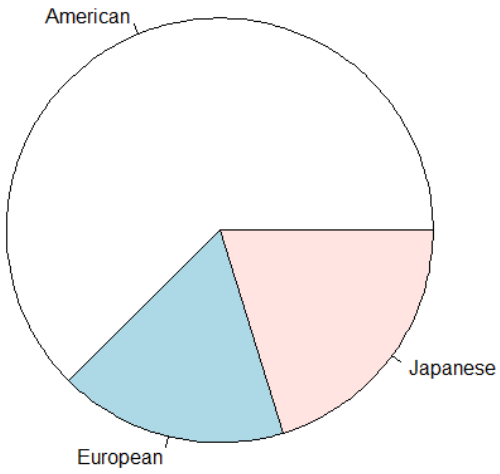
ISYE 7406
HW 5

Introduction:
This analysis seeks to compare the performance of a random forest, boost, tree, LDA, and KNN model in classifying the country of origin for cars in the Auto dataset. This data was taken from the Statlib Library at Carnegie Mellon University. This cleaned dataset has 392 rows and 7 independent variables. In this analysis, origin is our response variable, and our predictor variables are mpg, displacement, weight, cylinders, horsepower, acceleration, and year. Displacement represents the volume of the cylinders in the engine. The models will be trained and tuned using the training data set. The models will then be compared using 500 iterations of Monte Carlo cross validation on the training and testing sets.

Exploratory Data Analysis:
Origin was chosen to be our response variable in this analysis. Origin represents the country of origin, with 1=American, 2=European, and 3=Japanese. Looking at our response variable, the majority of cars are of American origin (245) with the remaining fraction split between European (68) and Japanese (73) origin. This is significant, and the unevenness of our response will factor into our choice to use “information” for our splitting criteria for our tree model.

Breakdown of origin (response variable)



Next we calculate the covariance of each independent variable with our response. We see changes in displacement as the leading cause of change in origin and we will use that as part of our models.

Correlation of independent variables with origin

Variable	Covariance w/ origin
displacement	**-0.61**
weight	-0.59
cylinders	-0.57
mpg	0.57
horsepower	-0.46
year	0.18
acceleration	0.21

Before selecting other independent variables I mapped out the top 20 multicollinearity between independent variables, as seen below. All of the most predictive variables are correlated with displacement, so we opt not to use weight, cylinders, mpg, or horsepower. While acceleration and year do not have the highest correlation with our response variable, they have a tolerable level of multicollinearity with displacement so we opt to include them in our models.

Multicollinearity of independent variables (top 20)

Var1	Var2	Correlation
displacement	cylinders	0.95
weight	displacement	0.93
weight	cylinders	0.90
horsepower	displacement	0.90
weight	horsepower	0.86
horsepower	cylinders	0.84
weight	mpg	-0.83
displacement	mpg	-0.81
horsepower	mpg	-0.78
cylinders	mpg	-0.78
acceleration	horsepower	-0.69
year	mpg	0.58

acceleration	displacement	-0.54
acceleration	cylinders	-0.50
acceleration	mpg	0.42
acceleration	weight	-0.42
year	horsepower	-0.42
year	displacement	-0.37
year	cylinders	-0.35
year	weight	-0.31

Method:
In this section I will provide a brief explanation of each model, their optimal values chosen, and the comparison that was run with all the models. For each model we used the independent variables of displacement, acceleration, and year.

- **Random Forest:** This model is an ensemble method of a decision tree applying bagging. In addition to bagging, this approach uses random feature selection to avoid a dominant feature skewing the results. In my model I found the optimal number of trees to be 173 by testing values from 100 to 550. Looking at the feature importance of all features, I found displacement to be the most important, which is in line with our earlier findings.
- **Boosting (gbm):** This is another ensemble approach that combines the predictions of multiple weak learners to make an accurate prediction. Models iterate over the data with residuals being weighted, in this way the errors of a previous model are corrected by the future iterations. I found the optimal interaction depth to be 2 by testing values from 1 to 10 and shrinkage to be .04 by testing .01 to .1.
- **Decision Tree:** This model seeks to split the data to reduce entropy and increase purity of the dataset. Decision trees are a greedy algorithm that looks for the largest improvement per split. Trees are also trimmed back via pruning to avoid overfitting. I used the information splitting criteria rather than Gini as it is better suited to unbalanced responses.
- **LDA:** This classification model seeks a combination of features that split maximizes separation between classes and reduces dimensionality. Uses Fisher's criterion to find direction that optimally separates classes.
- **KNN:** Our last model uses k points near the new point based on distance to predict the class of the new point. We found the optimal k value to be 2, however 38 was very close, and less likely to be skewed by outliers, so we opted to use that.

