Math 218: Final Report

Project Haydoja

Payoja and Hayden

Dec 6

Introduction

There is a plethora of statistical tools available to understand any given set of data. Data scientists and statisticians often use either supervised or unsupervised statistical learning methods to model their data to both better understand trends and extrapolate predictions. For the our project, however, we decided to exclusively focus on supervised learning, mainly for two reasons. First, supervised learning methods are always guided by a response variable which ensures our research objective is clearly defined. Second, supervised learning models have relevant performance metrics (i.e. RMSE for regression problems or misclass. rate for classification problems).

For this project, we decided to implement a variety of supervised learning techniques using the Ames Housing dataset, which is a publicly available dataset that includes extensive information on the sale of residential properties in Ames, Iowa from 2006 to 2010. It contains 1460 observations with 79 variables on various house specifications. The variables are a mix of continuous, categorical and discrete types.

The data is available on Kaggle.

We will investigate both regression and classification problems, with the primary goal of better understanding which statistical learning method perform best on different features from the Ames housing dataset. Below are our driving research questions:

1. Which statistical learning method most accurately predicts housing prices using other available housing features?
2. Which statistical learning method most accurately classifies houses by neighborhood based on the other available housing features?

Although our primary goal is to identify which statistical learning method performs best, we will also spend some time exploring and understanding which predictors affect our response variables and how.

Data Description

Rather than use all 79 available variables, we decided to only include features we thought were particularly relevant to our two driving research questions. This dataset also includes a lot of categorical variables, so we decided to only include the ones that can be easily translated to usable dummy/binary variables for the sake of simplicity.

Below is a description of the features we focused on for this project:

Categorical Variables:

- Neighborhood: Physical locations within Ames city limits
- CentralAir: Central air conditioning (Yes/No)
- PavedDrive: Paved driveway (Yes/No)
- GarageFinish: Interior finish of the garage
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- BsmtFinType1: Rating of basement finished area

Continuous Variables:

- SalePrice: Price of house
- PoolArea: Pool area in square feet
- LotArea: Lot size in square feet
- TotalBsmtSF: Total square feet of basement area
- GarageArea: Size of garage in square feet
- GrLivArea: Above ground living area square feet

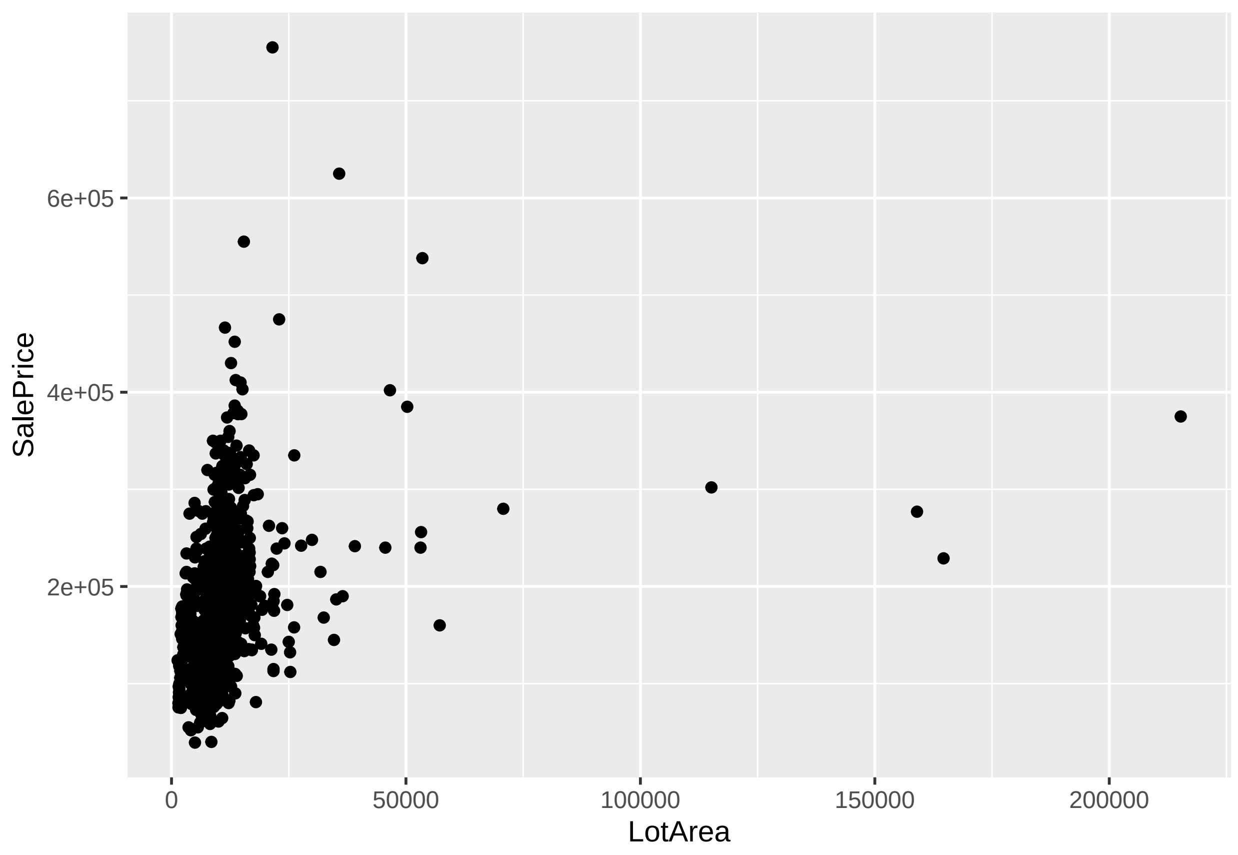
Discrete Variable

- FullBath: Number of full bathrooms above ground
- HalfBath: Number of half bathrooms above ground
- BedroomAbvGr: Number of bedrooms above ground
- KitchenAbvGr: Number of kitchens above ground
- Fireplaces: Number of fireplaces
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Year house was remodeled
- YrSold: Year house was sold

One caveat to this trimmed feature list is that there is a feature 'SaleCondition' that defines the sale of a house as either "normal" or one of several types of abnormal sales. We only want to look at the normal sales, so we've selected only those observations with SaleCondition = "normal".

