Predicting Median Value of Boston Housing

Lorenzo Gordon and Kasia Krueger

MA 5790

Spring 2022

Abstract

In 1970, the population of the city of Boston, Massachusetts was 641,071, and the surrounding metropolitan area's population was 3,708,710 (Boston, 2004). Data on census and housing was recorded in 1970 to collect neighborhood information such as average number of rooms per dwelling, pupil-teacher ratio, and per capita crime rate, and is now being analyzed to predict home values. The main goal of the study is to predict median home value of Boston, using 1970 census data. The relationship between predictors will be explored, and both linear and nonlinear models will be fit to the training data using cross validation training methods. The top models will be used to predict on the test set and the overall best model will then be selected.

Table of Contents

Lorenzo Gordon and Kasia Krueger
MA 5790

1. Background

The goal of this project was to find a data set that accomplished the following: a reasonably large data set with at least 200 samples and 10 predictors, and could be used to apply the concepts learned in this class to build a predictive model.

The Boston Housing data set appealed to our project as the variables do not focus on aspects of an individual home (e.g. square footage, lot size, etc.) but instead used neighborhood statistics such as accessibility to highways, distances to employments centers, pupil-teacher ratio, and so on, to predict the median home value of a Boston-area home. This data set shows that environmental and economic factors can have as much impact on home prices as the homes themselves.

The original intent of this dataset was to study Boston's air pollution's effect on home prices. The researchers believed that Bostonians would "pay more" for cleaner air (Cantaro, 2021). The data and variables collected during this study was necessary to isolate the independent influence of air pollution, hence the wide-ranging neighborhood variables. This dataset eventually became used to model and predict median home values in the Boston, Massachusetts area, given the dataset's useful and comprehensive neighborhood information.

The data are found in the *mlbench* package in R (R Core Team, 2020). All analyses have been conducted using RStudio Version 1.3.1093, and the analysis in this report is conducted using *caret*, *corrplot*, *pls*, *e1071*, and *ggplot* packages.

2. Variable Introduction and Definitions

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Massachusetts. The dependent variable is the median value of owner-occupied homes in USD 1000. There are 12 continuous variables, 1 categorical variable, and 506 observations. Below is a list of the variable, names as used in this analysis, with their descriptions.

Variable Name	Description
crim	per capita crime rate by town
Zn	proportion of residential land zoned for lots over 25,000 sq.ft
Indus	proportion of non-retail business acres per town
NOx	nitric oxides concentration (parts per 10 million)
Rm	average number of rooms per dwelling
Age	proportion of owner-occupied units built prior to 1940
Dis	weighted distances to five Boston employment centers
Rad	index of accessibility to radial highways
Tax	full-value property-tax rate per USD 10,000
Ptratio	pupil-teacher ratio by town
B	\(1000(B - 0.63)^2\) where \(B\) is the proportion of minority population by town

Lstat	percentage of lower status of the population
Chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
medv	Outcome - median value of owner-occupied homes in USD 1000

The relationship between predictors and median home value will be analyzed. The data will be preprocessed, including any sort of transformations or elimination of predictors needed. Following this, both linear and nonlinear classification and continuous models will be applied attempting to predict both outcomes as stated above.

3. Data Exploration

The scatterplots in **Figure 1** show there is an increasing relationship between median value and number of room, and a decreasing relationship between low status of population and median value of homes.

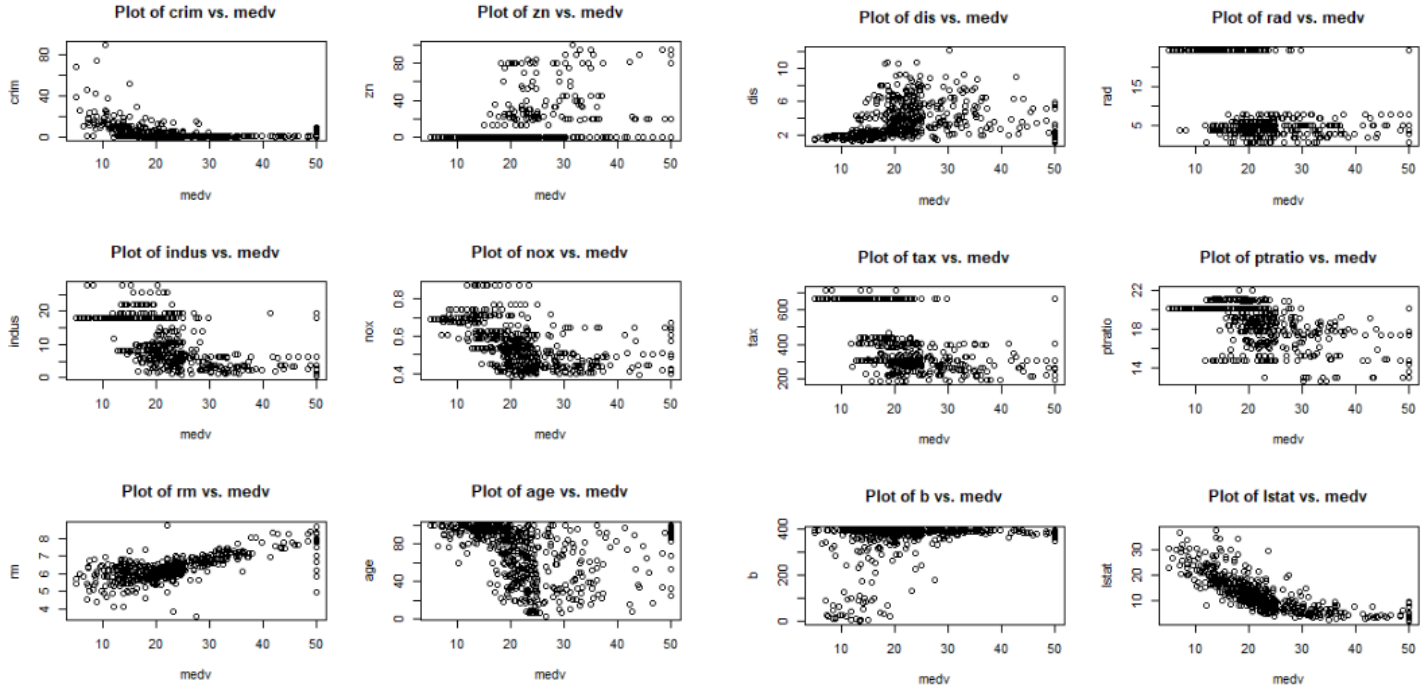


Figure 1: Scatterplots of Raw Predictors vs. Median Value

Figure 2 shows the histograms of continuous predictors. Heavily right and left skewed predictors can be seen in crime, zone, distance, and proportion of minority population (B). These predictors will be tested for skewness and transformed.

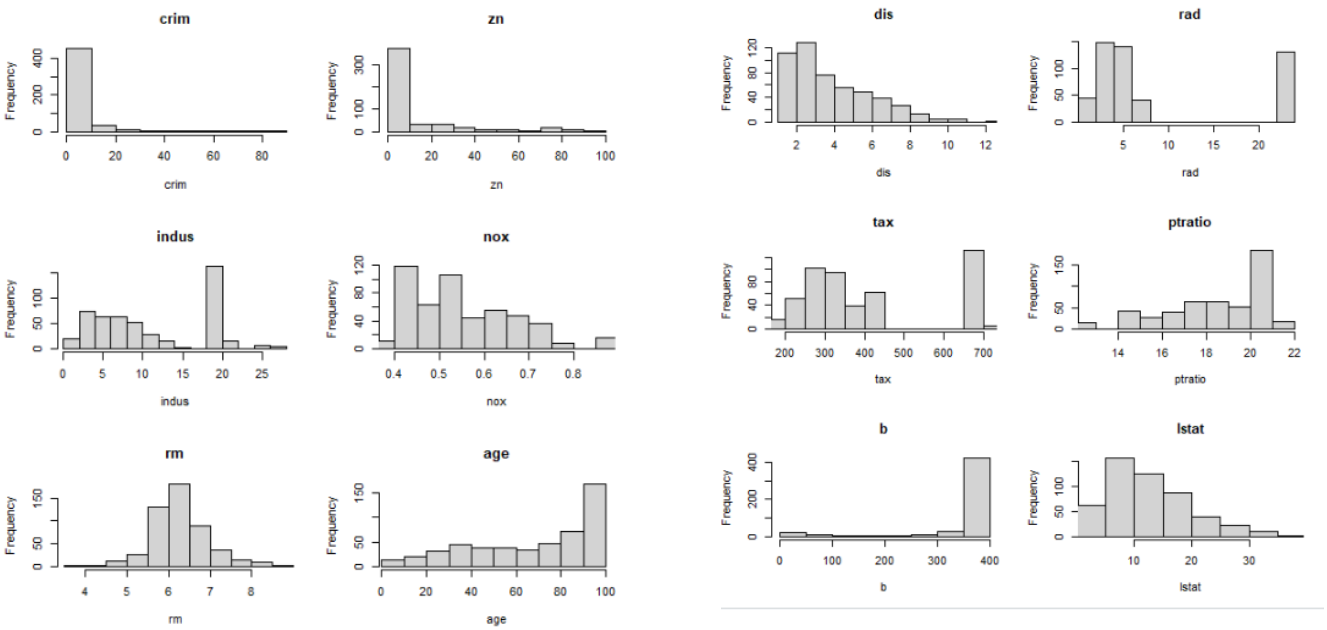


Figure 2: Histograms of Continuous Predictors

4. Data Pre-Processing

The next step in analyzing this data is to preprocess the data. There is no missing data in the outcome and in the predictors, so there is no need for imputation. There are no near zero variance predictors, so no predictors are removed.

a. Correlations

A correlation plot of the 11 continuous predictors was created to explore the relationship between predictors. **Figure 3** shows the correlation plot with the blue representing positive correlations between predictors and red representing negative correlations between predictors. Due to the small set of predictors, a high cut-off threshold is chosen for correlation to preserve predictors in the model. Using a cutoff threshold of .90, other are no correlated predictors over 0.9. There are no correlations higher than |0.85| between the predictors and the outcome.

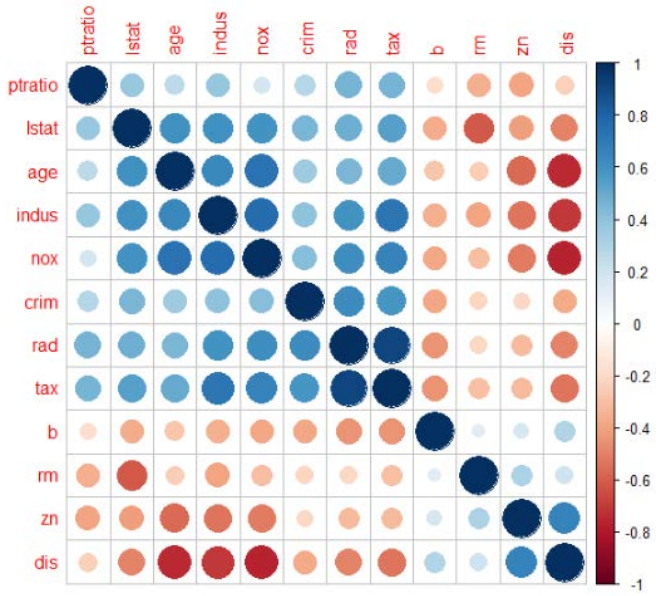


Figure 3: Correlation Plot of Continuous Predictors

b. Transformations

i. Skewness

To combat skewness, the predictors are centered and scaled, and a log transformation is performed to reduce skewness in the predictors with skewness value greater than 0.50. **Table 2** displays the skewness values before and after the transformation. Before transformation, 3 predictors were heavily skewed (shown in yellow) and 6 predictors were moderately skewed (shown in blue). After transformation, only two predictors are heavily skewed and only 1 predictor is moderately skewed.

Figure 4 shows an improvements to the skewness of the continuous predictors after the log transformation. The predictors appear more symmetrical and balanced.

