Homework 5 - Stat 495

Cassandra Jin

Due Friday, October 27th by midnight

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook(s), course materials in Moodle/Git repo, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

-

Homework 5 - Prompt - Write-up for Homework 4

```
kchouse <- read.csv("https://awagaman.people.amherst.edu/stat495/kc_house_data.csv", header = T)
```

Motivation to predict when Price is greater than $500,000

A real estate developer is interested in understanding the features that are predictive of homes selling for more than half a million dollars in this area of Washington, and has turned to you, a statistical consultant for help. The developer wants a model that can be applied to make predictions in this setting and wants to understand how the variables in the model are impacting it.

To practice new techniques from class, in your analysis, you are required to use both an appropriate generalized linear model and decision tree to address the developer's questions of interest, including a model comparison.

Write-up Purpose and some Guidance

We want to practice reporting our analyses. You have been contracted to do an analysis, and you prepared some potential models in Homework 4, and now need to report your process and subsequent findings to the client.

Things to think about:

- Your report should be well-organized and proofread.
- Your code should be readable and not run off pages. Employ good commenting practices.
- All code must be included (class requirement).
- The entire process must be reproducible.
- Review the class feedback on Homework 4 that will be provided and adjust your models/thinking as necessary before diving in to the write-up.
- Think carefully about the organization of the report. Is it useful for the reader to have pages of output and THEN a description of what to look for in that output?
- Should you assume the reader / audience here knows where values in the R output are?
- If you compute a number, and then write a sentence that says, the MSE is small, will the reader associate that number with being the MSE? What would a better sentence look like?
- Thinking back to the Tibshirani Writing activity may help you think about ways to improve as well. (Does each of your sections have a purpose? Do you have appropriate signposts, etc.)

Instructions for your submission

Your submission should include the following sections (along with all code necessary to reproduce your results):

- Introduction and Exploratory Data Analysis - could be two separate sections
- Your final GLM and relevant details
- Your final Tree and relevant details (can be before or after your GLM)
- Your Model Comparison (can be woven in sections above)
- Your Conclusion

Introduction

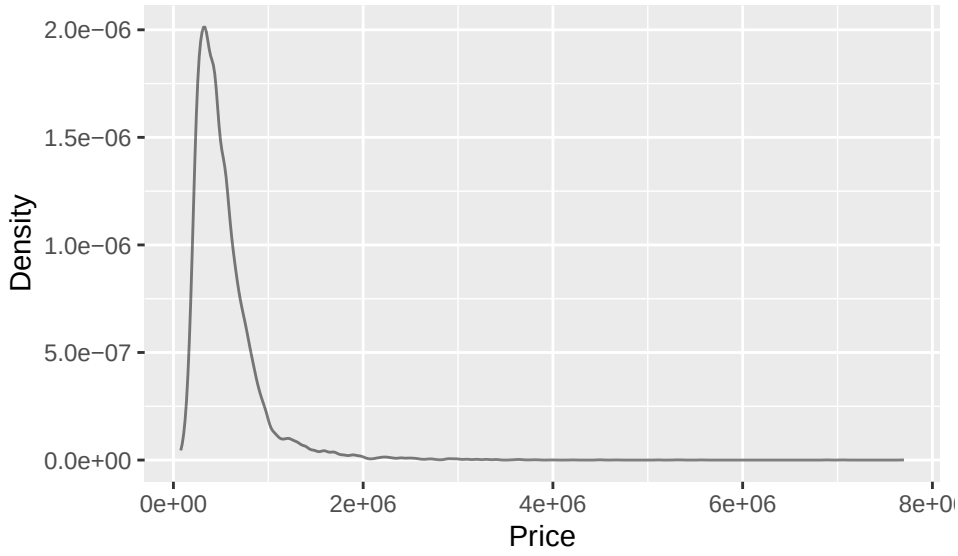
In today's dynamic real estate market, it is important to gain any insights possible into the intricate factors that might influence the sale prices of homes. Particularly, we have been tasked with providing comprehensive analysis of a data set of house sales in King County, Washington, with the goal of constructing a predictive model for when a house exceeds the threshold of half a million dollars based on certain property features.

The King County, Washington (State) house sales data set contains information on over 20,000 homes in the region, including details about their sale price, size, attributes, location, condition, and more. In order to understand which of these variables are individually appropriate to use for analysis, we will first perform exploratory data analysis on them and apply data transformations or completely eliminate uninformative variables. Then, to actually predict the binary response variable of whether or not a home sale price exceeds half a million dollars, we will build two models: a generalized linear model (GLM) and a decision tree model. The final GLM will include the full set of predictors and will output whether or not a predicted house price will be greater than $500,000. The decision tree model will offer a more visually-oriented map to follow to a conclusion based on the values of fewer input variables than the GLM. Thus, each model includes slightly different subsets of predictors but both offer large degrees of interpretability and accuracy in prediction. We will then go on to investigate the predictive power of the variables we choose to include with confusion matrices and more.

