Analyzing Local Market Trends to Predict Maryland Property Prices

Justin Mejia, Matthew Rutigliano, Emilee Sheaffer

2024-08-20

```
library(ggplot2)
library(leaps)
library(class)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ROCR)
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(dplyr)
library(kknn)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```r
library(arules)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(arulesViz)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
## The following object is masked from 'package:pROC':
##
##     auc
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(tree)
library(sf)
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
library(devtools)
```

```
## Loading required package: usethis
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## v purrr     1.0.2     v tibble    3.2.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x randomForest::combine() masks dplyr::combine()
## x Matrix::expand()        masks tidyr::expand()
## x dplyr::filter()         masks stats::filter()
## x dplyr::lag()            masks stats::lag()
## x purrr::lift()           masks caret::lift()
## x randomForest::margin()  masks ggplot2::margin()
## x Matrix::pack()          masks tidyr::pack()
## x arules::recode()        masks dplyr::recode()
## x Matrix::unpack()        masks tidyr::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(readxl)
```

Machine Learning Approaches to Predicting Residential Property Prices in Maryland: An Analysis of Local Market Trends

###Executive Summary

Our study investigates what variables impact the average sales price of housing in Maryland, utilizing data measured from January 2022 - May 2024. The housing sales data is primarily pulled from the Maryland Board of Realtors, in addition to utilizing the Maryland Office of Tourism and the U.S. Bureau of Economic Analysis (BEA) for additional variables, such as geographic region and personal income per capita, to make our analysis more robust. An important distinction is that housing data in our study includes sales of homes, condos, and co-ops.

The ideal goal of the study is to effectively determine which independent variables have an impact on the average sales price of housing. We predicted that region, personal income per capita, and season would have the biggest impact on housing sales prices. We also predicted that new listings and median days on market will show a negative correlation with sale price. The aim is to build a model that successfully predicts the sale prices of new properties that enter the market based on local market characteristics and housing market trends.

Variables:

- County: Counties of Maryland
- Region: Geographic regions of Maryland
- Month: Months of the year
- Quarter: Calendar quarters of the year
- Season: Calendar season of the year
- Year: Years (2022 - 2024)
- Median Days on Market: Measures the median days that a home is listed on the market per county by month
- Units Sold: Number of homes sold per county by month
- New Listings: Measures the number of new listings on the market per county by month
- Personal Income Per Capita: Measures the personal income for the average person per county by year, Numerical in US Dollars
- Average Sales Price: Average sales price of homes sold per county by month
- Median Sales Price: Median sales price of homes sold per county by month
- Units Pending: Number of homes under contract but not yet sold per county by month
- Active Inventory: Number of homes on the market per county by month
- Months Inventory: Measures the rate at which homes are sold per county by month. Notes the relationship between the number of homes sold in a month by the total number of homes for sale at the end of the month

Part I. Data Preparation

```r
# Load dataframe
df <- read_excel("Maryland_Housing_Stats.xlsx")

# Remove already extracted feature - did not want to include these variables in our analyses
df <- df %>% select(-`MONTH_YEAR`)
df <- df %>% select(-MEDIAN_SALE_PRICE)

# Corrections/simplifications - Making sure region naming convention is consistent & renaming income va
df$REGION <- gsub("Western Maryland", "Western", df$REGION)
names(df)[names(df) == "PERSONAL_INCOME_PER_CAPITA"] <- "INCOME"

# Variable pre-processing - converting some variables in our data set to numeric variables to make it e
df$UNITS_SOLD <- as.numeric(df$UNITS_SOLD)
df$UNITS_PENDING <- as.numeric(df$UNITS_PENDING)
df$ACTIVE_INVENTORY <- as.numeric(df$ACTIVE_INVENTORY)
df$MONTHS_INVENTORY <- as.numeric(df$MONTHS_INVENTORY)
df$MEDIAN_DAYS_MARKET <- as.numeric(df$MEDIAN_DAYS_MARKET)
df$NEW_LISTINGS <- as.numeric(df$NEW_LISTINGS)
# Factoring our categorical variables with separate levels i.e. Season - factored 1-4 covering each sea
df$COUNTY <- as.factor(df$COUNTY)
```

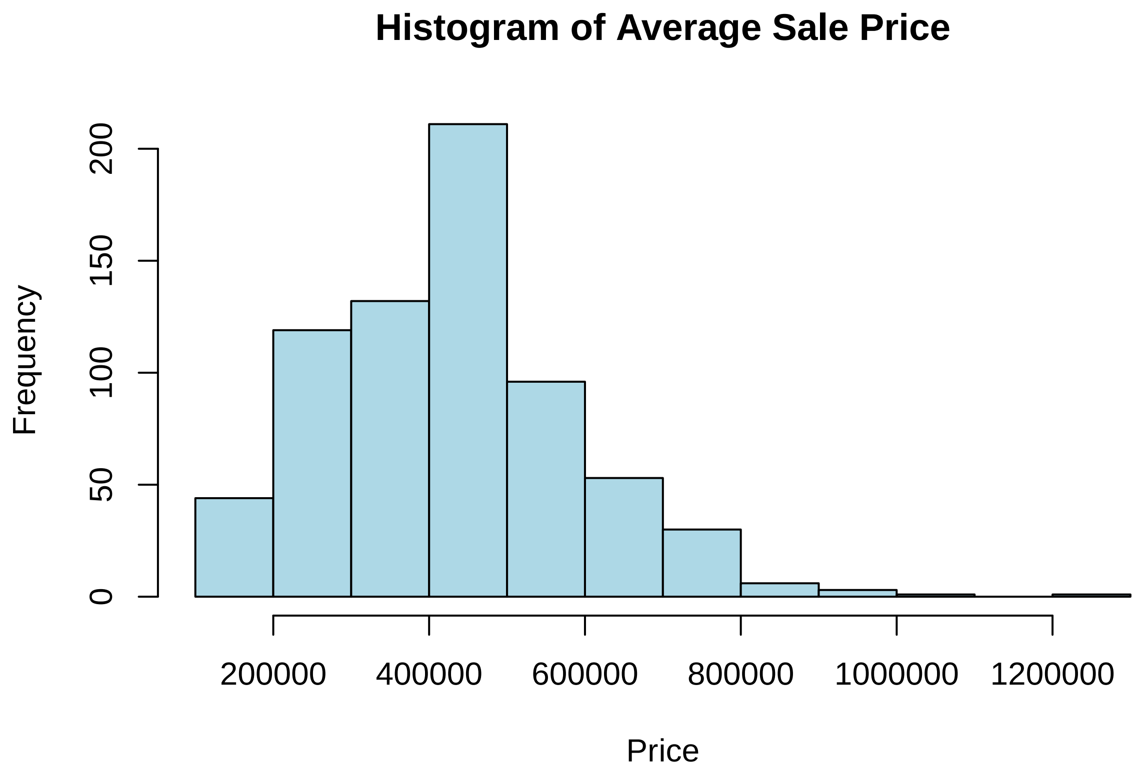
```r
df$MONTH <- as.factor(df$MONTH)
df$SEASON <- as.factor(df$SEASON)
df$QUARTER <- as.factor(df$QUARTER)
df$REGION <- as.factor(df$REGION)

# Summary - summarizing the variables in our dataset
summary(df)
```

```
##                  COUNTY          MONTH            YEAR        UNITS_SOLD
##  Allegany County     : 29   April    : 72   Min.   :2022   Min.   :   9.0
##  Anne Arundel County: 29   February: 72   1st Qu.:2022   1st Qu.:  54.0
##  Baltimore City      : 29   January : 72   Median :2023   Median : 130.0
##  Baltimore County    : 29   March    : 72   Mean   :2023   Mean   : 253.9
##  Calvert County      : 29   May      : 72   3rd Qu.:2023   3rd Qu.: 331.2
##  Caroline County     : 29   August   : 48   Max.   :2024   Max.   :1376.0
##  (Other)             :522   (Other) :288
##  AVG_SALE_PRICE    UNITS_PENDING     ACTIVE_INVENTORY MONTHS_INVENTORY
##  Min.   : 116234   Min.   :  11.00   Min.   :  39.0   Min.   :0.500
##  1st Qu.: 310967   1st Qu.:  57.75   1st Qu.: 147.0   1st Qu.:1.200
##  Median : 425650   Median : 134.00   Median : 210.0   Median :1.600
##  Mean   : 428284   Mean   : 263.97   Mean   : 410.7   Mean   :1.831
##  3rd Qu.: 512105   3rd Qu.: 343.00   3rd Qu.: 430.0   3rd Qu.:2.300
##  Max.   :1279814   Max.   :1354.00   Max.   :2484.0   Max.   :5.400
##
##  MEDIAN_DAYS_MARKET  NEW_LISTINGS       SEASON     QUARTER      INCOME
##  Min.   : 4.00       Min.   :  13.0   Autumn:144   1:216   Min.   : 38480
##  1st Qu.: 8.00       1st Qu.:  69.0   Spring:216   2:192   1st Qu.: 55624
##  Median :11.00       Median : 150.0   Summer:144   3:144   Median : 67452
##  Mean   :14.33       Mean   : 305.6   Winter:192   4:144   Mean   : 67048
##  3rd Qu.:18.00       3rd Qu.: 373.2                        3rd Qu.: 74512
##  Max.   :72.00       Max.   :1669.0                        Max.   :101208
##
##              REGION
##  Capital Region: 87
##  Central        :174
##  Eastern Shore  :261
##  Southern       : 87
##  Western        : 87
##
##
```

```r
# Set seed to 123
set.seed(123)

# Partition Data (at least 10 observations for every variable in training set)
inTrain <- sample(nrow(df), 0.7*nrow(df))
train <- data.frame(df[inTrain,])
test <- data.frame(df[-inTrain,])
```

Data Exploration

```
# Histogram of Avg Price - breaking down frequency of where housing prices fall across Maryland
hist(df$AVG_SALE_PRICE, breaks = 9, main = "Histogram of Average Sale Price", xlab = "Price", ylab = "F
```



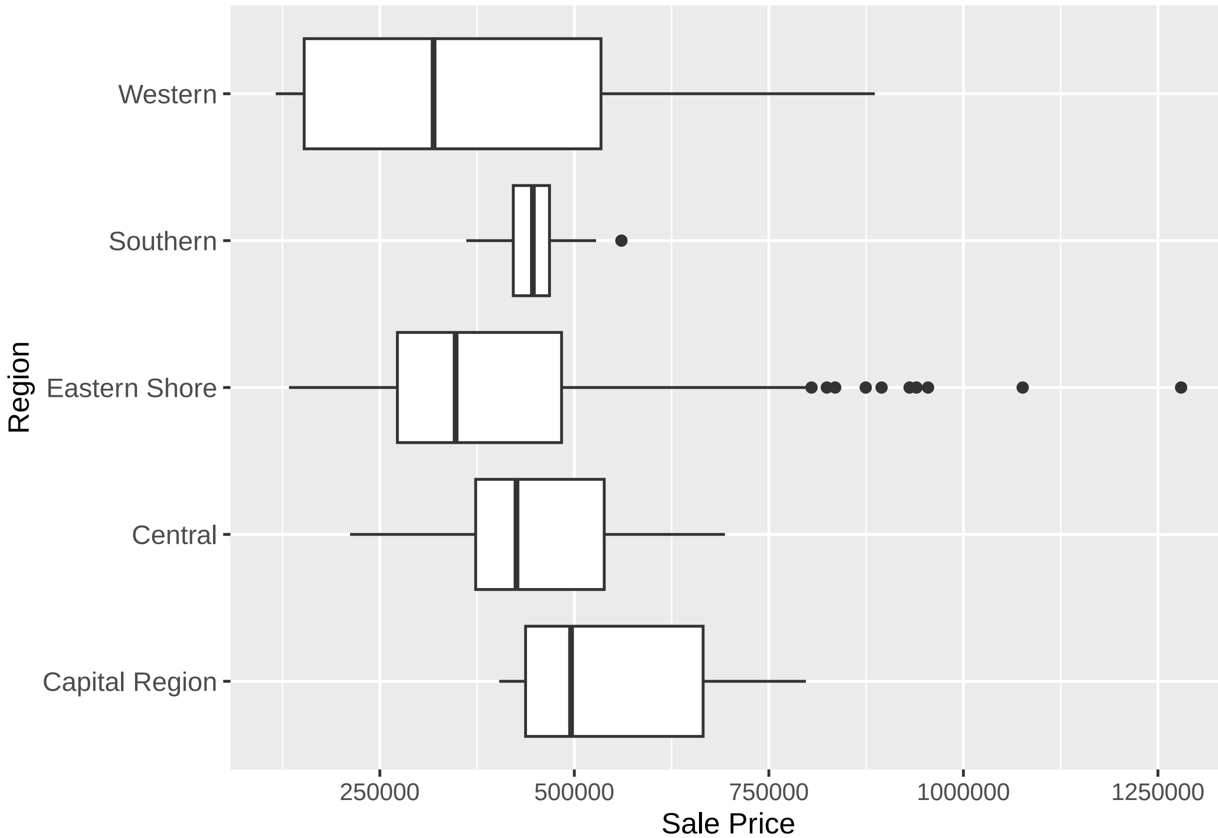
The distribution appears to be right-skewed (positively skewed), meaning that there are more properties with lower average sale prices, and fewer properties with higher prices. The bulk of the data is concentrated between $200,000 and $600,000. The most frequent average sale price range, corresponding to the highest bar, appears to fall between $400,000 and $500,000. There are a few instances of much higher average sale prices, such as outlying prices extending beyond $800,000 and up to more than $1,200,000.