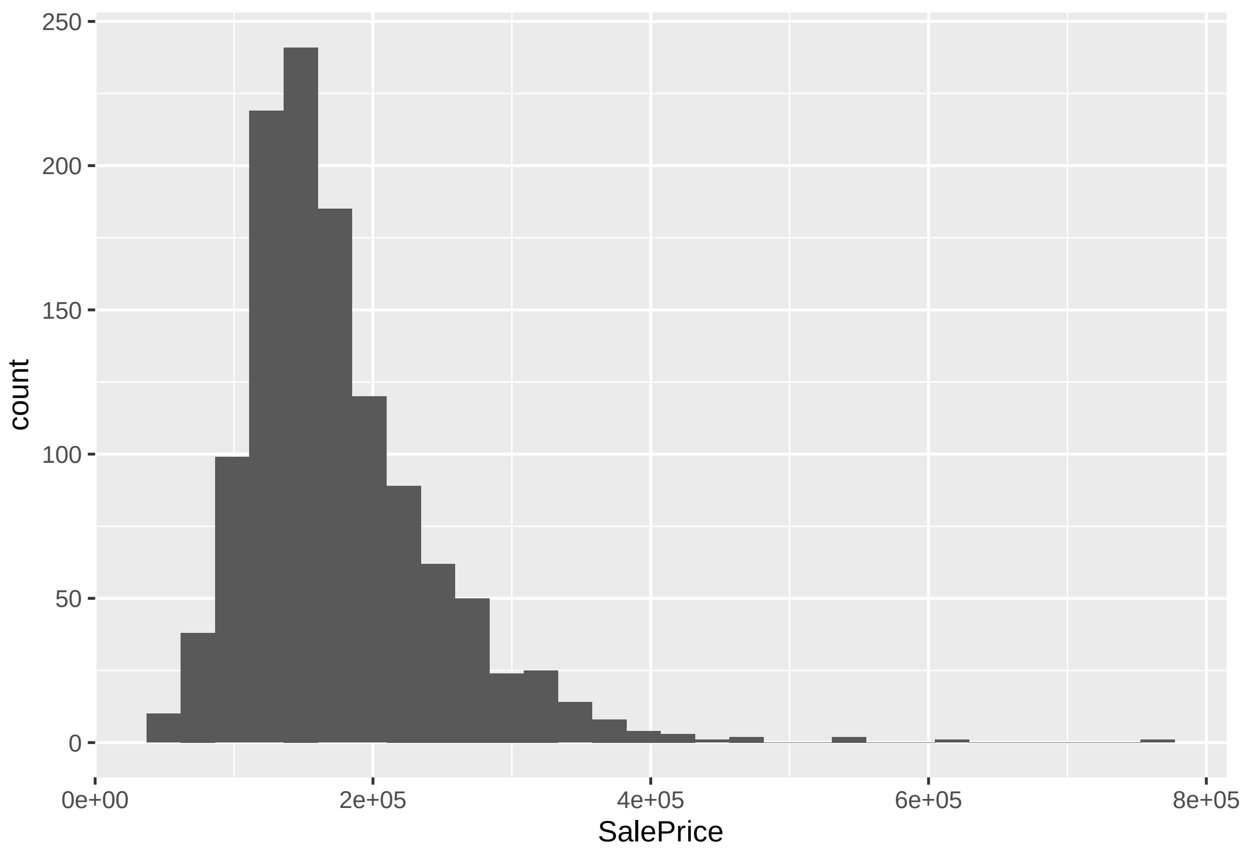
EDA

Investigating SalePrice Variable The first exploratory question we wanted to investigate is if there is, in fact, a correlation between how big a house is and how much it costs:

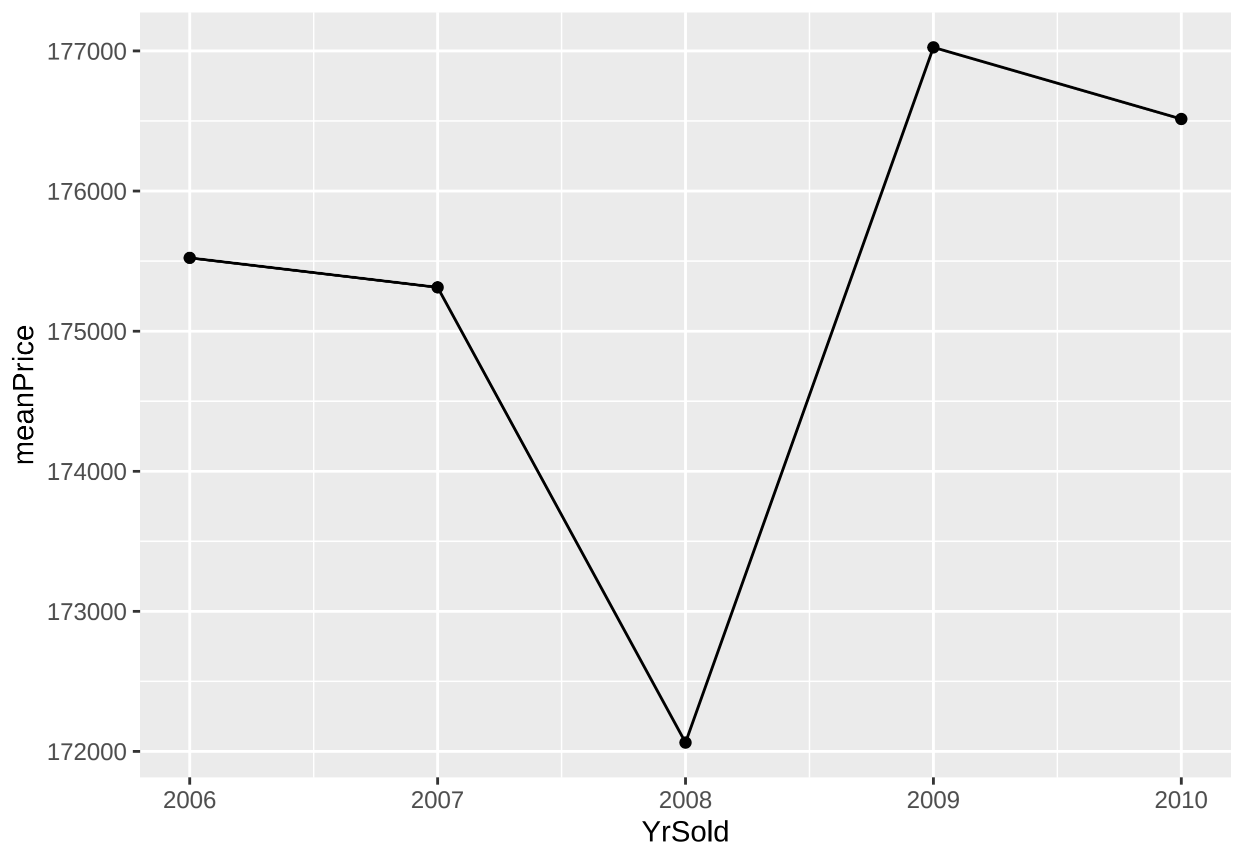


This plot suggests that, although there may be a slight positive correlation between the LotArea and SalePrice, there are likely other variables that contribute to how expensive a house is.

