Classification, Decision Trees and Random Forests

Setup

```
library(dplyr)
library(nnet)
library(MASS)
library(tree)
library(randomForest)
library(ggplot2)
```

We will work with the `pierce_county_house_sales` data that was referenced in the exam. These data are avaialble here: GitHub link. Our goal will be to answer the obvious (and difficult) question: What makes a house cost more or less?

Once you have the data downloaded and saved in the folder you want, they can be loaded with an appropriate modification of the following command. I put the file in a subdirectory of my working directory called `data`. If you do something different, edit the following command appropriately.

```
load("data/pierce_county_house_sales.rda")
```

Take a look at the data.

```
View(pierce_county_house_sales)
summary(pierce_county_house_sales)
```

The data need a little preparation for our purposes. First, we want to make the *character* variables into *factor* variables. Many modeling tools do this automatically, but not all. We could change each variable one at a time, but the `mutate` and `across` commands allow us to adjust all seven of these variables at once.

```
pierce_county_house_sales <- pierce_county_house_sales %>%
  mutate(
    across(c("hvac_description",  "exterior", "interior", "roof_cover",
           "waterfront_type", "view_quality", "utility_sewer"),
         as.factor))
```

Second, we want a categorical response variable to practice our classification tools. We'll build one out of `sale_price`.

```
pierce_county_house_sales$price_category <- cut(pierce_county_house_sales$sale_price,
                      breaks = c(0, 300000, 400000, 500000, 1000000, Inf))
```

Run `summary` again and notice what has changed.

(1) Does the `price_category` variable seem to capture essential information about `sale_price`? In what ways might it be preferable to use this categorical variable instead of the numeric one? In what ways is the numeric response preferable?

Classification

Now we want to use the other variables to try to determine the price category for one of these houses. We'll avoid using `sale_price` since that would be cheating. The `sale_date` variable causes problems for some of our tools so we'll sometimes avoid that as well.

Multinomial Regression

Let's try multinomial regression first. The following code builds the model,

```
mlr.house <- multinom(price_category~.-sale_price, data=pierce_county_house_sales)
```

computes the predicted values from the model,

```
mlr.pred <- predict(mlr.house)
```

and then builds a table, called a **confusion matrix**, that compares the predicted category to the actual category.

```
table(mlr.pred, pierce_county_house_sales$price_category)
```

```
##
## mlr.pred        (0,3e+05] (3e+05,4e+05] (4e+05,5e+05] (5e+05,1e+06] (1e+06,Inf]
##   (0,3e+05]           751           399            78            61           6
##   (3e+05,4e+05]      1205          3849          1340           250          11
##   (4e+05,5e+05]       169           907          2184           861           4
##   (5e+05,1e+06]       133           176           854          3183         163
##   (1e+06,Inf]           0             1             4            66         159
```

To assess the accuracy, we can add the entries on the diagonal, and see what fraction of the total houses were correctly categorized.

```
mean(mlr.pred==pierce_county_house_sales$price_category)
```

```
## [1] 0.6022362
```

(2) What percentage of houses in the $\$300K - \$400K$ range were correctly categorized?

(3) Some of the houses that sold for over a million dollars were seriously miscategorized. Can you find any reasons why this might be the case? (This will require some digging into the data.)

Discriminant analysis

We can run linear discriminant analysis and see how it does with these data. As above, we build a model, compute predictions, and then compute the percentage of correct predictions.

