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 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

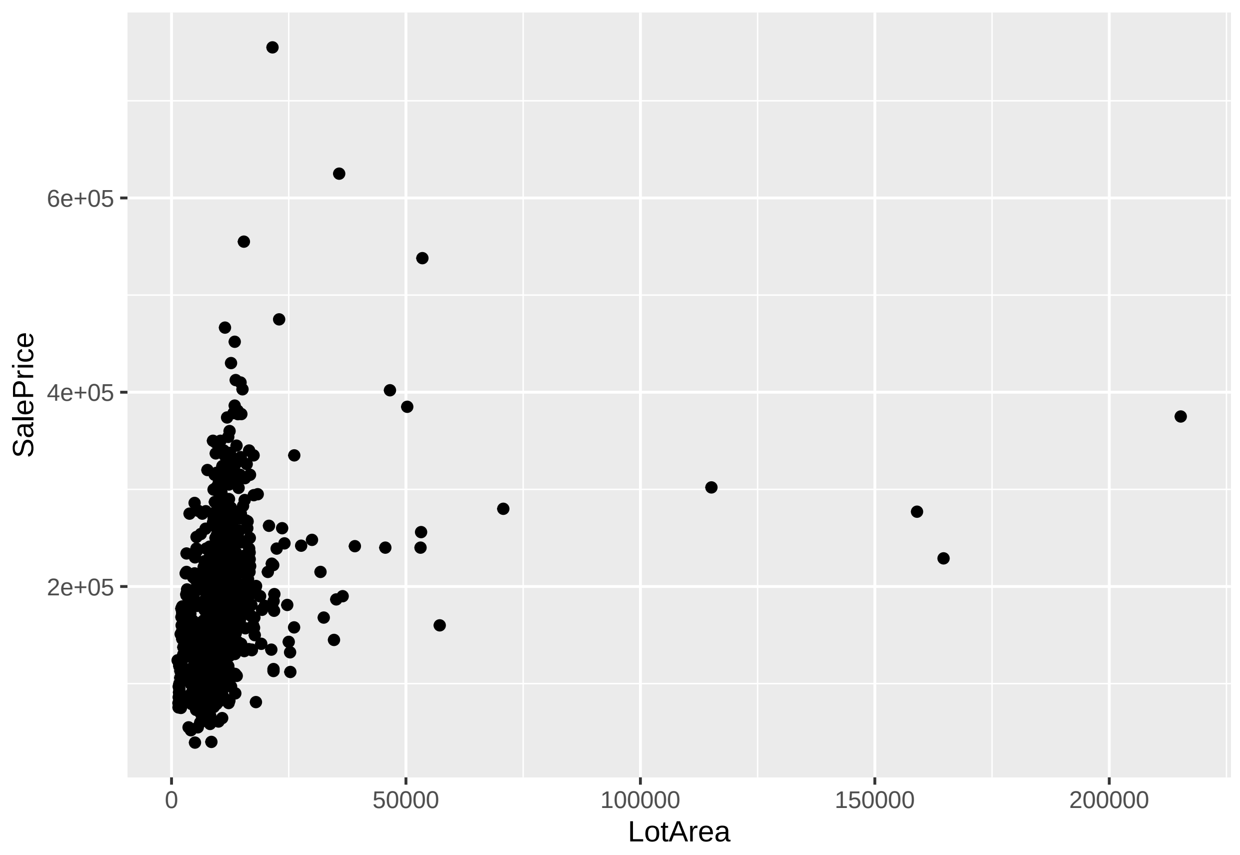
Discrete Variable

- 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".