Let's now look a bit more at the SalePrice variable and its distribution:



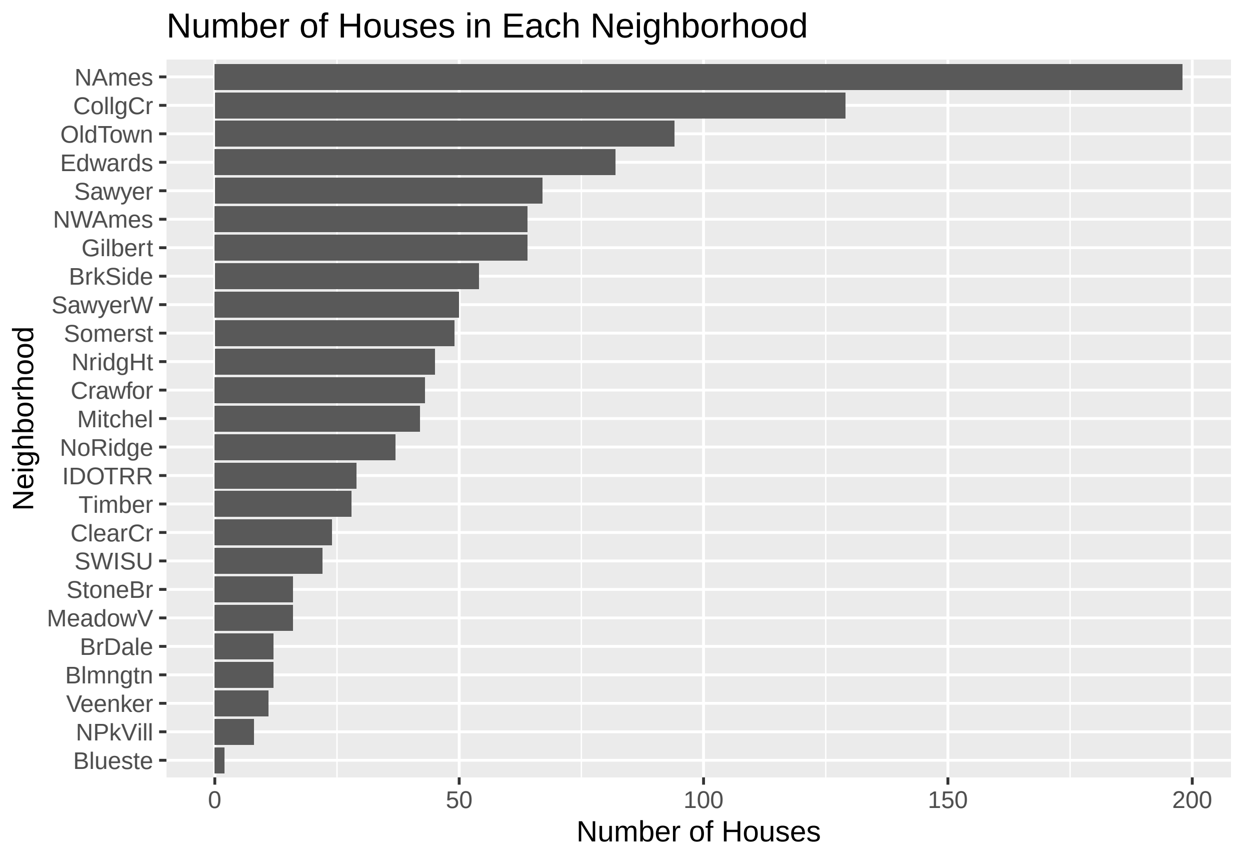
250
200
150
100
50
0
count
0e+00
2e+05
4e+05
6e+05
8e+05
SalePrice



Judging from the histogram, the distribution of SalePrice in our data set is slightly right skewed. This suggests that there are some outlier properties with significantly higher sale prices. To account for this, we will add a logarithmic transformation to the SalePrice variable (see data cleaning section).

In the second plot, we can observe a huge drop in average sale price of houses in our data coinciding with the 2008 housing market crash. To try to account for this, we decided to replace YrSold with a binary variable 'Sold_08' that will be Y if a house was sold in 2008, and N otherwise:

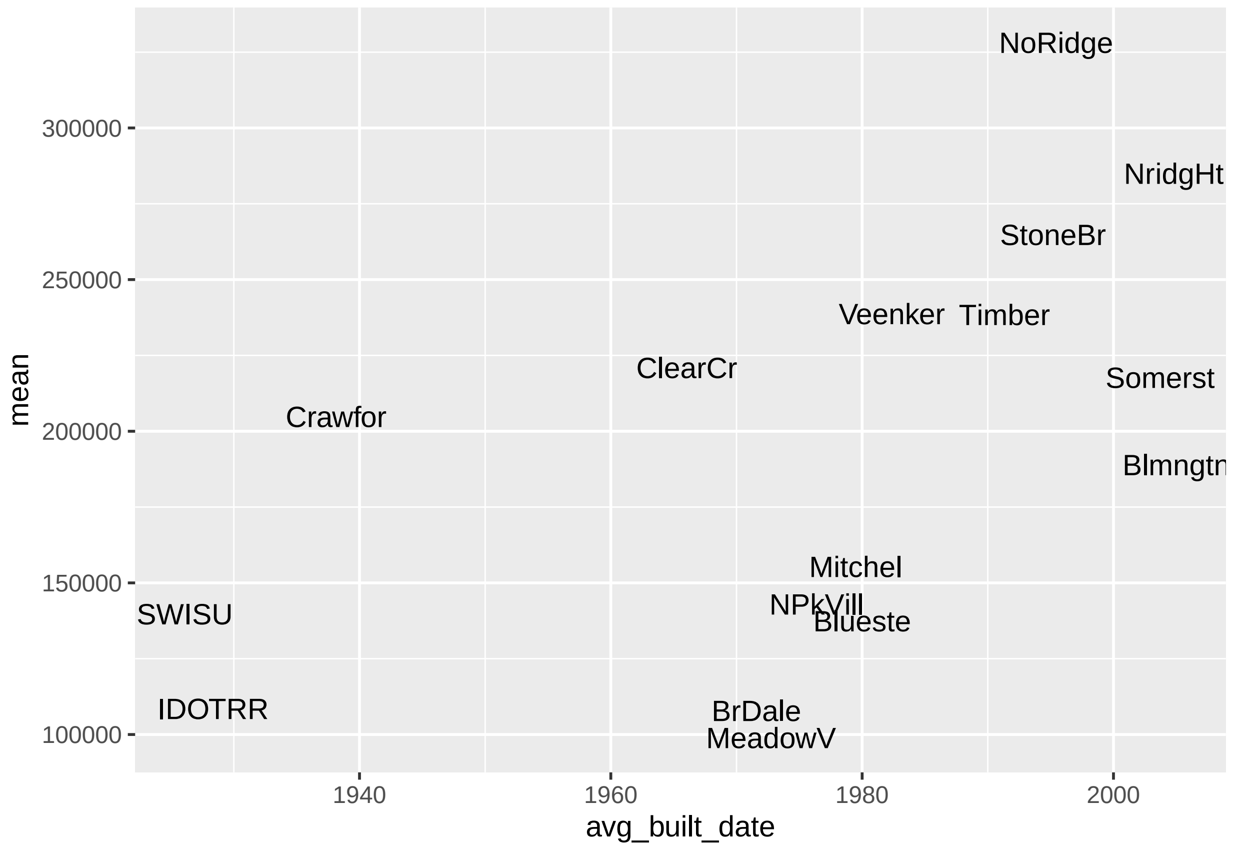
Investigating Neighborhood Variable We also wanted to look at the distribution of houses across neighborhoods, as well as how expensive the homes are in each of the neighborhoods:



This plot reveals that there is a pretty big gradient of neighborhood sizes. To simplify our models and their interpretations, we want to reduce the number of neighborhood categories by grouping some of the neighborhoods into an 'Other' category.

