House Pricing Prediction

DS502 Final Project

Yufei Lin, Jingfeng Xia, Jinhong Yu, Shijing Yang, Yanze Wang

Nov 29 2020

Introduction

Description of the Problem

Being able to predict the price of a house tends to be an important skill for both the seller and consumers. For the seller, they could make better sales and consumers could have better understanding when they try to make a purchase. Therefore, in this project, we are planning to make prediction of house price based on the 79 different predictors provided by Kaggle dataset to determine values of residential homes in Ames, Iowa. We are planning to use PCA, Cross-validation, Linear regression, and Ridge and LASSO regression for prediction. Furthermore, we will be utilizing decision trees for more accurate prediction result as a comparison with other methods.

Description of the Dataset

In terms of the dataset, the entire data set consists of two pieces of data organized as training data set and test data set respectively. Whereas for each of the dataset, approximately 80 columns corresponding parameters would be evaluated with the prediction of house price. Some noteworthy predictors include the location classification, utilities, environment of neighborhood, house style and condition, area, year of built, and number of functioning rooms. There are over 1400 row data points in both the training data set and the test data set. The sale prices in the train dataset are given as a parameter in the form of five or six figure full flat integers. The test data set will be applied to different regression models in order to distinguish the disparities of different model performances.

Approaches

Given that our data is aimed at predicting Sale Price of a house, it is unreasonable to require a model to fit the exact value of the dataset but only to reach an estimation within a certain range. Therefore, we have decided to use both regression and classification approaches to look at the problem. For regression method, we are going to look at if a prediction is within the range of the actual price $\pm 5\%$, we will say it is an accurate prediction. For classification prediction, we will be tagging the data into several different groups, and would be fitting the threshold accordingly with models like SVM.

Data Processing

Read in Data

We have chosen to eliminate the Id column from this dataset because Id has nothing to do with our prediction and would mess up our prediction. And we store the "test.csv" data in a variable **vault** for future testing, and save the "train.csv"" data from Kaggle to a variable named **HousePricing** for further processing.

Pairs of Categories (Iris)

```
## [1] "Hi"
```

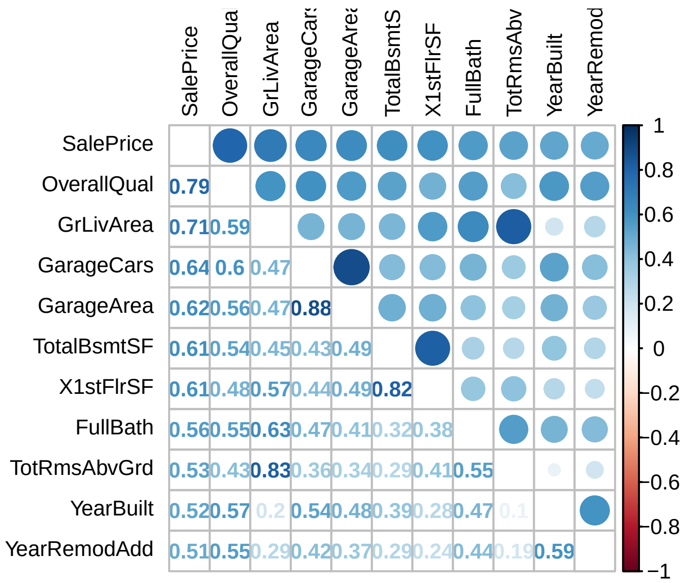
Feature Engineering

We choose to take the log of Sale Price, our y-value.