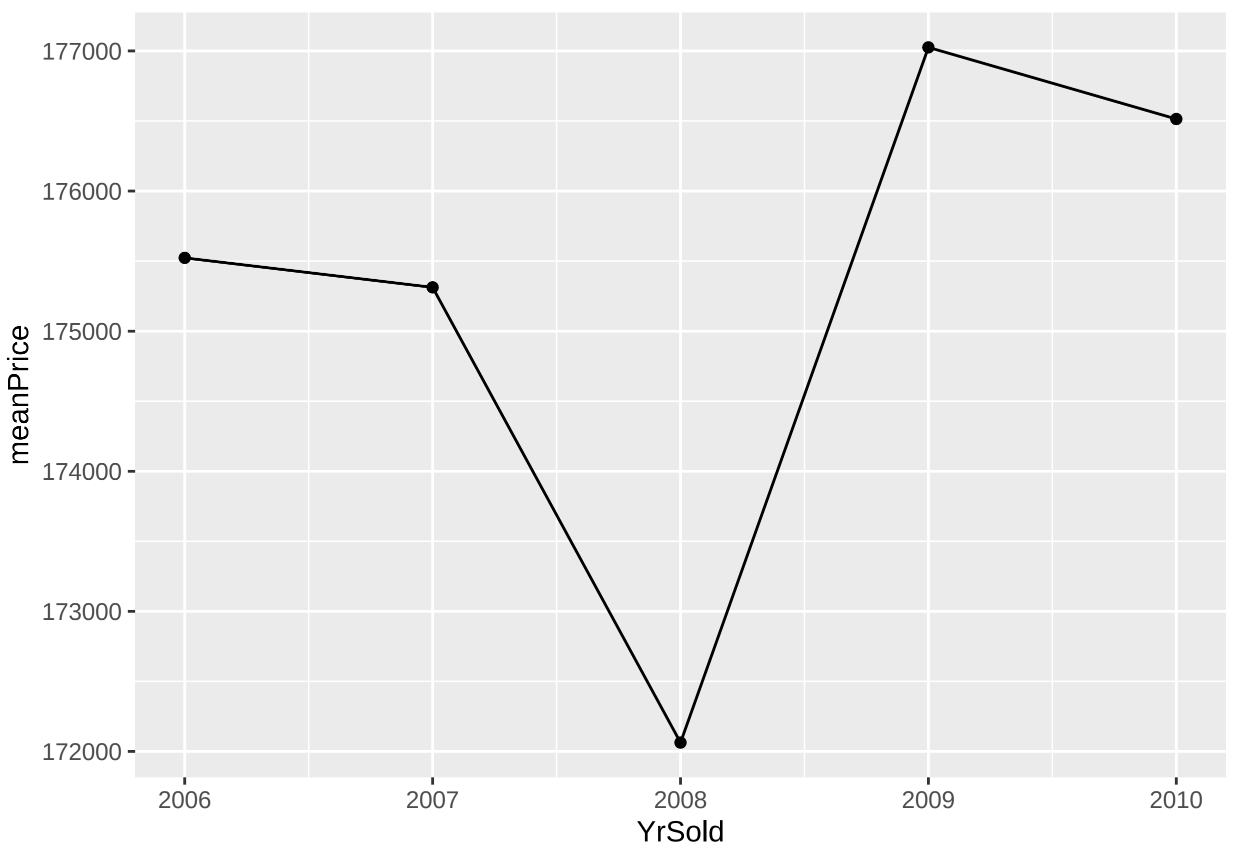
EDA

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:

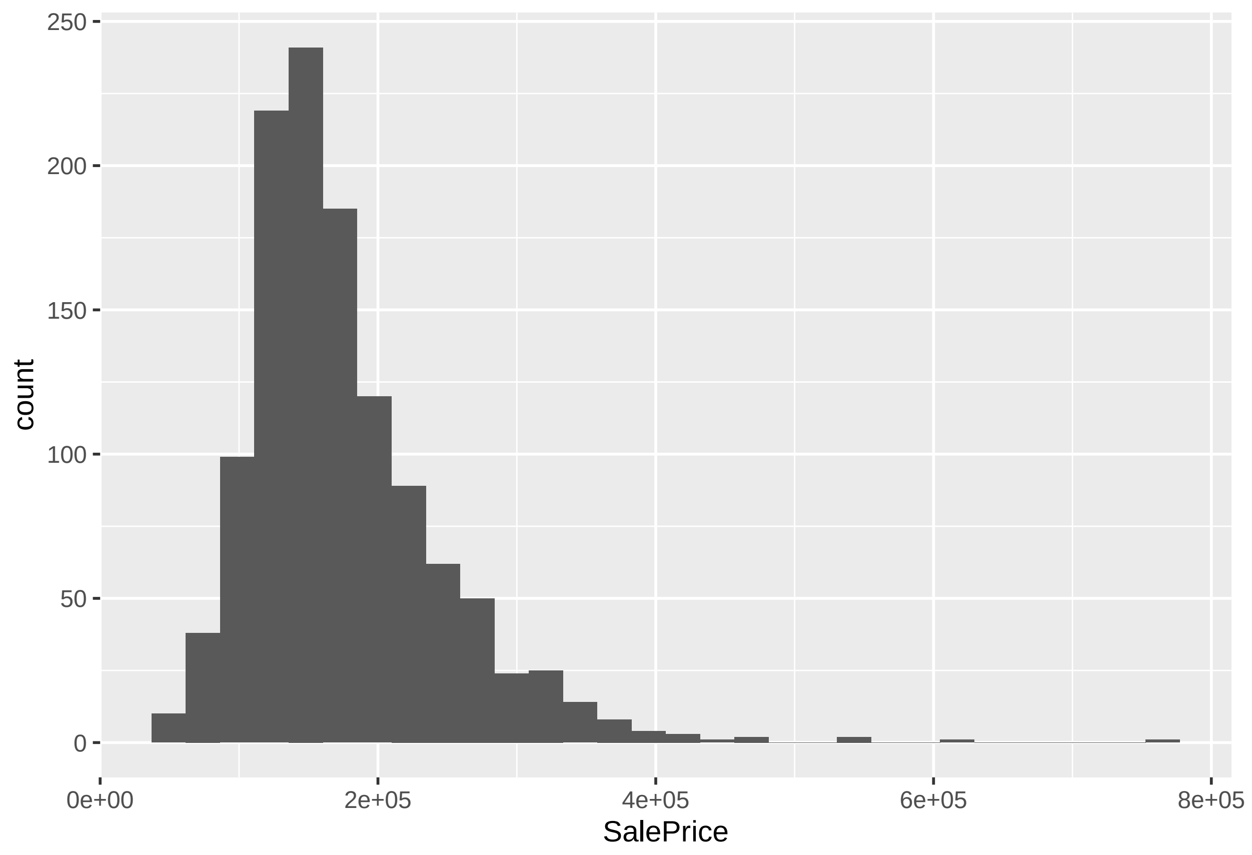


This plot suggests that, although there may be a slight positive correlation between the LotArea and SalePrice, there are likely other variables that contribute to how expensive a house is.

