Kaggle: Housing Prices

Description + Data

Objective: Predict the housing price based off of the given variables and sale prices.

Data: Information is based off of residential homes in Ames, Iowa. Columns include information about:

- Expected information: lot area, heating, central air, number of bedrooms and bathrooms, etc.
- Unexpected information: roof style, fireplace quality, land slope, lot configuration and many more

Description + Data

Training Data

Total Columns: 81, including SalePrice, the dependent variable

Total Observations: 1460

Test Data

Total Columns: 80, missing SalePrice, the dependent variable

Total Observations: 1459

To test data, we split our training set into training and test subsets to work around the official test data not including SalePrice.

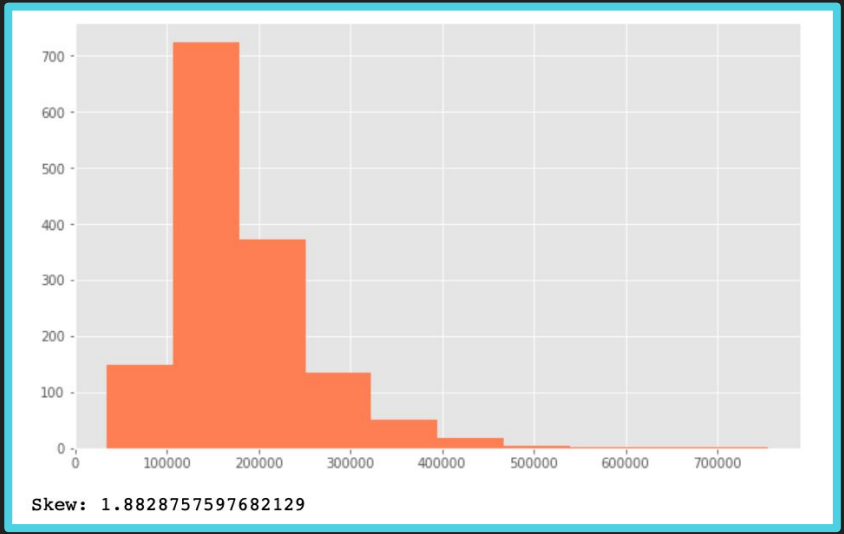
Beata’s Approach

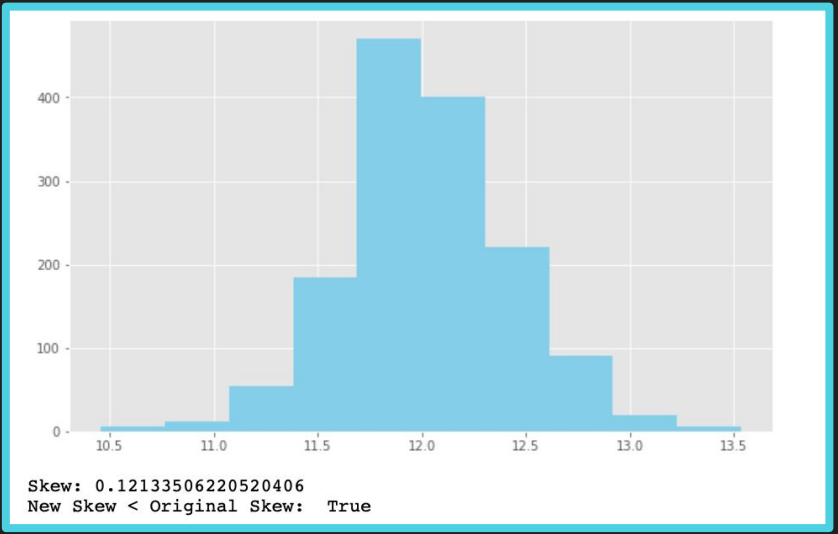
Data Cleaning

- **Null values** → changed to mean for quantitative columns; ‘None’ for qualitative columns.
- **Qualitative variables** → changed to binary/dummy
- **SalePrice** → right-skewed; used log transformation to transform to a distribution closer to Normal.