In this section, we convert all missing value based on the following rules:

1. Categorical: fill in most common
2. Numeric: fill in median/average

Convert all train to HousePricing



Boostraping

Seperate into Test and Training Set

Spearate by 70% train, 30% test.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.46   11.78   12.03   12.03   12.30   13.52
```

```
##   0   1
## 538 483
```

Hierachical

Each team member bootstraps training data.

Prediction Algorithms

We choose to use PCR, Random Forest, GAM, Lasso and Ridge, Splines and Linear Regression to look at how each model would be suitable for our regression analysis.

Each model needs a cross validation algorithm Remember to report RMSE

Regression Methods

1. PCR (Iris)

Cross Validation

2. Random Forest

Explanation

We have chosen this model because random forest is based on a collection of decision trees that could help us get better understanding of which tree and division contribute to which section such that we could have a better picture of the overall importance of each different factor in the prediction.

Prepare Model

We have 199 independent variables in the data set, therefore we have set mtry(Number of randomly selected variables for each split) to be the square root of that number for maximum performance of the model.

The following is the result from Random Forest algorithm:

Call: randomForest(formula = SalePrice ~ ., data = train, mtry = sqrt(totalIV), importance = TRUE) Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 7

Mean of squared residuals: 0.01256672 % Var explained: 92.31

Check Accuracy

We then need to check accuracy, as assumed before, we would look at whether the predicted data is within the ±5% range. The following is the result.

```
## [1] "We have the accuracy of the model approximately 99.54%"
```

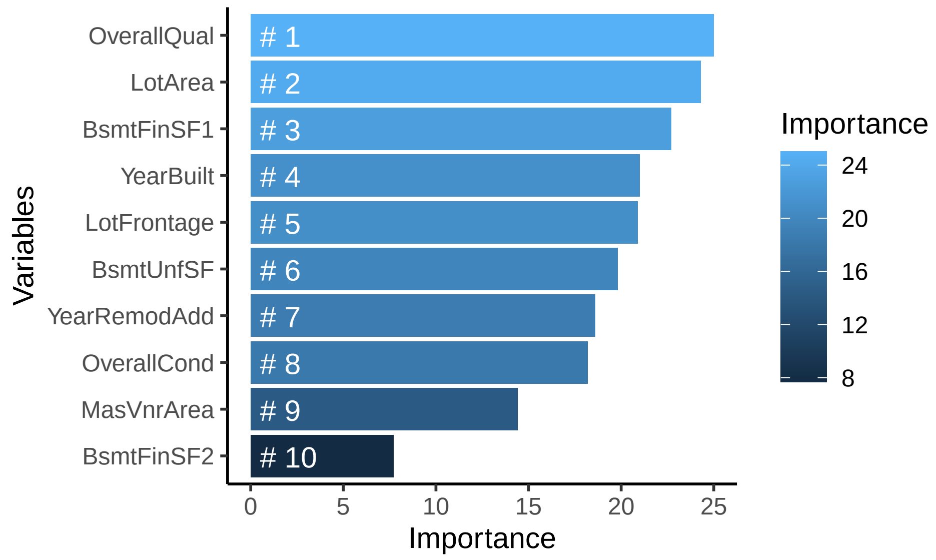
Error Metrics

Then let us take a look at the MSE of this model:

```
## [1] "We have the MSE of the model approximately 0.01120"
```

Variable Importance

Here we are going to show the top 10 most important variables in predicting sale price of a house.



From the random forest analysis, we have discovered that the top three most important factors for predicting sale price are the following:

1. OverallQual (Overall Quality of the building)
2. BsmtFinSF1 (Type 1 finished square feet)
3. LotArea (Area of Parking Lot of the Building)

Cross Validation

In the cross validation, we have chosen to look at R^2, RMSE and MAE.

R2	RMSE	MAE
0.9235	0.1058	0.06361

3. GAM

Explanation

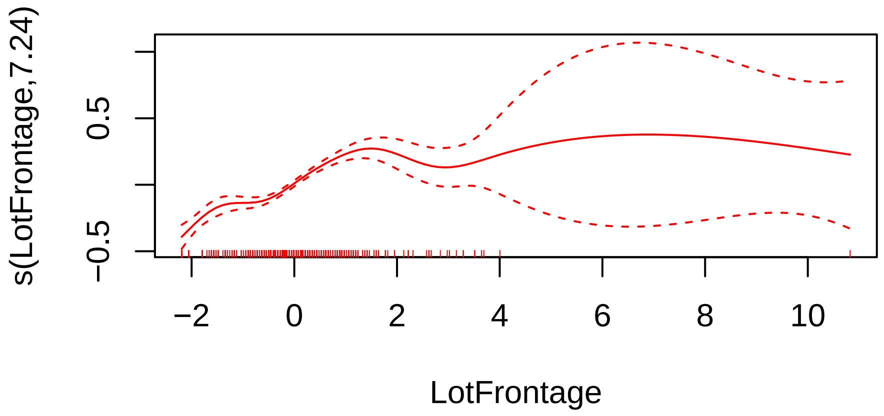
We have chosen GAM as one of our models because it produces an analysis on those factors that have less linear relationship with the result, for instance LotFrontage, YearRemodAdd, and MasVnrArea that are having relatively high importance but also high p-value that makes them not very linear related to SalePrice.

1) GAM1

In this model, we have LotFrontage, YearRemodAdd and MasVnrArea as predictors, with YearRemodAdd having a degree of freedom 2. We obtain the following result:

```
## [1] "Deviance of Model 1 approximately 78.40"
```

```
## [1] "Accuracy of Model 1 approximately 97.27%"
```

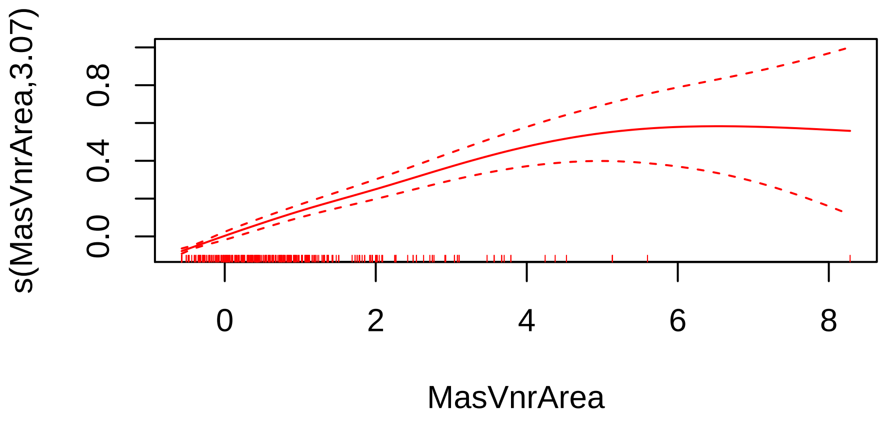


2) GAM2

In this model, we have LotFrontage, YearRemodAdd and MasVnrArea as predictors. None of them has a degree of freedom in the fit. We obtain the following result:

```
## [1] "Deviance of Model 2 approximately 83.24"
```

```
## [1] "Accuracy of Model 2 approximately 96.81%"
```

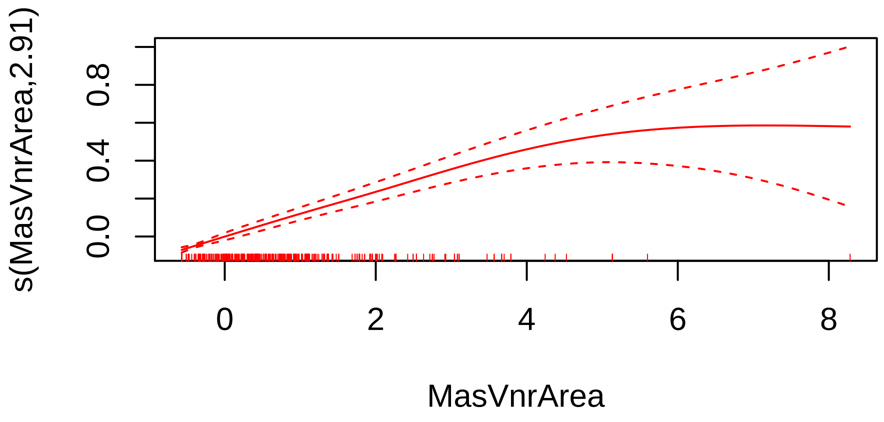


3) GAM3

In this model, we have LotFrontage, YearRemodAdd and MasVnrArea as predictorswith LotFrontage having a degree of freedom of 3. We obtain the following result:

```
## [1] "Deviance of Model 3 approximately 80.39"
```

```
## [1] "Accuracy of Model 3 approximately 97.04%"
```



GAM Summary

We then take an ANOVA test to understand which model is the best and we have the following result:

```
## Analysis of Deviance Table
##
## Model 1: SalePrice ~ s(LotFrontage) + ns(YearRemodAdd, 2) + MasVnrArea
## Model 2: SalePrice ~ LotFrontage + YearRemodAdd + s(MasVnrArea)
## Model 3: SalePrice ~ ns(LotFrontage, 3) + YearRemodAdd + s(MasVnrArea)
##   Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
## 1    1008.9     78.401
## 2    1014.2     83.243 -5.3171  -4.8422 11.729 1.394e-11 ***
## 3    1012.4     80.393  1.8127   2.8499 20.249 1.060e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that from the anova test that P-value for the second model is the smallest, therefore, it is the most preferred.

Cross Validation

Then, we conduct a cross-validation on the second model only.

R2	RMSE	MAE
0.5522	0.2539	0.1886

4. Ridge Regression

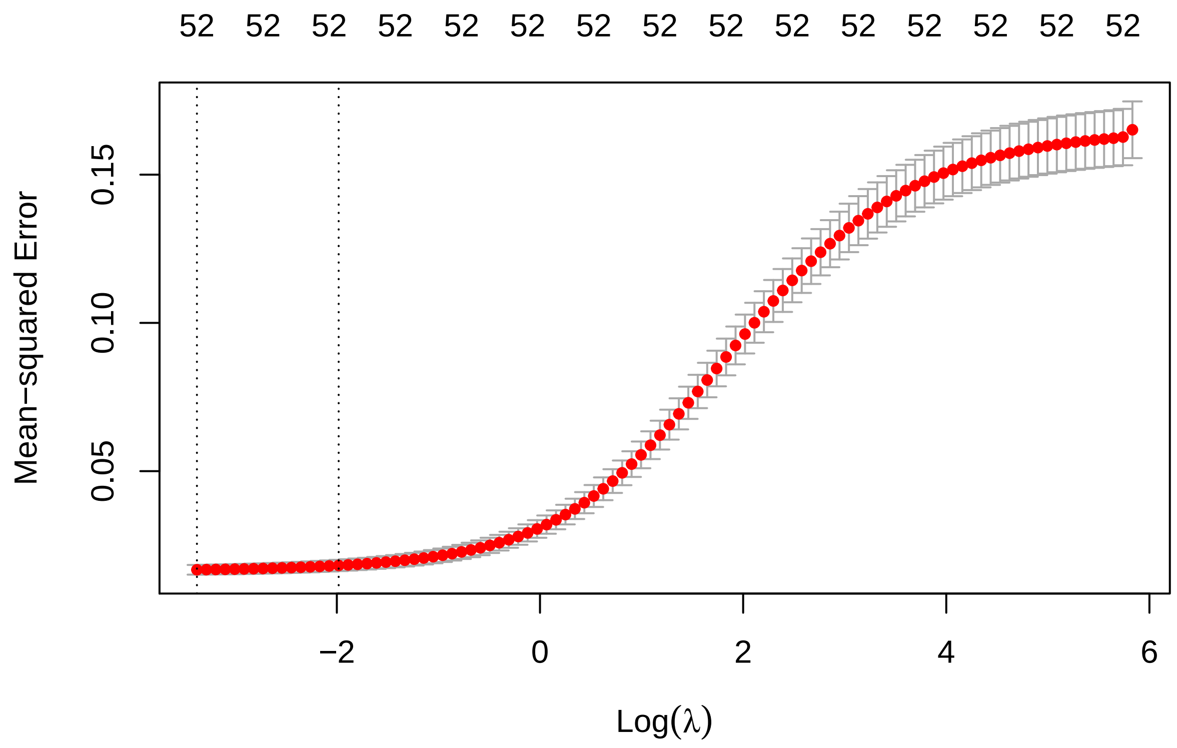
Explanation

Ridge regression is very similar to linear regression, ridge regression is also try to minimize the RSS, but it add a penalty term, try to prevent cause overfitting when add more predictors.

Prepare Model

1. Bootstrap Training Data
2. Initial Lambda
3. Optimal Lambda

```
## [1] 0.03413164
```



Check Accuarcy

```
## [1] "Accuracy of Ridge is approximately 97.04%"
```

Cross Validation

```
## [1] 46033.59
```

X1	RMSE	MAE
0.6287	196967	178673

5. Lasso Regression

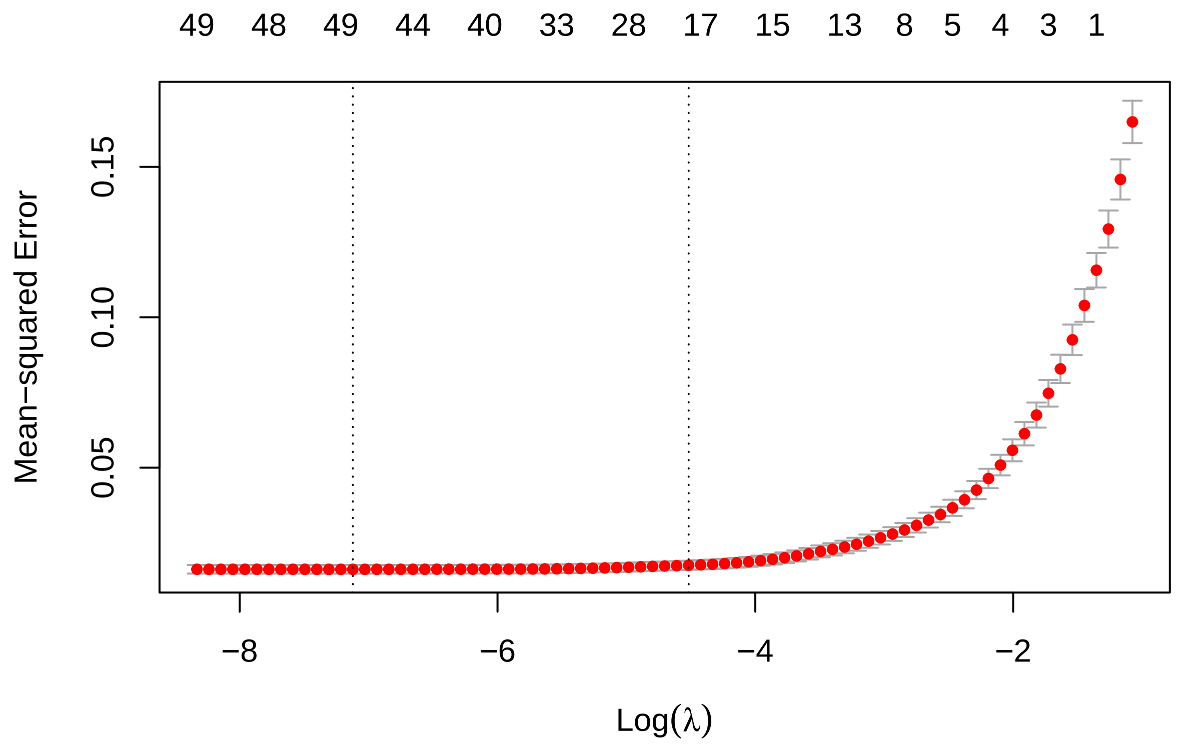
Explanation

Compare to ridge regression, lasso and ridge regression both try to minimize the RSS and have a penalty term to prevent overfitting. But Lasso regression has an advantage that lasso will make some predictors' coefficient to be 0, which could use for model selection.