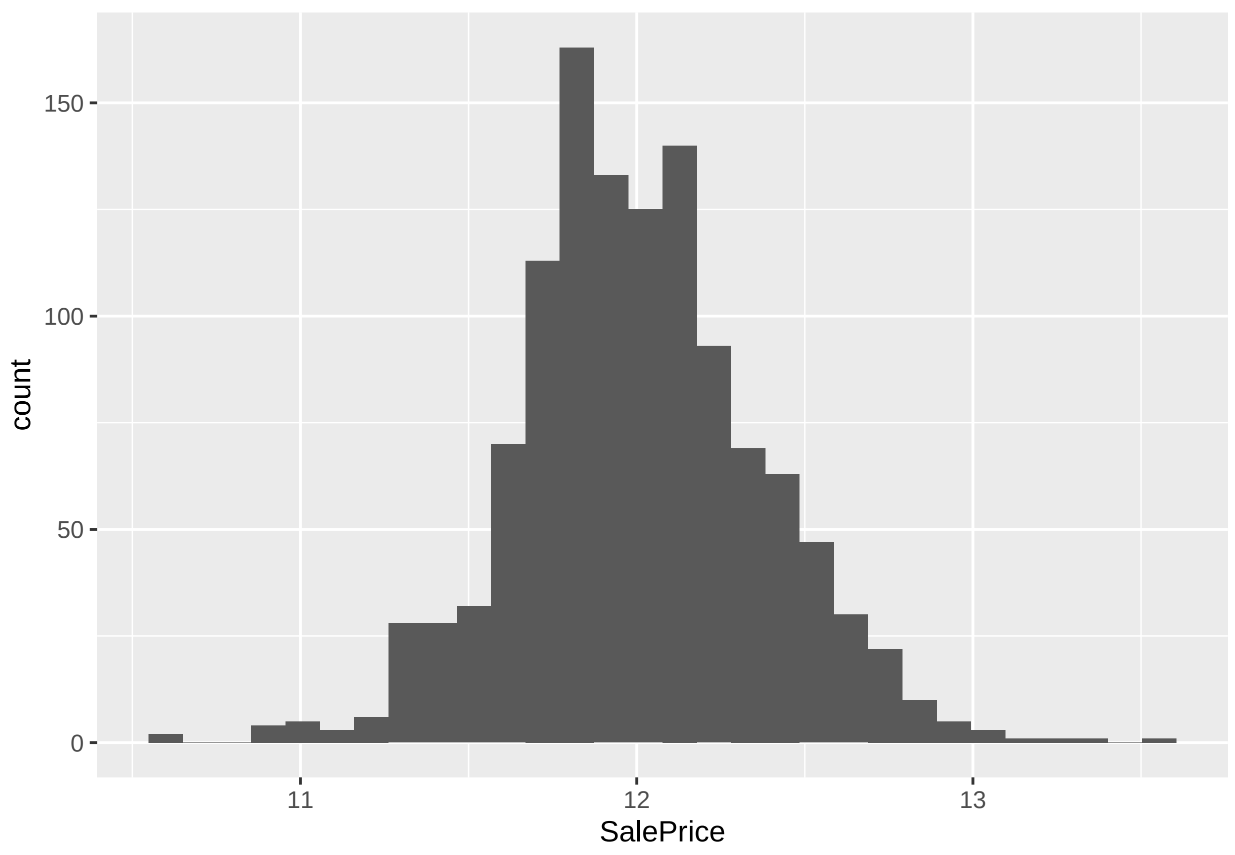
Let's now look a bit more at the SalePrice variable and how it changed over the years.



