Joseph Annand

Dr.Kokkotos

DSE 6111

Final Project

Predictive Modeling Final Project: A Comparison of Modeling Techniques

**Executive Summary**

In this report, we look at three different problems, each pertaining to their own data set. The first problem looks at predicting a qualitative response: whether a student in higher education in Portugal will drop out of school based one a set of 20 predictor variables. A training subset of the original data is used to create various classification models including logistic regression, linear and quadratic discriminant analysis, naïve Bayes’ classifier, the non-parametric k-nearest neighbor algorithm, and classification tree methods. Every trained model was used to predict whether a student dropped out or not based on the same test data set. Confusion matrices were generated for each model and the test error rate was determined by the percentage of predictions that were misclassified. The results of the comparison of the models determined that logistic regression and linear discriminant analysis were best at classifying the dropout response when modeled using a set of statistically significant predictors, i.e. predictors that had coefficient estimates with p-values less than 0.05. Their test error rates were 12.4% and 12.5%, respectively. More flexible models yielded larger misclassification rates than more rigid models. In the quantitative model, we sought to predict the quality of white wine based on various chemical characteristics of each wine in the data set. The models that were trained using half of the original data set in order to predict the quality of wine included multiple linear regression, regularization general linear models (Ridge Regression and the LASSO), subset selection general linear models (best subset, forward stepwise, and backward stepwise), dimension reduction partial least squares model, regression tree, pruned regression tree, bagging, random forest, and boosting. All trained models were used to predict the quality of white wine based on the test data set. Ridge regression, the LASSO, and subset selection models did not improve the test mean squared error compared to a multiple linear regression model that only used statistically significant predictors. Tree-based methods (bagging, random forests, and boosting) resulted in a significant decrease in the test mean squared error. The most important predictors of wine quality as determined by the random forest model were alcohol, free sulfur dioxide, volatile acidity, and chlorides. The random forest model had the lowest test error with a test MSE of 0.407, which means, on average, the random forest will predict a quality rating 0.638 units off the actual value. The principal components regression problem involved predicting the average credit balance of customers based on financial and other demographic factors. The model was fit using a training data set and the ideal number of principal components was determined using cross-validation. That number was determined to be 11. The PCR model yielded a test MSE of 10691.12, which means that on average, the model is off by $103.40 in its prediction of a customer’s average credit balance. This test MSE was no different than that of a multiple linear regression model trained with the same training set and predicting average credit balance based on the same test data set.

**Data & Approach**

*Qualitative Problem*

The data set includes 4424 records of undergraduate students in higher education who either dropped out of school or did not drop out. Each record has 21 attributes that are evaluated and used as predictor variables in a model to predict if a student drops out or not. Of the 4424 records, 1421 of them dropped out, which is equal to a little less than a third of the total number of records. The original data set includes three possible values of the target variable: dropout, enrolled, and graduate. In this analysis, all records where the target was equal to enrolled and graduate were set to “Student” to indicate that the student associated with that record has not or did not drop out. Before loading the data set, the column names had to be manually adjusted to remove spaces and parentheses in the names. Upon successfully loading data into R Studio program and adjusting the values in the target column to meet criteria for binary classification problem, the target column was converted from character class to factor class.

The total data set was split in half where one half was assigned to the training data set, which was used to train the various models to predict whether a student will drop out or not. The second half of the data was assigned to the test data and used to measure the accuracy of the model in predicting the response.

The following models are trained using the training set and then used to predict the test error rate using the test data set: logistic regression, linear discriminant analysis, quadratic discriminant analysis, naïve Bayes classifier, k-nearest neighbor, classification tree model, bagging, random forest, and boosting.

*Quantitative Problem*

The data set consists of 4898 observations with 12 attributes connected to each observation. Each observation is a record of 12 different characteristics of a different type of wine. The characteristics of interest include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxides, total sulfur dioxides, total sulfur dioxides, density, pH, sulphates, alcohol, and quality.

For the purpose of this research, we are interested in determining the best regression model to predict the quality of the wine using some, or all, of the other characteristics in the data set.

We begin with a multiple linear regression model and then try to improve upon it using Ridge Regression and The LASSO shrinkage class models. Following shrinkage methods, subset selection models; including best subset, forward stepwise, and backward stepwise; are trained and evaluated. Partial least squares and various tree-based methods are used to model the data and predict the quality of white wine. All models are trained using the same subset of observations of the total data set, which is equal to half the total number of observations. The test mean squared error is determined by predicting the quality of white wine from the observations of the test data set, which is half of the total data set not used in the training set.

*Principal Components Regression Problem*

For the Principal Components regression problem, the Credit data set that is part of the ISLR2 package is used. The focus of this problem is to predict the average credit card balance of a credit holder based on a variety of characteristics. The data set includes 400 records and 10 different features that are not the customer’s average balance. these features include Income, Credit limit, Credit rating, number of cards, Age, Education, home ownership status, student status, marital status, and region. The data set is split in half to form a training data set and test data set.

The PCR model is trained using the training data set. Each predictor is standardized prior to generating the principal components. A ten-fold cross-validation error is computed for each possible number of principal components.

**Detailed Findings**

*Qualitative Problem*

To begin, the total data set is fit using a logistic regression model. The summary of the model reveals information about each predictor. The p-value of each predictor variable is calculated, and those with a p-value less than 0.05 are statistically significant. This information is used to subset our full data set to include only the statistically significant predictors.

Using only the significant predictors from the first logistic regression model, a new logistic regression models trained using the training data set. The contrasts() function reveals that R has assigned 0 to “Dropout” and 1 to the “Student” response. Therefore, if the prediction value is less than 0.5, the response is equal to “Dropout” while all other predicted responses are set equal to “Student.” The test error rate for the trained logistic regression model is 12.4%.

A linear discriminant analysis (LDA) model is trained and used to predict the response. The LDA model yields a similar test error rate as the logistic regression model with a misclassification rate of 12.5%. Unsurprisingly this is not much different from logistic regression since there are only two classes of the response.

The training data is fit to a quadratic linear discriminant analysis (QDA) model and used to predict the responses in the test data. The QDA model yields a test error rate of 15.3%, slightly worse than logistic regression and LDA. The increase in test error rate suggests that the increased flexibility of the QDA model leads to a worse prediction error rate. This may lead us to believe that the assumption that there is a common covariance among the two response classes is better suited for this problem rather than assuming each class has a its own covariance matrix.

A naive Bayes classifier model was applied to the training data set. The model yields a test error rate of 16.3%, which is considerably worse than those of the logistic regression and LDA models.

The non-parametric K-nearest neighbor (KNN) model is evaluated with four different values of k: 1, 3, 5, and 10. Figure 1 shows the test error rates for the KNN model for each value of k. The model was trained using the same training data set as the previous models that were evaluated. The lowest test error amongst the four k values is 21.2%. Interestingly, similar to the observations in the test error rate for QDA, the results of the KNN modeling suggests that less flexibility leads to better accuracy: the test error rate is higher for k = 1 or 3 compared to k = 5 and 10.

*A screenshot of a video game

Description automatically generated*

Figure 1

Using all predictors, a classification tree model is trained using the training data set. Figure 2 shows the tree model that was created. The tree has 8 terminal nodes, and the variables actually used in the tree construction are second semester curricular units approved, tuition fees up-to-date, age at enrollment, second semester curricular units enrolled, and second semester curricular units grade. The training error rate is 12.9% while the test error rate is 13.7%.

A diagram of a graph

Description automatically generated with medium confidence

Figure 2

Pruning the classification tree model attempts to reduce the test error rate. Cross-validation is used to determine what tree size yields the lowest training error. Figure 3 illustrates that plotting error over the size of the tree shows that a tree size of 7 yields the lowest error.

A graph with a line

Description automatically generated

Figure 3

The tree is pruned so that its size matches that determined from the plot. Figure 4 is a plot of the pruned tree. In the pruned tree, second semester curricular units grade is no longer a predictor used. The pruned tree has 7 terminal nodes. Despite the changes, the pruned tree does not perform any differently than the original classification tree, but it did illustrate a simple tree that cuts down the number of predictors needed to create an equally accurate model.

A graph with text and numbers

Description automatically generated with medium confidence

Figure 4

A bagging model is used to predict whether a student will dropout or not. The bagging model is a random forest that uses all the predictors in the data set. The model is trained using the training data set. The out-of-bag observations estimated a test error rate of 13.5%. Despite using bootstrap procedure, the test error rate is equal to that of the original and pruned classification trees.

Using the training data set, a random forest is used to model the data. Similar to the bagging model, a number of decision trees are built using a bootstrap procedure. The random forest model considers a random sample of the predictors that is equal to the square root of the total number of predictors, which is four, at each split rather than all of the predictors. Figure 5 visualizes the importance of the predictors in creating the model and predicting whether a student will drop out. It also reveals that first and second semester curricular units approved, tuition fees being up-to-date, and second semester curricular unit grades are the most important predictors.

The forest contains 500 trees and the out-of-bag estimate of the error rate is 12.8%. This differs from the actual test error rate calculated using the test data set, which was 13.4%. The random forest does not perform considerably differently than the bagging model.