To do so, we first isolated the neighborhoods with fewer than 50 homes:

We then looked at how these 16 neighborhoods vary in terms of average price and average house age:



Right off the bat, we decided to merge NoRidge (Northridge) and NridgHt (Northridge Heights) because they both are comprised primarily of new, expensive homes. Also, a quick Google search revealed that the two neighborhoods border each other geographically. We defined this combined neighborhood as 'GrNoRidge' or Greater Northridge:

Other than that, there weren't really any other neighborhood merges that made sense geographically. And if we were to merge all the neighborhoods with < 50 houses, we would get an 'Other' neighborhood category that accounts for the second highest number of houses in the dataset. This would likely skew how our model predicts neighborhood classification.

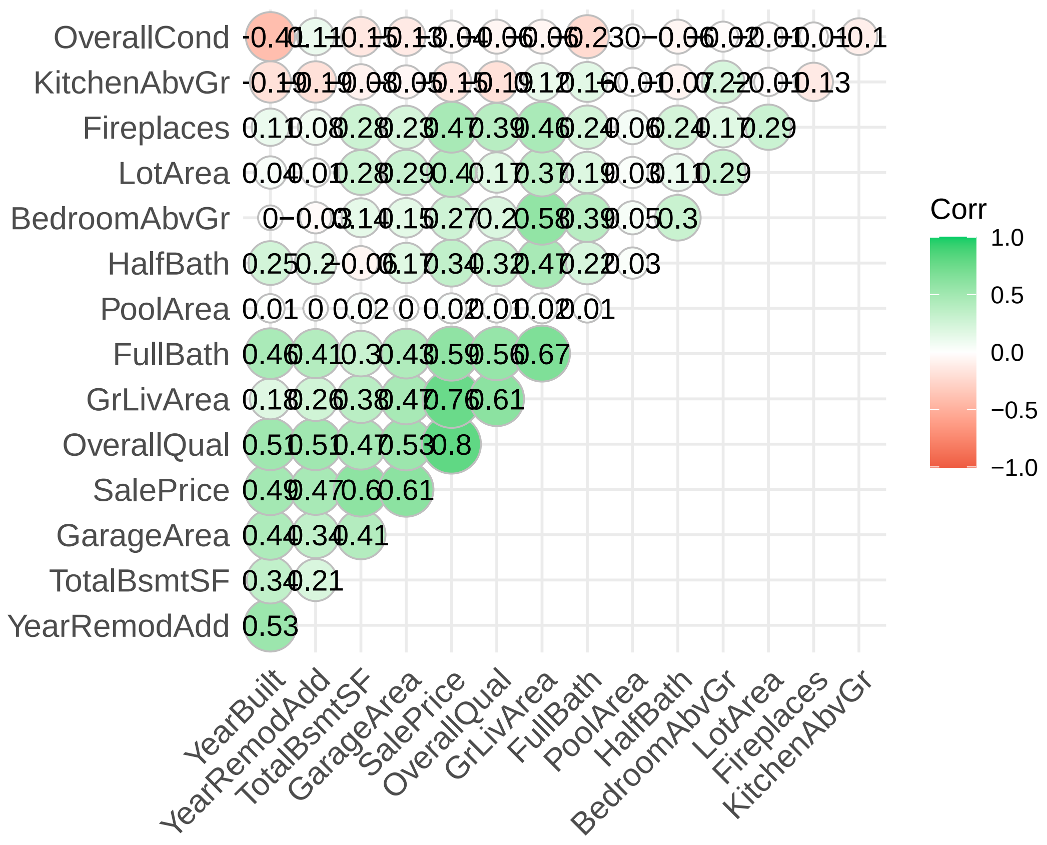
So, we decided look at how much data we would be losing if we just dropped the neighborhoods with < 50 houses or < 30 houses:

```
## [1] 314
```

```
## [1] 180
```

We decided to go ahead and drop the 180 houses in neighborhoods with < 30 houses:

Correlation between some of the numeric and discrete data fields (most importantly SalePrice):



As seen from above, housing prices seem to be the most correlated (positively) with overal quality of the house, above ground living area (in sq. ft), total basement area (in sq. ft) and garage area (in sq. ft). The price is also moderately correlated with number of full bathrooms.

Data Cleaning and Scaling

Feature Engineering There are a couple problems with the way our variables are formatted. First, if a house has not been remodeled, the YearRemodAdd will be the same as YearBuilt. Let's look at how many houses this case applies to:

```
## [1] 536
```

Since over half the houses have never been remodeled, we just decided to convert YearRemodAdd to a binary variable:

Additionally, PavedDrive is a categorical variable with three possible class: Y (if paved), P (if partially paved) and N (if dirt/gravel). For simplicity, we decided to recode the variable so that PavedDrive is a binary variable that takes Y(Yes) if it is paved and N (No) otherwise.

Scaling/Transformations Here is where we want to apply the log transformation to our SalePrice variable:

Now, we want to scale all of our numeric variables:

Handling NAs One final issue with our data is that some features still have NA values:

```
## Neighborhood   CentralAir   PavedDrive GarageFinish     BldgType   HouseStyle
##            0            0            0           47            0            0
## BsmtFinType1    SalePrice     PoolArea      LotArea  TotalBsmtSF   GarageArea
##           30            0            0            0            0            0
##    GrLivArea     FullBath     HalfBath BedroomAbvGr KitchenAbvGr   Fireplaces
##            0            0            0            0            0            0
##  OverallQual  OverallCond    YearBuilt      Sold_08        Remod
##            0            0            0            0            0
```

Both 'GarageFinish' and 'BsmtFinType1' are NA when a house doesn't have that respective feature. To handle this, we replaced those NA values with their own classification:

Methodology

Regression

To answer our first research question - "Which statistical learning method most accurately predicts housing prices using other available housing features?" - we decided to use five different regression techniques: simple linear regression, k-fold CV regression, ridge regression, lasso regression, and XGBoost.

For all models aside from K-fold CV regression, we divide our data into two sets 'train' and 'test'. The 'train' set contains 80% of our entire data where as the 'test' set contains the remaining 20%. The response variable is 'SalePrice' and the predictors are the other 22 features we included.

Below is a brief description of each method:

1. Simple Linear Regression We trained our simple linear regression model on test data to predict SalePrice using all available predictors. To assess the performance of our model, we predicted the sale price of houses in the test set, and examine the test error rate by measuring the root mean squares error.

