STAT 4620 – Final Project

Michael Cooch, Julia Cuva, Krescens Kok, Zachary Meder, Jaima Schulte

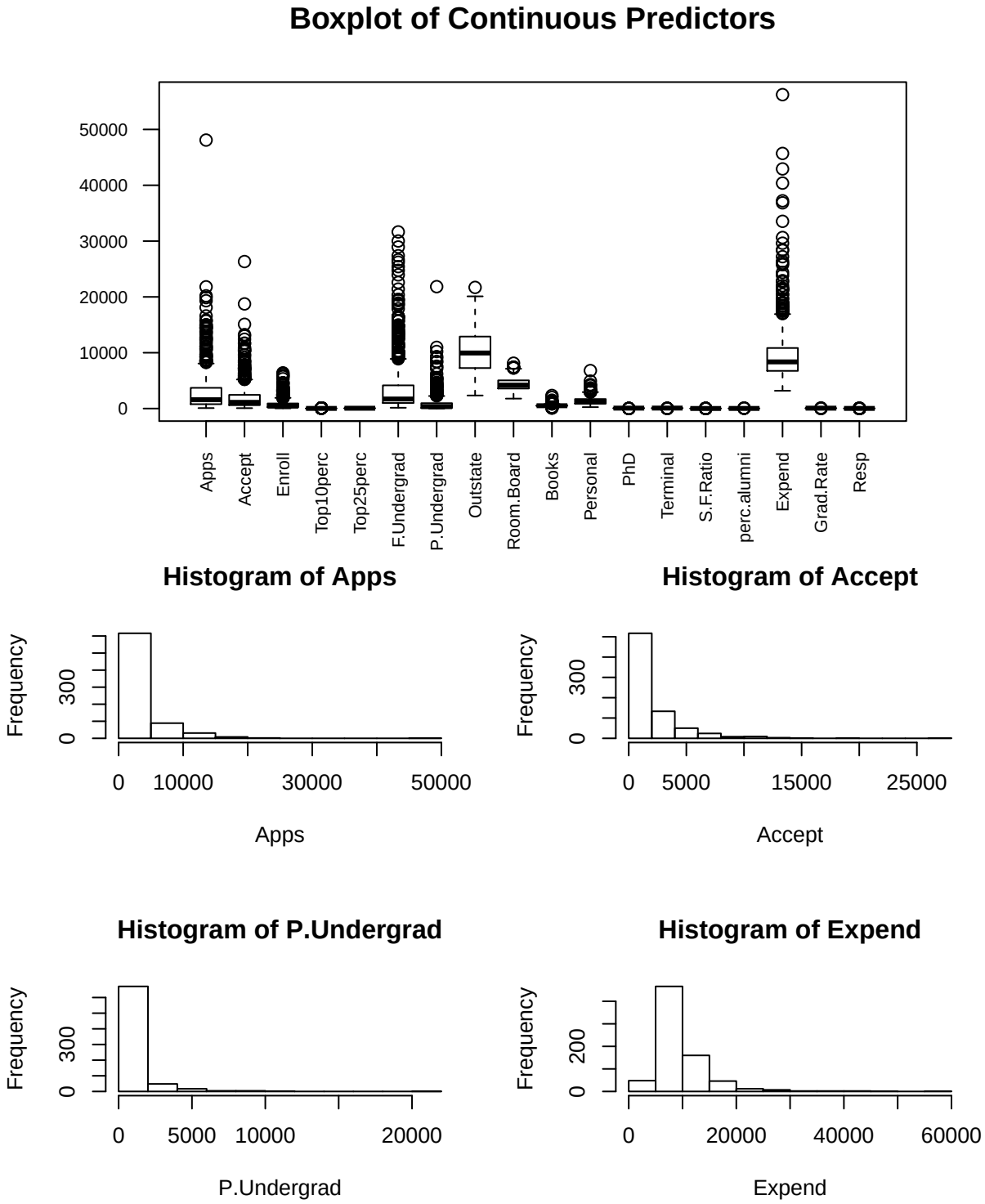
12/4/2020

Introduction

This dataset contains statistics for a large number of US Colleges from the 1995 issue of US News and World Report. It contains 777 colleges and 19 predictor variables. This dataset contains both quantitative and qualitative variables. After performing an exploratory data analysis on the dataset, it is evident that there are no duplicate rows, however, there are missing values in both response variables, which have been admitted.

Attribute Name	Description	Type of Data
X	Name of the college	Nominal
Private	Whether the college is a private school or not	Nominal
Apps	Number of applications received	Interval
Accept	Number of applications accepted	Interval
Enroll	Number of new students enrolled	Interval
Top10perc	Pct. new students from top 10% of H.S. class	Interval
Top25perc	Pct. new students from top 25% of H.S. class	Interval
F.Undergrad	Number of full time undergraduates	Interval
P.Undergrad	Number of part time undergraduates	Interval
Outstate	Out of state tuition	Interval
Room.Board	Room and board costs	Interval
Books	Estimated book costs	Interval
Personal	Estimated personal spending	Interval
PhD	Pct. of faculty with Ph.D.'s	Interval
Terminal	Pct. of faculty with terminal degree	Interval
S.F.Ratio	Student/faculty ratio	Interval
perc.alumni	Pct. alumni who donate	Interval
Expend	Instructional expenditure per student	Interval
Grad.Rate	Graduation rate	Interval
Resp	Response variable that we are trying to classify/predict	Interval
RespB	Response variable that we are trying to classify/predict	Nominal