Skewness values Before Transformation	**crim**	**Zn**	**Indus**	**Nox**	**Rm**
	5.1922223	2.2124881	0.2932747	0.7249897	0.4012223
	Age	**Dis**	**Rad**	**Tax**	**ptratio**
	-0.5954162	1.0057898	0.9988651	0.6659891	-0.7975743
	B	**Lstat**			
	-2.8732597	0.9010929			
Skewness values After Transformation	**crim**	**Zn**	**Indus**	**Nox**	**Rm**
	0.4035309	2.2124881	0.2932747	0.3556612	0.4012223
	Age	**Dis**	**Rad**	**Tax**	**ptratio**
	-0.5954162	0.1518258	0.28492	0.3285994	-0.797574
	B	Lstat			
	-2.8732597	-0.3183362			

Table 2: Predictor Skewness Values Before and After Transformation

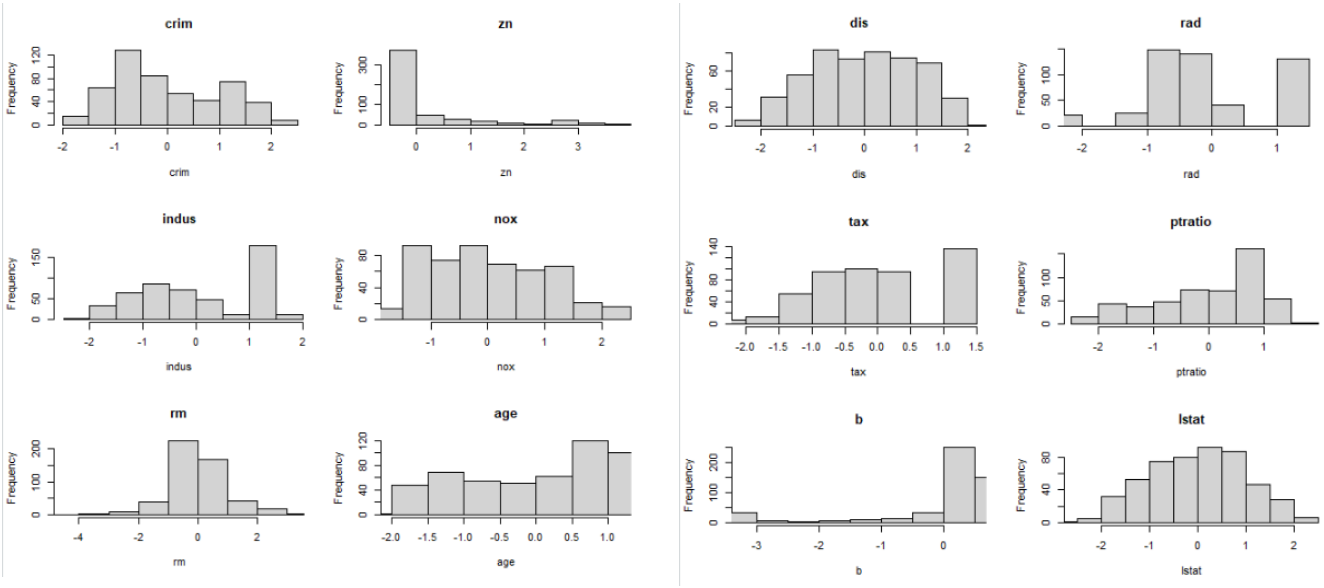


Figure 4: Histograms of Continuous Predictors for Skewness After Transformation

ii. Outliers

Figure 5 shows the boxplots of continuous predictors. An examination of the data finds that most predictors are balanced and that all outliers are valid.

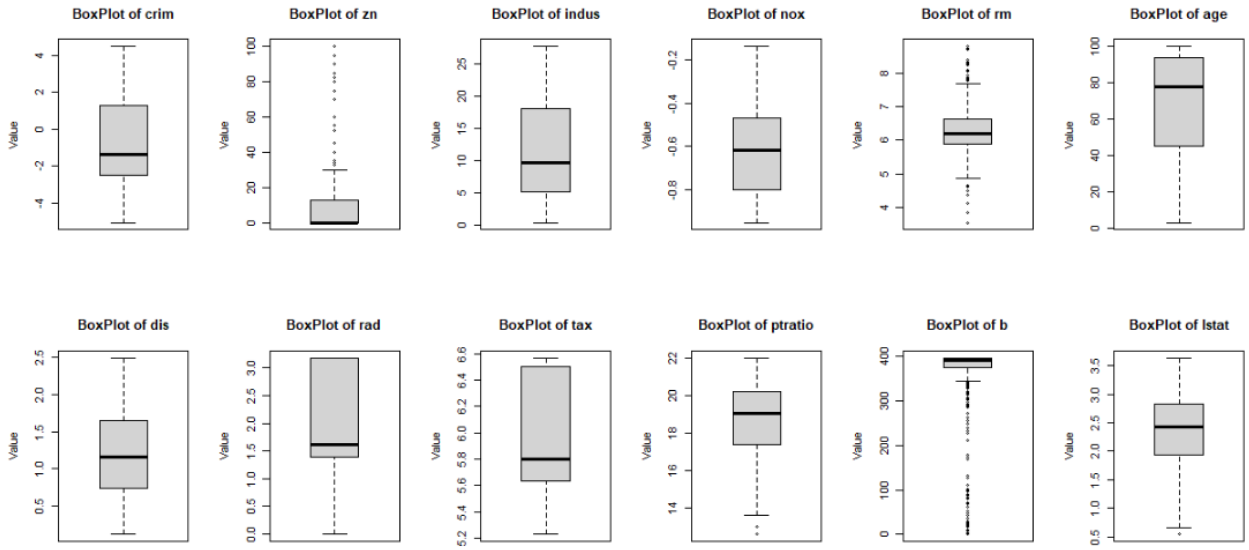


Figure 5: Boxplots of Continuous Predictors After Transformation

5. Splitting the Data

The data set is large so we can select 75% for training and 25% for testing. 380 sample are used for the training set and the remaining 126 samples are used for the testing/validation set. We use stratified random sampling in order to split the data and use training control to resample the data. The data is resampled using 3-fold cross-validation, repeated 5 times. Since some model building requires X to be numeric, the categorical variable, the Charles River dummy variable (chas), is removed and 12 continuous predictors are used.

For classification models, the median home values are binned into two equal classes: low and high. There are 256 "low" observations and 250 "high" observations. The skewness of the median home value is very small, -0.328, showing that the values are evenly distributed and balanced as shown in the histogram in **Figure 6.**

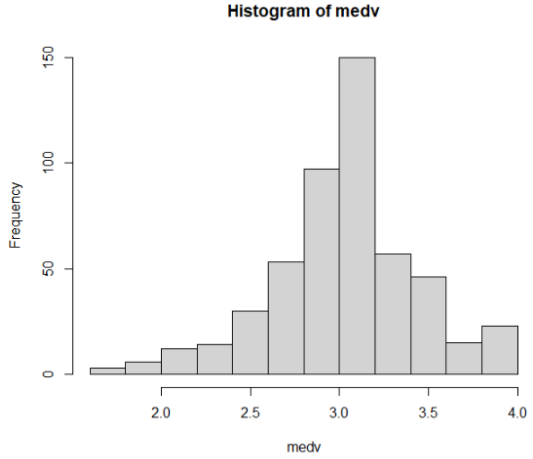


Figure 6: Histogram of Median Home Value

6. Model building

a. Continuous Outcome of Median Home Value

Using the training data, first set of linear models tuned were to predict the median home value, which is a continuous variable. Both linear and non-linear models were trained to the data and supplemental figures for tuning parameters and are shown in the first appendix. The models were tuned using 3-fold cross-validation repeated 5 times. **Table 3** shows the results for these models when predicting on the training set. The non-linear regression models performed better overall, with the SVM and MARS models creating the smallest RMSE and largest R-squared values for the training set.

Linear Models

	RMSE	**R-Squared**
Ordinary Linear Regression	0.1988	0.7696
Generalized Linear Regression with PCA	0.2153	0.7216
Partial Least Squares	0.2104	0.7337

Non-Linear Models		
K-Nearest Neighbors	0.1909	0.7826
SVM	0.1652	0.8355
Elastic Net	0.2030	0.7521
MARS	0.1795	0.8099

Table 3: Summary of Continuous Models (train)

b. Categorical Outcome for Median Home Value with Two Levels

The next set of models trained were created to predict the median home value which have been categorized into one of two equal classes, high or low. We use ROC and AUC as the optimization parameter. Supplemental figures are available in the second appendix for each model.

Table 4 shows the results of the models. Both linear and nonlinear classification models performed well. The models performed similarly to the continuous outcome models with penalized GLM and support vector machine model performing with the highest ROC on the training sets. SVM model also has one of the highest accuracies in the training set.

Classification Linear Model					
	Specificity	**Sensitivity**	**ROC**	**Accuracy**	**Kappa**
Logistic	0.8475	0.8572	0.9357	0.851	0.7019
Linear Discriminant Analysis	0.8207	0.8716	0.9382	0.844	0.6881
Partial Least Squares Discriminant Analysis	0.8294	0.9152	0.9422	0.8431	0.6862
Nearest Shrunken Centroids	0.6364	0.8159	0.9282	0.707	0.4143
Penalized GLM	0.8530	0.8625	0.9445	0.8100	0.6188
Classification Nonlinear Models					
Mixture Discriminant Analysis	0.8373	0.8750	0.9453	0.8365	0.6726
Neural Network	0.8643	0.8850	0.9511	0.8594	0.7185
Flexible Discriminant Analysis	0.8218	0.8627	0.9304	0.8631	0.7257
SVM	0.8687	0.8908	0.9552	0.8614	0.7226

Table 4: Summary of Classification Linear Models (train)

c. Choosing best model

i. Continuous Outcome Models

The top two continuous outcome models are SVM and MARS. Predicting on the test set found that the support vector machine model has the highest R-squared value and lowest RMSE. **Figure 7** shows the percentage of lower status of the population variable (lstat) was found to be the most important predictor variable in the SVM model, followed by the proportion of non-retail business acres per town (indus), and the number of rooms in the home (rm).

Top Continuous Outcome Models		
	RMSE	**R-Squared**
SVM	0.15094803	0.87194241
MARS	0.1647556	0.8459229

Table 5: Summary of Best Continuous Outcome Models (test)

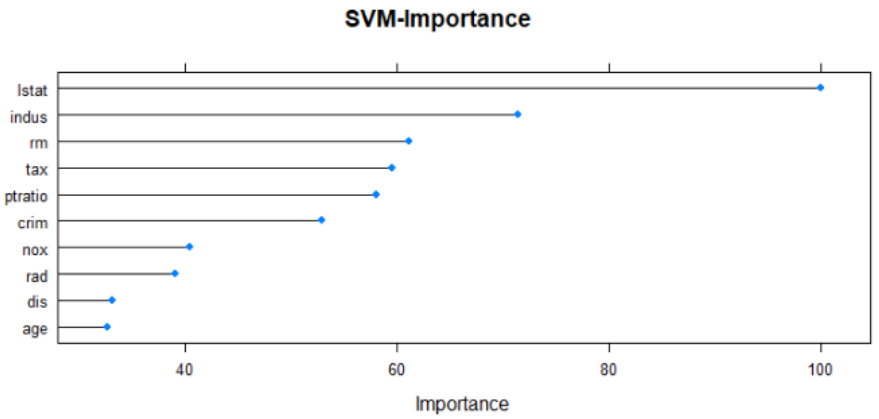


Figure 7: Variable Importance Plot for SVM Model

ii. **Categorical Outcome Models**

The top two categorical outcome models are SVM and Penalized GLM. Predicting on the test set found that the support vector machine model has the highest area under the curve. The confusion matrix shows the true negatives and true positives are properly predicted and the accuracy rate is 0.9234.

Top Categorical Outcome Models	**Accuracy**	**Kappa**	**AUC**
Penalized GLM	0.8839	0.7677	0.8915
SVM	0.9234	0.84684	0.9343

Table 6: Summary of Best Continuous Outcome Models (test)

Support Vector Machine (SVM) Model		
Prediction	Reference	
	Low	high
low	172	24
High	20	163

Table 7: SVM Confusion Matrix

7. Summary

For continuous models, we conclude that the best model found during the analysis was the support vector machine (SVM) model. This model performed the best on the test set, and the resulting R-squared rate was high at 0.8719. The advantage of using a continuous model it is easy to understand how the parameters affect the outcome, median home value. The analysis found that two environmental and locational variables were most important in determining median home value, followed by the number of rooms in the home. The percentage of lower status of the population variable and the proportion of non-retail business acres per town were considered more important in the SVM model than the attributes of a home itself.

Interestingly, for categorical models, we conclude that the best model found during this analysis was also the support vector machine model used to predict the two-level outcome of low or high median value of a home. The resulting accuracy rate on the test set was 0.9234.

