Notebook 2: Classification

Ruben Mathew

Intro

In this notebook we look at a few different classification methods and how they can yield different levels of accuracy. The data we use is housing information (https://www.kaggle.com/datasets/syuzai/perth-house-prices) from Perth, Australia.

Set-up

We start by resetting the environment and reading in the Perth housing information via a csv file. We clean up the data a bit by fixing the 'NULL' values to their correct counterparts, and including only complete cases, reducing the amount of observations from 33656 to 20692 (still a sizeable amount).

```
rm(list = ls()) # Reset Environment
df <- read.csv("perth.csv")
df$GARAGE[df$GARAGE == 'NULL'] <- 0
df$GARAGE <- as.integer(df$GARAGE)
df$BUILD_YEAR[df$BUILD_YEAR == 'NULL'] <- NA
df <- df[complete.cases(df), ]
df$SUBURB <- factor(df$SUBURB)
```

Train/Test Partitions

Next we separate the data into train and test partitions (80/20) in order to make our models. This leads to a test set of approx 4.1k observations and train set of approx 16.5k observations.

```
set.seed(4829)
i <- sample(1:nrow(df), .8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

Now we run some data exploration methods and techniques on the train data. This will allow us to be more familiar with content, range, and expected values of each feature and get an idea of what might be an interesting model to create.

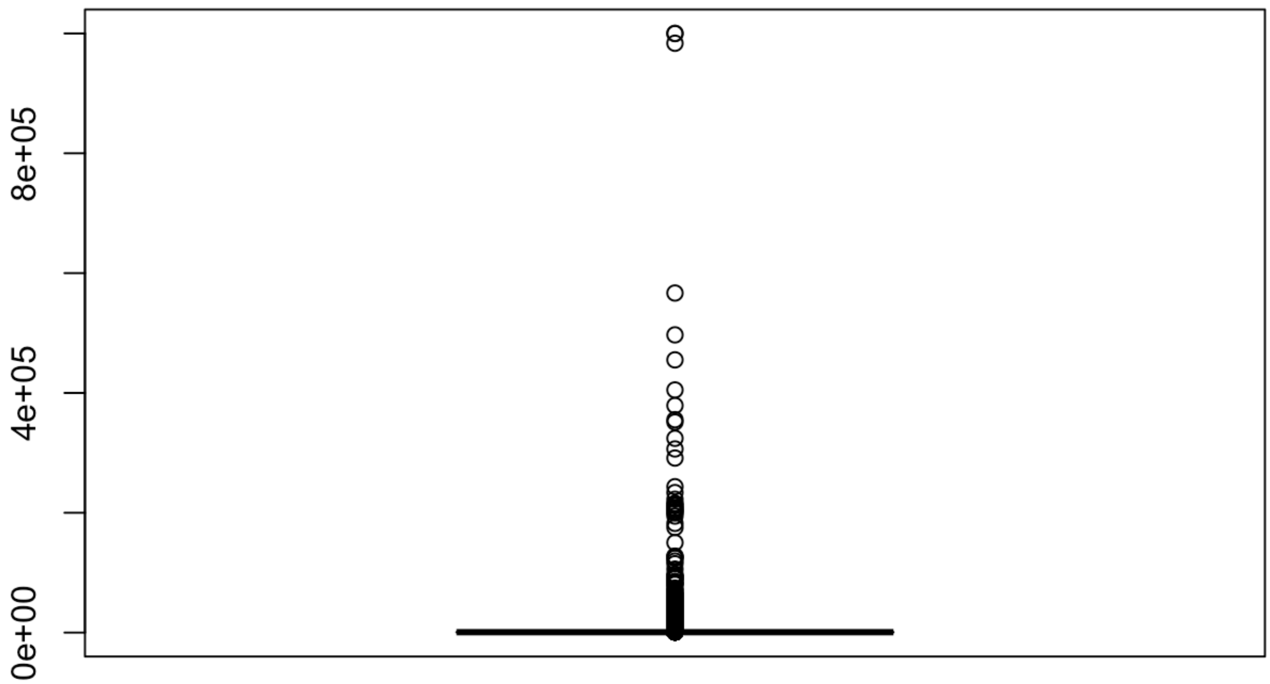
```
str(train)
```