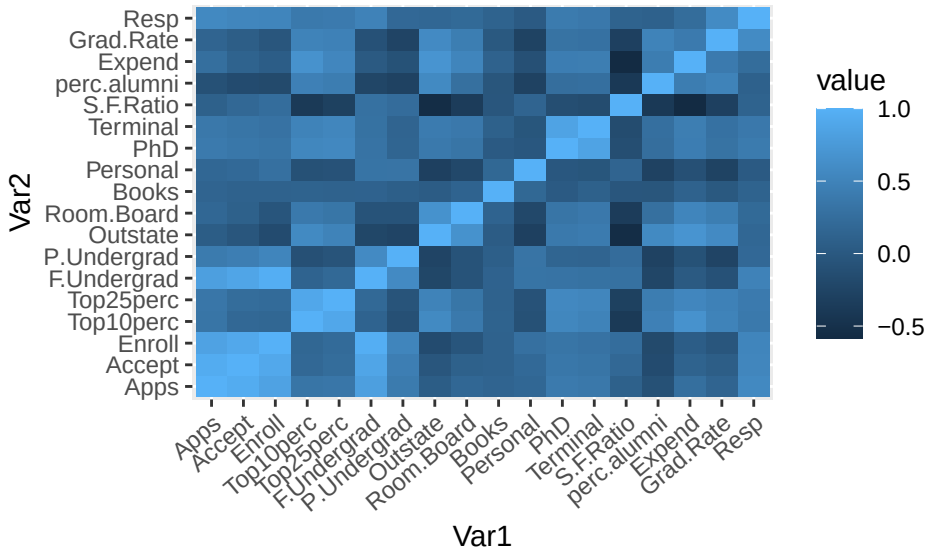
We looked at the boxplots of all the quantitative variables to easily identify the outliers of all the variables.



It seems that `Apps`, `Accept`, `F.Undergrad`, and `Expend` all have obvious outliers. To get a better visual of the variables that could have outliers, the histograms are provided below to see which outliers seem reasonable to remove. Since the outliers of `Apps`, `Accept`, and `Expend` are pretty far from the rest of the data, it would be reasonable to remove these data points. The following two rows were removed due to being outliers:

```
df= df[-c(467, 274), ]
```

Variable Selection



Looking at the correlation plot above, we can see that `Top10perc` and `Top25perc`; `Accept`, `Enroll` and `Apps`; and `Terminal` and `PhD` are highly correlated. For this reason, we decided to remove `Top10perc`, `Accept`, and `Terminal` from further analyses to reduce the number of dimensions in our data. We decided to remove `Top10perc` because it is encompassed in the `Top25perc` variable. `Accept` and `Terminal` were removed due to the high correlation with other closely related predictors.

Training/Testing

For building the models we decided to use 80% of the data for training and the remaining 20% for testing.

Continuous Modeling

In order to predict the continuous response, `Resp`, we investigated several statistical learning models: LASSO, Ridge, PCR/PLS, Decision trees and CART variations. We will evaluate and compare the performance of these models using the test Mean Squared Error (MSE).

LASSO

The first model that we ran was a LASSO regression, which is a type of linear regression that uses a shrinkage penalty to fit the model to the data. It can also perform variable selection by shrinking less correlated variables to 0, and eliminate them from the model. The parameters for building/fitting the model are as follows: training data for both x (predictors) and y (`Resp`), `alpha = 1`, lambda. We used cross validation to find the optimal lambda value, which you can see below.

```
lasso.fit = cv.glmnet(train.matrix, dfRespTrain[, "Resp"], alpha=1, lambda=grid)
cv.lambda = lasso.fit$lambda.min
```

```
lasso.predictions = predict(lasso.fit, newx=test.matrix, s=cv.lambda)
mse.lasso <- mean((dfRespTest[, "Resp"] - lasso.predictions)^2)
```

The lambda value used in the LASSO regression was chosen using cross validation and had a value of 0.031. The MSE for the fitted LASSO regression is 3.559. In this model, only the `Top25perc` coefficient was shrunk to 0.

Ridge

Similar to LASSO, Ridge is a shrinkage method, with the main difference being that Ridge does not perform variable selection (i.e. all parameters are in the final model with shrunken estimates). Setting up the model involves creating a model matrix for the predictors and a vector of the y values, in this case `Resp`. Additionally the parameters for building/fitting the model are as follows: training data for both x (predictors) and y (`Resp`), `alpha = 0`, lambda. We used cross validation to find the optimal lambda value, which you can see below.

The cross validation lambda values was 0.235. After finding the cross validated value of lambda, we fit the model using the following.

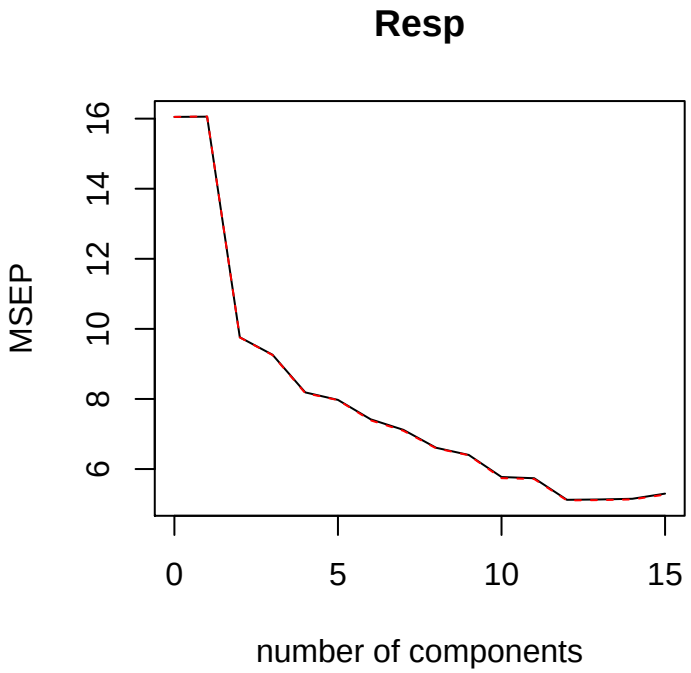
```
fit_ridge_cont = glmnet(dfRespTrain_x, dfRespTrain_y, alpha = 0, lambda = lambda_cv_cont)
```

Lastly we used the model and the testing datasets for x and y to find the model's error. Using this value we got a MSE of 3.453, slightly lower than the LASSO model.

