Predicting Housing Prices

Devansh Desai
Westen Weiss
Lauren Franks
Kevin Zhang
John Palmieri

The Ames, IA Dataset

- Collected from 2006 - 2010 by Dean De Cock of Truman State University
- Describes sale of residential property in the town
- 80 predictors that extensively describe each house
  - Square footage
  - Bathrooms
  - Land slope
  - Masonry veneer type
  - Heating
  - Existence of an elevator
- 1 Response variable: Sale Price

About the Dataset

- Obtained this dataset through a Kaggle data science competition
- 1460 training observations
- 1459 testing observations without a response variable
- To fully use our dataset, we had to use Kaggle's test prediction submission tool
- The returned value was the root mean square logarithmic error (RMSLE)

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

- Makes it hard to work backwards and tune models specifically for the test data
- RMSLE ranges from 0 to ∞
- 15th best: 0.11032 score | 1,000th best: 0.12753 score

About the Dataset

- Clearly, the RMSLE makes it hard to judge how well the models did
- Logarithmic scale is odd but anywhere between 0.11 - 0.14 is a good score
- Models we tried:
  - Linear regression with forward & backward selection
  - Principal component regression (PCR)
  - Partial least squares regression (PLS)
  - Ridge regression
  - Lasso
  - K-nearest neighbors (KNN)
  - Generalized additive model (GAM)
  - Regression tree with pruning
  - Bagging
  - Boosting
  - Random Forests

Questions of Interest

- What predictors are the most important in predicting the price of a house?
- How do predictors vary across poor and wealthy neighborhoods?
- What month is best to buy a house in terms of $ / sq. ft?
- Is there any apparent impact of the 2008 economic recession on predictors?

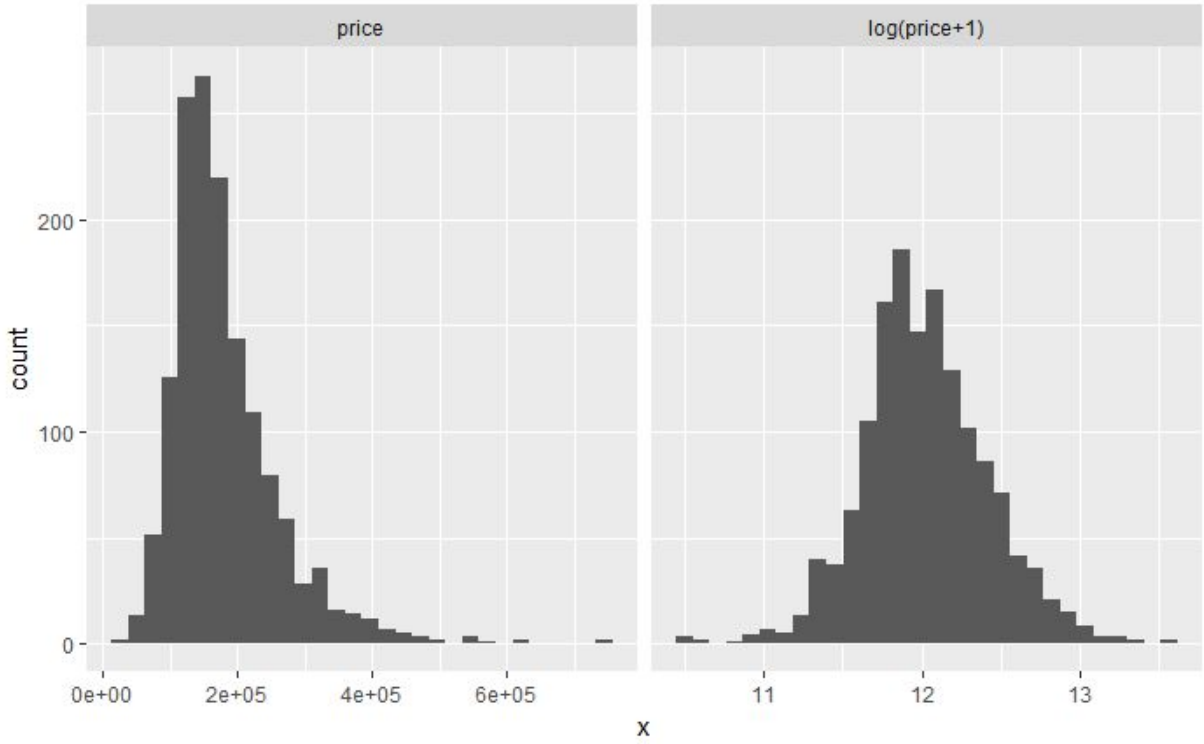
Preprocessing

Preprocessing

- ***Primary goal: reduce width of dataset and prepare data for modeling***
- **Data preparations**
  - Dummy coding categorical variables
  - Correcting skewness in numeric features
  - Imputed mean for missing numeric features
- **Reducing width**
  - Converting scalable categorical variables to numeric type
  - Consolidating dummy coded variables
  - Removing less descriptive categorical variables

Skewness

- fBasics - skewness()
- All numerical variables evaluated
- Log transformation
- Helps model training



Dummy Coding

Neighborhood
CollgCr
Veenker
CollgCr
Crawfor
NoRidge
Mitchel
Somerst

CollgCr	Veenker	Crawfor	NoRidge	Mitchel	Somerst
1	0	0	0	0	0
0	**1**	0	0	0	0
1	0	0	0	0	0
0	0	**1**	0	0	0
0	0	0	**1**	0	0
0	0	0	0	**1**	0
0	0	0	0	0	**1**

Scalable Categorical Features

GarageQual	CollgCr
Ex	5
NA	0
Gd	4
TA	3
FA	2
Gd	4
Po	1

Ex - excellent - 5

Gd - good - 4

TA - average - 3

FA - fair - 2

Po - poor - 1

NA - non-existent - 0

Creative Consolidation

Ext1 AsbShng	Ext1 AsphShn	Ext1 BrkComm	Ext2 AsbShng	Ext2 AsphShn	Ext2 BrkComm
1	0	0	**1**	0	0
0	**1**	0	0	0	**1**
0	**1**	0	0	**1**	0
0	0	**1**	**1**	0	0

Ext AsbShng	Ext AsphShn	Ext BrkComm
1	0	0
0	**1**	**1**
0	**1**	0
1	0	**1**

- 2 ExteriorType variables
  - 16 dummy coded columns each
- Logical OR operation done between corresponding ExteriorTypes

Removing Predictors

- Removed:
  - Scarcely used levels within categorical features (ex. “Mix” level of plumbing predictor)
  - When a category was almost entirely composed of one level
  - Predictor columns that were almost entirely NA, 0, or missing (ex. Only 20/1460 observations present, like “Alley Access”)
- Checked low correlation b/w response before removal

After dummy coding we had 288 predictors

After all preprocessing, we had 212 predictors

Models

Model Results: Forward & Backward Selection

- Leaps package
- Regsubsets function
- Problems:
  - Summaries only gave best 4 variable model
  - Considering 213 variables
- Ran best four variable model using lm() function
- R squared value of 0.8219
- All significant variables

```
Call:
lm(formula = y ~ LotArea + OverallQual + YearBuilt + GrLivArea,
    data = trainy_processed)

Residuals:
     Min       1Q   Median       3Q      Max
-1.48273 -0.08452  0.00975  0.10324  0.55851

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.3643868  0.3829851   0.951    0.342
LotArea     0.1464273  0.0092937  15.756   <2e-16 ***
OverallQual 0.1252794  0.0048829  25.657   <2e-16 ***
YearBuilt   0.0033905  0.0001815  18.681   <2e-16 ***
GrLivArea   0.3959281  0.0181941  21.761   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1688 on 1455 degrees of freedom
Multiple R-squared:  0.8219,    Adjusted R-squared:  0.8214
F-statistic:  1679 on 4 and 1455 DF,  p-value: < 2.2e-16
```

