I) Introduction

The purpose of this project was to investigate the performance of various machine learning method in predicting marital status of couples based on answers to a set of survey questions. We will compare the accuracy among the models, as well as identify which of the questions asked on the survey were particularly effective at classifying couples by marital status.

II) Dataset

The dataset analyzed for this project comes from “Divorce Prediction Using Correlation Based Feature Selection And Artificial Neural Networks”, a University of California Irvine study. The data consisted of 84 divorced (Class=1) and 86 married (Class=0) couples' opinions on 54 marriage-related questions on a 5-factor scale (0-4) and contained no null values. For each of the attributes, a general trend observed was that the responses were mostly bimodally distributed, heavy on 0, 3, and 4. Although the dataset is fairly small (n=170), The distribution of classes (84 divorced to 86 married) roughly reflects the real life statistics of marital status among couples (where roughly 50% of married couples end up divorced)

III) Methods

Overview:
A mixture of linear and non-linear classifiers was chosen to analyze the data. To compare both the accuracy and consistency of machine learning methods, these reduced models were tested on 50 different random seeds (with an 80/20 train/test split) in order to yield average test MSEs and test MSE variances.

Given the small size of the data and the large number of predictors, it made sense to select a reduced classifier prior to testing to avoid overfit models. Thus, various forms of variable selection were employed to select the best reduced models for each method. Since we want to measure the overall effectiveness of models given any random split of training/test data, best model selection for each method were performed independently at every seeds (rather than predetermined based on the data of the first seed, then run the same models across all seeds).

Logistic Regression – A reduced logistic regression model was determined by lasso selection, a method that selects linear models by placing penalties on models with more predictors, with the optimal tuning parameter calculated through 5-fold cross validation. Predicted values, ranging from 0 to 1, were converted into factors with a threshold of 0.5 (<0.5=married, ≥ 0.5=divorced).

Linear Discriminant Analysis – Reduced LDA models were determined by forward (beginning with a null model, then adding important attributes) and backward selection (beginning with a full model, then removing less important attributes). The optimal best subset was determined based on cross-validation of test errors. In cases where different best subsets yielded the same test error, the less flexible model (smaller subset) was chosen to minimize chance of overfitting.

Tree-based Models – Multiple tree-based methods were considered: 1) a pruned, single classification tree 2) bagged trees and 3) a random forest with 7 variables (approximately the square root of p) considered at each split. Useful predictors for each split were selected using the Gini Index metric.

K-Nearest Neighbors – The unsupervised learning method KNN was also tested, with the optimal number of k determined through k-fold cross validation. In cases where different values of k yielded the same test error, the less flexible model (larger k) was chosen to minimize chance of overfitting.

All of the above methods, except for KNN, also allowed for identification of important attributes. Across 50 seeds, the top 5 most important attributes as identified by each method were aggregated in the following ways:

- For logistic regression, the 5 most common attributes that appear in the 50 chosen lasso-reduced models
- For LDA, the 5 most common attributes that appear across the 50 chosen optimal best subsets models
- For trees, the 5 most common attributes used in the root node of the 50 chosen pruned trees
- For bagging and random forest, the 5 attributes that resulted in the most decrease in Gini impurity across the 50 chosen models

IV) Results

Overall, variable selection for each method resulted in fairly small models. Across the 50 seeds, the size of logistic regression models ranged from 5-10 predictors, both forward and backward-selected LDA ranged from 1-4 predictors, trees ranged from 1-3 splits, and KNN’s optimal k generally very high, as analyzed in the graph below. All of this indicates that simple models were sufficient to separate the classes.

