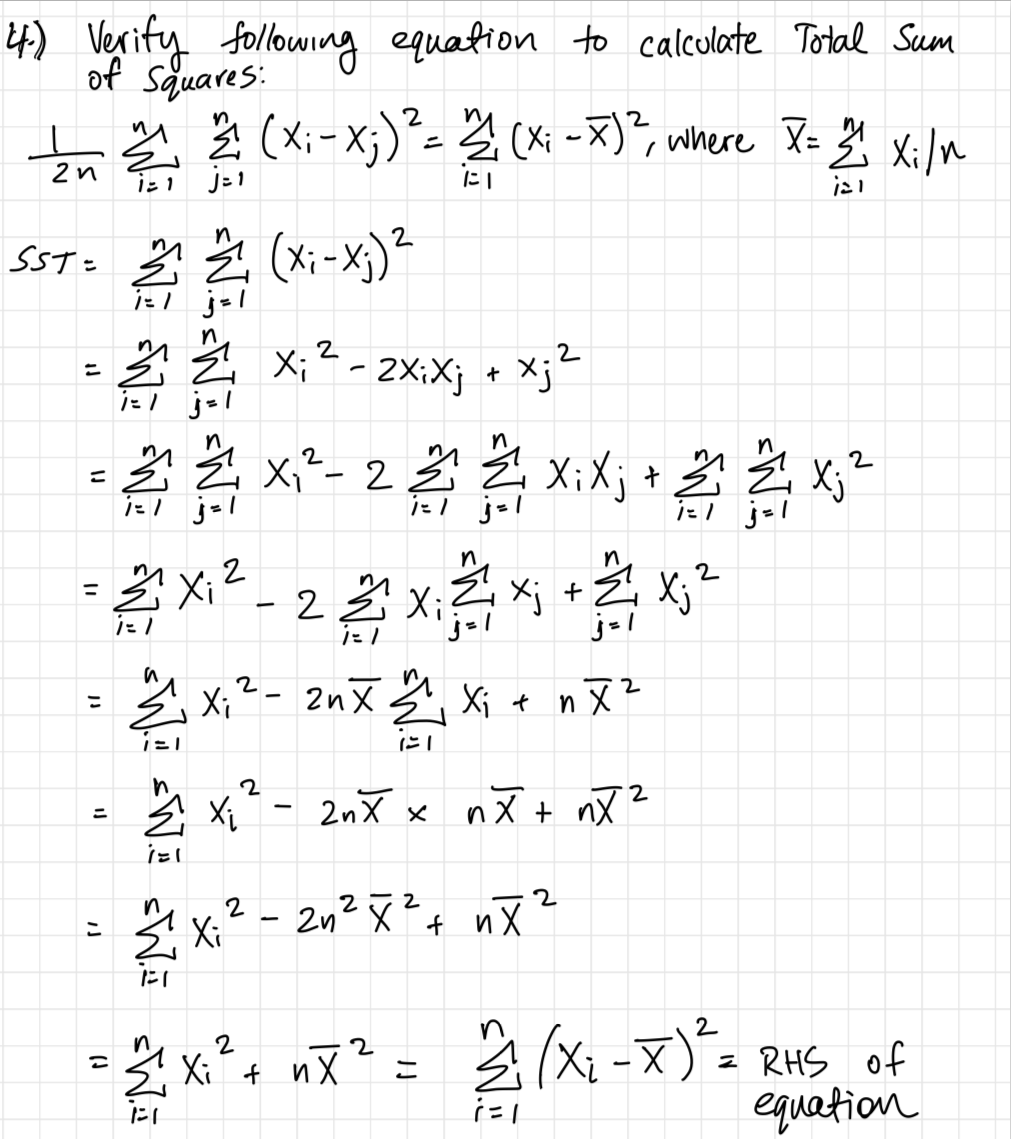
Mean Salaries within clusters:



The range between the mean salary between Cluster 1 and 2 is not as spread out using the K-Means algorithm

3e. The K-means algorithm gives more sensible results, since it is more effective in splitting the data into 2 clusters. Also, the mean salaries and cluster means, using the K-means algorithm seem to be more scaled than those of the dendrogram approach.