This distribution is expected in housing markets where a majority of homes fall into a more affordable range, with higher-priced homes being less common.

```
# Boxplot of Avg Price by Region - breaking out distribution of housing sale price by region
ggplot(df, aes(x = AVG_SALE_PRICE, y = REGION)) + geom_boxplot() +
  labs(x = "Sale Price", y = "Region") +
  theme(axis.text.y = element_text(size = 10), main="Average Sale Price by Region", horizontal = TRUE)
```

```
## Warning in plot_theme(plot): The `main` theme element is not defined in the
## element hierarchy.
```

```
## Warning in plot_theme(plot): The `horizontal` theme element is not defined in
## the element hierarchy.
```

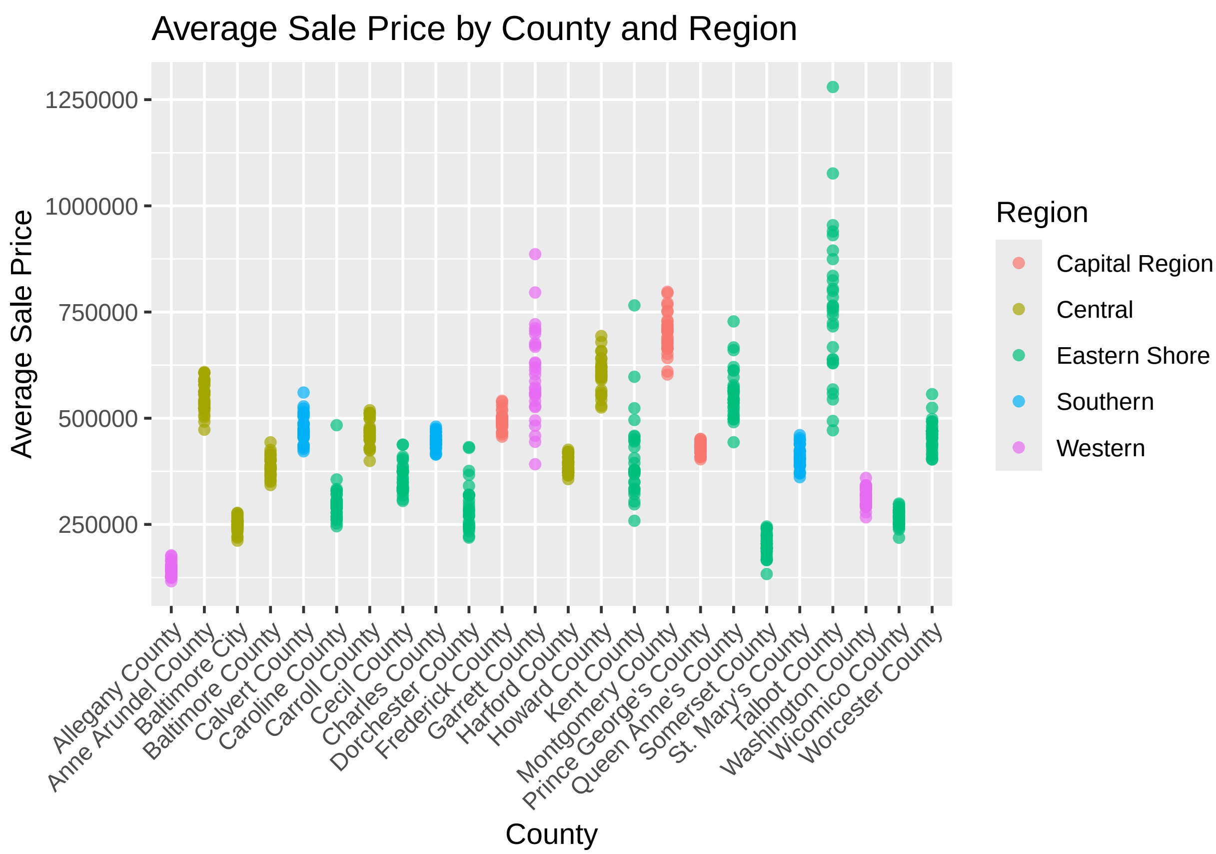


While the Capital Region of Maryland surrounding Washington D.C. had the highest median of average housing sales price, at around $500,000, the Eastern Shore of Maryland had the largest average housing sales price in Maryland overall (more than $1,250,000).

Unsurprisingly, the Eastern Shore of Maryland saw the largest distribution of average sales price, with both the lowest average sales price and largest average sales price occurring in the same region.

Southern Maryland had the narrowest distribution of average sales price, with most home prices falling between $375,000 and $500,000.

```
# Scatterplot - how average sales price is distributed by county and region across Maryland. Average sa
ggplot(df, aes(x = COUNTY, y = AVG_SALE_PRICE, color = REGION)) +
  geom_point(alpha = 0.7) +
  labs(title = "Average Sale Price by County and Region",
       x = "County", y = "Average Sale Price") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_color_discrete(name = "Region")
```



Talbot County in the Eastern Shore of Maryland saw the largest distribution of average home sales prices. Montgomery County in the Capital Region had the largest average home sales prices in Maryland, while Allegany County in Western Maryland saw the lowest average home sales prices, followed by Somerset County in Eastern Maryland.