A graph with a number of text

Description automatically generated with medium confidence

Figure 5

Boosting models with various shrinkage parameters were trained using the student dropout data. Each boosting model was created assuming a Bernoulli distribution of the response to account for the binary classification problem, 1000 trees, and an interaction depth of 7 to match that of the pruned classification tree. Figure 6 records the test error associated with different shrinkage parameters. Small shrinkage parameters, 0.001, and 0.01, resulted in 100% test error rate. The lowest test error rate observed was from a shrinkage parameter of 0.750. That test error rate is equal to 19.6%.

A screenshot of a computer

Description automatically generated

Figure 6

Figure 7 summarizes what was learned from creating all the different classification models. Logistic regression and linear discriminant analysis yield the lowest test error rate at 12.4% and 12.5%, respectively.

A screenshot of a computer

Description automatically generated

Figure 7

*Quantitative Problem*

Correlations between the predictors and the response are checked prior to modeling using least squares regression. The correlation to quality is 1.000 as expected. There are not any other attributes particularly strongly correlated to the quality of white wine. Alcohol and density have correlation coefficients of greatest magnitude. Chlorides, volatile acidity, and total sulfur dioxide are next strongly correlated to quality.

*A graph of different sizes and shapes

Description automatically generated with medium confidence*

Figure 8

When fitting all the white wine data to a multiple least squares linear regression, The F-statistic for the multiple linear regression with all predictors is well above 1, meaning our regression fit is statistically significant. The predictors that are statistically significant and have a p-value less than 0.05 are residual sugar, free sulfur dioxide, density, pH, sulphates, and alcohol. Figure 8 shows the residuals of the least squares model. The three distinct diagonal lines may be attributed to the limited possibilities of values for the response.

A multiple linear regression model was then trained using half of the observations from the white wine data set. Unlike the first model that included all the possible predictor variables, the trained model used the seven statistically significant predictors from the first model. The F-statistic for the trained model is still much greater than 1. All the p-values for the predictors are less than 0.05, so each predictor is significant in this model.

Rounding to three decimal places, the test mean squared error (MSE) for the trained linear model with all predictors is 0.573, which is slightly greater than that of the trained model with seven predictors. The F-statistic is noticeably lower for the model with all predictors, but it is still well above 1.

The training data is fit using a ridge regression model, which is a shrinkage method that will attempt to shrink the estimated coefficients to reduce variance and better fit the data. To determine the best tuning parameter, or lambda, cross-validation is used. Figure 9 shows the mean squared error plotted over the logarithm of the tuning parameter. The tuning parameter that minimizes the MSE is 0.0384732.

A graph of a curve

Description automatically generated

Figure 9

The ridge regression model yields a test mean squared error of 0.574, which is very similar to the test error the linear model with all predictors yielded.

A graph of a number of lines

Description automatically generated with medium confidence

Figure 10

The training data is fit to a LASSO model, which is similar to the ridge regression model; however, the LASSO has the ability to shrink coefficient estimates to zero and essentially perform a best subset selection. A plot of coefficients over L1 seen in Figure 10 show that many coefficients converge to zero or near zero.

A graph of a log line

Description automatically generated with medium confidence

Figure 11

Figure 11 shows a similar plot to that in Figure 10 but for the LASSO model. The tuning parameter that minimizes the mean squared error for the Lasso model is 0.0005713149. With a tuning parameter so close to zero, the shrinkage effect on the coefficients is minimal and the model is likely to not change the test error significantly compared to least squares regression.

Looking at the coefficients of the Lasso model, the coefficient for fixed acidity is zero, which essentially means that the model removed fixed acidity as a predictor of quality.

A partial least squares model is trained using the training data set and fit using a cross-validation method.

A line graph with text

Description automatically generated

Figure 12

The root mean squared error determined through cross-validation gradually decreases from 1 component to 8 components. The value of the root mean squared error does not change much with the number of components greater than 8. The summary of the partial least squares fit provides the amount of variance that is explained in the predictors and the response. The highest amount of variance explained in the response is 27.95% while the highest amount of variance in the predictors is 100% when 11 components are used.

Figure 12 shows the mean squared error plotted over the number of components used within the partial least squares model. As the number of components increases from 0 to 3, the mean squared error reduces significantly. After more than 3 components are used in the model, the mean squared error does not change much.

The partial least squares model is used to predict the quality of the white wine with 3 as the number of components. The test mean squared error for PLS is 0.578.

The best subset selection model is used to select the best set of predictors that minimizes the test mean squared error. A validation set approach is used to find the best subset. This approach finds that the best subset of predictors is fixed acidity, volatile acidity, residual sugar, free sulfur dioxide, density, pH, sulphates, and alcohol, which is a total of 8 predictors from the original 11. The minimized test MSE is 0.573.

Best subset selection is performed again using a k-fold cross-validation approach this time. With k = 10, the cross-validation also determines that the best subset contains 8 predictors. Figure 13 plots the estimated MSE over the number of predictors included in the subset. The test MSE is calculated to be 0.568.

A line graph with numbers and lines

Description automatically generated

Figure 13

Both forward and backward stepwise subset selection using cross-validation determines that the best model uses 8 predictors. Figures 14 and 15 show the estimated MSE plotted over number of predictors in the subset for forward stepwise and backward stepwise, respectively. While forward stepwise yields the same test error as best subset selection, the backward stepwise selection model yields a slightly lower test MSE of 0.563.

A line graph with numbers and a line

Description automatically generated

Figure 14

A line graph with numbers and a line

Description automatically generated

Figure 15

The training data is used to create a regression tree model that will be used to predict the quality of white wine using the test data set observations. The regression tree uses four predictors to construct the tree: alcohol, volatile acidity, free sulfur dioxide, and chlorides. Alcohol is the most used predictor, as it used in the first decision and in the last level of the tree The tree contains 6 terminal nodes.

Using the regression tree to predict the quality of white wine, the test MSE is 0.593.

A diagram of a chemical reaction

Description automatically generated

Figure 16

In attempt to improve the regression tree model, the original tree is pruned. Figure 17 shows that using cross-validation to determine the ideal size of the tree does not give any new information: the ideal size of a regression tree for our training data is the same as that of the original tree we created. Predictably, the pruned tree with a size of 6 seen in Figure 16 gives the same tree and yields virtually the same test MSE.

A line graph with numbers and points

Description automatically generated

Figure 17

A bagging model is created to attempt to use a series of bootstrapped regression trees to better predict the quality of white wine. The bagging model considers all predictors as split candidates at each split in the trees. 500 trees are used in the model, the mean of squared residuals is 0.433, and 43.96% of the variance is explained with this model.

Using the bagging model to predict quality in the test data set yields a test MSE of 0.408, which is the lowest of any model thus far.

Similar to bagging, the random forest tries to predict the response using 500 regression trees. The random forest model, however, uses considers only a random selection of 3 predictors at each split. The model has a slightly lower mean of squared residuals than the bagging model at 0.423, but explains 45.23% of the variance of the model.

Looking at the graphs indicating the importance of predictors in the random forest model in Figure 18, alcohol, volatile acidity, free sulfur dioxide, and chlorides are revealed to be four of the most important predictors to predict the quality of white wine. These are the same predictors featured in the pruned regression tree model.

A chart of alcohol content

Description automatically generated with medium confidence

Figure 18

The random forest model yields the lowest test MSE thus far at 0.407 when applied to the test data set.

A boosting model, which attempts to improve on bagging by sequentially creating trees and using information from previous trees, was created using a set of different shrinkage parameter. Each model used 1000 trees and a tree size equal to 4. Looking at Figure 19, as the shrinkage parameter increases, the training error decreases; however, the minimum test error, 0.467, is seen when lambda equals 0.02.

A screenshot of a computer

Description automatically generated

Figure 19

A screenshot of a computer

Description automatically generated

Figure 20

Figure 20 shows the test errors for all the models used to predict the quality fo white wine with the test data set. Least squares regression and the generalized linear models (GLMs) all yield a similar test mean squared error when trying to predict the quality of white wine using the test data set. The regression tree model performs worse on the test data than the GLMs; however, other tree-based methods yield significantly lower mean squared errors than all other models. The best of these is the random forest model with a test MSE of 0.407, which means, on average, the random forest will predict a quality rating 0.638 units off the actual value.

*Principal Components Regression Problem*

A graph showing a number of components

Description automatically generated

Figure

The validation plot in Figure 21 shows that the number of components that minimize the mean squared error is 11. When the number of components is set equal to 11, the PCR model yields a test MSE of 10691.12, which means that on average, the model is off by $103.40 in its prediction of a customer’s average credit balance.

Fitting the data to a least squares regression demonstrates that the PCR model does not improve the test MSE compared to an ordinary multiple linear regression model.

**Validity & Reliability Assessment**

Predicting a student dropping out school in higher education is best modeled using logistic regression or linear discriminant analysis. Models with more flexibility yield higher misclassification rates; therefore, based on the findings in this report, it is recommended that the classification model used on this data or data similar to this set is rigid and has a more rigid or linear decision boundary. We used a validation set approach to compare the models in this report, and only one training set and test set was used on each model. In the future, the test error may be determined using a cross-validation approach so that way a distribution of test errors is generated for each model. Tree-based methods yield the lowest test mean squared errors for modeling white wine quality. Like the qualitative problem, more test data sets should be used in the future to ensure robustness. From the comparison of different models, it was discovered that alcohol, volatile acidity, free sulfur dioxide, and chlorides were the most important predictors of white wine quality. These predictors should be focused on in future projects involving this data or similar data. Principal components regression on the Credit data did not yield different results from an ordinary least squares model when predicting credit balance; however, it could be useful in predicting other variables in the data set.