PCR/PLS

The PCR and PLS models are used to find the ideal number of components in the model to reduce the MSE. Scale is set to `TRUE` so that the data is standardized, which helps improve the interpretation of the model. Cross validation is specified in the validation parameter to test the model's ability to predict new data in order to identify overfitting or bias. The main difference between these two models is that PCR is unsupervised and PLS is supervised.

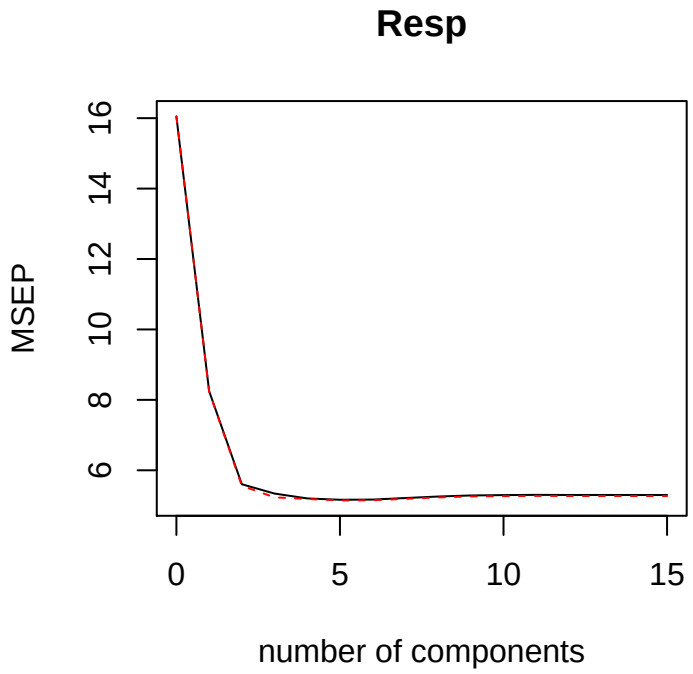
PCR:



As we can see from the plot above, M=12 (where M is the number of components) has the lowest MSE value. M=13 and M=14 does almost equally well. Furthermore, the summary showed M=12, M=13, and M=14 all led to over 95% for X and over 69% for Y. With these observations, we will compute the MSE using the test data for these three models. Looking at the three values, it is evident that 13 components produces the smallest MSE value.

Number of Components	MSE
M = 12	3.556
M = 13	3.523
M = 14	3.560

PLS:

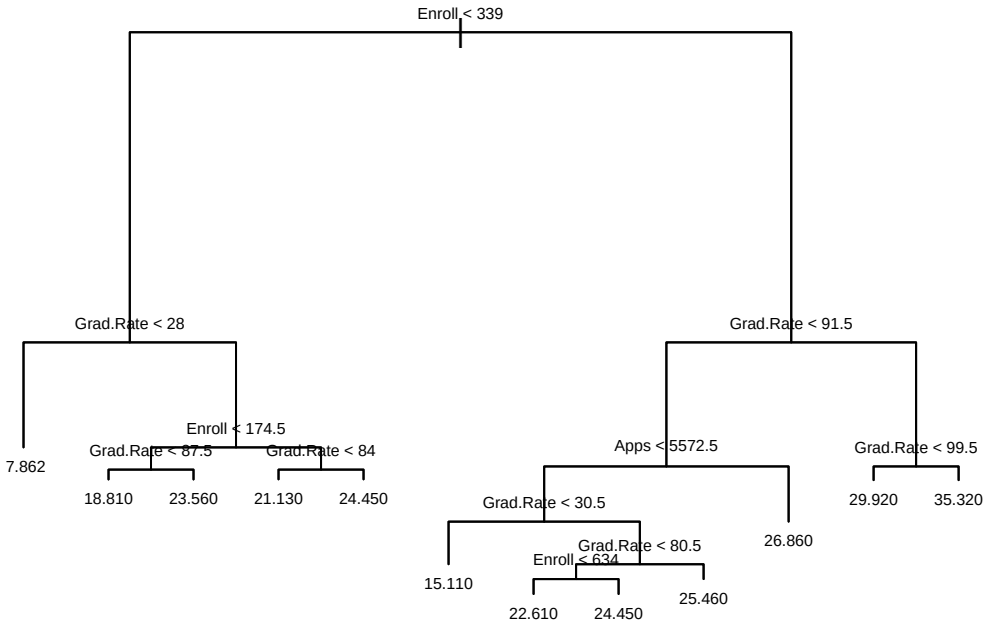


Using PLS, M=5 and M=6 have the lowest MSE values. However, M=12 explains 93.25% of the variability in X and 70.25% in Y, while M=5 only explains 73.66% of the variability in X and 70.66% in Y. With these observations, we will evaluate the MSE using the test data for M=5, M=6, and M=12. Looking at the results below, it seems that M=6 produces the smallest MSE value (although, very close MSE values to the other values), therefore having 12 PLS components is suitable for fitting the college dataset.