Linear Regression Model (Avg Price)

```
# LRM 0, all variables as independent variables (judging by the NAs, multicollinearity present)
LRMprice0 <- lm(AVG_SALE_PRICE~ ., data=train)
summary(LRMprice0)
```

```
##
## Call:
## lm(formula = AVG_SALE_PRICE ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -290318  -18425    -578   13630  489429
##
## Coefficients: (10 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -3.529e+07  1.475e+07  -2.393 0.017111 *
## COUNTYAnne Arundel County     3.126e+05  8.565e+04   3.649 0.000294 ***
## COUNTYBaltimore City          6.859e+04  9.122e+04   0.752 0.452545
```

```
## COUNTYBaltimore County        1.820e+05  6.869e+04   2.650 0.008329 **
## COUNTYCalvert County          2.833e+05  5.446e+04   5.203 3.01e-07 ***
## COUNTYCaroline County         1.310e+05  3.024e+04   4.331 1.84e-05 ***
## COUNTYCarroll County          2.677e+05  5.845e+04   4.581 6.03e-06 ***
## COUNTYCecil County            1.974e+05  2.860e+04   6.900 1.80e-11 ***
## COUNTYCharles County          2.643e+05  4.295e+04   6.153 1.70e-09 ***
## COUNTYDorchester County       1.272e+05  2.849e+04   4.463 1.03e-05 ***
## COUNTYFrederick County        2.810e+05  6.354e+04   4.423 1.23e-05 ***
## COUNTYGarrett County          4.275e+05  2.495e+04  17.132  < 2e-16 ***
## COUNTYHarford County          1.936e+05  5.319e+04   3.640 0.000304 ***
## COUNTYHoward County           3.523e+05  9.302e+04   3.787 0.000173 ***
## COUNTYKent County             2.138e+05  5.264e+04   4.061 5.78e-05 ***
## COUNTYMontgomery County       4.123e+05  1.158e+05   3.559 0.000412 ***
## COUNTYPrince George's County  2.551e+05  6.494e+04   3.928 9.94e-05 ***
## COUNTYQueen Anne's County     3.477e+05  7.354e+04   4.728 3.04e-06 ***
## COUNTYSomerset County         7.105e+04  2.473e+04   2.873 0.004265 **
## COUNTYSt. Mary's County       2.216e+05  4.638e+04   4.778 2.41e-06 ***
## COUNTYTalbot County           5.376e+05  8.923e+04   6.025 3.56e-09 ***
## COUNTYWashington County       1.569e+05  2.736e+04   5.733 1.83e-08 ***
## COUNTYWicomico County         1.257e+05  2.044e+04   6.149 1.74e-09 ***
## COUNTYWorcester County        2.657e+05  4.494e+04   5.911 6.78e-09 ***
## MONTHAugust                   6.913e+03  1.371e+04   0.504 0.614353
## MONTHDecember                -1.329e+04  1.559e+04  -0.852 0.394512
## MONTHFebruary                -2.550e+04  1.210e+04  -2.107 0.035680 *
## MONTHJanuary                 -2.571e+04  1.235e+04  -2.081 0.038003 *
## MONTHJuly                     1.525e+04  1.268e+04   1.203 0.229775
## MONTHJune                     2.322e+04  1.247e+04   1.862 0.063280 .
## MONTHMarch                   -1.755e+04  1.131e+04  -1.552 0.121394
## MONTHMay                      1.682e+04  1.152e+04   1.460 0.144960
## MONTHNovember                 1.229e+04  1.367e+04   0.899 0.369270
## MONTHOctober                  2.285e+04  1.402e+04   1.629 0.103965
## MONTHSeptember                4.174e+03  1.305e+04   0.320 0.749247
## YEAR                          1.746e+04  7.322e+03   2.385 0.017492 *
## UNITS_SOLD                    6.658e+01  6.520e+01   1.021 0.307749
## UNITS_PENDING                -1.295e+01  1.017e+02  -0.127 0.898778
## ACTIVE_INVENTORY             -2.575e+01  4.194e+01  -0.614 0.539669
## MONTHS_INVENTORY             -1.276e+03  9.113e+03  -0.140 0.888668
## MEDIAN_DAYS_MARKET            5.293e+02  4.517e+02   1.172 0.241876
## NEW_LISTINGS                  1.392e+01  7.291e+01   0.191 0.848715
## SEASONSpring                         NA         NA      NA       NA
## SEASONSummer                         NA         NA      NA       NA
## SEASONWinter                         NA         NA      NA       NA
## QUARTER2                             NA         NA      NA       NA
## QUARTER3                             NA         NA      NA       NA
## QUARTER4                             NA         NA      NA       NA
## INCOME                        2.192e+00  1.923e+00   1.140 0.254846
## REGIONCentral                        NA         NA      NA       NA
## REGIONEastern Shore                  NA         NA      NA       NA
## REGIONSouthern                       NA         NA      NA       NA
## REGIONWestern                        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55110 on 444 degrees of freedom
```

```
## Multiple R-squared:  0.8884, Adjusted R-squared:  0.8779
## F-statistic: 84.18 on 42 and 444 DF,  p-value: < 2.2e-16
```

```r
# LRM 1, Season subbed in for Month, Year, Quarter. County subbed in for Region
LRMprice1 <- lm(AVG_SALE_PRICE~ COUNTY+UNITS_SOLD+UNITS_PENDING+ACTIVE_INVENTORY+MONTHS_INVENTORY+MEDIAN
summary(LRMprice1)
```

```
##
## Call:
## lm(formula = AVG_SALE_PRICE ~ COUNTY + UNITS_SOLD + UNITS_PENDING +
##     ACTIVE_INVENTORY + MONTHS_INVENTORY + MEDIAN_DAYS_MARKET +
##     NEW_LISTINGS + SEASON + INCOME, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -299063  -19605   -2089   15879  490101
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -1.068e+05  6.126e+04  -1.743 0.082053 .
## COUNTYAnne Arundel County     2.474e+05  7.773e+04   3.182 0.001561 **
## COUNTYBaltimore City        4.260e+04  8.901e+04   0.479 0.632481
## COUNTYBaltimore County      1.571e+05  6.665e+04   2.357 0.018847 *
## COUNTYCalvert County        2.295e+05  4.630e+04   4.956 1.02e-06 ***
## COUNTYCaroline County       1.016e+05  2.600e+04   3.907 0.000108 ***
## COUNTYCarroll County        2.148e+05  5.062e+04   4.243 2.68e-05 ***
## COUNTYCecil County          1.761e+05  2.654e+04   6.634 9.34e-11 ***
## COUNTYCharles County        2.345e+05  3.902e+04   6.011 3.81e-09 ***
## COUNTYDorchester County     8.419e+04  2.250e+04   3.742 0.000206 ***
## COUNTYFrederick County      2.288e+05  5.635e+04   4.060 5.78e-05 ***
## COUNTYGarrett County        3.913e+05  2.095e+04  18.675  < 2e-16 ***
## COUNTYHarford County        1.531e+05  4.771e+04   3.208 0.001430 **
## COUNTYHoward County         2.560e+05  7.759e+04   3.299 0.001047 **
## COUNTYKent County           1.504e+05  4.193e+04   3.588 0.000369 ***
## COUNTYMontgomery County     3.130e+05  1.028e+05   3.043 0.002477 **
## COUNTYPrince George's County  2.609e+05  6.414e+04   4.068 5.59e-05 ***
## COUNTYQueen Anne's County     2.573e+05  5.832e+04   4.411 1.29e-05 ***
## COUNTYSomerset County       6.916e+04  2.429e+04   2.847 0.004613 **
## COUNTYSt. Mary's County     1.809e+05  4.032e+04   4.487 9.18e-06 ***
## COUNTYTalbot County         4.205e+05  6.837e+04   6.150 1.70e-09 ***
## COUNTYWashington County     1.470e+05  2.656e+04   5.535 5.28e-08 ***
## COUNTYWicomico County       1.317e+05  2.050e+04   6.422 3.40e-10 ***
## COUNTYWorcester County      2.223e+05  3.893e+04   5.709 2.06e-08 ***
## UNITS_SOLD                  7.745e+01  5.795e+01   1.336 0.182082
## UNITS_PENDING               6.392e+00  9.781e+01   0.065 0.947928
## ACTIVE_INVENTORY           -3.599e+01  4.152e+01  -0.867 0.386529
## MONTHS_INVENTORY            1.299e+04  8.104e+03   1.603 0.109645
## MEDIAN_DAYS_MARKET          3.968e+02  4.305e+02   0.922 0.357218
## NEW_LISTINGS               -1.563e+01  6.918e+01  -0.226 0.821342
## SEASONSpring               -4.995e+03  8.293e+03  -0.602 0.547298
## SEASONSummer                6.081e+03  8.430e+03   0.721 0.471082
## SEASONWinter               -2.681e+04  8.638e+03  -3.104 0.002028 **
## INCOME                      4.727e+00  1.442e+00   3.278 0.001125 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55600 on 453 degrees of freedom
## Multiple R-squared:  0.8841, Adjusted R-squared:  0.8757
## F-statistic: 104.8 on 33 and 453 DF,  p-value: < 2.2e-16
```

```
# LRM 2, Season subbed in for Month, Year, Quarter. Region subbed in for County
LRMprice2 <- lm(AVG_SALE_PRICE~ REGION+UNITS_SOLD+UNITS_PENDING+ACTIVE_INVENTORY+MONTHS_INVENTORY+MEDIA
summary(LRMprice2)
```

```
##
## Call:
## lm(formula = AVG_SALE_PRICE ~ REGION + UNITS_SOLD + UNITS_PENDING +
##     ACTIVE_INVENTORY + MONTHS_INVENTORY + MEDIAN_DAYS_MARKET +
##     NEW_LISTINGS + SEASON + INCOME, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -245283  -43057   -1690   36835  574354
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.886e+05  3.563e+04  -5.292 1.85e-07 ***
## REGIONCentral       -7.645e+04  1.447e+04  -5.283 1.95e-07 ***
## REGIONEastern Shore -3.193e+04  2.088e+04  -1.530 0.126805
## REGIONSouthern      -1.487e+04  2.012e+04  -0.739 0.460203
## REGIONWestern        1.761e+04  2.409e+04   0.731 0.465256
## UNITS_SOLD           1.841e+02  8.313e+01   2.214 0.027307 *
## UNITS_PENDING       -3.115e+01  9.098e+01  -0.342 0.732225
## ACTIVE_INVENTORY    -8.765e+01  2.351e+01  -3.728 0.000217 ***
## MONTHS_INVENTORY     2.357e+04  8.172e+03   2.884 0.004103 **
## MEDIAN_DAYS_MARKET  -4.005e+02  6.174e+02  -0.649 0.516826
## NEW_LISTINGS        -7.541e+00  8.080e+01  -0.093 0.925677
## SEASONSpring        -1.580e+04  1.227e+04  -1.288 0.198546
## SEASONSummer        -8.384e+03  1.276e+04  -0.657 0.511475
## SEASONWinter        -3.034e+04  1.304e+04  -2.326 0.020445 *
## INCOME               9.313e+00  3.372e-01  27.620  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 88010 on 472 degrees of freedom
## Multiple R-squared:  0.6975, Adjusted R-squared:  0.6885
## F-statistic: 77.74 on 14 and 472 DF,  p-value: < 2.2e-16
```

```
# Performance of LRM Models

# Predict on training data
train_preds_LRMprice0 <- predict(LRMprice0, newdata = train)
train_preds_LRMprice1 <- predict(LRMprice1, newdata = train)
train_preds_LRMprice2 <- predict(LRMprice2, newdata = train)

# Predict on test data
test_preds_LRMprice0 <- predict(LRMprice0, newdata = test)
```

```r
test_preds_LRMprice1 <- predict(LRMprice1, newdata = test)
test_preds_LRMprice2 <- predict(LRMprice2, newdata = test)

# Calculate average house price for train and test data
avg_price_train <- mean(train$AVG_SALE_PRICE)
avg_price_test <- mean(test$AVG_SALE_PRICE)

# Calculate RMSE and MAE for train and test data
rmse_train_LRMprice0 <- rmse(train$AVG_SALE_PRICE, train_preds_LRMprice0)
rmse_train_LRMprice1 <- rmse(train$AVG_SALE_PRICE, train_preds_LRMprice1)
rmse_train_LRMprice2 <- rmse(train$AVG_SALE_PRICE, train_preds_LRMprice2)

rmse_test_LRMprice0 <- rmse(test$AVG_SALE_PRICE, test_preds_LRMprice0)
rmse_test_LRMprice1 <- rmse(test$AVG_SALE_PRICE, test_preds_LRMprice1)
rmse_test_LRMprice2 <- rmse(test$AVG_SALE_PRICE, test_preds_LRMprice2)

mae_train_LRMprice0 <- mae(train$AVG_SALE_PRICE, train_preds_LRMprice0)
mae_train_LRMprice1 <- mae(train$AVG_SALE_PRICE, train_preds_LRMprice1)
mae_train_LRMprice2 <- mae(train$AVG_SALE_PRICE, train_preds_LRMprice2)

mae_test_LRMprice0 <- mae(test$AVG_SALE_PRICE, test_preds_LRMprice0)
mae_test_LRMprice1 <- mae(test$AVG_SALE_PRICE, test_preds_LRMprice1)
mae_test_LRMprice2 <- mae(test$AVG_SALE_PRICE, test_preds_LRMprice2)

# Normalize RMSE and MAE for train and test data
normalized_rmse_train_LRMprice0 <- rmse_train_LRMprice0 / avg_price_train * 100
normalized_rmse_train_LRMprice1 <- rmse_train_LRMprice1 / avg_price_train * 100
normalized_rmse_train_LRMprice2 <- rmse_train_LRMprice2 / avg_price_train * 100

normalized_rmse_test_LRMprice0 <- rmse_test_LRMprice0 / avg_price_test * 100
normalized_rmse_test_LRMprice1 <- rmse_test_LRMprice1 / avg_price_test * 100
normalized_rmse_test_LRMprice2 <- rmse_test_LRMprice2 / avg_price_test * 100

normalized_mae_train_LRMprice0 <- mae_train_LRMprice0 / avg_price_train * 100
normalized_mae_train_LRMprice1 <- mae_train_LRMprice1 / avg_price_train * 100
normalized_mae_train_LRMprice2 <- mae_train_LRMprice2 / avg_price_train * 100

normalized_mae_test_LRMprice0 <- mae_test_LRMprice0 / avg_price_test * 100
normalized_mae_test_LRMprice1 <- mae_test_LRMprice1 / avg_price_test * 100
normalized_mae_test_LRMprice2 <- mae_test_LRMprice2 / avg_price_test * 100

# Print normalized results for training and test data
cat("LRMprice0:\n")
```

```
## LRMprice0:
```

```r
cat("Train Normalized RMSE (% of avg price):", normalized_rmse_train_LRMprice0, "\n")
```

```
## Train Normalized RMSE (% of avg price): 12.17426
```

```r
cat("Test Normalized RMSE (% of avg price):", normalized_rmse_test_LRMprice0, "\n")
```

```
## Test Normalized RMSE (% of avg price): 11.81459
```

```r
cat("Train Normalized MAE (% of avg price):", normalized_mae_train_LRMprice0, "\n")
```

```
## Train Normalized MAE (% of avg price): 6.715687
```

```r
cat("Test Normalized MAE (% of avg price):", normalized_mae_test_LRMprice0, "\n\n")
```

```
## Test Normalized MAE (% of avg price): 7.128518
```

```r
cat("LRMprice1:\n")
```

```
## LRMprice1:
```

```r
cat("Train Normalized RMSE (% of avg price):", normalized_rmse_train_LRMprice1, "\n")
```

```
## Train Normalized RMSE (% of avg price): 12.40589
```

```r
cat("Test Normalized RMSE (% of avg price):", normalized_rmse_test_LRMprice1, "\n")
```

```
## Test Normalized RMSE (% of avg price): 12.04365
```

```r
cat("Train Normalized MAE (% of avg price):", normalized_mae_train_LRMprice1, "\n")
```

```
## Train Normalized MAE (% of avg price): 7.10078
```

```r
cat("Test Normalized MAE (% of avg price):", normalized_mae_test_LRMprice1, "\n\n")
```

```
## Test Normalized MAE (% of avg price): 7.165602
```

```r
cat("LRMprice2:\n")
```

```
## LRMprice2:
```

```r
cat("Train Normalized RMSE (% of avg price):", normalized_rmse_train_LRMprice2, "\n")
```

```
## Train Normalized RMSE (% of avg price): 20.04587
```

```r
cat("Test Normalized RMSE (% of avg price):", normalized_rmse_test_LRMprice2, "\n")
```

```
## Test Normalized RMSE (% of avg price): 22.67785
```

```r
cat("Train Normalized MAE (% of avg price):", normalized_mae_train_LRMprice2, "\n")
```

```
## Train Normalized MAE (% of avg price): 13.97181
```

```r
cat("Test Normalized MAE (% of avg price):", normalized_mae_test_LRMprice2, "\n")
```

```
## Test Normalized MAE (% of avg price): 15.51499
```

LRMprice0 and LRMprice1 have similar normalized errors. This tells us that these models are quite consistent in their performance. On average, the predictions from these models are off by about 10% of the average house price (for RMSE) and 6.5% (for MAE).

LRMprice2 has higher normalized errors. The predictions are, on average, off by about 18.8% of the average house price (for RMSE) and 14.0% (for MAE). This indicates that LRMprice2 is less accurate compared to the other models.

We should use LRM 0 or 1.