**References**

Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T., and Reis,J.. (2009). Wine Quality. UCI Machine Learning Repository. <https://doi.org/10.24432/C56S3T>.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). Springer. https://www.statlearning.com/

M.V.Martins, D. Tolledo, J. Machado, L. M.T. Baptista, V.Realinho. (2021) "Early prediction of student’s performance in higher education: a case study" Trends and Applications in Information Systems and Technologies, vol.1, in Advances in Intelligent Systems and Computing series. Springer. DOI: 10.1007/978-3-030-72657-7\_16

**Appendix**

Qualitative Problem Code

## Load libraries

library(ISLR2)  
library(MASS)

##   
## Attaching package: 'MASS'

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

library(e1071)

## Warning: package 'e1071' was built under R version 4.3.2

library(class)  
library(tree)

## Warning: package 'tree' was built under R version 4.3.2

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.2

## randomForest 4.7-1.1

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

library(gbm)

## Warning: package 'gbm' was built under R version 4.3.2

## Loaded gbm 2.1.8.1

## Prepare data set

full.data <- read.table("student\_data.csv", sep=";", header = T)  
View(full.data)  
  
student.data <- na.omit(full.data)  
student.data$Target[student.data$Target=="Enrolled"] <- "Student"  
student.data$Target[student.data$Target=="Graduate"] <- "Student"  
  
length(student.data$Target[student.data$Target == "Dropout"]) # 1421 dropout records

## [1] 1421

student.data$Target <- as.factor(student.data$Target)

## Split data set into train and test data

set.seed(1)  
train <- sample(1:nrow(student.data), 0.5 \* nrow(student.data))  
test <- (-train)

# Create data frame to track test errors of each model  
model.errors <- data.frame(model = c("Log Regression", "LDA", "QDA", "NB Classifier",  
 "KNN", "Tree", "Pruned Tree", "Bagging", "RF",  
 "Boosting"),  
 test.error = rep(NA, 10))

## Logistic Regression with All Predictors

log.student <- glm(Target ~ ., data = student.data, family = binomial)  
  
summary(log.student)

##   
## Call:  
## glm(formula = Target ~ ., family = binomial, data = student.data)  
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) 4.513e-01 7.488e-01 0.603  
## Marital.status 1.267e-01 9.782e-02 1.295  
## Application.mode -6.871e-04 3.713e-03 -0.185  
## Application.order -8.428e-02 4.302e-02 -1.959  
## Course -9.950e-05 3.603e-05 -2.762  
## Daytime.evening.attendance -3.113e-02 1.797e-01 -0.173  
## Previous.qualification 1.107e-02 5.395e-03 2.052  
## Previous.qualification.grade -3.115e-03 4.590e-03 -0.679  
## Nacionality -3.550e-02 1.069e-02 -3.321  
## Mother.qualification -1.117e-02 3.990e-03 -2.800  
## Father.qualification 4.386e-03 3.909e-03 1.122  
## Mother.occupation 1.237e-02 4.854e-03 2.548  
## Father.occupation -2.069e-03 5.058e-03 -0.409  
## Admission.grade 4.493e-03 4.271e-03 1.052  
## Displaced -3.412e-01 1.161e-01 -2.938  
## Educational.special.needs -2.619e-01 4.193e-01 -0.624  
## Debtor -4.585e-01 1.663e-01 -2.756  
## Tuition.fees.up.to.date 2.417e+00 1.815e-01 13.319  
## Gender -2.905e-01 1.059e-01 -2.743  
## Scholarship.holder 5.687e-01 1.386e-01 4.104  
## Age.at.enrollment -4.785e-02 9.435e-03 -5.071  
## International 1.975e+00 5.752e-01 3.434  
## Curricular.units.1st.sem.credited -1.429e-01 7.894e-02 -1.810  
## Curricular.units.1st.sem.enrolled 1.649e-02 1.015e-01 0.162  
## Curricular.units.1st.sem.evaluations 7.084e-03 2.398e-02 0.295  
## Curricular.units.1st.sem.approved 2.964e-01 5.169e-02 5.735  
## Curricular.units.1st.sem.grade -5.742e-02 2.372e-02 -2.421  
## Curricular.units.1st.sem.without.evaluations 1.365e-01 9.474e-02 1.441  
## Curricular.units.2nd.sem.credited -2.274e-01 8.441e-02 -2.694  
## Curricular.units.2nd.sem.enrolled -4.842e-01 9.774e-02 -4.954  
## Curricular.units.2nd.sem.evaluations 3.862e-02 2.272e-02 1.700  
## Curricular.units.2nd.sem.approved 6.136e-01 4.765e-02 12.877  
## Curricular.units.2nd.sem.grade 6.915e-02 2.236e-02 3.093  
## Curricular.units.2nd.sem.without.evaluations 1.120e-01 7.833e-02 1.430  
## Unemployment.rate -8.048e-02 2.099e-02 -3.834  
## Inflation.rate -2.374e-02 3.614e-02 -0.657  
## GDP -8.831e-03 2.501e-02 -0.353  
## Pr(>|z|)   
## (Intercept) 0.546677   
## Marital.status 0.195344   
## Application.mode 0.853204   
## Application.order 0.050125 .   
## Course 0.005749 \*\*   
## Daytime.evening.attendance 0.862479   
## Previous.qualification 0.040126 \*   
## Previous.qualification.grade 0.497404   
## Nacionality 0.000898 \*\*\*  
## Mother.qualification 0.005117 \*\*   
## Father.qualification 0.261874   
## Mother.occupation 0.010839 \*   
## Father.occupation 0.682497   
## Admission.grade 0.292814   
## Displaced 0.003304 \*\*   
## Educational.special.needs 0.532325   
## Debtor 0.005848 \*\*   
## Tuition.fees.up.to.date < 2e-16 \*\*\*  
## Gender 0.006091 \*\*   
## Scholarship.holder 4.07e-05 \*\*\*  
## Age.at.enrollment 3.95e-07 \*\*\*  
## International 0.000594 \*\*\*  
## Curricular.units.1st.sem.credited 0.070322 .   
## Curricular.units.1st.sem.enrolled 0.870944   
## Curricular.units.1st.sem.evaluations 0.767641   
## Curricular.units.1st.sem.approved 9.75e-09 \*\*\*  
## Curricular.units.1st.sem.grade 0.015479 \*   
## Curricular.units.1st.sem.without.evaluations 0.149581   
## Curricular.units.2nd.sem.credited 0.007053 \*\*   
## Curricular.units.2nd.sem.enrolled 7.27e-07 \*\*\*  
## Curricular.units.2nd.sem.evaluations 0.089180 .   
## Curricular.units.2nd.sem.approved < 2e-16 \*\*\*  
## Curricular.units.2nd.sem.grade 0.001985 \*\*   
## Curricular.units.2nd.sem.without.evaluations 0.152778   
## Unemployment.rate 0.000126 \*\*\*  
## Inflation.rate 0.511307   
## GDP 0.724043   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 5554.5 on 4423 degrees of freedom  
## Residual deviance: 2739.7 on 4387 degrees of freedom  
## AIC: 2813.7  
##   
## Number of Fisher Scoring iterations: 6

sig\_predictors <- c("Application.order", "Course", "Previous.qualification",  
 "Nacionality", "Mother.qualification", "Mother.occupation",  
 "Displaced", "Debtor", "Tuition.fees.up.to.date",  
 "Gender", "Scholarship.holder", "Age.at.enrollment", "International",  
 "Curricular.units.1st.sem.approved", "Curricular.units.1st.sem.grade",  
 "Curricular.units.2nd.sem.credited", "Curricular.units.2nd.sem.enrolled",  
 "Curricular.units.2nd.sem.approved", "Curricular.units.2nd.sem.grade",  
 "Unemployment.rate", "Target")  
  
student.data <- student.data[, sig\_predictors]

## Logistic Regression

glm.student <- glm(Target ~ ., data = student.data, subset = train,  
 family = binomial)  
  
summary(glm.student)