```
## 'data.frame':	16553 obs. of  19 variables:
##  $ ADDRESS         : chr  "24 Diosma Way" "30 Hamilton Street" "6/240 Burke Drive" "5/3 Rockingham Road" ...
##  $ SUBURB          : Factor w/ 278 levels "Alexander Heights",..: 46 82 9 106 258 104 116 236 244 90 ...
##  $ PRICE           : int  575000 1720000 840000 505000 610000 477000 1575000 680000 505000 870000 ...
##  $ BEDROOMS        : int  4 4 3 2 4 3 4 4 4 3 ...
##  $ BATHROOMS       : int  2 2 2 2 2 1 2 2 2 2 ...
##  $ GARAGE          : int  2 2 2 1 2 1 1 2 2 2 ...
##  $ LAND_AREA       : int  600 1127 297 930 744 466 711 603 576 481 ...
##  $ FLOOR_AREA      : int  178 176 133 108 185 104 362 247 270 145 ...
##  $ BUILD_YEAR      : chr  "2004" "1915" "1987" "2009" ...
##  $ CBD_DIST        : int  15000 13200 9600 16900 14100 9900 19900 18700 26600 5700 ...
##  $ NEAREST_STN     : chr  "Thornlie Station" "North Fremantle Station" "Bull Creek Station" "Fremantle Station" ...
##  $ NEAREST_STN_DIST: int  2900 2100 4200 3400 1000 1700 4800 6600 4600 2000 ...
##  $ DATE_SOLD       : chr  "08-2020\n" "12-2017\n" "06-2017\n" "04-2020\n" ...
##  $ POSTCODE        : int  6155 6158 6156 6163 6024 6018 6025 6110 6065 6014 ...
##  $ LATITUDE        : num  -32.1 -32 -32 -32.1 -31.8 ...
##  $ LONGITUDE       : num  116 116 116 116 116 ...
##  $ NEAREST_SCH     : chr  "CANNING VALE COLLEGE" "JOHN CURTIN COLLEGE OF THE ARTS" "APPLECROSS SENIOR HIGH SCHOOL" "FREMANTLE COLLEGE" ...
##  $ NEAREST_SCH_DIST: num  1.95 1.36 1.97 1.34 1.33 ...
##  $ NEAREST_SCH_RANK: int  68 25 34 128 86 135 58 38 92 14 ...
```

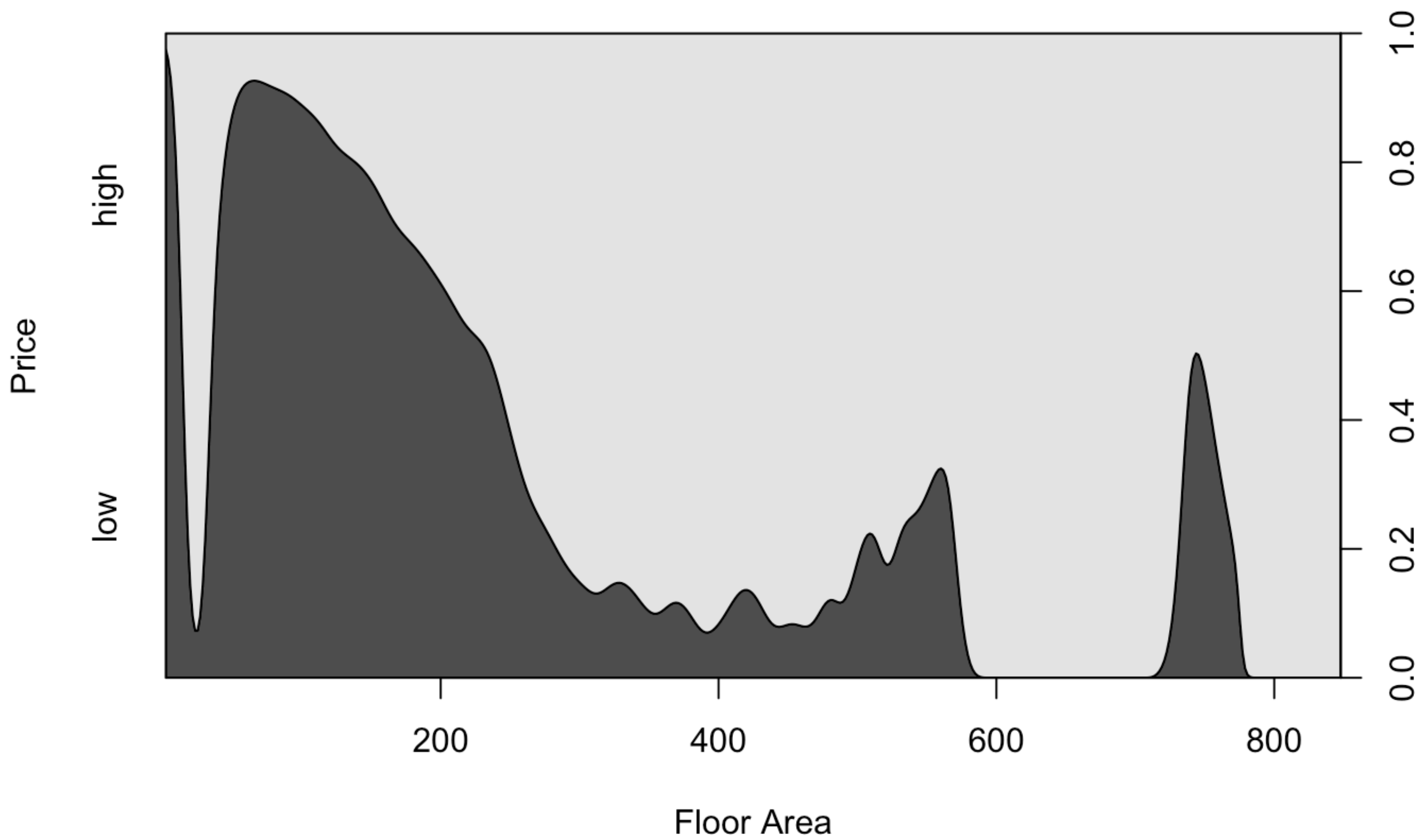
```
summary(train)
```