2. K-fold CV Linear Regression To implement K-fold CV linear regression, we used the same modeling technique as simple linear regression, but this time using 10-fold cross validation. To assess model performance, we took the average of the test RMSE across all 10 folds.

3. Ridge Regression We implemented the ridge regression to examine whether shrinking features that are not as important improves the performance of the model. Using the 22 predictors, we fit the ridge model on the training set, with λ chosen by 10-fold cross-validation. To validate the performance of our model, we predicted the sale price of houses in the test set and then examined the test error rate by measuring the root mean squares error. The R package required to implement this method is 'glmnet'.

4. Lasso Regression 22 predictors, although better than having too few predictors, may be too many to use for this research question. Thus, we want to implement the lasso method so that coefficients of features that are not as important shrinks to 0. Using the 22 predictors, we fit the lasso model on the training set, with λ chosen by cross-validation with k-fold equals 10. To validate the performance of our model, we predicted the sale price of houses in the test set using the lasso model fit on the training set. Then, examined the test error rate by measuring the root mean squares error. The R package required to implement this method is 'glmnet'.

5. XGBoost The XGBoost algorithm stands for eXtreme Gradient Boosting and is one of the most popular gradient boosted trees algorithms. At a high level, the XGBoost algorithm works by building decision trees on the training data, and then combining the predictions made by these trees to make a final prediction. The decision trees are built in a way that tries to minimize the loss function, which is a measure of how far the predicted values are from the true values in the training data.

One key feature of the XGBoost algorithm is that it uses regularization to prevent overfitting. This is achieved by adding a regularization term to the loss function, which penalizes models that are too complex.

For our project, we trained the XGBoost algorithm on all available predictors - we had to convert categorical variables to dummy variables because XGBoost can't handle categorical data. To assess model performance, we predicted sale price of houses in the test set using the model and calculated RMSE. The R package required to implement this method is 'xgboost'.

Classification

To answer our second research question relating to the classification problem, "Which statistical learning method most accurately classifies houses by neighborhood based on the other available housing features?", we decided to use two different classification models: Naive Bayes and Tree-Based Classification. For both methods we divide our data into two sets 'train' and 'test'. The 'train' set contains 80% of our entire data where as the 'test' set contains the remaining 20%. The train and the test set are the same for all both models.

Below is a brief description of each method:

1. Naive Bayes To implement Naive Bayes, we used the naiveBayes() function, which is part of the e1071 library, on our test set. To evaluate the performance of our model, we predicted the response on the test data and produced a contingency table comparing the true test labels to the predictions.

2. Tree-Based Classification To implement this method, we fit a decision classification tree to the training data, using 'Neighborhood' as the response variable. We used all available predictors to fit the model. To evaluate the performance of our model, we predictED the response on the test data and produceD a contingency table comparing the true test labels to the predictions.

Results

Which statistical learning method most accurately predicts housing prices using other available housing features?

Simple Linear Regression

```
## [1] 17498.74
```

```
##         (Intercept)   NeighborhoodCollgCr   NeighborhoodCrawfor
##         11.984336666          -0.020320265           0.096289317
##  NeighborhoodEdwards   NeighborhoodGilbert NeighborhoodGrNoRidge
##         -0.070188801          -0.036984504           0.036917039
##  NeighborhoodMitchel     NeighborhoodNAmes    NeighborhoodNWAmes
##         -0.060737184          -0.056755422          -0.090504809
##  NeighborhoodOldTown    NeighborhoodSawyer   NeighborhoodSawyerW
##         -0.058499751          -0.055116299          -0.044823431
##  NeighborhoodSomerst           CentralAirY           PavedDriveY
##          0.036500114           0.063531406           0.026587955
##    GarageFinishNoGar       GarageFinishRFn       GarageFinishUnf
##         -0.055978535          -0.011545133          -0.018924517
##       BldgType2fmCon        BldgTypeDuplex          BldgTypeTwnhs
##         -0.013603867          -0.065421718          -0.085519419
##       BldgTypeTwnhsE      HouseStyle1.5Unf       HouseStyle1Story
##         -0.037631235          -0.028764804          -0.002239676
##     HouseStyle2.5Fin      HouseStyle2.5Unf       HouseStyle2Story
##          0.020545084          -0.007380128           0.019187240
##      HouseStyleSFoyer        HouseStyleSLvl       BsmtFinType1BLQ
##          0.027183926           0.033387764          -0.016476017
##       BsmtFinType1GLQ       BsmtFinType1LwQ    BsmtFinType1NoBsmt
##          0.011539716          -0.049196909           0.028554978
##       BsmtFinType1Rec       BsmtFinType1Unf              PoolArea
##         -0.019468609          -0.064072277           0.001555512
##               LotArea           TotalBsmtSF            GarageArea
##          0.027004642           0.068463846           0.030318497
##             GrLivArea              FullBath              HalfBath
##          0.113924490           0.013863218           0.015770673
##          BedroomAbvGr          KitchenAbvGr            Fireplaces
##         -0.004049960          -0.005291752           0.020053060
##           OverallQual           OverallCond             YearBuilt
##          0.077919001           0.056394987           0.074122946
##             Sold_08Y                RemodY
##          0.003978057           0.013134782
```

Based on the output of the regression model, the 5 variables that seems to affect the sale price of the houses are GrLivArea, NeighborhoodCrawfor (a dummy that indicates whether house is in 'Crawfor' neighborhood), OverallQual, YearBuilt and TotalBsmtSF.Since the numeric variables are scaled, it is a little tricky to estimate by how much these variables affect the Sale price of the house. However, the sign on the coefficients of these variables are informative in that it indicates in which direction the price of the house will change as these variable changes. For example, the coefficient of 0.114 on GrLivArea implies that, controlling for other factors, an increase in above ground living area(in sq ft) increases the sale price of the house. The coefficient of 0.0963 on NeighborhoodCrawfor implies that, controlling for other factors, the sale price of houses in Crawfor Neighborhood are higher than sale price of houses in BrkSide Neighborhood (which is our baseline neighborhood).

The Root Mean Squared Error (RMSE), which has been re-scaled to USD, is 17498.74.

This means that the prediction of houses in the test set is, on average, off by $17,498.74. This error seems small given that the sale prices of the houses are much higher.

CV Regression

```
## [1] 17950.85
```

The Root Mean Squared Error (RMSE), which has been re-scaled to USD, is 17950.85.

This means that the prediction of houses in the test set is,on average, off by $17,950.85. Again, this error seems relatively small.