##   
## Call:  
## glm(formula = Target ~ ., family = binomial, data = student.data,   
## subset = train)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 6.547e-01 5.571e-01 1.175 0.239881   
## Application.order -9.465e-02 6.105e-02 -1.550 0.121039   
## Course -1.164e-04 4.725e-05 -2.463 0.013775 \*   
## Previous.qualification 9.697e-03 7.080e-03 1.370 0.170816   
## Nacionality -4.010e-02 1.172e-02 -3.421 0.000625 \*\*\*  
## Mother.qualification -4.052e-03 4.843e-03 -0.837 0.402774   
## Mother.occupation 1.013e-02 3.149e-03 3.217 0.001297 \*\*   
## Displaced -4.217e-01 1.600e-01 -2.636 0.008382 \*\*   
## Debtor -6.741e-01 2.310e-01 -2.918 0.003519 \*\*   
## Tuition.fees.up.to.date 2.400e+00 2.608e-01 9.205 < 2e-16 \*\*\*  
## Gender -4.124e-01 1.461e-01 -2.823 0.004763 \*\*   
## Scholarship.holder 4.595e-01 1.901e-01 2.418 0.015622 \*   
## Age.at.enrollment -4.895e-02 1.088e-02 -4.497 6.88e-06 \*\*\*  
## International 1.970e+00 6.608e-01 2.981 0.002870 \*\*   
## Curricular.units.1st.sem.approved 2.546e-01 5.879e-02 4.330 1.49e-05 \*\*\*  
## Curricular.units.1st.sem.grade -4.446e-02 3.029e-02 -1.468 0.142157   
## Curricular.units.2nd.sem.credited -3.589e-01 6.538e-02 -5.490 4.01e-08 \*\*\*  
## Curricular.units.2nd.sem.enrolled -3.526e-01 6.793e-02 -5.191 2.09e-07 \*\*\*  
## Curricular.units.2nd.sem.approved 5.850e-01 5.829e-02 10.035 < 2e-16 \*\*\*  
## Curricular.units.2nd.sem.grade 6.859e-02 2.893e-02 2.371 0.017748 \*   
## Unemployment.rate -5.821e-02 2.681e-02 -2.172 0.029891 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2770.5 on 2211 degrees of freedom  
## Residual deviance: 1387.6 on 2191 degrees of freedom  
## AIC: 1429.6  
##   
## Number of Fisher Scoring iterations: 6

glm.probs <- predict(glm.student, student.data[test, ], type = "response")  
glm.pred <- rep("Student", nrow(student.data[test, ]))  
  
contrasts(student.data$Target)

## Student  
## Dropout 0  
## Student 1

glm.pred[glm.probs < 0.5] <- "Dropout"  
table(glm.pred, student.data$Target[test])

##   
## glm.pred Dropout Student  
## Dropout 516 75  
## Student 199 1422

glm.error <- mean(glm.pred != student.data$Target[test])  
glm.error

## [1] 0.1238698

model.errors[model.errors$model == "Log Regression", "test.error"] <- glm.error

## Linear Discriminant Analysis

lda.student <- lda(Target ~ ., data = student.data, subset = train)  
lda.student

## Call:  
## lda(Target ~ ., data = student.data, subset = train)  
##   
## Prior probabilities of groups:  
## Dropout Student   
## 0.3191682 0.6808318   
##   
## Group means:  
## Application.order Course Previous.qualification Nacionality  
## Dropout 1.533994 8756.80 5.432011 2.260623  
## Student 1.788181 8930.02 3.899070 1.899734  
## Mother.qualification Mother.occupation Displaced Debtor  
## Dropout 20.63314 10.30312 0.4645892 0.23229462  
## Student 18.48938 11.63878 0.5989376 0.05909695  
## Tuition.fees.up.to.date Gender Scholarship.holder Age.at.enrollment  
## Dropout 0.6756374 0.5113314 0.1062323 25.97592  
## Student 0.9760956 0.2795485 0.3266932 21.78486  
## International Curricular.units.1st.sem.approved  
## Dropout 0.02974504 2.594901  
## Student 0.02589641 5.747012  
## Curricular.units.1st.sem.grade Curricular.units.2nd.sem.credited  
## Dropout 7.406225 0.4518414  
## Student 12.192280 0.5624170  
## Curricular.units.2nd.sem.enrolled Curricular.units.2nd.sem.approved  
## Dropout 5.752125 1.947592  
## Student 6.474768 5.615538  
## Curricular.units.2nd.sem.grade Unemployment.rate  
## Dropout 5.935216 11.60949  
## Student 12.231881 11.55252  
##   
## Coefficients of linear discriminants:  
## LD1  
## Application.order -3.086898e-02  
## Course -3.918505e-05  
## Previous.qualification 4.689505e-03  
## Nacionality -1.775997e-02  
## Mother.qualification -1.387344e-03  
## Mother.occupation 3.678917e-03  
## Displaced -1.233246e-01  
## Debtor -3.779885e-01  
## Tuition.fees.up.to.date 1.188291e+00  
## Gender -1.833625e-01  
## Scholarship.holder 1.608477e-01  
## Age.at.enrollment -2.021151e-02  
## International 8.167896e-01  
## Curricular.units.1st.sem.approved 1.341369e-01  
## Curricular.units.1st.sem.grade -2.361303e-02  
## Curricular.units.2nd.sem.credited -1.787996e-01  
## Curricular.units.2nd.sem.enrolled -2.784260e-01  
## Curricular.units.2nd.sem.approved 3.746244e-01  
## Curricular.units.2nd.sem.grade 4.295753e-02  
## Unemployment.rate -1.789627e-02

lda.pred <- predict(lda.student, student.data[test, ])  
lda.class <- lda.pred$class  
table(lda.class, student.data$Target[test])

##   
## lda.class Dropout Student  
## Dropout 499 60  
## Student 216 1437

lda.error <- (216 + 60) / (499 + 60 + 216 + 1437)  
lda.error

## [1] 0.124774

model.errors[model.errors$model == "LDA", "test.error"] <- lda.error

## Quadratic Discriminant Analysis

qda.student <- qda(Target ~ ., data = student.data, subset = train)  
qda.student

## Call:  
## qda(Target ~ ., data = student.data, subset = train)  
##   
## Prior probabilities of groups:  
## Dropout Student   
## 0.3191682 0.6808318   
##   
## Group means:  
## Application.order Course Previous.qualification Nacionality  
## Dropout 1.533994 8756.80 5.432011 2.260623  
## Student 1.788181 8930.02 3.899070 1.899734  
## Mother.qualification Mother.occupation Displaced Debtor  
## Dropout 20.63314 10.30312 0.4645892 0.23229462  
## Student 18.48938 11.63878 0.5989376 0.05909695  
## Tuition.fees.up.to.date Gender Scholarship.holder Age.at.enrollment  
## Dropout 0.6756374 0.5113314 0.1062323 25.97592  
## Student 0.9760956 0.2795485 0.3266932 21.78486  
## International Curricular.units.1st.sem.approved  
## Dropout 0.02974504 2.594901  
## Student 0.02589641 5.747012  
## Curricular.units.1st.sem.grade Curricular.units.2nd.sem.credited  
## Dropout 7.406225 0.4518414  
## Student 12.192280 0.5624170  
## Curricular.units.2nd.sem.enrolled Curricular.units.2nd.sem.approved  
## Dropout 5.752125 1.947592  
## Student 6.474768 5.615538  
## Curricular.units.2nd.sem.grade Unemployment.rate  
## Dropout 5.935216 11.60949  
## Student 12.231881 11.55252

qda.class <- predict(qda.student, student.data[test, ])$class  
table(qda.class, student.data$Target[test])

##   
## qda.class Dropout Student  
## Dropout 508 132  
## Student 207 1365

qda.error <- (207 + 132) / (508 + 132 + 207 + 1365)  
qda.error

## [1] 0.153255

model.errors[model.errors$model == "QDA", "test.error"] <- qda.error

## Naive Bayes Classifier

nb.student <- naiveBayes(Target ~ ., data = student.data, subset = train)  
nb.student

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## Dropout Student   
## 0.3191682 0.6808318   
##   
## Conditional probabilities:  
## Application.order  
## Y [,1] [,2]  
## Dropout 1.533994 1.134678  
## Student 1.788181 1.331806  
##   
## Course  
## Y [,1] [,2]  
## Dropout 8756.80 2261.171  
## Student 8930.02 1915.336  
##   
## Previous.qualification  
## Y [,1] [,2]  
## Dropout 5.432011 10.436790  
## Student 3.899070 9.663187  
##   
## Nacionality  
## Y [,1] [,2]  
## Dropout 2.260623 9.032176  
## Student 1.899734 6.968837  
##   
## Mother.qualification  
## Y [,1] [,2]  
## Dropout 20.63314 15.64445  
## Student 18.48938 15.63806  
##   
## Mother.occupation  
## Y [,1] [,2]  
## Dropout 10.30312 20.82031  
## Student 11.63878 29.21718  
##   
## Displaced  
## Y [,1] [,2]  
## Dropout 0.4645892 0.4990981  
## Student 0.5989376 0.4902764  
##   
## Debtor  
## Y [,1] [,2]  
## Dropout 0.23229462 0.4225953  
## Student 0.05909695 0.2358844  
##   
## Tuition.fees.up.to.date  
## Y [,1] [,2]  
## Dropout 0.6756374 0.4684681  
## Student 0.9760956 0.1528021  
##   
## Gender  
## Y [,1] [,2]  
## Dropout 0.5113314 0.5002260  
## Student 0.2795485 0.4489264  
##   
## Scholarship.holder  
## Y [,1] [,2]  
## Dropout 0.1062323 0.3083532  
## Student 0.3266932 0.4691598  
##   
## Age.at.enrollment  
## Y [,1] [,2]  
## Dropout 25.97592 8.437555  
## Student 21.78486 6.425049  
##   
## International  
## Y [,1] [,2]  
## Dropout 0.02974504 0.1700036  
## Student 0.02589641 0.1588790  
##   
## Curricular.units.1st.sem.approved  
## Y [,1] [,2]  
## Dropout 2.594901 2.884082  
## Student 5.747012 2.649718  
##   
## Curricular.units.1st.sem.grade  
## Y [,1] [,2]  
## Dropout 7.406225 5.991239  
## Student 12.192280 3.114367  
##   
## Curricular.units.2nd.sem.credited  
## Y [,1] [,2]  
## Dropout 0.4518414 1.721315  
## Student 0.5624170 1.998443  
##   
## Curricular.units.2nd.sem.enrolled  
## Y [,1] [,2]  
## Dropout 5.752125 2.090038  
## Student 6.474768 2.195365  
##   
## Curricular.units.2nd.sem.approved  
## Y [,1] [,2]  
## Dropout 1.947592 2.575680  
## Student 5.615538 2.430208  
##   
## Curricular.units.2nd.sem.grade  
## Y [,1] [,2]  
## Dropout 5.935216 6.105464  
## Student 12.231881 3.112023  
##   
## Unemployment.rate  
## Y [,1] [,2]  
## Dropout 11.60949 2.766519  
## Student 11.55252 2.638518

nb.class <- predict(nb.student, student.data[test, ])  
table(nb.class, student.data$Target[test])