Predicting Prices - LRMprice1

```r
prediction_data <- data.frame(
  COUNTY = factor(c("Anne Arundel County", "Baltimore County", "Montgomery County"), levels = levels(tra
  UNITS_SOLD = c(100, 150, 200),
  UNITS_PENDING = c(50, 75, 100),
  ACTIVE_INVENTORY = c(300, 400, 500),
  MONTHS_INVENTORY = c(1.5, 2.0, 2.5),
  MEDIAN_DAYS_MARKET = c(15, 20, 25),
  NEW_LISTINGS = c(10, 15, 20),
  SEASON = factor(c("Spring", "Summer", "Winter"),
                  levels = levels(train$SEASON)),
  INCOME = c(50000, 60000, 70000))

predicted_prices <- predict(LRMprice1, prediction_data)
predicted_prices
```

```
##        1        2        3
## 394498.2 371396.5 550484.8
```

Logistic Regression Model (Classifying Hot and Cold Markets)

Create a categorical target variable that indicates whether the market in a county is "Hot" or "Cold". "Hot" markets are defined as those with high sales volumes, high average sale prices, and low median days on the market. Conversely, "Cold" markets have lower sales volumes, lower prices, and longer time on the market.

```r
# Create the target variable (Hot or Cold Market)
dfglm <- df
median_avg_sale_price <- median(dfglm$AVG_SALE_PRICE, na.rm = TRUE)
median_days_market <- median(dfglm$MEDIAN_DAYS_MARKET, na.rm = TRUE)
dfglm$MARKET_STATUS <- ifelse(dfglm$AVG_SALE_PRICE > median_avg_sale_price & dfglm$MEDIAN_DAYS_MARKET <
dfglm$MARKET_STATUS <- as.factor(dfglm$MARKET_STATUS)

# To avoid data leakage, we remove the two variables that determine Market Status
dfglm <- dfglm %>% select(-AVG_SALE_PRICE, -MEDIAN_DAYS_MARKET)
```

```r
# Partition Data
set.seed(123)
inTrain2 <- sample(nrow(dfglm), 0.7 * nrow(df))
market.train <- dfglm[inTrain2, ]
temp2 <- dfglm[-inTrain2, ]
market.validation <- sample(nrow(temp2), 0.5 * nrow(temp2))
market.val <- temp2[market.validation, ]
market.test <- temp2[-market.validation, ]
rm(temp2)

# Logistic Regression Model
# GLMmarket1 <- glm(MARKET_STATUS ~ REGION + UNITS_SOLD + UNITS_PENDING + ACTIVE_INVENTORY + MONTHS_INV
GLMmarket1 <- glm(MARKET_STATUS ~ ., data = market.train, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(GLMmarket1)
```

```
##
## Call:
## glm(formula = MARKET_STATUS ~ ., family = binomial, data = market.train)
##
## Coefficients: (10 not defined because of singularities)
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 1.375e+03  3.804e+03   0.361  0.71778
## COUNTYAnne Arundel County   1.607e+01  3.472e+03   0.005  0.99631
## COUNTYBaltimore City       -6.546e-01  4.485e+03   0.000  0.99988
## COUNTYBaltimore County     -7.421e+00  4.435e+03  -0.002  0.99866
## COUNTYCalvert County        1.674e+01  3.472e+03   0.005  0.99615
## COUNTYCaroline County      -2.597e+00  4.882e+03  -0.001  0.99958
## COUNTYCarroll County        1.754e+01  3.472e+03   0.005  0.99597
## COUNTYCecil County          1.539e+01  3.472e+03   0.004  0.99646
## COUNTYCharles County        1.709e+01  3.472e+03   0.005  0.99607
## COUNTYDorchester County    -1.004e+00  4.753e+03   0.000  0.99983
## COUNTYFrederick County      1.933e+01  3.472e+03   0.006  0.99556
## COUNTYGarrett County        1.875e+01  3.472e+03   0.005  0.99569
## COUNTYHarford County       -6.741e+00  4.856e+03  -0.001  0.99889
## COUNTYHoward County         1.470e+01  3.472e+03   0.004  0.99662
## COUNTYKent County           1.203e+01  3.472e+03   0.003  0.99724
## COUNTYMontgomery County     1.210e+01  3.472e+03   0.003  0.99722
## COUNTYPrince George's County 1.793e+01  3.472e+03   0.005  0.99588
## COUNTYQueen Anne's County   1.341e+01  3.472e+03   0.004  0.99692
## COUNTYSomerset County       2.133e+00  4.970e+03   0.000  0.99966
## COUNTYSt. Mary's County     1.442e+01  3.472e+03   0.004  0.99669
## COUNTYTalbot County         1.247e+01  3.472e+03   0.004  0.99713
## COUNTYWashington County    -2.446e+00  4.838e+03  -0.001  0.99960
## COUNTYWicomico County      -1.421e+00  4.923e+03   0.000  0.99977
## COUNTYWorcester County      1.499e+01  3.472e+03   0.004  0.99656
## MONTHAugust                 5.590e-02  1.013e+00   0.055  0.95598
## MONTHDecember              -2.904e+00  1.338e+00  -2.170  0.02999 *
## MONTHFebruary              -3.456e+00  1.054e+00  -3.279  0.00104 **
```

```
## MONTHJanuary                -4.293e+00  1.085e+00  -3.957 7.58e-05 ***
## MONTHJuly                    1.856e+00  9.373e-01   1.980  0.04775 *
## MONTHJune                    2.600e+00  9.462e-01   2.748  0.00600 **
## MONTHMarch                  -1.865e+00  8.557e-01  -2.180  0.02926 *
## MONTHMay                     2.666e+00  9.424e-01   2.829  0.00468 **
## MONTHNovember               -5.172e-01  1.035e+00  -0.500  0.61730
## MONTHOctober                -2.151e-01  1.119e+00  -0.192  0.84757
## MONTHSeptember               9.375e-01  9.986e-01   0.939  0.34783
## YEAR                        -6.932e-01  7.736e-01  -0.896  0.37021
## UNITS_SOLD                  -9.997e-04  5.594e-03  -0.179  0.85817
## UNITS_PENDING                9.866e-03  9.124e-03   1.081  0.27957
## ACTIVE_INVENTORY            -6.076e-03  3.979e-03  -1.527  0.12676
## MONTHS_INVENTORY            -1.325e+00  7.791e-01  -1.701  0.08891 .
## NEW_LISTINGS                -5.985e-04  6.520e-03  -0.092  0.92686
## SEASONSpring                        NA         NA      NA       NA
## SEASONSummer                        NA         NA      NA       NA
## SEASONWinter                        NA         NA      NA       NA
## QUARTER2                            NA         NA      NA       NA
## QUARTER3                            NA         NA      NA       NA
## QUARTER4                            NA         NA      NA       NA
## INCOME                       1.933e-04  2.276e-04   0.849  0.39577
## REGIONCentral                       NA         NA      NA       NA
## REGIONEastern Shore                 NA         NA      NA       NA
## REGIONSouthern                      NA         NA      NA       NA
## REGIONWestern                       NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 593.11  on 486  degrees of freedom
## Residual deviance: 179.07  on 445  degrees of freedom
## AIC: 263.07
##
## Number of Fisher Scoring iterations: 19
```

```r
# Predict on validation set
market.logistic_preds.val <- predict(GLMmarket1, newdata = market.val, type = "response")
market.logistic_class.val <- ifelse(market.logistic_preds.val > 0.5, "Hot", "Cold")

# Confusion Matrix for Logistic Regression vaidation
confusion_matrix_logistic.val <- table(Predicted = market.logistic_class.val, Actual = market.val$MARKET
print(confusion_matrix_logistic.val)
```

```
##          Actual
## Predicted Cold Hot
##      Cold   70   6
##      Hot     5  23
```

```r
cat("The validation error for this LR is:", (11)/(11+23+70))
```

```
## The validation error for this LR is: 0.1057692
```

```r
# Predict on test set
market.logistic_preds.test <- predict(GLMmarket1, newdata = market.test, type = "response")
market.logistic_class.test <- ifelse(market.logistic_preds.test > 0.5, "Hot", "Cold")

# Confusion Matrix for Logistic Regression test
confusion_matrix_logistic.test <- table(Predicted = market.logistic_class.test, Actual = market.test$MA
print(confusion_matrix_logistic.test)
```

```
##          Actual
## Predicted Cold Hot
##      Cold   73   5
##      Hot     3  24
```

```r
cat("The test error for this LR is:", (8)/(73+8+24))
```

```
## The test error for this LR is: 0.07619048
```

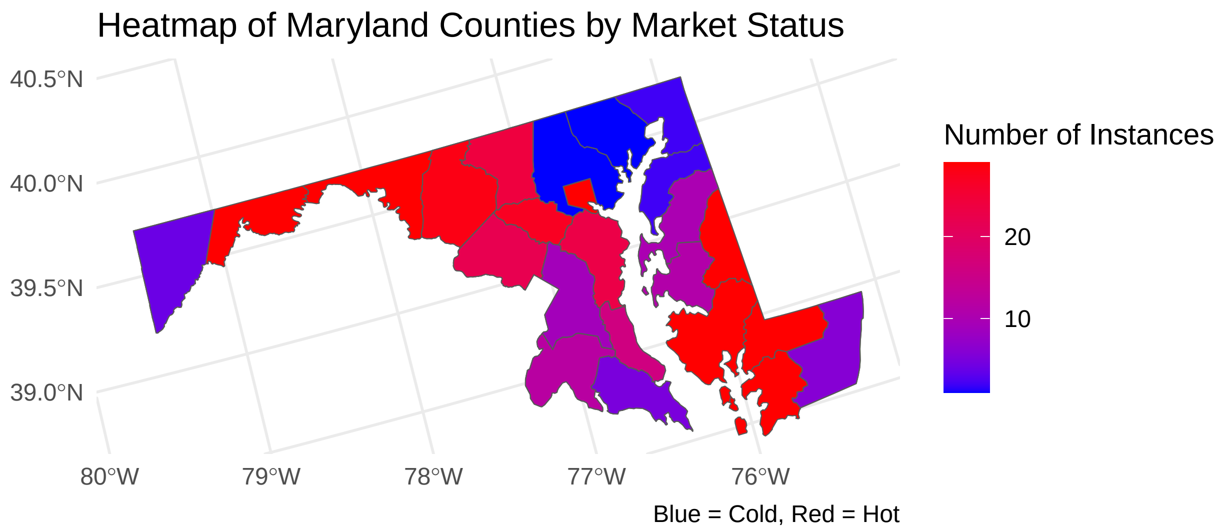
```r
# Here we created a heat map to display the Maryland county's market status
# Group the data by county and market status, then count the occurrences
county_status <- dfglm %>%
  group_by(COUNTY, MARKET_STATUS) %>%
  summarise(Count = n()) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'COUNTY'. You can override using the
## `.groups` argument.
```

```r
# Load Maryland shapefile for county boundaries (https://github.com/UrbanInstitute/urbnmapr)
# devtools::install_github("UrbanInstitute/urbnmapr")
library(tidyverse)
library(urbnmapr)
map <- get_urbn_map("counties", sf = TRUE)
map$county_name <- gsub("Baltimore city", "Baltimore City", map$county_name)
maryland_map <- map %>%
  filter(state_name == "Maryland")
```

```
## old-style crs object detected; please recreate object with a recent sf::st_crs()
```

```r
# Join the count data with the spatial data
map_data <- maryland_map %>%
  left_join(county_status, by = c("county_name" = "COUNTY"))

# Plot the heatmap
ggplot(data = map_data) +
  geom_sf(aes(fill = Count)) +
  scale_fill_gradient(low = "blue", high = "red") +
  labs(title = "Heatmap of Maryland Counties by Market Status",
       fill = "Number of Instances",
       caption = "Blue = Cold, Red = Hot") +
  theme_minimal()
```



KNN Model (Classifying Hot and Cold Markets)

```
# Prepare the data
dfknn <- dfglm
dfknn[,c(3:8, 11)] <- scale(dfknn[,c(3:8, 11)])

# Convert categorical variables to dummy variables
df_dummies <- model.matrix(~ COUNTY + MONTH + SEASON + REGION + QUARTER, data = dfknn)

# Remove the intercept column (the first column)
df_dummies <- df_dummies[, -1]

# Combine dummy variables with the rest of the dataset
dfknn <- cbind(dfknn[, !(names(dfknn) %in% c("COUNTY", "MONTH", "SEASON", "REGION", "QUARTER"))], df_du

# Partition data
set.seed(123)
inTrain2 <- sample(nrow(dfknn), 0.7 * nrow(dfknn))
market.train <- dfknn[inTrain2, ]
temp2 <- dfknn[-inTrain2, ]
market.validation <- sample(nrow(temp2), 0.5 * nrow(temp2))
market.val <- temp2[market.validation, ]
market.test <- temp2[-market.validation, ]
rm(temp2)
```

```r
train_input <- as.matrix(market.train[,-8])
train_output <- as.matrix(market.train[,8])
validate_input <- as.matrix(market.val[,-8])
test_input <- as.matrix(market.test[,-8])
```

```r
# KNN with K=3 (for reference) for Hot and Cold Classification
class.prediction <- knn(train_input, train_input, train_output, k=3)

# Training confusion matrix and error rate
confusion.train <- table(market.train$MARKET_STATUS, class.prediction)
error.train <- 1 - (confusion.train[1,1] + confusion.train[2,2]) / sum(confusion.train)

# Validation confusion matrix and error rate
class.prediction.val <- knn(train_input, validate_input, train_output, k=3)
confusion.validation <- table(market.val$MARKET_STATUS, class.prediction.val)
error.validation <- 1 - (confusion.validation[1,1] + confusion.validation[2,2]) / sum(confusion.validat

# Test confusion matrix and error rate
class.prediction.test <- knn(train_input, test_input, train_output, k=3)
confusion.test <- table(market.test$MARKET_STATUS, class.prediction.test)
error.test <- 1 - (confusion.test[1,1] + confusion.test[2,2]) / sum(confusion.test)
confusion.test
```