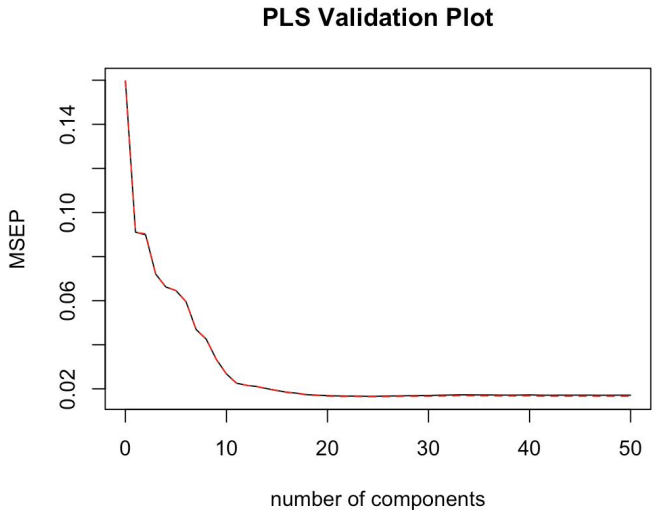
Forward & Backward Selection: Considering Polynomial & Interaction Terms

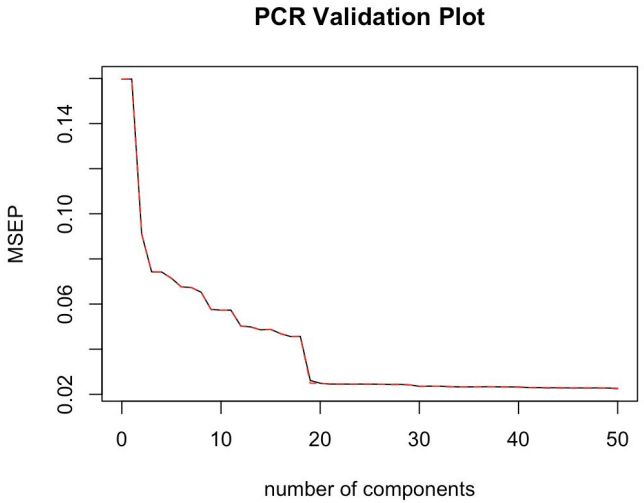
- Adding interaction variables and polynomial terms to increase R squared value
- R squared = 0.8534
- All variables significant except 3rd degree Overall Quality

```
Call:
lm(formula = y ~ LotArea + poly(OverallQual, 5) + YearBuilt +
    GrLivArea * LotArea + OverallCond, data = trainy_processed)

Residuals:
     Min       1Q   Median       3Q      Max
-1.31818 -0.07818  0.00816  0.08718  0.47244

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)             -6.9772809  1.5309329  -4.558 5.61e-06 ***
LotArea                  0.6805983  0.1604695   4.241 2.36e-05 ***
poly(OverallQual, 5)1    5.7256682  0.2420238  23.657  < 2e-16 ***
poly(OverallQual, 5)2    1.1137460  0.1642413   6.781 1.73e-11 ***
poly(OverallQual, 5)3    0.2049034  0.1579315   1.297 0.194694
poly(OverallQual, 5)4   -0.5175955  0.1550377  -3.339 0.000864 ***
poly(OverallQual, 5)5   -0.8087564  0.1549219  -5.220 2.04e-07 ***
YearBuilt                0.0046544  0.0001837  25.333  < 2e-16 ***
GrLivArea                1.1318309  0.2025952   5.587 2.76e-08 ***
OverallCond              0.0658551  0.0041655  15.810  < 2e-16 ***
LotArea:GrLivArea       -0.0749518  0.0220638  -3.397 0.000700 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1535 on 1449 degrees of freedom
Multiple R-squared:  0.8534,    Adjusted R-squared:  0.8524
F-statistic: 843.7 on 10 and 1449 DF,  p-value: < 2.2e-16
```

Model Results: PCR & PLS

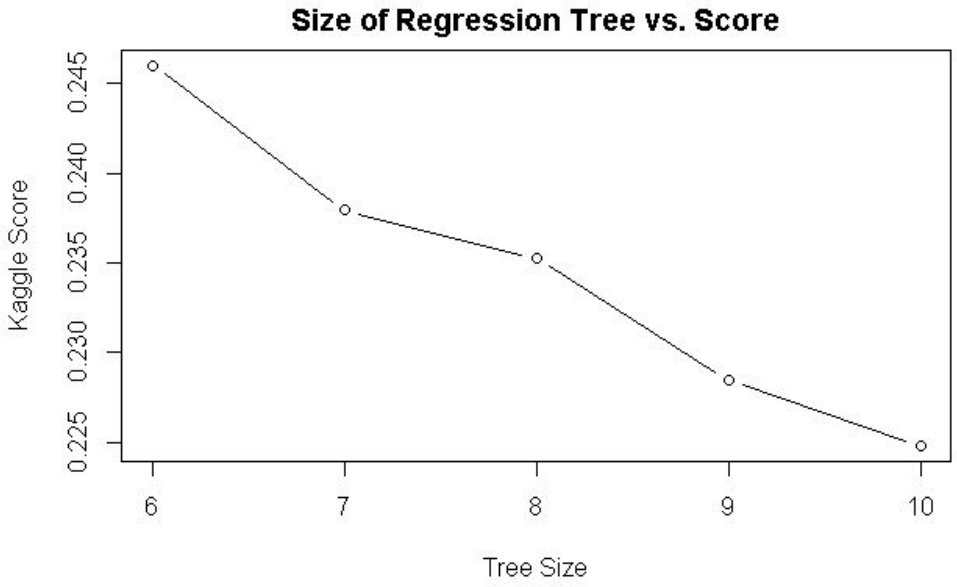
- PLS Package
- PCR and PLSR functions
- Tuning parameter used: ncomp
  - Number of components to include in model
  - Analyzed 5, 10, 20, 50, and 200 components in model
- PCR Kaggle score = 0.4188
- PLS Kaggle score = 0.3301