##   
## nb.class Dropout Student  
## Dropout 514 159  
## Student 201 1338

nb.error <- (201 + 159) / (514 + 159 + 201 + 1338)  
nb.error

## [1] 0.1627486

model.errors[model.errors$model == "NB Classifier", "test.error"] <- nb.error

## K-Nearest Neighbor

length(sig\_predictors)

## [1] 21

train.X <- as.matrix(student.data[train, sig\_predictors[-21]])  
test.X <- as.matrix(student.data[test, sig\_predictors[-21]])  
train.target <- student.data$Target[train]  
  
k.values <- c(1, 3, 5, 10)  
knn.errors <- data.frame(k.value = k.values,  
 pred.error = rep(NA, length(k.values)))  
for (x in 1:length(k.values)) {  
 set.seed(2)  
 knn.pred <- knn(train.X, test.X, train.target, k = k.values[x])  
 knn.errors[x, "pred.error"] <- mean(knn.pred != student.data$Target[test])  
}  
  
knn.errors # k = 5 lowest test error

## k.value pred.error  
## 1 1 0.2373418  
## 2 3 0.2255877  
## 3 5 0.2124774  
## 4 10 0.2188065

model.errors[model.errors$model == "KNN", "test.error"] <- min(knn.errors$pred.error)

## Classification tree

# Classification Tree Model  
  
tree.student <- tree(Target ~ ., data = student.data, subset = train)  
summary(tree.student)

##   
## Classification tree:  
## tree(formula = Target ~ ., data = student.data, subset = train)  
## Variables actually used in tree construction:  
## [1] "Curricular.units.2nd.sem.approved" "Tuition.fees.up.to.date"   
## [3] "Age.at.enrollment" "Curricular.units.2nd.sem.enrolled"  
## [5] "Curricular.units.2nd.sem.grade"   
## Number of terminal nodes: 8   
## Residual mean deviance: 0.6947 = 1531 / 2204   
## Misclassification error rate: 0.1293 = 286 / 2212

plot(tree.student)  
text(tree.student, pretty = 0)

A black and white text

Description automatically generated

tree.pred <- predict(tree.student, student.data[test, ], type = "class")  
table(tree.pred, student.data$Target[test])

##   
## tree.pred Dropout Student  
## Dropout 497 86  
## Student 218 1411

tree.error <- mean(tree.pred != student.data$Target[test])  
tree.error

## [1] 0.1374322

model.errors[model.errors$model == "Tree", "test.error"] <- tree.error

## Pruned Tree

set.seed(4)  
cv.student <- cv.tree(tree.student, FUN = prune.misclass)  
cv.student

## $size  
## [1] 8 7 3 2 1  
##   
## $dev  
## [1] 288 288 392 395 706  
##   
## $k  
## [1] -Inf 0.00 20.25 23.00 316.00  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(cv.student$size, cv.student$dev, type = "b")

A graph of a number and a line

Description automatically generated

pruned.student <- prune.misclass(tree.student, best = 7)  
plot(pruned.student)  
text(pruned.student, pretty = 0)

A black and white text on a white background

Description automatically generated

summary(pruned.student)

##   
## Classification tree:  
## snip.tree(tree = tree.student, nodes = 7L)  
## Variables actually used in tree construction:  
## [1] "Curricular.units.2nd.sem.approved" "Tuition.fees.up.to.date"   
## [3] "Age.at.enrollment" "Curricular.units.2nd.sem.enrolled"  
## Number of terminal nodes: 7   
## Residual mean deviance: 0.7098 = 1565 / 2205   
## Misclassification error rate: 0.1293 = 286 / 2212

pruned.pred <- predict(pruned.student, student.data[test, ], type = "class")  
table(pruned.pred, student.data$Target[test])

##   
## pruned.pred Dropout Student  
## Dropout 497 86  
## Student 218 1411

prune.error <- mean(pruned.pred != student.data$Target[test])  
prune.error

## [1] 0.1374322

model.errors[model.errors$model == "Pruned Tree", "test.error"] <- prune.error

## Bagging

set.seed(5)  
bag.student <- randomForest(Target ~ ., data = student.data, subset = train,  
 mtry = length(sig\_predictors)-1, importance = T)  
bag.student

##   
## Call:  
## randomForest(formula = Target ~ ., data = student.data, mtry = length(sig\_predictors) - 1, importance = T, subset = train)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 20  
##   
## OOB estimate of error rate: 13.52%  
## Confusion matrix:  
## Dropout Student class.error  
## Dropout 505 201 0.28470255  
## Student 98 1408 0.06507304

bag.pred <- predict(bag.student, student.data[test, ], type = "class")  
table(bag.pred, student.data$Target[test])

##   
## bag.pred Dropout Student  
## Dropout 512 95  
## Student 203 1402

bag.error <- mean(bag.pred != student.data$Target[test])  
bag.error

## [1] 0.1347197

model.errors[model.errors$model == "Bagging", "test.error"] <- bag.error

## Random Forest

# Random Forest using default sqrt(p) predictors  
  
set.seed(5)  
rf.student <- randomForest(Target ~ ., data = student.data, subset = train,  
 importance = T)  
rf.student

##   
## Call:  
## randomForest(formula = Target ~ ., data = student.data, importance = T, subset = train)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 12.79%  
## Confusion matrix:  
## Dropout Student class.error  
## Dropout 511 195 0.27620397  
## Student 88 1418 0.05843293

varImpPlot(rf.student)

A close-up of a diagram

Description automatically generated

rf.pred <- predict(rf.student, newdata = student.data[test, ], type = "class")  
rf.error <- mean(rf.pred != student.data$Target[test])  
rf.error

## [1] 0.1338156

model.errors[model.errors$model == "RF", "test.error"] <- rf.error

## Boosting

tunings <- c(0.001, 0.01, 0.2, 0.5, 0.75, 1.0)  
boost.results <- data.frame(shrinkage = tunings,  
 test.error = rep(NA, length(tunings)))  
student.data$Target <- as.numeric(student.data$Target)  
student.data$Target <- apply(student.data, 1, FUN = function(x) x[1] - 1)  
  
for (x in 1:length(tunings)) {  
 set.seed(6)  
 boost.student <- gbm(Target ~ ., data = student.data[train, ],  
 distribution = "bernoulli", n.trees = 1000,  
 interaction.depth = 7, shrinkage = tunings[x])  
 boost.pred <- predict(boost.student, newdata = student.data[test, ],   
 type = "response", n.trees = 1000)  
 boost.results[x, "test.error"] <- mean(boost.pred != student.data$Target[test])  
}  
  
boost.results

## shrinkage test.error  
## 1 0.001 1.0000000  
## 2 0.010 1.0000000  
## 3 0.200 0.8792948  
## 4 0.500 1.0000000  
## 5 0.750 0.1957505  
## 6 1.000 0.1957505

model.errors[model.errors$model == "Boosting", "test.error"] <- min(boost.results$test.error)

## Comparison of Models

model.errors

## model test.error  
## 1 Log Regression 0.1238698  
## 2 LDA 0.1247740  
## 3 QDA 0.1532550  
## 4 NB Classifier 0.1627486  
## 5 KNN 0.2124774  
## 6 Tree 0.1374322  
## 7 Pruned Tree 0.1374322  
## 8 Bagging 0.1347197  
## 9 RF 0.1338156  
## 10 Boosting 0.1957505

## Logistic Regression Using Even Less Predictors

# Reinitialize student data after boosting changes  
student.data <- na.omit(full.data)  
student.data$Target[student.data$Target=="Enrolled"] <- "Student"  
student.data$Target[student.data$Target=="Graduate"] <- "Student"  
student.data$Target <- as.factor(student.data$Target)  
student.data <- student.data[, sig\_predictors]  
  
log.final <- glm(Target ~ Curricular.units.2nd.sem.approved + Tuition.fees.up.to.date  
 + Age.at.enrollment + Curricular.units.2nd.sem.enrolled, data = student.data,  
 subset = train, family = binomial)  
summary(log.final)