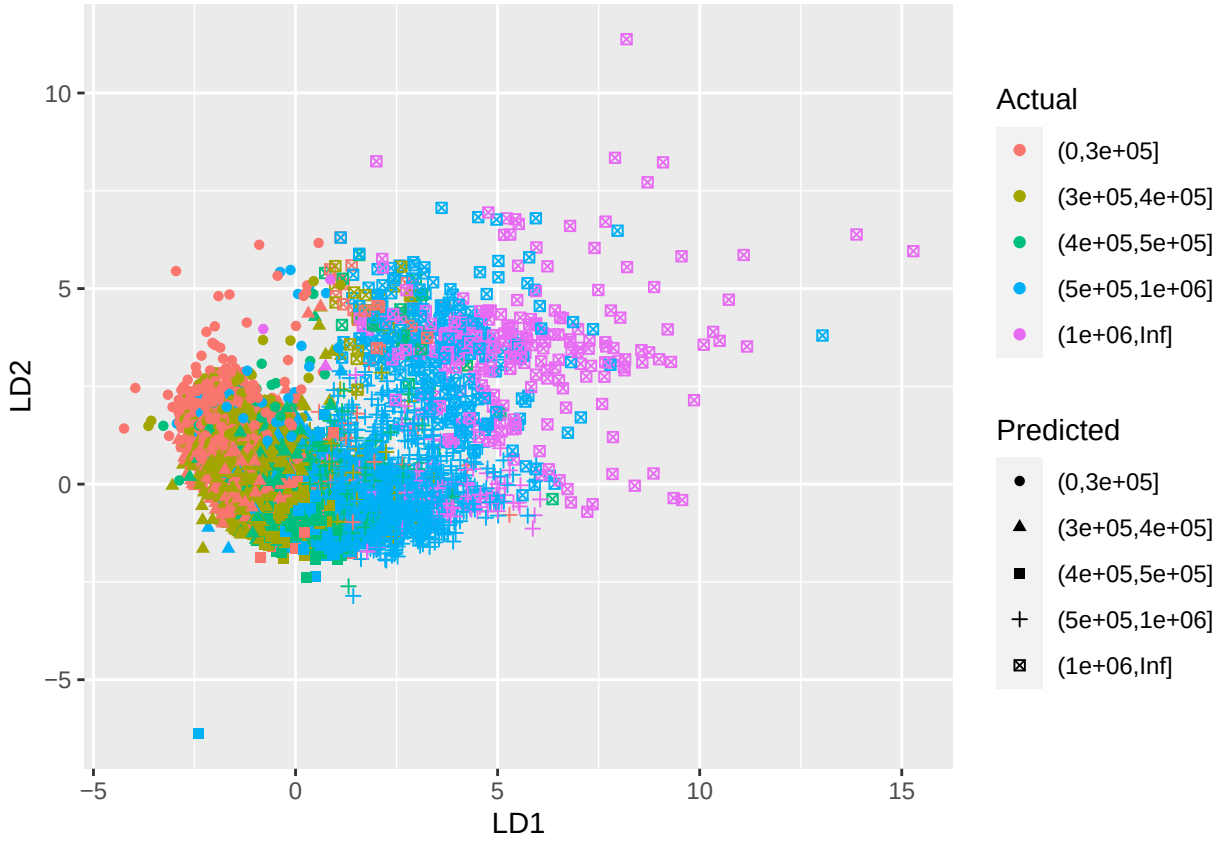
```
lda.house <- lda(price_category~.-sale_price, data = pierce_county_house_sales)
lda.pred <- predict(lda.house)
mean(lda.pred$class==pierce_county_house_sales$price_category)
```

```
## [1] 0.5834424
```

One nice thing about linear discriminant analysis is that the discriminants define a basis for a vector space that should do a reasonably good job of showing the difference between the categories. We'll talk more about this when we discuss Principal Components Analysis and other related tools. For now, we'll just use the output.

```
lda.obs <- as.data.frame(lda.pred$x)
ggplot(lda.obs,
       aes(LD1, LD2,
```

```
          color=pierce_county_house_sales$price_category,
          shape=lda.pred$class)) +
  geom_point()+
  labs(shape="Predicted", color="Actual")
```