4.



**Python Code (or R Code)**

setwd("C:/ann/fall 2023/stat 4360")

# Question 1a

# Install and load required libraries

install.packages("ISLR")

library(ISLR)

install.packages("tree")

install.packages("DAAG")

library(tree)

library(DAAG)

#Hitters dataset

data("Hitters")

# Remove rows with missing data

Hitters <- na.omit(Hitters)

# Extract predictors and response

predictors <- hitters\_data <- Hitters[, -which(names(Hitters) == "Salary")]

response <- log(Hitters$Salary)

# (a) Fit a tree to the data

tree\_model <- tree(response ~ ., data = predictors)

# Display the tree graphically

plot(tree\_model)

text(tree\_model, pretty = 0)

# Summarize the tree

summary(tree\_model)

# Predict and calculate LOOCV

tree\_pred <- predict(tree\_model, newdata = predictors)

tree\_loocv\_mse <- mean((tree\_pred - response)^2)

print(paste("Estimated test MSE for un-pruned tree:", tree\_loocv\_mse))

#Question 1b

# (b) Use LOOCV to determine whether pruning is helpful

cv\_tree <- cv.tree(tree\_model, FUN = prune.tree)

best\_tree\_size <- cv\_tree$size[which.min(cv\_tree$dev)]

# Prune the tree with the optimal size

pruned\_tree\_model <- prune.tree(tree\_model, best = best\_tree\_size)

# Display the pruned tree graphically

plot(pruned\_tree\_model)

text(pruned\_tree\_model, pretty = 0)

# Summarize the pruned tree

summary(pruned\_tree\_model)

# Predict and calculate LOOCV for the pruned tree

pruned\_tree\_pred <- predict(pruned\_tree\_model, newdata = predictors)

pruned\_tree\_loocv\_mse <- mean((pruned\_tree\_pred - response)^2)

print(paste("Estimated test MSE for pruned tree:", pruned\_tree\_loocv\_mse))

# Compare the best pruned and un-pruned trees

print(paste("Optimal size for pruned tree:", best\_tree\_size))

# Question 1c

install.packages("randomForest")

library(randomForest)

bagging\_model <- randomForest(response ~ ., data = predictors, ntree = 1000)

# Predict and calculate LOOCV for bagging

bagging\_pred <- predict(bagging\_model, newdata = predictors)

bagging\_loocv\_mse <- mean((bagging\_pred - response)^2)

print(paste("Estimated test MSE for bagging (B=1000):", bagging\_loocv\_mse))

var\_importance\_bagging <- importance(bagging\_model)

print(var\_importance\_bagging)

# Question 1d

num\_predictors <- ncol(predictors)

mtry\_value <- round(num\_predictors / 3)

rf\_model <- randomForest(response ~ ., data = predictors, ntree = 1000, mtry = mtry\_value)

# Predict and calculate LOOCV for random forest

rf\_pred <- predict(rf\_model, newdata = predictors)

rf\_loocv\_mse <- mean((rf\_pred - response)^2)

print(paste("Estimated test MSE for random forest (B=1000, m=p/3):", rf\_loocv\_mse))

# Display variable importance for random forest

var\_importance\_rf <- importance(rf\_model)

print(var\_importance\_rf)

# Question 1e

install.packages("gbm")

library(gbm)

boost\_model <- gbm(response ~ ., data = predictors, distribution = "gaussian", n.trees = 1000, interaction.depth = 1, shrinkage = 0.01)

# Predict and calculate LOOCV for boosting

boost\_pred <- predict(boost\_model, newdata = predictors, n.trees = 1000)

boost\_loocv\_mse <- mean((boost\_pred - response)^2)

print(paste("Estimated test MSE for boosting (B=1000, d=1, λ=0.01):", boost\_loocv\_mse))

# Display variable importance for boosting

var\_importance\_boost <- summary(boost\_model)

print(var\_importance\_boost)