Ridge Regression

```
## [1] 16196.87
```

```
## 50 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)           11.951
## NeighborhoodCollgCr    0.020
## NeighborhoodCrawfor    0.118
## NeighborhoodEdwards   -0.043
## NeighborhoodGilbert   -0.001
## NeighborhoodGrNoRidge  0.086
## NeighborhoodMitchel   -0.024
## NeighborhoodNAmes     -0.025
## NeighborhoodNWAmes    -0.047
## NeighborhoodOldTown   -0.055
## NeighborhoodSawyer    -0.033
## NeighborhoodSawyerW   -0.005
## NeighborhoodSomerst    0.071
## CentralAirY            0.064
## PavedDriveY            0.039
## GarageFinishNoGar     -0.059
## GarageFinishRFn       -0.012
## GarageFinishUnf       -0.032
## BldgType2fmCon        -0.008
## BldgTypeDuplex        -0.047
## BldgTypeTwnhs         -0.070
## BldgTypeTwnhsE        -0.026
## HouseStyle1.5Unf      -0.033
## HouseStyle1Story      -0.011
## HouseStyle2.5Fin       0.039
## HouseStyle2.5Unf       0.045
## HouseStyle2Story       0.013
## HouseStyleSFoyer       0.024
## HouseStyleSLvl         0.021
## BsmtFinType1BLQ       -0.015
## BsmtFinType1GLQ        0.027
## BsmtFinType1LwQ       -0.047
## BsmtFinType1NoBsmt     0.019
## BsmtFinType1Rec       -0.026
## BsmtFinType1Unf       -0.056
## PoolArea               0.001
## LotArea                0.027
## TotalBsmtSF            0.065
## GarageArea             0.037
## GrLivArea              0.090
## FullBath               0.023
## HalfBath               0.019
## BedroomAbvGr           0.002
## KitchenAbvGr          -0.011
## Fireplaces             0.023
## OverallQual            0.074
## OverallCond            0.048
## YearBuilt              0.049
## Sold_08Y               0.003
## RemodY                 0.002
```

Based on the output of the regression model, the 5 variables that seems to affect the sale price of the houses are Neighborhood_Crawfor (a dummy that indicates whether house is in 'Crawfor' neighborhood), GrLivArea, NeighborhoodGrNoRidge (a dummy that indicates whether house is in 'GrNoRidge' neighborhood), OverallQual and NeighborhoodSomerst (a dummy that indicates whether house is in 'Somerst' neighborhood). The interpretation is similar to that of linear regression.

The RMSE error, which has been re-scaled to USD, is 16196.87.

This implies that the prediction of houses in the test set is, on average, off by $16,196.87. This is the smallest test RMSE we've seen so far.

Lasso Regression

```
## 50 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)           11.956572516
## NeighborhoodCollgCr    .
## NeighborhoodCrawfor    0.095668029
## NeighborhoodEdwards    .
## NeighborhoodGilbert    .
## NeighborhoodGrNoRidge  0.023284669
## NeighborhoodMitchel    .
## NeighborhoodNAmes      .
## NeighborhoodNWAmes     .
## NeighborhoodOldTown   -0.017074752
## NeighborhoodSawyer     .
## NeighborhoodSawyerW    .
## NeighborhoodSomerst    .
## CentralAirY            0.052943886
## PavedDriveY            0.001347142
## GarageFinishNoGar      .
## GarageFinishRFn        .
## GarageFinishUnf       -0.002022007
## BldgType2fmCon         .
## BldgTypeDuplex        -0.015950561
## BldgTypeTwnhs          .
## BldgTypeTwnhsE         .
## HouseStyle1.5Unf       .
## HouseStyle1Story       .
## HouseStyle2.5Fin       .
## HouseStyle2.5Unf       .
## HouseStyle2Story       .
## HouseStyleSFoyer       .
## HouseStyleSLvl         .
## BsmtFinType1BLQ        .
## BsmtFinType1GLQ        0.027863807
## BsmtFinType1LwQ        .
## BsmtFinType1NoBsmt     .
## BsmtFinType1Rec        .
## BsmtFinType1Unf       -0.029769478
## PoolArea               .
## LotArea                0.020323495
## TotalBsmtSF            0.050394907
## GarageArea             0.039465417
## GrLivArea              0.122618938
## FullBath               .
## HalfBath               0.004210934
## BedroomAbvGr           .
## KitchenAbvGr          -0.006587478
## Fireplaces             0.018723069
## OverallQual            0.096694832
## OverallCond            0.040888713
## YearBuilt              0.081612655
## Sold_08Y               .
## RemodY                 .

## [1] 17882.54
```

Based on the output, the lasso model scaled the coefficients on 28 of the 49 variables to 0. The interpretation of the coefficients of some of the categories in Neighborhoods, Building Type, HouseStyle, Basement Finish Type and Garage finish getting scaled to 0 is a bit trickier since these variables were categorical variables with multiple classes but only coefficients of certain classes were scaled to 0. It is hard to tell whether this implies these variables are not important covariates or only certain classes of these categorical variables are important when predicting for saleprice.

The 5 variables that seems to affect the sale price of the houses are GrLivArea, YearBuilt, NeighborhoodCrawfor (a dummy that indicates whether house is in 'Crawfor' neighborhood), OverallQual and CentralAirY(a dummy that indicates whether house is in 'Crawfor' neighborhood). The interpretation for GrLivArea, NeighborhoodCrawfor and OverallQual is similar to that of Linear Regression. The coefficient of 0.08 on YearBuilt implies that, controlling for other variables, newwly built houses are more expensive. Likewise, the coefficient on CentralAirY implies that, controlling for other variables, houses that have central air conditioning have sale prices higher than houses that don't have central air conditioning.

The RMSE error, which has been re-scaled to USD, is 17882.54. This implies that the prediction of houses in the test set is, on average, off by $17,882.54.

XGBoost We chose to include the code chunk where we prepare the XGBoost model data just because this is a method we haven't looked at in this class.

```r
#Create dataset that XGBoost can handle
xg_dat <- dummy_cols(dat, remove_first_dummy = TRUE)

#Drop old categorical columns
cat_name <- names(xg_dat)[-which(sapply(xg_dat, is.numeric))]
xg_dat <- xg_dat[,!(names(xg_dat) %in% cat_name)]

#Replace numeric NAs with 0s.
xg_dat[is.na(xg_dat)] = 0

#Create test and train sets
train_x <- data.matrix(select(xg_dat[train_ids,], -SalePrice))
train_y<-xg_dat[train_ids,]$SalePrice

test_x<-data.matrix(select(xg_dat[test_ids,], -SalePrice))
test_y<-xg_dat[test_ids,]$SalePrice

xgb_train<-xgb.DMatrix(data = train_x, label = train_y)
xgb_test<-xgb.DMatrix(data = test_x, label = test_y)
```

Note: an interesting feature of XGBoost is tuning different hyperparameters. We did not have time to fully investigate/learn how these parameters work, so we ignored most of them.

If we look at the model at each iteration, the train RMSE decreases slightly. It is hard to understand what exactly the train RMSE values mean just because they are calculated based on the log-transformed SalePrice value.

However, we can convert the test RMSE values back to USD values using the exp() function:

```
## [1] 18810.44
```

This means that the prediction of houses in the test set is, on average, off by $18,810.44.

Which statistical learning method most accurately classifies houses by neighborhood based on the other available housing features?