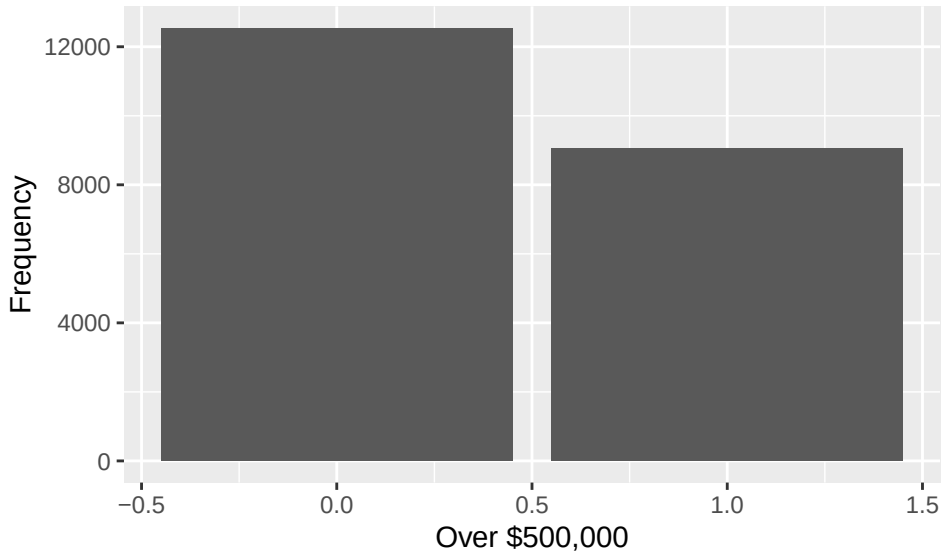
Through this report, we aim to equip the real estate developer with the necessary tools and insights to make informed decisions in the dynamic and competitive real estate market of this area in Washington.

Exploratory Data Analysis

```
# original density distribution of response variable, price
gf_dens(~ price, data = kchouse,
              ylab = "Density",
              xlab = "Price")
```




```
kchouse <- kchouse %>%
  # make binary variable for Price > $500000
  mutate(overhalfmil = ifelse(price > 500000, 1, 0)) %>% # 1=yes, 0=no
  # make binary variables for some quantitative ones, since most observations are 0
  mutate(basement = ifelse(sqft_basement > 0, 1, 0),
         renovated = ifelse(yr_renovated > 0, 1, 0)) %>%
  # log-transform variables to fix skewness
  mutate(log_sqft_living = log(sqft_living),
         log_sqft_lot = log(sqft_lot),
         log_sqft_above = log(sqft_above),
         log_sqft_living15 = log(sqft_living15),
         log_sqft_lot15 = log(sqft_lot15)) %>%
  # drop pre-transformed variables
  select(-price, -sqft_living, -sqft_lot, -sqft_above, -sqft_living15,
         -sqft_lot15, -sqft_basement, -yr_renovated, -id, -date) %>%
  # remove outlier where bedrooms = 33
  subset(bedrooms != 33) %>%
  # make binary variable for when condition = 5
  mutate(condition5 = ifelse(condition == 5, 1, 0))

# bar graph and tally for Price > $500000 frequency
gf_bar(~ overhalfmil, data = kchouse,
       ylab = "Frequency",
       xlab = "Over $500,000")
```