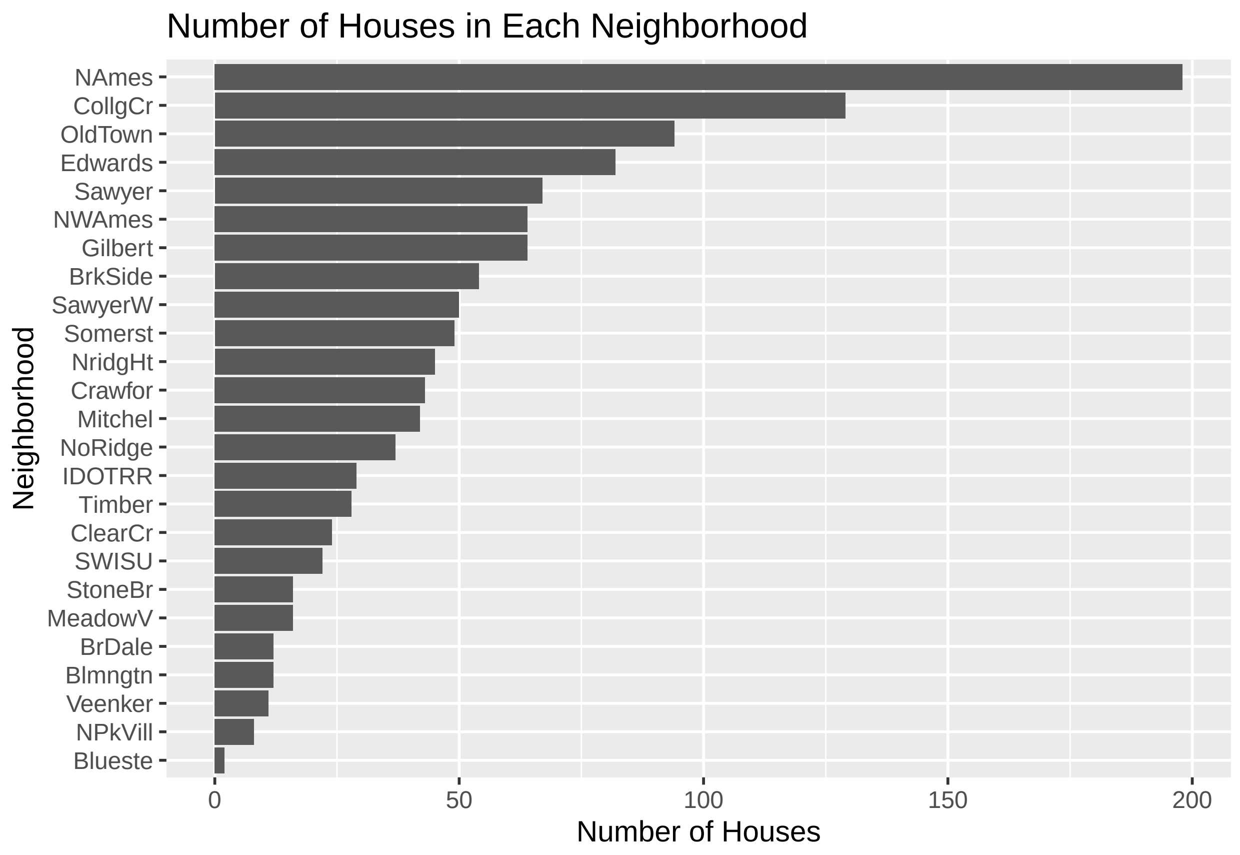
From the plot, we can observe a huge drop in average sale price of houses in our data coinciding with the 2008 housing market crash. To try to account for this, we decided to replace YrSold with a binary variable ‘Sold_08’ that will be Y if a house was sold in 2008, and N otherwise.



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 to ensure that sale price is normally distributed as shown in the plot below.