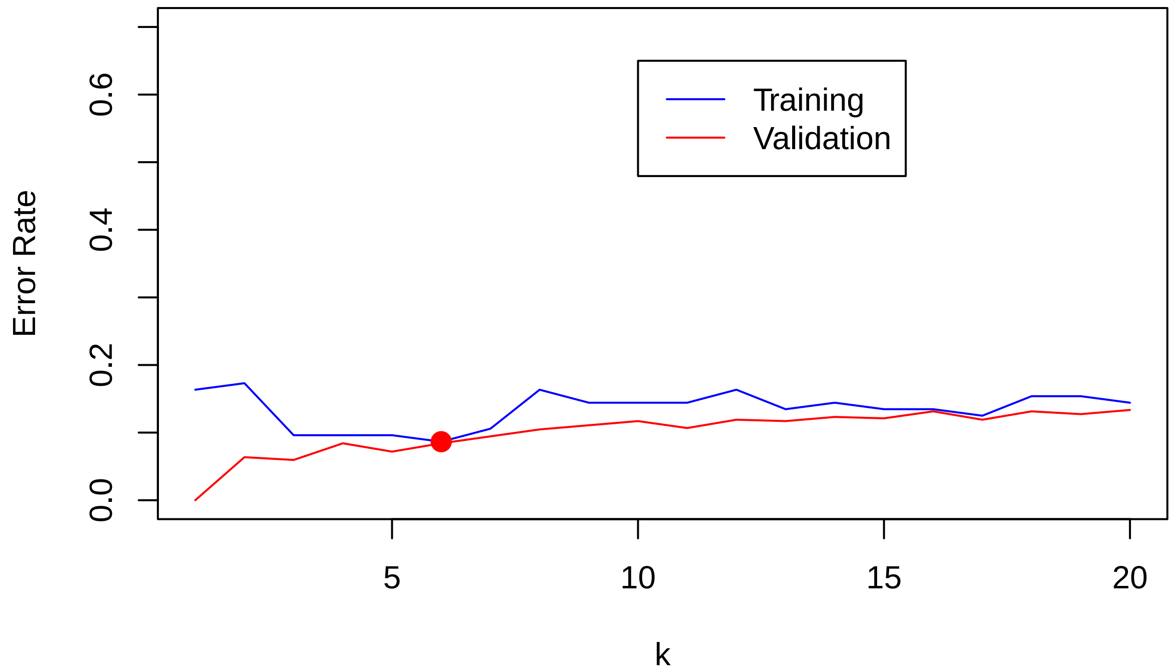
```
##       class.prediction.test
##        Cold Hot
##   Cold   75   1
##   Hot     8  21
```

```r
error.test
```

```
## [1] 0.08571429
```

```r
# Finding the best K
kmax <- 20
ER1 <- rep(0, kmax)
ER2 <- rep(0, kmax)
set.seed(123)

for (i in 1:kmax) {
  prediction <- knn(train_input, train_input, train_output, k=i)
  prediction2 <- knn(train_input, validate_input, train_output, k=i)
  CM1 <- table(market.train$MARKET_STATUS, prediction)
  ER1[i] <- (CM1[1,2] + CM1[2,1]) / sum(CM1)
  CM2 <- table(market.val$MARKET_STATUS, prediction2)
  ER2[i] <- (CM2[1,2] + CM2[2,1]) / sum(CM2)
}

# Plot
plot(c(1, kmax), c(0, 0.7), type="n", xlab="k", ylab="Error Rate")
lines(ER1, col="red")
lines(ER2, col="blue")
legend(10, 0.65, c("Training", "Validation"), lty=c(1,1), col=c("blue", "red"))
```

```r
z <- which.min(ER2) # z = 6 in this case
points(z, ER2[z], col="red", cex=2, pch=20)
```




```r
cat("Minimum Validation Error k:", z, "\n")
```

```
## Minimum Validation Error k: 6
```

```r
cat("Training Error:", ER1[z])
```

```
## Training Error: 0.08418891
```

```r
cat("Minimum Validation Error:", ER2[z], "\n")
```

```
## Minimum Validation Error: 0.08653846
```

```r
# Test error rate for best K
prediction3 <- knn(train_input, test_input, train_output, k=z)
confusion.test <- table(market.test$MARKET_STATUS, prediction3)
error.test <- 1 - (confusion.test[1,1] + confusion.test[2,2]) / sum(confusion.test)
cat("Test Error:", error.test, "\n")
```

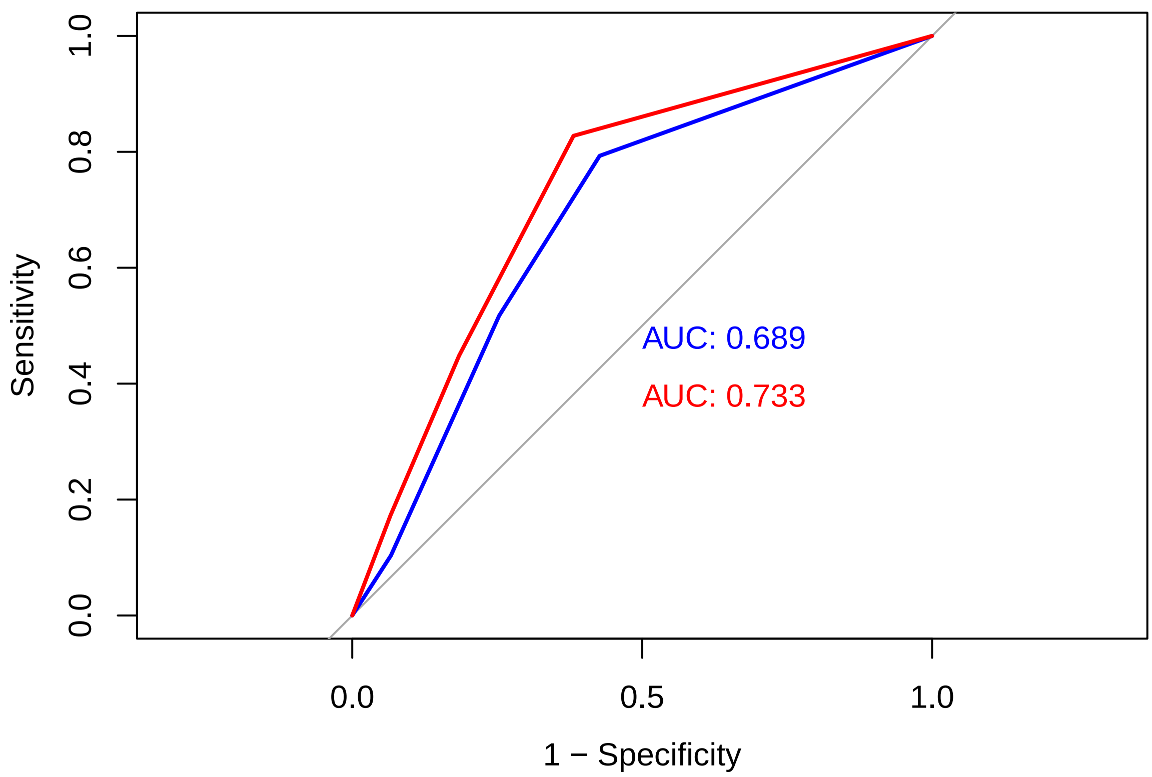
```
## Test Error: 0.08571429
```

```
# ROC Curves for best KNN model
actual.val <- market.val$MARKET_STATUS
actual.test <- market.test$MARKET_STATUS

prediction4 <- knn(train_input, validate_input, train_output, k=z, prob=TRUE)
predicted.probability <- attr(prediction4, "prob")
predicted.probability.knn <- ifelse(prediction4 == "Yes", predicted.probability, 1 - predicted.probabil
roc_rose <- plot(roc(actual.val, predicted.probability.knn), print.auc = TRUE, col = "blue", legacy.axe
```

```
## Setting levels: control = Cold, case = Hot

## Setting direction: controls < cases
```

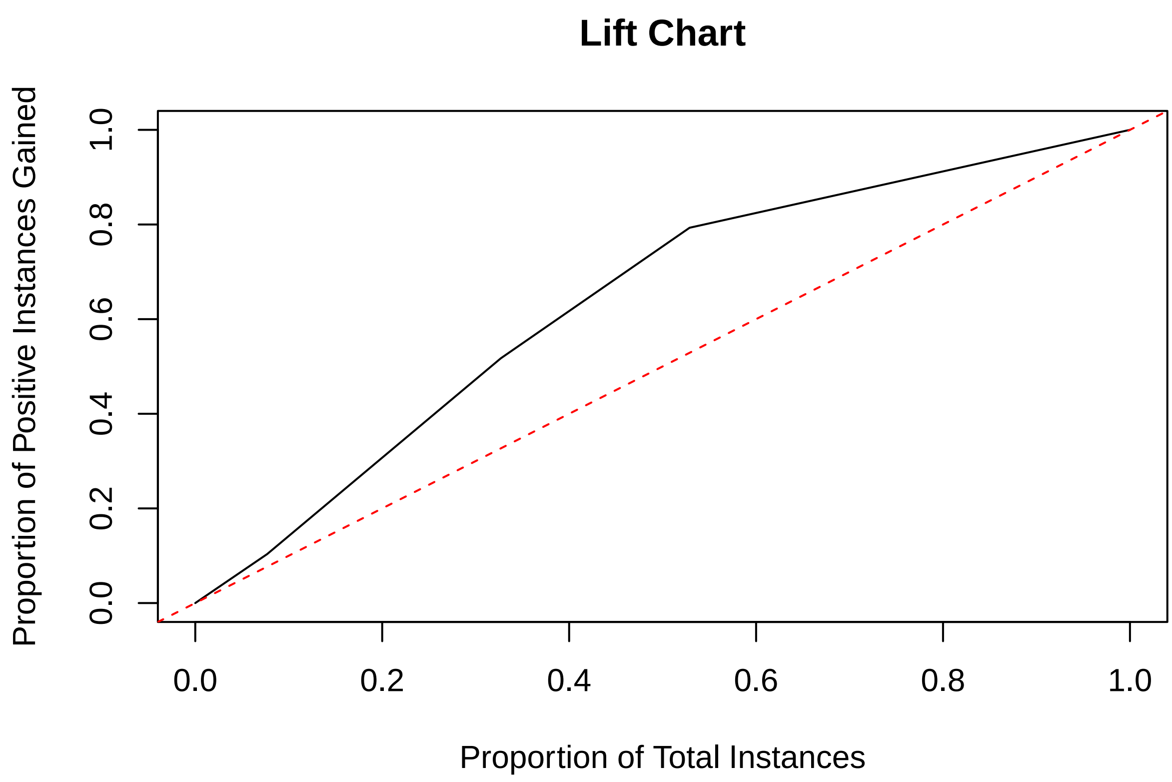
```
prediction5 <- knn(train_input, test_input, train_output, k=z, prob=TRUE)
predicted.probability <- attr(prediction5, "prob")
predicted.probability.test <- ifelse(prediction5 == "Yes", predicted.probability, 1 - predicted.probabil
roc_rose <- plot(roc(actual.test, predicted.probability.test), print.auc = TRUE, add=TRUE, col = "red",
```

```
## Setting levels: control = Cold, case = Hot
## Setting direction: controls < cases
```




```
# Lift Chart KNN
actuals <- market.val$MARKET_STATUS
```

```r
predictions <- predicted.probability.knn
pred <- prediction(predictions, actuals)
perf <- performance(pred, measure = "tpr", x.measure = "rpp")
plot(perf, colorize = FALSE, main = "Lift Chart", xlab="Proportion of Total Instances", ylab="Proportio
abline(a=0, b=1, lty=2, col="red")
```




```r
# Linear Probability Model (LPM) for Hot and Cold Classification
lpm.market.train <- market.train # to create new df for this model
lpm.market.val <- market.val
lpm.market.test <- market.test
lpm.market.train$MARKET_STATUS_n <- as.numeric(lpm.market.train$MARKET_STATUS) - 1
lpm.market.val$MARKET_STATUS_n <- as.numeric(lpm.market.val$MARKET_STATUS) - 1
lpm.market.test$MARKET_STATUS_n <- as.numeric(lpm.market.test$MARKET_STATUS) - 1
lpm.actual.val <- lpm.market.val$MARKET_STATUS
lpm.actual.test <- lpm.market.test$MARKET_STATUS
model <- lm(MARKET_STATUS_n~ . -MARKET_STATUS, data=lpm.market.train)

# The validation error rate for the LPM
predicted.probability.LPM <- predict(model, newdata=lpm.market.val)
lpm.predicted <- ifelse(predicted.probability.LPM > 0.5, 1, 0)
conf.LPM <- table(lpm.actual.val, lpm.predicted)
error.LPM <- 1 - (conf.LPM[1,1] + conf.LPM[2,2]) / sum(conf.LPM)
cat("The validation error rate for the LPM:", error.LPM, "\n")
```

```
## The validation error rate for the LPM: 0.125
```

```r
# The test error rate for the LPM
predicted.probability.LPM.test <- predict(model, newdata=lpm.market.test)
lpm.predicted.test <- ifelse(predicted.probability.LPM.test > 0.5, 1, 0)
conf.LPM <- table(lpm.actual.test, lpm.predicted.test)
error.LPM <- 1 - (conf.LPM[1,1] + conf.LPM[2,2]) / sum(conf.LPM)
cat("The test error rate for the LPM:", error.LPM, "\n")
```

```
## The test error rate for the LPM: 0.0952381
```

```r
# Logistic Regression Model for Hot and Cold Classification
fit <- glm(MARKET_STATUS ~ ., data=market.train, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
predicted.probability.LR <- predict(fit, type="response", newdata=market.val)
predicted <- ifelse(predicted.probability.LR > 0.5, 1, 0)
conf.LR <- table(actual.val, predicted)
error.LR <- 1 - (conf.LR[1,1] + conf.LR[2,2]) / sum(conf.LR)
cat("The validation error for LR is:", error.LR)
```

```
## The validation error for LR is: 0.1057692
```

```r
predicted.probability.LR.test <- predict(fit, type="response", newdata=market.test)
predicted <- ifelse(predicted.probability.LR.test > 0.5, 1, 0)
conf.LR <- table(actual.test, predicted)
error.LR <- 1 - (conf.LR[1,1] + conf.LR[2,2]) / sum(conf.LR)
cat("The test error for LR is:", error.LR)
```

```
## The test error for LR is: 0.07619048
```

```r
roc_rose <- plot(roc(actual.val, predicted.probability.knn), print.auc = TRUE, col = "cyan", legacy.axe
```

```
## Setting levels: control = Cold, case = Hot
```