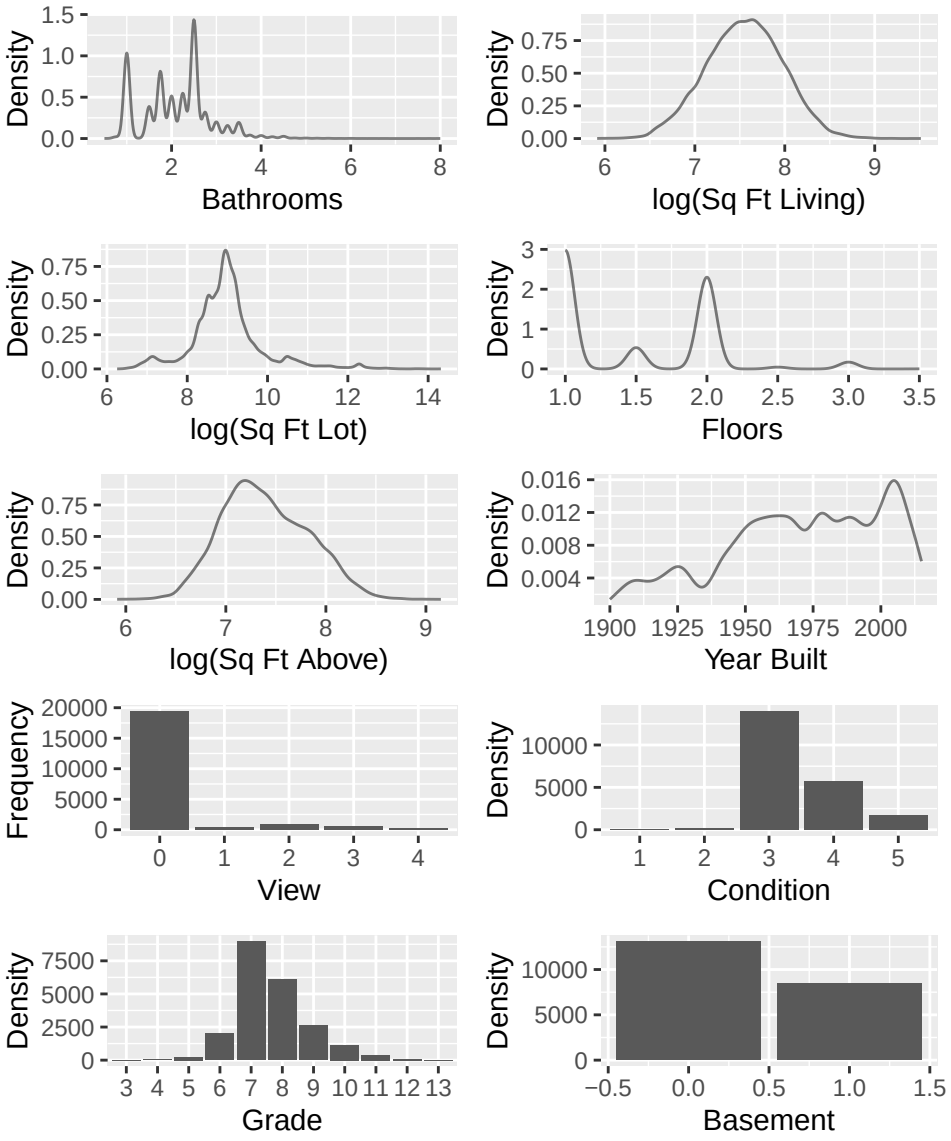



```
tally(~ overhalfmil, format = "percent", data = kchouse)
```

```
## overhalfmil
##       0       1
## 58.0941 41.9059
```

```
# density plots for quantitative variables
p1 <- gf_dens(~ bathrooms, data = kchouse,
              ylab = "Density",
              xlab = "Bathrooms")
p2 <- gf_dens(~ log_sqft_living, data = kchouse,
              ylab = "Density",
              xlab = "log(Sq Ft Living)")
p3 <- gf_dens(~ log_sqft_lot, data = kchouse,
              ylab = "Density",
              xlab = "log(Sq Ft Lot)")
p4 <- gf_dens(~ floors, data = kchouse,
              ylab = "Density",
              xlab = "Floors")
p5 <- gf_dens(~ log_sqft_above, data = kchouse,
              ylab = "Density",
              xlab = "log(Sq Ft Above)")
p6 <- gf_dens(~ yr_built, data = kchouse,
              ylab = "Density",
              xlab = "Year Built")

# bar graphs for qualitative variables
p7 <- gf_bar(~ as.factor(view), data = kchouse,
             ylab = "Frequency",
             xlab = "View")
p8 <- gf_bar(~ as.factor(condition), data = kchouse,
              ylab = "Density",
              xlab = "Condition")
p9 <- gf_bar(~ as.factor(grade), data = kchouse,
              ylab = "Density",
              xlab = "Grade")
p10 <- gf_bar(~ basement, data = kchouse,
```

```
              ylab = "Density",
              xlab = "Basement")

# plot above plots in a grid
cowplot::plot_grid(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, ncol = 2)
```



Here, we have exploratory data analysis (EDA) for the response variable and only the predictors that will be included in the final GLM and decision tree model.

We choose to apply the log transform to a few of the variables, which all exhibit extreme skewness. Doing so does make the model slightly more complicated to interpret, but for the sake of predictability, our choice to transform the variables and thus push them into more usable distributions with more convenient spread will allow the later model to be fit more reliably on the data (Note: although the the variable `sqft_basement` is left-skewed, we cannot apply the log transform because the values are 0 for some observations). Since `waterfront` and `renovated` have so many observations of one level and so few of the other, we may consider getting rid of them in the model fitting later on.

Before diving into any analysis, it is preferred to have independent observations. But since our data set revolves around houses, many of which are likely in neighborhoods, we cannot be sure that we are working

with independent observations of the response variable or any of the variables. However, it is still worth considering how fitted models of the complete, untouched data set may be able to help us predict whether or not a house costs over $500,000.

To additionally prepare the data set for steps after we have constructed the models, we create a 0.75/0.25 training/test split. As the name suggests, the training set is used to train the model, and the testing set is then used to assess how well the model does by serving as "new," unseen data. This gives an estimate of how the model will perform in real-world scenarios, so we will be able to determine how many houses the GLM and decision tree model correctly predict to cost over $500,000.

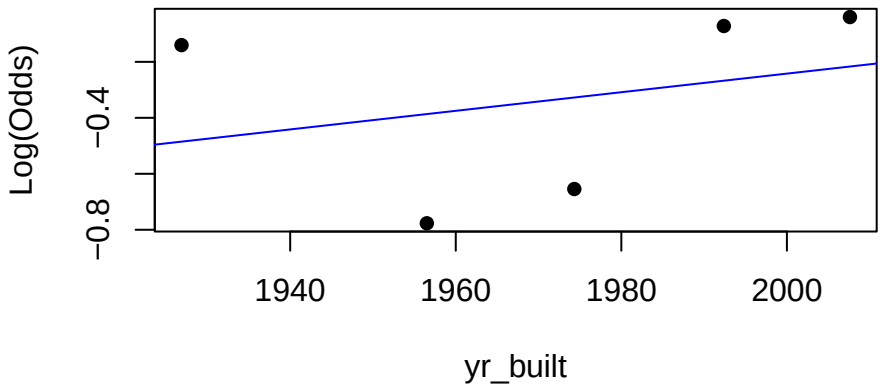
```
set.seed(495)
n <- nrow(kchouse)

# split data into training (0.75) and test (0.25) set
train_index <- sample(1:n, 0.75 * n)
test_index <- setdiff(1:n, train_index)
train <- kchouse[train_index, ]
test <- kchouse[test_index, ]
```

GLM and Results

Here, we build a generalized linear model to perform logistic regression. The GLM is a useful model for this investigation because it extends the linear regression model to accommodate a broader range of data types and distributions, allowing us to consider predictor variables that are categorical, like the condition rating of a house observation and whether or not it is located on a waterfront.

