Machine Learning Group Project-
Predicting Housing Price

Group members:
Kelsey MacMillan, April Wang, Keyang Zhang, Roger Wu

Problem Statement

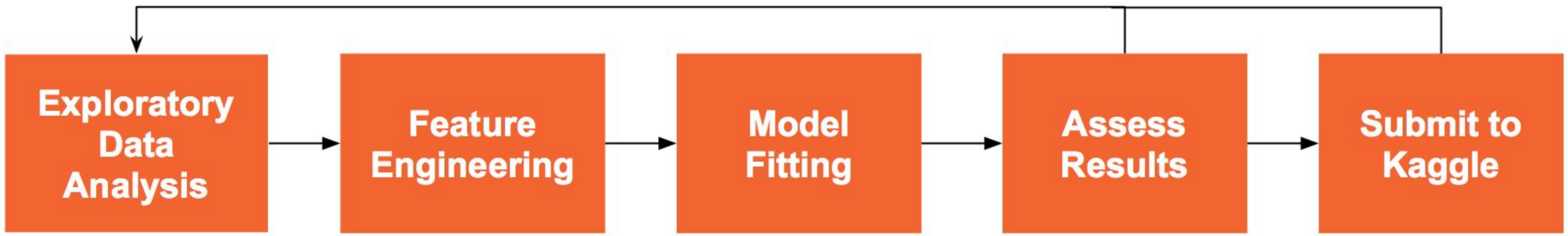
Kaggle Competition:

Predict home prices in Ames, Iowa based on 79 different features.

- square footage
- lot size
- year home was built
- number of bedrooms
- neighborhood etc.

Introduction

- Target: Sale Price (log transformed)
- 79 total features
  - 23 categorical, 23 ordinal, 14 discrete, 19 continuous
  - Over 200 total features after one hot encoding of categorical/ordinal
- 1460 training observations



Hypotheses

1. Not all variables are useful.
   - E.g. first floor square footage, number of rooms, second floor square footage, and lot size likely contain redundancy (multicollinearity)
   - Model needs to include regularization and/or feature selection

2. Most predictors can be linearized with appropriate transformation.

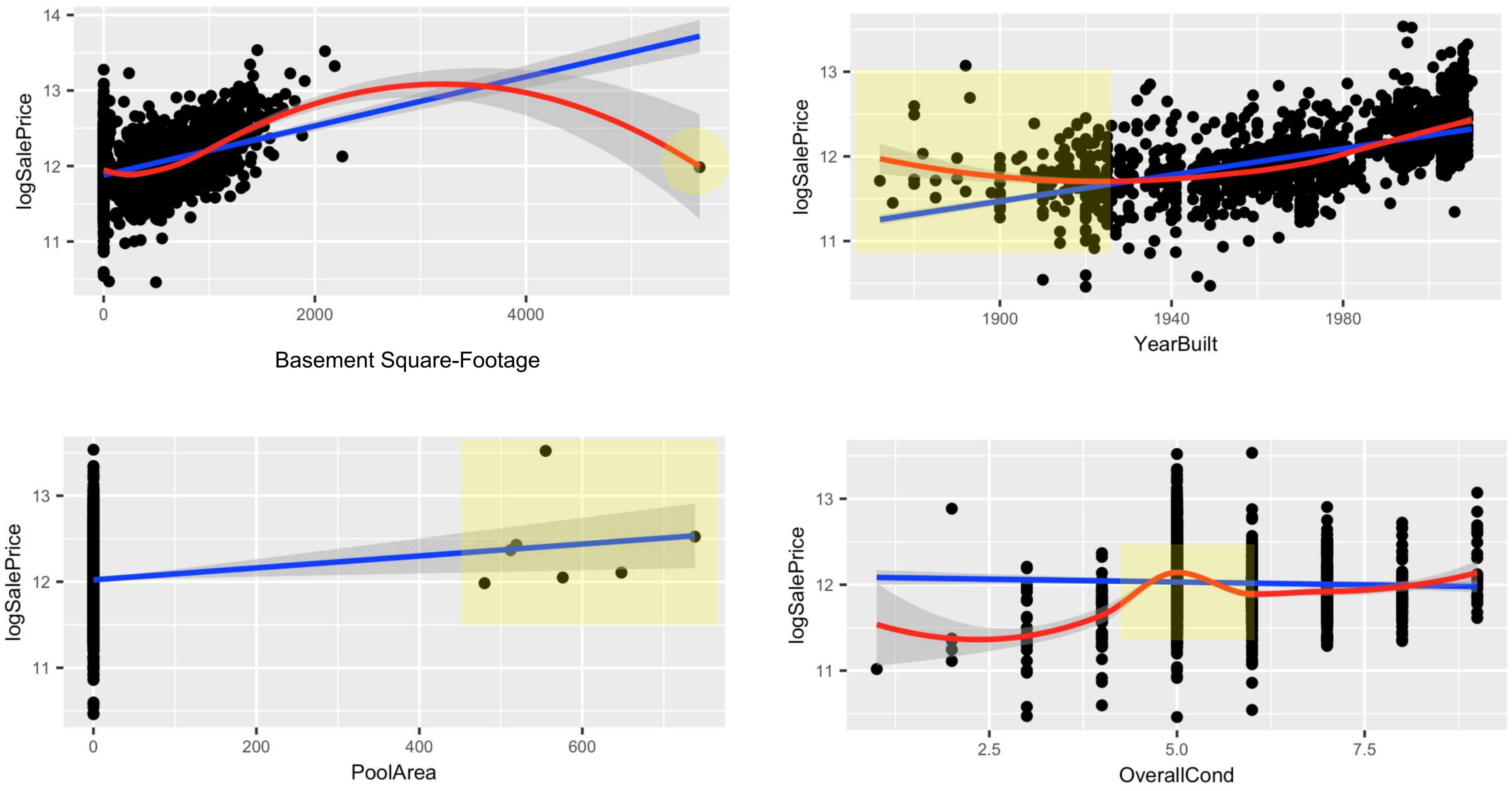
   - Allows us to use a low-variance linear model without major bias penalty