Lorenzo Gordon and Kasia Krueger
MA 5790

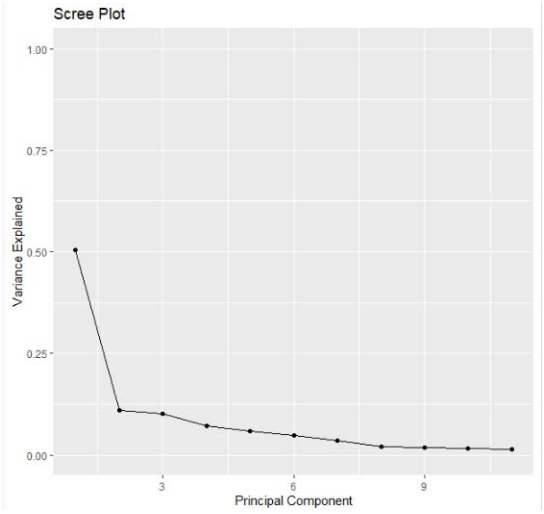
Appendix 1: Supplemental Material for Continuous Outcome Models

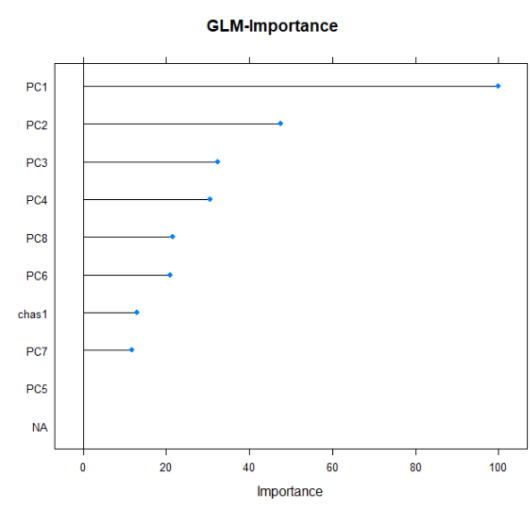
a. GLM with PCA

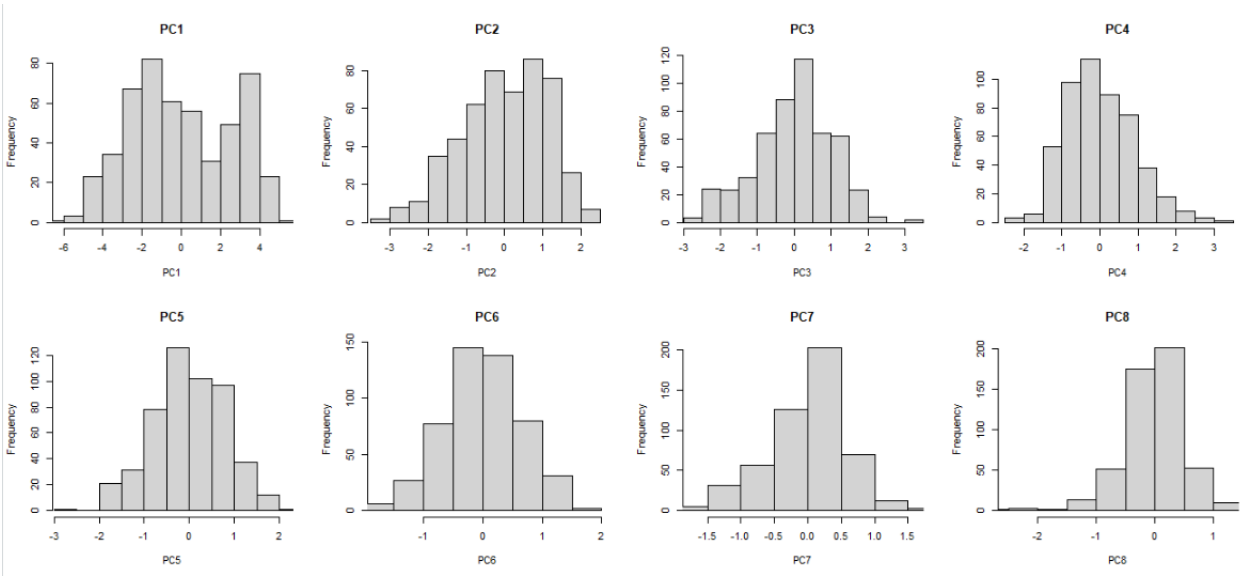
We performed a principal component analysis and found 8 PCs that explain 95% of the variance. Using 3-fold cross validation repeated 5 times for the resampling method, a generalized linear model is run using PCA data.

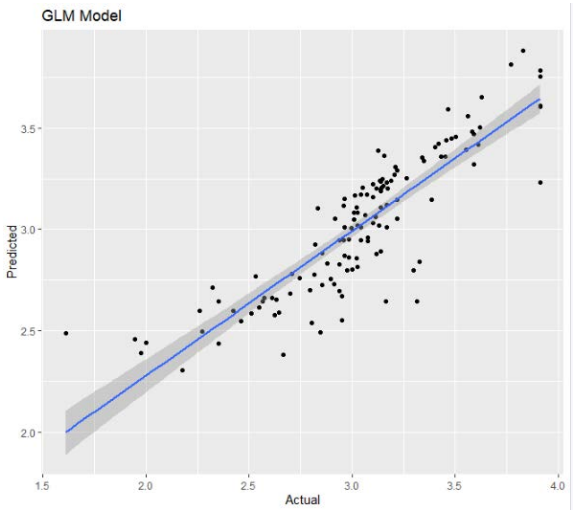
PCs needed to explain variability in data

% variance	0.588	0.106	0.094	0.071	0 .055	0.036	0.029	0.020



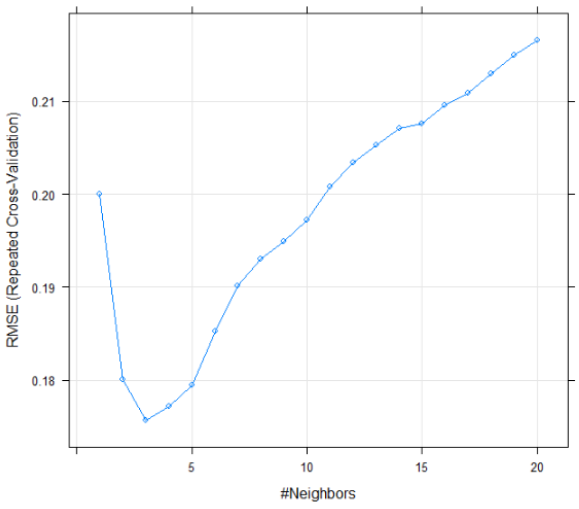


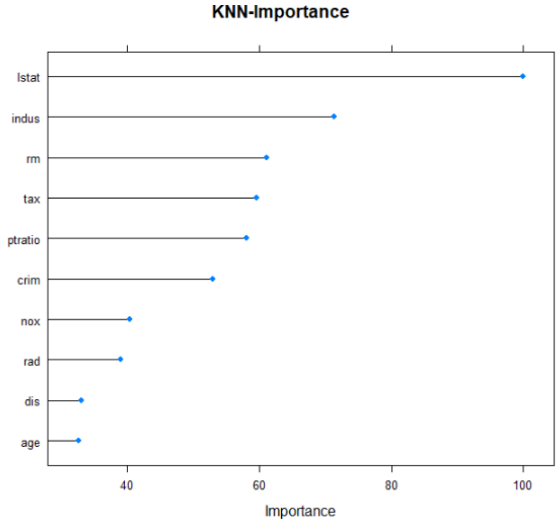


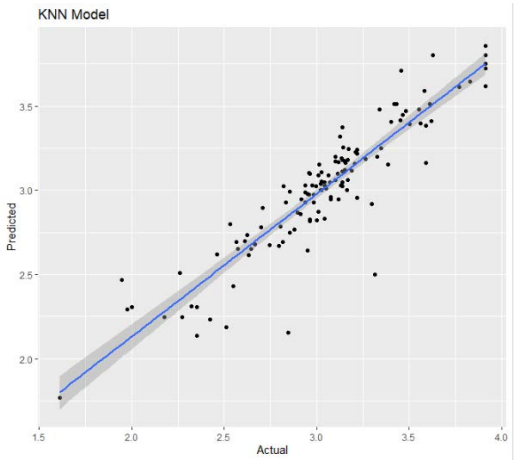


b. K-Nearest Neighbors

The optimal value of KNN was 3-nearest neighbors.

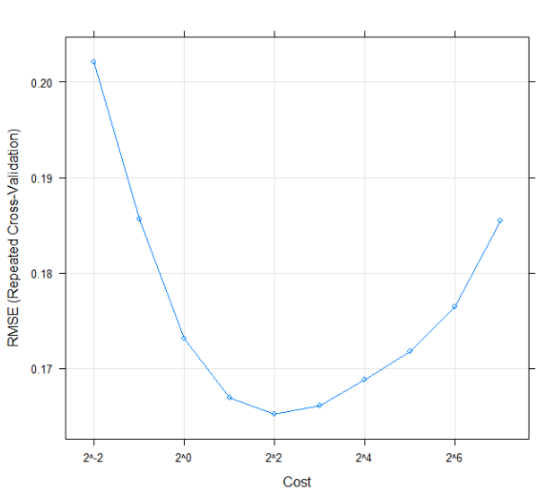


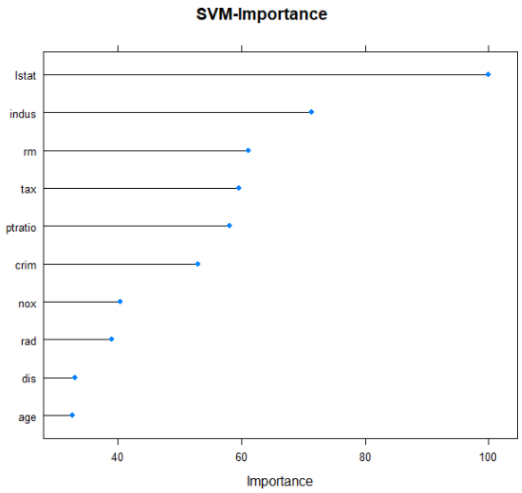


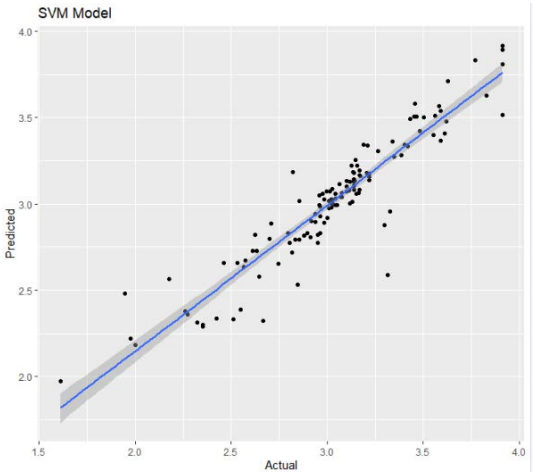


c. Support Vector Machine

The optimum model chosen used a value of sigma = 0.073135107007627, epsilon = 0.1, and cost C = 4.

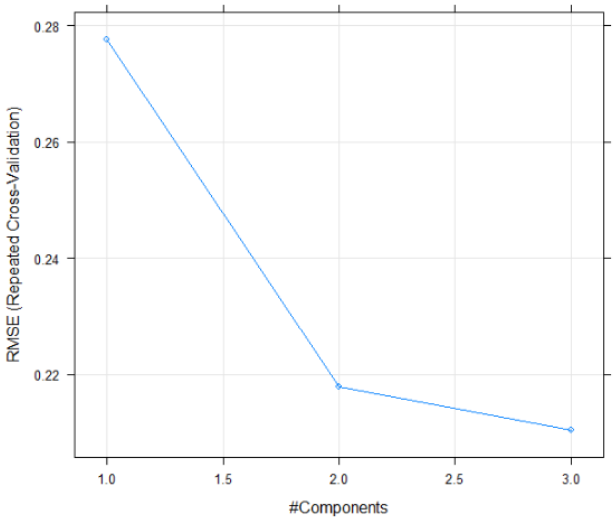


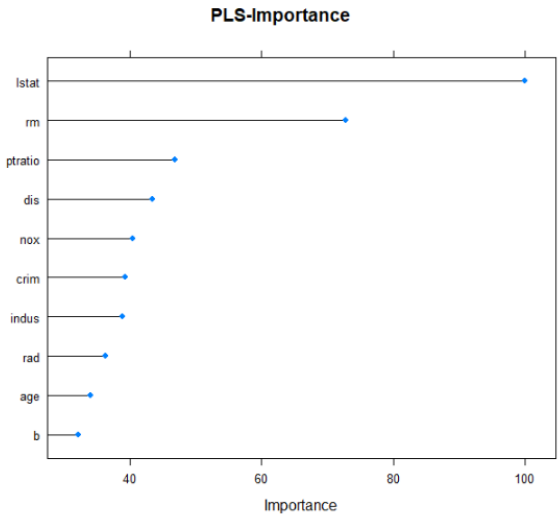


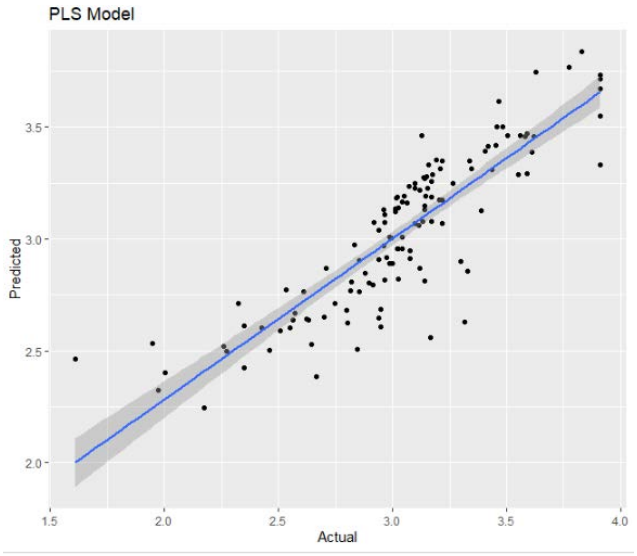


d. Partial Least squares

The optimal number of components chosen was 3.

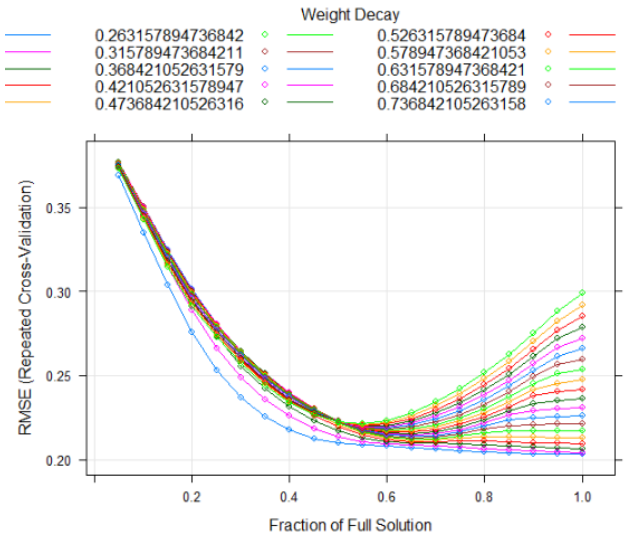


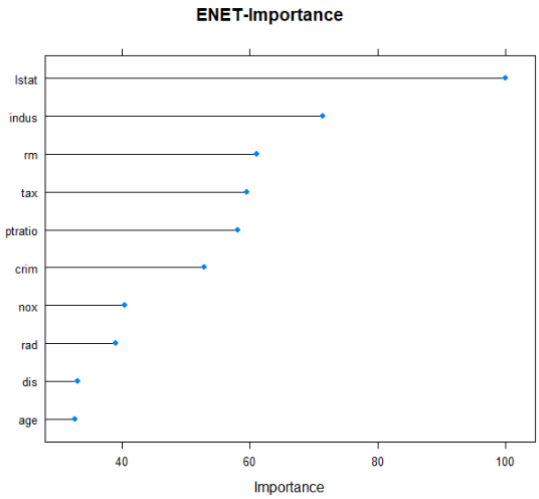


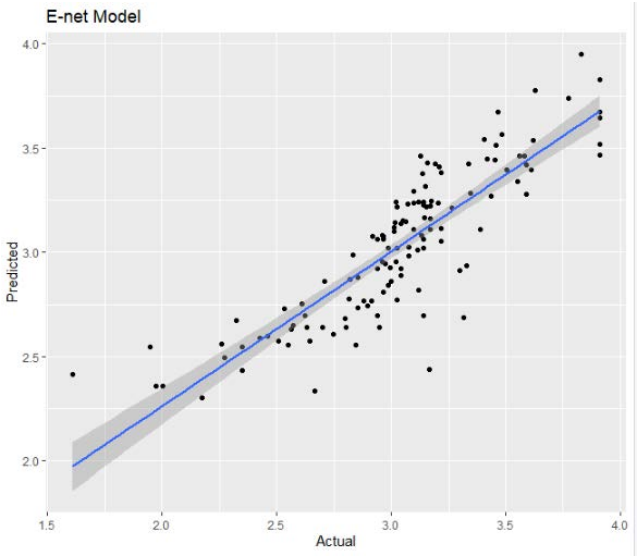


e. Elastic Net

The optimum model selected uses fraction = 1 and lambda = 0.

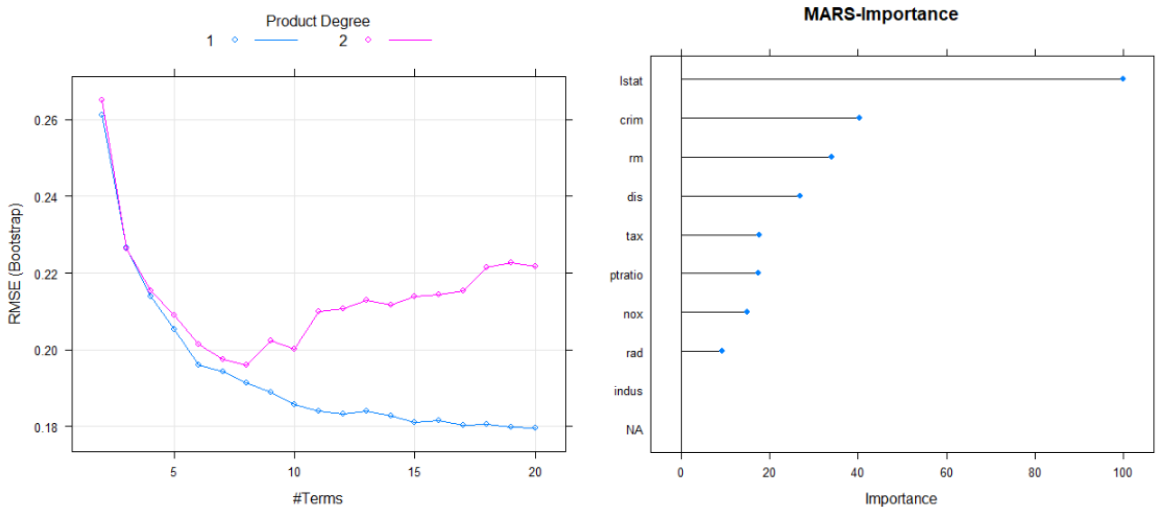


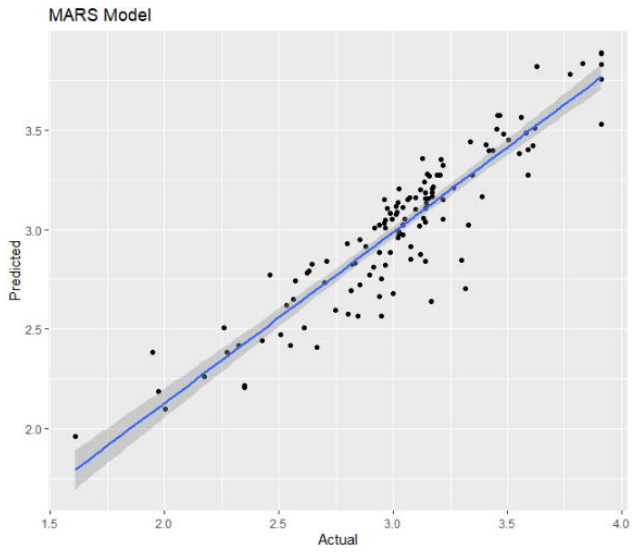




f. MARS

The two tuning parameters for the MARS model is the degree and the number of terms used in the model. The optimum model selected uses nprune = 20 and degree = 1.

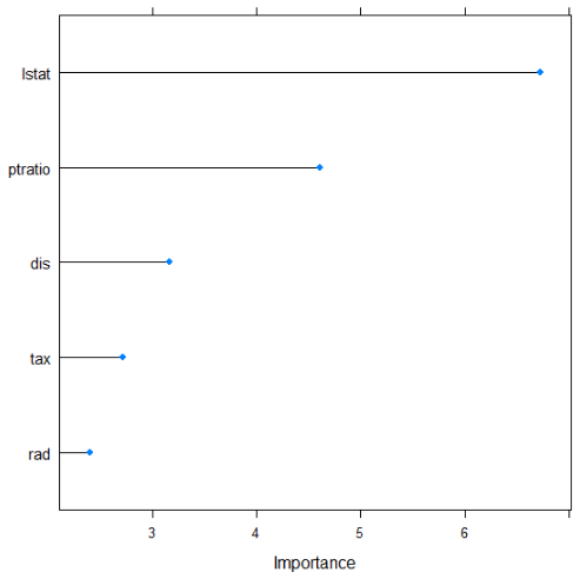


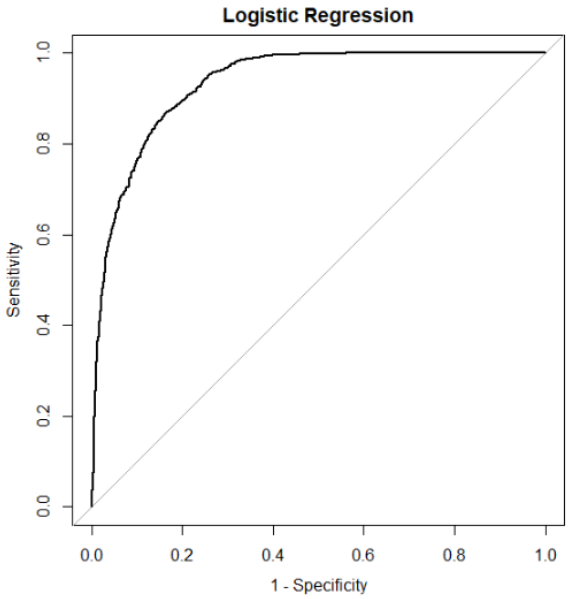


Appendix 2: Supplemental Material for Categorical Outcome Models with Two Levels

a. Logistic

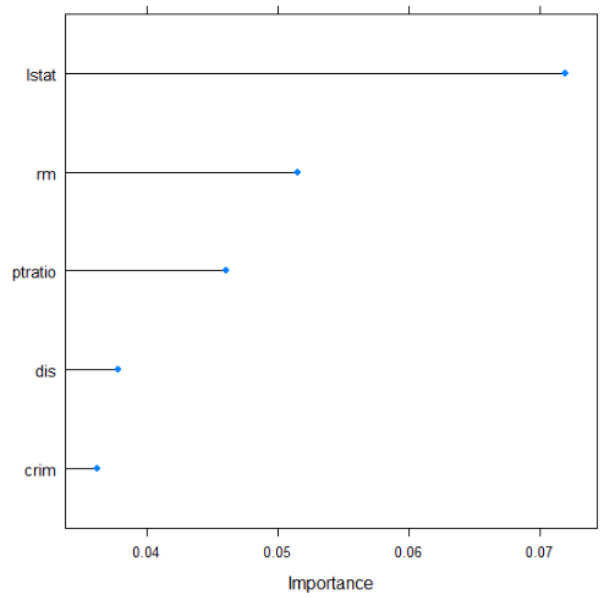
No tuning parameters. Area under the curve: 0.933

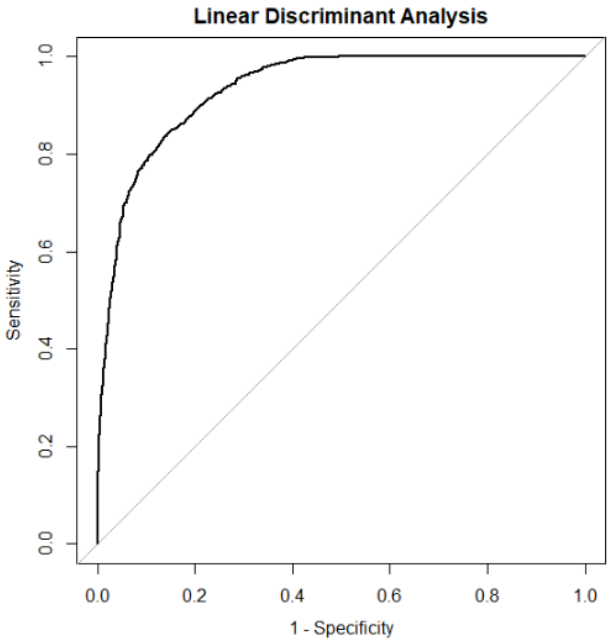




b. LDA

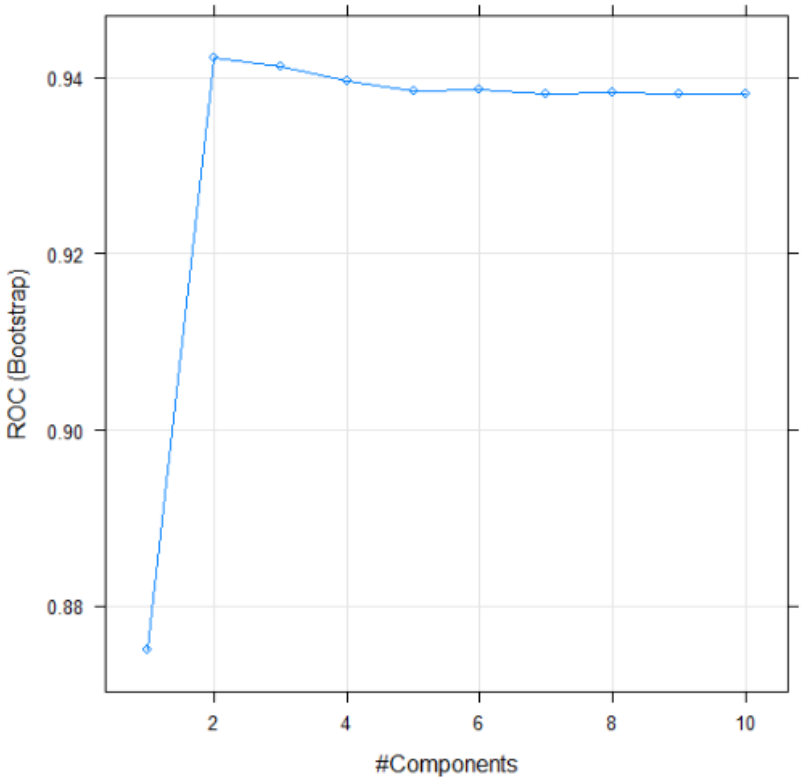
No tuning parameters. Area under the curve: 0.94

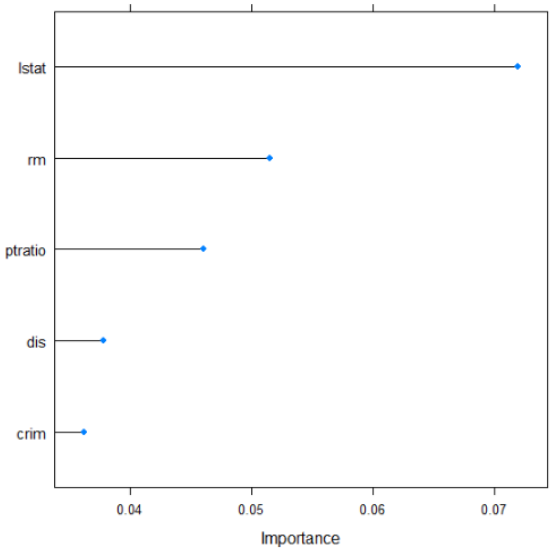


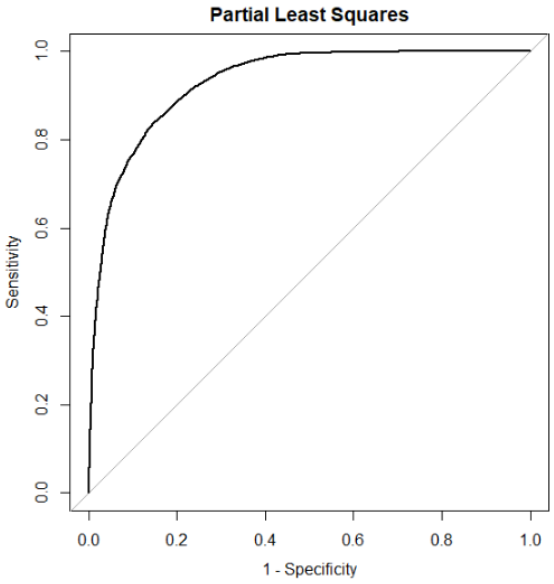


c. PLS Discriminant Analysis

The only linear model trained with any tuning parameters was the partial least squares model. The optimal number of components selected as shown below is two. Area under the curve: 0.9334

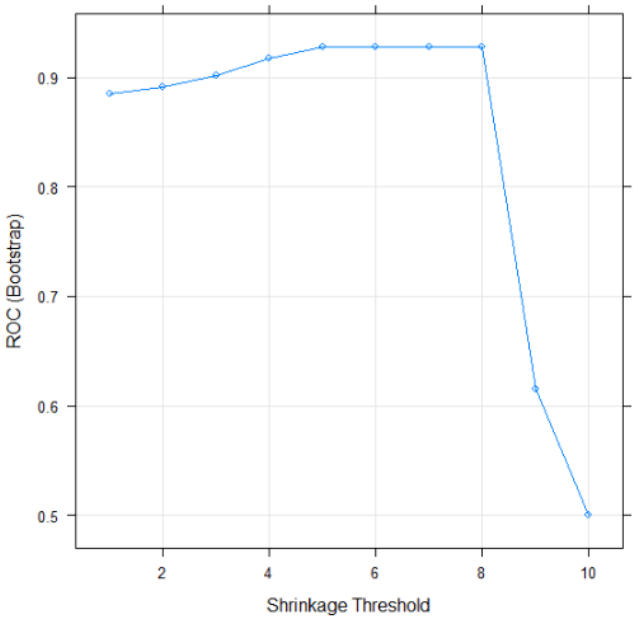


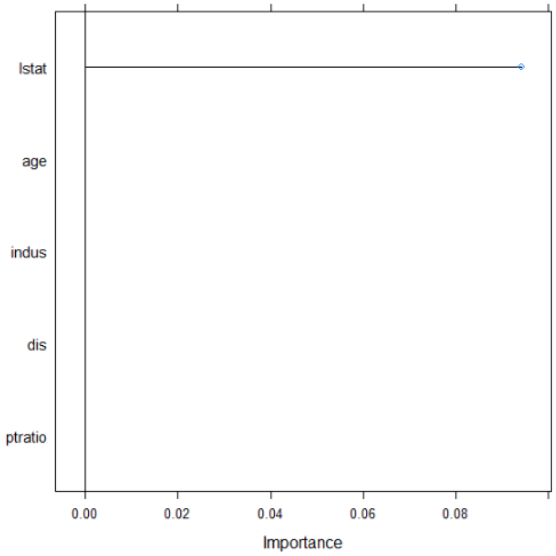


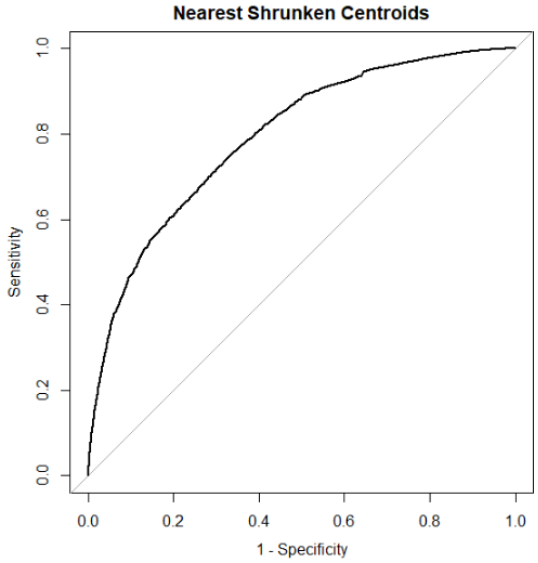


d. Nearest Shrunken Centroid

The optimum model selected a threshold of 6. Area under the curve: 0.8075

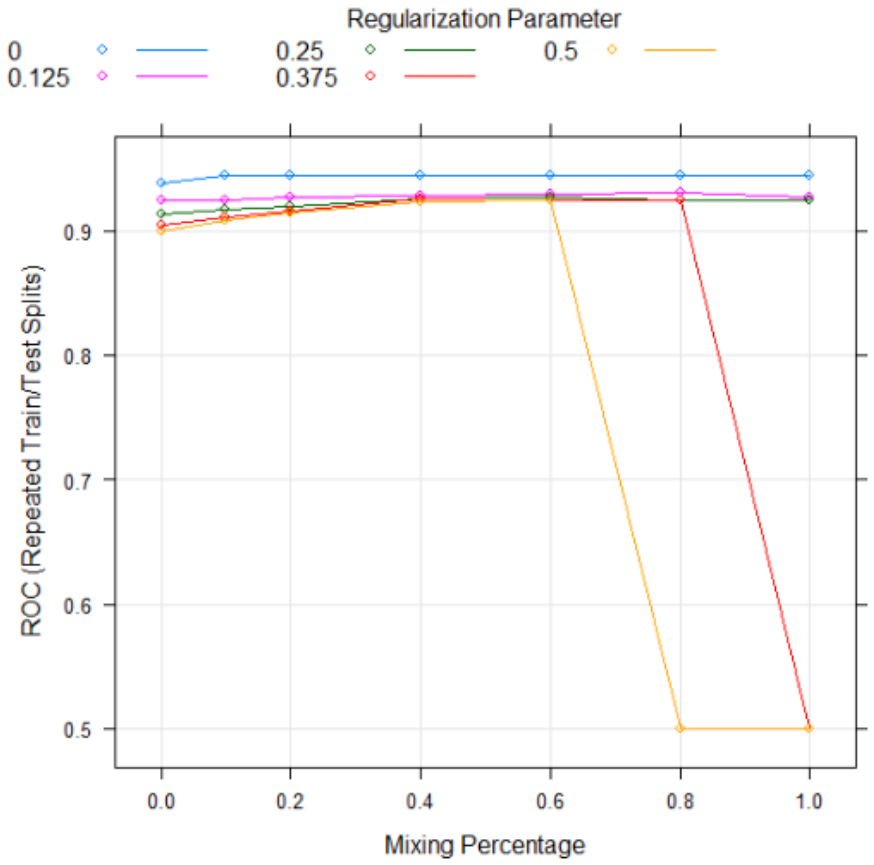


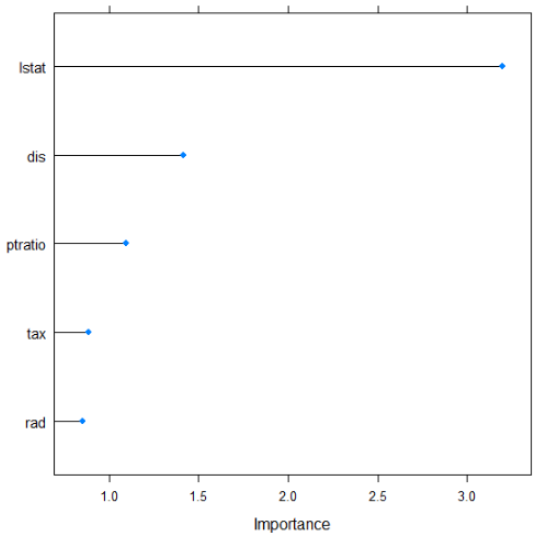


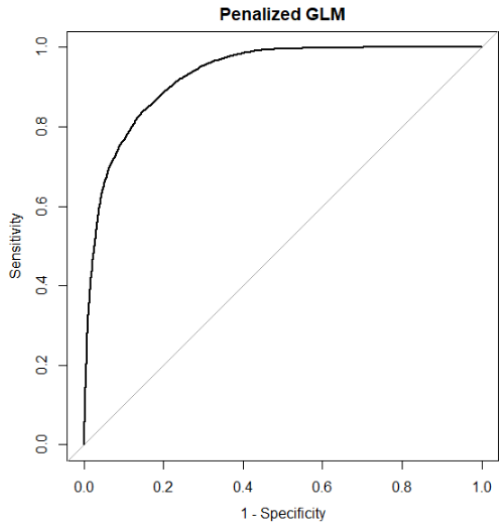


e. Penalized GLM

ROC was used to select the optimal model using the largest value. The optimum model selected uses alpha = 0.1 and lambda = 0. Area under the curve: 0.8915

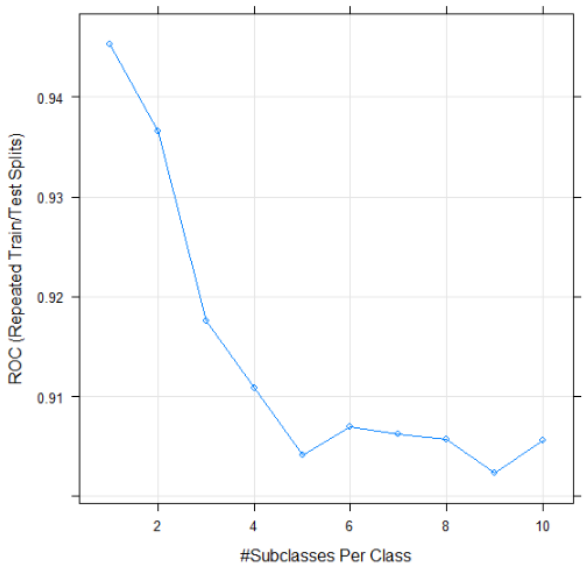


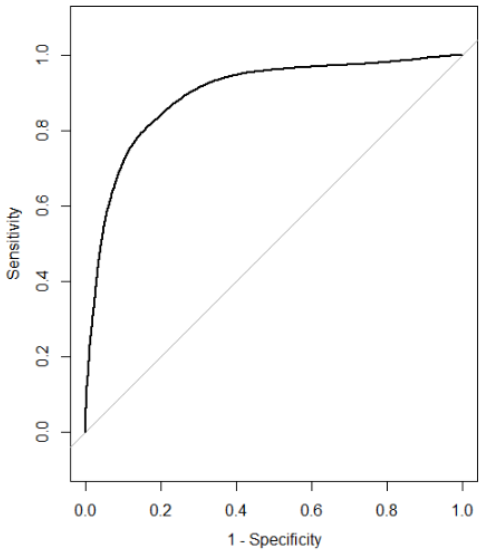




f. MDA

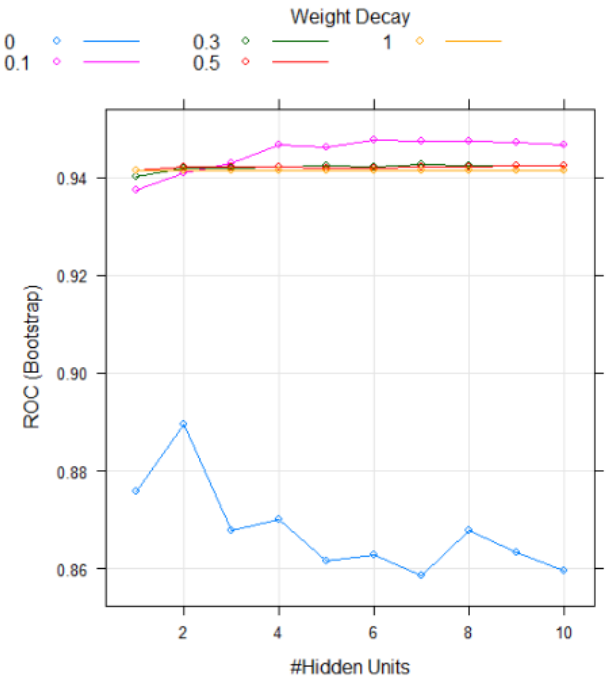
ROC was used to select the optimal model using the largest value. The final value used for the model was subclasses = 1. Area under the curve: 0.8952

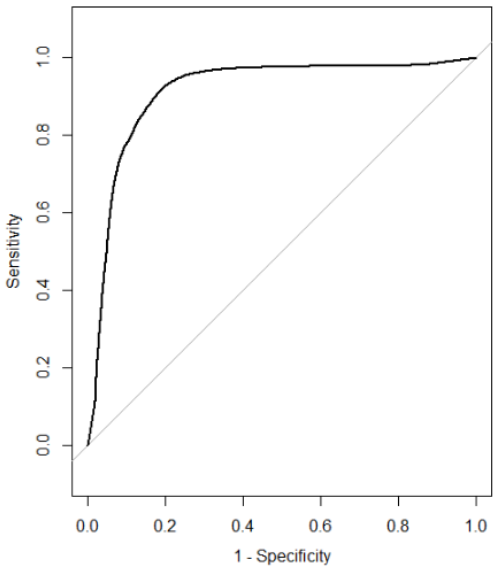




g. Neural Network

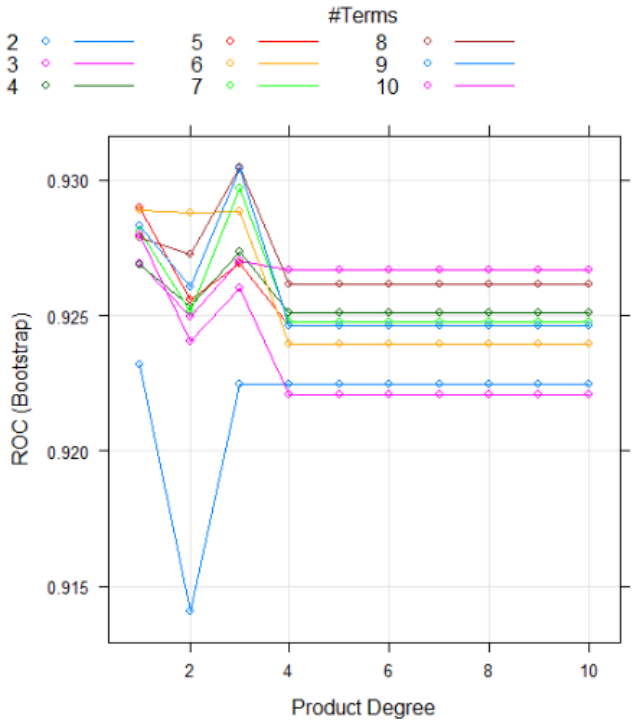
The optimum model selected uses size = 6 and decay = 0.1. Area under the curve: 0.9119

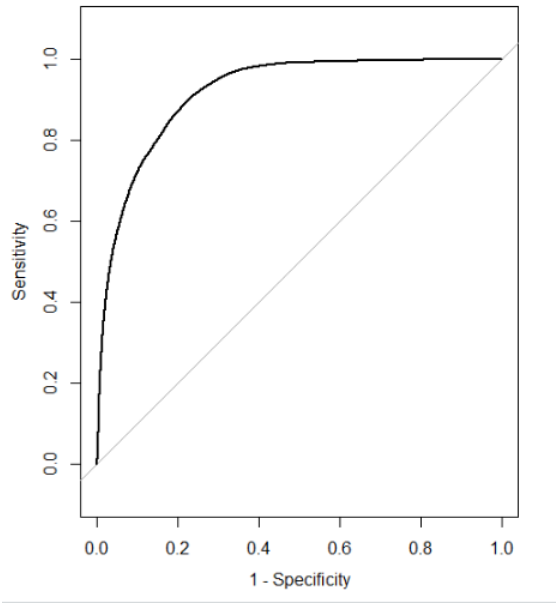




h. FDA

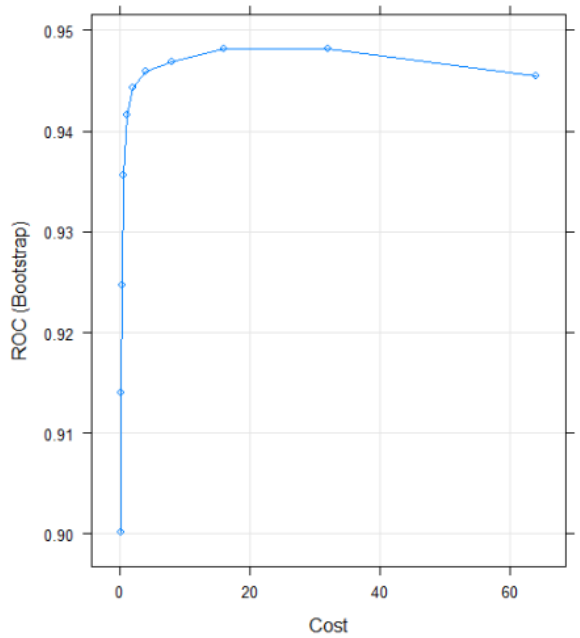
The optimum model selected uses degree = 3 and nprune = 8. Area under the curve: 0.9181

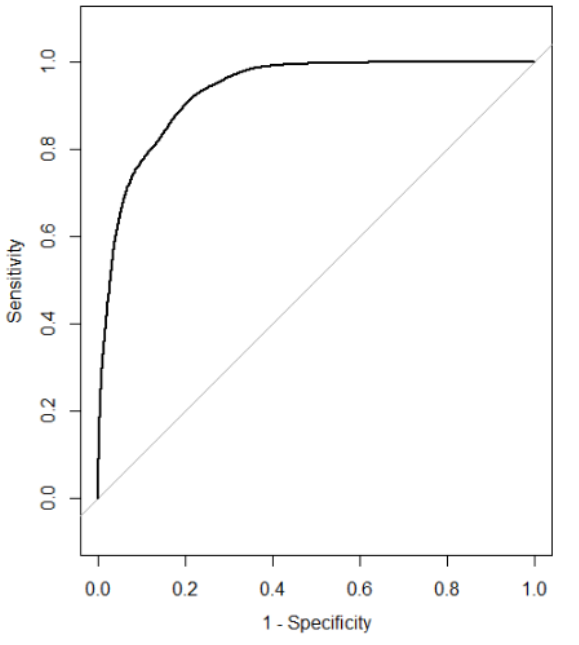




i. SVM

Tuning parameter 'sigma' was held constant at a value of 0.02296279 ROC was used to select the optimal model using the largest value. The optimum model selected uses sigma = 0.02296279 and C = 16. Area under the curve: 0.9343





Sources

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Cantaro, M. (2021, December 13). What You Didn't Know About the Boston Housing Dataset. Medium. https://towardsdatascience.com/things-you-didnt-know-about-the-boston-housing-dataset-2e87a6f960e8

Boston Metropolitan Area: Population from 1790. (2004). DEMOGRAPHIA. http://www.demographia.com/db-bos1790.htm

R Code

```
#

# Boston Housing Study

#


library(caret)

library(corrplot)

library(pls)

library(mlbench)

library(e1071)


# load data set from mlbench package

data(BostonHousing)

str(BostonHousing)


# crim per capita crime rate by town

# zn proportion of residential land zoned for lots over 25,000 sq.ft

# indus proportion of non-retail business acres per town

# chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

# nox nitric oxides concentration (parts per 10 million)

# rm average number of rooms per dwelling
```

```
# age proportion of owner-occupied units built prior to 1940
# dis weighted distances to five Boston employment centers
# rad index of accessibility to radial highways
# tax full-value property-tax rate per USD 10,000
# ptratio pupil-teacher ratio by town
# b 1000(B â 0.63)^2 where B is the proportion of blacks by town
# lstat percentage of lower status of the population
# medv median value of owner-occupied homes in USD 1000âs

# medv is the outcome variable.

# scatter plots of raw predictors vs outcome (medv)
chasIndex <- grep("chas", colnames(BostonHousing))
medvIndex <- grep("medv", colnames(BostonHousing))
Pnames <- colnames(BostonHousing[,-c(medvIndex, chasIndex)])
par(mfrow = c(2,2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BostonHousing")[predictor]))
  plot(BostonHousing$medv, predictors,
       main = paste("Plot of", predictor, "vs. medv"),
       ylab = predictor,
       xlab = "medv"
  )
}

### data pre-processing ###

# check for NA's in outcome
medvNAs <- is.na(BostonHousing$cmedv)
```

```
medvNAs


# check for NA's in predictors
NAs <- is.na(BostonHousing)
NAsTrue <- grep("TRUE", NAs)
NAsTrue


# no missing data, no need for imputation


# histograms of raw predictors
chasIndex <- grep("chas", colnames(BostonHousing))
medvIndex <- grep("medv", colnames(BostonHousing))
Pnames <- colnames(BostonHousing[, -c(medvIndex, chasIndex)])
par(mfrow = c(2, 2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BostonHousing")[predictor]))
  hist(predictors,
       main = paste(predictor),
       xlim =c(min(predictors), max(predictors)),
       xlab = predictor,
       ylab = "Frequency"
  )
}


# check skewness of raw predictors
skew <- apply(BostonHousing[, -chasIndex], 2, skewness)
skew


# log transform crim, nox, dis, rad, tax, lstat, and medv
```

```
transVars <- c("crim", "nox", "dis", "rad", "tax", "lstat", "medv")
for (var in transVars) {
  varIndex <- grep(var, colnames(BostonHousing))
  BostonHousing[,varIndex] <- log(BostonHousing[,varIndex])
}

# skewness of log transformed predictors
skew <- apply(BostonHousing[, -chasIndex], 2, skewness)
skew

# histograms of skew transformed predictors
chasIndex <- grep("chas", colnames(BostonHousing))
medvIndex <- grep("medv", colnames(BostonHousing))
Pnames <- colnames(BostonHousing[, -c(medvIndex, chasIndex)])
par(mfrow = c(2, 2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BostonHousing")[predictor]))
  hist(predictors,
       main = paste(predictor),
       xlim =c(min(predictors), max(predictors)),
       xlab = predictor,
       ylab = "Frequency"
  )
}

# boxplots to check for outliers
chasIndex <- grep("chas", colnames(BostonHousing))
medvIndex <- grep("medv", colnames(BostonHousing))
Pnames <- colnames(BostonHousing[,-c(medvIndex, chasIndex)])
```

```
par(mfrow = c(2,2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BostonHousing")[predictor]))
  boxplot(predictors,
          main = paste("BoxPlot of", predictor),
          ylab = "Value"
  )
}

# examination of the data finds all outliers are valid data

# correlation plot
medvIndex <- grep("medv", colnames(BostonHousing))
chasIndex <- grep("chas", colnames(BostonHousing))
corrplot::corrplot(cor(BostonHousing[-c(medvIndex, chasIndex)]), order="hclust")

# check for correlated predictors
cor90 <- findCorrelation(cor(BostonHousing[,-c(medvIndex, chasIndex)]), cutoff=0.90)
cor90
colnames(BostonHousing[,-c(medvIndex, chasIndex)])[cor90]

# correlation between predictors and outcome
corrValues <- apply(BostonHousing[,-c(medvIndex, chasIndex)],
      MARGIN  = 2,
      FUN = function(x ,y) cor(x, y),
      y = BostonHousing$medv)
corrValues

# plot predictors against each other
```

```
par(mfrow = c(1,1))
pairs(BostonHousing[,-c(medvIndex, chasIndex)])

# check for near zero variance predictors
nearZeroVar(BostonHousing[,-c(medvIndex, chasIndex)])

# no near zero predictors

### linear model ###
set.seed(0)
lmMod <- lm(medv ~ ., data = BostonHousing)
summary(lmMod)

# coefficients for crim, zn, nox, dis, tax, ptratio, and lstat are negative

# the following parameters decrease medv
# crim per capita crime
# zn proportion of residential land zoned for lots over 25,000 sq.ft  value
# dis weighted distances to five Boston employment centers
# nox nitric oxides concentration (parts per 10 million)
# tax full-value property-tax rate per USD 10,000
# ptratio pupil-teacher ratio by town
# lstat percentage of lower status of the population

# the following parameters increase medv
# indus proportion of non-retail business acres per town
# chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
# rm average number of rooms per dwelling
# age proportion of owner-occupied units built prior to 1940
```

```
# rad index of accessibility to radial highways
# b 1000(B â 0.63)^2 where B is the proportion of blacks by town

# transform the predictors with BoxCox no PCA
trans <- preProcess(BostonHousing[,-medvIndex],
   method = c("BoxCox", "center", "scale"))
trans

# apply the non PCA transformation
BHtrans <- predict(trans, BostonHousing)
head(BHtrans)


### explore the data ###

# split outcome and predictors
medvIndex <- grep("medv", colnames(BHtrans))
BHOutcome <- BHtrans$medv
BHPredictors <- BHtrans[,-medvIndex]

# histograms of BoxCox transformed variables
chasIndex <- grep("chas", colnames(BHPredictors))
Pnames <- colnames(BHPredictors[,-chasIndex])
par(mfrow = c(2, 2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BHPredictors")[predictor]))
  hist(predictors,
       main = paste(predictor),
       xlim =c(min(predictors), max(predictors)),
```

```
      xlab = predictor,
      ylab = "Frequency"
  )
}

# boxplots of factor predictor against the outcome variable
par(mfrow = c(1,1))
boxplot(BHOutcome ~ BHtrans$chas, data = BHtrans,
   ylab = "medv",
   xlab = "chas")

### spend the data ###

# select 75% for training and 25% for testing
set.seed(0)
trainIndex <- sample(1:length(BHOutcome), length(BHOutcome)*0.75, replace = FALSE)
head(trainIndex)
length(trainIndex)
BHPredictorsTrain <- BHPredictors[trainIndex,]
BHPredictorsTest <- BHPredictors[-c(trainIndex),]
BHOutcomeTrain <- BHOutcome[trainIndex]
BHOutcomeTest <- BHOutcome[-c(trainIndex)]
medv <- BHOutcomeTrain
BHTrainSet <- cbind(medv, BHPredictorsTrain)
medv <- BHOutcomeTest
BHTestSet <- cbind(medv, BHPredictorsTest)

# correlation between all components
corr <- cor(BHPredictors[,-chasIndex])
```

```
corr

# correlation plot
chasIndex <- grep("chas", colnames(BHPredictors))
corrplot::corrplot(cor(BHPredictors[,-chasIndex]), order="hclust")



### model building ###

# GLM with PCA data
medvIndex <- grep("medv", colnames(BostonHousing))
trans <- preProcess(BostonHousing[,-medvIndex],
                    method = c("BoxCox", "center", "scale", "pca"))
trans

# apply the transformation
BHtransPCA <- predict(trans, BostonHousing)
head(BHtransPCA)
# eight components provide 95% of the variance

# calculate total variance explained by each component
medvIndex <- grep("medv", colnames(BHtransPCA))
chasIndex <- grep("chas", colnames(BHtransPCA))
var <- sapply(BHtransPCA[,-c(medvIndex, chasIndex)], "var")
var <- var / sum(var)
var

# scree plot of PCA data
qplot(c(1:8), var) +
```

```
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)

medvIndex <- grep("medv", colnames(BHtransPCA))
BHOutcomePCA <- BHtransPCA$medv
BHPredictorsPCA <- BHtransPCA[,-medvIndex]
BHPredictorsPCATrain <- BHPredictorsPCA[trainIndex,]
BHPredictorsPCATest <- BHPredictorsPCA[-c(trainIndex),]
BHOutcomePCATrain <- BHOutcomePCA[trainIndex]
BHOutcomePCATest <- BHOutcomePCA[-c(trainIndex)]
medv <- BHOutcomePCATrain
BHTrainPCASet <- cbind(medv, BHPredictorsPCATrain)
medv <- BHOutcomePCATest
BHTestPCASet <- cbind(medv, BHPredictorsPCATest)

# histograms of pca components
chasIndex <- grep("chas", colnames(BHPredictorsPCA))
Pnames <- colnames(BHPredictorsPCA[,-chasIndex])
par(mfrow = c(2, 2))
for (predictor in Pnames) {
  predictors <- as.vector(unlist(get("BHPredictorsPCA")[predictor]))
  hist(predictors,
       main = paste(predictor),
       xlim =c(min(predictors), max(predictors)),
       xlab = predictor,
       ylab = "Frequency"
```

```
 )
}

set.seed(0)
glmMod <- train(medv ~ ., data = BHTrainPCASet,
                method = "glm",
                trControl = trainControl(method = "repeatedcv",
                number = 3, repeats = 5))
glmMod
glmMod$finalModel

plot(varImp(glmMod), 10, main="GLM-Importance")

# predict on test set
set.seed(0)
glmPred <- predict(glmMod, newdata = BHTestPCASet)
glmValues <- postResample(pred = glmPred, obs = BHOutcomePCATest)
glmValues

p <- ggplot(BHTestSet, aes(x = medv, y = glmPred))
p +
 labs(title = "GLM Model", x = "Actual", y = "Predicted") +
 geom_point() +
 stat_smooth(method = glm, formula = 'y~x')



# knn

# find optimum K value
```

```
set.seed(0)
knnMod <- train(medv ~ ., data = BHTrainSet,
                method = "knn",
                # Center and scaling will occur for new predictions too
                #preProc = c("center", "scale"),
                tuneGrid = data.frame(.k = 1:20),
                trControl = trainControl(method = "repeatedcv", repeats = 5))
knnMod
plot(knnMod)
knnMod$finalModel

# k = 3 is optimum value

set.seed(0)
knnMod <- train(medv ~ ., data = BHTrainSet,
                method = "knn",
                tuneGrid = data.frame(.k = 2:5),
                trControl = trainControl(method = "repeatedcv",
                number = 3, repeats = 5))
knnMod

plot(varImp(knnMod), 10, main="KNN-Importance")

# predict on testing set
set.seed(0)
knnPred <- predict(knnMod, newdata = BHTestSet)
knnValues <- postResample(pred = knnPred, obs = BHOutcomeTest)
knnValues
```

```
p <- ggplot(BHTestSet, aes(x = medv, y = knnPred))
p +
   labs(title = "KNN Model", x = "Actual", y = "Predicted") +
   geom_point() +
   stat_smooth(method = glm, formula = 'y~x')



# SVM
set.seed(0)
svmMod <- train(medv ~ ., data = BHTrainSet,
   method = "svmRadial",
   tuneLength = 10,
   trControl = trainControl(method = "repeatedcv",
   number = 3, repeats = 5))
svmMod
plot(svmMod)

plot(varImp(svmMod), 10, main="SVM-Importance")
svmMod$finalModel

# line plot of the average performance
plot(svmMod, scales = list(x = list(log = 2)))

# predict on testing set
set.seed(0)
svmPred <- predict(svmMod, newdata = BHTestSet)
svmValues <- postResample(pred = svmPred, obs = BHOutcomeTest)
svmValues
```

```
p <- ggplot(BHTestSet, aes(x = medv, y = svmPred))
p +
   labs(title = "SVM Model", x = "Actual", y = "Predicted") +
   geom_point() +
   stat_smooth(method = glm, formula = 'y~x')



# PLS
set.seed(0)
plsMod <- train(medv ~ ., data = BHTrainSet,
   method = "pls",
   trControl = trainControl(method = "repeatedcv",
   number = 3, repeats = 5))
plsMod
plot(plsMod)
plsMod$finalModel

plot(varImp(plsMod), 10, main="PLS-Importance")

# predict on testing set
set.seed(0)
plsPred <- predict(plsMod, BHTestSet, ncomp = 8)
plsValues <- postResample(pred = plsPred, obs = BHOutcomeTest)
plsValues

p <- ggplot(BHTestSet, aes(x = medv, y = plsPred))
p +
  labs(title = "PLS Model", x = "Actual", y = "Predicted") +
  geom_point() +
```

```
  stat_smooth(method = glm, formula = 'y~x')



# elastic net
enetGrid = expand.grid(.lambda=seq(0,1,length=20), .fraction=seq(0.05, 1.0, length=20))
set.seed(0)
enetMod = train(medv ~ ., data = BHTrainSet,
                method="enet",
                tuneGrid = enetGrid,
                trControl=trainControl(method="repeatedcv",
                number = 3, repeats=5))
enetMod
plot(enetMod)
enetMod$finalModel

plot(varImp(enetMod), 10, main="ENET-Importance")

# predict on testing set
set.seed(0)
enetPred <- predict(enetMod, BHTestSet)
enetValues <- postResample(pred = enetPred, obs = BHOutcomeTest)
enetValues

p <- ggplot(BHTestSet, aes(x = medv, y = enetPred))
p +
  labs(title = "E-net Model", x = "Actual", y = "Predicted") +
  geom_point() +
  stat_smooth(method = glm, formula = 'y~x')
```

```
# MARS
marsGrid = expand.grid(.degree = 1:2, .nprune = 2:20)
set.seed(0)
marsMod = train(x = BHPredictorsTrain, y = BHOutcomeTrain,
                method = "earth",
                tuneGrid = marsGrid)
marsMod
plot(marsMod)
marsMod$finalModel

plot(varImp(marsMod), 10, main="MARS-Importance")

# predict on testing set
set.seed(0)
marsPred <- predict(marsMod, BHTestSet)
marsValues <- postResample(pred = marsPred, obs = BHOutcomeTest)
marsValues

p <- ggplot(BHTestSet, aes(x = medv, y = marsPred))
p +
  labs(title = "MARS Model", x = "Actual", y = "Predicted") +
  geom_point() +
  stat_smooth(method = glm, formula = 'y~x')

marsMod$finalModel

# MARS importance plot
plot(varImp(marsMod), 10, main="MARS-Importance")
```

```
# compare models based on their cross-validation statistics.

# create a resamples object from the models:
resamp <- resamples(list(KNN = knnMod,
                         SVM = svmMod,
                         GLM = glmMod,
                         PLS = plsMod,
                         ENET = enetMod))
summary(resamp)

# compare testing results
table <- rbind(knnValues,
               svmValues,
               glmValues,
               plsValues,
               enetValues,
               marsValues)
table

# SVM has the smallest RMSE and largest R-squared



#
# bin outcome variable, medv, and use categorical models
#

# histogram of medv
hist(BHOutcome,
```

```
    main = "Histogram of medv",
    xlab = "medv",
    ylab = "Frequency"
)

# check skewness
skew <- skewness(BHOutcome)
skew

if (0) {
# bin medv into 3 equal groups, low, medium, and high price
medvCat <- rep(1, length(BHOutcome))
medv <- BHtrans$medv