Upon checking the conditions for logistic regression, we find that most are satisfied. Specifically, one of the conditions is that the logit-transformed response probability (i.e. the log odds of response) must be linear relative to the predictors. Most of the variables satisfy this by demonstrating linear relationships in their empirical logit plots (not included), but the variable `yr_built` raises slight concern (shown below).

```
# empirical logit plot
emplogitplot1(overhalfmil ~ yr_built, ngroups = 5, data = kchouse)
```



However, we first leave `yr_built` to see if it may be a significant predictor in the model, and we proceed with caution.

We start with an initial kitchen-sink model, meaning that it contains all possible predictor variables for overhalfmil, our response variable of whether or not a house costs more than $500,000.

```
# kitchen-sink logistic regression model
kslogmodel <- glm(overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
                    floors + log_sqft_above + basement + renovated + yr_built + zipcode +
                    log_sqft_living15 + log_sqft_lot15 + lat + long + waterfront +
                    as.factor(view) + as.factor(condition) + as.factor(grade),
                  data = train, family = binomial(logit))
msummary(kslogmodel)
```

```
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -5.87e+01   3.96e+03   -0.01  0.98817
## bedrooms              -1.88e-01   3.56e-02   -5.27  1.4e-07 ***
## bathrooms              2.62e-01   6.08e-02    4.30  1.7e-05 ***
## log_sqft_living        1.42e+00   2.38e-01    5.98  2.2e-09 ***
## log_sqft_lot           1.11e-01   7.42e-02    1.50  0.13395
## floors                 6.02e-01   6.82e-02    8.83  < 2e-16 ***
## log_sqft_above         8.25e-01   2.34e-01    3.53  0.00041 ***
## basement               5.19e-01   1.02e-01    5.10  3.4e-07 ***
## renovated              2.20e-01   1.35e-01    1.62  0.10427
## yr_built              -3.64e-02   1.41e-03  -25.85  < 2e-16 ***
## zipcode               -3.14e-03   6.42e-04   -4.90  9.8e-07 ***
## log_sqft_living15      1.88e+00   1.34e-01   14.06  < 2e-16 ***
## log_sqft_lot15        -3.44e-01   8.08e-02   -4.25  2.1e-05 ***
```

```
## lat                    8.88e+00   2.26e-01   39.24  < 2e-16 ***
## long                   2.24e-01   2.51e-01    0.89  0.37278
## waterfront             2.06e+00   6.44e-01    3.19  0.00142 **
## as.factor(view)1       1.10e+00   2.16e-01    5.12  3.1e-07 ***
## as.factor(view)2       6.08e-01   1.25e-01    4.87  1.1e-06 ***
## as.factor(view)3       1.12e+00   2.16e-01    5.20  2.0e-07 ***
## as.factor(view)4       2.10e+00   4.86e-01    4.33  1.5e-05 ***
## as.factor(condition)2  4.23e-01   7.54e-01    0.56  0.57474
## as.factor(condition)3  5.94e-01   6.74e-01    0.88  0.37832
## as.factor(condition)4  9.02e-01   6.74e-01    1.34  0.18072
## as.factor(condition)5  1.36e+00   6.78e-01    2.01  0.04422 *
## as.factor(grade)4     -3.28e+00   4.02e+03    0.00  0.99935
## as.factor(grade)5      1.12e+01   3.96e+03    0.00  0.99775
## as.factor(grade)6      1.05e+01   3.96e+03    0.00  0.99789
## as.factor(grade)7      1.16e+01   3.96e+03    0.00  0.99765
## as.factor(grade)8      1.30e+01   3.96e+03    0.00  0.99738
## as.factor(grade)9      1.44e+01   3.96e+03    0.00  0.99709
## as.factor(grade)10     1.56e+01   3.96e+03    0.00  0.99685
## as.factor(grade)11     2.81e+01   3.96e+03    0.01  0.99435
## as.factor(grade)12     2.74e+01   3.98e+03    0.01  0.99451
## as.factor(grade)13     2.66e+01   4.14e+03    0.01  0.99488
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22037  on 16196  degrees of freedom
## Residual deviance: 10864  on 16163  degrees of freedom
## AIC: 10932
##
## Number of Fisher Scoring iterations: 16
```

```
# likelihood ratio test
lrtest(kslogmodel)
```

```
## Likelihood ratio test
##
## Model 1: overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
##     floors + log_sqft_above + basement + renovated + yr_built +
##     zipcode + log_sqft_living15 + log_sqft_lot15 + lat + long +
##     waterfront + as.factor(view) + as.factor(condition) + as.factor(grade)
## Model 2: overhalfmil ~ 1
##   #Df LogLik  Df Chisq Pr(>Chisq)
## 1  34  -5432
## 2   1 -11018 -33 11173     <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# stepAIC
priceStep <- MASS::stepAIC(kslogmodel, trace = FALSE, direction = "both")
priceStep$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
##     floors + log_sqft_above + basement + renovated + yr_built +
```

```
##     zipcode + log_sqft_living15 + log_sqft_lot15 + lat + long +
##     waterfront + as.factor(view) + as.factor(condition) + as.factor(grade)
##
## Final Model:
## overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
##     floors + log_sqft_above + basement + renovated + yr_built +
##     zipcode + log_sqft_living15 + log_sqft_lot15 + lat + waterfront +
##     as.factor(view) + as.factor(condition) + as.factor(grade)
##
##
##     Step Df Deviance Resid. Df Resid. Dev     AIC
## 1                        16163    10863.5 10931.5
## 2 - long  1 0.793434     16164    10864.3 10930.3
```