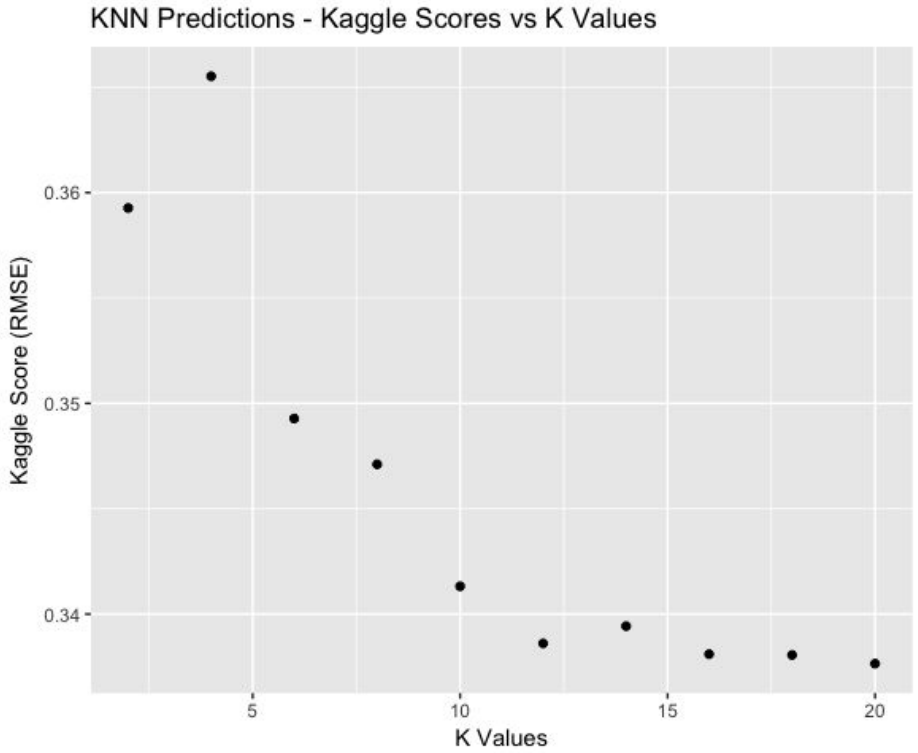
Model Results: Regression Tree

- Tree package
- Full tree has 10 splits
- More splits improves accuracy on train and test set
- Kaggle score: 0.2248
- Splits occurred on variables:
  - OverallQual
  - GrLivArea
  - TotalBsmtSF
  - CentralAirY
  - YearBuilt
  - GarageCars



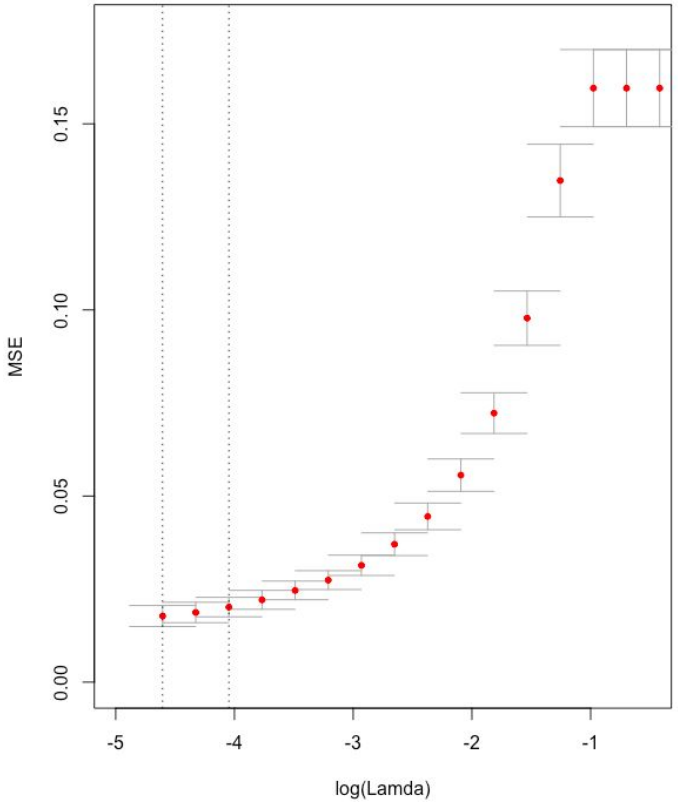
Model Results: KNN

- Class package
- Ran models for k = 2 to k = 20 in steps of 2 and k = 20 to k = 50 in steps of 5
- Model fit improved with increase in k parameter until k = 20
- Fit leveled out at around 0.34 for k > 20
- Kaggle Score: 0.3376



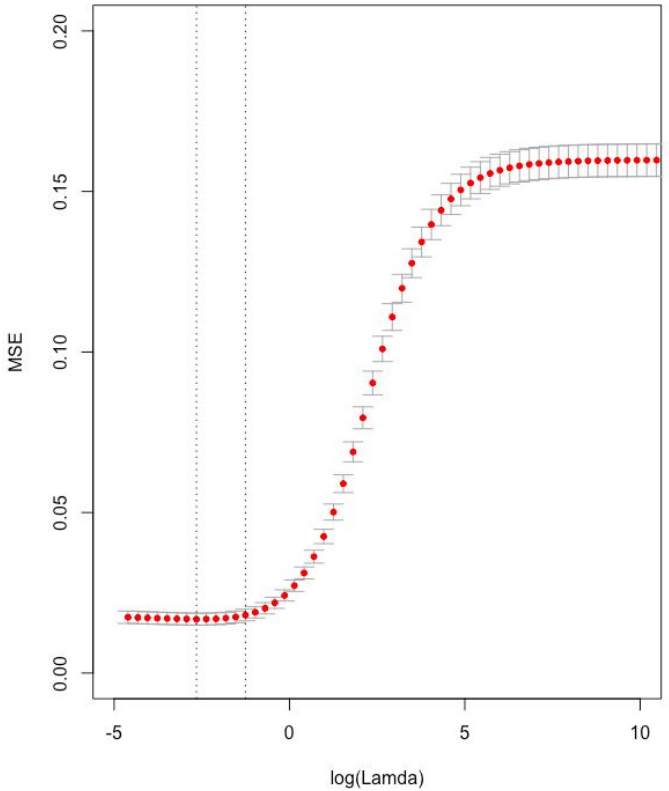
Model Results: Lasso

- GLMNET Package
- Function: lasso.pred()
- Tuning Parameter: Lambda (λ) = 0.09326033
    - Value for Lambda that produced best prediction
- Kaggle Score: 0.13053



Model Results: Ridge Regression

- GLMNET Package
- Function: ridge.pred()
- Tuning Parameter: Lambda (λ) = 0.07054802
  - Value for Lambda that produced best prediction
- Kaggle Score: 0.13243

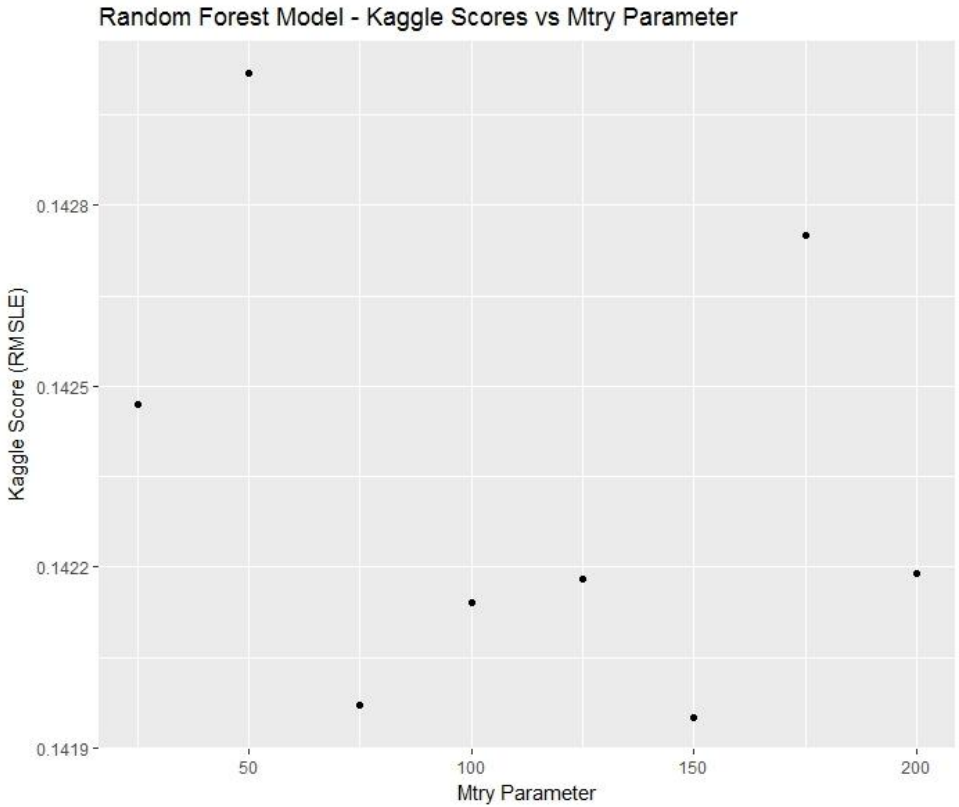


Model Results: Bagging

- Bagging uses bootstrapped data and analyzes all 212 predictors when deciding what predictor to split on in its trees
  - mtry = 212
  - trees = 20,000
- It's hard to overfit with a model such as bagging
- Using more trees doesn't really overfit but rather reduces test variance as the prediction gets averages across all of the trees
- Scores did not change at all when using 2000, 5000, or 20,000 trees
- Kaggle score: 0.14546