##   
## Call:  
## glm(formula = Target ~ Curricular.units.2nd.sem.approved + Tuition.fees.up.to.date +   
## Age.at.enrollment + Curricular.units.2nd.sem.enrolled, family = binomial,   
## data = student.data, subset = train)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.374988 0.332232 -1.129 0.259   
## Curricular.units.2nd.sem.approved 0.782657 0.036030 21.723 < 2e-16 \*\*\*  
## Tuition.fees.up.to.date 2.493032 0.224053 11.127 < 2e-16 \*\*\*  
## Age.at.enrollment -0.053731 0.008827 -6.087 1.15e-09 \*\*\*  
## Curricular.units.2nd.sem.enrolled -0.479326 0.041541 -11.539 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2770.5 on 2211 degrees of freedom  
## Residual deviance: 1499.4 on 2207 degrees of freedom  
## AIC: 1509.4  
##   
## Number of Fisher Scoring iterations: 5

log.probs <- predict(log.final, student.data[test, ], type = "response")  
log.pred <- rep("Student", nrow(student.data[test, ]))  
  
contrasts(student.data$Target)

## Student  
## Dropout 0  
## Student 1

log.pred[glm.probs < 0.5] <- "Dropout"  
table(log.pred, student.data$Target[test])

##   
## log.pred Dropout Student  
## Dropout 516 75  
## Student 199 1422

log.error <- mean(log.pred != student.data$Target[test])  
log.error

## [1] 0.1238698

Quantitative Problem Code

## Load Libraries

library(ISLR2)  
library(MASS)

##   
## Attaching package: 'MASS'

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

library(leaps)

## Warning: package 'leaps' was built under R version 4.3.2

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.3.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(pls)

## Warning: package 'pls' was built under R version 4.3.2

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

library(boot)  
library(tree)

## Warning: package 'tree' was built under R version 4.3.2

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.2

## randomForest 4.7-1.1

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

library(gbm)

## Warning: package 'gbm' was built under R version 4.3.2

## Loaded gbm 2.1.8.1

library(BART)

## Warning: package 'BART' was built under R version 4.3.2

## Loading required package: nlme

## Loading required package: nnet

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:boot':  
##   
## aml

## Load wine data sets

# Import wine quality data  
wine.data <- read.csv("winequality-white.csv", sep=";", na.strings = "?", stringsAsFactors = T)  
View(wine.data)

white\_cor <- cor(wine.data, use="complete.obs")  
print(white\_cor[12, ])

## fixed.acidity volatile.acidity citric.acid   
## -0.113662831 -0.194722969 -0.009209091   
## residual.sugar chlorides free.sulfur.dioxide   
## -0.097576829 -0.209934411 0.008158067   
## total.sulfur.dioxide density pH   
## -0.174737218 -0.307123313 0.099427246   
## sulphates alcohol quality   
## 0.053677877 0.435574715 1.000000000

## Track test errors across models

model.errors <- data.frame(model = c("Least Squares", "Ridge Regression", "The LASSO", "PLS",  
 "Best Subset", "Forward Stepwise", "Backward Stepwise", "Tree",   
 "Pruned Tree", "Bagging", "RF", "Boosting"),  
 test.error = rep(NA, 12))

### Multiple Linear Regression

lm.wine <- lm(quality ~ ., data = wine.data)  
summary(lm.wine)

##   
## Call:  
## lm(formula = quality ~ ., data = wine.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.8348 -0.4934 -0.0379 0.4637 3.1143   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.502e+02 1.880e+01 7.987 1.71e-15 \*\*\*  
## fixed.acidity 6.552e-02 2.087e-02 3.139 0.00171 \*\*   
## volatile.acidity -1.863e+00 1.138e-01 -16.373 < 2e-16 \*\*\*  
## citric.acid 2.209e-02 9.577e-02 0.231 0.81759   
## residual.sugar 8.148e-02 7.527e-03 10.825 < 2e-16 \*\*\*  
## chlorides -2.473e-01 5.465e-01 -0.452 0.65097   
## free.sulfur.dioxide 3.733e-03 8.441e-04 4.422 9.99e-06 \*\*\*  
## total.sulfur.dioxide -2.857e-04 3.781e-04 -0.756 0.44979   
## density -1.503e+02 1.907e+01 -7.879 4.04e-15 \*\*\*  
## pH 6.863e-01 1.054e-01 6.513 8.10e-11 \*\*\*  
## sulphates 6.315e-01 1.004e-01 6.291 3.44e-10 \*\*\*  
## alcohol 1.935e-01 2.422e-02 7.988 1.70e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7514 on 4886 degrees of freedom  
## Multiple R-squared: 0.2819, Adjusted R-squared: 0.2803   
## F-statistic: 174.3 on 11 and 4886 DF, p-value: < 2.2e-16

# Create training and test data  
set.seed(10)  
train.wine <- sample(1:nrow(wine.data), 0.5 \* nrow(wine.data))  
test.wine <- (-train.wine)

# Train a linear model with statistically significant predictors  
lm.train <- lm(quality ~ volatile.acidity + residual.sugar + free.sulfur.dioxide  
 + density + pH + sulphates + alcohol, data = wine.data,  
 subset = train.wine)  
summary(lm.train)

##   
## Call:  
## lm(formula = quality ~ volatile.acidity + residual.sugar + free.sulfur.dioxide +   
## density + pH + sulphates + alcohol, data = wine.data, subset = train.wine)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.6963 -0.5029 -0.0299 0.4676 2.7588   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.145e+02 2.001e+01 5.721 1.19e-08 \*\*\*  
## volatile.acidity -1.824e+00 1.532e-01 -11.912 < 2e-16 \*\*\*  
## residual.sugar 6.649e-02 8.020e-03 8.291 < 2e-16 \*\*\*  
## free.sulfur.dioxide 2.928e-03 9.647e-04 3.035 0.002432 \*\*   
## density -1.133e+02 2.002e+01 -5.660 1.69e-08 \*\*\*  
## pH 3.764e-01 1.071e-01 3.515 0.000448 \*\*\*  
## sulphates 6.084e-01 1.384e-01 4.397 1.14e-05 \*\*\*  
## alcohol 2.389e-01 2.897e-02 8.245 2.66e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7496 on 2441 degrees of freedom  
## Multiple R-squared: 0.2748, Adjusted R-squared: 0.2727   
## F-statistic: 132.1 on 7 and 2441 DF, p-value: < 2.2e-16

# Use trained linear model to predict the wine quality  
lm.predict <- predict(lm.train, wine.data[test.wine, ])  
lm.mse <- mean((lm.predict - wine.data$quality[test.wine])^2)  
lm.mse

## [1] 0.5698636

model.errors[model.errors$model == "Least Squares", "test.error"] <- lm.mse

# Train the first linear model  
first.lm.wine <- lm(quality ~ ., data = wine.data, subset = train.wine)  
summary(first.lm.wine)

##   
## Call:  
## lm(formula = quality ~ ., data = wine.data, subset = train.wine)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.7823 -0.4940 -0.0422 0.4655 2.7949   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.971e+02 3.262e+01 6.040 1.77e-09 \*\*\*  
## fixed.acidity 1.173e-01 3.218e-02 3.646 0.000272 \*\*\*  
## volatile.acidity -1.741e+00 1.608e-01 -10.827 < 2e-16 \*\*\*  
## citric.acid -9.765e-02 1.340e-01 -0.729 0.466327   
## residual.sugar 9.651e-02 1.233e-02 7.828 7.31e-15 \*\*\*  
## chlorides -6.428e-01 8.578e-01 -0.749 0.453708   
## free.sulfur.dioxide 3.384e-03 1.200e-03 2.820 0.004842 \*\*   
## total.sulfur.dioxide -1.155e-04 5.495e-04 -0.210 0.833529   
## density -1.975e+02 3.305e+01 -5.977 2.61e-09 \*\*\*  
## pH 7.697e-01 1.581e-01 4.868 1.20e-06 \*\*\*  
## sulphates 7.251e-01 1.419e-01 5.109 3.48e-07 \*\*\*  
## alcohol 1.330e-01 4.123e-02 3.225 0.001276 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7478 on 2437 degrees of freedom  
## Multiple R-squared: 0.2795, Adjusted R-squared: 0.2762   
## F-statistic: 85.93 on 11 and 2437 DF, p-value: < 2.2e-16

# Use first linear model to predict the wine quality  
first.lm.predict <- predict(first.lm.wine, wine.data[test.wine, ])  
first.lm.mse <- mean((first.lm.predict - wine.data$quality[test.wine])^2)  
first.lm.mse

## [1] 0.5731913

### Ridge Regression

# Create matrix of x, the predictors, and vector of y, the response  
x <- model.matrix(quality ~ ., wine.data)[, -1]  
y <- wine.data$quality  
  
y.test <- y[test.wine]

# Create a lambda grid and use it to form ridge regression model  
lambda.grid <- 10^seq(10, -2, length = 100)  
ridge.mod <- glmnet(x[train.wine, ], y[train.wine], alpha = 0, lambda = lambda.grid,  
 thresh = 1e-12)  
summary(ridge.mod)

## Length Class Mode   
## a0 100 -none- numeric  
## beta 1100 dgCMatrix S4   
## df 100 -none- numeric  
## dim 2 -none- numeric  
## lambda 100 -none- numeric  
## dev.ratio 100 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 6 -none- call   
## nobs 1 -none- numeric