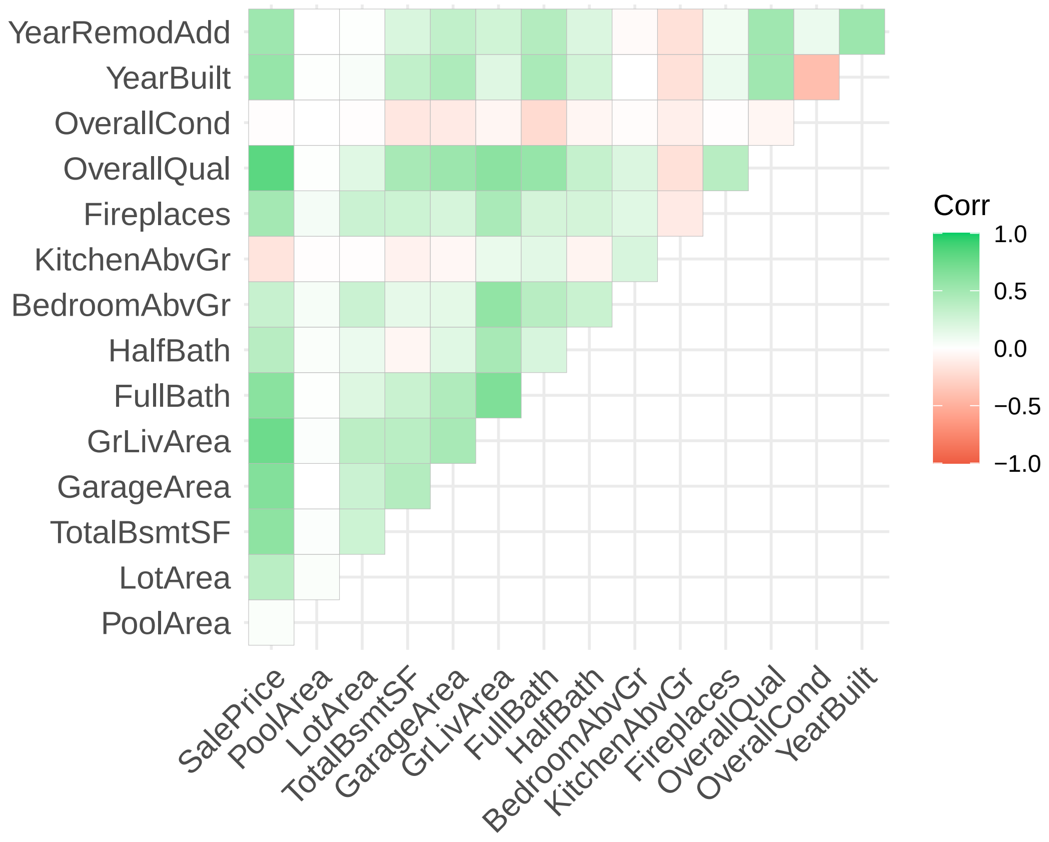


Investigating Neighborhood Variable We also wanted to look at the distribution of houses across neighborhoods, as well as how expensive the homes are in each of the neighborhoods..



This plot reveals that there is a pretty big gradient of neighborhood sizes. To simplify our models and their interpretations, we 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. Additionally, 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.

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 that takes "Y" if the house has been remodelled and "N" otherwise.

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.

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 NA values in ‘GarageFinish’ with “NoGar” and NA values in ‘BsmtFinType1’ with “NoBsmt”.

Our final dataset has a total of 1018 observations. The mean SalePrice of a house in our final data set is:

```
## [1] 175104.7
```

Likewise, the standard deviance in the SalePrice of a house in our final dataset is:

```
## [1] 68375.12
```

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 four different regression techniques: linear regression (with no shrinkage), ridge regression, lasso regression, and XGBoost.

For all models, we **randomly** 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. The response variable is ‘SalePrice’ and the predictors are the other 22 features we included. All numeric variables have been scaled to account for skewness.

Below is a brief description of each method:

1. Linear Regression (with no shrinkage) We trained our simple linear regression model on test data to predict SalePrice using all available predictors. For categorical variables, we create dummy variables.To assess the performance of our model, we predicted the sale price of houses in the test set, and examined the test error rate by measuring the root mean squares error.

2. 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. For categorical variables, we create dummy variables. 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’.

3. 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. For categorical variables, we create dummy variables. We realize that interpreting this might be tricky as only coefficient on some of the classes might be shrunk to 0, but we believe features such as neighborhood are important in determining the sale price.Thus, we chose to create dummies for categorical variables instead of getting rid of them. 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’.

4. 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 **randomly** divideour 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%. Since data was randomly assigned to the test and train set, we did not make sure that the neighborhoods are well represented in both the train and test set (implication of this discussed later in Limitations). The train and the test set are the same for both models. The response variable is 'Neighborhood' and the predictors are the other 22 features we included. The numeric variables will not be scaled since neither Naive Bayes nor Tree-Based method relies on distance between two observations.

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 train 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. The houses were classified into neighborhoods based on the class with the highest probability.

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. Lastly, we decided to prune the classification tree and check whether pruning improved the prediction of the response on the test set.

Results

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

Linear Regression (with no shrinkage)

```
## [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).

To find the Root Mean Squared Error (RMSE), we re-scaled the log price back to USD using $RMSE = \frac{1}{n}\sum(e^{y_i} - e^{\hat{y_i}})^2$, where y=log of price. The RMSE for the linear model is USD 17498.74.

This means that the prediction of houses in the test set is, on average, off by USD 17,498.74. This error is small given that the the standard deviation of the sale price of the houses, as determined earlier in EDA, is USD 68,375.12.

Ridge Regression

```
## [1] 17884.36
```

```
## 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         -0.071
## BldgTypeTwnhsE        -0.026
## HouseStyle1.5Unf      -0.033
## HouseStyle1Story      -0.011
## HouseStyle2.5Fin       0.039
## HouseStyle2.5Unf       0.045
## HouseStyle2Story       0.013
## HouseStyleSFoyer       0.025
## HouseStyleSLvl         0.021
## BsmtFinType1BLQ       -0.015
## BsmtFinType1GLQ        0.027
## BsmtFinType1LwQ       -0.047
## BsmtFinType1NoBsmt     0.019
## BsmtFinType1Rec       -0.026
## BsmtFinType1Unf       -0.057
## PoolArea               0.001
## LotArea                0.027
## TotalBsmtSF            0.065
## GarageArea             0.037
## GrLivArea              0.091
## 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 from logscale to USD, is 17,884.36.

This implies that the prediction of houses in the test set is, on average, off by $17,884.36. Although the RMSE is slightly larger compared to the Linear Regression (with no shrinkage), we believe that this error is small given that the the standard deviation of the sale price of the houses, as determined earlier in EDA, is USD 68,375.12.