```
##    ADDRESS                  SUBURB          PRICE           BEDROOMS
##  Length:16553       Henley Brook:  169   Min.   :  52000   Min.   : 1.000
##  Class :character   Iluka       :  163   1st Qu.: 432500   1st Qu.: 3.000
##  Mode  :character   Darch       :  158   Median : 580000   Median : 4.000
##                     Butler      :  152   Mean   : 693204   Mean   : 3.654
##                     Huntingdale :  152   3rd Qu.: 845000   3rd Qu.: 4.000
##                     Gwelup      :  145   Max.   :2440000   Max.   :10.000
##                     (Other)     :15614
##    BATHROOMS        GARAGE          LAND_AREA        FLOOR_AREA
##  Min.   :1.000   Min.   : 0.000   Min.   :    61   Min.   :  1.0
##  1st Qu.:1.000   1st Qu.: 2.000   1st Qu.:   495   1st Qu.:133.0
##  Median :2.000   Median : 2.000   Median :   680   Median :176.0
##  Mean   :1.836   Mean   : 2.018   Mean   :  2612   Mean   :186.8
##  3rd Qu.:2.000   3rd Qu.: 2.000   3rd Qu.:   810   3rd Qu.:227.0
##  Max.   :7.000   Max.   :50.000   Max.   :999999   Max.   :849.0
##
##   BUILD_YEAR           CBD_DIST     NEAREST_STN        NEAREST_STN_DIST
##  Length:16553       Min.   : 1300   Length:16553       Min.   :   46
##  Class :character   1st Qu.:10200   Class :character   1st Qu.: 1600
##  Mode  :character   Median :15900   Mode  :character   Median : 3000
##                     Mean   :18413                      Mean   : 4227
##                     3rd Qu.:24400                      3rd Qu.: 5200
##                     Max.   :56900                      Max.   :34300
##
##   DATE_SOLD            POSTCODE       LATITUDE        LONGITUDE
##  Length:16553       Min.   :6003   Min.   :-32.46   Min.   :115.7
##  Class :character   1st Qu.:6030   1st Qu.:-32.05   1st Qu.:115.8
##  Mode  :character   Median :6066   Median :-31.94   Median :115.8
##                     Mean   :6087   Mean   :-31.95   Mean   :115.9
##                     3rd Qu.:6150   3rd Qu.:-31.82   3rd Qu.:115.9
##                     Max.   :6558   Max.   :-31.60   Max.   :116.3
##
##  NEAREST_SCH        NEAREST_SCH_DIST   NEAREST_SCH_RANK
##  Length:16553       Min.   : 0.07091   Min.   :  1.00
##  Class :character   1st Qu.: 0.86122   1st Qu.: 39.00
##  Mode  :character   Median : 1.30686   Median : 65.00
##                     Mean   : 1.70255   Mean   : 72.36
##                     3rd Qu.: 1.96873   3rd Qu.:105.00
##                     Max.   :23.25437   Max.   :139.00
##
```

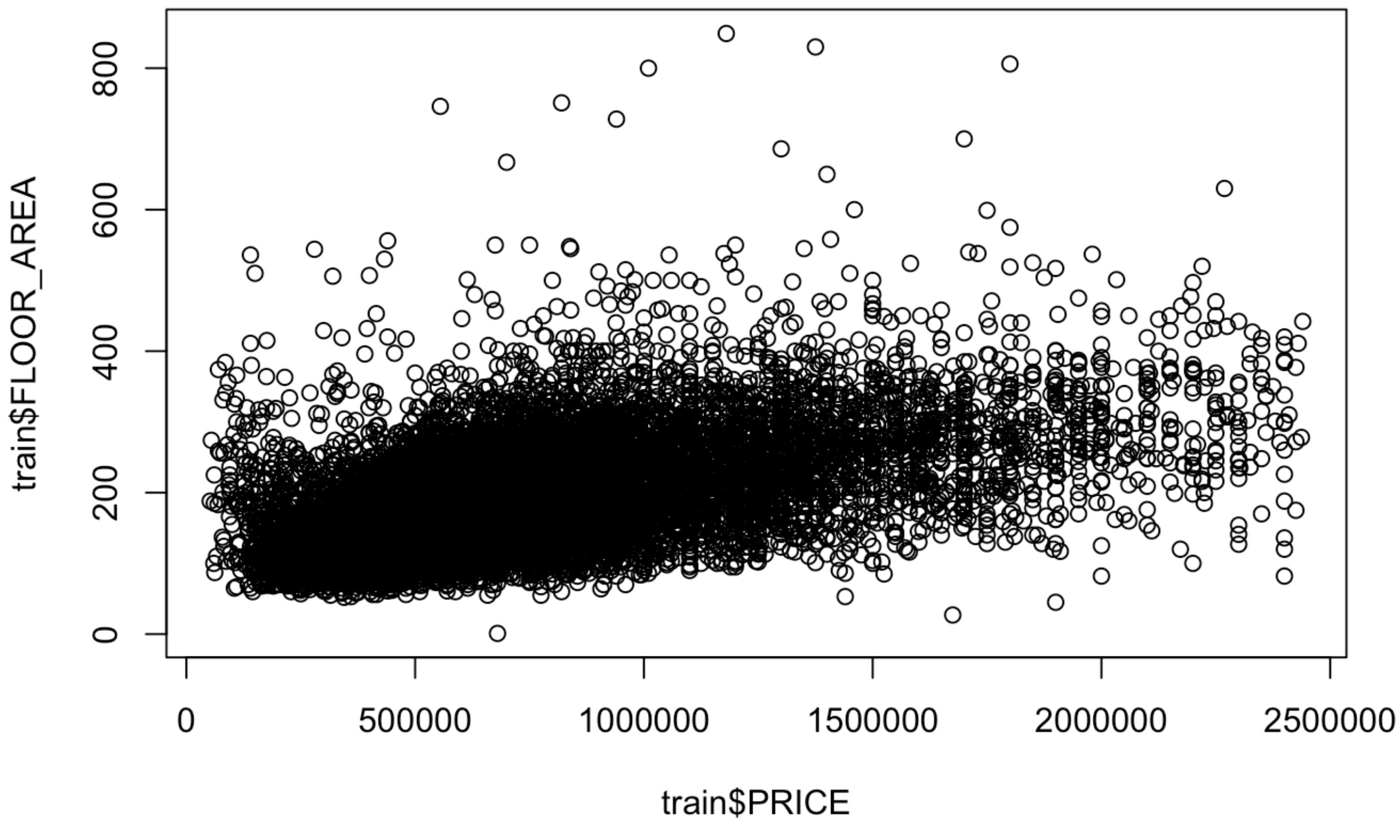
```
boxplot(train$LAND_AREA)
```