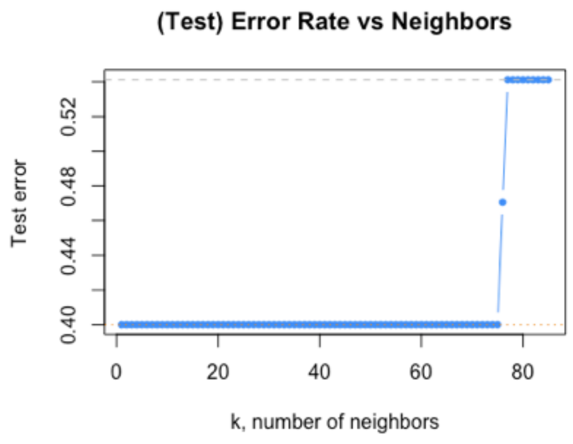


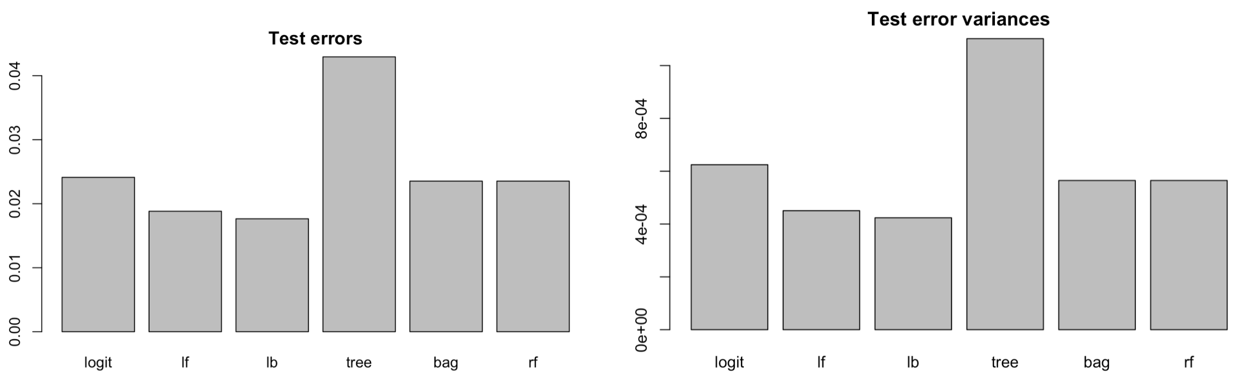
Chart 1: Test error rates of KNN model as a function of number of neighbors. As is apparent, high values of k were sufficient to minimize test errors. In training, the highest value of k that minimizes errors is always chosen to lower the chance of overfit models.

All methods except for KNN achieved fairly low error rates, ranging between 0 and 5%. KNN, however, yielded an error rate of 49.6%, which is as about as inaccurate as random guessing.

Backward-selected LDA turned out to be the best performing model, with both minimum mean test error and test error variance. The method also seemingly had no outliers negatively influencing the distribution of test errors, highlighting its consistency. The tables and plots below outline the full result (for easier interpretability, KNN results were omitted from the plots – See Appendix for plots that includes KNN results).

Method	**Mean Test Error**	**Test Error Vaiance**
Logit	0.02411765	0.000624603
Fwd LDA	0.01882353	0.000450533
Bwd LDA	**0.01764706**	**0.000423699**
Tree	0.04294118	0.00110197
Bag	0.02352941	0.000564932
Random Forest	0.02352941	0.000564932
KNN	0.49602941	0.001730567

Table 1: Mean test error and variance of errors



Charts 2-3: Bar graph visualization of mean test error and variance of errors

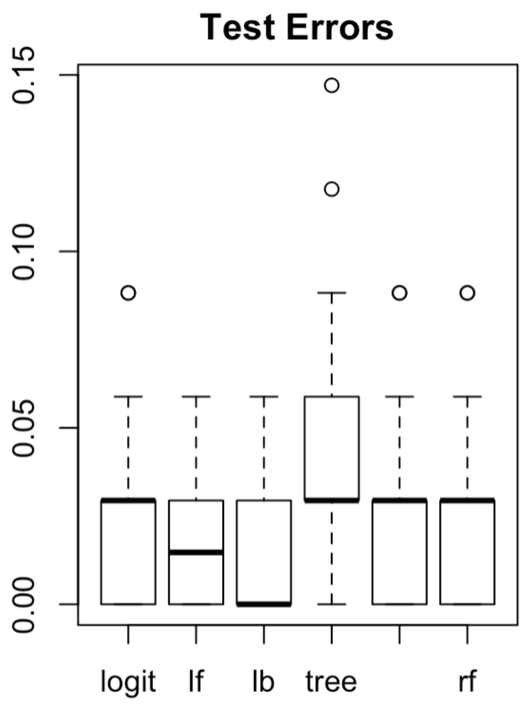


Chart 4: Boxplot visualization of test errors across 50 seeds

All methods except for KNN also allowed for important variable selection. The table below outlines the ranking of the top 5 most important attributes as determined by each method (procedure outlined above in the methods section). Overall, the methods generally identified the same set of attributes as important. Based on the result, responses to questions 17, 18, 19, and 40 seem to the ones that are most predictive of marital status.

