Final Project Report

Team ML_Masters: Philip Osborne, Pithayuth Charnsethikul & Raveena Kshatriya

YouTube Video: https://www.youtube.com/watch?v=fXrftNH-ZSU

I. Overview

In this report we present the results of our experimentation in the Kaggle Housing Prices competition. The goal was to create a regression model to predict the sale price of houses based on a mix of categorical, numerical and ordinal features. We experimented with several ML models and our final model is a stacked model consisting of a *Random Forest Regressor*, *Support Vector Regressor (SVR)*, *Ridge Regression*, *Lasso Regression*, and a *Light Gradient Boosting Machine (LGBM)* with a final RMSE of 0.11725 on the test set provided by Kaggle.

II. Data Preprocessing

To preprocess the data for the models, we filled missing values using the average of numerical variables and adding an 'Unknown' category for categorical variables. We converted the ordinal categorical variables using numerical mappings, and also engineered new features by creating combinations of some related variables. Outlying data points were also carefully examined and removed from the training dataset, and several irrelevant features or features with large percentages of missing data were removed from the feature set. Lastly all of the categorical (non-ordinal) variables were one-hot encoded, and numerical variables were scaled to fit a normal distribution. When training the models, the log of the target variable was used to match the Kaggle scoring guidelines.

III. Experiments

The evaluation metric used during training was the Root Mean Squared Error between the log of the target variable and the model's predicted value. During training, 5-Fold cross validation was used to tune hyperparameters and measure the model's ability to generalize to unseen data. We present the performance of each regression model in terms of RMSE in the below table:

Model	5-Fold CV Score	Best Test Set Score
Linear Regression w/ Feature Selection	0.1337	0.15148
Ridge	0.1123	0.13393
Lasso	0.1092	0.13499
ElasticNet	0.1091	0.13469

Kernel Ridge	0.1096	0.13208
SVR	0.1094	0.12024
Decision Tree	0.1868	0.18957
XGBoost	0.1157	0.13910
Multi-Layer Perceptron	0.1501	0.12720
Stack1 (Lasso, ElasticNet, Ridge, Kernel Ridge, SVR, Bayesian Ridge)	0.1096	0.12012
Blend (Ridge, Kernel Ridge, SVR, Stack1)	0.1092	0.12153
Stack (Random Forest Regressor, SVR, Ridge, Lasso, and LGBM Regressor)	**0.1181**	**0.11725**

Figure 1: RMSE for each regression model

Note: Linear regression was performed using only the 12 features with highest correlation with the target variable:
GrLivArea, TotalSF, TotalQual, GarageCars, TotalBath, KitchenQual, TotalBsmtSF, OverallQualCond, FullBath, TotRmsAbvGrd, YearBuilt, YearRemodAdd

IV. Hypothesis & Observations

From figure 1, we can see that models with lower complexity (Vanilla *Linear Regression* and *Decision Tree*) suffer from higher bias and are not able to learn all of the complex relationships within the dataset. Models with greater complexity (*Ridge*, *Lasso*, *ElasticNet*, *Kernel Ridge*, *SVR*, etc.), on the other hand, have better performances and many of them are able to beat the first baseline (0.138) but only *SVR* is able to beat the second baseline (0.121).
We also notice that many of these models (*Ridge*, *Lasso*, *ElasticNet*, *Kernel Ridge*) are overfitted because even though their 5-fold scores are less than the second baseline, their test scores are greater than. On the other hand, a *Multi-Layer Perceptron* seems to do much better on the test set. We suspect that because a *Multi-Layer Perceptron* (neural network) is more complex and powerful than other models, it can better understand the relationships within the dataset.

At this point, we also hypothesize that we should combine several model's predictions together so that we can get more accurate predictions. At first, we manually assign a weight to each model. Unfortunately, we get worse prediction accuracy --- *Blend* has worse accuracy than *SVR*. Hence, to intelligently decide the weight for each model, we apply an ensemble method, Stacking. We choose a list of models and set a meta regressor to *Linear Regression*. In Stacking, each model is trained individually and then the meta regressor is trained by the prediction results of the individual models. In other words, we combine model's predictions

based on the weights decided by *Linear Regression*. As we expected, *Stack* boosts the final prediction accuracy to 0.11725.

To examine the features that are most important in our final Stack model, we examine the permutation importance of each feature. The permutation feature importance is a measure of the decrease in model score when a given feature is randomly shuffled, which breaks the relationship between the feature and the target. From this we can get an estimate about how much a feature contributes to the predictive power of our model.

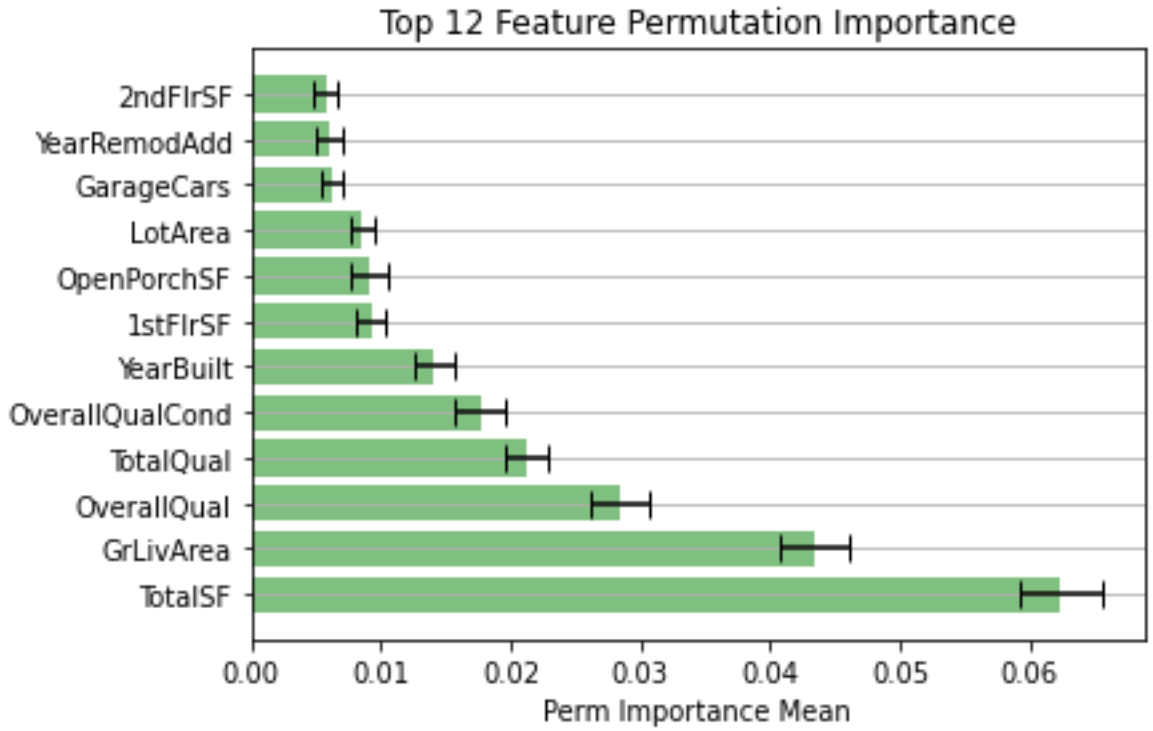

Figure 2: Top 12 Feature Permutation Importance Means

V. Conclusions

Feature Importance:

We determined the features that have the highest correlation with the target variable, which tells us about which features are most important in predicting the target variable. These features make intuitive sense when we think about what some of the most important factors are when people are evaluating the price of a house. This correlation analysis confirms that certain features like Total Square Footage, Overall Quality, Number of bathrooms, and the age of the home are indeed some of the most important factors that contribute to the price of a home. We can see that very similar features also contribute greatly to the predictive power of our final model. This gives us confidence in the quality of our final model, since we can see that our model is placing a high weightage on the most important features in the dataset based on correlation and what we intuitively know contributes to the price of a house.

Model Optimization:
Although most models can not individually beat the second baseline, ensemble methods or combinations of multiple models can potentially help us get better accuracy. Using a stack model not only allows us to leverage the predictive power of several different models, it also properly balances each model's contribution so that it makes sure that the combination of models will yield the optimal predictions.

VI. References

- https://towardsdatascience.com/interpreting-machine-learning-models-c7646393c270
- https://scikit-learn.org/stable/modules/permutation_importance.html
- https://www.kaggle.com/nareshbhat/outlier-the-silent-killer
- https://www.kaggle.com/zugariy/regression-blending-and-stacking-v-02
- https://rasbt.github.io/mlxtend/user_guide/regressor/StackingCVRegressor/