Prepare Model

1. Initial Lambda
2. Optimal Lambda

```
## [1] 0.0008070391
```



Coefficient From Lasso Regression

```
##   (Intercept)     GrLivArea   OverallQual     YearBuilt   TotalBsmtSF
## 12.0374312437  0.1307855104  0.1276312075  0.0670097466  0.0547514093
##    BsmtFinSF1   OverallCond    GarageArea       LotArea    Fireplaces
##  0.0375886705  0.0305147403  0.0256908720  0.0214944049  0.0168255229
##  YearRemodAdd    GarageCars   LotFrontage    WoodDeckSF      HalfBath
##  0.0164982977  0.0144717987  0.0100360064  0.0088479262  0.0039680129
##    X3SsnPorch     AlleyNone  BsmtFullBath      FullBath  MSSubClass90
##  0.0009300888  0.0006802116  0.0003756644  0.0001123031 -0.0528705263
## MSSubClass160  MSSubClass30
## -0.0633525184 -0.0830386183
```

Check Accuarcy

```
## [1] "Accuracy of Lasso is approximately 41.46%"
```

Cross Validation

```
## [1] 45995.61
```

X1	RMSE	MAE

X1	RMSE	MAE
0.6183	196432	178743

6. Splines (Jingfeng)

Cross Validation

7. Linear Regression (Yanze)

```
##
## Call:
## lm(formula = SalePrice ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67845 -0.06336  0.00444  0.06557  0.43214
##
## Coefficients: (3 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   12.044624   0.043236 278.580  < 2e-16 ***
## LotFrontage    0.015225   0.005528   2.755 0.005988 **
## LotArea        0.022984   0.007649   3.005 0.002727 **
## OverallQual    0.114612   0.007845  14.610  < 2e-16 ***
## OverallCond    0.045989   0.005324   8.638  < 2e-16 ***
## YearBuilt      0.108952   0.011497   9.477  < 2e-16 ***
## YearRemodAdd   0.008011   0.006812   1.176 0.239863
## MasVnrArea    -0.005699   0.004924  -1.157 0.247375
## BsmtFinSF1     0.109804   0.010445  10.512  < 2e-16 ***
## BsmtFinSF2     0.029786   0.004592   6.487 1.40e-10 ***
## BsmtUnfSF      0.067866   0.008780   7.729 2.69e-14 ***
## TotalBsmtSF          NA         NA      NA       NA
## X1stFlrSF      0.081339   0.010752   7.565 8.97e-14 ***
## X2ndFlrSF      0.131644   0.013272   9.919  < 2e-16 ***
## LowQualFinSF  -0.001389   0.007537  -0.184 0.853832
## GrLivArea            NA         NA      NA       NA
## BsmtFullBath   0.009746   0.006469   1.507 0.132202
## BsmtHalfBath  -0.005072   0.004973  -1.020 0.308017
## FullBath       0.021026   0.007134   2.947 0.003282 **
## HalfBath       0.017933   0.006233   2.877 0.004100 **
## BedroomAbvGr  -0.011296   0.006795  -1.662 0.096747 .
## KitchenAbvGr  -0.001669   0.008678  -0.192 0.847573
## TotRmsAbvGrd   0.004146   0.009634   0.430 0.667077
## Fireplaces     0.018612   0.005363   3.471 0.000542 ***
## GarageYrBlt   -0.003526   0.005580  -0.632 0.527625
## GarageCars     0.021496   0.010900   1.972 0.048883 *
## GarageArea     0.015030   0.011204   1.341 0.180085
## WoodDeckSF     0.013986   0.005085   2.750 0.006062 **
## OpenPorchSF    0.006727   0.004944   1.361 0.173881
## EnclosedPorch  0.005046   0.005067   0.996 0.319545
## X3SsnPorch     0.008578   0.003102   2.765 0.005795 **
## ScreenPorch    0.015987   0.004432   3.607 0.000325 ***
## PoolArea      -0.024816   0.007592  -3.269 