Applying the likelihood ratio test to the kitchen-sink model, we find that the overall model is significant. But based on the output for this kitchen-sink logistic model, we can eliminate the variables `log_sqft_lot`, `renovated`, `long`, `grade`, and all `condition` levels besides 5, because the p-values of their regression coefficients are not significant. Running stepAIC on the model also results in `long` being left out, and using LASSO on the model drives the coefficients for bedrooms, zipcode, log_sqft_above to 0 (but this results in a lower rate of successful predictions).

```
# reduced logistic model
logmod <- glm(overhalfmil ~ bedrooms + bathrooms + log_sqft_living + floors +
                log_sqft_above + basement + yr_built + zipcode + log_sqft_living15 +
                log_sqft_lot15 + lat + waterfront + as.factor(view) + condition5,
              data = train, family = binomial(logit))
msummary(logmod)
```

```
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.09e+02   5.24e+01   -2.07   0.0386 *
## bedrooms          -3.42e-01   3.35e-02  -10.21  < 2e-16 ***
## bathrooms          3.91e-01   5.78e-02    6.77  1.3e-11 ***
## log_sqft_living    1.89e+00   2.28e-01    8.28  < 2e-16 ***
## floors             7.56e-01   6.47e-02   11.69  < 2e-16 ***
## log_sqft_above     1.75e+00   2.22e-01    7.89  3.1e-15 ***
## basement           7.11e-01   9.79e-02    7.26  3.9e-13 ***
## yr_built          -2.91e-02   1.17e-03  -24.79  < 2e-16 ***
## zipcode           -2.99e-03   5.32e-04   -5.61  2.0e-08 ***
## log_sqft_living15  2.62e+00   1.22e-01   21.46  < 2e-16 ***
## log_sqft_lot15    -2.28e-01   3.63e-02   -6.27  3.6e-10 ***
## lat                8.66e+00   2.08e-01   41.68  < 2e-16 ***
## waterfront         1.57e+00   5.99e-01    2.62   0.0088 **
## as.factor(view)1   1.12e+00   2.04e-01    5.47  4.6e-08 ***
## as.factor(view)2   7.92e-01   1.18e-01    6.73  1.7e-11 ***
## as.factor(view)3   1.29e+00   2.01e-01    6.41  1.5e-10 ***
## as.factor(view)4   2.35e+00   4.38e-01    5.37  7.9e-08 ***
## condition5         5.68e-01   9.05e-02    6.27  3.5e-10 ***
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22037  on 16196  degrees of freedom
## Residual deviance: 11995  on 16179  degrees of freedom
## AIC: 12031
##
```

```
## Number of Fisher Scoring iterations: 6
```

```
anova(logmod, kslogmodel, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: overhalfmil ~ bedrooms + bathrooms + log_sqft_living + floors +
##     log_sqft_above + basement + yr_built + zipcode + log_sqft_living15 +
##     log_sqft_lot15 + lat + waterfront + as.factor(view) + condition5
## Model 2: overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
##     floors + log_sqft_above + basement + renovated + yr_built +
##     zipcode + log_sqft_living15 + log_sqft_lot15 + lat + long +
##     waterfront + as.factor(view) + as.factor(condition) + as.factor(grade)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1     16179      11995
## 2     16163      10864 16     1131   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova table shows us that the reduced model is a more well-fitted model than the kitchen-sink model.

```
logmaugment <- augment(kslogmodel, type.predict = "response")
logmaugment <- mutate(logmaugment, binprediction = round(.fitted, 0))
with(logmaugment, table(overhalfmil, binprediction))
```

```
##            binprediction
## overhalfmil    0    1
##           0 8276 1119
##           1 1409 5393
```

```
# proportion correctly predicted with kitchen-sink model
(8276+5393)/(8276+5393+1119+1409)
```

```
## [1] 0.843922
```

```
logmaugment <- augment(logmod, type.predict = "response")
logmaugment <- mutate(logmaugment, binprediction = round(.fitted, 0))
with(logmaugment, table(overhalfmil, binprediction))
```

```
##            binprediction
## overhalfmil    0    1
##           0 8108 1287
##           1 1535 5267
```

```
# proportion correctly predicted with reduced model
(8108+5267)/(8108+5267+1287+1535)
```

```
## [1] 0.82577
```

Neither the kitchen-sink model nor the reduced model do a good job at making predictions, even on the training set, but we see that the kitchen-sink model predicts whether a house will be over $500,000 better than our reduced, since $0.844 > 0.826$. For the sake of predictability, we proceed with the full kitchen-sink logistic model and check one more metric of fit.