Beata's Approach

Log transformation of SalePrice





Beata's Approach

Getting Data Ready for the Model

1. Split Pandas dataframe into two: binary variables and quantitative variables
2. Split those two into training (n = 1160) and test (n = 300) and set target for training and test
3. Converted to NumPy array
4. Standardized the training and test quant variables by subtracting the test mean and dividing by the test standard deviation
5. Combined the binary and quant variables together for training data and for test data

Beata’s Approach

Plan: Use 5-fold cross-validation to determine the:

- number of layers
- drop level
- regularization level
- batch size

- **Metric:** MSE (Original Kaggle competition uses RMSE to gauge performance)
- **Loss:** MSE
- **Optimizer:** Adam

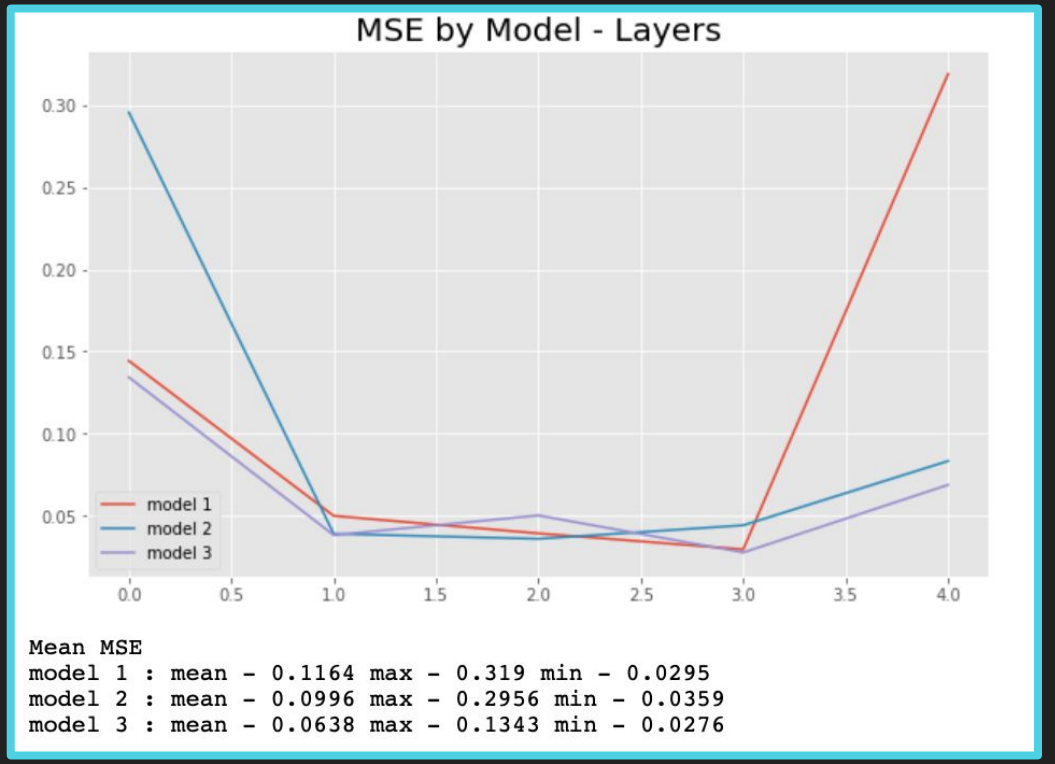
Beata's Approach

Layers: Decide how many layers to include:

Model 1 - 4 layers

Model 2 - 5 layers

→ **Model 3 - 6 layers**



Beata's Approach

Drop: determine the drop level

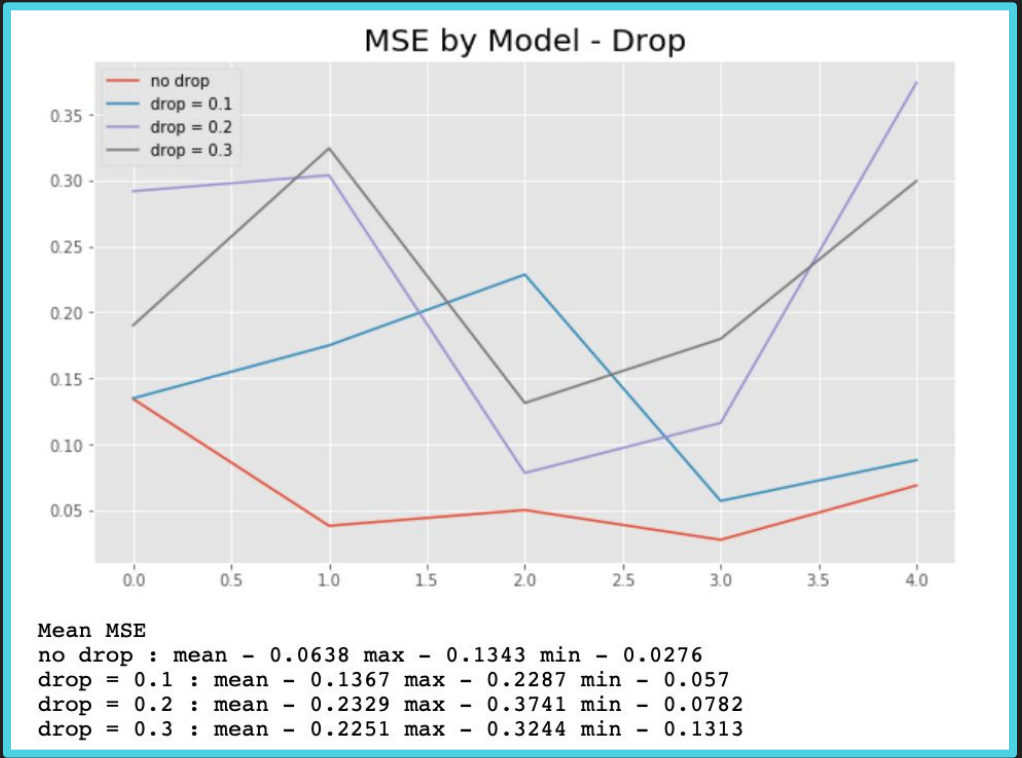
Model 1 - No Drop

→ **Model 2 - Drop = 0.1**

Model 3 - Drop = 0.2

Model 4 - Drop = 0.3

Note: second best was drop = 0.3



Beata's Approach

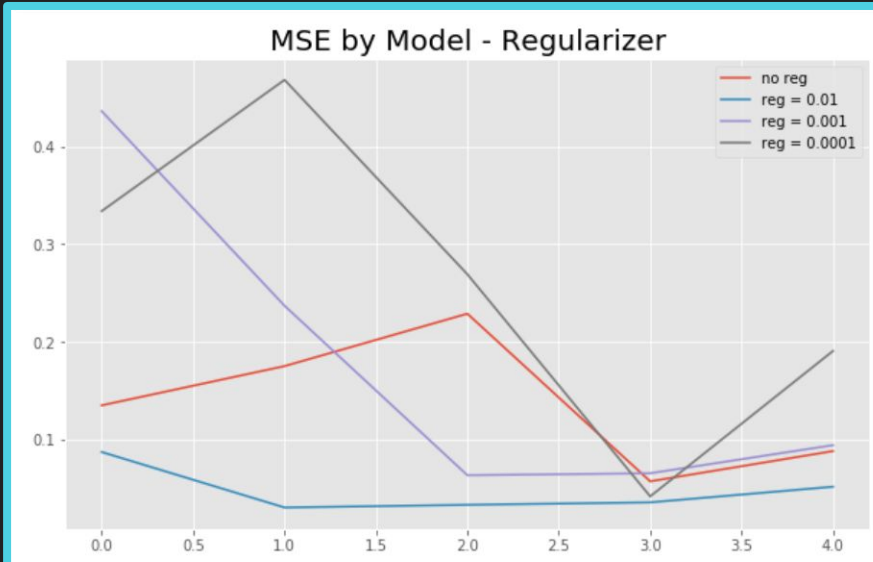
Regularizer: determine the level for an L1 regularizer:

Model 1 - No Regularizer

→ **Model 2 - Reg = 0.01**

Model 3 - Reg = 0.001

Model 4 - Reg = 0.0001



Mean MSE
no reg : mean - 0.1367 max - 0.2287 min - 0.057
reg = 0.01 : mean - 0.0474 max - 0.087 min - 0.0302
reg = 0.001 : mean - 0.1791 max - 0.436 min - 0.0633
reg = 0.0001 : mean - 0.2606 max - 0.4681 min - 0.0416

Beata's Approach

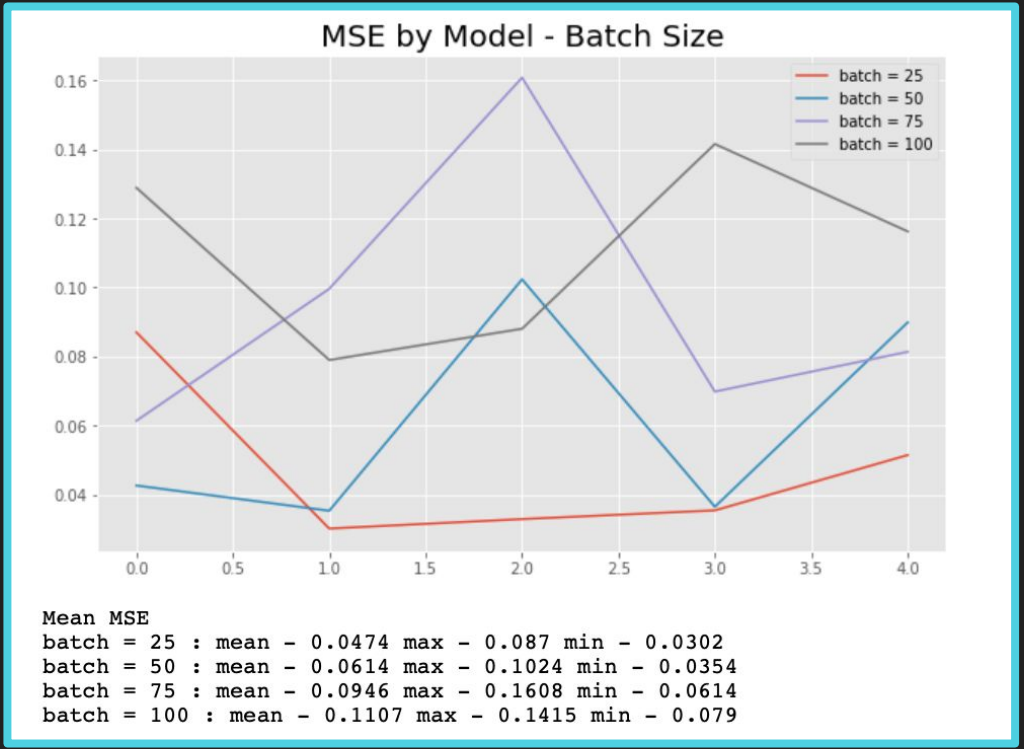
Batch Size: best batch size to use for training

→ **25**

50

75

100



Beata’s Approach

Final Model:

→ 6 layers (nodes: 256; 256; 128; 128; 64; 64)