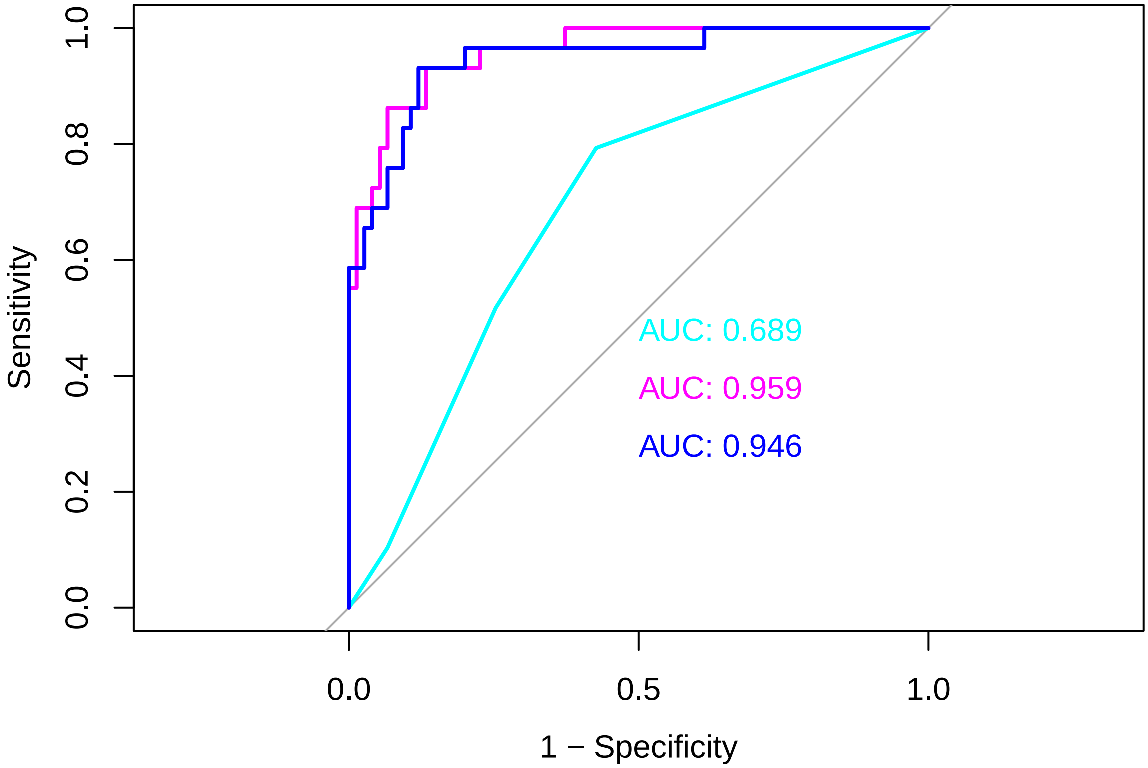
```
## Setting direction: controls < cases
```

```r
roc_rose <- plot(roc(actual.val, predicted.probability.LR), print.auc = TRUE, col = "magenta", legacy.a
```

```
## Setting levels: control = Cold, case = Hot
## Setting direction: controls < cases
```

```r
roc_rose <- plot(roc(actual.val, predicted.probability.LPM), print.auc = TRUE, col = "blue", legacy.axe
```

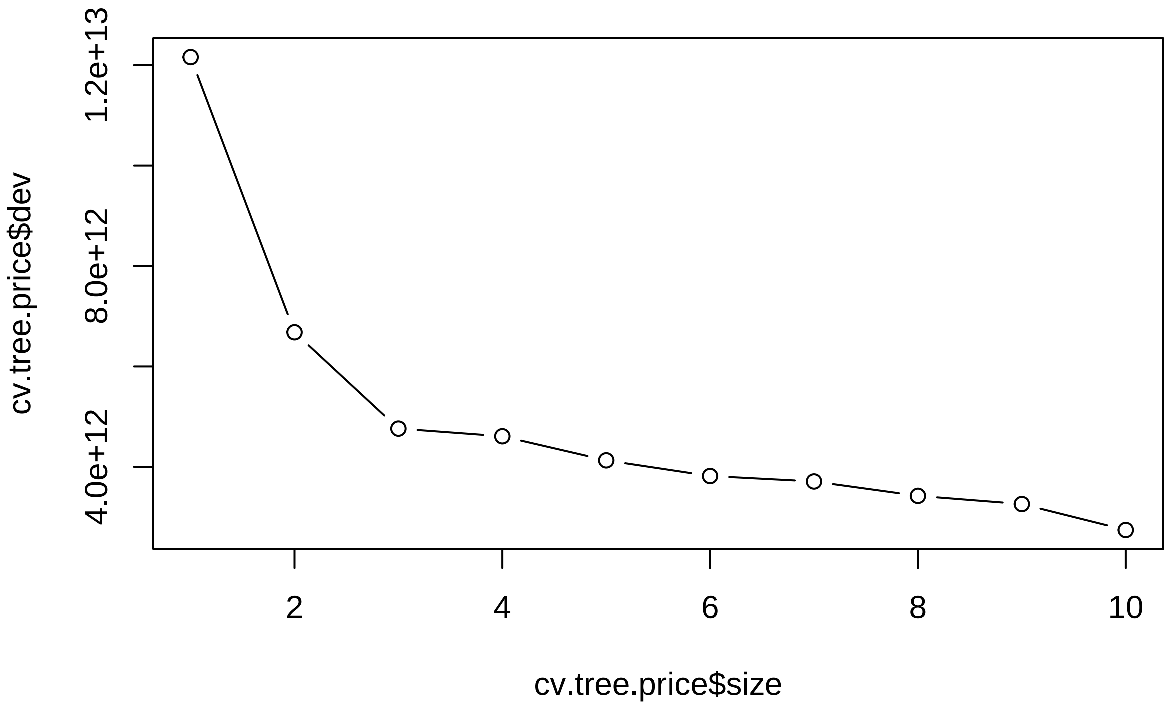
```
## Setting levels: control = Cold, case = Hot
## Setting direction: controls < cases
```



Tree (Predicting Sale Price)

- Very poor results on this tree for predicting price

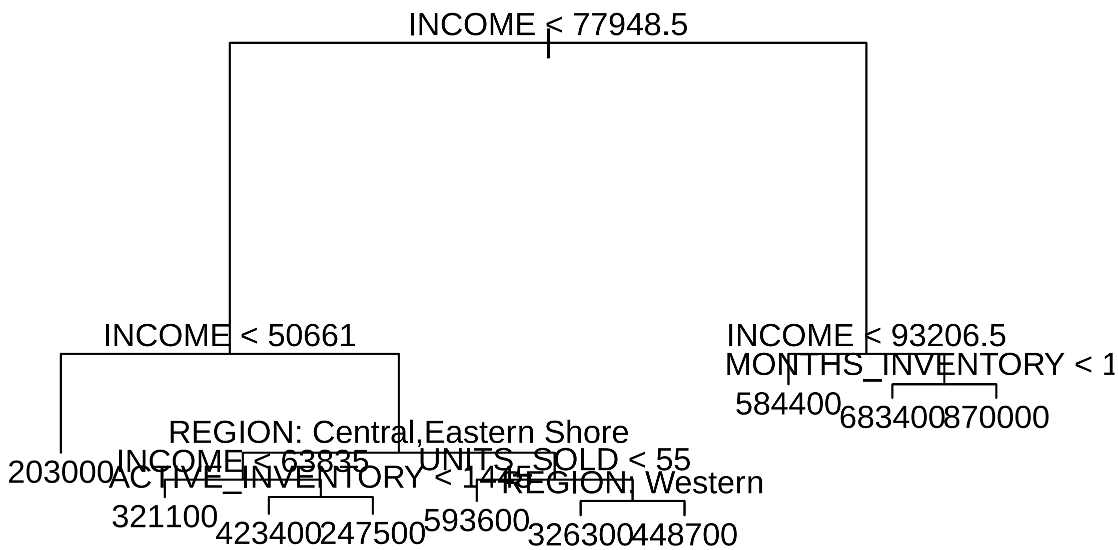
```
# Generate the best pruned regression tree model to predict price. Season subbed in for Month, Year, Qu
set.seed(123)
tree.price <- tree(AVG_SALE_PRICE~ REGION+UNITS_SOLD+UNITS_PENDING+ACTIVE_INVENTORY+MONTHS_INVENTORY+MED
cv.tree.price <- cv.tree(tree.price, FUN = prune.tree)
plot(cv.tree.price$size, cv.tree.price$dev, type ='b')
```




```r
# Find values of the minimum deviance
min_dev <- which(cv.tree.price$dev == min(cv.tree.price$dev))

# Extract the minimum corresponding size
min_sizes <- cv.tree.price$size[min_dev]
(min(min_sizes))
```

```
## [1] 10
```

```r
# Prune the tree by best = the min above
prune.price <- prune.tree(tree.price, best=10)
plot(prune.price)
text(prune.price, pretty=0)
```




```r
# Test Data
price.test <- test$AVG_SALE_PRICE
prune.pricepred <- predict(prune.price, test)

# Calculate MSE and RMSE for Test Data
mse_test_prune <- mean((prune.pricepred - price.test)^2)
rmse_test_prune <- sqrt(mse_test_prune)

# Calculate average house price for test data
avg_price_test <- mean(price.test)

# Normalize MSE and RMSE for Test Data
normalized_mse_test_prune <- mse_test_prune / avg_price_test * 100
normalized_rmse_test_prune <- rmse_test_prune / avg_price_test * 100

# Print normalized results for test data
cat("Test Normalized MSE (% of avg price):", normalized_mse_test_prune, "\n")
```

```
## Test Normalized MSE (% of avg price): 1023973
```

```r
cat("Test Normalized RMSE (% of avg price):", normalized_rmse_test_prune, "\n")
```

```
## Test Normalized RMSE (% of avg price): 15.63134
```

```r
# Training Data
prune.pricepred.train <- predict(prune.price, train)

# Calculate MSE and RMSE for Training Data
mse_train_prune <- mean((prune.pricepred.train - train$AVG_SALE_PRICE)^2)
rmse_train_prune <- sqrt(mse_train_prune)

# Calculate average house price for train data
avg_price_train <- mean(train$AVG_SALE_PRICE)

# Normalize MSE and RMSE for Training Data
normalized_mse_train_prune <- mse_train_prune / avg_price_train * 100
normalized_rmse_train_prune <- rmse_train_prune / avg_price_train * 100

# Print normalized results for training data
cat("Train Normalized MSE (% of avg price):", normalized_mse_train_prune, "\n")
```

```
## Train Normalized MSE (% of avg price): 973057.4
```

```r
cat("Train Normalized RMSE (% of avg price):", normalized_rmse_train_prune, "\n")
```

```
## Train Normalized RMSE (% of avg price): 15.0041
```