```
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE***#
Association <- function(model)
{
  Con_Dis_Data <- cbind(model$y, model$fitted.values)
  ones <- Con_Dis_Data[Con_Dis_Data[, 1] == 1, ]
```

```
  zeros <- Con_Dis_Data[Con_Dis_Data[, 1] == 0, ]
  conc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  disc <- matrix(0, dim(zeros)[1], dim(ones)[1])
  ties <- matrix(0, dim(zeros)[1], dim(ones)[1])
    for (j in 1:dim(zeros)[1])
    {
      for (i in 1:dim(ones)[1])
      {
        if (ones[i, 2] > zeros[j, 2])
        {conc[j, i] = 1}
        else if (ones[i, 2] < zeros[j, 2])
        {disc[j, i] = 1}
        else if (ones[i, 2] == zeros[j, 2])
        {ties[j, i] = 1}
      }
    }
  Pairs <- dim(zeros)[1]*dim(ones)[1]
  PercentConcordance <- (sum(conc)/Pairs)*100
  PercentDiscordance <- (sum(disc)/Pairs)*100
  PercentTied <- (sum(ties)/Pairs)*100
  return(list("Percent Concordance" = PercentConcordance,
              "Percent Discordance" = PercentDiscordance,
              "Percent Tied" = PercentTied, "Pairs" = Pairs))
}
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE ENDS***#

Association(kslogmodel)
```

```
## $`Percent Concordance`
## [1] 92.7642
##
## $`Percent Discordance`
## [1] 7.23574
##
## $`Percent Tied`
## [1] 3.5991e-05
##
## $Pairs
## [1] 63904790
```

Compared to a 50% likelihood of success from a coin flip, the model concordance being 92.8% indicates that the model fits reasonably well.

Tree and Results

Another method of predicting our response variable is building a decision tree. This is a graphical (and thus slightly easier to interpret) representation of a decision-making process, where each node of the tree represents a specific feature, and the branches that connect these nodes represent the possible outcomes of the decision. At the bottom leaf nodes of the tree, we arrive at the final predictions.

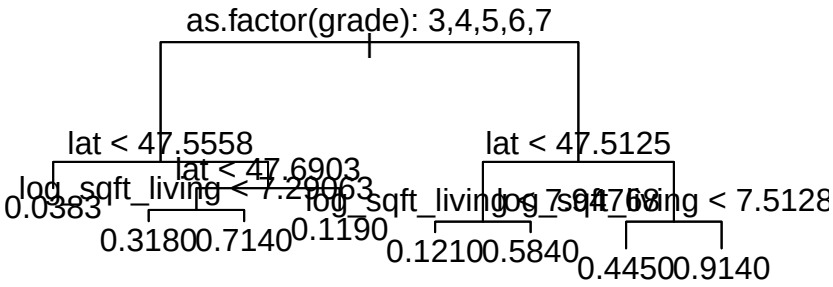
In this context, we construct a classification tree (rather than regression), because we are trying to predict a binary outcome–the house price is either over $500,000 or it is not. First, we will try with the *tree* package.

```
set.seed(495)

tree_house <- tree(overhalfmil ~ bedrooms + bathrooms + log_sqft_living + log_sqft_lot +
                     floors + log_sqft_above + basement + renovated + yr_built + zipcode +
                     log_sqft_living15 + log_sqft_lot15 + lat + long + waterfront +
                     as.factor(view) + as.factor(condition) + as.factor(grade), train)
summary(tree_house)
```

```
##
## Regression tree:
## tree(formula = overhalfmil ~ bedrooms + bathrooms + log_sqft_living +
##     log_sqft_lot + floors + log_sqft_above + basement + renovated +
##     yr_built + zipcode + log_sqft_living15 + log_sqft_lot15 +
##     lat + long + waterfront + as.factor(view) + as.factor(condition) +
##     as.factor(grade), data = train)
## Variables actually used in tree construction:
## [1] "as.factor(grade)" "lat"              "log_sqft_living"
## Number of terminal nodes:  8
## Residual mean deviance:  0.109 = 1770 / 16200
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.9140 -0.1190 -0.0383  0.0000  0.0862  0.9620
```

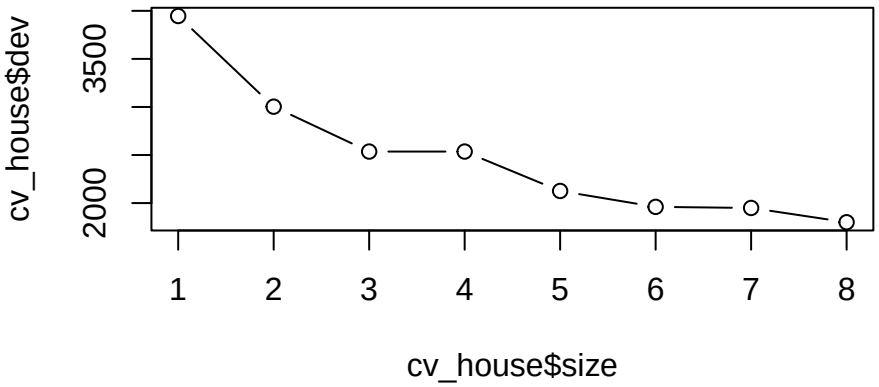
The output of `summary()` indicates that only a few of the many predictor variables have been used in constructing the tree. Next, we plot the tree.