→ L1 regularizer value of 0.01 on each layer

→ 0.1 drop-out layer

→ Optimizer: ADAM

→ Loss, Metrics: MSE

Beata's Approach

Performance on Test Set:

→ Mean Abs. Error:
- Log Value: 0.0923
- SalePrice: $16,830.02

→ MSE:
- Log: 0.0165
- SalePrice: $644,268,729.10

→ Root MSE:
- Log: 0.1284
- SalePrice: $25,382.45

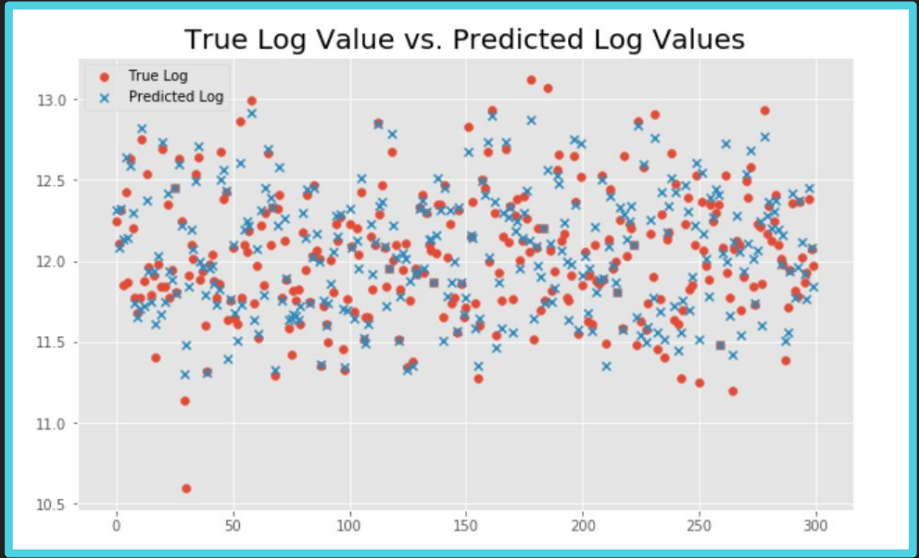
Test Set Descriptive Stats:

SalePrice:
→ Mean: $181,577.94
→ SD: $73,653.13

Log:
→ Mean: 12.0351
→ SD: 0.3825

Beata's Approach

True vs. Predicted

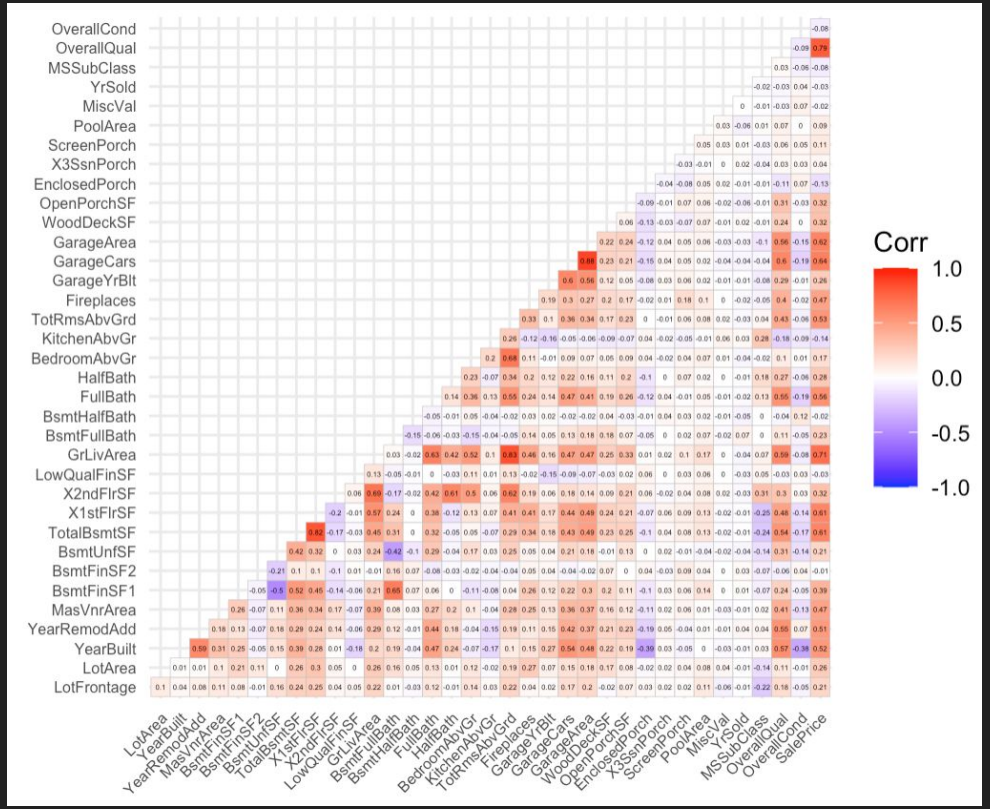


Varun’s Approach - Data Cleaning

- Null Values
  - NA values in categorical data set to new category “None”
  - No missing values for ordinal variables
- Multi-level categorical variables
  - Transformed into several binary variables in order to be fit in regression model
- SalePrice target variable
  - Log transform to make SalePrice variable normal

Varun’s Approach - Data Exploration

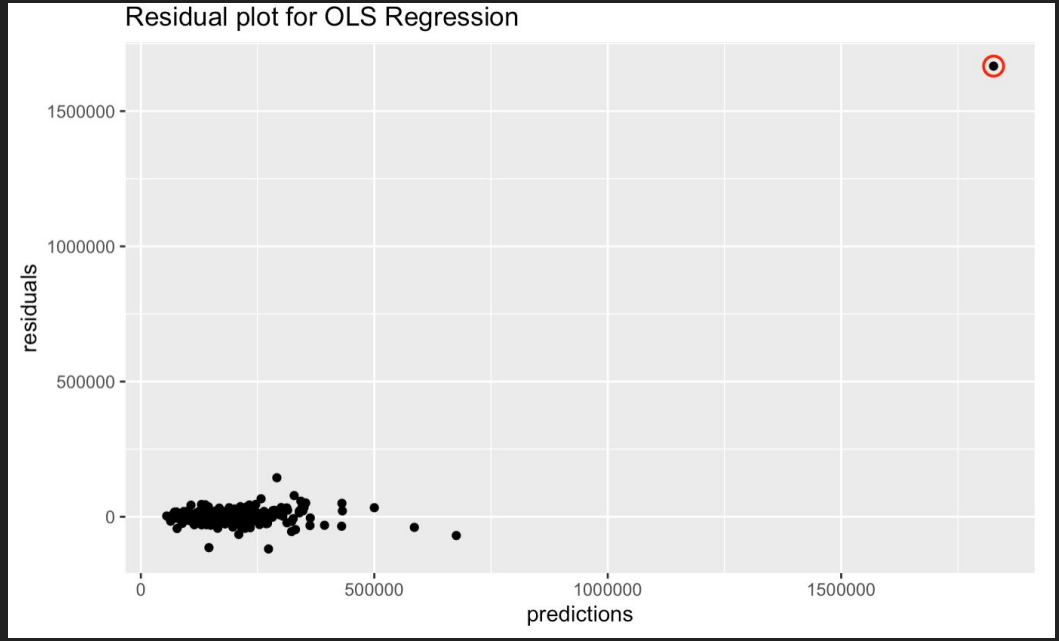
- Selected all continuous variables and the three ordinal variables (MSSubClass, OverallQual and OverallCond) to be plotted
- Target variable SalePrice at the end of the plot
- Overall Quality and total above ground square feet are the most correlated



Varun’s Approach - OLS Regression

Performance on Test Set:

- Mean absolute error
  - Log(SalePrice): 0.089
  - SalePrice: $19,285.74
- Mean squared error
  - Log(SalePrice): 0.031
  - SalePrice: $8,198,008,609
- Root Mean squared error
  - Log(SalePrice): 0.176
  - SalePrice: $90,542.85

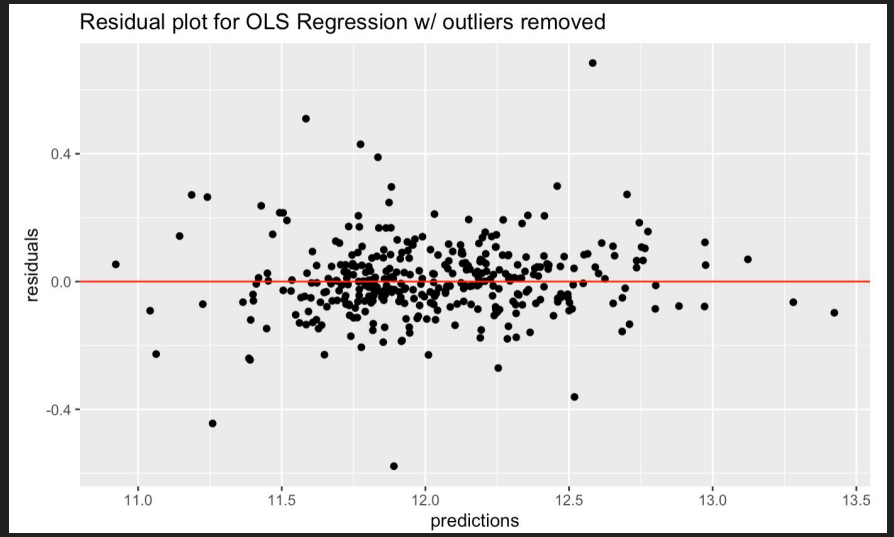


*Outlier: 6 bedroom, 4.5 bathroom 13,170 total sq. feet house sold for $160k. OLS predicts sale price to be close to 2m.

Varun’s Approach - OLS Regression

Performance on Test Set:

- Mean absolute error
  - Log(SalePrice): 0.083
  - SalePrice: $14,697.66
- Mean squared error
  - Log(SalePrice): 0.014
  - SalePrice: $485,709,563
- Root Mean squared error
  - Log(SalePrice): 0.176
  - SalePrice: $22,038.82



Varun’s Approach - Building on OLS: Ridge vs Lasso

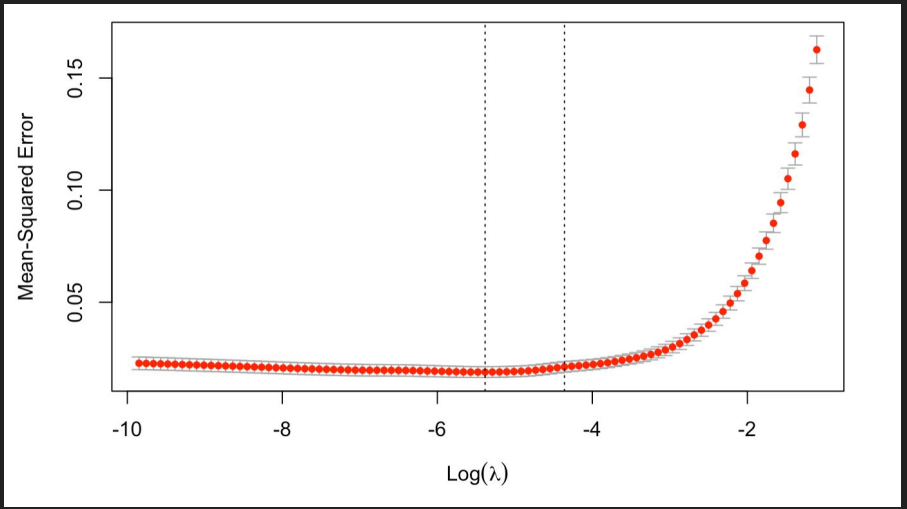
- Ridge and Lasso are shrinkage methods that apply a penalty term to reduce complexity
  - Ridge and Lasso regression apply L2(ridge) and L1(lasso) penalty to OLS
- Both have a tuning parameter lambda
- Lambda controls the shrinkage, larger lambda = greater shrinkage
- Coefficients are shrunk in ridge towards zero but will never reach zero
- If lambda is large enough in lasso, coefficients get forced to zero
- Built-in variable selection property