Tree (Classifying Hot and Cold Markets)

```r
# Using df from the GLM model
# Partition Data
set.seed(123)
inTrain2 <- sample(nrow(dfglm), 0.7 * nrow(df))
market.train <- dfglm[inTrain2, ]
temp2 <- dfglm[-inTrain2, ]
market.validation <- sample(nrow(temp2), 0.5 * nrow(temp2))
market.val <- temp2[market.validation, ]
market.test <- temp2[-market.validation, ]
rm(temp2)

# Create Deep Tree
tree.market <- tree(MARKET_STATUS~ ., market.train)
summary(tree.market)
```

```
##
## Classification tree:
## tree(formula = MARKET_STATUS ~ ., data = market.train)
## Variables actually used in tree construction:
## [1] "COUNTY"           "MONTH"            "MONTHS_INVENTORY" "UNITS_SOLD"
## Number of terminal nodes:  14
## Residual mean deviance:  0.374 = 176.9 / 473
## Misclassification error rate: 0.07392 = 36 / 487
```

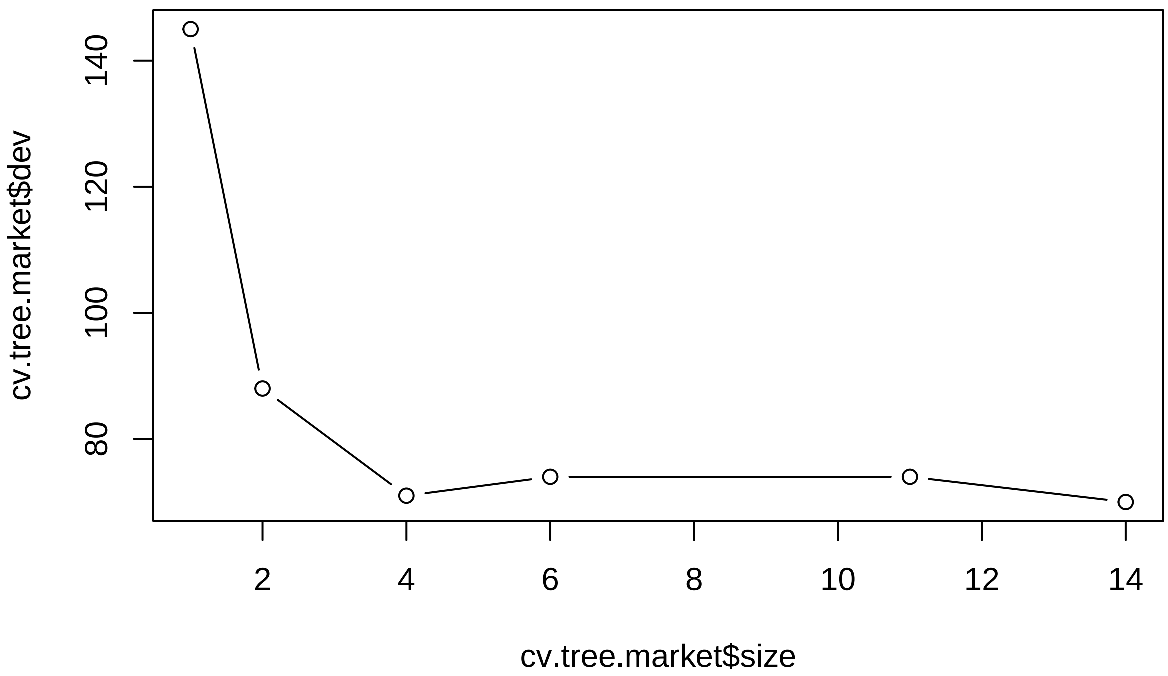
```r
# Plot
set.seed(123)
cv.tree.market <- cv.tree(tree.market, FUN = prune.misclass)
names(cv.tree.market)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
cv.tree.market
```

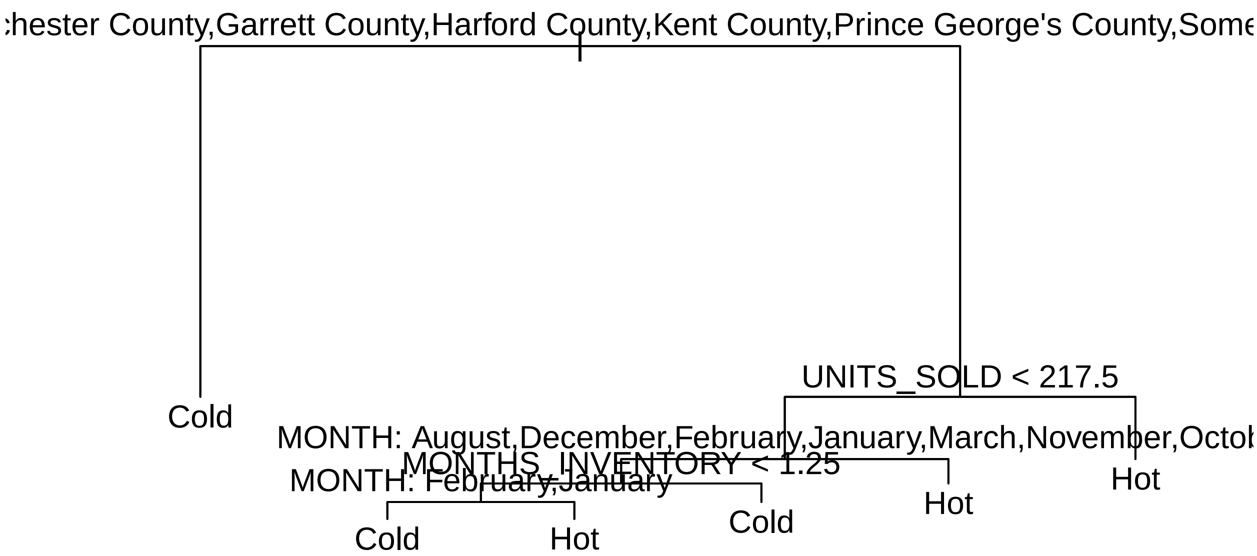
```
## $size
## [1] 14 11  6  4  2  1
##
## $dev
## [1]  70  74  74  71  88 145
##
## $k
## [1]      -Inf  1.666667  2.000000  3.500000 11.000000 65.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv.tree.market$size, cv.tree.market$dev, type="b")
```



Tree of size 4 delivers the minimum variance.

```r
# Create best prune tree
prune.market <- prune.misclass(tree.market, best=5)
plot(prune.market)
text(prune.market, pretty=0)
```




```r
# Test error rate
market.prunetree.pred <- predict(prune.market, market.test, type="class")
(CM <- table(market.test$MARKET_STATUS, market.prunetree.pred))
```

```
##       market.prunetree.pred
##        Cold Hot
##   Cold   72   4
##   Hot     7  22
```

```r
Acc <- (CM[1,1]+CM[2,2])/sum(CM)
cat("The test error for this Tree is:", 1-Acc)
```

```
## The test error for this Tree is: 0.1047619
```

```r
# Graph depicting the test ROC's for the classification tree and logistic regression model.
market.prunetree.pred <- as.numeric(market.prunetree.pred)
market.logistic_class.test <- factor(market.logistic_class.test)
```

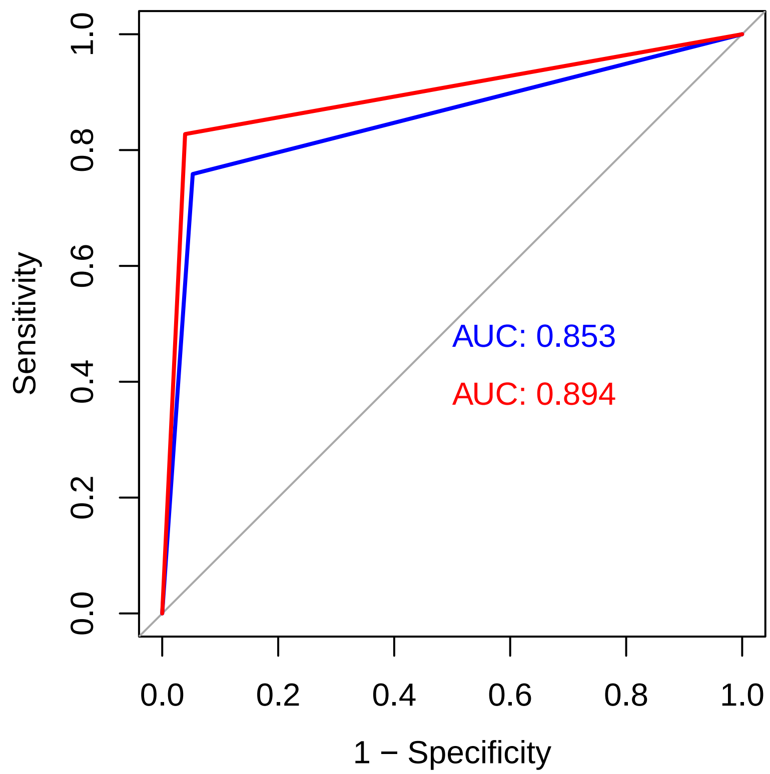
```r
market.logistic_class.test <- as.numeric(market.logistic_class.test)
par(pty="s")
roc_rose <- plot(roc(market.test$MARKET_STATUS, market.prunetree.pred), print.auc = TRUE, legacy.axes=T
```

```
## Setting levels: control = Cold, case = Hot
```

```
## Setting direction: controls < cases
```

```r
roc_rose <- plot(roc(market.test$MARKET_STATUS, market.logistic_class.test), print.auc = TRUE, legacy.a
```

```
## Setting levels: control = Cold, case = Hot
## Setting direction: controls < cases
```



Logistic Regression model is better at classifying hot and cold markets versus the classification tree.

Random Forest

```r
# Random Forest to market status.
set.seed(123)
rf.market2 <- randomForest(MARKET_STATUS~ ., data=market.train, mtry=4, importance=TRUE)
rf.market.test2 <- predict(rf.market2, newdata=market.test)
```

```r
rf.market.test2 <- as.numeric(rf.market.test2)
market.status.test <- as.numeric(market.test$MARKET_STATUS)

# test error (MSE)
mean((rf.market.test2-market.status.test)^2)
```

```
## [1] 0.06666667
```

```r
# matrix
(CM <- table(market.test$MARKET_STATUS, rf.market.test2))
```

```
##       rf.market.test2
##         1  2
##   Cold 76  0
##   Hot   7 22
```

```r
Acc <- (CM[1,1]+CM[2,2])/sum(CM)
TN <- CM[1,1]  # True Negative
FP <- CM[1,2]  # False Positive
FN <- CM[2,1]  # False Negative
TP <- CM[2,2]  # True Positive
cat("The test error for the Random Forest is:", 1-Acc)
```

```
## The test error for the Random Forest is: 0.06666667
```

```r
# Calculate Sensitivity (Recall or True Positive Rate)
sensitivity <- TP / (TP + FN)
print(paste("Sensitivity:", sensitivity))
```

```
## [1] "Sensitivity: 0.758620689655172"
```

```r
# Calculate Specificity (True Negative Rate)
specificity <- TN / (TN + FP)
print(paste("Specificity:", specificity))
```

```
## [1] "Specificity: 1"
```

```r
# Calculate Accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)
print(paste("Accuracy:", accuracy))
```