Number of Components	MSE
M = 5	3.593
M = 6	3.551
M = 12	3.541

Decision Tree

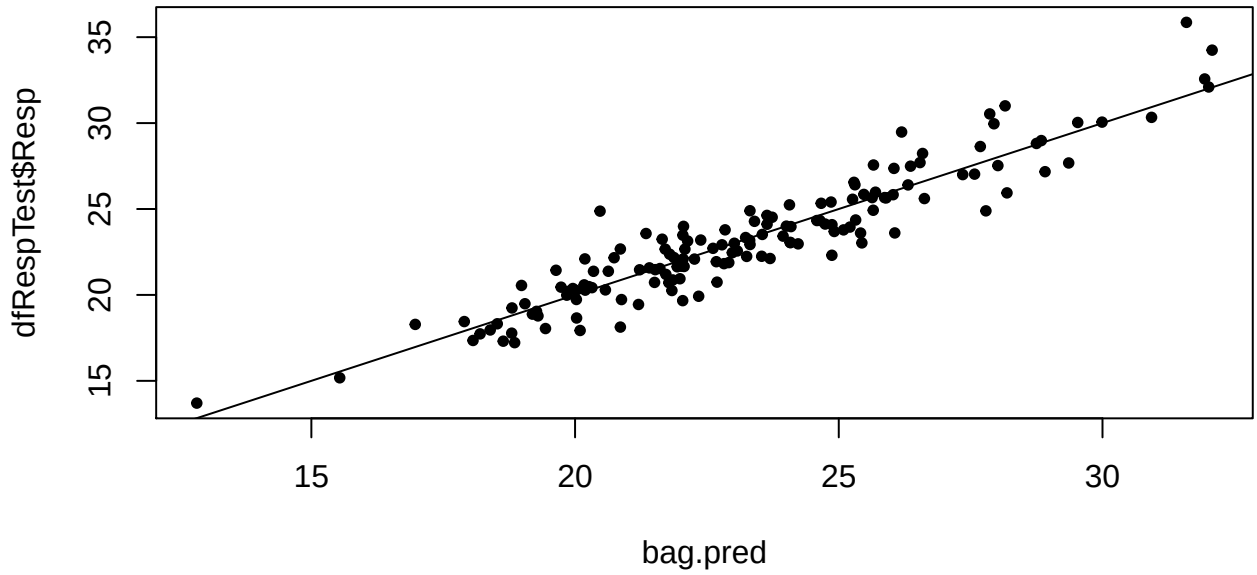
```
set.seed(4620)
cv.fit = cv.tree(tree.fit)
size = cv.fit$size[which.min(cv.fit$dev)]
prune.fit = prune.tree(tree.fit, best = size)
plot(prune.fit)
text(prune.fit, pretty = 0, cex = 0.5)
```



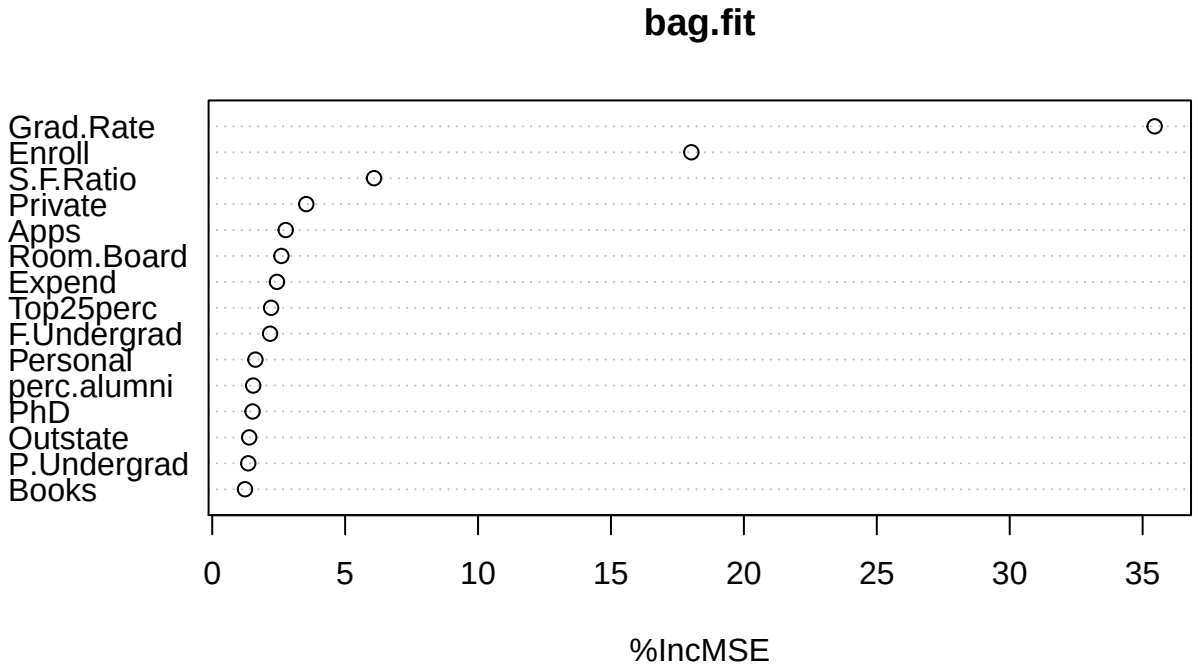
First, we fit a decision tree to predict `Resp` which achieves a test MSE of 2.85. Next, we use cross-validation to prune the tree but our results are the same. The number of terminal nodes is 12 and the test MSE is 2.85. A visual representation of the tree is plotted above.

Bagging

Next, we fit a bagging model to attempt to reduce the variance of the decision tree. We fit a bagging model with B = 100 since the model is not sensitive to B as long as it is big enough. We also set `mtry` equal to 15 so that the total number of predictors to consider at each split is the total amount of predictors in our dataset.