Method	**Ranked Important Attributes (top 5)**
Logit	40, 18. 17, 19, 26
Fwd LDA	40, 18. 17, 6, 19
Bwd LDA	40, 6, 26, 18, 17

Tree	19, 12, 18, 10, 13
Bag	18, 40, 19, 17, 26
Random Forest	18, 40, 19, 17, 11

Table 2: Ranked top 5 important attributes as determined by each method

This means that in real-life, one could sufficiently classify couples by marital status based on knowledge of responses to just these following 4 attributes:

- My spouse and I have similar ideas about how marriage should be
- We share the same views about being happy in our life with my spouse
- My spouse and I have similar ideas about how roles should be in marriage
- We're just starting a discussion before I know what's going on

V) Discussion

The overall result provides several insights into the data. Overall, the models (except for KNN) showed very high performance, with error rates ranging between 0 and 5% and error variances below 0.2%. This means that the models were generally very accurate and very consistent in their performances. Given that this was a simple binary classification task of two fairly distinct classes in real life, this general level of performance is not surprising.

The poor performance of KNN is most likely due to scaling of the Attribute response values. Seen from preliminary analysis of the data, the responses for the attributes are bimodally distributed among 0, and combination of 3 and/or 4. For the other models, this would not have made a significant difference. However, in KNN the range of the values are an important part. KNN calculates the distance between the observations. The distance between 0 and 3 and/or 4 is large considering that they are the two ends of the possible responses for each attribute. Overall, this may have contributed to the inaccuracy of KNN.

Aside from KNN, single-tree models stood out as significantly less accurate than other models. Looking at the boxplots of test errors, it can be seen that the mean test errors for trees were significantly impacted by a couple of large outliers. This is most likely due to the fact that single-tree models are immensely susceptible to overfitting. Coupled with being trained on such a small dataset and the large number of predictors to choose from, it’s understandable that tree models overall had both high error and variance.

The fact that LDA outperforms logistic regression suggests that the real-life distributions of predictors (the responses to survey questions) are roughly normally distributed with equal variances. Additionally, high performance among linear classifiers such as LDA and logistic regression indicates that, despite a large number of predictors, the classes were highly linearly separable, and that perhaps simple, linear machine learning methods are more effective at predicting divorce rates than more complex non-linear models. The fact that only a small number of attributes among the 54 presented in the dataset stood out as important predictors (attributes identified in the result section were by far the most significant, while the remaining attributes had little predictive power) supports this interpretation. In other words, the real-life difference between a married and divorced couple can be explained by just a few, highly predictive characteristics of their marriage.

VI) Conclusion

The analysis indicates that due to the high level of linear separability among the data's classes, couples in real life can be easily classified by marital status using a small number of predictors and simplistic machine learning methods. Among the models tested, backward-selected LDA yielded the highest performance on test data, and questions 17, 18, 19, and 40 of the surveys proved to be particularly predictive attributes.

divorce_project

Minh Nguyen

12/4/2019

Preparing the data

```
library(readxl)
data <- read_excel("divorce.xlsx")
divorced <- ifelse(data$Class==0, "No", "Yes") # convert martial status from numeric to factor
data <- data.frame(data, divorced)
set.seed(1) # Variable selection for reduced models will be based on the 1st seed.
split <- sample(1:nrow(data), nrow(data)*0.8) # 80/20 train/test split
train <- data[split,]
test <- data[-split,]
divorced.test <- divorced[-split]
```

Logistic on seeds 1-50

```
logit.lasso.errors <- c()
logit.lasso.best.attr <- c()
for (i in 1:50) {
  set.seed(i)
  split <- sample(1:nrow(data), nrow(data)*0.8)
  train <- data[split,]
  test <- data[-split,]
  divorced.test <- divorced[-split]

  library(glmnet)
  logit.lasso.model <- cv.glmnet(as.matrix(train[0:54]), as.matrix(train[55]), alpha=1, nfolds=5, famil
  pred.logit.lasso <- predict(logit.lasso.model, s = logit.lasso.model$lambda.min, newx =as.matrix(test
  pred.logit.lasso <- ifelse(pred.logit.lasso<0.5,"No","Yes")
  logit.lasso.errors[i] <- sum(pred.logit.lasso!=divorced.test)/length(divorced.test)

  attributes <- row.names(data.frame(coef(logit.lasso.model)[,1][coef(logit.lasso.model)[,1]!=0]))
  logit.lasso.best.attr <- append(logit.lasso.best.attr, c(attributes)[2:length(attributes)])
}
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-1
```

