



PROJECT 2 – REPORT

HOUSE PRICES: ADVANCED REGRESSION TECHNIQUES

Submitted by:

Tazein Fatma



Executive Summary

PROBLEM: Prediction of House Prices

This project is based on the Kaggle competition - “**House Prices – Advanced Regression Techniques**”. It requires us to predict SalePrice of residential homes in Ames, Iowa, based on independent variables/features given in the data set. This is an interesting problem, as we see that based on past Ames Housing data, sale price of each house varies based on as many as 79 features. There are some obvious factors like quality of the house, total living area, year in which the house was built but there are lot of other not so obvious factors, that play an important role in guiding the house prices. With so many independent variables, this study provides an opportunity to clean the data and test the relevance of features and do detailed feature analysis. We are supposed to run regression models/algorithms on this data and choose the ones that can provide better predictions. We are required to submit a CSV file of predictions of house prices on Kaggle and test the accuracy. The file should contain a house ID and Sales price columns. Submissions are evaluated on [Root-Mean-Squared-Error \(RMSE\)](#) between the log of the predicted value and the logarithm of the observed sales price.

DATA :

Files provided by Kaggle: **train.csv** - the training set & **test.csv** - the test set , **data_description.txt** and **sample_submission.csv**

The “train” data set has 1460 rows and 81 columns with 79 independent variables, one ‘Id’ column and one ‘SalePrice’ column, which is our dependent variable. The test set has 1459 rows and 80 columns as it does not contain the “SalePrice” column that we are supposed to predict. It is interesting to see, that there can be as many as 79 characteristics of the houses that buyers might be interested in looking at, before buying a house. There are features like OverallQual, GrLivArea, GarageCars, YearBuilt, Fireplaces and so on. Data is a mix of continuous and categorical variables. There are lot of missing values which need to be updated with some meaningful values.

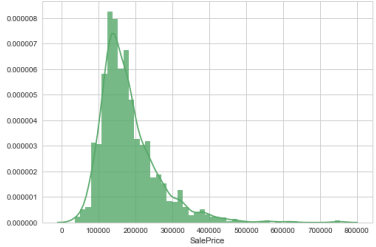
FINDINGS:

This was my first modelling project and provided lot of learnings. I kept the model simple and spent most of the time in fixing the missing values in each of the feature, substituting them with “None”, “0”, most occurring values or some other meaningful terms. I found that simply removing missing value fields or features is not wise and we should try to keep the sanctity of data. After fixing all the missing values, I ran Lasso regression, Kernel Ridge Regression, Elastic-Net regression and Gradient Boosting models. I fine-tuned the parameters within each model and tried to arrive at a combination which gave me better submission score. I submitted CSV files based on each model individually first and then dropped the Kernel Regression method as it provided worst score. In the end I tried an ensemble model using weighted averages of Lasso, Elastic-Net and Gradient boost, which gave the best score out of all combinations.

Analysis of relevance of independent variables (features)

Train set has 1460 rows and 81 columns and Test set has 1459 rows and 80 columns. There are 35 continuous variables and 43 categorical features.

SalePrice – The Dependent Variable:

| | | |
|---|---|---|
|  | count 1460.000000 mean 180921.195890 std 79442.502883 min 34900.000000 25% 129975.000000 50% 163000.000000 75% 214000.000000 max 755000.000000 | OverallQual- 0.8171846144867654 GrLivArea- 0.7009269871427154 GarageCars- 0.6806248726581887 GarageArea- 0.6508876811435949 TotalBsmtSF- 0.6121342283262261 1stFlrSF- 0.596981323185535 FullBath- 0.5947706649972533 YearBuilt- 0.5865701927897146 YearRemodAdd- 0.5656077814623213 GarageYrBlt- 0.5410727772673495 TotRmsAbvGrd- 0.5344224002094375 Fireplaces- 0.48944954515747635 |
| Right-skewed – Log transformed it to get normal distribution. | SalePrice – Descriptive Stats | Top features – correlation with SalePrice |

Dropped the “ID” field and “SalePrice” field:

As ‘ID’ is irrelevant for prediction and “SalePrice” is our dependent variable which we need to predict.

Combined Train & Test Set: Before starting detailed feature analysis, I combined the Train & Test Data.

Checked Missing Values for 79 Features: On close analysis it shows that lot of fields in the data have missing values. There were almost **34 features** that had missing values, out of those “PoolQC”, MiscFeature, Alley, Fence, Fireplace had highest missing values. As many as **2909** for PoolQC and so on.

