Machine Learning Project

House Prices (Kaggle)

Team: Boost 5
DongHyun, Hua, James, Kevin, Zhe

Outline

- EDA
  Numerical Features
  Categorical Features
  Ordinal Features

- Pre-Processing
  Data Import/Cleaning and
  Imputation

- Baseline Model Performance

- Data Engineering
  Feature Engineering
  Feature Selection

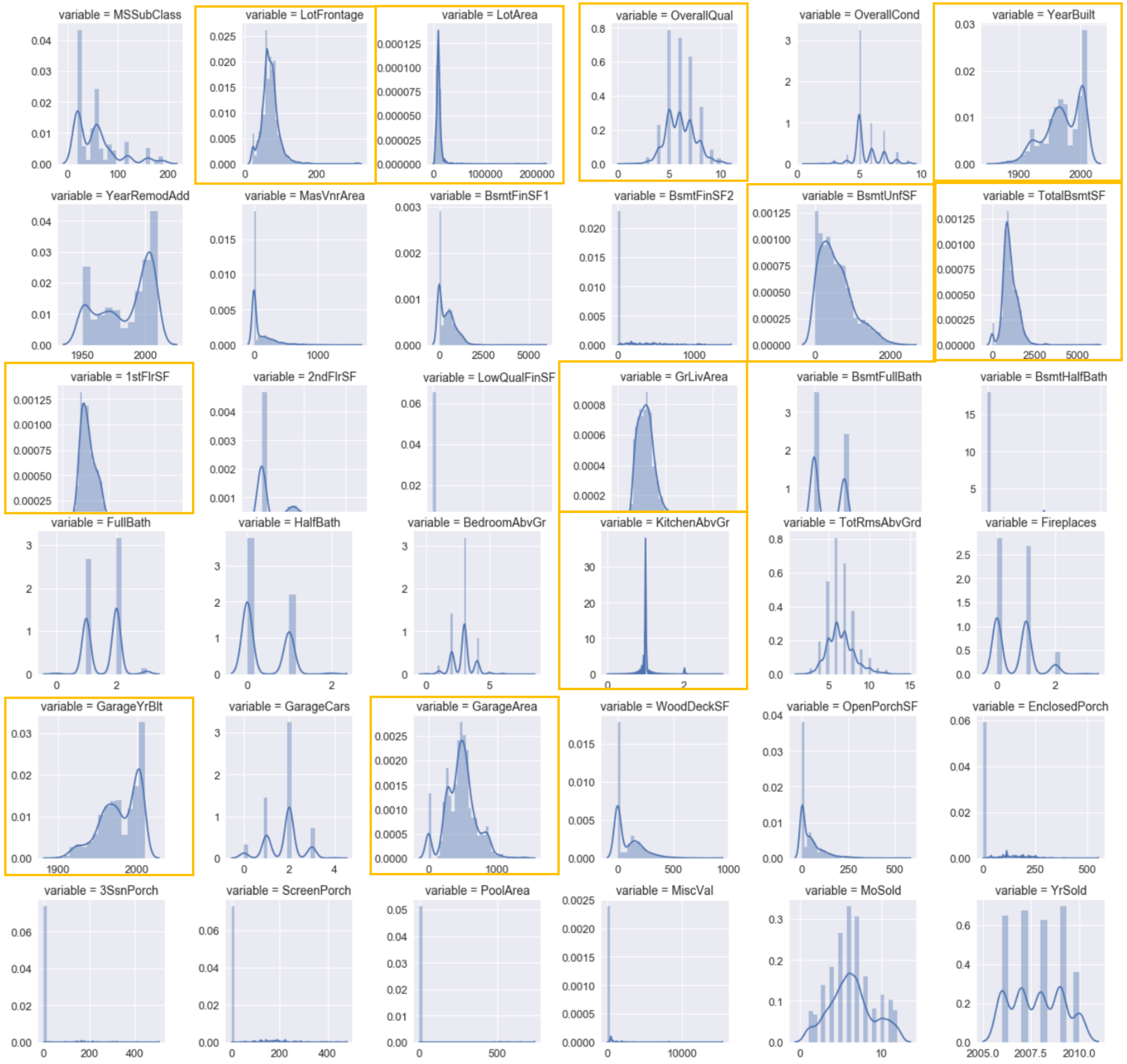
- Model Selection
  5 Model Performances

- Future Work

Exploring Data Analysis
And
Baseline Model Performance

Numerical Data Exploration

Boost 5



Numerical Data

```
quantitative = [f for f in house.columns if house.dtypes[f] != 'object']
qualitative = [f for f in house.columns if house.dtypes[f] == 'object']
```

```
f = pd.melt(house, value_vars=quantitative)
g = sns.FacetGrid(f, col='variable', col_wrap=6, sharex=False, sharey=False)
g = g.map(sns.distplot, "value")
```

Some great features for log transformation:,,,

Total Basement	OverallQual
YearBuilt	BsmtUniSF
GrLivArea	Garage Year Built
LotFrontage	LotArea
TotalBsmtSF	1st Floor
GarageArea	KitchenAbvGr

Numerical Data Correlations (Heatmap)