LDA on seeds 1-50

```
lda.fwd.errors <- c()
lda.bwd.errors <- c()

lda.fwd.best.attr <- c()
lda.bwd.best.attr <- c()
```

```
for (i in 1:50) {
  set.seed(i)
  split <- sample(1:nrow(data), nrow(data)*0.8)
  train <- data[split,]
  test <- data[-split,]
  divorced.test <- divorced[-split]
  library(leaps)
  regfit.fwd <- regsubsets(Class~.-divorced,data=train,method="forward", nvmax = 54) # fwd       select
  regfit.bwd <- regsubsets(Class~.-divorced,data=train,method="backward", nvmax = 54) # same thing for
  library(MASS)
  subset.fwd.errors <- c()
  subset.bwd.errors <- c()
  for (j in 1:54){
    formula <- as.formula (paste("Class", paste(names(coef(regfit.fwd, id=j))[2:(j+1)], collapse=' + '
    lda.fwd.model <- lda(formula, data=train)
    lda.fwd.pred <- predict(lda.fwd.model,test)
    subset.fwd.errors[j] <- sum(lda.fwd.pred$class!=test$Class)/length(divorced.test)

    formula <- as.formula (paste("Class", paste(names(coef(regfit.bwd, id=j))[2:(j+1)], collapse=' + '
    lda.bwd.model <- lda(formula, data=train)
    lda.bwd.pred <- predict(lda.bwd.model,test)
    subset.bwd.errors[j] <- sum(lda.bwd.pred$class!=test$Class)/length(divorced.test)
  }
  best.subset.fwd <- which.min(subset.fwd.errors)
  best.subset.fwd <- names(coef(regfit.fwd, id=best.subset.fwd))[2:(best.subset.fwd+1)]
  best.subset.bwd <- which.min(subset.bwd.errors)
  best.subset.bwd <- names(coef(regfit.bwd, id=best.subset.bwd))[2:(best.subset.bwd+1)]

  lda.fwd.best.attr <- append(lda.fwd.best.attr, c(best.subset.fwd))
  lda.bwd.best.attr <- append(lda.bwd.best.attr, c(best.subset.bwd))

  formula.fwd <- as.formula (paste("Class", paste(best.subset.fwd, collapse=' + ' ), sep = "~"))
  lda.fwd.model <- lda(formula.fwd, data=train)
  lda.fwd.pred <- predict(lda.fwd.model,test)
  lda.fwd.errors[i] <- sum(lda.fwd.pred$class!=test$Class)/length(divorced.test)

  formula.bwd <- as.formula (paste("Class", paste(best.subset.bwd, collapse=' + ' ), sep = "~"))
  lda.bwd.model <- lda(formula.bwd, data=train)
  lda.bwd.pred <- predict(lda.bwd.model,test)
  lda.bwd.errors[i] <- sum(lda.bwd.pred$class!=test$Class)/length(divorced.test)
}
```

Tree-based on seeds 1-50

```
tree.errors <- c()
bag.errors <- c()
rf.errors <- c()

tree.best.attr <- c()
bag.best.attr <- c()
rf.best.attr <- c()
```

```r
for(i in 1:50){
  set.seed(i)
  split <- sample(1:nrow(data), nrow(data)*0.8)
  train <- data[split,]
  test <- data[-split,]
  divorced.test <- divorced[-split]

  library(tree)

  tree.model <- tree(divorced~.-Class, data=train)
  cv.tree.model <- cv.tree(tree.model)
  prune.tree.model <- prune.tree(tree.model, best=cv.tree.model$size[which.min(cv.tree.model$dev)])
  prune.tree.pred <- predict(prune.tree.model, test, type="class")
  tree.errors[i] <- sum(prune.tree.pred!=divorced.test)/length(divorced.test)
  tree.best.attr[i] <- summary(prune.tree.model)$used[1]

  library(randomForest)
  bag.model <- randomForest(divorced~.-Class, data=train, mtry=54, importance=TRUE)
  bag.pred <- predict(bag.model, test, type="class")
  bag.errors[i] <- sum(bag.pred!=divorced.test)/length(divorced.test)
  bag.best.attr[i] <- which(importance(bag.model, type=2)==max(importance(bag.model, type=2)))

  rf.model <- randomForest(divorced~.-Class, data=train, mtry=7, importance=TRUE)
  rf.pred <- predict(rf.model, test, type="class")
  rf.errors[i] <- sum(rf.pred!=divorced.test)/length(divorced.test)
  rf.best.attr[i] <- which(importance(rf.model, type=2)==max(importance(rf.model, type=2)))
}
```

```
## randomForest 4.6-14

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

KNN on seeds 1-50