```
cdplot(factor(ifelse(train$PRICE>=mean(df$PRICE), "high", "low"))~train$FLOOR_AREA, x
lab="Floor Area", ylab="Price")
```




```
plot(train$FLOOR_AREA~train$PRICE)
```



Originally I had wanted to use LAND_AREA, the room counts, and position to predict the price (specifically if it was above or below average). However, via the data exploration, I was able to determine that LAND_AREA had many outliers and it would be difficult to get good information out of it, so as a substitute, we can use FLOOR_AREA.

Logistic Regression

Here we train the model with the training data using regular Logistic Regression. We can make a summary of this model and check the accuracy of it to compare with other models.

```
glm1 <- glm(factor(ifelse(train$PRICE>=mean(train$PRICE), "high", "low"))~FLOOR_AREA+
BEDROOMS+BATHROOMS+GARAGE+LONGITUDE+LATITUDE, data=train, family=binomial)

summary(glm1)
```

```
##
## Call:
## glm(formula = factor(ifelse(train$PRICE >= mean(train$PRICE),
##     "high", "low")) ~ FLOOR_AREA + BEDROOMS + BATHROOMS + GARAGE +
##     LONGITUDE + LATITUDE, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7693  -0.8770   0.4939   0.8082   4.4655
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.915e+02  2.157e+01 -22.784  < 2e-16 ***
## FLOOR_AREA  -1.786e-02  4.152e-04 -43.005  < 2e-16 ***
## BEDROOMS     4.639e-01  3.353e-02  13.835  < 2e-16 ***
## BATHROOMS   -3.916e-01  4.416e-02  -8.868  < 2e-16 ***
## GARAGE      -1.310e-01  1.653e-02  -7.923 2.32e-15 ***
## LONGITUDE    4.309e+00  1.867e-01  23.082  < 2e-16 ***
## LATITUDE     1.399e-01  1.078e-01   1.298    0.194
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21889  on 16552  degrees of freedom
## Residual deviance: 17130  on 16546  degrees of freedom
## AIC: 17144
##
## Number of Fisher Scoring iterations: 4
```

```
pred1 <- predict(glm1, newdata=test, type="response")
probs <- ifelse(pred1>0.5, 1, 0)
acc1 <- mean(probs==as.integer(factor(ifelse(test$PRICE>=mean(test$PRICE), "high", "l
ow"))))
print(paste("glm1 accuracy = ", acc1))
```

```
## [1] "glm1 accuracy =  0.158733993718289"
```

We see that with our logistic model, using not only the the floor area, but also the amount of primary rooms (bedrooms, bathrooms, garages), and location (longitude and latitude), only has an accuracy of 15.7% when estimating price. The model also seems to think Latitude is not a good predictor.

kNN Classification

Here we use kNN (k Nearest Neighbors) as a different model using the same predictors to see if we can get something more accurate.

```
kNN.train <- data.frame(matrix(ncol = 6, nrow = nrow(train)))
kNN.test <- data.frame(matrix(ncol = 6, nrow = nrow(test)))

library(class)
for(column in 4:7){
  kNN.train[, column-3] <- train[, column]
  kNN.test[, column-3] <- test[, column]
}

for(column in 15:16){
  kNN.train[, column-10] <- train[, column]
  kNN.test[, column-10] <- test[, column]
}

# normalize data
means <- sapply(kNN.train, mean)
stdvs <- sapply(kNN.train, sd)
kNN.train <- scale(kNN.train, center=means, scale=stdvs)
kNN.test <- scale(kNN.test, center=means, scale=stdvs)

train.labels <- as.factor(ifelse(train$PRICE>=mean(train$PRICE), "high", "low"))
test.labels <- as.factor(ifelse(test$PRICE>=mean(test$PRICE), "high", "low"))

pred2 <- knn(train=kNN.train, test=kNN.test, cl=train.labels, k=10)

results <- pred2 == test.labels
acc2 <- length(which(results == TRUE)) / length(results)
print(paste("kNN accuracy = ", acc2))
```