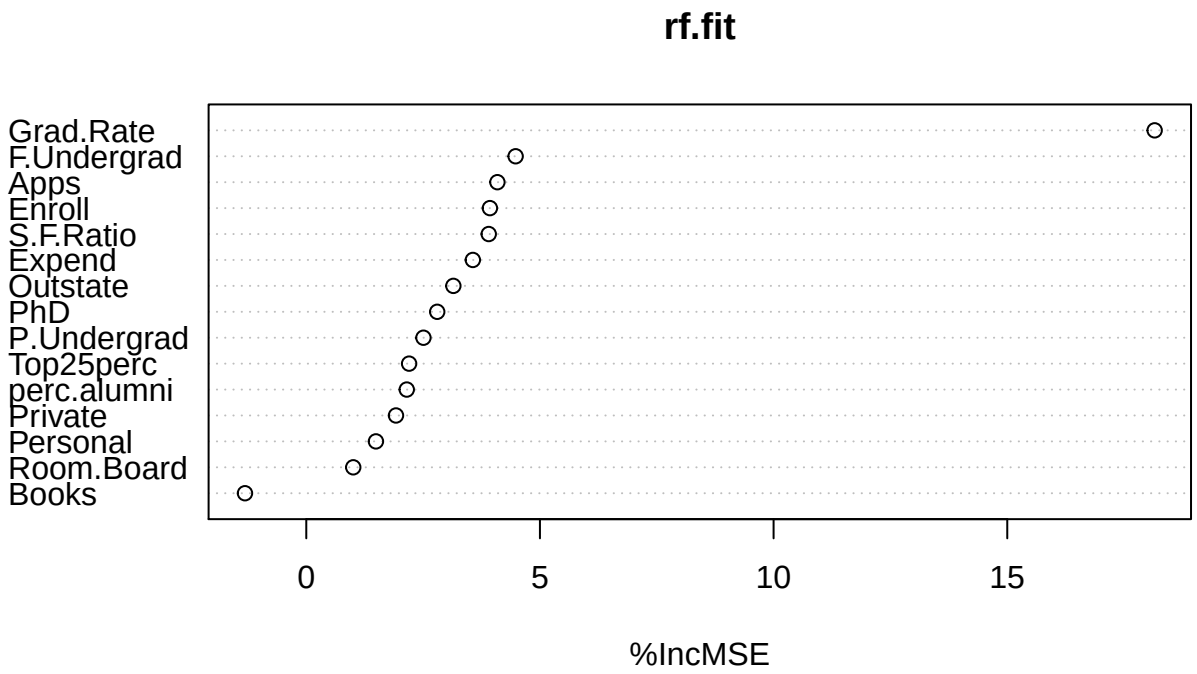
In the plot above we can see the performance of our bagging model. The predictions are on the x-axis and the and the test `Resp` values are on the y-axis. This model has a test MSE of 1.652 which is an improvement over the decision tree.



The bagging model is a collection of many trees (in our model it is 100), so the interpretation becomes less clear than that of a single decision tree. The variable importance measures and plot above shows the relative error increase when that variable is removed from the bagging model. We can see that `Grad.Rate` and `Enroll` are the most important.

Random Forest

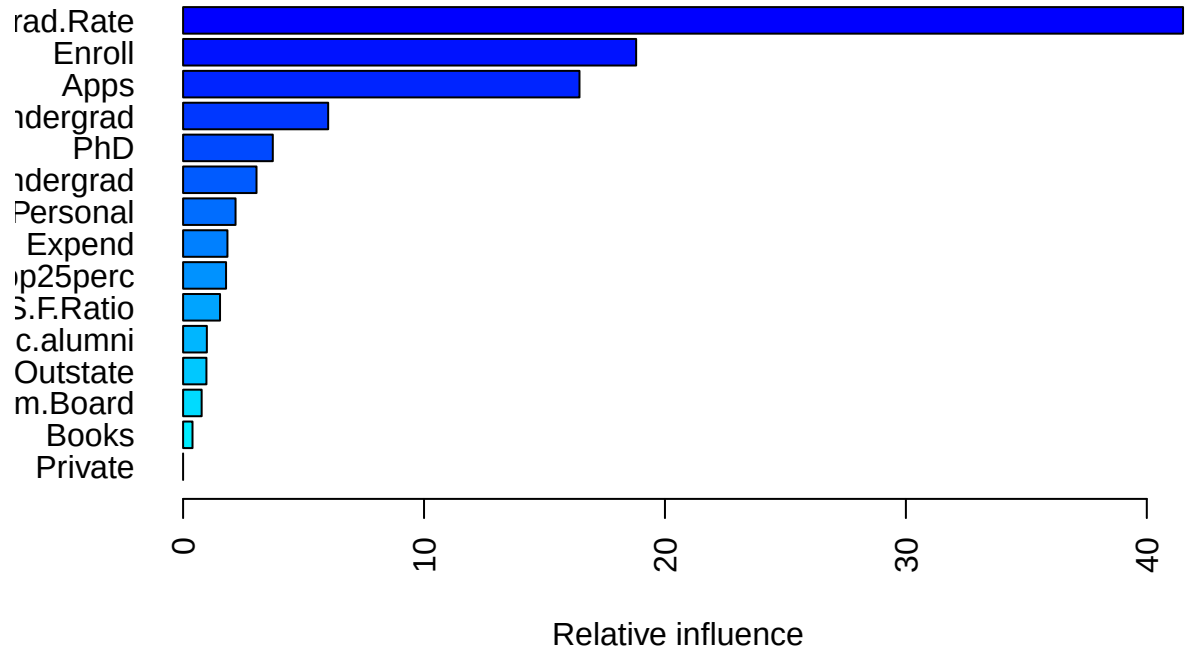
An issue with the bagging model above is that there is a lot of overlap in the observations appearing in each bootstrap sample. In order to reduce the variance even further by decorrelating the trees in the bagging model, we fit a random forest model. We set the number of trees equal to 100 but set `mtry` equal to 5, so at each split in the tree a random sample of 5 out of the 15 predictors are available to be selected for splitting. This should make the trees decorrelated from one another and reduce the variance.



The test MSE for the random forest model is 2.008, which is greater than the bagging model. This means that decorrelating the trees did not lead to a reduction of the variance. The plot again shows the relative error increase when that variable is removed from the bagging model. Once again, `Grad.Rate` is the most important variable in the random forest model.

Boosting

Lastly, we try fitting a boosting model to improve performance. We use `n.trees = 100`, `interaction.depth = 2`, and use 10-fold cross validation to select the best number of trees. `distribution` is set to gaussian since we are predicting the `Resp`, a continuous response, and the default learning rate of 0.1 is used.



The optimal number of trees selected by CV is 96. This results in a test MSE of 1.521, which is the best of the tree methods. The relative influence plot shows that `Grad.Rate` and `Enroll` are once again some of the most important variables.

Comparison of Models

Model	MSE
LASSO	3.54
Ridge	3.45
PCR	3.52
PLS	3.54
Decision Tree	2.85
Bagging	1.65
RF	2.01
Boosting	1.52

The boosting model had the lowest test MSE out of the eight models fit, thus performed the best in predicting `Resp`.

Categorical Modeling

In addition to the continuous models above, we fit several models to the categorical response, `RespB`: LASSO, Ridge, LDA, QDA, Decision Trees, and CART variations. When evaluating the performance of these models in predicting `RespB,` we focused on the accuracy to compare the different models.