Model Results: Random Forest

- Tried various mtry parameters ranging from 25 - 200
- Scattered results
  - mtry = 150 was the best
  - mtry = 75 close second
- Kaggle score: 0.14195

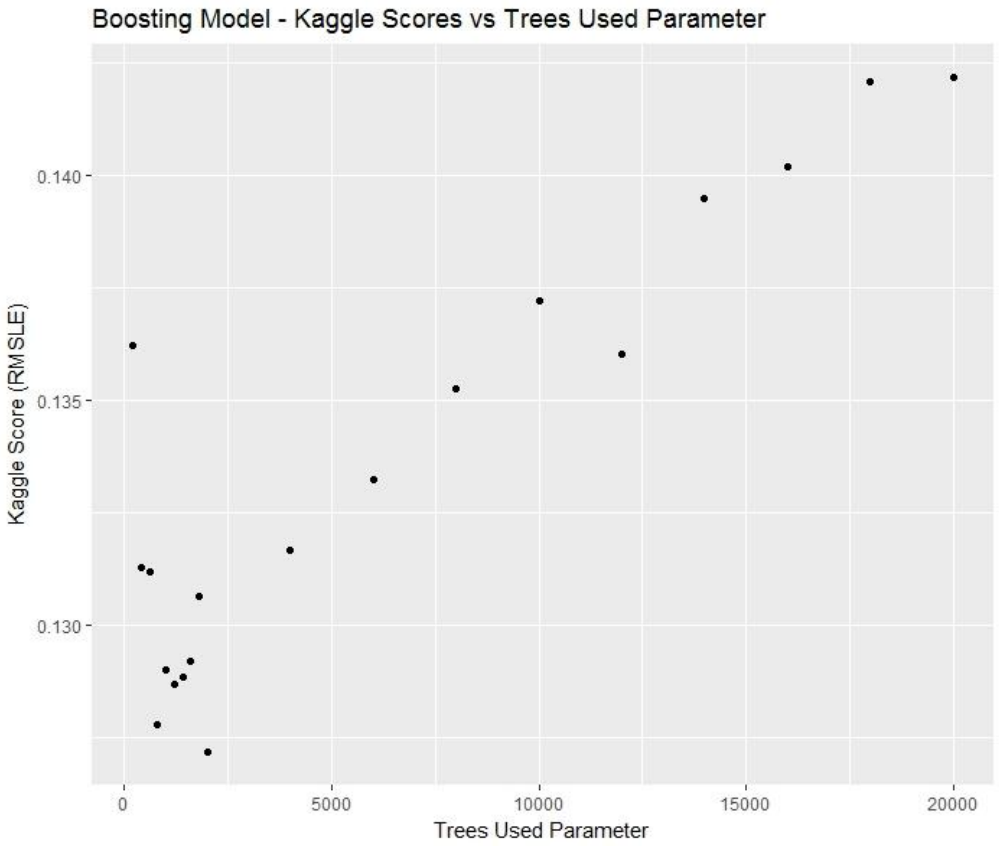


Model Results: Boosting

- In R, we can vary the learning rate (shrinkage parameter), the number of trees use (n.trees), or both
- Both parameters control the amount of learning
- Set shrinkage = 0.1 and varied n.trees parameter
- Possible to overfit with boosting

Model Results: Boosting

- First tried trees from 2000 - 20,000 in steps of 2,000
- Then tried trees from 200 to 2000 in steps of 200
- Kaggle score: 0.12717



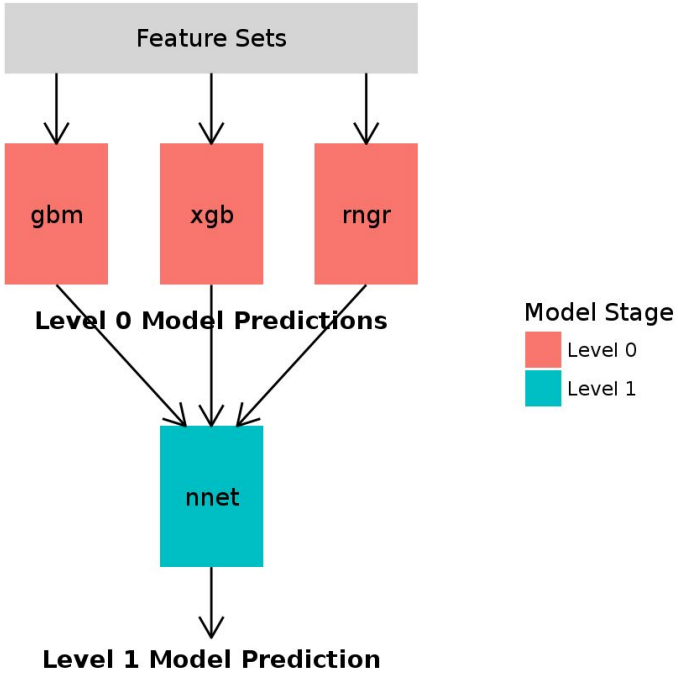
Model Results: Variable Importance for Boosting

Predictor	Relative Influence
OverallQual	30.2705
GrLivArea	17.9921
TotalBsmtSF	9.3658
OpenPorchSF	4.2911
BsmtFinSF1	3.8385
YearBuilt	3.6752
GarageArea	2.8878
GarageCars	2.8863
LotArea	2.7715
X1stFloorSF	2.4284