```r
knn.errors <- c()
library(FNN)
for(i in 1:50){
  set.seed(i)
  split <- sample(1:nrow(data), nrow(data)*0.8)
  train <- data[split,]
  test <- data[-split,]
  divorced.test <- as.factor(test$Class)

  err_k <- c()
  for (j in 1:85) {
    predicted = knn(train = train[1:54], test = train[1:54], cl=as.factor(train$Class), k=j)
    err_k[j] = mean(divorced.test!=predicted)
  }
  knn.errors[i] = min(err_k)
}
```

Displaying results

- Printing out means and variances of test errors across 50 seeds for each model

```
error.means <- data.frame(c(mean(logit.lasso.errors),mean(lda.fwd.errors),mean(lda.bwd.errors),mean(tre
rownames(error.means) <- c("logit", "lf", "lb", "tree","bag","rf","knn")
colnames(error.means) <- c("mean errors")
print(error.means)
```

```
##       mean errors
## logit  0.02411765
## lf     0.01882353
## lb     0.01764706
## tree   0.04294118
## bag    0.02352941
## rf     0.02352941
## knn    0.49602941
```

```
print(min(error.means))
```

```
## [1] 0.01764706
```

```
error.var <- data.frame(c(var(logit.lasso.errors),var(lda.fwd.errors),var(lda.bwd.errors),var(tree.erro
rownames(error.var) <- c("logit", "lf", "lb", "tree","bag","rf","knn")
colnames(error.var) <- c("mean error variances")

print(error.var)
```

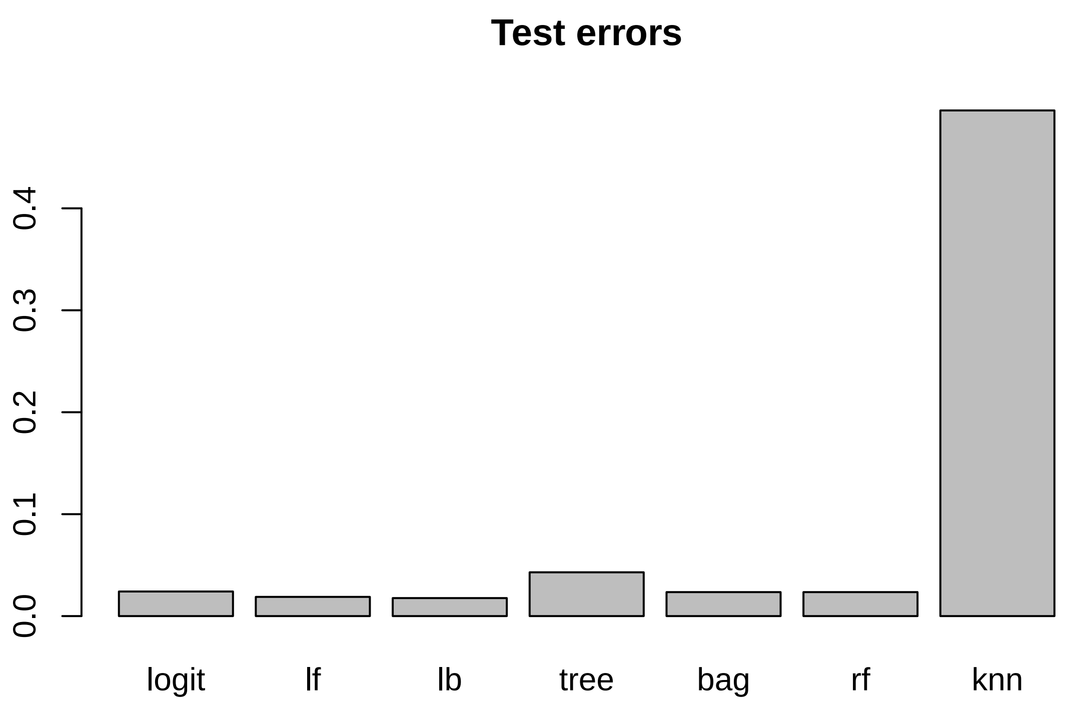
```
##       mean error variances
## logit         0.0006246028
## lf            0.0004505332
## lb            0.0004236989
## tree          0.0011019702
## bag           0.0005649319
## rf            0.0005649319
## knn           0.0017305672
```