Lasso Regression

```
## 50 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)           11.950427527
## NeighborhoodCollgCr    .
## NeighborhoodCrawfor    0.092091271
## NeighborhoodEdwards   -0.008086089
## NeighborhoodGilbert    .
## NeighborhoodGrNoRidge  0.023784540
## NeighborhoodMitchel    .
## NeighborhoodNAmes      .
## NeighborhoodNWAmes     .
## NeighborhoodOldTown   -0.008574331
## NeighborhoodSawyer     .
## NeighborhoodSawyerW    .
## NeighborhoodSomerst    .
## CentralAirY            0.056074124
## PavedDriveY            0.004948370
## GarageFinishNoGar     -0.001182437
## GarageFinishRFn        .
## GarageFinishUnf       -0.001172454
## BldgType2fmCon         .
## BldgTypeDuplex        -0.016866661
## BldgTypeTwnhs          .
## BldgTypeTwnhsE         .
## HouseStyle1.5Unf       .
## HouseStyle1Story       .
## HouseStyle2.5Fin       .
## HouseStyle2.5Unf       .
## HouseStyle2Story       .
## HouseStyleSFoyer       .
## HouseStyleSLvl         .
## BsmtFinType1BLQ        .
## BsmtFinType1GLQ        0.028811115
## BsmtFinType1LwQ        .
## BsmtFinType1NoBsmt     .
## BsmtFinType1Rec        .
## BsmtFinType1Unf       -0.027843356
## PoolArea               .
## LotArea                0.020944803
## TotalBsmtSF            0.051140962
## GarageArea             0.038010841
## GrLivArea              0.121091732
## FullBath               .
## HalfBath               0.006811651
## BedroomAbvGr           .
## KitchenAbvGr          -0.007309638
## Fireplaces              0.019684023
## OverallQual             0.094752176
## OverallCond             0.040388637
## YearBuilt               0.081351828
## Sold_08Y                .
## RemodY                  .

## [1] 18747.09
```

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 from log price to USD, is 18747.09. This implies that the prediction of houses in the test set is, on average, off by $18747.09. This error is small given that the the standard deviation of the sale price of the houses, as determined earlier in EDA, is USD 68,375.12.

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 $RMSE = \frac{1}{n}\sum(e^{y_i} - e^{\hat{y_i}})^2$, where y=log of price.

```
## [1] 19928.01
```

This means that the prediction of houses in the test set is, on average, off by $19928.01. This error is still small given that the the standard deviation of the sale price of the houses, as determined earlier in EDA, is USD 68,375.12.

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

Naive Bayes

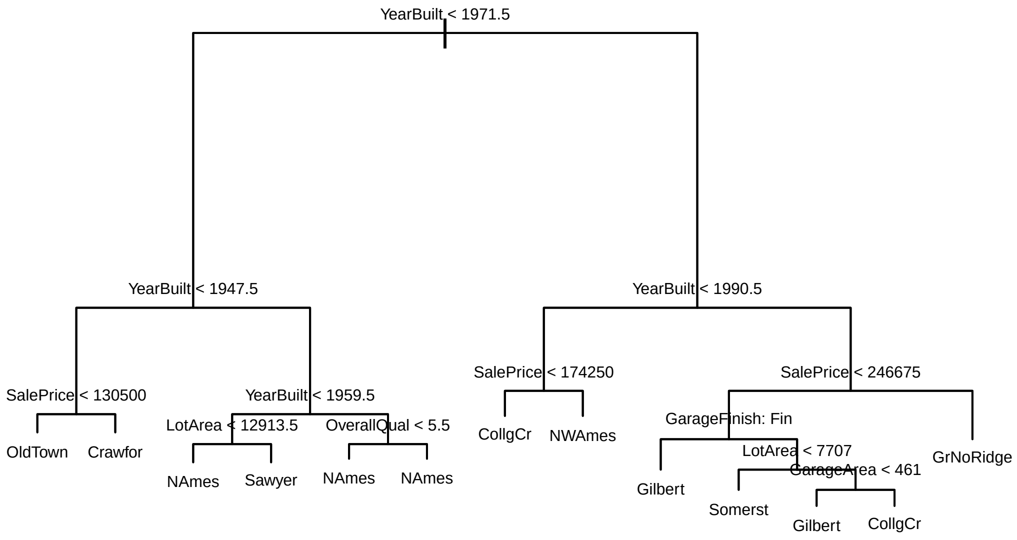
```
## [1] 0.6372549
```

```
##            preds
## true         BrkSide CollgCr Crawfor Edwards Gilbert GrNoRidge Mitchel NAmes
##   BrkSide          5       0       1       0       0         0       0     0
##   CollgCr          0       3       0       1       0         0       0     0
##   Crawfor          1       1       6       0       0         0       0     0
##   Edwards          4       2       2       2       0         0       0     5
##   Gilbert          0       1       0       0       7         1       0     0
##   GrNoRidge        0       1       0       0       1         8       0     0
##   Mitchel          0       4       0       0       1         0       0     1
##   NAmes            2       4       3       3       2         0       0    29
##   NWAmes           0       6       1       0       2         0       0     1
##   OldTown          9       0       1       1       0         0       0     0
##   Sawyer           0       1       0       0       0         0       0     6
##   SawyerW          0       3       0       1       2         0       0     1
##   Somerst          0       1       0       0       0         0       0     0
##            preds
## true         NWAmes OldTown Sawyer SawyerW Somerst
##   BrkSide         0       1      0       0       0
##   CollgCr         0       0      4       0      17
##   Crawfor         0       0      0       0       0
##   Edwards         0       1      1       0       3
##   Gilbert         0       0      0       0       7
##   GrNoRidge       0       0      0       0       4
##   Mitchel         0       0      3       0       2
##   NAmes           0       0      6       1       0
##   NWAmes          0       0      0       0       0
##   OldTown         0       5      0       0       0
##   Sawyer          0       1      2       0       0
##   SawyerW         0       0      0       0       2
##   Somerst         0       0      0       0       7
```