Trees for classification

Trees are nice and intuitive. Let's build one.

```
tree.house <- tree(price_category~.-sale_price-sale_date, pierce_county_house_sales)
summary(tree.house)
```

```
##
## Classification tree:
## tree(formula = price_category ~ . - sale_price - sale_date, data = pierce_county_house_sales)
## Variables actually used in tree construction:
## [1] "house_square_feet"    "bathrooms"            "basement_square_feet"
## Number of terminal nodes:  7
## Residual mean deviance:  2.188 = 36770 / 16810
## Misclassification error rate: 0.4767 = 8015 / 16814
```

The following commands draw and label the tree.

```
plot(tree.house)
text(tree.house)
```

More information about the tree, including probabilities that an observation is in a particular price category if it is in a particular node, can be read from the output of the following command.

```
tree.house # Note the legend at the top of the output.
```

As with any model you build, the predict function calculates model predictions. Setting `type="class"` gives predicted categories that can be compared to reality, as we did above.

```
tree.pred <- predict(tree.house, type="class")
mean(tree.pred==pierce_county_house_sales$price_category)
```

```
## [1] 0.5233139
```

(4) According to classification accuracy, how do the three models we have seen so far compare?

Cross validation

A problem with everything we have done so far to build and evaluate our models is that we are computing **training error**. It is better for assessing the predictive quality of a model to see how it does on data that were not used in its construction. This is what we call **test error**.

To evaluate test error, we need to hold back some of our data and not use it to build the model. The following code pulls out 30% of the data to be used for testing and uses the remaining 70% for training.

```
set.seed(2) # Before doing anything random, set a seed
train <- sample(1:nrow(pierce_county_house_sales),
                0.7*nrow(pierce_county_house_sales))
houses.test <- pierce_county_house_sales[-train,]
tree.house <- tree(price_category~.-sale_price-sale_date,
                   data=pierce_county_house_sales, subset=train)
plot(tree.house)
text(tree.house)
tree.pred <- predict(tree.house, newdata=houses.test, type="class")
confusion_matrix <- table(tree.pred,houses.test$price_category)
confusion_matrix
mean(tree.pred==houses.test$price_category)
```

(5) Rebuild the other two models above and compute the test error instead of the training error. Does the ranking change?

Bagging and Random Forests

Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.

Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2 , the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n. In other words, averaging a set of observations reduces variance. Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set.

In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. If the response is numeric, we then average all the predictions to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f^{*b}(x)$$

If the response is categorical, then $\hat{f}_{bag}(x)$ is the most commonly occurring class among the $\hat{f}^{*b}(x)$. This is called **bagging**.

Random Forest using all predictors is bagging.

There are two tricks to the Random Forest algorithm. Bagging is one of them.

```
bag.house <- randomForest(price_category~.-sale_price,
                          data=pierce_county_house_sales,
                        subset=train,mtry=18,importance=TRUE)
bag.house
bag.pred <- predict(bag.house,newdata = houses.test)
mean(bag.pred==houses.test$sale_price)
```

Letting the less obvious predictors have their turn

The other trick to a Random Forest is that at each step, not all of the predictors are considered. The model is built by randomly ignoring all but a subset of the predictors every time a split is considered.

```
rf.house <- randomForest(price_category~.-sale_price,data=pierce_county_house_sales,
                        subset=train,importance=TRUE)
rf.pred <- predict(rf.house,newdata=houses.test)
mean(rf.pred==houses.test$price_category)
```

Interpreting a Random Forest

The Random Forest algorithm produces lots of trees, and it can be hard to interpret the result. One thing we can do is look at how often a variable was used in building a tree and how much cumulative effect at improving the error each predictor had.

```
importance(rf.house)
varImpPlot(rf.house)
```

(6) Using only the top 3-5 variables that the Random Forest lists as important and perhaps some of their interactions, build a multinomial logistic regression model to predict `price_category`, or if you prefer, a linear model to predict `sale_price`. What conclusions do you draw about our original question - what factors most influence the price of a house, and how?