corr >= 0.5

1st Floor & Total Basement

Total Rooms Grd & Gr Liv Area

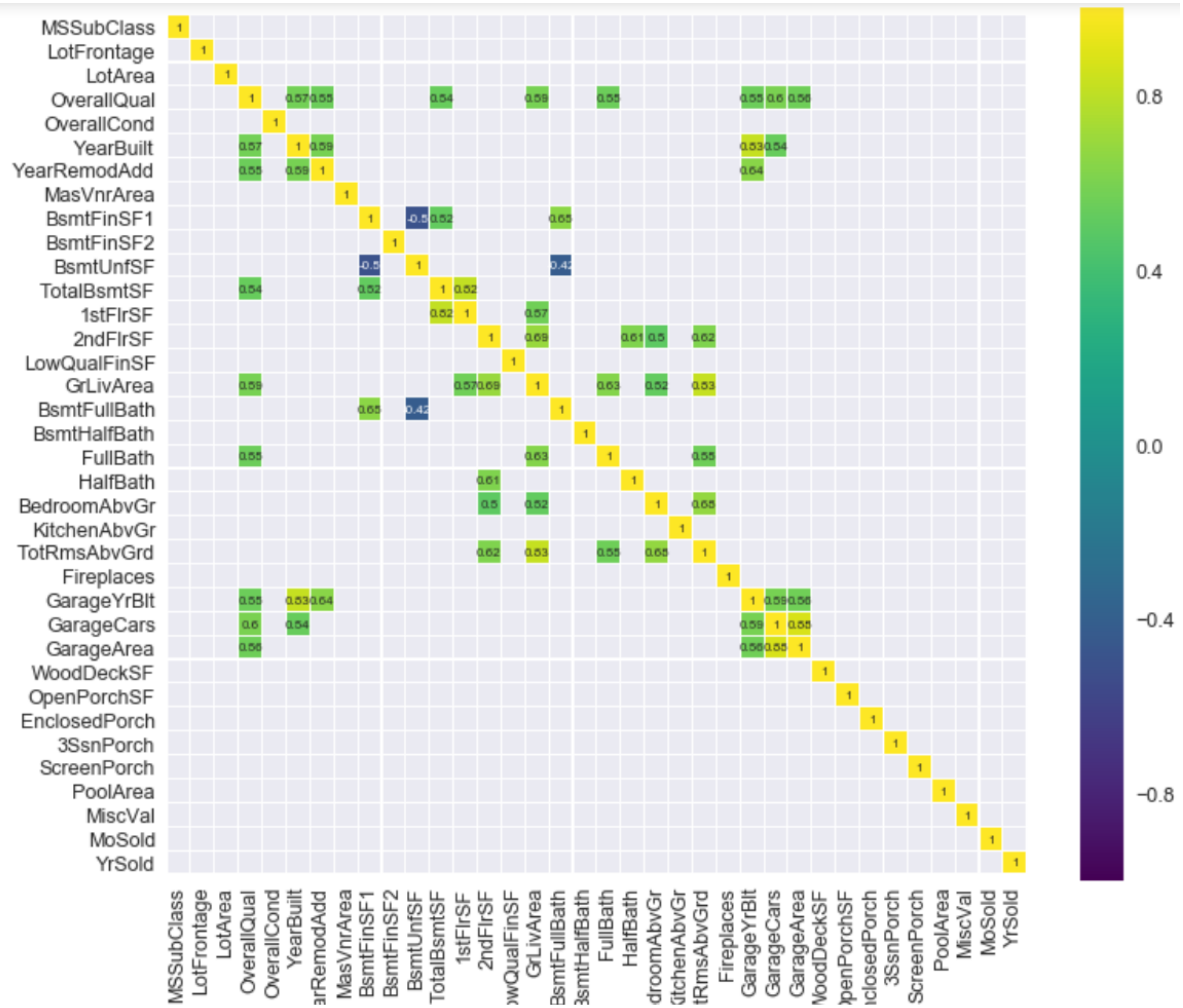
Garage Year Built & Year Built

Garage Area & Garage Cars

corr <= -0.4

Basement Full Bath

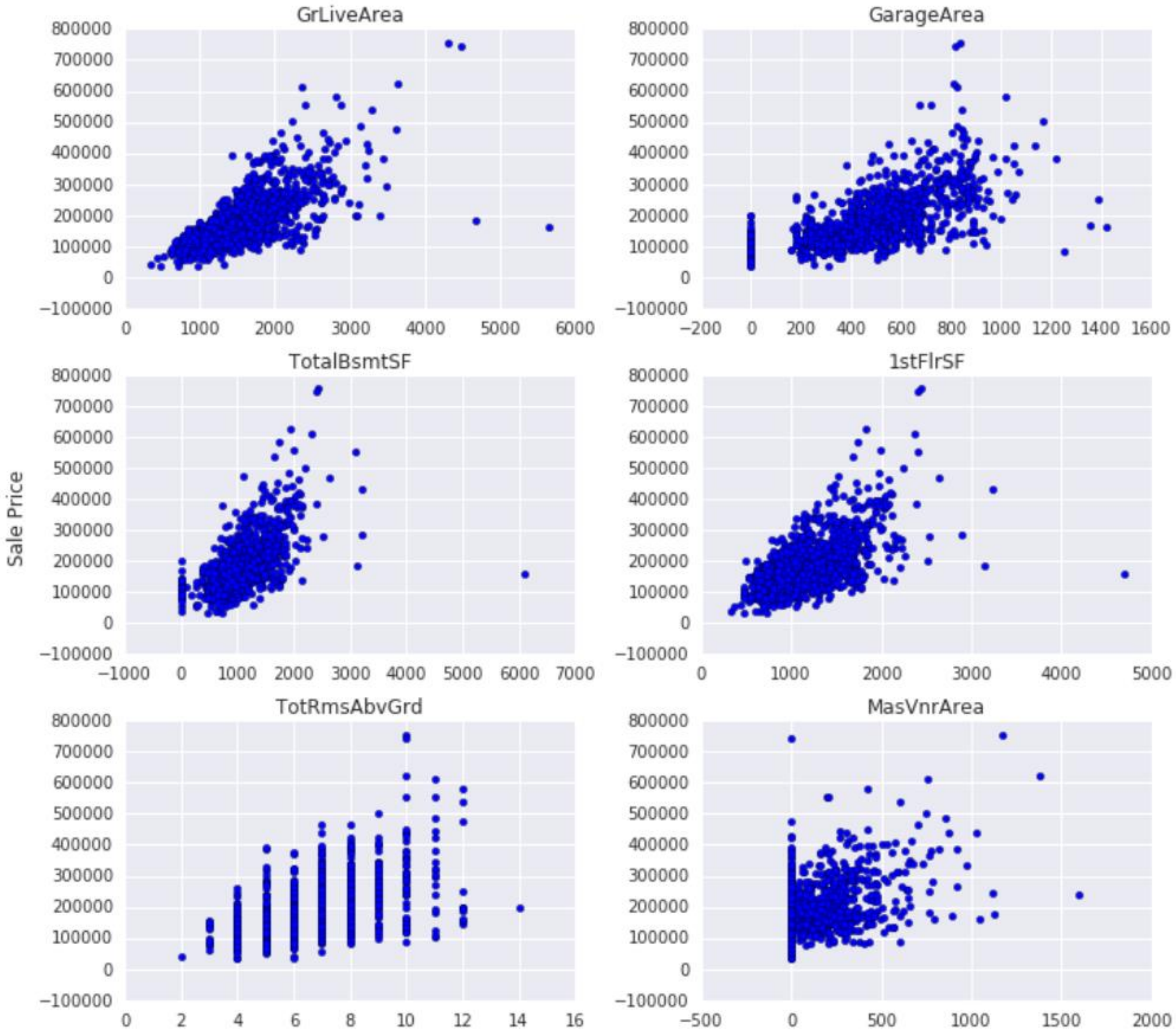
Basement Unif



Numerical Data on Sale Price

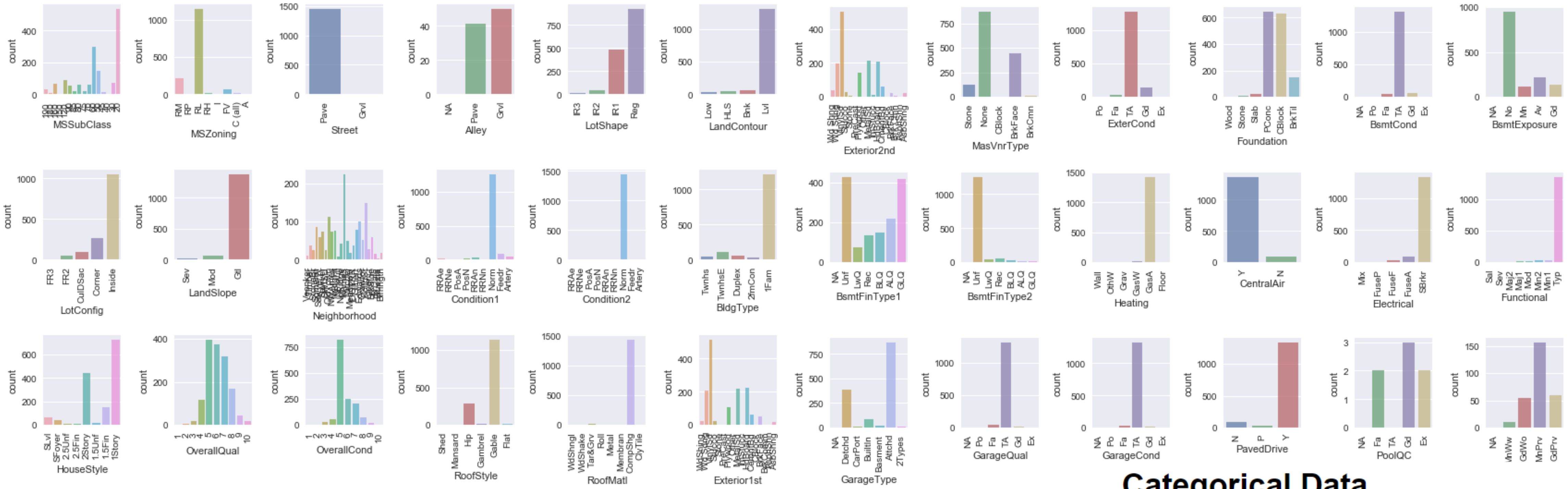
Top 6 Numerical Features by their correlation with Sale Price on scatterplot

- GrLivArea
- GarageArea
- TotalBsmtSF
- 1stFlrSF
- TotRmsAbvGrd
- MasVnrArea



Categorical Data

36 Categorical Features

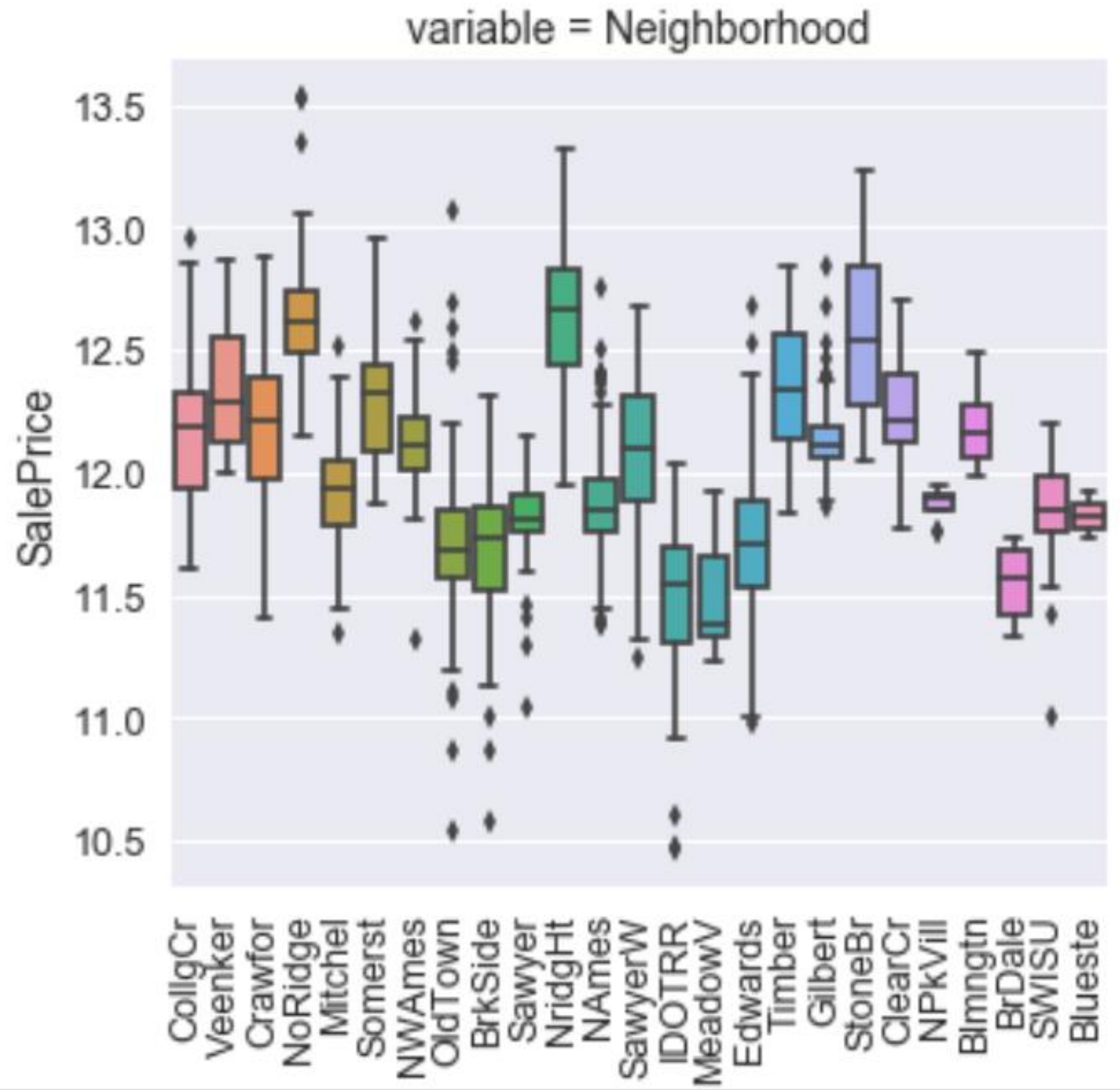


Categorical Data

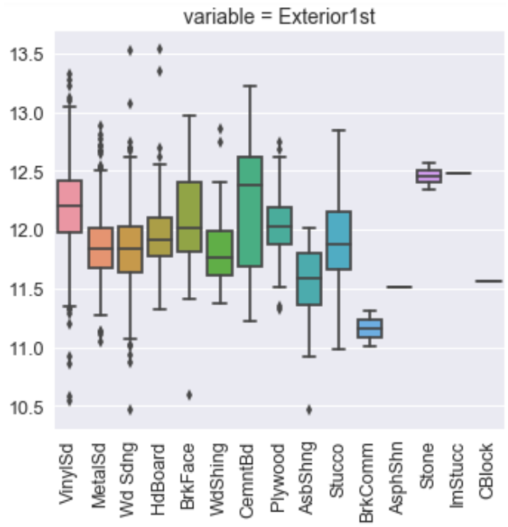
```
for c in qualitative:
    house[c] = house[c].astype('category')
    if house[c].isnull().any():
        house[c] = house[c].cat.add_categories(['MISSING'])
        house[c] = house[c].fillna('MISSING')
```

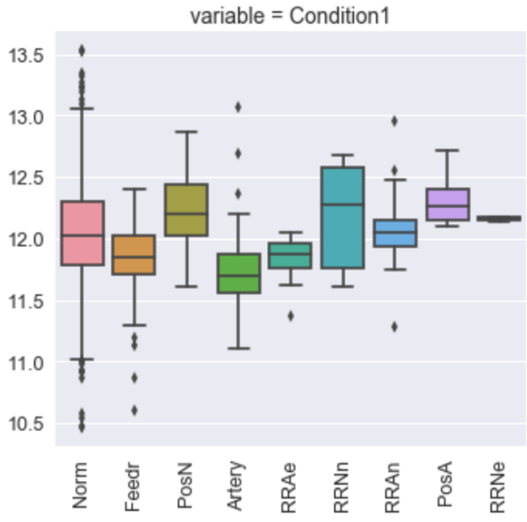
Categorical Data Correlations

Boost 5



'Neighborhood' feature has the leargest impact on SalesPrice having many different obs in a wide range of Sales price.





Exterior and Condition seem to have some strong impact as well

Ordinal Features

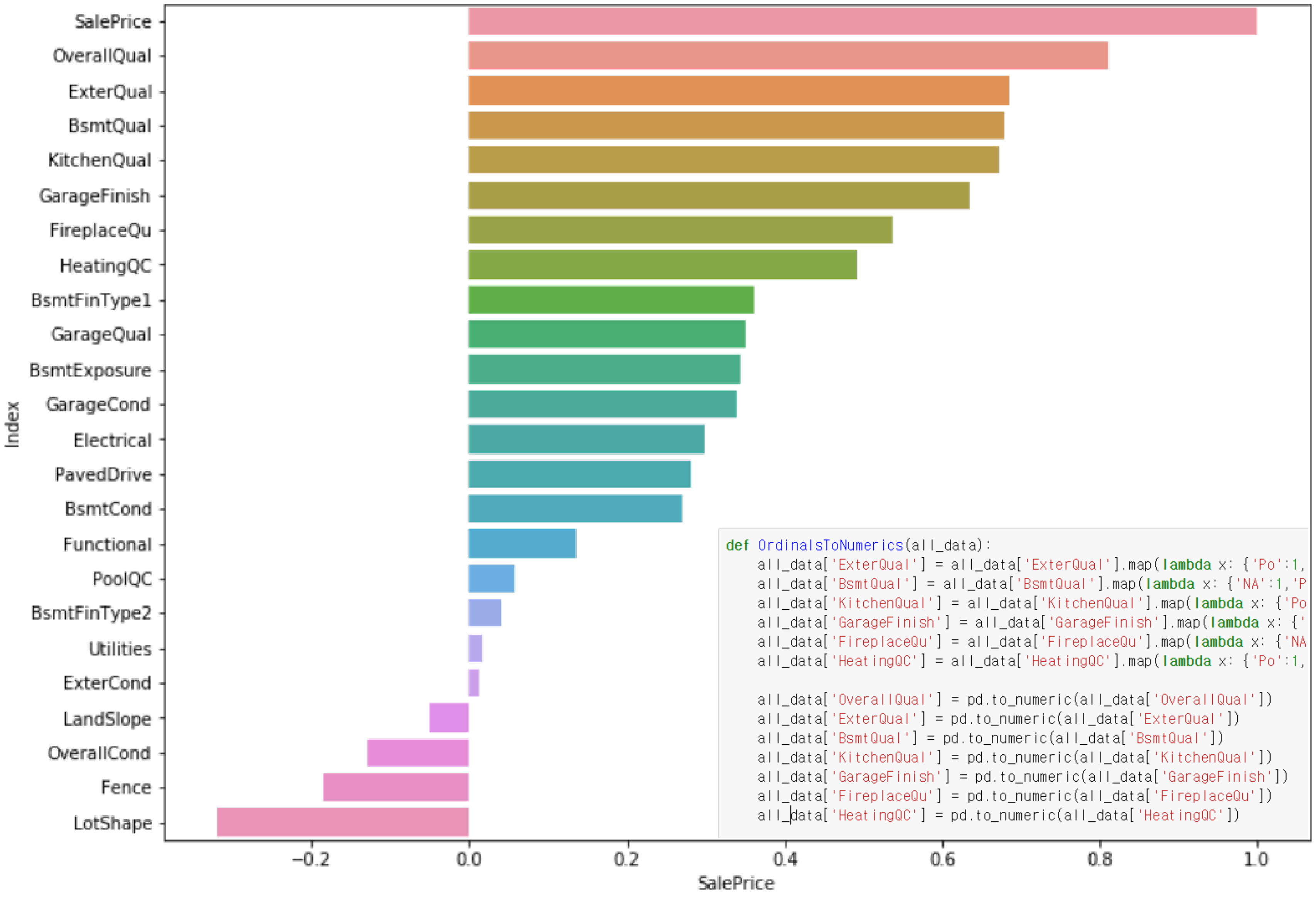
Converted categorical data with non-numerical features with rank into ordinal categories

Seems there are at least 5 house conditions from ordinal features having impact on SalePrice

- Overall Quality
- Exterior Quality
- Basement Quality
- Kitchen Quality
- Garage Finish

And the least correlated…

LotShape, Fence, OverallCond, etc




```
def OrdinalsToNumerics(all_data):
    all_data['ExterQual'] = all_data['ExterQual'].map(lambda x: {'Po':1,
    all_data['BsmtQual'] = all_data['BsmtQual'].map(lambda x: {'NA':1,'P
    all_data['KitchenQual'] = all_data['KitchenQual'].map(lambda x: {'Po
    all_data['GarageFinish'] = all_data['GarageFinish'].map(lambda x: {'
    all_data['FireplaceQu'] = all_data['FireplaceQu'].map(lambda x: {'NA
    all_data['HeatingQC'] = all_data['HeatingQC'].map(lambda x: {'Po':1,

    all_data['OverallQual'] = pd.to_numeric(all_data['OverallQual'])
    all_data['ExterQual'] = pd.to_numeric(all_data['ExterQual'])
    all_data['BsmtQual'] = pd.to_numeric(all_data['BsmtQual'])
    all_data['KitchenQual'] = pd.to_numeric(all_data['KitchenQual'])
    all_data['GarageFinish'] = pd.to_numeric(all_data['GarageFinish'])
    all_data['FireplaceQu'] = pd.to_numeric(all_data['FireplaceQu'])
    all_data['HeatingQC'] = pd.to_numeric(all_data['HeatingQC'])
```

Pre-processing

Data Import and Cle

Missing Values

Imported all data without filtering NA and transfer 'fake' NA into something else

Train/Test dataset Conversion

1. Combined test.csv with train.csv and used data documentation to extract the data type and the factor levels
2. then converted object into nominal & ordinal categorical data.
3. Chose top 7 ordinal features with high correlation with sale price, then converted them into numeric data.

Data Imputatio

KNN

used the KNN to impute the numeric data and for the categorical data.

Dummification

After imputation, we used 'get dummy' function to transfer categorical variable into dummy variables.

Baseline Model

Baseline Model

Baseline Model - Algorith

Multi Linear Models

Lasso, Ridge and Elastic Net to train the linear model.

Tree based Models

GBM, Random Forest and XGboost to fit the tree-model. Then we put random noise in data frame in order to use Random Forest to find the feature importance of each feature.

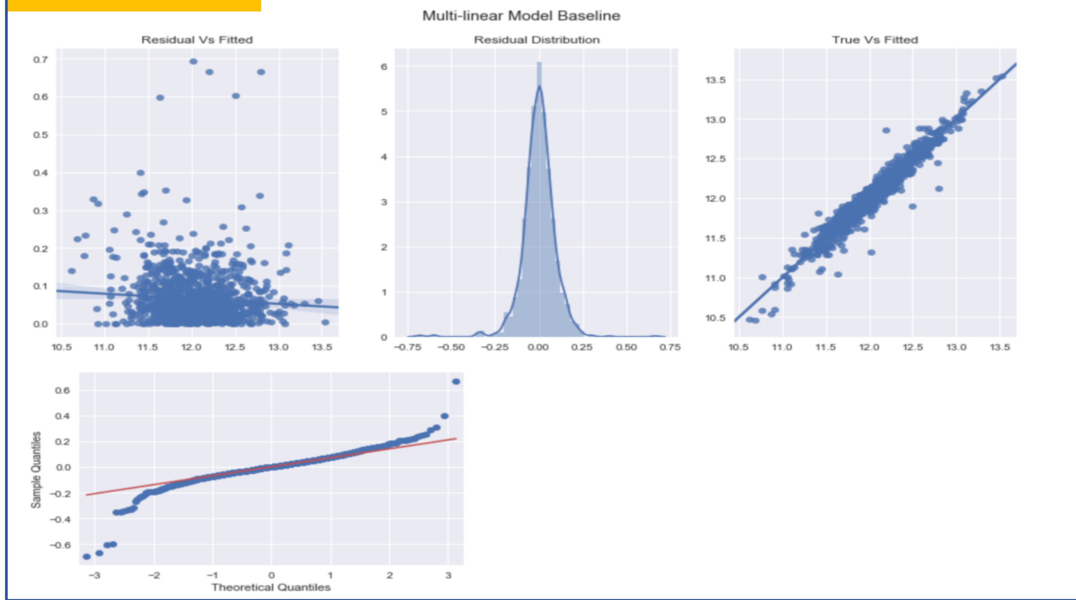
Baseline Model Performance

Boost 5

OL

```
LM_Baseline score: 0.94542
Train RMSE: 0.09436
Test RMSE: 0.13085
```

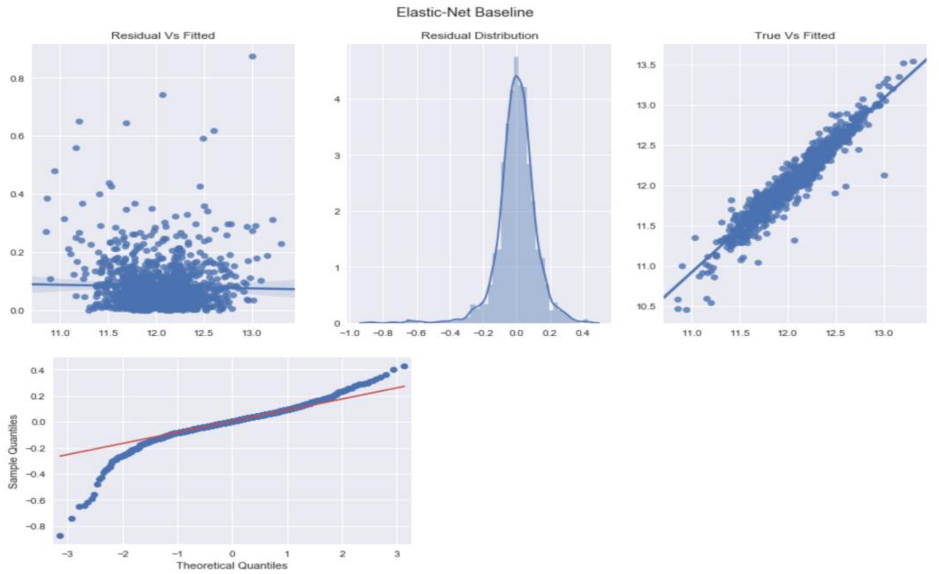
Residual Plot



Elastic Net

```
EN_Baseline score: 0.91527
Train Sqrt MSE: 0.11757
Test Sqrt MSE: 0.10059
```

Residual Plot



↳ *The fitted values don’ t have constant variance.*

Data Engineering

Feature Engineering

1. Added total sqfootage feature

2. Garage
   Replacing missing data with 0 on GarageYrBlt, GarageArea and GarageCars (Since No garage = no cars in such garage.)

3. Basement
   Missing values are likely zero for having no basement; BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath and BsmtHalfBath

4. LotFrontage
   Since the area of each street connected to the house property most likely have a similar

5. Neighborhood
   Area to other houses in its neighborhood , we can fill in missing values by the median LotFrontage of the neighborhood.

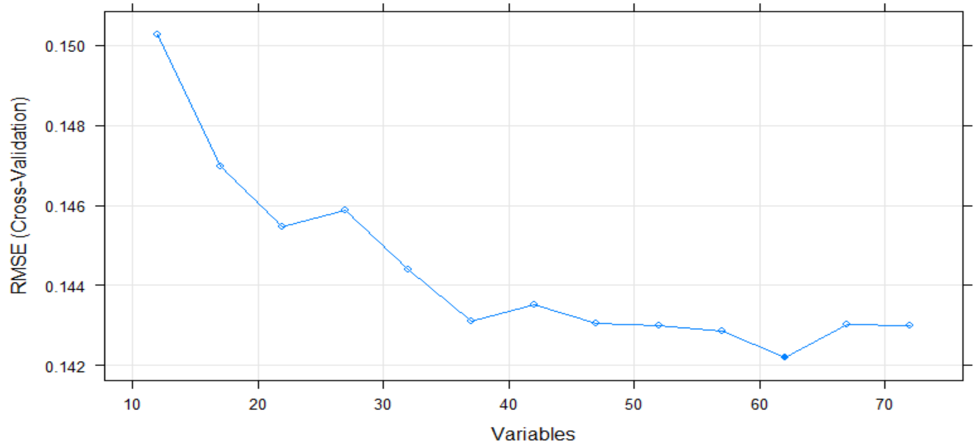
Feature Selection

Methods: Backward selection using Random Forest and Cross Validation

- R package Caret: Recursive Feature Elimination (RFE)
  - R's Random Forest package can handle categorical variables as-is if #levels < 33
- Backward selected w/ cross validation (CV)
  - remove 5 variables at a time from full set of 72 variables, and check CV RMSE, to find the best result.

RFE backward selection result

62 (out of 72) variables yields min RMSE



10 variables removed from RMSE

Alley, ExterCond, BsmtFinSF2, Heating, Electrical, BsmtHalfBath, PavedDrive, Fence, MiscFeature, MoSold

Feature Selection

Feature importance analysis (with noise variables)

- Added one [0,1] uniform and one standard normal noise variable as “new features”
- Ran Random Forest (w/ best tuned hyperparameters) and checked feature importance ranking compared w/ two noise variables
- The relative ranking of the two noise variables could vary a lot, so it’s better to add two differently distributed noise variables for the check
- Select the variables w/ feature importance ranking lower than the noise variables as additional feature removal candidates

9 more variables to remove:BsmtFinType2

LandContour	Enclosed
SaleType	Functional
LotConfig	ScreenPorch
YrSold	LandSlope
Porch	

Feature Selection

R based feature analysis, results used in Python modeling to drop features

- Tested (1) drop 10 variables identified from backward selection (2) drop additional 9 variables that has less feature importance than noise variables.
- Both dropping will improve linear regression model performance (i.e. lower RMSE), e.g. base linear regression, Ridge, Lasso, ElasticNet.
- Overall stacked final model w/ best ElasticNet + best GBM + best RF performance is improved a little.

GBM

Better result
(after re-tuning)

Random Forest

Degraded
(after re-tuning)

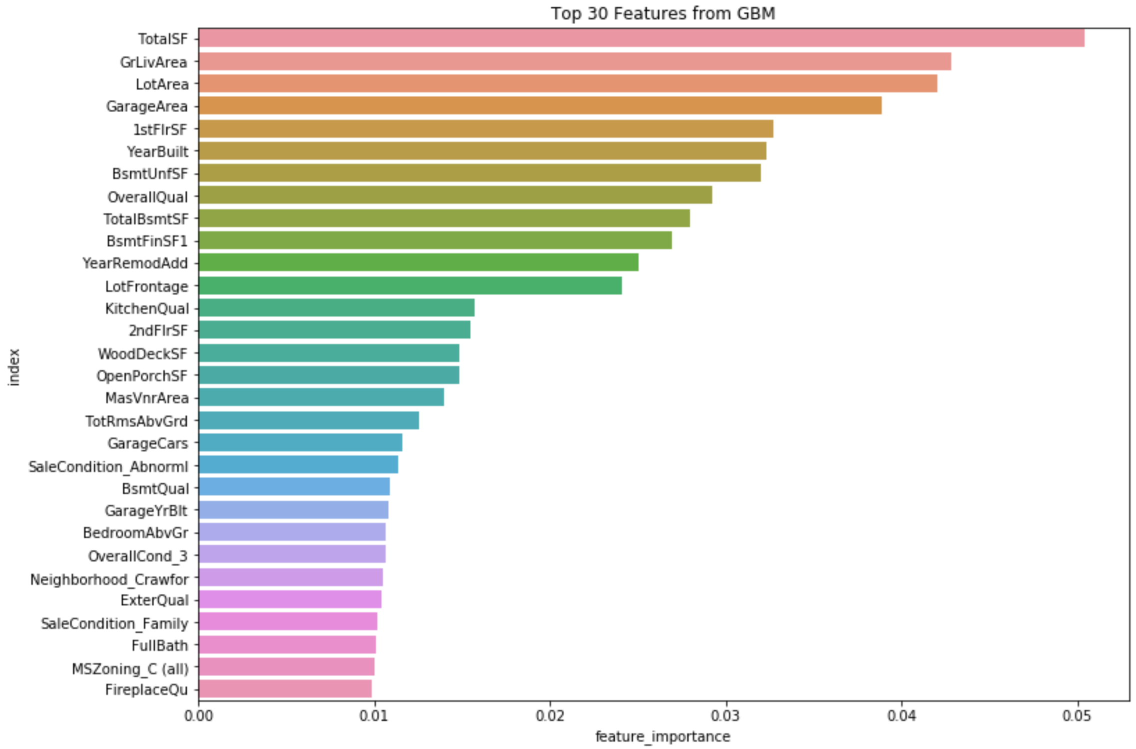
Stacked

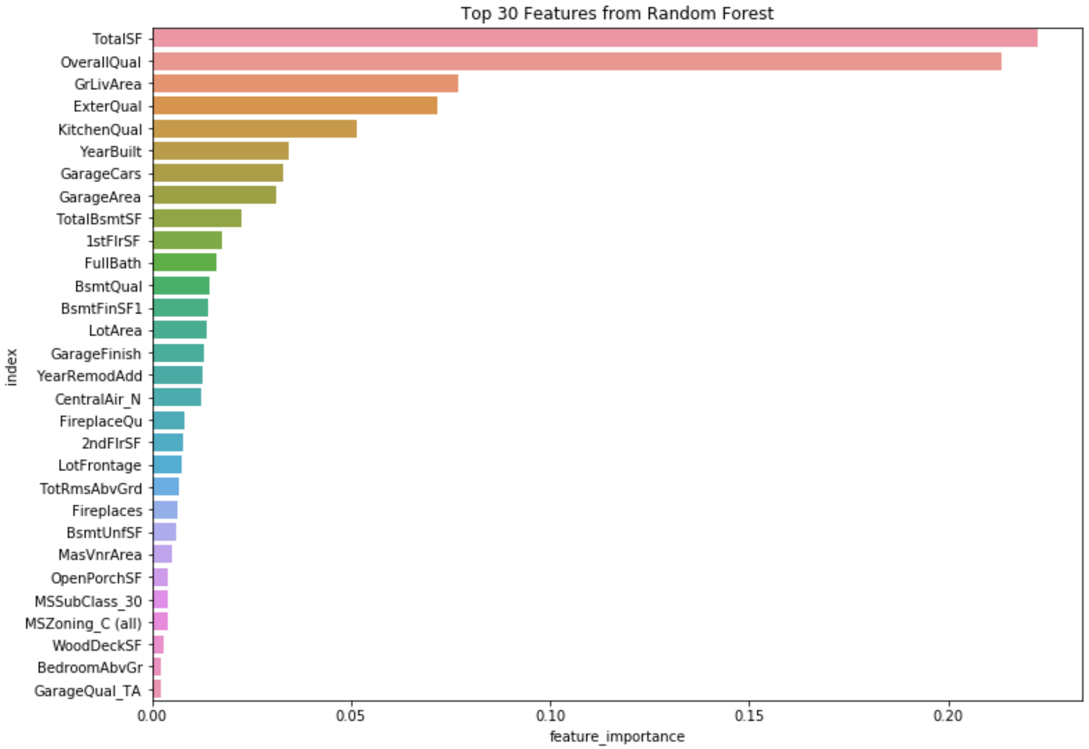
= ElasticNet
+ GBM
+ RF

Conclusion

dropping selected lowest importance features helps the overall modeling performance.

Feature importance analysis (with noise variables)





Top 30 important features from GBM and RF

Model Selectin & Stacking

Best single model selection

- Compared basic linear regression, best tunned Ridge, Lasso, ElasticNet, GBM, and RF, the **best performance is achieved by ElasticNet**.
- With best tunned ElasticNet model, **only about 32% features are finally used** (i.e. w/ non-zero model coeffiicents) from all the totally 280 feature variables (all categorical variables been converted into dummy variables).

Baseline Model Performance

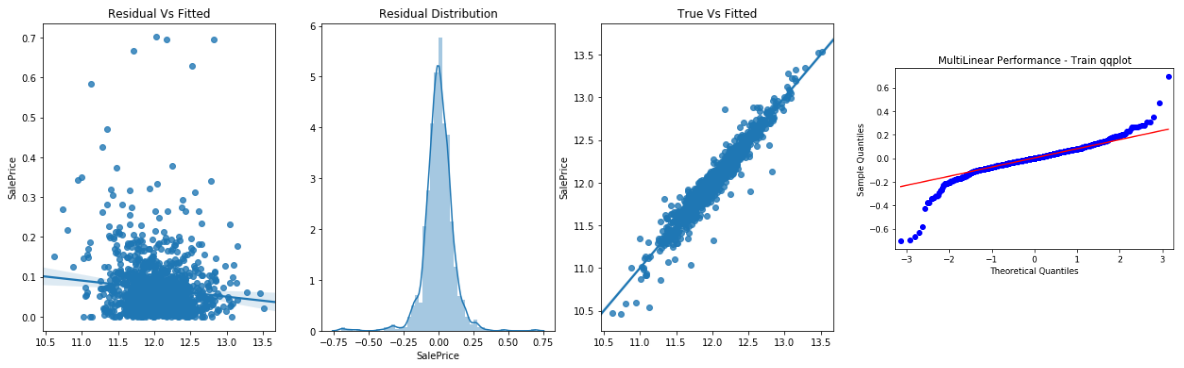
	OLS	Lasso	ElasticNet	GBM	RF
RMSE Training	0.094358	0.110304	0.117567	0.087697	0.058343
RMSE Testing	0.130852	0.097350	0.100592	0.108546	0.117223

Model Performance

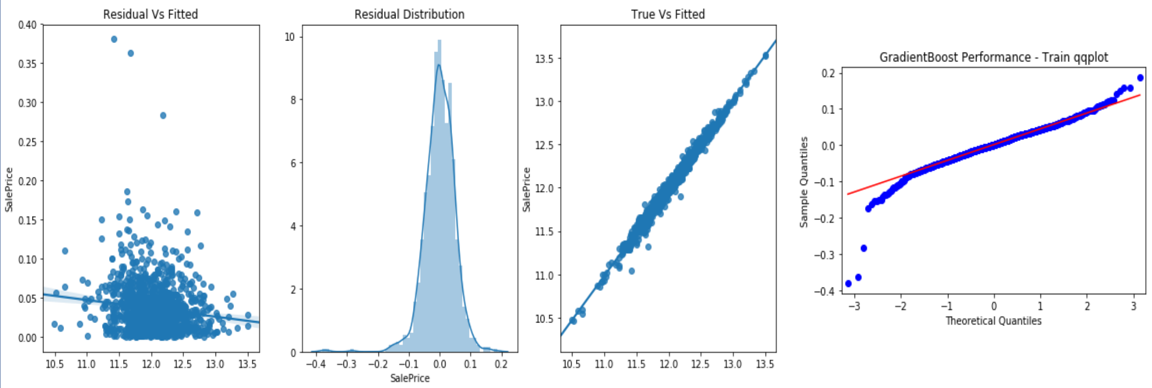
	OLS	Ridge	Lasso	ElasticNet	GBM	RF	Stacked
RMSE Training	0.1026	0.1150	0.1128	0.1128	0.0489	0.0624	0.0635
RMSE Testing	0.1273	0.0998	0.0962	0.0956	0.0976	0.1155	0.0973

Model Selection

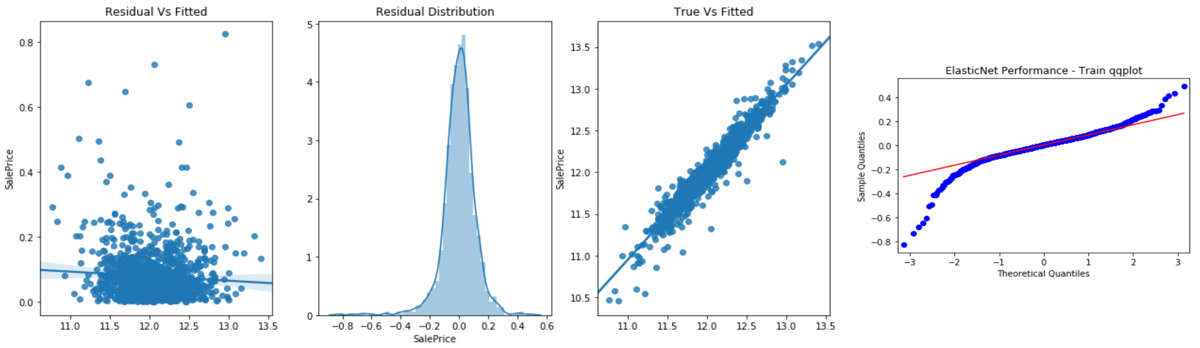
Multilinear Performance - Train



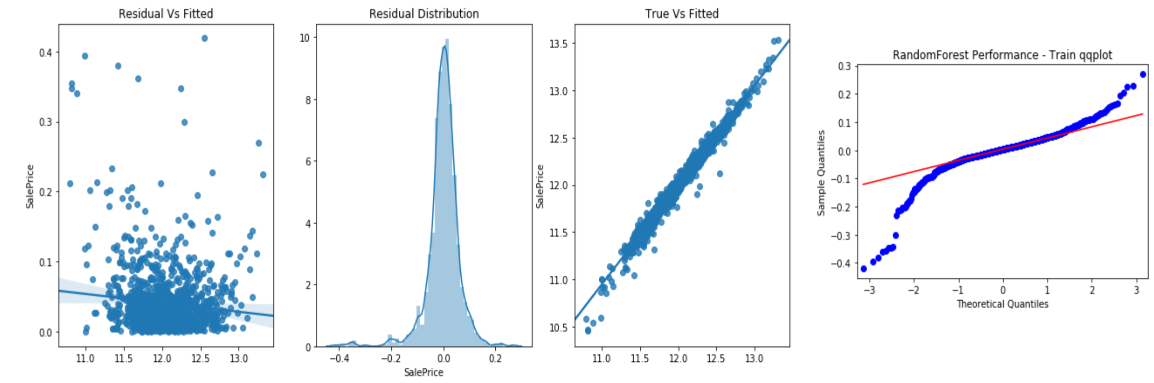
GradientBoost Performance - Train



ElasticNet Performance - Train



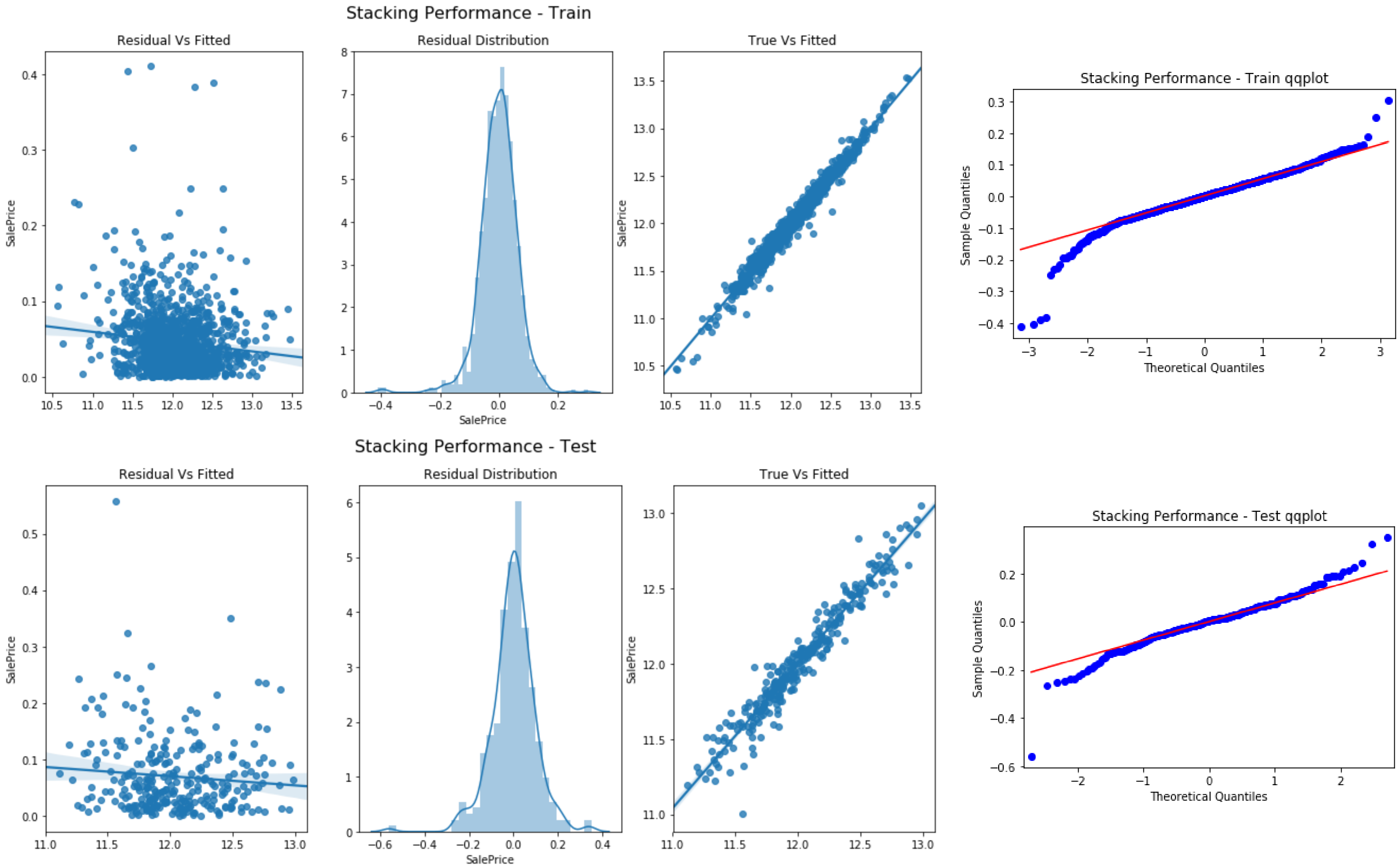
Random Forest Performance - Train



Stacked Model Performance (ElasticNet, GBM, and RF)

Trai
n

Test



Multi Model Stacking

Best multi-model stacking

- Stacked best ElasticNet, GBM, and RF models together
- Performance is a little bit lower than best ElasticNet model alone.
- Still deemed more robust than ElasticNet alone, and firstly submitted to Kaggle: score: 0.12906 (top 29%).
- Confirmed by submitting best ElasticNet model alone to Kaggle: score: **0.13252** (worse) and best GBM model alone: score: **0.13113** (worse)

Future Work

- Try other advance models
  XGBoost, SVR, etc. and tune w/ Bayes Optimizer
- No converting categorical vars to dummy
  for tree-based models (H2O RF, etc.)
- Different feature selection for different models
  i.e. only drop features for linear models, but not tree-based non-linear models
- More preprocessing choices
  BoxCox transformation, PCA, etc.
- Outlier check and removal
- Clustering analysis
  generate new useful categorical features
- Feature selection
  try other advanced algorithms e.g. Genetic algo, and simulated annealing, from R Caret package.