Naive Bayes

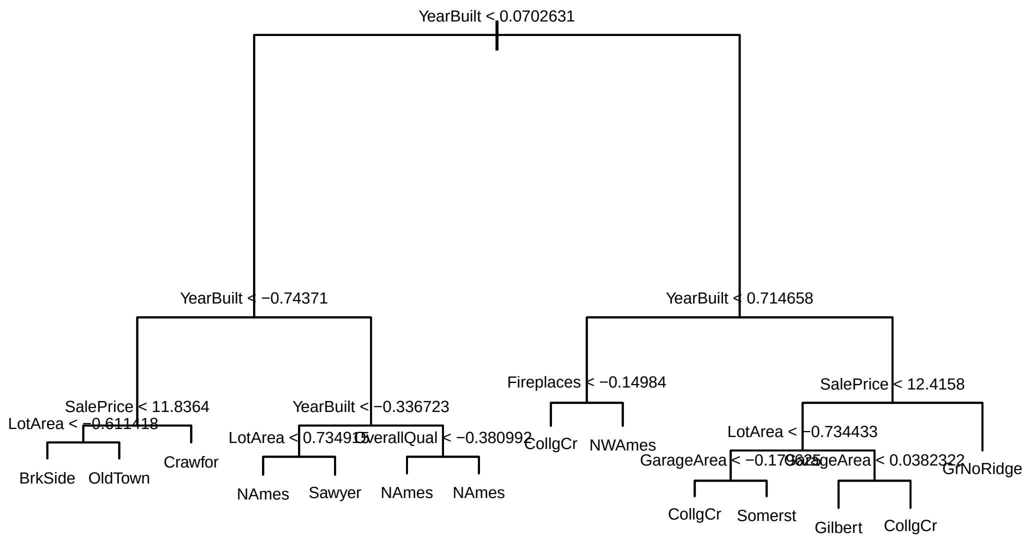
```
## [1] 0.5588235
```

```
##            y_pred
##             BrkSide CollgCr Crawfor Edwards Gilbert GrNoRidge Mitchel NAmes
##   BrkSide         3       0       0       0       0         0       0     0
##   CollgCr         0       3       0       0       0         3       0     0
##   Crawfor         1       1       4       0       0         0       0     0
##   Edwards         2       2       0       0       1         0       0     2
##   Gilbert         0       1       0       0       8         0       0     0
##   GrNoRidge       0       0       0       0       1         6       0     0
##   Mitchel         0       2       0       0       0         0       0     0
##   NAmes           0       3       1       1       1         0       0    14
##   NWAmes          0       1       0       0       2         0       0     0
##   OldTown         4       0       1       0       0         0       0     0
##   Sawyer          0       1       0       1       0         0       0     3
##   SawyerW         0       2       0       0       1         0       0     1
##   Somerst         0       0       0       0       0         0       0     0
##            y_pred
##             NWAmes OldTown Sawyer SawyerW Somerst
##   BrkSide        0       1      0       0       0
##   CollgCr        0       0      1       0       5
##   Crawfor        0       0      0       0       0
##   Edwards        0       1      2       0       2
##   Gilbert        0       0      0       0       0
##   GrNoRidge      0       0      0       0       0
##   Mitchel        0       0      1       0       0
##   NAmes          0       0      3       1       0
##   NWAmes         0       0      0       0       0
##   OldTown        0       3      0       0       0
##   Sawyer         0       1      2       0       0
##   SawyerW        0       0      0       0       0
##   Somerst        0       0      0       0       2
```

The misclassification rate is ~0.56, which is really high.

Tree

```
##
## Classification tree:
## tree(formula = as.factor(Neighborhood) ~ ., data = tree_dat[train_ids,
##     ])
## Variables actually used in tree construction:
## [1] "YearBuilt"   "SalePrice"   "LotArea"     "OverallQual" "Fireplaces"
## [6] "GarageArea"
## Number of terminal nodes:  14
## Residual mean deviance:  2.491 = 2247 / 902
## Misclassification error rate: 0.4596 = 421 / 916
```



Based on the tree, it seems like YearBuilt is the most important factor for determining the class of the species. Since the data is scaled, it is a little tricky to interpret the tree but we we wanted to we could un-scale the data to know the exact value the split was the based on. For example, the root of the tree, YearBuilt < 0.07, can be un-scaled so that the root note is:

```
## [1] 1971.492
```

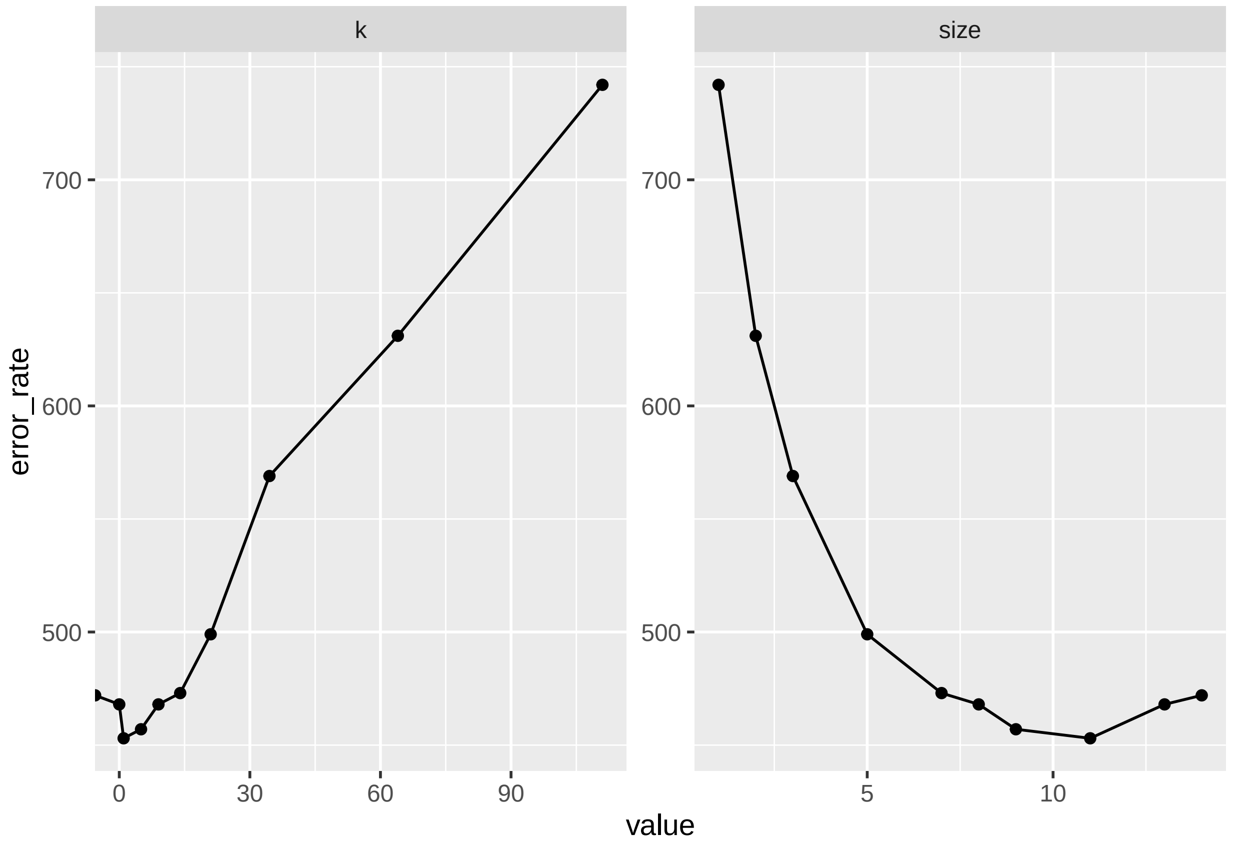
The contingency table for the test set is:

```
##              true
## preds         BrkSide CollgCr Crawfor Edwards Gilbert GrNoRidge Mitchel NAmes
##   BrkSide           2       0       0       1       0         0       0     0
##   CollgCr           0      10       0       4       1         0       2     1
##   Crawfor           1       0       4       0       0         0       0     0
##   Edwards           0       0       0       0       0         0       0     0
##   Gilbert           0       0       0       0       8         0       0     0
##   GrNoRidge         0       1       0       0       0         6       0     0
##   Mitchel           0       0       0       0       0         0       0     0
##   NAmes             0       0       1       4       0         0       0    22
##   NWAmes            0       1       0       1       0         1       1     1
##   OldTown           1       0       1       1       0         0       0     0
##   Sawyer          0       0       0       1       0         0       0     0
##   SawyerW         0       0       0       0       0         0       0     0
##   Somerst         0       0       0       0       0         0       0     0
##             true
## preds       NWAmes OldTown Sawyer SawyerW Somerst
##   BrkSide        0       4      0       0       0
##   CollgCr        0       0      1       2       2
##   Crawfor        0       2      0       0       0
##   Edwards        0       0      0       0       0
##   Gilbert        0       0      0       1       0
##   GrNoRidge      0       0      0       0       0
##   Mitchel        0       0      0       0       0
##   NAmes          0       1      6       1       0
##   NWAmes         3       0      1       0       0
##   OldTown        0       1      0       0       0
##   Sawyer         0       0      0       0       0
##   SawyerW        0       0      0       0       0
##   Somerst        0       0      0       0       0
```