Once our models were built, they were run on 500 iterations of Monte Carlo cross validation to measure their testing error. The results were averaged and their variance was also taken. The results are below.

Results:

Model	Test Error	Variance
RF	0.18	0.0010
Boost	0.21	0.0012
Tree	0.23	0.0016
KNN	0.28	0.0014
LDA	0.30	0.0011

Findings:
Based on our 500 iterations of Monte Carlo cross validation. We found the Random Forest model to perform the best in terms of test error as well as variance. Not only does this model make the most accurate predictions, but also does so most consistently. Our other ensemble method was the next best performance in terms of testing error and third in terms of variance. It seems that ensemble methods in general are more reliable than single models.

One thing to consider is the ability to fit the data so well in ensemble methods will likely lead to overfitting. In scenarios where less is known about the data or there is more variance in samples an individual model may be a better approach than an ensemble.

Appendix:

```
#install.packages('randomForest')
#install.packages('gbm')
#install.packages('rpart')
library(gbm)
library(randomForest)
library(rpart)
library(MASS)
library(class)

#load data
data<-read.csv('Auto.csv')
head(data)
dim(data)
data

pie(table(data$origin),labels=c("American","European","Japanese"))
table(data$origin)
```

```
#EDA
cor_vals<-c(cor(data$mpg,data$origin),
            cor(data$cylinders,data$origin),
            cor(data$displacement,data$origin),
            cor(data$horsepower,data$origin),
            cor(data$weight,data$origin),
            cor(data$acceleration,data$origin),
            cor(data$year,data$origin)
)

colnames<-(colnames(data))
covariance_response<-cbind(colnames[1:7],cor_vals)
covariance_response


corr<-cor(data[,-8])
corr
auto_vec <- as.vector(corr)
auto_vec[!lower.tri(corr)] <- NA
indices <- order(abs(auto_vec), decreasing=TRUE,na.last = NA)
sorted_cor <- auto_vec[indices]
row_indices <- row(corr)[indices]
col_indices <- col(corr)[indices]
cor_pairs <- data.frame(
  Var1 = rownames(corr)[row_indices[0:20]],
  Var2 = colnames(corr)[col_indices[0:20]],
  Correlation = sorted_cor[0:20]
)
cor_pairs

displacement+acceleration+year
#set test data set to 30% due to small size of data set
set.seed(7406)
flag <- sort(sample(392,118, replace = FALSE))
autotrain <- data[-flag,]
autotest <- data[flag,]

y1<-autotrain$origin
y2<-autotest$origin

#random forest
rforestout<-NULL
for (i in 101:550){
```

```
rfmodel<-randomForest(as.factor(origin) ~displacement+acceleration+year, data=autotrain,
type='class',importance=TRUE,ntree=i)
testpred<-predict(rfmodel, autotest, type='class')
rf_class_error<-mean(testpred!=y2)
rforestout<-c(rforestout,rf_class_error)
}
which.min(rforestout)
plot(rforestout,ylab="Categorization Error",xlab="Number Trees")
#70
rfmodel2<-randomForest(as.factor(origin) ~., data=autotrain,
type='class',importance=TRUE,ntree=178)
importance(rfmodel2)
varImpPlot(rfmodel2)

#randomForest(as.factor(origin)~displacement+weight+mpg, data=autotrain,
type='class',importance=TRUE,ntree=178)

#boost model
gbmmodel <- gbm(as.factor(origin) ~displacement+acceleration+year,data=autotrain,
distribution = 'multinomial',cv=5)
gbmperf<-gbm.perf(gbmmodel, method="cv")
summary(gbmperf)

gbmpredict<-predict(gbmmodel,newdata = autotest, n.trees=gbmperf, type="response",cv=5)
gbmpredict
out<-NULL
for (i in 1:nrow(gbmpredict)){
  row<-which.max(gbmpredict[i,,])
  out<-c(out,row)
}
out

b_class_error<-mean(out!=y2)
b_class_error

#interaction depth (2)
interaction_depth_test<-NULL
for (i in 1:10){
gbmmodel2<-gbm(as.factor(origin) ~displacement+acceleration+year,data=autotrain,
distribution = 'multinomial',cv=5,interaction.depth=i)
gbmpredicttemp<-predict(gbmmodel2,newdata = autotest, n.trees=gbmperf,
