House Prices: Advanced Regression Techniques

Definition

Project Overview

The real estate domain encompasses buying, selling and renting of property and houses. Whenever we buy or sell a house, we often need a real estate agent to quote a price for the same since they know which factors are important and how much they affect the house's price, in other words domain knowledge. This project aims to build a model that can help the estate agent explain and the consumer understand the factors or features that affect house prices the most, and also the price itself.

This paper Modeling Home Prices Using Realtor Data is an example of predicting house prices using 12 features extracted from realtor data. In this project we try to solve a similar ongoing kaggle competition, but with a much larger feature set. The motivation of this project is feature engineering.

Problem Statement

In this project our goal is to predict the value of the target variable SalePrice for each house in the test set. Our solution takes form of a linear regression model learned over a reduced feature set, which itself is a result of feature engineering.

Metrics

Metrics based on Sum of Squared Error (SSE) is the most common metric when it comes to evaluating regression model's predictions. Since the values of the target variable is in the order of 10^5 our models are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price as specified here.

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

$$where\ n = number\ of\ observations,\ p_i = predicted\ value\ and\ a_i = actual\ value$$

Analysis

Data Exploration (Data Source)

The dataset presented here contains information about 2919 individual residential properties, henceforth houses, sold in Ames, Iowa from 2006 to 2010. Each house has **79 exploratory features** explaining, but not limited to, **sale, position, dwelling, living area, basement, amenities, garage, porch, exterior finish, lot and number of rooms, kitchen and bathrooms** of the house. Every house has an identifier attached to it, an Id variable.

The dataset is evenly divided into training and testing set containing 1460 and 1459 data points respectively. The training set contains the target variable SalePrice, which is to be predicted for each house in the test set. As inferred from above, the training set contains 1460 Rows X 81 Columns and the testing set contains 1459 Rows X 80 Columns.

On a glance there seems to be 37 numerical features and 43 categorical features however many categorical features are ordinal in nature. Additionally there are a lot of missing values in the dataset. In total there are 6965 missing values out of 116800 (approx 6%) in the training set and 7000 missing values out of 115261 (approx 6%) in the testing set spread across 34 features. There are few features, namely PoolQC, MiscFeature and Alley exhibiting more than 90 percent missing values while some, namely BsmtFinSF2, BsmtFinSF1, Exterior2nd, BsmtUnfSF, TotalBsmtSF, Exterior1st, SaleType, Electrical, KitchenQual, GarageArea and GarageCars with only one missing value.

Many features exhibit high skewness, example Utilities while some exhibit very low skewness, example OverallQual. The target variable SalePrice itself is skewed which is not surprising. Regarding outliers the author of the dataset suggest to remove all the houses with GrLivArea greater than 4000 to remove all potential outliers.

Exploratory Visualization

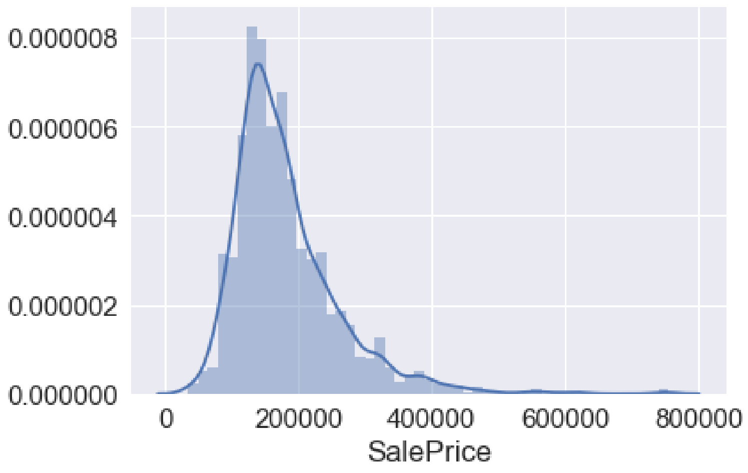

Figure 1 The figure on the right shows a distribution plot of the target variable SalePrice. As mentioned above it exhibits skewness, consisting almost all the houses in the range [34900, 400000]. We would convert this to a normal distribution using log transformation.

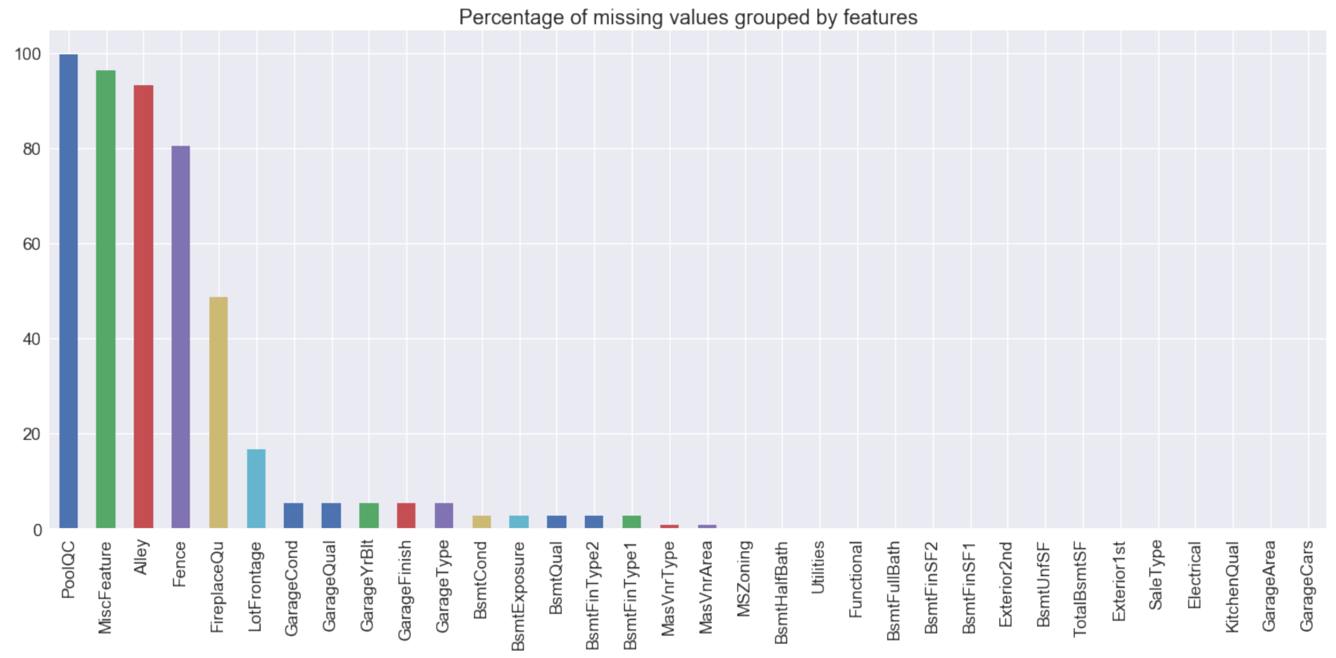

Figure 2 The above figure shows percentage of missing values per feature. Features with more than 95 percent missing values are a good candidate for feature removal. On the other hand we can remove the data points itself when there is only one missing value.

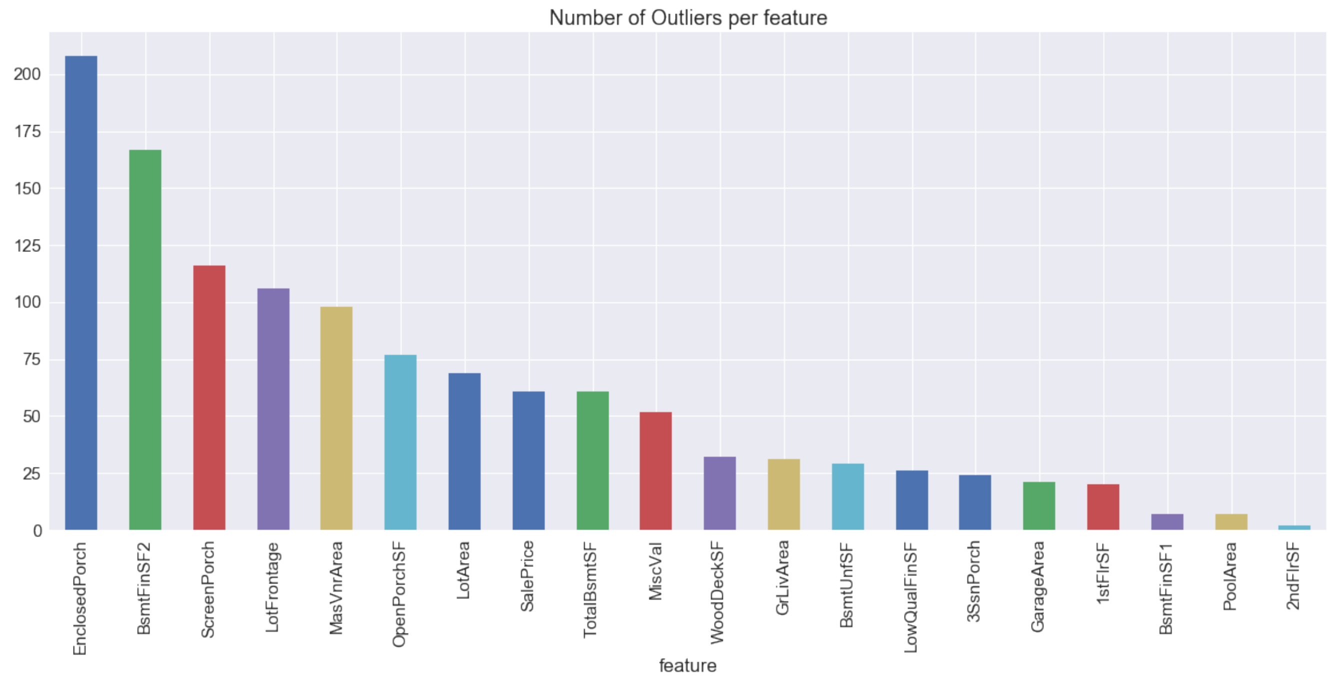

Figure 3 The above figure shows Number of Outliers per feature. This is a result of Inter-Quartile Range (IQR) on 20 continuous numerical features. Data points affecting most features can be considered outliers and thus can be removed.

Algorithms and Techniques

Major steps in this project involves missing value imputation, outlier analysis and removal, using correlation for feature reduction, log transformation of skewed features and building a linear regression model.

- For missing value imputation of a numerical feature we fill it with the mean of the training set, however when the missing value denotes absence of the said feature we fill it with 0.
- For missing value imputation of a categorical feature we fill it with the mode of the training set, however when the missing value denotes absence of the said feature we fill it with a literal "None".
- For outlier analysis we use Inter-Quartile range, since it's robust to extreme values and works over skewed features as well, on continuous numerical feature and removed outliers which affect at least 7 features.
- For measuring linear collinearity between numerical features we use Pearson correlation. An accompanied scatter matrix is used for ordinal features. However a mean and median barplot is used to measure relationship between nominal features and the target variable.
- A natural log transformation $\ln(p+1)$ is used to remove skewness since the features then become normally distributed which reduces the effect of outlier and makes the fit better.
- Finally we build three linear regression models learned over the reduced feature set as shown below and optimize the best among them using GridSearchCV.
    - LinearRegression, a linear regression model as part of sklearn.

      A linear regression is one of the simplest form of supervised learning algorithms where a linear relationship is established between the dependent variable and one or more independent variables (explanatory variables). In our case, the dependent variable is the target variable SalePrice, and the independent variables are the features in the reduced feature set. Since we would have more than one feature in the reduced feature set we would employ multiple linear regressions to solve this problem. It takes the form as shown below.

      $$y = X\beta + \varepsilon \ \ where$$
      $$y \text{ is the vector of observed values}, \ X \text{ is the feature vector},$$
      $$\beta \text{ is the vector of coefficients and } \varepsilon \text{ is the vector of errors}.$$

When expanded it takes the form as shown below.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

- GradientBoostingRegressor, an ensemble regression model also as part of sklearn.

  Gradient Boosting builds an additive model in a forward stage-wise fashion while optimizing arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

- LGBMRegressor, an ensemble regression model as part of LightGBM.

  LightGBM is similar to Gradient Boosting however in Gradient Boosting the regression tree are split depth/level wise while in LightGBM they are split leaf wise. They are also faster and have better accuracy when compared to Gradient Boosting algorithms.

Benchmark

Our benchmark model is a result of linear regression on **YrSold, MoSold, LotArea and BedroomAbvGr**. The RMSLE of this model is our benchmark score as suggested on the competition page. Below are the metrics for the benchmark model.

RMSLE Training	RMSLE Validation
0.3728	0.3126

Methodology

Data Preprocessing

This sections covers missing value imputation, outlier analysis and removal and converting ordinal feature values to numbers. Note that we did not perform log transformation and one hot encoding at this stage since a feature reduction would minimize our efforts later on since we would not be worrying about skewness of removed features.

We started off by finding features with missing values and their respective percentage. A measure of how many missing values isn't necessary to fill them but they would be desired for feature reduction as mentioned in Figure 2.

A total of 34 feature were found with missing values as below.

- PoolQC, MiscFeature, Alley, Fence, FireplaceQu, MasVnrType, Exterior2nd, GarageCond, GarageQual, GarageYrBlt, GarageFinish, GarageType, BsmtCond, BsmtExposure, BsmtQual, BsmtFinType2 and BsmtFinType1. Missing values for these feature were filled with literal "None",
- BsmtHalfBath, BsmtFullBath, BsmtFinSF2, BsmtFinSF1, BsmtUnfSF, TotalBsmtSF, MasVnrArea, GarageArea and GarageCars. Missing values for these feature are filled with 0.
- LotFrontage. Missing values for the same is filled with mean of the training set.
- MSZoning, Utilities, Exterior1st, SaleType, Electrical and KitchenQual. Missing values for these features are filled with mode of the respective feature in the training set.
- Functional. Missing value for the same is filled with "Typ" as suggested by the data description.

Once the missing values were filled we moved onto outlier analysis using IQR and removal of the same. An IQR performed on the below 20 continuous numerical feature presented with many outlier for the same as depicted in Figure 3.

- LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal and SalePrice

Among the outliers for all the features, houses with Id **198, 524, 692, 1183 and 1299** were found to be affecting at least 7 features and were removed. The author's suggestion of removing houses with GrLivArea greater than 4000 was also confirmed and it was found that all the houses with the said criteria was part of the actual outliers removed. After removal of outliers the training set contained 1455 houses.

Now that the outliers were removed we moved onto mapping categorical (ordinal) features to numbers which signifies ranks. A total of 23 feature, as shown below, were mapped to ordinal numbers.

- Street, Alley, LotShape, Utilities, LandSlope, ExterQual, ExterCond, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, HeatingQC, CentralAir, KitchenQual, Functional, FireplaceQu, GarageFinish, GarageQual, GarageCond, PavedDrive, PoolQC and Fence

In general the ordered value started from 1 and incremented by 1 for each feature value. However a "None" value was mapped to 0. At this point the updated dataset is saved onto disk for further analysis.

Implementation

Our next task was to reduce the number of feature using bivariate analysis to determine correlation between them. In a linear regression model, often one of the feature in a feature pair with high correlation is removed since the relationship with the target variable can be easily explained by the other variable as shown below.

$$\langle a \mid b \rangle = corr_{a,b}\ (large) \quad \langle a \mid target \rangle = corr_{a,target} \quad \langle b \mid target \rangle = corr_{b,target}$$
$$remove\ a\ if\ corr_{b,target} > corr_{a,target}\ else\ remove\ b$$

Since we have large number of features we have divided them into smaller subsets coherent within the subset while being non-coherent inter-subsets. In this way we can remove some features in the subset while still accounting for the aggregation of the same. The features set are divided as below.

- **feature_set_sale:** MiscVal, MoSold, YrSold, SaleType and SaleCondition
- **feature_set_position:** Neighborhood, Condition1 and Condition2
- **feature_set_dwelling:** MSSubClass, MSZoning, HouseStyle and BldgType
- **feature_set_living_area:** 1stFlrSF, 2ndFlrSF, LowQualFinSF and GrLivArea
- **feature_set_basement:** BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinSF1, BsmtFinType2, BsmtFinSF2, BsmtUnfSF and TotalBsmtSF
- **feature_set_rooms_kitchen_bathrooms:** BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr, KitchenAbvGr, KitchenQual and TotRmsAbvGrd
- **feature_set_amenities:** Utilities, Heating, HeatingQC, CentralAir, Electrical, Fireplaces, FireplaceQu, PoolArea, PoolQC and MiscFeature
- **feature_set_garage:** GarageType, GarageYrBlt, GarageFinish, GarageCars, GarageArea, GarageQual and GarageCond

- **feature_set_porch:** WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch and ScreenPorch
- **feature_set_exterior_finish:** OverallQual, RoofStyle, RoofMatl, Exterior1st, Exterior2nd, ExterQual and ExterCond
- **feature_set_lot:** LotFrontage, LotArea, LotShape and LotConfig
- **feature_set_miscellaneous:** PavedDrive, Street, Alley, Fence, OverallCond, YearBuilt, YearRemodAdd, Foundation, Functional, MasVnrArea, MasVnrType, LandContour and LandSlope

21 features were chosen as candidate for feature removal because they either had very low correlation with the target variable or were correlated with another feature that explained the target variable better than itself, as shown below.

- MiscValue, YrSold, 2ndFlrSF, LowQualFinSF, BsmtCond, BsmtFinSF1, BsmtFinSF2, BsmtFinType2, BsmtUnfSF, BedroomAbvGr, BsmtHalfBath, Utilities, Fireplaces, PoolQC, MiscFeature, GarageYrBlt, GarageArea, GarageCond, 3SsnPorch, YearRemodAdd and LandSlope

After our first phase of feature reduction we were left with 58 features in the reduced feature set. Since we never captured relationship between inter-subsets we repeated the same process as above for the reduced feature set. 6 features were chosen as candidate for feature removal for the same reason as above, as shown below.

- HeatingQC, GarageQual, BsmtQual, FullBath, GarageFinish and YearBuilt

Hence we were left with 52 features in the reduced feature set, as shown below, which became our final feature set.

- MoSold, SaleType, SaleCondition, Neighborhood, Condition1, Condition2, MSSubClass, MSZoning, HouseStyle, BldgType, 1stFlrSF, GrLivArea, BsmtExposure, BsmtFinType1, TotalBsmtSF, BsmtFullBath, HalfBath, KitchenAbvGr, KitchenQual, TotRmsAbvGrd, Heating, CentralAir, Electrical, FireplaceQu, PoolArea, GarageType, GarageCars, WoodDeckSF, OpenPorchSF, EnclosedPorch, ScreenPorch, OverallQual, RoofStyle, RoofMatl, Exterior1st, Exterior2nd, ExterQual, ExterCond, LotShape, LotFrontage, LotArea, LotConfig, PavedDrive, Street, Alley, Fence, OverallCond, Foundation, Functional, MasVnrArea, MasVnrType and LandContour

Once we had our reduced feature set, the latter part of data pre-processing was applied to it making them ready for regression models, namely log transformation and one hot encoding. Every numerical feature with skewness value greater than 0.75 (22 features) was log transformed using the function defined in Section: Algorithms and Techniques.

While performing one hot encoding for categorical (nominal) features it was found that performing them separately on training and testing set resulted in unequal number of features since feature values in them need not be the same. Hence one hot encoding was applied to the combined dataset and then the training and testing set was reconstructed from the same. The number of features after one hot encoding was 190.

The training set was split into training (80%) and validation set (20%) to aid in cross validation. We built three regression model, as below, with their default parameters and measured the RMSLE for training and validation set.

- LinearRegression, a linear regression model as part of sklearn, for it's simplicity.
- GradientBoostingRegressor, an ensemble regression model also as part of sklearn, for it's improved accuracy.
- LGBMRegressor, an ensemble regression model as part of LightGBM, for it's improved accuracy and performance.

All the three models easily surpassed the benchmark model, however GradientBoostingRegressor was found to be the best among them since it had lower RMSLE on both the training and validation set among all. The same was further optimized using GridSearchCV and the predictions from this optimized model was saved onto disk completing the goal of this project.

Refinement

As mentioned above the best model GradientBoostingRegressor, was further optimized using GridSearchCV with the following hyperparameters.

- **n_estimators:** 200, 400, 800, 1600 and 3200
- **learning_rate:** 1.0, 0.5, 0.05, 0.025 and 0.001
- **max_depth:** 2, 3 and 4
- **min_samples_split:** 2 and 4
- **min_samples_leaf:** 2, 4, 8 and 16
- **max_features:** 16, 32, "sqrt"

Below are the metrics for the benchmark model, initial model (GradientBoostingRegressor) and the final solution (Optimized GradientBoostingRegressor).

	Training Time	Predicting Time	RMSLE Training	RMSLE Validation
Model				
GradientBoostingRegressor (Optimized)	1.941	0.016	0.1173	0.0991
GradientBoostingRegressor	0.691	0.002	0.1276	0.1056
Benchmark	0.001	0.001	0.3728	0.3126

As evident from the table above we saw a large decrease of 65.77% and 66.22% in RMSLE Training and RMSLE Validation respectively in the initial solution when compared to the Benchmark model. Moreover there was a slight decrease of 8.07% and 6.16% in RMSLE Training and RMSLE Validation in the final solution when compared to the initial solution as a result of GridSearchCV.

Results

Model Evaluation and Validation

The GradientBoostingRegressor with the following parameter values resulted from GridSearchCV.

- **n_estimators:** 3200 [default: 100]
    - Number of regression trees to generate. A higher value is more computationally expensive but allows for a robust model.
- **learning_rate:** 0.025 [default: 0.1]
    - The learning rate. A lower value is more computationally expensive but allows for a robust model.
- **max_depth:** 2 [default: 3]
    - The maximum depth of the regression tree. A higher value will learn complex relationship from each sample. Controls overfitting.
- **min_samples_split:** 2 [default: 2]
    - Minimum number of samples required in a node to split. A higher value will ensure that the model generalizes well. Too high a value results in underfitting.
- **min_samples_leaf:** 4 [default: 1]
    - Minimum number of samples required in the leaf node. Changes the splitting criteria to ensure generalization.
- **max_features:** 16 [default: n_features]
    - Number of features to consider while searching for a best split. Square root on n_features works great. Higher value can lead to overfitting.

The trade off for choosing higher n_estimators and lower learning_rate is a computationally expensive model but a more robust and generalized model. The parameter max_depth value's suggest that since our number training samples are small compared to the number of features the model needed to learn some complex relationship. The parameter min_samples_split is at it's default value. The parameter min_samples_leaf value's also suggest the need to learn complex relationship since higher value underfits. The parameter max_features value's is a little greater than sqrt(n_features).

Justification

	Training Time	Predicting Time	RMSLE Training	RMSLE Validation
Model				
GradientBoostingRegressor (Optimized)	1.941	0.016	0.1173	0.0991
GradientBoostingRegressor	0.691	0.002	0.1276	0.1056
LGBMRegressor	0.696	0.009	0.1348	0.1076
LinearRegression	0.037	0.001	0.1271	0.1093
Benchmark	0.001	0.001	0.3728	0.3126

As evident from the table above we saw a decrease of at least 63.84% and 65.03% in RMSLE Training and RMSLE Validation respectively in all the above model when compared to the benchmark model. Moreover as compared to the benchmark model, we saw a decrease of 68.54% and 68.30% in RMSLE Traning and RMSLE Validation in the optimized final model making it adequate for the problem at hand. Additionally the training and predicting time were well below reasonable limit.

Conclusion

Free-Form Visualization

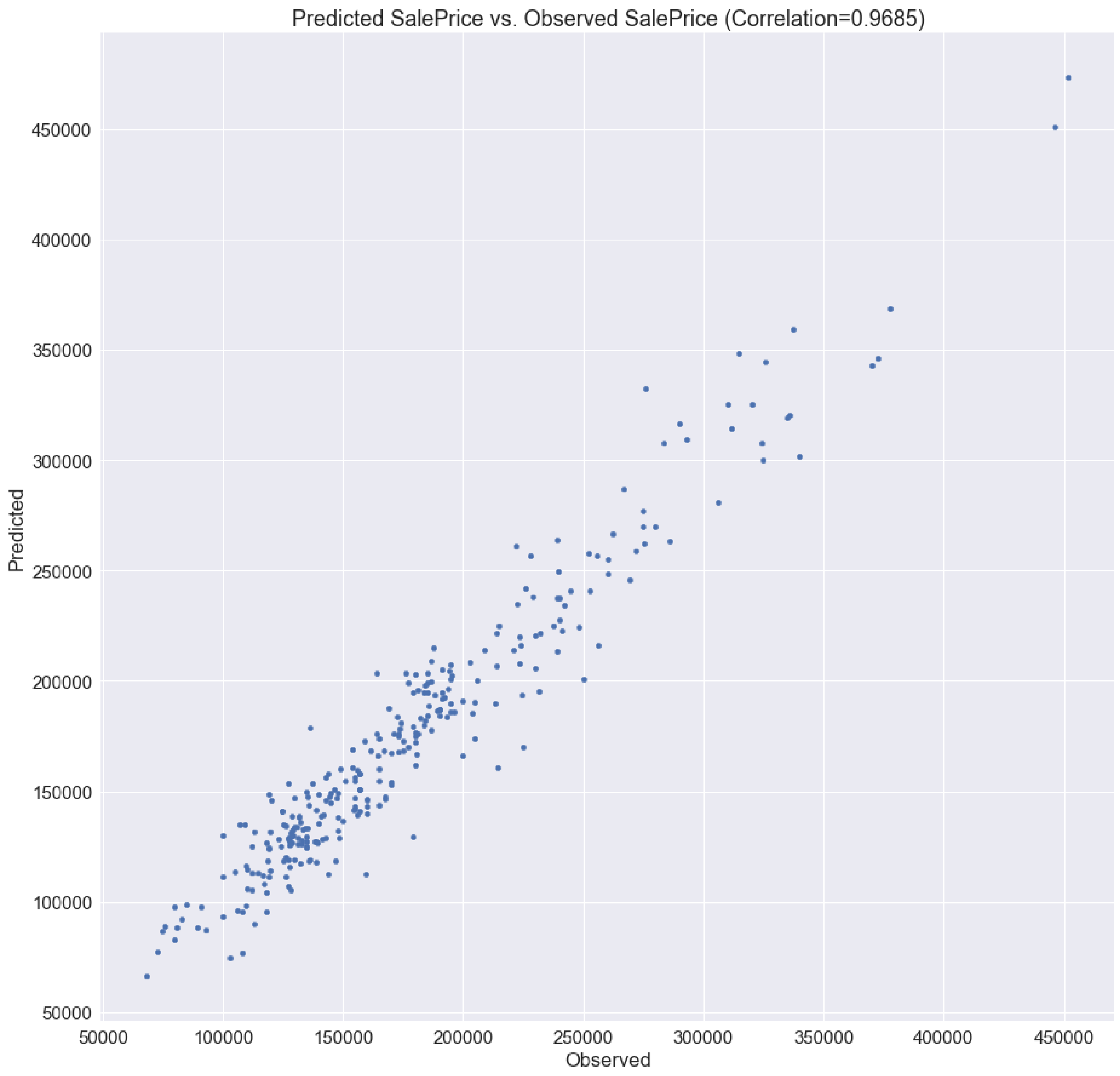

Figure 4 Predicted SalePrice vs. Observed SalePrice.

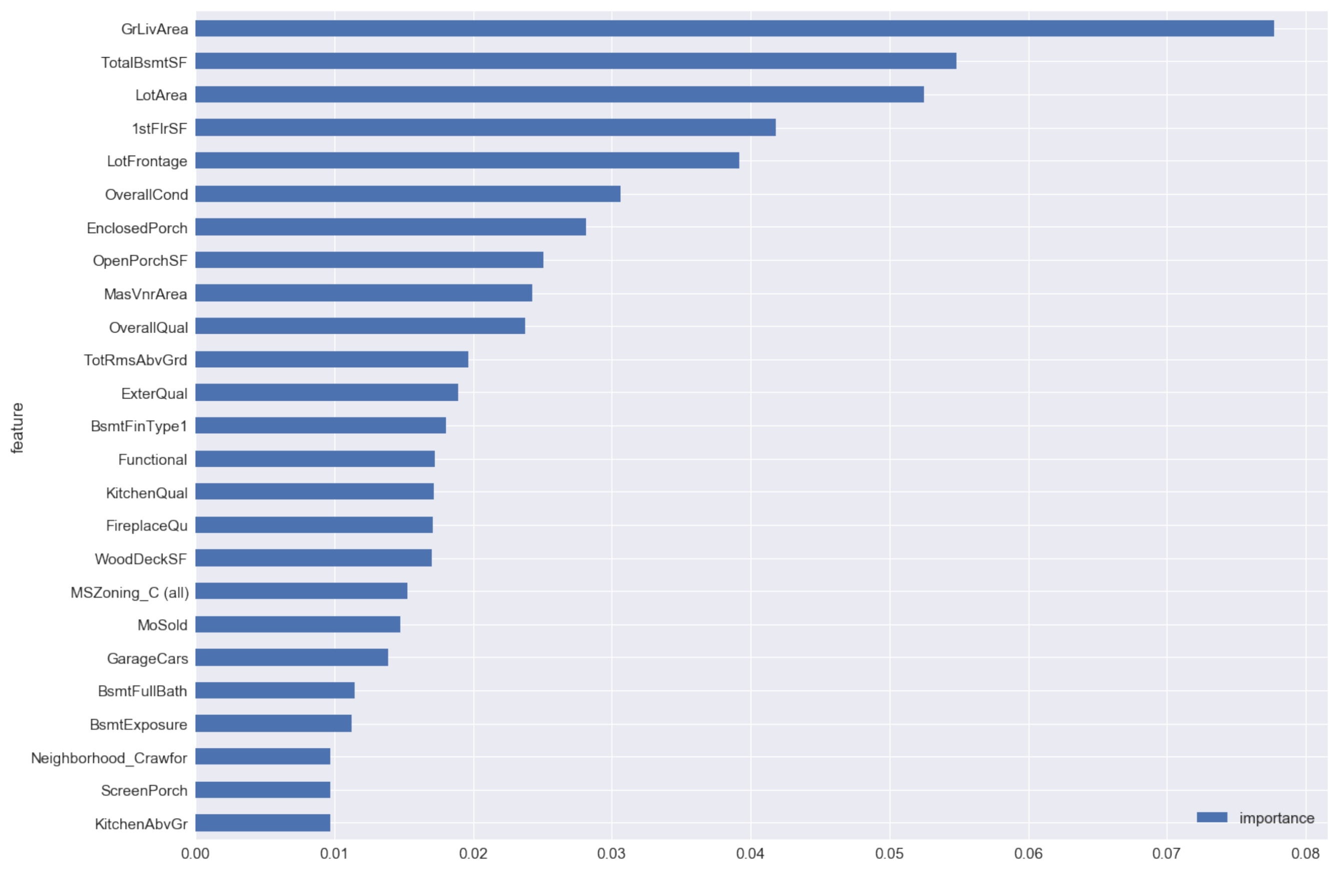

Figure 5   Top 25 feature importance weights.

As depicted in Figure 4 we see that predicted SalePrice shows near perfect correlation with the observed SalePrice. Additionally to explain SalePrice we find that that the above feature shown in Figure 5 do a good job since by the 23rd feature, it's weight drops to a value lower than 0.01.

Reflection

We started off by filling missing data to ensure that further analysis would not require dropping of null values. After that outlier were removed using IQR. We mapped ordinal values to numbers and then began feature reduction in two phases since there were a lot of features to analyze. In the first phase we performed bivariate analysis within feature subset and also with the target variable reducing the number of features in each subset. This process was carried out for each subset. We proceeded to perform the same process inter-feature subset and reduced the number of features even further. Henceforth the latter stages of data preprocessing were performed namely log transformation and one-hot encoding. After this three linear regression models were built, the best among them optimized using GridSearchCV and predictions using the same were saved on the disk concluding the project.

Once the model was built the most important feature were extracted using a parameter feature_importances_. We see that the most essential features of houses explain SalePrice the most, example feature related to living area, basement, lot, porch, rooms, bathrooms and kitchen. We see OverallCond, OverallQual, ExterQual and KitchenQual in the important features since they are an overall measure of that feature. MSZoning_C (all) is important since it denoted a Commercial property. The presence of MoSold is surprising since it's odd to think of a particular house with price fluctuation w.r.t. MoSold unless the feature actually affects the environment rather than system itself, for example in supply chain mechanism.

Improvement

In the bivariate analysis process we use Pearson correlation coefficient to measure linear relationship between numerical feature, however we had to resort to scatter plots and barplots when it came to categorical feature. In the section where we analyze feature in batches we could have segregated them even further in numerical and categorical feature. This would have resulted in 2 fold feature subsets than before. But this would enable the use of Spearman and Kendall correlation to gauge correlation between categorical feature which could in turn give us a better reduced feature set and a possibly a better and more generalized regression model.

Another improvement is the use of model stacking to combine the result of various regression models in an additive manner to generate better predictions since we would we negating the effects of overfitting in every model by averaging out the predictions. It could take the form as shown below.

$$prediction_i = w_1 \cdot prediction_1 + w_2 \cdot prediction_2 + \ ... \ w_n \cdot prediction_n$$
$$where\ prediction_i = i_{th}\, prediction$$
$$prediction_j = predition\, of\ \ j_{th} regression\ \ model \quad for\, 1 \leqslant j \leqslant n$$
$$0 < w_k < 1 \quad for\, 1 \leqslant k \leqslant n$$
$$and\ w_1 + w_2 + ... + w_n = 1$$

Thank you