```
print(min(error.var))
```

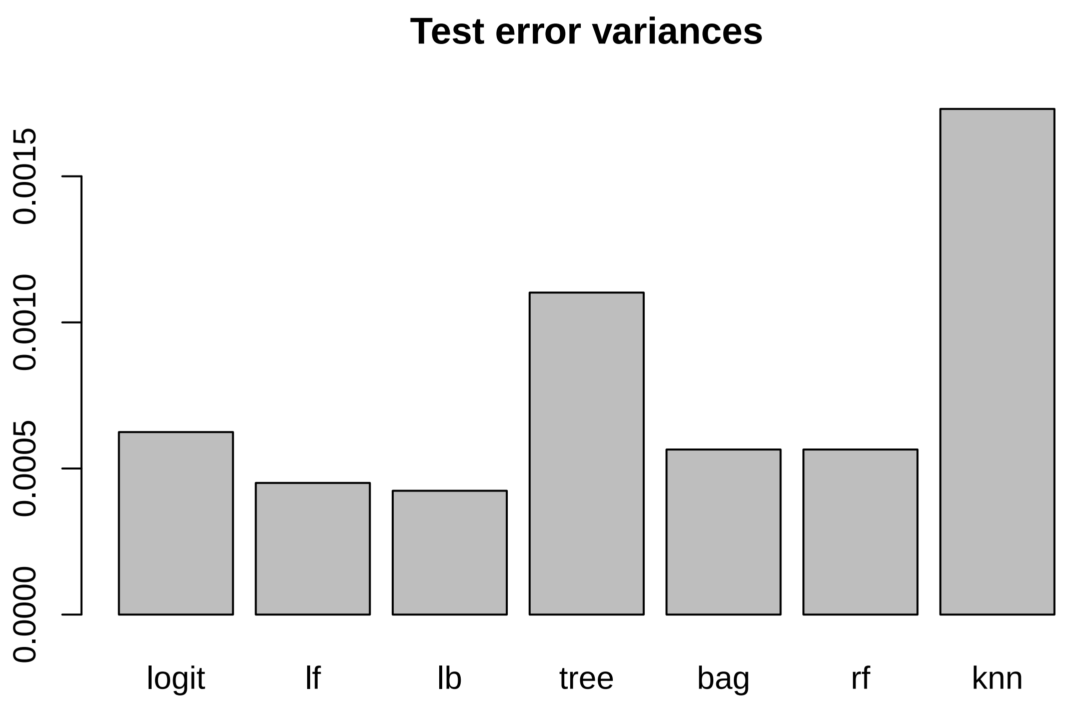
```
## [1] 0.0004236989
```

- Printing out plots

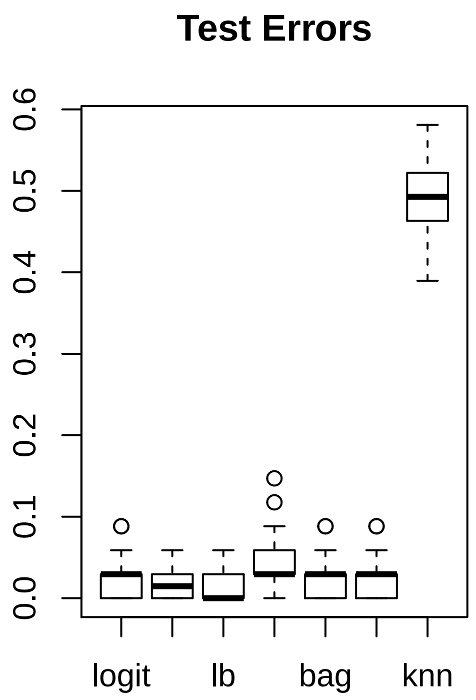
```
barplot(c(mean(logit.lasso.errors), mean(lda.fwd.errors), mean(lda.bwd.errors), mean(tree.errors), mean
```