The misclassification rate is ~0.64, 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"  "GarageFinish"
## [6] "GarageArea"
## Number of terminal nodes:  13
## Residual mean deviance:  2.551 = 2043 / 801
## Misclassification error rate: 0.4889 = 398 / 814
```



Based on the tree, it seems like YearBuilt is the most important factor for determining the class of the species since the root node is YearBuilt<1971.5.

The contingency table for the test set is:

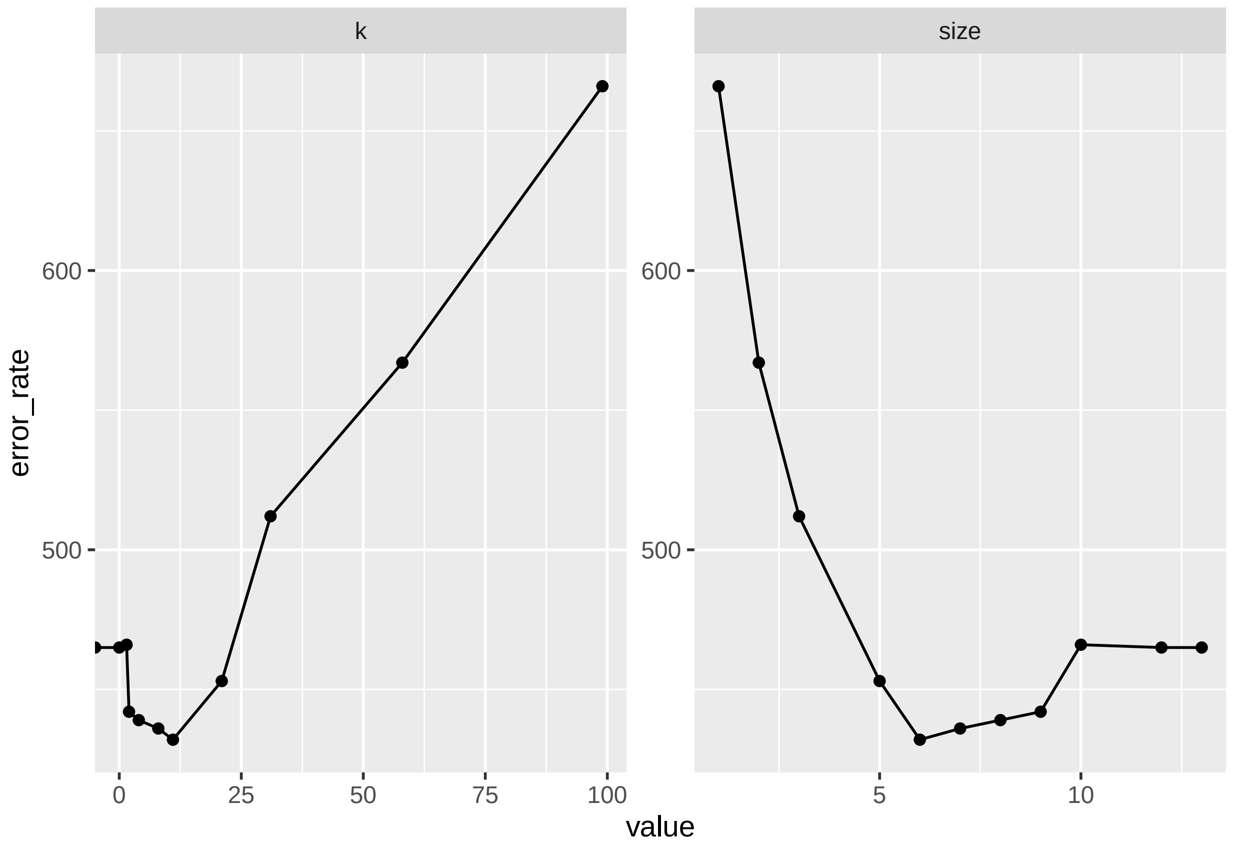
```
##              preds
## true          BrkSide CollgCr Crawfor Edwards Gilbert GrNoRidge Mitchel NAmes
##   BrkSide           0       0       1       0       0         0       0     0
##   CollgCr           0      15       0       0       4         2       2     0
##   Crawfor           0       0       6       0       0         0       0     1
##   Edwards           0       0       2       0       1         0       1     7
##   Gilbert           0       1       0       0      13         1       0     0
##   GrNoRidge         0       0       0       0       0        12       0     0
##   Mitchel         0       1       0       0       2          0       4     3
##   NAmes           0       1       0       0       0          0       1    45
##   NWAmes          0       1       0       0       0          0       1     2
##   OldTown         0       0       6       0       0          0       0     2
##   Sawyer          0       1       0       0       0          0       1     8
##   SawyerW         0       3       0       0       2          0       0     2
##   Somerst         0       4       0       0       0          1       0     0
##            preds
## true        NWAmes OldTown Sawyer SawyerW Somerst
##   BrkSide        0       6      0       0       0
##   CollgCr        0       0      0       0       2
##   Crawfor        0       0      1       0       0
##   Edwards        0       4      1       0       4
##   Gilbert        1       0      0       0       0
##   GrNoRidge      1       0      0       0       1
##   Mitchel        1       0      0       0       0
##   NAmes          1       1      1       0       0
##   NWAmes         6       0      0       0       0
##   OldTown        0       8      0       0       0
##   Sawyer         0       0      0       0       0
##   SawyerW        2       0      0       0       0
##   Somerst        0       0      0       0       3
```

The misclassification rate is:

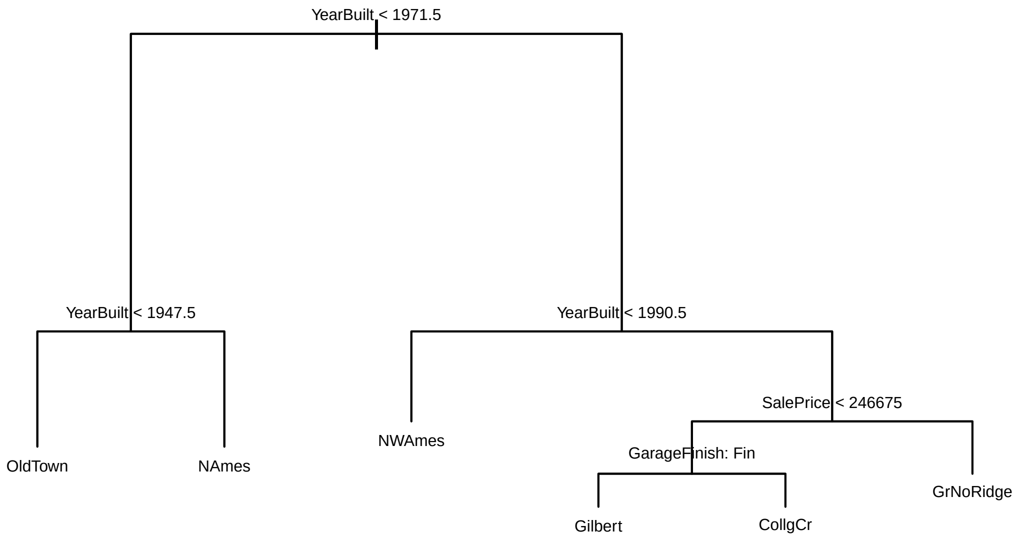
```
## [1] 0.4509804
```

The misclassification rate of ~ 0.45 is significantly better compared to the missclassification rate observed using Naive Bayes. Having said that, misclassification rate of ~ 0.45 is still quite high. So we decided to check whether pruning the tree will improve the results.