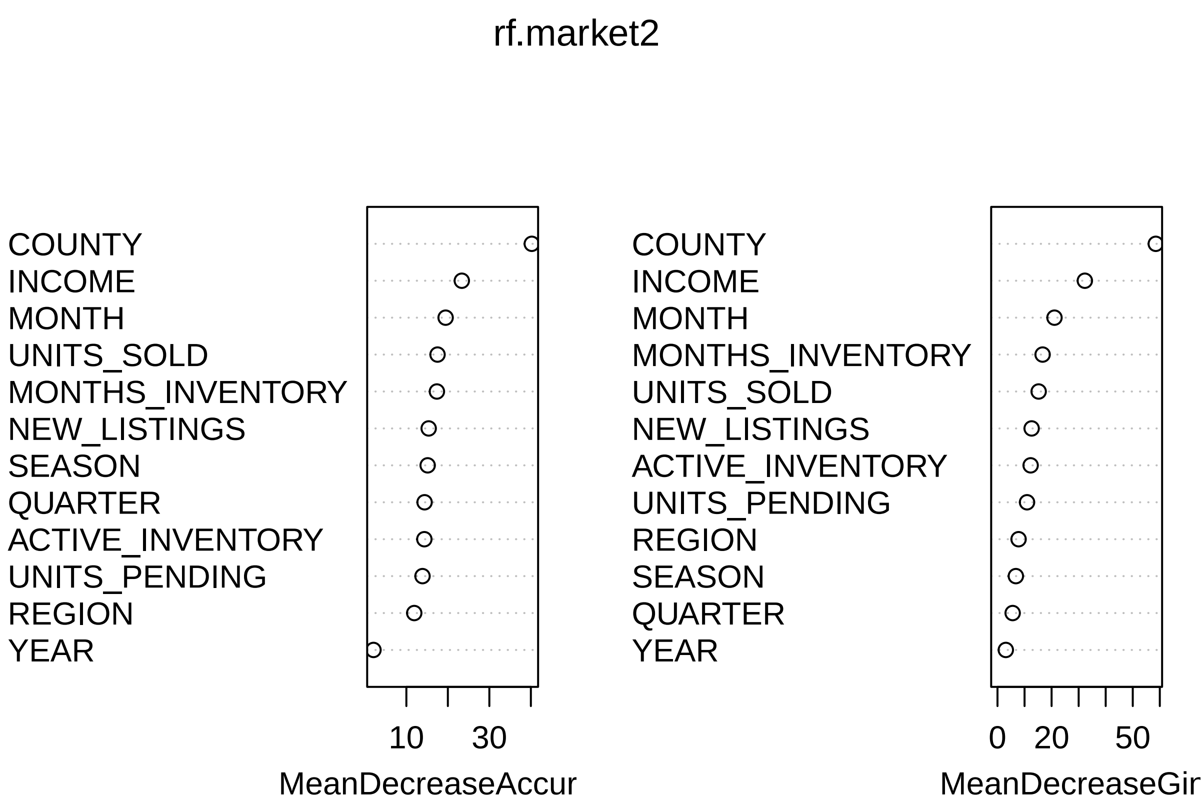
```
## [1] "Accuracy: 0.933333333333333"
```

```r
# Generate the variable importance plot
importance(rf.market2)
```

```
##                       Cold       Hot MeanDecreaseAccuracy MeanDecreaseGini
## COUNTY           40.415781 19.473424             40.21895        58.507280
## MONTH            17.546950 11.138589             19.45737        21.064349
```

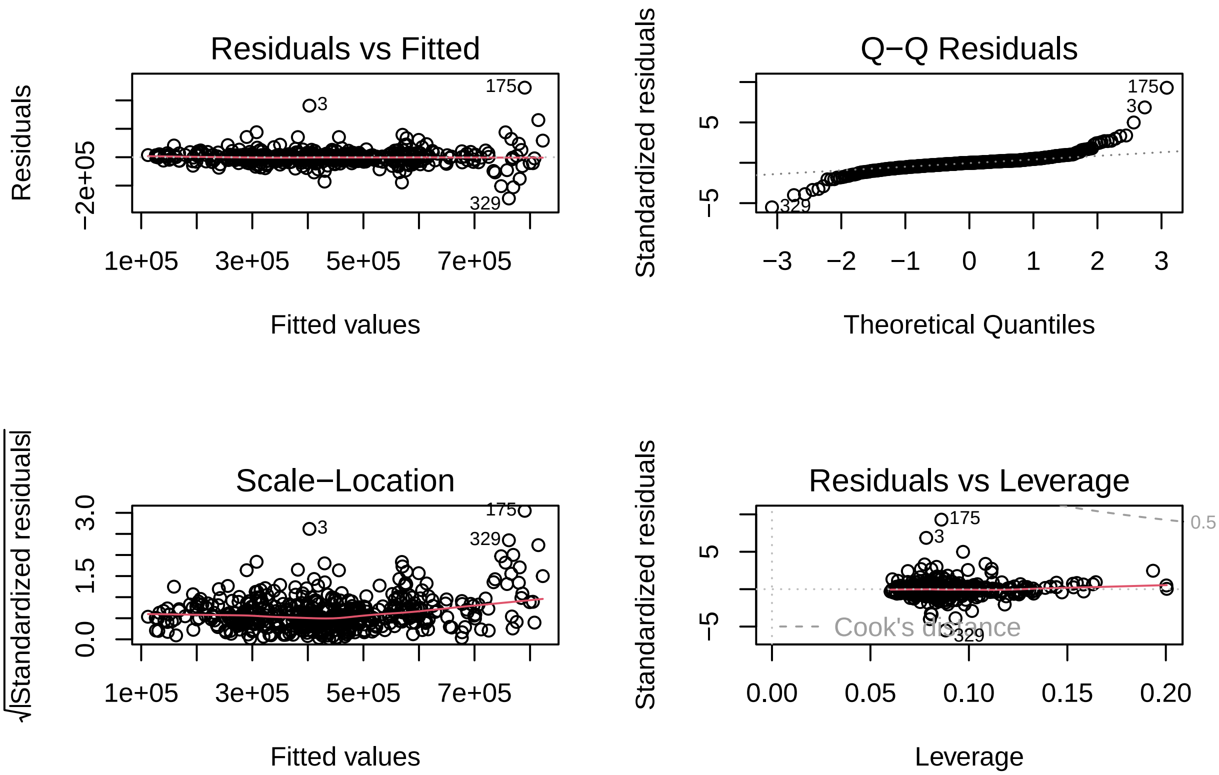
```
## YEAR               1.238327  2.025859             2.07443          3.078892
## UNITS_SOLD        11.766785 13.632697            17.51005         15.216927
## UNITS_PENDING      8.522010 10.902682            13.88392         10.922817
## ACTIVE_INVENTORY 11.871325  7.227284            14.33228         12.253079
## MONTHS_INVENTORY 12.601380 13.984442            17.35605         16.674921
## NEW_LISTINGS      13.005249  8.275982            15.37487         12.622023
## SEASON            12.096680 11.400797            15.13271          6.790103
## QUARTER           12.372154  9.128809            14.38734          5.606860
## INCOME            20.981598 17.571271            23.36175         32.291901
## REGION            11.307684  3.849822            11.89614          7.796991
```

```r
varImpPlot(rf.market2)
```

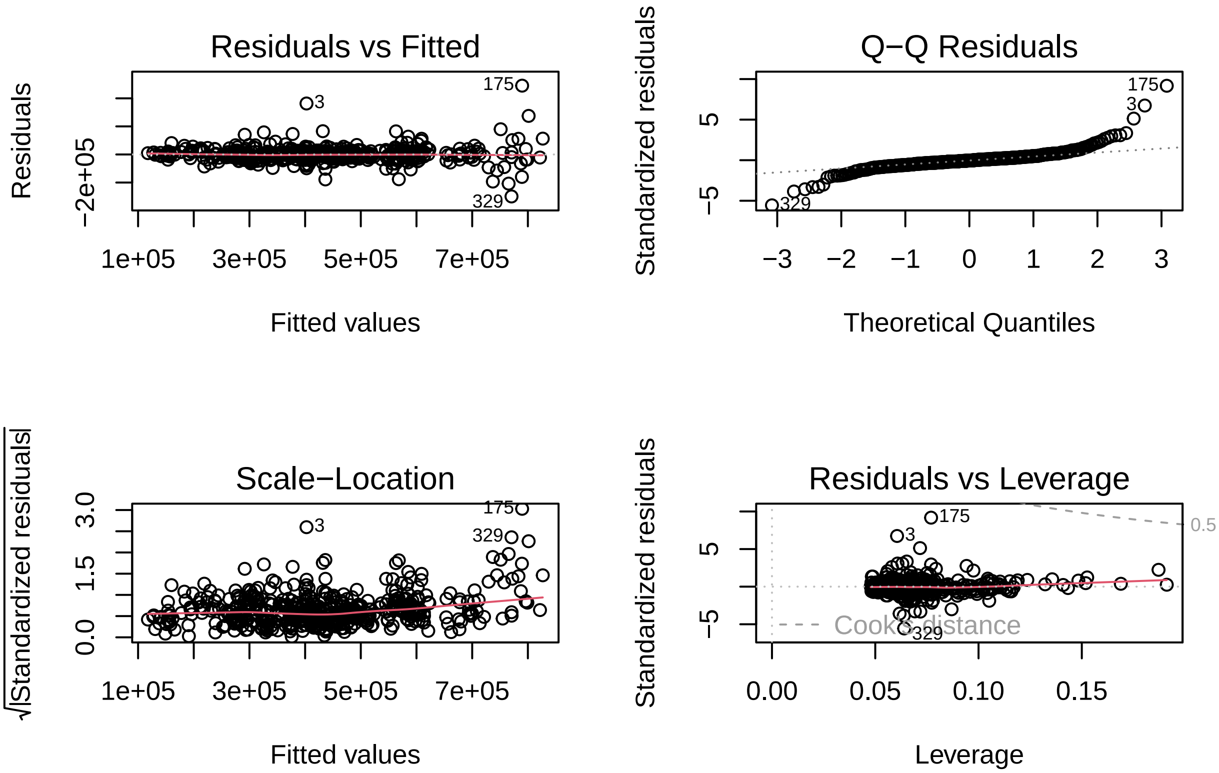


Assumption Checking

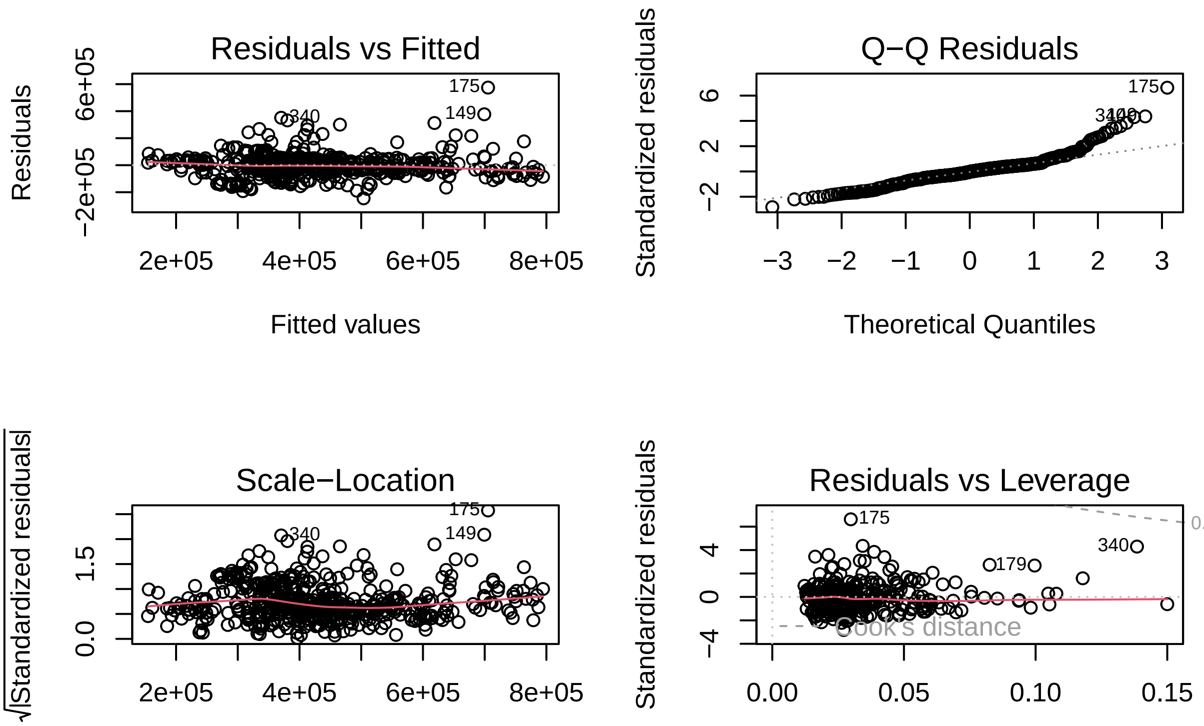
```r
# Check our linearity assummption
par(mfrow = c(2, 2))  # 2x2 plot layout
plot(LRMprice0)
```




```r
# Diagnostic plots for LRMprice1
plot(LRMprice1)
```




```r
# Diagnostic plots for LRMprice2
plot(LRMprice2)
```



Overall Evaluation: Homoscedasticity: The residuals vs. fitted and scale-location plots both suggest that the assumption of homoscedasticity is met, as there is no clear pattern or funnel shape in the residuals. Linearity: The residuals vs. fitted plot does not show any systematic patterns, which suggests that the linearity assumption is reasonably met. Normality of Residuals: The Q-Q plot indicates that the residuals are approximately normally distributed, although there are some deviations at the tails, which may warrant further investigation. Influential Points: The residuals vs. leverage plot shows a few potentially influential points, but overall, the model does not seem to be unduly influenced by these observations.