```
## [1] "kNN accuracy =  0.862285576226142"
```

After testing the data at a few different values of k (3-15), and the accuracy seems to fall at about a range of 85-87% which is already much more accurate than the logistic model. This is due to the high variance of the model, allowing it to fit to the data more.

Decision Tree

Now we use a decision tree model which may have a bit lower accuracy but should be more transparent as to what the predictors are being used for.

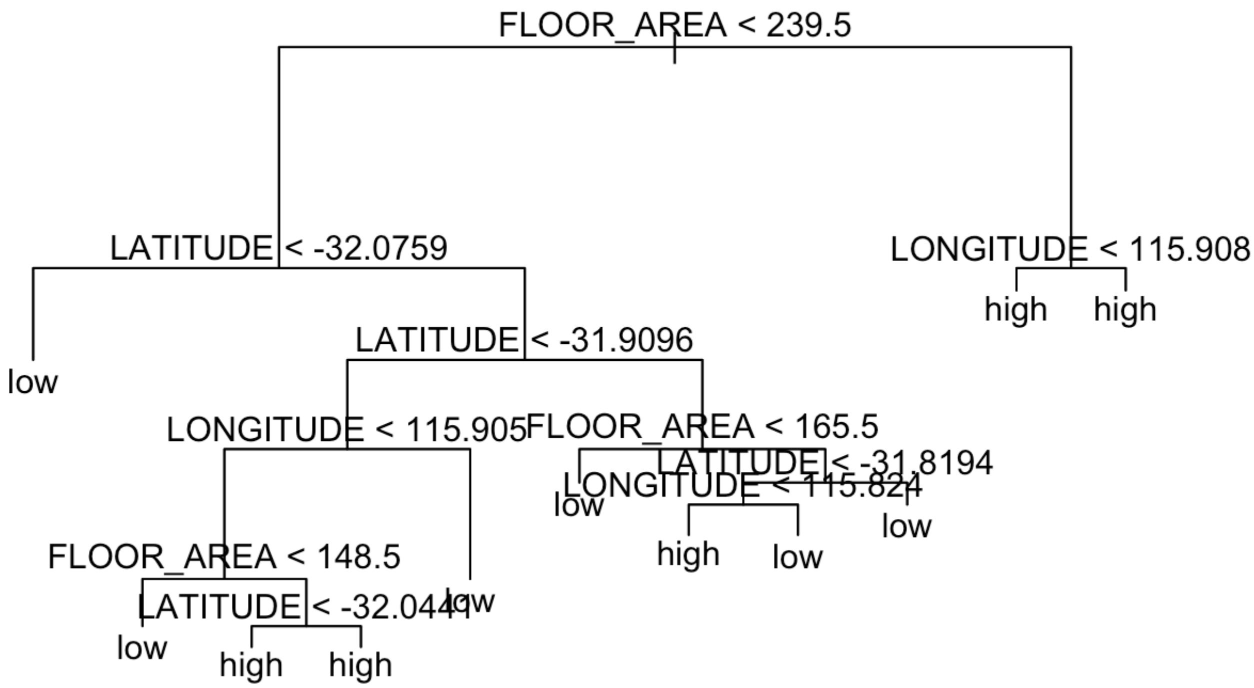
```r
library(tree)
#perth.tree <- tree(factor(ifelse(PRICE>=mean(PRICE), "high", "low"))~FLOOR_AREA+BEDR
OOMS+BATHROOMS+GARAGE, data=train, method="class")
perth.tree <- tree(factor(ifelse(PRICE>=mean(PRICE), "high", "low"))~FLOOR_AREA+BEDRO
OMS+BATHROOMS+GARAGE+LONGITUDE+LATITUDE, data=train, method="class")
perth.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 16553 21890.0 low ( 0.37425 0.62575 )
##    2) FLOOR_AREA < 239.5 13232 15560.0 low ( 0.27486 0.72514 )
##      4) LATITUDE < -32.0759 2662   972.2 low ( 0.04470 0.95530 ) *
##      5) LATITUDE > -32.0759 10570 13450.0 low ( 0.33283 0.66717 )
##       10) LATITUDE < -31.9096 4960  6875.0 low ( 0.49375 0.50625 )
##         20) LONGITUDE < 115.905 3322  4143.0 high ( 0.68423 0.31577 )
##           40) FLOOR_AREA < 148.5 1427  1970.0 low ( 0.46111 0.53889 ) *
##           41) FLOOR_AREA > 148.5 1895  1587.0 high ( 0.85224 0.14776 )
##             82) LATITUDE < -32.0441 507   672.7 high ( 0.62130 0.37870 ) *
##             83) LATITUDE > -32.0441 1388   655.8 high ( 0.93660 0.06340 ) *
##         21) LONGITUDE > 115.905 1638  1118.0 low ( 0.10745 0.89255 ) *
##       11) LATITUDE > -31.9096 5610  5464.0 low ( 0.19055 0.80945 )
##         22) FLOOR_AREA < 165.5 2934  1775.0 low ( 0.08998 0.91002 ) *
##         23) FLOOR_AREA > 165.5 2676  3273.0 low ( 0.30082 0.69918 )
##           46) LATITUDE < -31.8194 966  1339.0 low ( 0.49793 0.50207 )
##             92) LONGITUDE < 115.824 407   343.9 high ( 0.85012 0.14988 ) *
##             93) LONGITUDE > 115.824 559   618.0 low ( 0.24150 0.75850 ) *
##           47) LATITUDE > -31.8194 1710  1660.0 low ( 0.18947 0.81053 ) *
##    3) FLOOR_AREA > 239.5 3321  3580.0 high ( 0.77025 0.22975 )
##      6) LONGITUDE < 115.908 2493  2163.0 high ( 0.84356 0.15644 ) *
##      7) LONGITUDE > 115.908 828  1140.0 high ( 0.54952 0.45048 ) *
```