type="response",cv=5)
out<-NULL
for (i in 1:nrow(gbmpredicttemp)){
```

```
  row<-which.max(gbmpredicttemp[i,,])
  out<-c(out,row)
}
b_class_error_temp<-mean(out!=y2)
interaction_depth_test<-c(interaction_depth_test,b_class_error_temp)
}
interaction_depth_test

#shrinkage
shrinkage_test<-NULL
for (i in seq(.01,.1,.01)){
gbmmodel2<-gbm(as.factor(origin) ~displacement+acceleration+year,data=autotrain,
distribution = 'multinomial',cv=5,interaction.depth=2,shrinkage=i)
gbmpredicttemp<-predict(gbmmodel2,newdata = autotest, n.trees=gbmperf,
type="response",cv=5)
out<-NULL
for (i in 1:nrow(gbmpredicttemp)){
  row<-which.max(gbmpredicttemp[i,,])
  out<-c(out,row)
}
b_class_error_temp<-mean(out!=y2)
shrinkage_test<-c(shrinkage_test,b_class_error_temp)
}
shrinkage_test

#gbm(as.factor(origin) ~displacement+acceleration+year,data=autotrain, distribution =
'multinomial',cv=5,interaction.depth=2,shrinkage=.04)

#tree
rpartmodel <- rpart(as.factor(origin) ~displacement+acceleration+year,data=autotrain,
method="class", parms=list(split="information"))
plot(rpartmodel,compress=TRUE)
text(rpartmodel)

rpartpredict<-predict(rpartmodel, autotest,type="class")
tree_error<-mean(rpartpredict!=y2)
tree_error

plotcp(rpartmodel)
print(rpartmodel$cptable)

optcp <- which.min(rpartmodel$cptable[, "xerror"]);
cp1 <- rpartmodel$cptable[optcp, "CP"]
cp1
```

```
rpartprune <- prune(rpartmodel,cp=cp1)

rpartpredict2<-predict(rpartprune, autotest,type="class")
tree_error2<-mean(rpartpredict2!=y2)
tree_error2


#LDA
ldamodel <- lda(as.factor(origin)~displacement+acceleration+year,data=autotrain)
ldapredict <- predict(ldamodel, autotest[,1:7])$class
lda_class_error<-mean(ldapredict!=y2)
lda_class_error

#knn
knnout<-NULL
for (i in 1:99){
knnpred<-knn(autotrain[,c('displacement','acceleration','year')],autotest[,c('displacement','acceler
ation','year')],autotrain[,8],k=i)
knn_class_error<-mean(knnpred!=y2)
knnout<-c(knnout,knn_class_error)
}
which.min(knnout)
plot(knnout)

#knn(autotrain[,c('displacement','acceleration','year')],autotest[,c('displacement','acceleration','ye
ar')],autotrain[,8],k=38)

#CV model test
finalout<-NULL
for (i in 1:500){
  flag2 <- sort(sample(392,118, replace = FALSE))
  autotrain2 <- data[-flag2,]
  autotest2 <- data[flag2,]
  y12<-autotrain2$origin
  y22<-autotest2$origin

  #random forrest
  rffinal<-randomForest(as.factor(origin)~displacement+weight+mpg, data=autotrain2,
type='class',importance=TRUE,ntree=178)
  testpred2<-predict(rffinal, autotest2, type='class')
  te0<-mean(testpred2!=y22)

  #gbm
```

```
  gbmfinal<-gbm(as.factor(origin) ~displacement+acceleration+year,data=autotrain2, distribution
= 'multinomial',cv=5,interaction.depth=2,shrinkage=.04)
  gbmpredict2<-predict(gbmfinal,newdata = autotest2, n.trees=90, type="response",cv=5)
  gbmout<-NULL
  for (j in 1:nrow(gbmpredict2)){
    row<-which.max(gbmpredict2[j,,])
    gbmout<-c(gbmout,row)
  }
  te1<-mean(gbmout!=y22)

  #tree
  rpartmodel2 <- rpart(as.factor(origin) ~displacement+acceleration+year,data=autotrain2,
method="class", parms=list(split="information"))
  optcp2 <- which.min(rpartmodel2$cptable[, "xerror"]);
  cp2 <- rpartmodel2$cptable[optcp2, "CP"]
  rpartprune2 <- prune(rpartmodel2,cp=cp2)
  rpartpredict_final<-predict(rpartprune2, autotest2,type="class")
  te2<-mean(rpartpredict_final!=y22)

  #Lda
  ldamodel2 <- lda(as.factor(origin)~displacement+acceleration+year,data=autotrain2)
  ldapredict2 <- predict(ldamodel2, autotest2[,c("displacement", "acceleration", "year")])$class
  te3<-mean(ldapredict2!=y22)

  #knn

knnpred2<-knn(autotrain2[,c('displacement','acceleration','year')],autotest2[,c('displacement','acc
eleration','year')],autotrain2[,8],k=38)
  te4<-mean(knnpred2!=y22)

  finalout<-rbind(finalout,c(te0,te1,te2,te3,te4))
}
finalout

apply(finalout,2, mean)
apply(finalout,2, var)
```