Additional Analysis

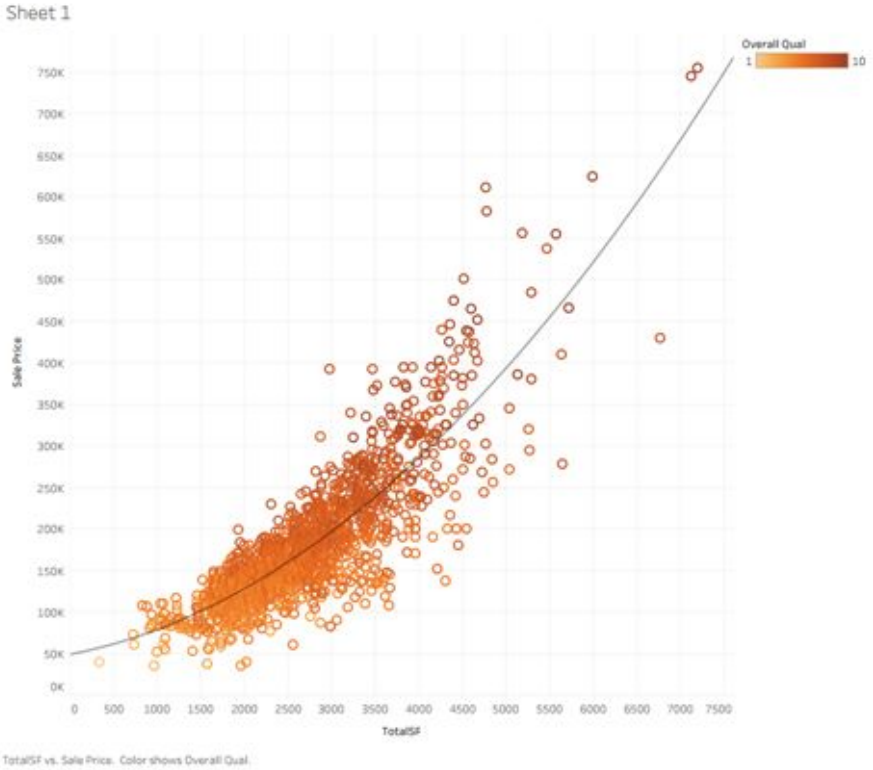
Stacked Model

- Three “level 0” models
  - Gradient boosting
  - Extreme gradient boosting
  - Random forests
- Neural network - “level 1”
  - Combines output from level 0 models
  - Lowers score to ~0.126



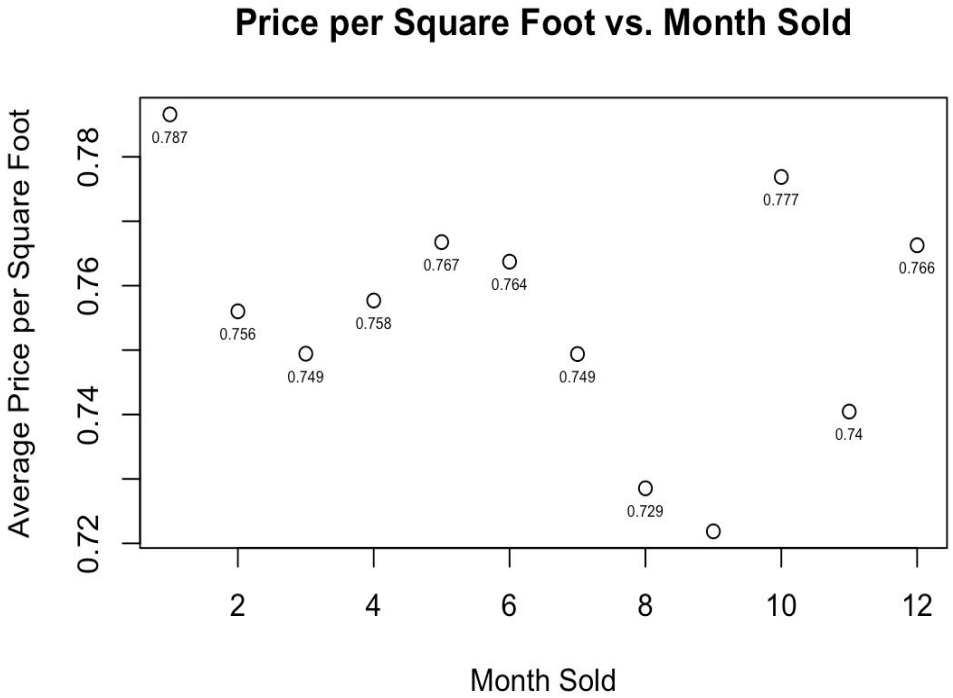
Custom Numeric Feature : Total Square Footage

- Total SF
- Polynomial regression
- Used Tableau
- Kaggle score: .23110
  - Full decision tree: .22482



What month is best to buy a house?

- Created a new variable
  - NewVar = Price/total square feet
  - Measures the cost-effectiveness of a purchase
- Aggregated data by month and computed summary statistics
- Plotted to view any trends in data



Neighborhood Analysis

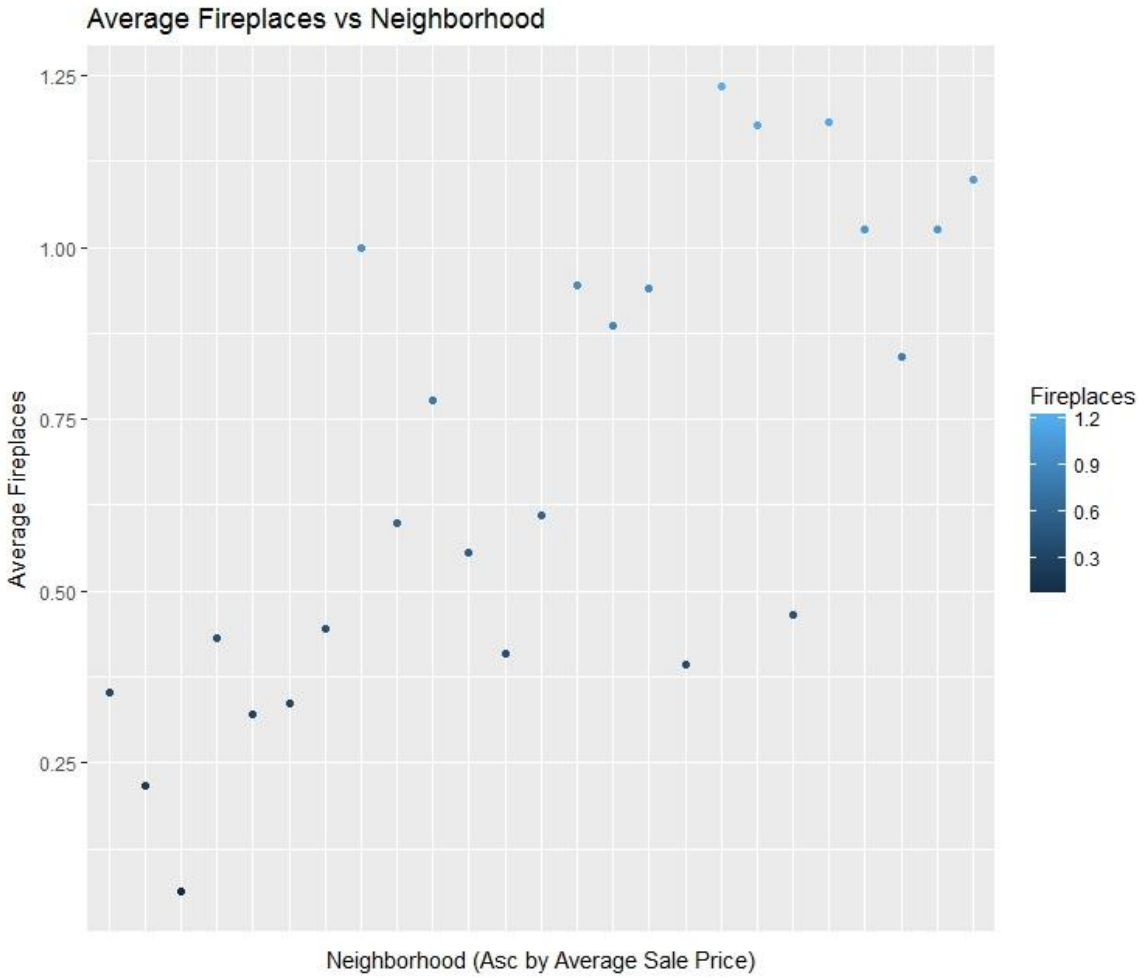
- To rank neighborhoods by wealth, we can average the sale price for each neighborhood and sort by the average
- Wealthiest to poorest neighborhoods by above method
  - 1) Northridge
  - 2) Northridge Heights
  - ...
  - 25) Meadow Village
- Using these rankings, we can observe a neighborhood's effects on various other predictors in the model