```r
summary(perth.tree)
```

```
##
## Classification tree:
## tree(formula = factor(ifelse(PRICE >= mean(PRICE), "high", "low")) ~
##     FLOOR_AREA + BEDROOMS + BATHROOMS + GARAGE + LONGITUDE +
##         LATITUDE, data = train, method = "class")
## Variables actually used in tree construction:
## [1] "FLOOR_AREA" "LATITUDE"   "LONGITUDE"
## Number of terminal nodes:  11
## Residual mean deviance:  0.7912 = 13090 / 16540
## Misclassification error rate: 0.1679 = 2780 / 16553
```

```
plot(perth.tree)
text(perth.tree, pretty=0)
```




```
pred3 <- predict(perth.tree, newdata=test, type="class")
table(pred3, as.factor(ifelse(test$PRICE>=mean(test$PRICE), "high", "low")))
```

```
##
## pred3  high  low
##   high 1163  302
##   low   388 2286
```

```
acc3 <- mean(pred3 == factor(ifelse(test$PRICE>=mean(test$PRICE), "high", "low")))

print(paste("DT accuracy = ", acc3))
```

```
## [1] "DT accuracy =  0.833293065957961"
```

As expected, the accuracy is a little bit lower at 83.3%.

What is interesting to note, is that the tree function deemed all of the primary room counts as irrelevant predictors. This is most likely because FLOOR_AREA is already highly correlated to the number of primary rooms in the house, making it redundant. However depending on the LONGITUDE and LATITUDE, different FLOOR_AREAs result in prices above or below average. This makes sense logically because different neighborhoods will cost more or less to live in.

Results

The models from most to least accurate for this data was:

- kNN
- Decision Tree
- Logistic Regression

Logistic was not very good for this dataset mostly because of its high bias. As shown in the Decision Tree, two of the most important predictors was LONGITUDE and LATITUDE and it wasn't a linear relationship for the classification.

Decision Tree was only a little bit less accurate than kNN and is still pretty good for this dataset. Even though it was less accurate than kNN, it gave a bit more insight as to what were important factors for price.

kNN was the most accurate and very good for the dataset, however it definitely took more work to make it accurate. Splicing and Normalizing the data were important components that **had** to be done in order to get a higher accuracy.