```r
barplot(c(var(logit.lasso.errors), var(lda.fwd.errors), var(lda.bwd.errors), var(tree.errors), var(bag.
```

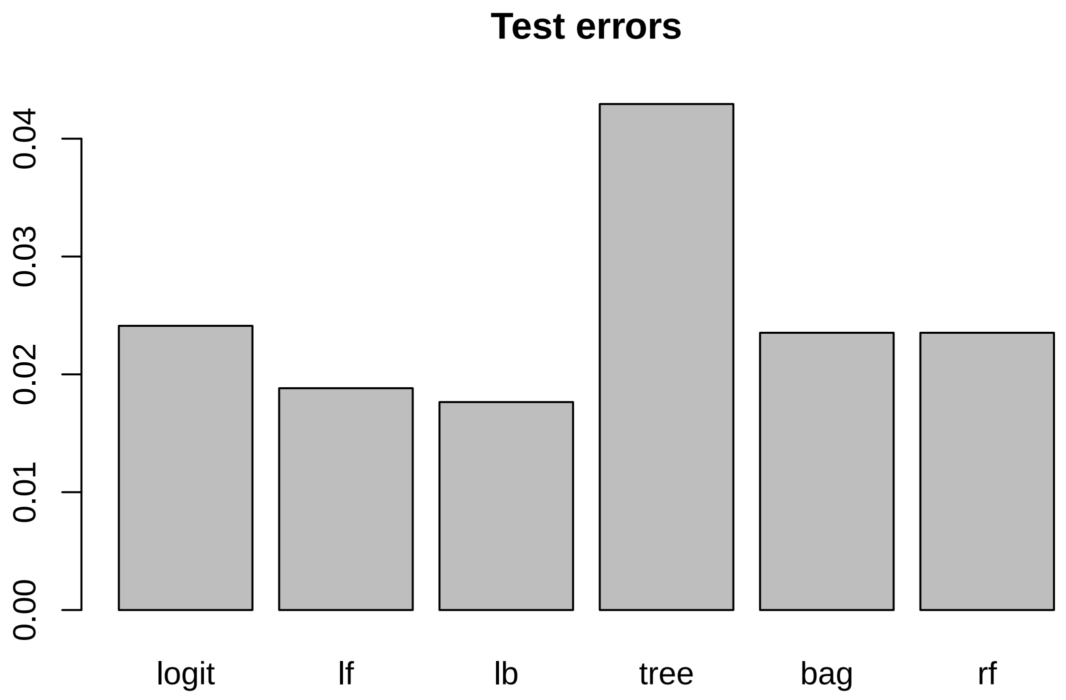