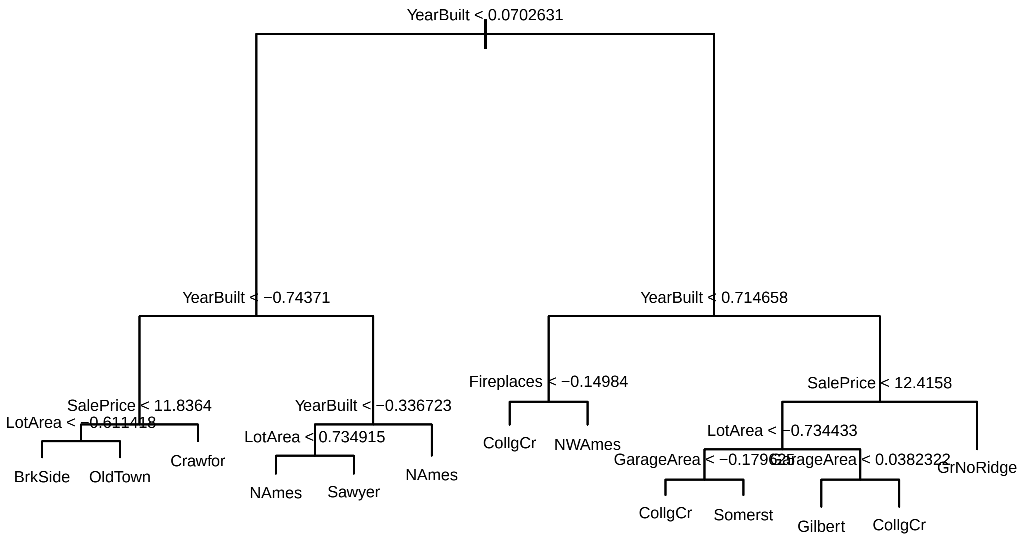
The misclassification rate is:

```
## [1] 0.4509804
```

The misclassification rate of ~ 0.45 is still quite high. So we decided to check whether pruning the tree will improve the results.



Based on the output, the tree with 13 terminal nodes seems to be the best. Thus, we will prune the original tree to obtain the 13 node tree.



The misclassification rate on the test set is then:

```
## [1] 0.4509804
```

The misclassification rate of the pruned tree did not change significantly. This makes sense because pruning only reduced the number of terminal nodes by 1–the original tree had 14 terminal nodes whereas the pruned one has 13.

Discussion

To summarize the results of our regression models, we found that the Root Mean Squared Error (RMSE) was 17498.74 for simple linear regression model, 17950.85 for linear regression with cross-validation, 16196.87 for ridge regression model and 17882.54 for the lasso model and 18810.44 for the XGBoost method. Thus, our best performing model was the ridge regression. This indicates that most of the predictors that were used in the regression impact the sale price significantly. Overall, we think that the our best performing model is a good choice to predict the sale prices as RMSE of 16196.87 is low if we compare it to the sale price of houses, which are much higher.

To summarize the results of our classification models, we found the misclassification rates of the Naive Bayes and Decision Tree models to be approximately .67 and .45, respectively. Although the tree-based method performs significantly better, we still think that the misclassification rate is really high. This suggests that perhaps the neighborhoods are not clustered based solely on the predictors we used.

Limitations

There are several big limitations to this project that we struggled to address. First, this dataset has so many variables to use that it was challenging to select the most important ones. We tried using Ridge and Lasso to handle this issue, but it got pretty messy with all the categorical variables.

Second, we arbitrarily set a seed and used the same train/test split for all our regression models (aside from the k-fold CV linear regression). Because we were only working with just over 1000 observations, the way we split our data could have a huge impact on the resulting RMSE. For example, we tried setting several different seeds and re-running the models and each time we got significantly different error values. The only model which was relatively consistent was the k-fold CV regression, which makes intuitive sense. If we had more time, we would've liked to investigate and implement other re-sampling techniques, specifically the CV XGBoost model.

The third main limitation with this project was handling the Neighborhood variable for classification. In our EDA section, we showed that there are a lot of neighborhoods that have very few houses. This makes it pretty difficult to create a model that will accurately predict those specific neighborhoods for a given test observation. We tried to account for this issue by dropping the neighborhoods with < 30 houses, but that is an imperfect solution and further limits the amount of data we had to train our models on.

The fourth main limitation with this project was interpreting coefficients with categorical variables that had been converted to dummy variables. Our dataset had categorical features that have many different values which in turn created a lot of dummy variables in our models. This made it challenging to identify which of the classifications for a given variable were actually statistically significant.