# Determine the best lambda, or tuning parameter, using cross-validation  
set.seed(2)  
cv.out <- cv.glmnet(x[train.wine, ], y[train.wine], alpha = 0)  
plot(cv.out)

A graph of a log

Description automatically generated

bestlam.ridge <- cv.out$lambda.min  
bestlam.ridge

## [1] 0.0384732

# Predict the response of test data using ridge regression with best tuning parameter  
ridge.pred <- predict(ridge.mod, s = bestlam.ridge, newx = x[test.wine, ])  
ridge.mse <- mean((ridge.pred - y.test)^2)  
ridge.mse

## [1] 0.5737655

model.errors[model.errors$model == "Ridge Regression", "test.error"] <- ridge.mse

### The Lasso

lasso.mod <- glmnet(x[train.wine, ], y[train.wine], alpha = 1, lambda = lambda.grid)  
plot(lasso.mod)

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values

A graph with numbers and lines

Description automatically generated

# Perform cross-validation to determine best tuning parameter  
set.seed(2)  
cv.out <- cv.glmnet(x[train.wine, ], y[train.wine], alpha = 1)  
plot(cv.out)

A graph of a function

Description automatically generated

bestlam.lasso <- cv.out$lambda.min  
bestlam.lasso

## [1] 0.0005713149

# Predict the response of test data and calculate MSE  
lasso.pred <- predict(lasso.mod, s = bestlam.lasso, newx = x[test.wine, ])  
lasso.mse <- mean((lasso.pred - y.test)^2)  
lasso.mse

## [1] 0.575157

model.errors[model.errors$model == "The LASSO", "test.error"] <- lasso.mse

The Lasso model is used to predict the quality of white wine using the best tuning parameter and the test data. The mean squared error for the Lasso model is 0.575, which is worse than the multiple linear regression and the ridge regression models.

lasso.coef <- predict(lasso.mod, type = "coefficients", s = bestlam.lasso)[1:11, ]  
lasso.coef

## (Intercept) fixed.acidity volatile.acidity   
## 4.910463e+01 0.000000e+00 -1.720231e+00   
## citric.acid residual.sugar chlorides   
## -2.552551e-02 3.824724e-02 -1.245298e+00   
## free.sulfur.dioxide total.sulfur.dioxide density   
## 2.961531e-03 -1.019355e-04 -4.727139e+01   
## pH sulphates   
## 2.011136e-01 4.309143e-01

lasso.coef[lasso.coef != 0]

## (Intercept) volatile.acidity citric.acid   
## 4.910463e+01 -1.720231e+00 -2.552551e-02   
## residual.sugar chlorides free.sulfur.dioxide   
## 3.824724e-02 -1.245298e+00 2.961531e-03   
## total.sulfur.dioxide density pH   
## -1.019355e-04 -4.727139e+01 2.011136e-01   
## sulphates   
## 4.309143e-01

### Partial Least Squares Regression

# Create PLS model on the wine wine quality data  
set.seed(2)  
pls.fit <- plsr(quality ~ ., data = wine.data, subset = train.wine, scale = T,  
 validation = "CV")  
summary(pls.fit)

## Data: X dimension: 2449 11   
## Y dimension: 2449 1  
## Fit method: kernelpls  
## Number of components considered: 11  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 0.8792 0.8002 0.7651 0.7549 0.7539 0.7538 0.7533  
## adjCV 0.8792 0.8002 0.7648 0.7546 0.7537 0.7535 0.7530  
## 7 comps 8 comps 9 comps 10 comps 11 comps  
## CV 0.7525 0.7509 0.7508 0.7509 0.7509  
## adjCV 0.7522 0.7507 0.7506 0.7507 0.7507  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 26.89 38.95 45.44 55.72 65.78 72.58 75.07 79.53  
## quality 17.47 25.07 27.06 27.26 27.31 27.39 27.71 27.89  
## 9 comps 10 comps 11 comps  
## X 85.17 93.34 100.00  
## quality 27.95 27.95 27.95

# Plot MSEP over the number of components  
validationplot(pls.fit, val.type = "MSEP")  
axis(side=1, at=seq(1, 20, by=1))

A graph of quality and number of components

Description automatically generated

# Predict quality of the wine using PLS  
pls.pred <- predict(pls.fit, newdata = x[test.wine, ], ncomp=3)  
pls.mse <- mean((pls.pred - y.test)^2)  
pls.mse

## [1] 0.5783051

model.errors[model.errors$model == "PLS", "test.error"] <- pls.mse

## Best Subset Selection

# Function to predict response using best subset selection  
predict.regsubsets <- function(object, newdata, id, ...) {  
 form <- as.formula(object$call[[2]])  
 mat <- model.matrix(form, newdata)  
 coefi <- coef(object, id = id)  
 xvars <- names(coefi)  
 mat[, xvars] %\*% coefi  
}

### Validation Set Approach

# Use validation set approach to determine best subset selection model  
regfit.best <- regsubsets(quality ~ ., data = wine.data[train.wine, ], nvmax = 11)  
  
# Create test matrix  
test.mat <- model.matrix(quality ~ ., data = wine.data[test.wine, ])  
  
# Compute test MSE for all possible amounts of variables used in the model  
val.errors <- rep(NA, 13)  
for (i in 1:11) {  
 coefi <- coef(regfit.best, id = i)  
 pred <- test.mat[, names(coefi)] %\*% coefi  
 val.errors[i] <- mean((wine.data$quality[test.wine] - pred)^2)  
}  
  
# Get coefficient estimates for model with best subset collection  
best.subset <- which.min(val.errors)  
val.errors[best.subset]

## [1] 0.5726076

coef(regfit.best, best.subset)

## (Intercept) fixed.acidity volatile.acidity residual.sugar   
## 2.047419e+02 1.199790e-01 -1.735146e+00 9.955035e-02   
## free.sulfur.dioxide density pH sulphates   
## 3.123752e-03 -2.054633e+02 8.047257e-01 7.251004e-01   
## alcohol   
## 1.290130e-01

### K-fold Cross-validation approach

# Best subset selection using cross-validation method  
  
k <- 10  
n <- nrow(wine.data)  
set.seed(11)  
folds <- sample(rep(1:k, length = n))  
cv\_sub.errors <- matrix(NA, k, 11,  
 dimnames = list(NULL, paste(1:11)))  
  
for (j in 1:k) {  
 cv\_sub.fit <- regsubsets(quality ~ .,  
 data = wine.data[folds != j, ],  
 nvmax = 11)  
 for (i in 1:11) {  
 pred.cv\_sub <- predict.regsubsets(cv\_sub.fit, wine.data[folds == j, ], id = i)  
 cv\_sub.errors[j, i] <- mean((wine.data$quality[folds == j] - pred.cv\_sub)^2)  
 }  
}  
  
cv\_sub.cv.errors <- apply(cv\_sub.errors, 2, mean)  
par(mfrow = c(1,1))  
plot(cv\_sub.cv.errors, type = "b")

A line graph with numbers and a line

Description automatically generated

which.min(cv\_sub.cv.errors)

## 8   
## 8

cv\_sub.mse <- cv\_sub.cv.errors[["8"]]  
cv\_sub.mse

## [1] 0.5683492

model.errors[model.errors$model == "Best Subset", "test.error"] <- cv\_sub.mse

Best subset selection is performed again using a k-fold cross-validation approach this time. With k = 10, the cross-validation also determines that the best subset contains 8 predictors. The test MSE is calculated to be 0.568.

## Forward Stepwise Selection

k <- 10  
n <- nrow(wine.data)  
set.seed(11)  
folds <- sample(rep(1:k, length = n))  
f.cv.errors <- matrix(NA, k, 11,  
 dimnames = list(NULL, paste(1:11)))  
  
for (j in 1:k) {  
 fstep.fit <- regsubsets(quality ~ .,  
 data = wine.data[folds != j, ],  
 nvmax = 11,  
 method = "forward")  
 for (i in 1:11) {  
 pred.forward <- predict.regsubsets(fstep.fit, wine.data[folds == j, ], id = i)  
 f.cv.errors[j, i] <- mean((wine.data$quality[folds == j] - pred.forward)^2)  
 }  
}  
  
forward.cv.errors <- apply(f.cv.errors, 2, mean)  
forward.cv.errors

## 1 2 3 4 5 6 7 8   
## 0.6359855 0.5965686 0.5824032 0.5832526 0.5800027 0.5743192 0.5711655 0.5683492   
## 9 10 11   
## 0.5687762 0.5689857 0.5691309

par(mfrow = c(1,1))  
plot(forward.cv.errors, type = "b")

A line graph with numbers and a line

Description automatically generated

which.min(forward.cv.errors)

## 8   
## 8

forward.mse <- forward.cv.errors[["8"]]  
forward.mse

## [1] 0.5683492

model.errors[model.errors$model == "Forward Stepwise", "test.error"] <- forward.mse