# Question 2a

#load diabetes data

diabetes <- read.csv("C:\\ann\\fall 2023\\stat 4360\\project 6\\diabetes(1).csv")

diabetes

install.packages("e1071")

install.packages("caret")

library(e1071)

library(caret)

# Set the seed for reproducibility

set.seed(123)

# Extract predictors and response

predictors <- diabetes[, -9] # Assuming the last column is the Outcome variable

response <- diabetes[, 9]

svmfit <- svm(Outcome ~ ., data = diabetes, kernel = "linear", cost = 0.1, scale = FALSE)

svmfit$index

summary(svmfit)

# Get the optimal cost parameter

optimal\_cost <- svm\_tune$best.parameters$cost

# Fit the support vector classifier with the optimal cost

svm\_model <- svm(Outcome ~ ., data = diabetes, kernel = "linear", cost = 0.1, scale = FALSE)

# Summarize key features of the fit

summary(svm\_model)

# Compute estimated test error rate using 10-fold cross-validation

cv\_results <- train(Outcome ~ ., data = diabetes, method = "svmLinear", trControl = trainControl(method = "cv", number = 10))

cv\_error\_rate <- 1 - cv\_results$Accuracy

# Question 2b

svmfit\_poly <- svm(Outcome ~ ., data = diabetes, kernel = "polynomial", degree = 2, cost = 0.1, scale = FALSE)

summary(svmfit\_poly)

# Compute estimated test error rate using 10-fold cross-validation

svmfit\_poly <- train(Outcome ~ ., data = diabetes, method = "svmPoly", trControl = trainControl(method = "cv", number = 10))

svmfit\_poly <- 1 - cv\_results\_poly$results$Accuracy

print(paste("Estimated test error rate for polynomial SVM:", cv\_error\_rate\_poly))

# Question 2c

svmfit\_radial <- svm(Outcome ~ ., data = diabetes, kernel = "radial", gamma = 1, cost = 0.1, scale = FALSE)

summary(svmfit\_radial)

# Question 3c

install.packages("cluster")

library(cluster)

# Standardize the variables

Hitters$League <- as.numeric(as.factor(Hitters$League))

Hitters$Division <- as.numeric(as.factor(Hitters$Division))

Hitters$NewLeague <- as.numeric(as.factor(Hitters$NewLeague))

standardized\_predictors <- scale(Hitters)

# Hierarchical clustering with complete linkage and Euclidean distance

hclust\_model <- hclust(dist(standardized\_predictors), method = "complete")

# Cut the dendrogram to form two clusters

cut\_height <- 10

clusters <- cutree(hclust\_model, h = cut\_height)

# Summarize the cluster-specific means of the variables

cluster\_means <- aggregate(standardized\_predictors, by = list(cluster = clusters), FUN = mean)

print(cluster\_means)

# Summarize the mean salaries of the players in the two clusters

salary\_summary <- aggregate(Hitters$Salary, by = list(cluster = clusters), FUN = mean)

colnames(salary\_summary) <- c("Cluster", "Mean\_Salary")

print(salary\_summary)

# Plot the dendrogram

plot(hclust\_model, main = "Dendrogram of Hierarchical Clustering", xlab = "Players", sub = "")

rect.hclust(hclust\_model, k = 2, border = 2:3) # Highlight the two clusters with different colors

# Question 3d

# K-means clustering with K = 2

kmeans\_model <- kmeans(standardized\_predictors, centers = 2, nstart = 20)

# Assign cluster labels to the original data

clusters <- kmeans\_model$cluster

# Summarize the cluster-specific means of the variables

cluster\_means <- aggregate(standardized\_predictors, by = list(cluster = clusters), FUN = mean)

print(cluster\_means)

# Summarize the mean salaries of the players in the two clusters

salary\_summary <- aggregate(Hitters$Salary, by = list(cluster = clusters), FUN = mean)

colnames(salary\_summary) <- c("Cluster", "Mean\_Salary")

print(salary\_summary)