```
plot(tree_house)
text(tree_house, pretty = 0)
```



We use the `cv.tree()` function to see whether pruning the tree will improve performance.

```
cv_house <- cv.tree(tree_house)
plot(cv_house$size, cv_house$dev, type = 'b')
```

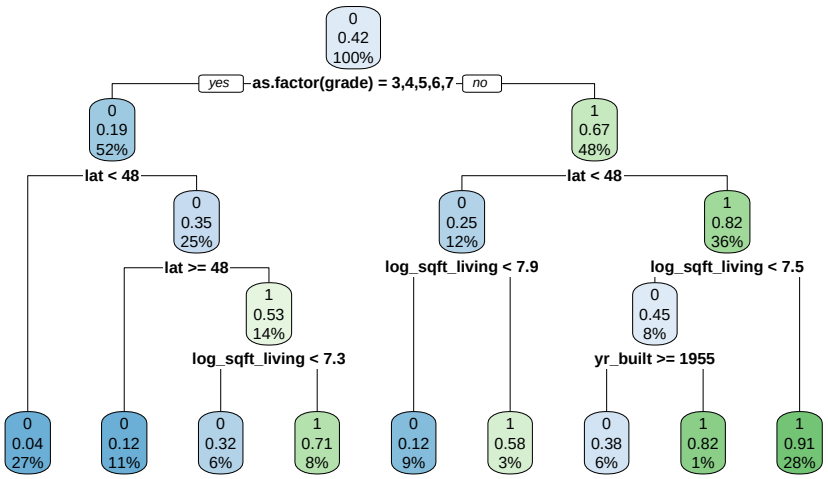


This plot shows that the full tree clearly has the lowest error, as it decreases all the way to the right end that has the largest-size model. If we instead use the rpart() function and start with the kitchen-sink model (as it proved to be most predictively effective in the GLM approach above) and apply a few control options, we have the following:

```
set.seed(495)
# kitchen-sink tree model
kchouse.rpart <- rpart(overhalfmil ~ bedrooms + bathrooms + log_sqft_living +
                         log_sqft_lot + floors + log_sqft_above + basement + renovated +
                         yr_built + zipcode + log_sqft_living15 + log_sqft_lot15 + lat +
                         long + waterfront + as.factor(view) + as.factor(condition) +
                         as.factor(grade), method ="class", data = train)
printcp(kchouse.rpart)
```

```
##
## Classification tree:
## rpart(formula = overhalfmil ~ bedrooms + bathrooms + log_sqft_living +
##     log_sqft_lot + floors + log_sqft_above + basement + renovated +
##     yr_built + zipcode + log_sqft_living15 + log_sqft_lot15 +
##     lat + long + waterfront + as.factor(view) + as.factor(condition) +
##     as.factor(grade), data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] as.factor(grade) lat              log_sqft_living  yr_built
##
## Root node error: 6802/16197 = 0.42
##
## n= 16197
##
##         CP nsplit rel error xerror     xstd
## 1 0.39194      0    1.0000 1.0000 0.009234
## 2 0.14496      1    0.6081 0.6081 0.008159
## 3 0.02602      2    0.4631 0.4640 0.007411
## 4 0.01955      5    0.3850 0.3899 0.006923
## 5 0.01794      6    0.3655 0.3768 0.006829
## 6 0.01323      7    0.3475 0.3572 0.006681
## 7 0.01000      8    0.3343 0.3433 0.006572
```

```
rpart.plot(kchouse.rpart)
```

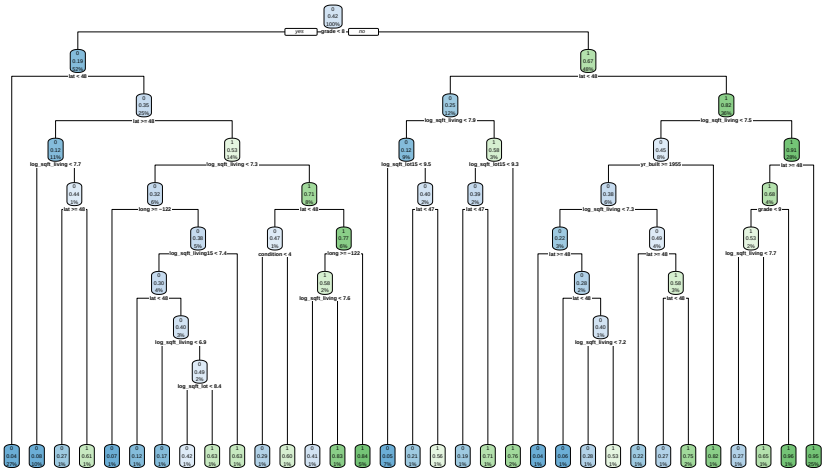



```
# tree with variables eliminated by LASSO
kchouse.rpart2 <- rpart(overhalfmil ~ bathrooms + log_sqft_living + log_sqft_lot +
                          floors + basement + renovated + yr_built + log_sqft_living15 +
                          log_sqft_lot15 + lat + long + waterfront + view + condition +
                          grade, method ="class", data = train,
                        control = rpart.control(minbucket = 100, cp = 0, minsplit = 100))
printcp(kchouse.rpart2)
```