```r
par(mfrow = c(1,2))
boxplot(logit.lasso.errors, lda.fwd.errors, lda.bwd.errors, tree.errors,bag.errors,rf.errors, knn.error
```

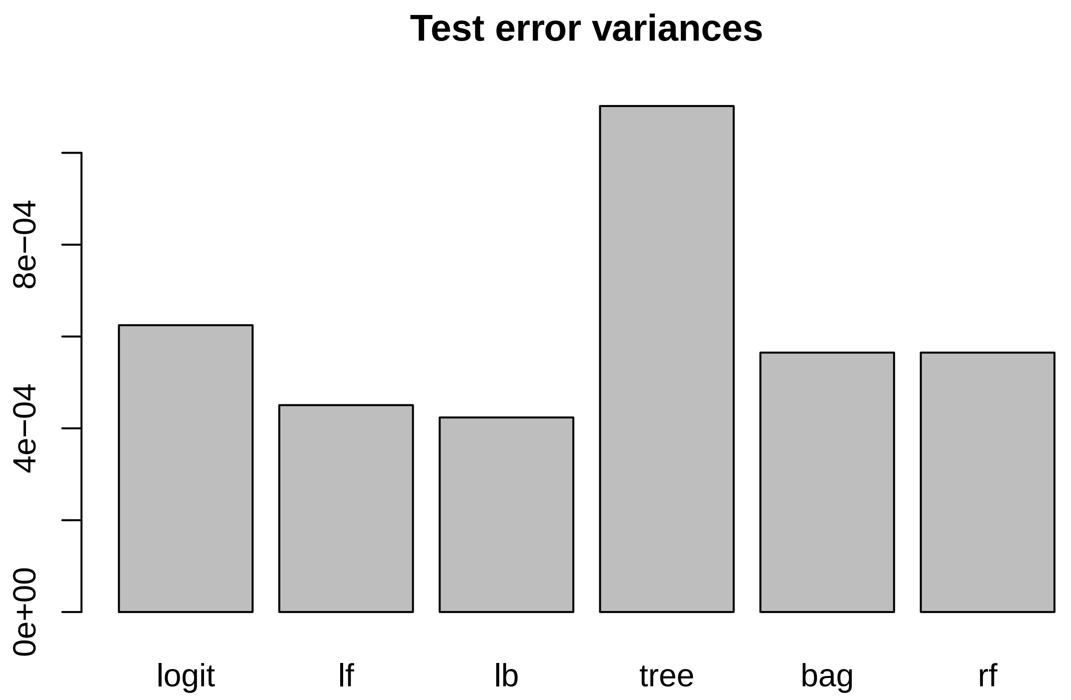


- Plots with KNN results removed

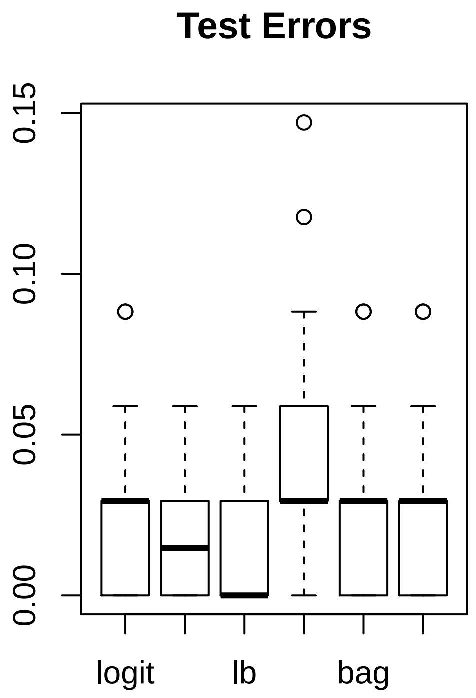
```
barplot(c(mean(logit.lasso.errors), mean(lda.fwd.errors), mean(lda.bwd.errors), mean(tree.errors), mean
```



```
barplot(c(var(logit.lasso.errors), var(lda.fwd.errors), var(lda.bwd.errors), var(tree.errors), var(bag.
```



```r
par(mfrow = c(1,2))
boxplot(logit.lasso.errors, lda.fwd.errors, lda.bwd.errors, tree.errors,bag.errors,rf.errors, names=c("
```



- Analyzing most important predictors

```r
sort(table(logit.lasso.best.attr),decreasing=TRUE)[1:5]
```

```
## logit.lasso.best.attr
## Atr40 Atr18 Atr17 Atr19 Atr26
##    50    45    42    35    32
```

```r
sort(table(lda.fwd.best.attr),decreasing=TRUE)[1:5]
```

```
## lda.fwd.best.attr
## Atr40 Atr18 Atr17  Atr6 Atr19
##    44    20    13    12     8
```

```
sort(table(lda.bwd.best.attr),decreasing=TRUE)[1:5]
```

```
## lda.bwd.best.attr
## Atr40  Atr6 Atr26 Atr18 Atr17
##    43    13    10     9     8
```

```
sort(table(tree.best.attr),decreasing=TRUE)[1:5]
```

```
## tree.best.attr
## 19 12 18 10 13
## 29 10  6  2  1
```

```
sort(table(bag.best.attr),decreasing=TRUE)[1:5]
```

```
## bag.best.attr
## 18 40 19 17 26
## 22 12  8  5  2
```

```
sort(table(rf.best.attr),decreasing=TRUE)[1:5]
```

```
## rf.best.attr
## 18 40 19 17 11
## 24  8  6  5  3
```