LASSO

When predicting a categorical response, the LASSO regression uses a penalized version of a logistic regression. It again performed variable selection by shrinking less correlated variables to 0, and eliminating them from the model.

```
lasso.fit2 = cv.glmnet(train.matrix, dfRespBTrain[, "RespB"], alpha=1, lambda = grid,
                       family = "binomial")
cv.lambda2 = lasso.fit2$lambda.min
probabilities = predict(lasso.fit2, newx = data.matrix(dfRespBTest[-c(16)]),
                        type = "response", s = cv.lambda2)
predicted.classes <- ifelse(probabilities > 0.5, "Yes", "No")
```

```
##            Reference
## Prediction No Yes
##        No  58  23
##        Yes  2  66
```

The lambda value used in the LASSO regression was again chosen using cross validation and had a value of 0.01 and the accuracy obtained was 0.832. The LASSO classification shrunk the following variables to 0: `F.Undergrad`, `Outstate`, `Room.Board`, `Personal`, and `Expend`.

Ridge

The Ridge model for the categorical response involves a similar setup to the model for the continuous response. The main difference is that you have to specify `family = "binomial"`, so we are specifying a binary response and should use logistic regression. Other than that, the setup and parameters are the same, including finding lambda using cross validation. The optimal lambda value can be seen below.

The cross validation lambda values was 0.023. After finding the lambda value we fit the model using the following.

```
fit_ridge_bin = glmnet(dfRespBTrain_x, dfRespBTrain_y, alpha = 0,
                       lambda = lambda_cv_bin, family="binomial")
```

```
##           Reference
## Prediction No Yes
##        No  53  13
##        Yes  7  76
```

From the confusion matrix above, we can see that the accuracy for the Ridge model is 0.87.

LDA

For the LDA model, we assume the variances between the two classes of `RespB` are equal.

```
respB.lda.fit = lda(RespB ~ ., data = dfRespBTrain)
```

```
##           Reference
## Prediction No Yes
##        No  49  16
##        Yes 11  73
```

The LDA model had 122 correct predictions - having a test accuracy of 81.9%.

QDA

QDA differs from LDA because we assume that each of the two class in `RespB` has its own variance-covariance matrix.

```
respB.qda.fit = qda(RespB ~ ., data = dfRespBTrain)
```

```
##           Reference
## Prediction No Yes
##        No  54  34
##        Yes  6  55
```

The QDA model had 109 correct predictions - having a test accuracy of 73.2%.

KNN

KNN is a non-parametric approach to classification that uses a sample-based estimate of a conditional probability in order to arrive at a classification rule.

```
set.seed(4620)
respB.knn.preds = knn(train.knn, test.knn, train.respB, k=28)
cm.knn <- confusionMatrix(respB.knn.preds, dfRespBTest$RespB, positive = 'Yes')
cm.knn$table
```

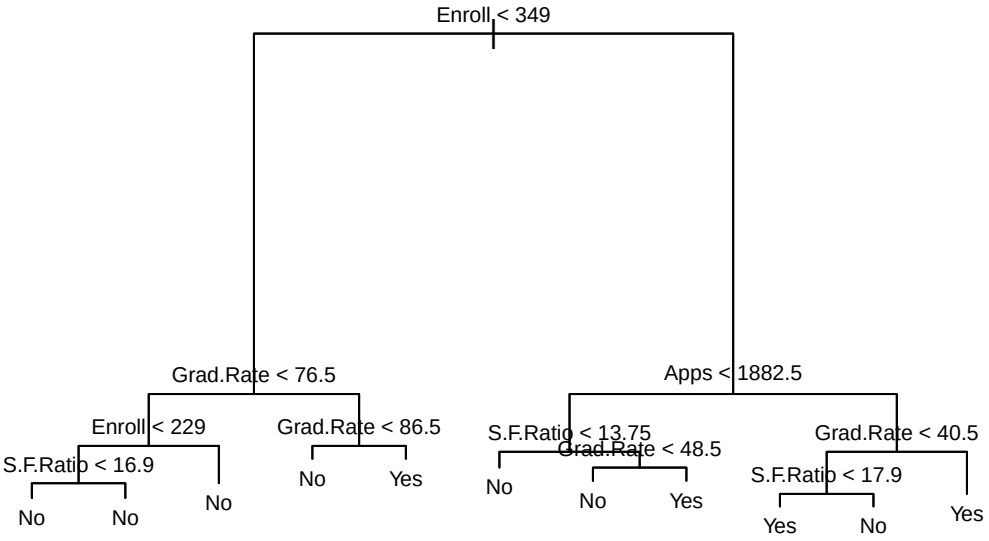
```
##           Reference
## Prediction No Yes
##        No  49  18
##        Yes 11  71
```

The KNN model had 109 correct predictions - having a test accuracy of 80.5%. KNN performs better than QDA but still worse than LDA, thus LDA is the best performing model of these three.

Decision Tree

```
set.seed(4620)
tree.fitB <- tree(RespB ~ ., data = dfRespBTrain, split="gini")
```

```
set.seed(4620)
cv.fitB <- cv.tree(tree.fitB, FUN=prune.misclass)
# FUN=prune.misclass uses misclassification to prune the tree
sizeB <- cv.fitB$size[which.min(cv.fitB$dev)]
prune.fitB <- prune.tree(tree.fitB, best = sizeB)
plot(prune.fitB)
text(prune.fitB, pretty = 0, cex = 0.7)
```




```
##           Reference
## Prediction No Yes
##        No  56  20
##        Yes  4  69
```

We first fit decision tree to predict `RespB` which uses the Gini index as a measure of node impurity to choose the predictor at each split. This tree results in a test accuracy of about 81% and has 36 terminal nodes. Next, we use cross-validation to prune the tree which results in a tree with only 11 terminal nodes, a significant reduction from the prior tree, and can be seen in the plots above. The test accuracy on the pruned tree increases to 84%.