Neighborhood Analysis

- Each data point here describes the average sale price group by the neighborhood
- Data is in ascending order

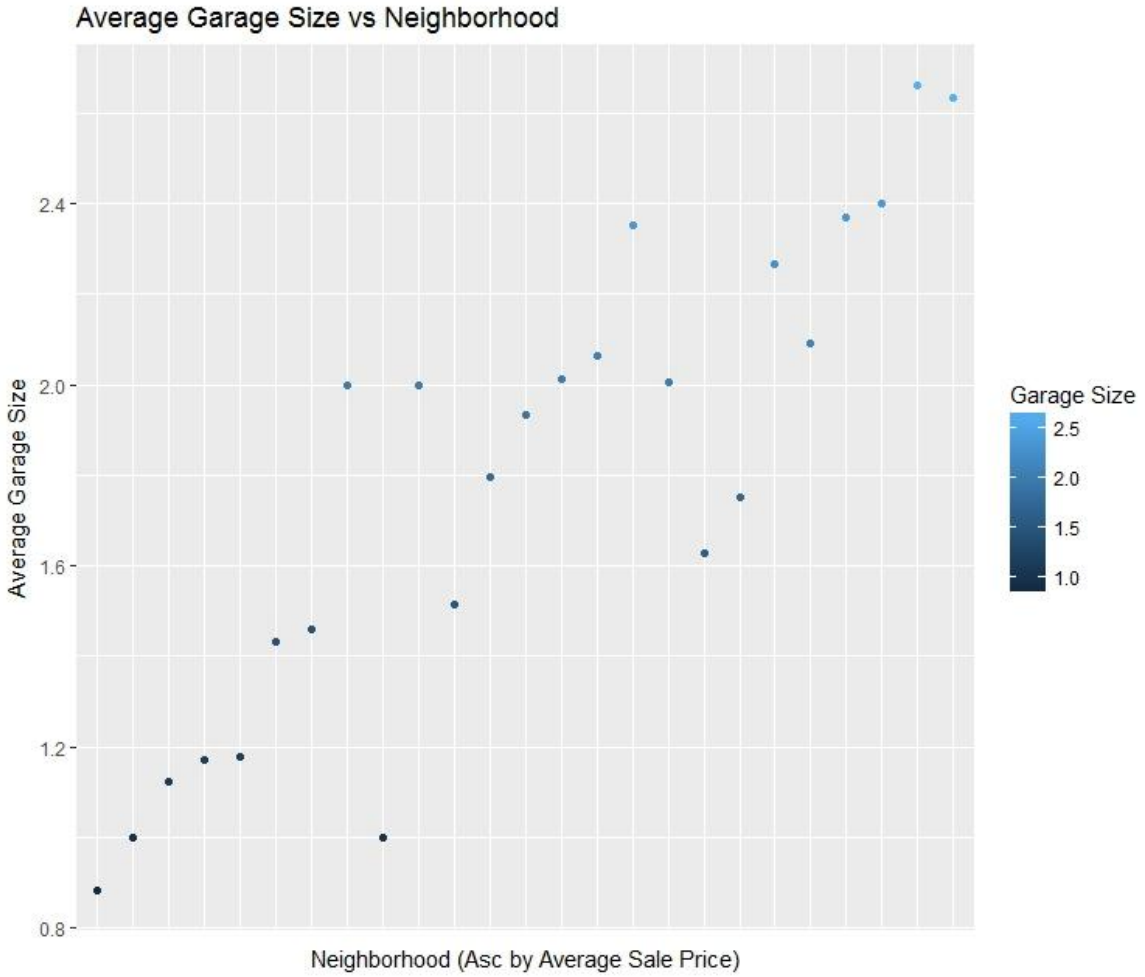
Neighborhood Analysis

- Correlation: 0.6841



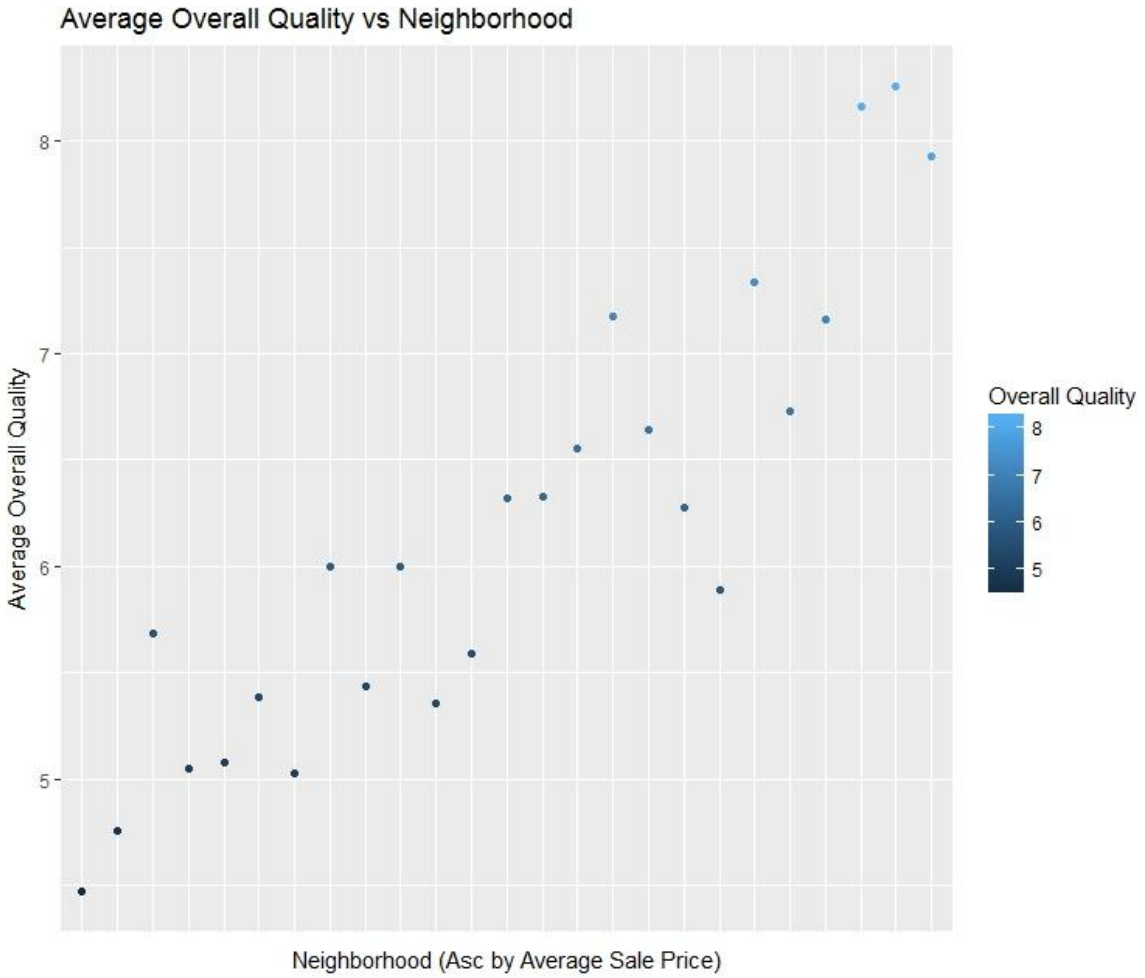
Neighborhood Analysis

- Correlation: 0.8567



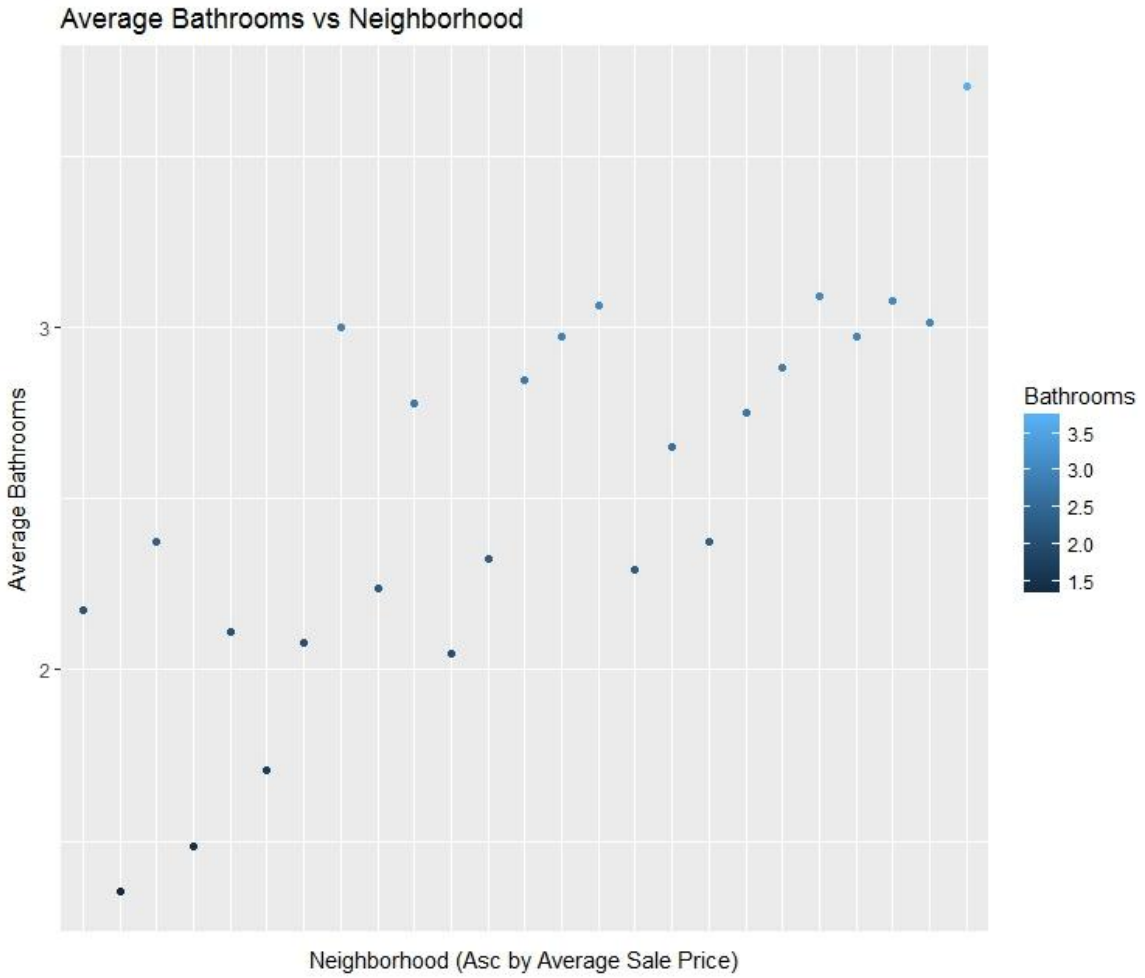
Neighborhood Analysis

- Correlation: 0.9283



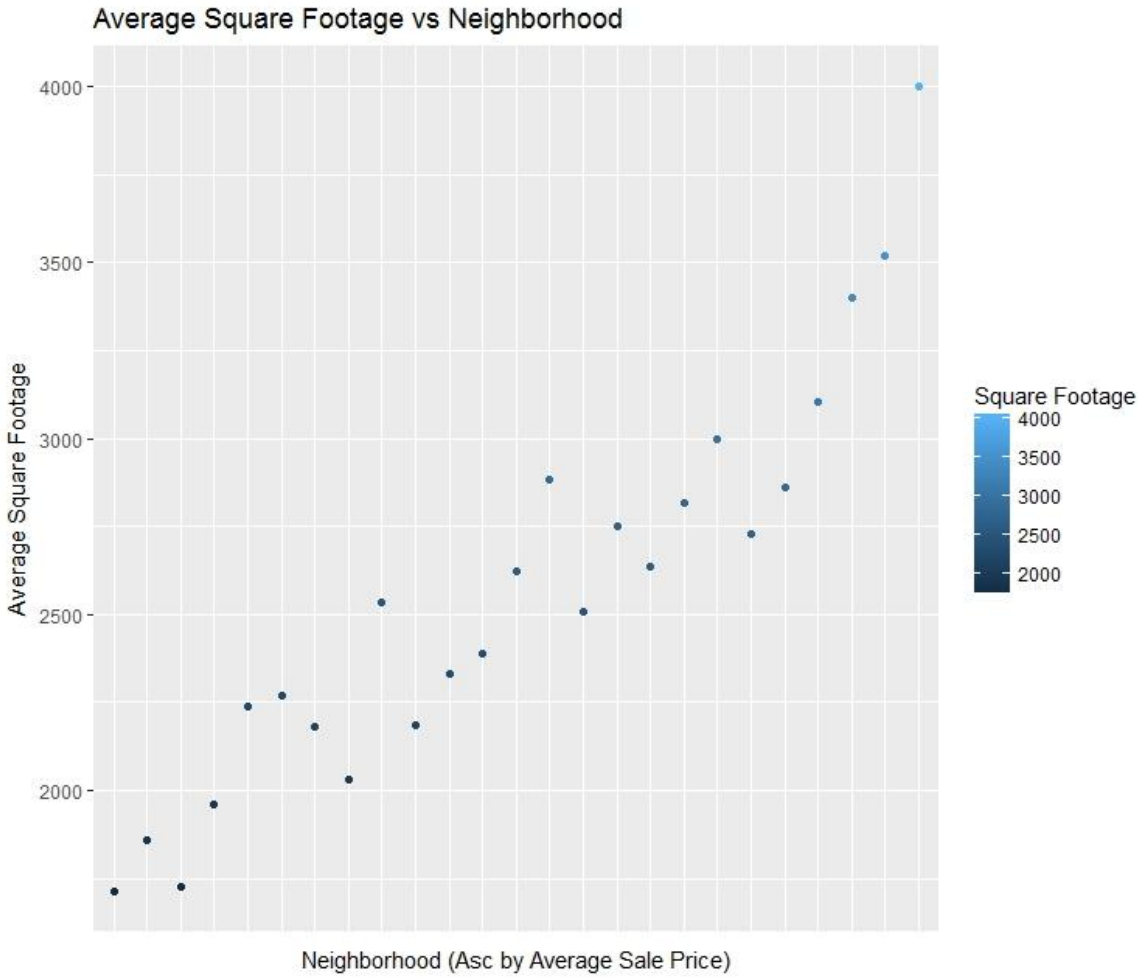
Neighborhood Analysis

- Correlation: 0.7646



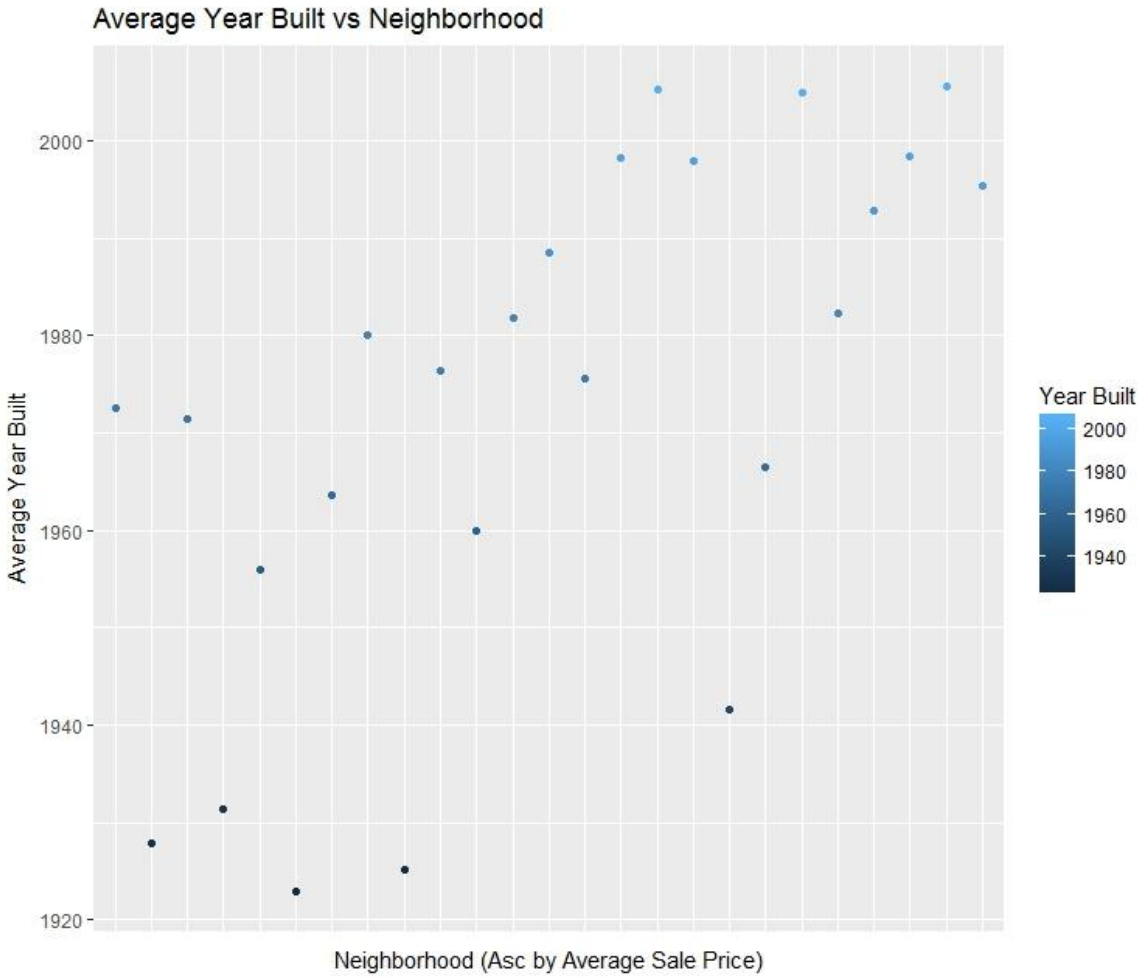
Neighborhood Analysis

- Correlation: 0.9635



Neighborhood Analysis

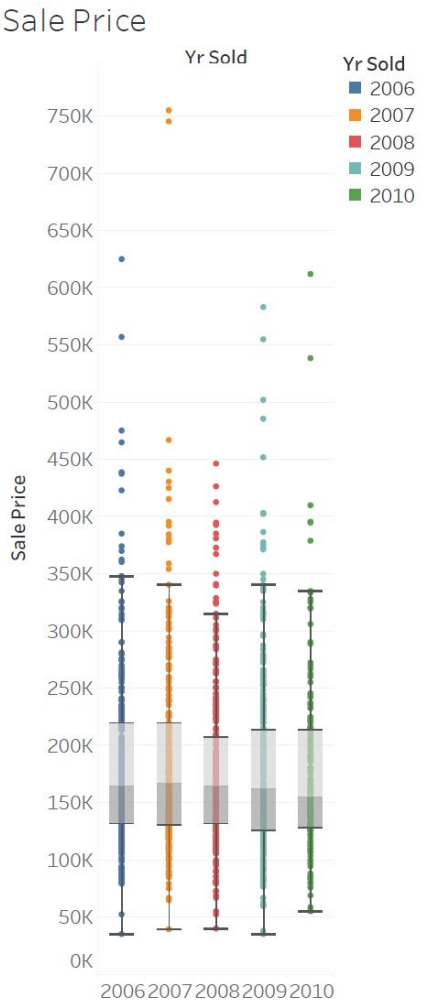
- Correlation: 0.6240



Economic Recession Analysis

- Data collected for houses sold between 2006-2010