```
##
## Classification tree:
## rpart(formula = overhalfmil ~ bathrooms + log_sqft_living + log_sqft_lot +
##     floors + basement + renovated + yr_built + log_sqft_living15 +
##     log_sqft_lot15 + lat + long + waterfront + view + condition +
##     grade, data = train, method = "class", control = rpart.control(minbucket = 100,
##     cp = 0, minsplit = 100))
##
## Variables actually used in tree construction:
## [1] condition         grade             lat               log_sqft_living
## [5] log_sqft_living15 log_sqft_lot      log_sqft_lot15    long
## [9] yr_built
##
## Root node error: 6802/16197 = 0.42
##
## n= 16197
##
##          CP nsplit rel error xerror     xstd
## 1  0.391944      0    1.0000 1.0000 0.009234
## 2  0.144957      1    0.6081 0.6081 0.008159
## 3  0.026022      2    0.4631 0.4643 0.007413
## 4  0.019553      5    0.3850 0.3897 0.006922
## 5  0.017936      6    0.3655 0.3731 0.006801
## 6  0.013231      7    0.3475 0.3553 0.006667
## 7  0.007939      8    0.3343 0.3400 0.006546
## 8  0.006175      9    0.3264 0.3346 0.006502
## 9  0.005440     10    0.3202 0.3220 0.006398
## 10 0.003822     13    0.2989 0.3155 0.006343
## 11 0.003087     15    0.2912 0.3076 0.006275
## 12 0.002695     17    0.2851 0.3054 0.006256
## 13 0.002205     20    0.2770 0.2995 0.006204
```

```
## 14 0.001691      22    0.2726 0.2970 0.006182
## 15 0.001323      24    0.2692 0.2970 0.006182
## 16 0.001274      26    0.2665 0.2959 0.006173
## 17 0.000294      29    0.2627 0.2908 0.006126
## 18 0.000000      32    0.2618 0.2895 0.006114
```

```
rpart.plot(kchouse.rpart2)
```



```
# kitchen-sink tree model on training set
train2 <- mutate(train, predhouse1 = predict(kchouse.rpart, type="class"))
tally(predhouse1 ~ overhalfmil, data = train2)
```

```
##           overhalfmil
## predhouse1    0    1
##          0 8390 1269
##          1 1005 5533
```

```
(8398+5522)/(8398+5522+1024+1253)
```

```
## [1] 0.859418
```

```
# on test set
test2 <- mutate(test, predhouse2 = predict(kchouse.rpart, type="class", newdata = test))
tally(predhouse2 ~ overhalfmil, data = test2)
```

```
##           overhalfmil
## predhouse2    0    1
##          0 2801  413
##          1  350 1835
```

```
(2779+1865)/(2779+1865+345+411)
```

```
## [1] 0.86
```

```
# reduced tree model on training set
train3 <- mutate(train, predhouse3 = predict(kchouse.rpart2, type="class"))
tally(predhouse3 ~ overhalfmil, data = train3)
```

```
##           overhalfmil
## predhouse3    0    1
##          0 8436  822
##          1  959 5980
```

```
(8455+5960)/(8455+5960+815+967)
```

```
## [1] 0.88998
```

```
# on test set
test3 <- mutate(test, predhouse4 = predict(kchouse.rpart2, type="class", newdata = test))
tally(predhouse4 ~ overhalfmil, data = test3)
```

```
##           overhalfmil
## predhouse4    0    1
##          0 2812  271
##          1  339 1977
```

```
(2774+1999)/(2774+1999+350+277)
```

```
## [1] 0.883889
```

The kitchen-sink model with no control options predicts correctly 85.9% of the time on the training set, and 86% of the time on the test set. This is alright, but next we construct a tree that leaves out a few variables eliminated by LASSO (mentioned in the GLM section). When we also add control options for the minimum numbers of observations needed in starting and resulting nodes of the tree, as well as the extremely small Cp value of 0, the outputted tree is grown much more extensively, and we achieve a higher prediction rate of over 88% on both the training and the test set. Thus, for a well-performing classification tree, we include only the variables suggested by LASSO in our tree model *kchouse.rpart2*, along with some control parameters.

Conclusion

Given a data set of house sales in King County, Washington, we were tasked with constructing a predictive model for when a house exceeds the threshold of half a million dollars based on certain property features. To do so, we built two models: a generalized linear model (GLM) and a decision tree model. The final GLM is a kitchen-sink model, meaning that it includes the full set of predictors. Because it contains so many variables, the final GLM is not necessarily simple to use or efficient, but given that our goal is to predict house sale price as accurately as possible, we choose the the full model which correctly predicts 84.4% of the time, compared to attempted GLM's with fewer variables (through summary output, LASSO, etc.) but with smaller accuracy rate as well. Using the full set of variables from this "best" GLM, we proceeded to construct a decision tree model. However, a smaller tree model proved to perform better than th. Our final classification tree model takes fewer input variables than our GLM, leaving out `bedrooms`, `zipcode`, `log_sqft_above` while predicting correctly at a rate of 88.4%. For performance, interpretability, as well as having fewer variables involved, we would recommend using the reduced classification tree model along the controls specified in the function.

The following point was previously mentioned in the EDA section, but it is important to understand: the presence of neighborhoods means that we likely do not have independence among observations. To resolve this, a future direction we may consider . We could also merge this kchouse data set with others containing information on factors like income, educational resources, and more. This might uncover a relationship between our response variable and data that are easily accessible, which would in turn help to predict home price more effectively.