Bagging

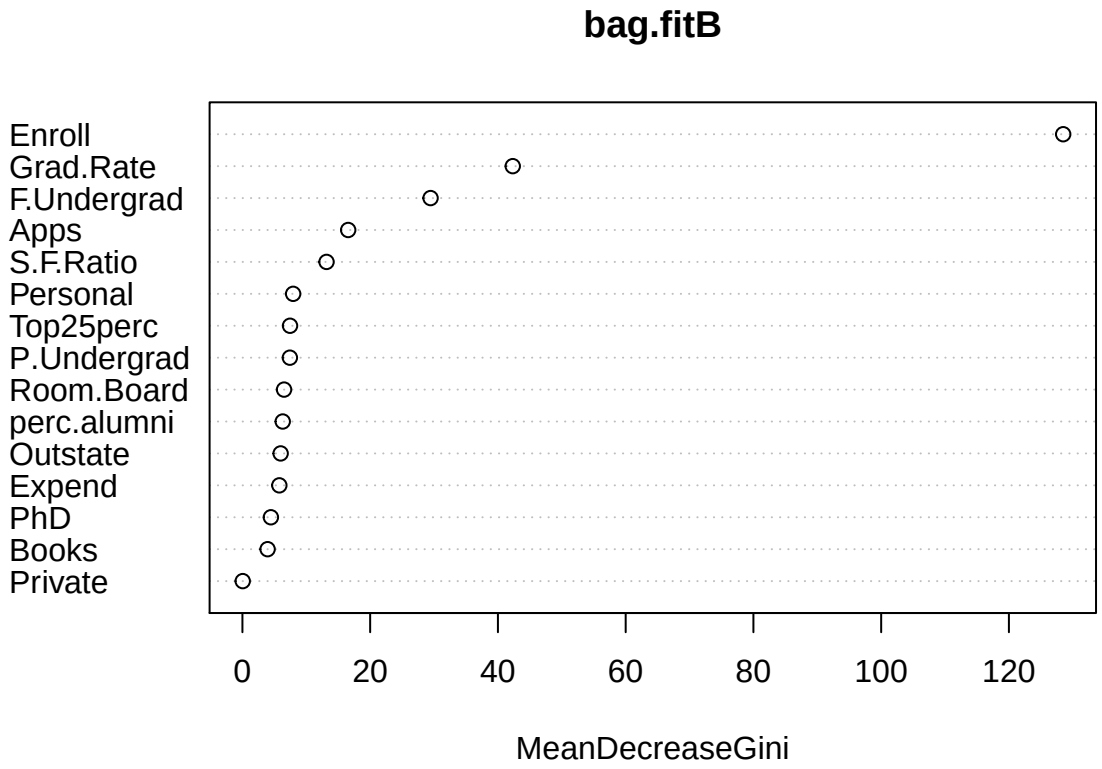
We fit a bagging model to attempt to reduce the variance of the decision tree in making predictions of `RespB`. We also set `mtry` equal to 15 again so that the total number of predictors to consider at each split is the total

amount of predictors in our dataset. `type` is set to `"class"` so that the predicted responses are dichotomous.

```
set.seed(4620)
bag.fitB <- randomForest(RespB ~ ., data = dfRespBTrain, mtry=15, importance = TRUE,
                         ntree=100)
```

The test accuracy is about 89% which is an improvement over the decision tree. However, the training accuracy was 100% so there is likely overfitting in the training process.

```
##           Reference
## Prediction No Yes
##        No  55  12
##        Yes  5  77
```



From the plot above we can see that the most important variables in the bagging model are `Enroll`, `Grad.Rate`, and `F.Undergrad`. The importance is computed using the mean deacrease in Gini index for each variable relative to the maximum.

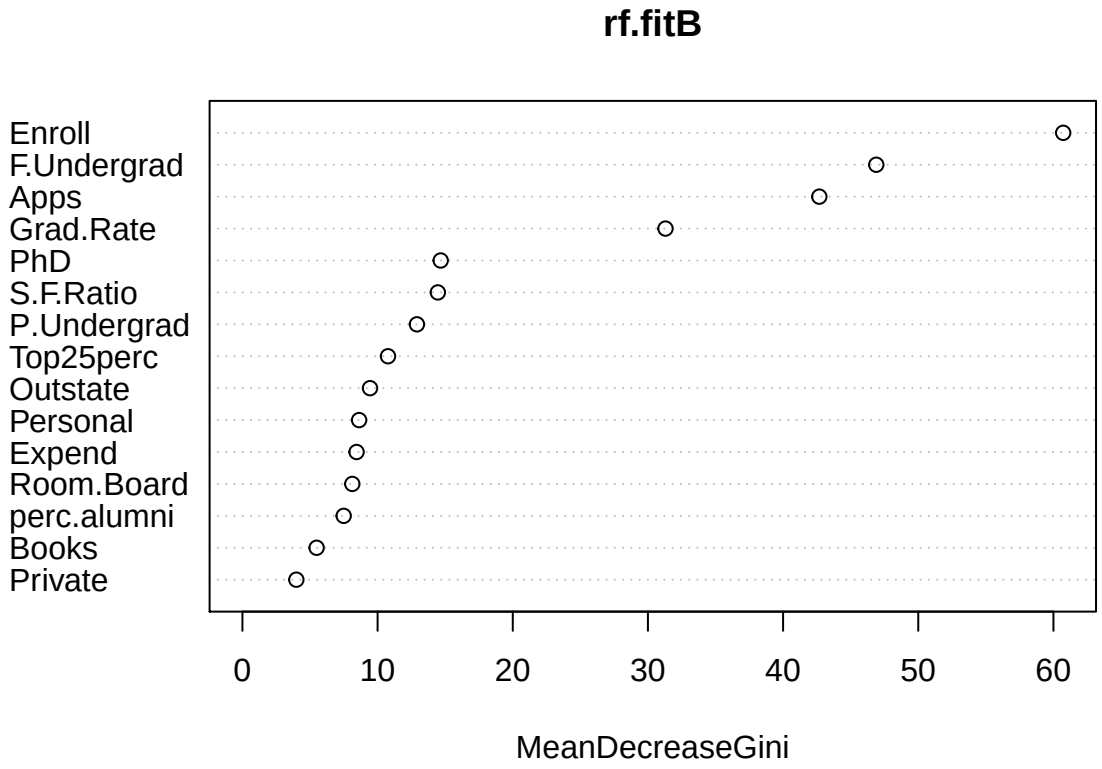
Random Forest

We fit a random forest classification model but set `mtry` equal to 3, so at each split in the tree a random sample of 3 out of the 15 predictors are available to be selected for splitting. This should make the trees decorrelated from one another and reduce the variance.

```
set.seed(4620)
rf.fitB <- randomForest(RespB ~ ., data = dfRespBTrain, importance = TRUE)
# mtry = sqrt(15) for classification, sqrt(15)~=3
```

```
##           Reference
## Prediction No Yes
##        No  53  13
```

```
##        Yes  7  76
```