```
## $size
##  [1] 13 12 10  9  8  7  6  5  3  2  1
##
## $dev
##  [1] 465 465 466 442 439 436 432 453 512 567 666
##
## $k
##  [1] -Inf  0.0  1.5  2.0  4.0  8.0 11.0 21.0 31.0 58.0 99.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```



Based on the output, the tree with 6 terminal nodes seems to be the best. This is a little surprising since out data consists of a total of 13 neighborhoods; so we expected to see a minimum of 13 terminal nodes. However, we still went ahead and pruned the original tree to obtain the 6 node tree to determine how it would affect the misclassification error.



The misclassification rate on the test set is then:

```
## [1] 0.4803922
```

Pruning increased the misclassification slightly from ~ 0.45 to ~ 0.48.

Discussion

To summarize the results of our regression models, we found that the Root Mean Squared Error (RMSE) was 17498.74 for linear regression model(with no shrinkage), 17884.36 for ridge regression model and 18707.09 for the lasso model and 19928.01 for the XGBoost method. Thus, our best performing model was linear regression (with no shrinkage). This indicates that most of the predictors that were used in the regression impact the sale price significantly; hence, shrinkage did not improve the RMSE. Overall, we think that the our best performing model is a good choice to predict the sale prices as RMSE of 17498.74 is low if we compare it to the standard deviation of the sale price of houses, which was USD 68,375.12.

To summarize the results of our classification models, we found the misclassification rates of the Naive Bayes and Decision Tree models to be approximately .64 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 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.

The fifth main limitation of this project is that for the misclassification problem, we randomly split 80% of our data into train set and 20% of our data into test set. We did not not ensure that the distribution of houses in each neighborhood was the same across the train and the test set, which may have affected our misclassification error rate. To investigate how much this could have affected the misclassification error rate, at the end of our analysis, we quickly checked how many houses belonged to each neighborhood in the test and train set. We found that the distribution was fairly even, which suggested that our misclaafication error rate would not have been significantly different had we maintained the distribution during the test/train split. Having said that, if we had more time, we would have liked to investigate by how much our error rate would change if we split the data into test and train set while mainteaining the distribution of houses in each neighborhood.