Feature Engineering

- Manually designing what and how features should enter the models
- Simple feature engineering, but huge performance gains

Feature Engineering - Implementation

- Removing skewness via log transformation
- Nonlinear relationships
- Interaction terms
- Grouping and/or dropping low frequency categories


logSalePrice
14
13
12
11
0
2000
4000
Basement Square-Footage
logSalePrice
13
12
11
1900
1940
1980
YearBuilt
logSalePrice
13
12
11
0
200
400
600
PoolArea
logSalePrice
13
12
11
2.5
5.0
7.5
OverallCond

Evaluation

- RMSE using logged values
  - Errors in predicting expensive houses and cheap houses will affect the result equally
- 1460 training observations / 1460 test observations (evaluated by Kaggle)

Algorithms Considered

Linear Methods

- Lasso
- Principal Components Regression

Tree Methods

- XGBoost
- Random Forest
- AdaBoost

Ensemble Models

- Average of Lasso and Decision-Tree Based Models

Lasso

Pros

- Feature selection
- Easy to interpret
- Preferred when *n* (number of observations) is small relative to *p* (number of features)

Cons

- Linear model
- Outliers/leverage points
- Multicollinearity affects interpretation

XG Boosting - How it works

- Performance-optimized implementation of gradient boosting on decision trees
- Fit sequential weak learners to the residuals of the previous set of weak learners
- Slightly different from Adaboost, which re-weights the observations in the loss function

XG Boosting

Pros

- Highly-tunable
- Nonparametric / Highly Flexible

Cons

- Many parameters to tune
- Worse than random forest or AdaBoost out-of-box
- Risk of overfitting

Ensemble Model

- Combine predictions from multiple (uncorrelated) models
- Outputs (predictions) of individual models act as inputs for the ensemble model

Ensemble Model

Pros

- Average out biases of linear model with nonlinear information
- Reduces variance by using uncorrelated models

Cons

- Black-box (uninterpretable)
- Weight-tuning is unstable

Other Models Considered

- Random forest
  - Too many features relative to number of observations
  - Nonparametric methods tend to perform poorly in these cases
- AdaBoost (Decision Trees)
  - Less tunable parameters
  - Sensitive to outliers
- Principal components regression
  - Similar to lasso but consistently performed worse

Final Model

Weighted Average of Lasso (75%) and XGBoost (25%)

- Weights picked based on CV error

Lasso - strong predictor if linear relationship holds

XGBoost - captures the nonlinear relationships (including interactions)

Results

Algorithm	CV RMSE	Kaggle Ranking
Ensemble	0.116	7%
Lasso	0.117	12%
PCR	0.123	28%
XGBoost	0.124	30%
Random Forest	0.139	54%
AdaBoost	0.171	77%

Conclusion & Experience Learned

A few notes regarding feature engineering

- Feature selection algorithms are not perfect
  - EDA-based feature engineering was superior
- Outlier points and unbalanced categories can heavily influence linear models
- With well-linearized predictors, non-parametric models are actually worse

Thank you!

Achievement:

Top 7% on Kaggle Competition!