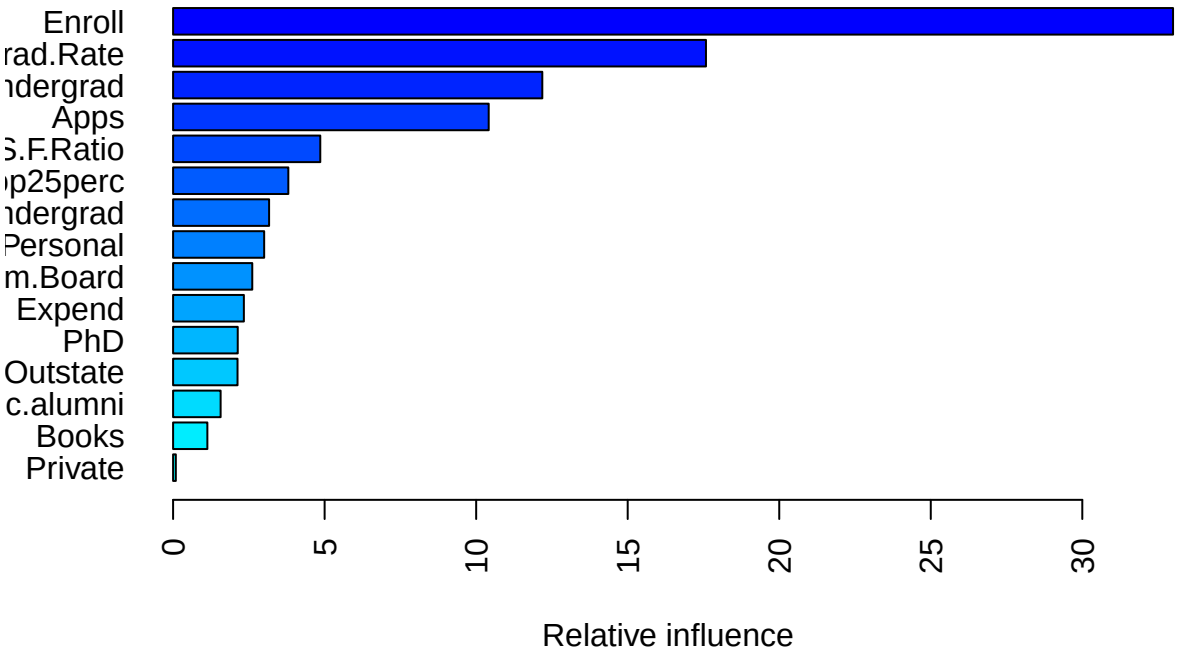


The test accuracy for the random forest classification model is about 87%, which is slightly less than that of the bagging model. The plot again shows the mean decrease in the Gini index when that variable is removed from the random forest model. `Enroll`, `F.Undergrad`, `Apps`, and `Grad.Rate` are the most important variables in the random forest classification model.

Boosting

Lastly, we fit a boosting classification model to predict the `RespB` class. We use `n.trees = 100`, `interaction.depth = 2`, and use 10-fold cross validation to select the best number of trees. `distribution` is set to bernoulli since we are predicting the `RespB`, a dichotomous response, and the default learning rate of 0.1 is used.

```
set.seed(4620)
trainB.boost <- dfRespBTrain %>% mutate(RespB = ifelse(RespB == "Yes", 1, 0))
testB.boost <- dfRespBTest %>% mutate(RespB = ifelse(RespB == "Yes", 1, 0))
boost.fitB <- gbm(RespB ~ .,
                  data = trainB.boost,
                  distribution = "bernoulli",
                  n.trees = 500,
                  interaction.depth = 2,
                  cv.folds = 10)
```




```
##           Reference
## Prediction  0  1
##          0 51 10
##          1  9 79
```

The optimal number of trees selected by CV is 110. This results in a test accuracy of about 87%, which is the best of the tree methods for classification. The relative influence plot shows that `Enroll`, `Grade.Rate`, `F.Undergrad`, `Apps` are once again some of the most important variables.

Comparison of Models

Model	Accuracy
LASSO	0.83
Ridge	0.87
LDA	0.82
QDA	0.73
KNN	0.81
Decision Tree	0.84
Bagging	0.89
RF	0.87
Boosting	0.87

When comparing the accuracies of the 9 models predicting the categorical response, the bagging model performed the best with an accuracy of 0.89.

Conclusion

Overall, we saw that the models we built both for the continuous response (`Resp`) and dichotomous response (`RespB`) performed very well. The best model in terms of the MSE for the continuous response was boosting,

however bagging had comparable performance. Lasso, Ridge, PCR, and PLS all performed similarly as well, with an MSE of around 3. Lastly, decision trees and random forests performed somewhere in between. We chose the boosting model as the best for predicting `Resp` due to it having the lowest test MSE and improving the bias-variance tradeoff. For the dichotomous response variable, we previously stated that bagging had the highest accuracy of 89%, but the accuracies of random forest, boosting, and ridge regression were just slightly lower at 87%. Given that the accuracies are very similar, we would be comfortable using any of these models to predict `RespB`. However, because bagging had the highest accuracy and improves the bias-variance tradeoff this is the model we chose as the best.