0.001118 **
## MiscVal       -0.003112   0.013928  -0.223 0.823227
## MoSold        -0.006233   0.003956  -1.575 0.115492
## YrSold        -0.008784   0.004183  -2.100 0.035969 *
## MSSubClass20  -0.039046   0.037669  -1.037 0.300199
## MSSubClass30  -0.085621   0.040206  -2.130 0.033459 *
## MSSubClass40  -0.037320   0.064151  -0.582 0.560873
## MSSubClass45  -0.014828   0.061835  -0.240 0.810531
## MSSubClass50  -0.009628   0.037936  -0.254 0.799707
## MSSubClass60  -0.131483   0.042114  -3.122 0.001849 **
## MSSubClass70   0.036275   0.044697   0.812 0.417232
## MSSubClass75  -0.071302   0.063907  -1.116 0.264821
## MSSubClass80   0.000160   0.042644   0.004 0.997007
## MSSubClass85  -0.007248   0.055564  -0.130 0.896244
## MSSubClass90  -0.114161   0.043619  -2.617 0.009002 **
## MSSubClass120 -0.062749   0.041708  -1.504 0.132783
## MSSubClass160 -0.200382   0.044570  -4.496 7.76e-06 ***
## MSSubClass180 -0.147953   0.061738  -2.396 0.016742 *
## MSSubClass190        NA         NA      NA       NA
## AlleyNone      0.046678   0.026691   1.749 0.080630 .
## AlleyPave      0.043421   0.034461   1.260 0.207970
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1231 on 971 degrees of freedom
## Multiple R-squared:  0.9126, Adjusted R-squared:  0.9082
## F-statistic:   207 on 49 and 971 DF,  p-value: < 2.2e-16
```

Cross Validation

8. Ensemble

Classifcation Methods

For classification method, we have separate the values of SalesPrice based on our mean, and we choose to use SVM with different kernels to make the prediction.

1. Support Vector Machine

```
##
## Call:
## svm(formula = SalePrice ~ ., data = train_cl, kernel = "linear",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.01923077
##     epsilon:  0.1
##
##
## Number of Support Vectors:  614
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1018849
##
## - Detailed performance results:
##   cost     error dispersion
## 1    1 0.1018849 0.02374549
## 2    5 0.1021264 0.02378902
## 3   10 0.1020303 0.02372300
```

```
##
## Call:
## svm(formula = SalePrice ~ ., data = train_cl, kernel = "polynomial",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##       gamma:  0.01923077
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  574
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 1.125682
##
## - Detailed performance results:
##   cost     error dispersion
## 1    1 14.454438  45.429369
## 2    5  1.353496   4.028119
## 3   10  1.125682   3.301781
```

```
##
## Call:
## svm(formula = SalePrice ~ ., data = train_cl, kernel = "radial",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.01923077
##     epsilon:  0.1
##
##
## Number of Support Vectors:  495
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.03755501
##
## - Detailed performance results:
##   cost      error  dispersion
## 1    1 0.04892331 0.007820688
## 2    5 0.03879432 0.007299333
## 3   10 0.03755501 0.007319277
```

Evaluation of different models

Root MSE

Choose best fit model

Discussion & Future Development

Resources

https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda