Housing Prices Prediction

In this report, predictive modeling on Ames housing data was explored. The data was preprocessed to remove outliers, make up missing values, and encode categorical features into numerical ones followed by applying linear models (using regularization), gradient boosting models. To optimize the modeling results stacking, grid search (for parameter optimization) was used.

Extreme outliers affect the efficiency of the model. Hence scatter plot of output feature, Sales Price, against a key input feature, Gross Living Area, was visualized and too deviant datapoints were eliminated from further preprocessing. This step was only performed on training set since output feature data is not available in testing set. Furthermore, the original data (both train and test) had incorrect datatypes, missing values, categorical features. It was preprocessed using standard data cleaning techniques

**Preprocessing**

1. **Converting Datatypes:** Some of the numerical features were incorrectly framed as object/categorical features. The data types were changed wherever needed.
2. **Imputing missing data:** Many features had missing values and needed imputation to make data ready for modeling. First approach was to use statistical measures (mode, median) of a group of records that share same values for other correlated features (e.g., For a record with missing MSZoning value, we check it’s MSSubClass and Neighborhood value and find mode MSZoning value of all the records that share the same MSSubClass and Neighborhood value. We use that mode MSZoning value for the missing value). Whenever this approach was not possible, absolute measures (mode/median of the whole dataset) were used for imputation. When these two cased did not seem to be right approach 0, ‘NA’, ‘None’ values were imputed. Few columns had disproportionate number of missing values and were excluded from further preprocessing.
3. **Feature Encoding:** The ordinal features were converted into numerical features to make them compatible with modeling. [Table with unique ordinal values and corresponding numerical values for any ordinal features]
4. **Removing Highly Correlated Input Features:** Using highly correlated features in predictive models, especially linear regression, results into unreliable parameter values (high variance). Hence features highly correlated to other features are eliminated from input feature set. [Sample Correlation matrix with few features]
5. **Removing Output Feature Skewness:** A common assumption for linear regression is thatthe residual/error values are normally distributed. This translates to output feature values been normally distributed. The output feature had right skewness and it was transformed using log1p transform to make it near normally distributed. This new output variable will be used for modeling. [Figures of SalePrice distribution and log1p(SalePrice) distribution]
6. **One Hot Encoding:** To include the categorical features into input feature set, they were one hot encoded into vectors. The dimension of the encoded feature vector is the number of unique categorical values for the feature. The presence of a particular categorical value is denoted as ‘1’ and absence as ‘0’. [Table with a feature set column and corresponding one hot encoded column]

After these steps data was finally ready to be fit into models and models to be used to predict test data housing prices. For all the models, 80% training data was used for training and remaining 20% was used for cross validation.

**Modeling**

1. **Linear Regression:** Linear Regression is the most common technique to estimate continuous output feature as a weighted linear combination of input features. The weights for each input features are calculated by optimizing Mean Squared Residuals (Residual being difference between observed value and estimated value). This is also known as Least Squared Error Estimator (LSE) of output features. Since we assume the output feature to be normally distributed in Linear Regression, the LSE happens to be the Maximum Likelihood Estimator (MLE).
2. **Random Forest:** Random Forest is ensemble technique which creates Regression Trees by splitting the data based optimal feature splitting criteria. It will create multiple Regression Trees by selected random subsamples of input data per tree. This is known as bagging. Additionally, for each tree, it also selects random subsamples of features to ensure the trees are not highly correlated. The results from all the trees are averaged to create the final estimated values. Due to bagging this algorithm avoids risk of over-fitting.
3. **Regularized Linear Regression:** Linear Regression predicts unbiased estimators but it might generate estimators with high variances when input features are correlated and feature weights are large. To overcome this issue, we can add a Regularization term to with Mean Squared Error term. Optimizing this summation will generate a biased estimator with relatively lower variance. The regularization term penalized large feature weight values, favors lower feature weight values thus generating estimator with lower variance compared to plain linear regression counterpart.
   1. **Ridge Regression:** In ridge regression the regularization term is an L­2 Norm of input feature weights scaled by (whole L2 Norm scaled by) scaler parameter. [Insert the equation]
   2. **Lasso Regression:** In Lasso Regression the regularization term is an L1 Norm of input feature weights scaled by (whole L2 Norm scaled by) scaler parameter. [Insert the equation]
   3. **Elastic Net:** In Elastic Net we use L1 Norm and L2 Norm, both scaled by different scaler parameters, as the regularization term. [Insert the Equation]
4. **Extreme Gradient Boosting (XGBoost):**  This technique builds an ensemble of regression trees by greedily boosting the information gain at each node split while building each tree. At each node the best possible split is searched over the available feature set. For each tree a subsample of features and datapoints is selected and the results of all the trees are averaged to give the final estimate.

**Results**

The regression models can be evaluated using R-Squared Score as well as Root Mean Squared Error (RMSE). The criteria for comparison were chosen to be R-squared score. R-squared score reflects the ability of the model to describe the observed data. Linear Regression and Random Forest gave comparatively similar results. Random Forest Regressor, consistently had slightly lower R-squared score and higher RMSE than Linear Regressor. Since lot of input features had linear relationship with output features, Linear Regressor might have done better job in replicating the observed data than the Random Forest Regressor. [Table with R2 Score and RMSE of RF and LR for same set of training and Cross Validation Data]. Additionally, the scale of data varied a lot from feature to feature and since the dataset was not large enough the subsamples to build Random Forest decision trees might not have been representative of original data. This might be the reason for r2 score fluctuation of Random Forest Regressor over varied combination of training and cross validation data. Also, the residual plot for Random Forest had wider spread than Linear Regression which reflects into respective RMSE values.

Since the input features were slightly correlated all the Regularized Regression models performed better than Linear Regression. The slight bias and lower variance (compared to Linear Regression) can be confirmed from summary statistics and residual plots of all three of the Regularized models. [Residual plots along with Residual Summary Statistics table for all the regressors].

To optimize the regularization parameters the Grid Search, with 5-fold cross validation, was performed on all three Regularized Regressors. The initial range of parameter was randomly chosen and the it was narrowed down to the around best value after each iteration. [Table with 5 parameter values and corresponding r2 score, RMSE for Ridge and Lasso] [ 5\*5 matrix; 1 each for r2 score, RMSE]

XGBoost model performed even better than all other models. Due to the inherent complexity of the model, it was difficult to find out the exact reason the model performed better than other models. The model was fed numerous combinations of tree depth, number of iterations, and learning rate to find the combination that yields best possible learning rate. Model performed better with shallower trees since current dataset was not large enough to prevent overfitting from deeper trees. The model did get better with increasing number of trees but the rate of improvement becomes really slow [5\*5 matrix for tree depth and number of trees (use learning rate 0.05); RMSE]

**Conclusion**

In this project, different regression models were explored on Ames dataset. The linear regressors performed better than random forest regressors due to dearth of datapoints compared to number of features. Since features were correlated the regularized linear regressors performed better than plain linear regressor. XGBoost was best model on current dataset due to bagging of numerous trees and extreme gradient boosting.