  - Investigate changes around the 2008: US recession
  - Consider the changes in relationship between year sold and “recession-related” predictors
  - Plot year sold against: sale condition, sale type, sale price

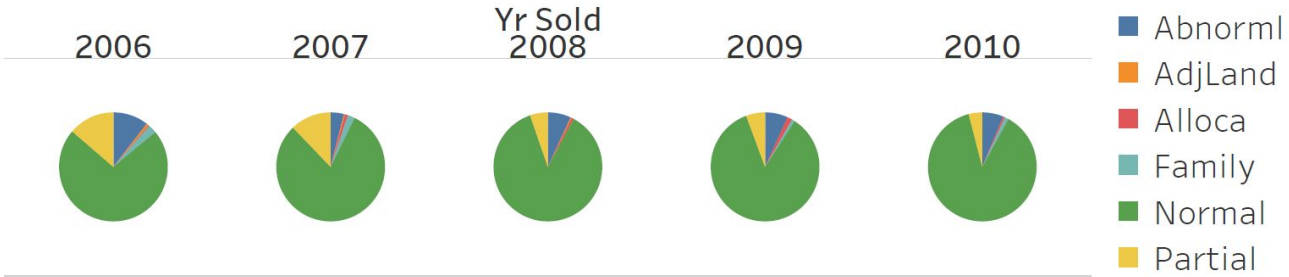
Recession Analysis



Recession Analysis cont.

Sale Type

Sale Type (color) broken down by Yr Sold.



Sale Condition

Sale Condition (color) broken down by Yr Sold.

Conclusion

Conclusion

- Tree Models most effective - why?
  - Within-model variable selection highly valuable
  - Trees identified similar sets of important variables
- Fairly successful in competition
  - Our boosting model's score places 965th in a pool of 2,225 teams

Model	Kaggle Score (RMSLE)
PCR	0.4188
KNN	0.33765
PLS	0.33010
GAM	0.30966
Regression Tree	0.22482
Bagging	0.14512
Ridge Regression	0.13243
Random Forest	0.13200
Lasso	0.13053
Boosting	0.12717
*Stacking	0.12600

Conclusion

- Most important in calculating sale price:
  - Overall home quality
  - Ground floor living area
  - Total basement square footage
- Certain seemingly irrelevant variables still significantly contributed

Predictor	Relative Influence
OverallQual	30.2705
GrLivArea	17.9921
TotalBsmtSF	9.3658
OpenPorchSF	4.2911
BsmtFinSF1	3.8385
YearBuilt	3.6752
GarageArea	2.8878
GarageCars	2.8863
LotArea	2.7715
X1stFloorSF	2.4284

Conclusion

- Representative of the US housing market?
- Safe to say best models would predict sale prices in similar, rural US towns
- Good grasp of important predictors