- After transforming categorical variables into binary/dummy variables there are 274 predictor variables
- Lasso possibly better in this case because of large amount of variables
- Non-important variables will be set to zero

Varun’s Approach - Lasso Regression

Lambda: Use 10-fold cross validation to determine value of tuning parameter lambda:

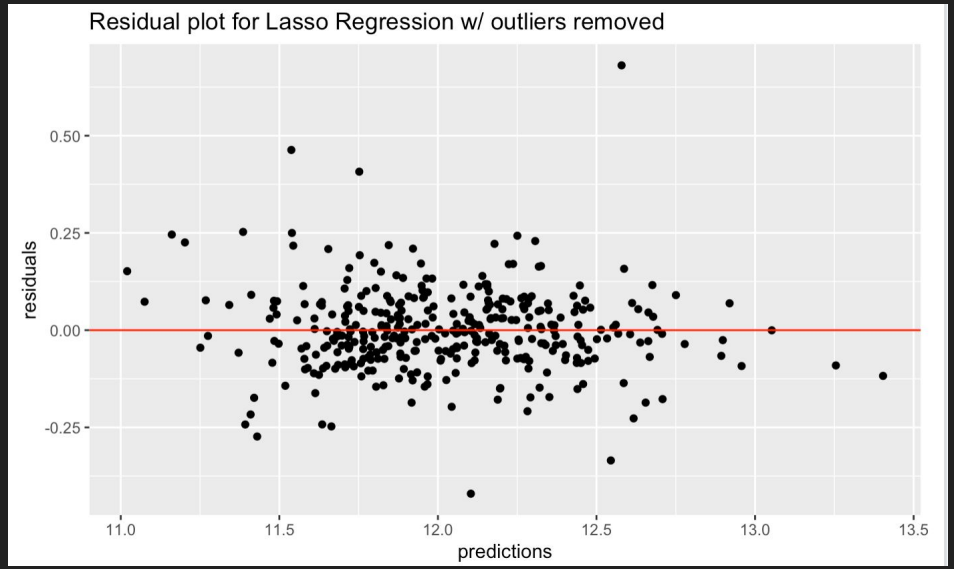
Choose lambda that gives min.
MSE: 0.0067



Varun's Approach - Lasso Regression

Performance on Test Set:

- Mean absolute error
  - Log(SalePrice): 0.077
  - SalePrice: $13,610.29
- Mean squared error
  - Log(SalePrice): 0.0116
  - SalePrice: $424,252,906
- Root Mean squared error
  - Log(SalePrice): 0.107
  - SalePrice: $20,597.40



* After cross-validation to tune lambda parameter, lasso set 187 predictors to zero, only kept 87 in the final model

Varun’s Approach - Conclusion

- Lasso worked off of OLS to improve it further
- Prevented over-fitting
- Lowered model complexity a lot

- Lasso regression is my final model/solution to this kaggle problem, predicts house prices with a mean error of $13.5k

Overall Conclusion

- Neural network tuned with 6 layers, 0.1 dropout rate and L1 regularization value of 0.01 trained on batches of 25 + Lasso were our chosen models
- Both performed very well, $13k-16k mean error
- Error less than 1 st. dev of prices

Thank you!