## Backward Stepwise Selection

# Backward Step-wise Subset Selection Using Cross Validation  
  
k <- 10  
n <- nrow(wine.data)  
set.seed(11)  
folds <- sample(rep(1:k, length = n))  
b.cv.errors <- matrix(NA, k, 11,  
 dimnames = list(NULL, paste(1:11)))  
  
for (j in 1:k) {  
 bstep.fit <- regsubsets(quality ~ .,  
 data = wine.data[folds != j, ],  
 nvmax = 11,  
 method = "backward")  
 for (i in 1:11) {  
 pred.backward <- predict.regsubsets(fstep.fit, wine.data[folds == j, ], id = i)  
 b.cv.errors[j, i] <-   
 mean((wine.data$quality[folds == j] - pred.backward)^2)  
 }  
}  
  
backward.cv.errors <- apply(b.cv.errors, 2, mean)  
backward.cv.errors

## 1 2 3 4 5 6 7 8   
## 0.6354239 0.5958132 0.5814734 0.5772214 0.5716449 0.5672763 0.5645731 0.5632897   
## 9 10 11   
## 0.5632278 0.5632074 0.5632012

par(mfrow = c(1,1))  
plot(backward.cv.errors, type = "b")

A line graph with numbers and a line

Description automatically generated

which.min(backward.cv.errors)

## 11   
## 11

backward.mse <- backward.cv.errors[["11"]]  
backward.mse

## [1] 0.5632012

model.errors[model.errors$model == "Backward Stepwise", "test.error"] <- backward.mse

## Regression Tree

# Create regression tree model for quality as response  
tree.wine <- tree(quality ~ ., data = wine.data, subset = train.wine)  
summary(tree.wine)

##   
## Regression tree:  
## tree(formula = quality ~ ., data = wine.data, subset = train.wine)  
## Variables actually used in tree construction:  
## [1] "alcohol" "volatile.acidity" "chlorides"   
## [4] "free.sulfur.dioxide"  
## Number of terminal nodes: 6   
## Residual mean deviance: 0.5665 = 1384 / 2443   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -3.5830 -0.3120 0.1072 0.0000 0.4171 2.6880

# Plot the regression tree  
plot(tree.wine)  
text(tree.wine, pretty = 0)

A diagram of a chemical reaction

Description automatically generated

tree.pred <- predict(tree.wine, newdata = wine.data[test.wine, ])  
tree.mse <- mean((tree.pred - y.test)^2)  
tree.mse

## [1] 0.5930002

model.errors[model.errors$model == "Tree", "test.error"] <- tree.mse

Using the regression tree to predict the quality of white wine, the test MSE is 0.593.

## Pruned Tree

# Use cross-validation to determine best tree size for pruning  
tree.cv <- cv.tree(tree.wine)  
plot(tree.cv$size, tree.cv$dev, type = "b")

A graph of a tree

Description automatically generated

# Prune tree and plot the tree  
prune.wine <- prune.tree(tree.wine, best = 6)  
plot(prune.wine)  
text(prune.wine, pretty = 0)

A diagram of a chemical reaction

Description automatically generated

# Predict response with pruned tree  
prune.pred <- predict(prune.wine, newdata = wine.data[test.wine, ])  
prune.mse <- mean((prune.pred - y.test)^2)  
prune.mse

## [1] 0.5930002

model.errors[model.errors$model == "Pruned Tree", "test.error"] <- prune.mse

## Bagging

set.seed(90)  
bag.wine <- randomForest(quality ~ ., data = wine.data, subset = train.wine,  
 mtry = (ncol(wine.data) - 1), importance = T)  
bag.wine

##   
## Call:  
## randomForest(formula = quality ~ ., data = wine.data, mtry = (ncol(wine.data) - 1), importance = T, subset = train.wine)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 11  
##   
## Mean of squared residuals: 0.4327978  
## % Var explained: 43.96

A bagging model is created to attempt to use a series of bootstrapped regression trees to better predict the quality of white wine. The bagging model considers all predictors as split candidates at each split in the trees. 500 trees are used in the model, the mean of squared residuals is 0.433, and 43.96% of the variance is explained with this model.

bag.pred <- predict(bag.wine, newdata = wine.data[test.wine, ])  
bag.mse <- mean((bag.pred - y.test)^2)  
bag.mse

## [1] 0.4079268

model.errors[model.errors$model == "Bagging", "test.error"] <- bag.mse

## Random Forest

set.seed(90)  
rf.wine <- randomForest(quality ~ ., data = wine.data[train.wine, ],  
 importance = T)  
rf.wine

##   
## Call:  
## randomForest(formula = quality ~ ., data = wine.data[train.wine, ], importance = T)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 0.4229552  
## % Var explained: 45.23

importance(rf.wine)

## %IncMSE IncNodePurity  
## fixed.acidity 26.59123 114.8717  
## volatile.acidity 55.69239 180.5992  
## citric.acid 36.74146 127.3700  
## residual.sugar 34.04928 141.7358  
## chlorides 36.94861 158.7531  
## free.sulfur.dioxide 51.79472 183.1726  
## total.sulfur.dioxide 31.05734 134.0279  
## density 31.05210 200.2182  
## pH 36.88850 127.2465  
## sulphates 25.36442 107.2996  
## alcohol 57.86859 308.8811

varImpPlot(rf.wine)

A chart of alcohol content

Description automatically generated

rf.pred <- predict(rf.wine, newdata = wine.data[test.wine, ])  
rf.mse <- mean((rf.pred - y.test)^2)  
rf.mse

## [1] 0.4066441

model.errors[model.errors$model == "RF", "test.error"] <- rf.mse

## Boosting

tunings <- c(0.001, 0.005, 0.01, 0.02, 0.03)  
boost.errors <- data.frame(shrinkage = tunings,  
 training.error = rep(NA, length(tunings)),  
 test.error = rep(NA, length(tunings)))  
  
for (x in 1:length(tunings)) {  
 set.seed(91)  
 boost.wine <- gbm(quality ~ ., data = wine.data[train.wine, ],  
 distribution = "gaussian", n.trees = 1000,  
 interaction.depth = 4, shrinkage = tunings[x])  
 boost.errors[x, "training.error"] <- mean(boost.wine$train.error)  
}  
  
for (x in 1:length(tunings)) {  
 set.seed(91)  
 boost.wine <- gbm(quality ~ ., data = wine.data[train.wine, ],  
 distribution = "gaussian", n.trees = 1000,  
 interaction.depth = 4, shrinkage = tunings[x])  
 yhat.boost <- predict(boost.wine, newdata = wine.data[test.wine, ],  
 n.trees = 1000)  
 boost.errors[x, "test.error"] <- mean((yhat.boost - y.test)^2)  
}  
  
boost.errors

## shrinkage training.error test.error  
## 1 0.001 0.6506781 0.5982912  
## 2 0.005 0.5243904 0.4939699  
## 3 0.010 0.4735161 0.4782166  
## 4 0.020 0.4225135 0.4671453  
## 5 0.030 0.3887048 0.4672300

model.errors[model.errors$model == "Boosting", "test.error"] <- min(boost.errors$test.error)

model.errors

## model test.error  
## 1 Least Squares 0.5698636  
## 2 Ridge Regression 0.5737655  
## 3 The LASSO 0.5751570  
## 4 PLS 0.5783051  
## 5 Best Subset 0.5683492  
## 6 Forward Stepwise 0.5683492  
## 7 Backward Stepwise 0.5632012  
## 8 Tree 0.5930002  
## 9 Pruned Tree 0.5930002  
## 10 Bagging 0.4079268  
## 11 RF 0.4066441  
## 12 Boosting 0.4671453

sqrt(min(model.errors$test.error))

## [1] 0.6376865

## Load Libraries

library(ISLR2)  
library(MASS)

##   
## Attaching package: 'MASS'

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

library(pls)

## Warning: package 'pls' was built under R version 4.3.2

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

View(Credit)

set.seed(1)  
train <- sample(1:nrow(Credit), nrow(Credit) / 2)  
test <- (-train)

set.seed(2)  
  
pcr.fit <- pcr(Balance ~ ., data = Credit, subset = train, scale = T,  
 validation = "CV")  
summary(pcr.fit)

## Data: X dimension: 200 11   
## Y dimension: 200 1  
## Fit method: svdpc  
## Number of components considered: 11  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 479.6 307.2 307.0 306.9 299.3 273.2 270.3  
## adjCV 479.6 306.7 306.7 307.2 298.2 272.8 268.1  
## 7 comps 8 comps 9 comps 10 comps 11 comps  
## CV 262.2 255.5 254.9 102.8 101.4  
## adjCV 261.8 255.0 254.3 102.4 101.1  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 25.58 40.56 51.56 61.42 70.87 79.34 87.20 94.74  
## Balance 59.81 59.96 60.18 63.82 69.74 70.97 71.72 74.04  
## 9 comps 10 comps 11 comps  
## X 98.19 99.98 100.00  
## Balance 74.33 95.89 96.04

validationplot(pcr.fit, val.type = "MSEP")  
axis(side=1, at=seq(1,20,by=1))

A graph with numbers and a line

Description automatically generated

pcr.pred <- predict(pcr.fit, Credit[test, ], ncomp = 11)  
pcr.mse <- mean((pcr.pred - Credit$Balance[test])^2)  
pcr.mse

## [1] 10691.12

sqrt(pcr.mse)

## [1] 103.3979

lm.fit <- lm(Balance ~ ., data = Credit, subset = train)  
lm.pred <- predict(lm.fit, Credit[test, ])  
lm.mse <- mean((lm.pred - Credit$Balance[test])^2)  
lm.mse

## [1] 10691.12