BHOutcome <- as.data.frame(cbind(medv, medvCat))
medvSort <- sort(BHOutcome$medv)
split1 <- medvSort[169]
split2 <- medvSort[337]
for (i in 1:nrow(BHOutcome)) {
  if (BHOutcome[i, "medv"] <= split1) {
    BHOutcome[i,"medvCat"] <- 0
  }
  if (BHOutcome[i, "medv"] >= split2) {
    BHOutcome[i,"medvCat"] <- 2
  }
}
BHOutcome$medvCat <- factor(BHOutcome$medvCat, labels = c("low", "medium", "high"))
numLow <- nrow(BHOutcome[which (BHOutcome$medvCat == "low"),])
numLow
numMed <- nrow(BHOutcome[which (BHOutcome$medvCat == "medium"),])
```

```
numMed
numHigh <- nrow(BHOutcome[which (BHOutcome$medvCat == "high"),])
numHigh
}

# bin medv into 2 equal groups, low and high price
medvCat <- rep(0, length(BHOutcome))
medv <- BHtrans$medv
BHOutcome <- as.data.frame(cbind(medv, medvCat))
medvSort <- sort(BHOutcome$medv)
split <- medvSort[252]
for (i in 1:nrow(BHOutcome)) {
  if (BHOutcome[i, "medv"] > split) {
    BHOutcome[i,"medvCat"] <- 1
  }
}
BHOutcome$medvCat <- factor(BHOutcome$medvCat, labels = c("low", "high"))
numLow <- nrow(BHOutcome[which (BHOutcome$medvCat == "low"),])
numLow
numHigh <- nrow(BHOutcome[which (BHOutcome$medvCat == "high"),])
numHigh
head(BHOutcome)
chasIndex <- grep("chas", colnames(BHPredictors))



# training control
ctrl <- trainControl(summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)
```

```
# split outcome into training and testing as previous
BHOutcomeTrain <- BHOutcome[trainIndex,]
BHOutcomeTest <- BHOutcome[-c(trainIndex),]

# training control
ctrl <- trainControl(summaryFunction = twoClassSummary,
                     #method = "LGOCV",
                     classProbs = TRUE,
                     savePredictions = TRUE)



# Logistic Regression Model

levels(BHOutcome$medvCat)

set.seed(1000)
lrFit = train(BHPredictorsTrain[-chasIndex], BHOutcomeTrain$medvCat,
              method = "glm",
              metric = "ROC",
              trControl = ctrl)
lrFit

# predict on test set
lrPred = predict(lrFit, BHPredictorsTest[-chasIndex])
lrValues <- postResample(pred = lrPred, obs =  BHOutcomeTest$medvCat)
lrValues

# test set
```

```
confusionMatrix(data = lrPred,
                reference = BHOutcomeTest$medvCat)


# training set
confusionMatrix(data = lrFit$pred$pred,
                reference = lrFit$pred$obs)


lrImpSim <- varImp(lrFit, scale = FALSE)
lrImpSim
par(mfrow = c(1,1))
plot(lrImpSim, top = 5, scales = list(y = list(cex = .95)))


# AUC
library(pROC)
lrRoc <- roc(response = lrFit$pred$obs,
             predictor = lrFit$pred$high,
             levels = rev(levels(lrFit$pred$obs)))
plot(lrRoc, legacy.axes = TRUE, main = "Logistic Regression")
lrAUC <- auc(lrRoc)
lrAUC


# LDA
library(MASS)
set.seed(1000)
ldaModel <- lda(BHPredictorsTrain[-chasIndex],
                grouping = BHOutcomeTrain$medvCat)
ldaModel
summary(ldaModel)
```

```
set.seed(1000)
ldaFit <- train(x = as.data.frame(BHPredictorsTrain[-chasIndex]),
                y = BHOutcomeTrain$medvCat,
                method = "lda",
                preProc = c("center", "scale"),
                metric = "ROC",
                trControl = ctrl)
ldaFit

# predict on test set
ldaPred = predict(ldaFit, BHPredictorsTest[-chasIndex])
ldaValues <- postResample(pred = ldaPred, obs = BHOutcomeTest$medvCat)
ldaValues

# test set
confusionMatrix(data = ldaPred,
                reference = BHOutcomeTest$medvCat)

# training set
confusionMatrix(data = ldaFit$pred$pred,
                reference = ldaFit$pred$obs)

#ldaImpSim <- varImp(ldaFit, scale = FALSE)
#ldaImpSim
#plot(ldaImpSim, top = 5, scales = list(y = list(cex = .95)))

# AUC
ldaRoc <- roc(response = ldaFit$pred$obs,
              predictor = ldaFit$pred$high,
```

```
        levels = rev(levels(ldaFit$pred$obs)))
plot(ldaRoc, legacy.axes = TRUE, main = "Linear Discriminant Analysis")
ldaAUC <- auc(ldaRoc)



# PLSDA

set.seed(1000)
plsdaModel <- plsda(BHPredictorsTrain[-chasIndex], BHOutcomeTrain$medvCat,
              scale = TRUE,
              ncomp = 10)
plsdaModel

set.seed(1000)
plsFit <- train(BHPredictorsTrain[-chasIndex], BHOutcomeTrain$medvCat,
            method = "pls",
            tuneGrid = expand.grid(.ncomp = 1:10),
            preProc = c("center","scale"),
            metric = "ROC",
            trControl = ctrl)
plsFit
plot(plsFit)

plsImpSim <- varImp(plsFit, scale = FALSE)
plsImpSim
plot(plsImpSim, top = 5, scales = list(y = list(cex = .95)))

# predict on testing
plsPred <- predict(plsFit, BHPredictorsTrain[-chasIndex])
```

```
plsValues <- postResample(pred = plsPred, obs = BHOutcomeTrain$medvCat)
plsValues

# test set
confusionMatrix(data = plsPred,
                reference = BHOutcomeTest$medvCat)

# training set
confusionMatrix(data = plsFit$pred$pred,
                reference = plsFit$pred$obs)

# AUC
plsRoc <- roc(response = plsFit$pred$obs,
              predictor = plsFit$pred$high,
              levels = rev(levels(plsFit$pred$obs)))
plot(plsRoc, legacy.axes = TRUE, main = "Partial Least Squares")
plsAUC <- auc(plsRoc)


# Nearest shrunken Centroids
nscGrid = expand.grid(.threshold = 1:10)

set.seed(1000)
nscFit = train(BHPredictorsTrain[-chasIndex], BHOutcomeTrain$medvCat,
               method = "pam",
               preProc = c("center","scale"),
               tuneGrid = nscGrid,
               metric = "ROC",
               trControl = ctrl)
```

```
nscFit
plot(nscFit)

nscImpSim <- varImp(nscFit, scale = FALSE)
nscImpSim
plot(nscImpSim, top = 5, scales = list(y = list(cex = .95)))

# predict on testing
nscPred <- predict(nscFit, BHPredictorsTrain[-chasIndex])
nscValues <- postResample(pred = nscPred, obs = BHOutcomeTrain$medvCat)
nscValues

# test set
confusionMatrix(data = nscPred,
                reference = BHOutcomeTest$medvCat)

# training set
confusionMatrix(data = nscFit$pred$pred,
                reference = nscFit$pred$obs)

# AUC
nscRoc <- roc(response = nscFit$pred$obs,
              predictor = nscFit$pred$high,
              levels = rev(levels(nscFit$pred$obs)))
plot(nscRoc, legacy.axes = TRUE, main = "Nearest Shrunken Centroids")
nscAUC <- auc(nscRoc)

############ GLM NET MODEL ############
```

```
library(glmnet)



#glmnetModel <- glmnet(x = bioTrain,
# y = injuryTrain,
# family = "binomial")



ctrl <- caret::trainControl(method = "LGOCV",
                            summaryFunction = twoClassSummary,
                            classProbs = TRUE,
                            ##index = list(simulatedTest[,1:4]),
                            savePredictions = TRUE)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(0, .5, length = 5))
set.seed(1000)
glmnMod <- train(x = BHPredictorsTrain[-chasIndex], y = BHOutcomeTrain$medvCat,
                 method = "glmnet",
                 tuneGrid = glmnGrid,
                 preProc = c("center", "scale"),
                 #metric = "ROC",
                 trControl = ctrl)
glmnMod
plot(glmnMod)

# predict on testing
glmnPred <- predict(glmnMod, BHPredictorsTrain[-chasIndex])
```

```
#test set values
glmnValues <- postResample(pred = glmnPred, obs = BHOutcomeTrain$medvCat)
glmnValues


# test set
confusionMatrix(data = glmnPred,
                reference = BHOutcomeTrain$medvCat)



# train set values
confusionMatrix(data = glmnMod$pred$pred,
                reference = glmnMod$pred$obs)
glmnImpSim <- varImp(glmnMod, scale = FALSE)
glmnImpSim
plot(glmnImpSim, top = 5, scales = list(y = list(cex = .95)))



# AUC
glmnRoc <- roc(response = glmnMod$pred$obs,
               predictor = glmnMod$pred$high,
               levels = rev(levels(glmnMod$pred$obs)))
plot(glmnRoc, legacy.axes = TRUE, main = "Penalized GLM")

glmnAUC <- auc(glmnRoc)
glmnAUC

# compare testing results
table <- rbind(lrValues,
               ldaValues,
```

```
        plsValues,
        nscValues,
        glmnValues)
table

table <- rbind(lrAUC,
        ldaAUC,
        plsAUC,
        nscAUC,
        glmnAUC)
table

################################################

# training control
ctrl <- trainControl(summaryFunction = twoClassSummary,
                     method = "LGOCV",
                     classProbs = TRUE,
                     savePredictions = TRUE)

############### MDA ###############
par(mfrow = c(1, 1))
mdaFit <- train(BHPredictorsTrain[-chasIndex],          BHOutcomeTrain$medvCat,
        method = "mda",
        metric = "ROC",
        tuneGrid = expand.grid(.subclasses = 1:10),
        trControl = ctrl)
mdaFit
plot(mdaFit)
```

```
# predict on testing set
mdaPred <- predict(mdaFit, BHPredictorsTrain[-chasIndex])
mdaValues <- postResample(pred = mdaPred, obs = BHOutcomeTrain$medvCat)
mdaValues

#test set
confusionMatrix(data = mdaPred,
                reference = BHOutcomeTrain$medvCat)

#train set
confusionMatrix(data = mdaFit$pred$pred,
                reference = mdaFit$pred$obs)

#mdaImpSim <- varImp(mdaFit, scale = FALSE)
#mdaImpSim
#plot(mdaImpSim, top = 5, scales = list(y = list(cex = .95)))

# AUC
mdaPred <- predict(mdaFit, BHPredictorsTrain[-chasIndex], type = "prob")
mdaRoc <- roc(response = mdaFit$pred$obs,
              predictor = mdaFit$pred$high,
              levels = rev(levels(mdaFit$pred$obs)))
plot(mdaRoc, legacy.axes = TRUE)
mdaAUC <- auc(mdaRoc)
mdaAUC

############### NNet ###############
```

```
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, .3, .5, 1))

maxSize <- max(nnetGrid$.size)

numWts <- (maxSize * (102 + 1) + (maxSize+1)*2) ## 102 is the number of predictors; 2 is the number of
classes


#takes 5 minutes:

nnetFit <- train(BHPredictorsTrain[-chasIndex],          BHOutcomeTrain$medvCat,

          method = "nnet",

          metric = "ROC",

          preProc = c("center", "scale", "spatialSign"),

          tuneGrid = nnetGrid,

          trace = FALSE,

          maxit = 2000,

          MaxNWts = numWts,

          trControl = ctrl)

nnetFit

plot(nnetFit)


# predict on testing set

nnetPred <- predict(nnetFit, BHPredictorsTrain[-chasIndex])

nnetValues <- postResample(pred = nnetPred, obs = BHOutcomeTrain$medvCat)

nnetValues

#test set

confusionMatrix(data = nnetPred,

          reference = BHOutcomeTrain$medvCat)


#train set

confusionMatrix(data = nnetFit$pred$pred,
```

```
            reference = nnetFit$pred$obs)


# AUC
nnetPred <- predict(nnetFit,  BHPredictorsTrain[-chasIndex], type = "prob")
nnetRoc <- roc(response = nnetFit$pred$obs,
           predictor = nnetFit$pred$high,
           levels = rev(levels(nnetFit$pred$obs)))
plot(nnetRoc, legacy.axes = TRUE)
nnetAUC <- auc(nnetRoc)
nnetAUC


############### FDA ###############


marsGrid <- expand.grid(.degree = 1:10, .nprune = 2:10)


fdaTuned <- train(BHPredictorsTrain[-chasIndex],              BHOutcomeTrain$medvCat,
              method = "fda",
              # Explicitly declare the candidate models to test
              tuneGrid = marsGrid,
              trControl = ctrl)


fdaTuned
plot(fdaTuned)




# predict on testing set
fdaPred <- predict(fdaTuned, BHPredictorsTrain[-chasIndex])
fdaValues <- postResample(pred = fdaPred, obs = BHOutcomeTrain$medvCat)
```

```
fdaValues


#test set
confusionMatrix(data = fdaPred,
                reference = BHOutcomeTrain$medvCat)


#train set
confusionMatrix(data = fdaTuned$pred$pred,
                reference = fdaTuned$pred$obs)




# AUC
fdaPred <- predict(fdaTuned, BHPredictorsTrain[-chasIndex], type = "prob")
fdaRoc <- roc(response = fdaTuned$pred$obs,
              predictor = fdaTuned$pred$high,
              levels = rev(levels(fdaTuned$pred$obs)))
plot(fdaRoc, legacy.axes = TRUE)
fdaAUC <- auc(fdaRoc)
fdaAUC


############### Support Vector Machines ###############
library(kernlab)
library(caret)
sigmaRangeReduced <- sigest(as.matrix(BHPredictorsTrain[-chasIndex]))

svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1],
                               .C = 2^(seq(-4, 6)))
svmRModel <- train(BHPredictorsTrain[-chasIndex],              BHOutcomeTrain$medvCat,
```

```
              method = "svmRadial",
              metric = "ROC",
              preProc = c("center", "scale"),
              tuneGrid = svmRGridReduced,
              fit = FALSE,
              trControl = ctrl)

svmRModel
plot(svmRModel)



# predict on testing set
svmRPred <- predict(svmRModel, BHPredictorsTrain[-chasIndex])
svmRValues <- postResample(pred = svmRPred, obs = BHOutcomeTrain$medvCat)
svmRValues

#test set
confusionMatrix(data = svmRPred,
              reference = BHPredictorsTrain[-chasIndex])

#train set
confusionMatrix(data = svmRModel$pred$pred,
              reference = svmRModel$pred$obs)

# AUC
svmRPred <- predict(svmRModel, BHPredictorsTrain[-chasIndex], type = "prob")
svmRRoc <- roc(response = svmRModel$pred$obs,
             predictor = svmRModel$pred$high,
             levels = rev(levels(svmRModel$pred$obs)))
```

```
plot(svmRRoc, legacy.axes = TRUE)
svmRAUC <- auc(svmRRoc)
svmRAUC

########## Naive Bayes ##########

install.packages("klaR")
library(klaR)
nbFit <- train(BHPredictorsTrain[-chasIndex],          BHOutcomeTrain$medvCat,
        method = "nb",
        metric = "ROC",
        ## preProc = c("center", "scale"),
        tuneGrid = data.frame(.fL = 2,.usekernel = TRUE,.adjust = TRUE),
        trControl = ctrl)

nbFit
## plot(nbFit) No tuning parameter for nb

plot(nbFit)



# predict on testing set
nbPred <- predict(nbFit, BHPredictorsTrain[-chasIndex])
nbValues <- postResample(pred = nbPred, obs = BHOutcomeTrain$medvCat)
nbValues

#test set
confusionMatrix(data = nbPred,
         reference = BHPredictorsTrain[-chasIndex])
```

```
#train set
confusionMatrix(data = nbFit$pred$pred,
                reference = nbFit$pred$obs)


# AUC
nbPred <- predict(nbFit, BHPredictorsTrain[-chasIndex], type = "prob")
nbRoc <- roc(response = injuryTrain,
             predictor = nbPred[,1],
             levels = levels(injuryTrain))
plot(nbRoc, legacy.axes = TRUE)
nbAUC <- auc(nbRoc)
nbAUC

############################################################


# compare testing results
table <- rbind(mdaValues,
               nnetValues,
               fdaValues,
               svmRValues#,nbValues)
)
table


table <- rbind(mdaAUC,
               nnetAUC,
               fdaAUC,
               svmRAUC)
table
```