Replaced NAs with meaningful values:

PoolQC : Has highest missing values with NA written. It assumed that NA means there is no pool. So, most of the houses are without pool. So I replaced NAs with “None”

```
full_data["PoolQC"] = full_data["PoolQC"].fillna("None")
```

FireplaceQu : data description says NA means "no fireplace"

```
full_data["FireplaceQu"] = full_data["FireplaceQu"].fillna("NoFireplace")
```

GarageYrBlt, GarageArea and GarageCars have NAs which means there is no Garage in these homes, so we replace them with '0'. Same way there are some basement related features which were continuous, so I replaced their NAs with '0', like BsmtFullBath, BsmtFinSF1 etc. There were some more Basement related features like BsmtQual, BsmtCond which were categorical and had NAs, so I replaced them with 'None', as NA indicates that there is no basement. Similarly, I went through each feature and replaced the NA values with either the mode of that feature or other meaningful value.

Dropped ‘Utilities’: All records for this feature are "AllPub", except for one "NoSeWa" and 2 NA . As this feature has only one value there is no point keeping it as it doesn't affect the prediction.

Transformed numerical variables into categorical as they are classification variables. For example,

MSSubClass has values like 20,40, 45, 60 up to 190. Same way ‘**OverallCond**’, ‘**YsSold**’ & ‘**MoSold**’ were changed to categorical.

Transformed all categorical variables to DUMMY variables.

Once all the features were dealt with I split them apart into Train & Test data-set.

Analysis of performance of different model types (different algorithms)

I applied regression models to the Train data using 5 K-fold cross validation technique.

Lasso Regression:

I tried varying alpha values for this model and **alpha=.0005** gave the best score within Lasso model.

Lasso uses **RobustScaler()** method which scale features using statistics that are robust to outliers.

I calculated lasso_preds values by fitting the model on train set and SalePrice predictions were made over the Test set.

RMSE score using k-fold cross validation was: 0.1352

On submission to Kaggle Lasso model gave the score of: **0.12673**

Kernel Ridge Regression:

I tried to improve the score of Kernel Ridge by changing the alpha values, changed the Kernel to 'Linear'. The RMSE score using k-fold cross validation was: 0.1469. Since, the RMSE score of this regression technique was worst amongst all other models used so I dropped this model.

Elastic Net Regression:

Elastic net regression is a hybrid approach that blends both penalization of the L2 and **L1** norms. By changing value between 0 and 1 it controls how much L2 or **L1** penalization is used (0 is ridge, 1 is lasso). I tried to vary the L1_ratio along with alpha values but the best combination was alpha = .0006 and L1_ratio = 0.9.

RMSE score using k-fold cross validation was: 0.1350

On submission to Kaggle Elastic-Net model gave the score of: **0.12705**

Gradient Boosting Regression:

I tried improving the score by using Gradient boosting with 'huber' loss, which makes it robust to the outliers in our data. Honestly, I do not yet understand all the parameters of Gradient Boost, so I could not vary the parameters too much. But still the RMSE score using this regression was the lowest.

RMSE score using k-fold cross validation was: 0.1252

On submission to Kaggle GBoost model gave the score of: **0.12738**

Ensemble Model:

I checked to see if using weighted average of above three models would change/improve the score or not. So, I played with few combinations of weights for each model and below mentioned combination seems to give the best score on Kaggle compared to the individual submissions.

Ensemble model is an average of Lasso, Gradient Boosting and Elastic Net predictions.

preds = 0.2*lasso_preds + 0.5*GBoost_preds + 0.3*ENet_preds

On submission to Kaggle Ensemble model gave the score of: **0.12295**

Comparison of submission scores all models

| | |
|--|----------------|
| LASSO Regression - RobustScaler() | 0.12673 |
| Elastic Net Regression - l1_ratio | 0.12705 |
| Gradient Boost Model - loss='huber' | 0.12738 |
| Weighted Average of all 3 | 0.12295 |

Conclusion:

The ensemble model using weighted average of Lasso, Gradient Boosting and Elastic-Net regression gives the best score. It is interesting to note that although the RMSE scores of Gradient Boosting were the lowest but the submission score on Kaggle was not the lowest out of all three. Using weighted average of all three, I was able to get best score out of all other methods.

Screen shot of CSV submission on Kaggle using ensemble model:

| Your most recent submission | | | | |
|--|-----------|-----------|----------------|---------|
| Name | Submitted | Wait time | Execution time | Score |
| submission_average2L_5G_3E.csv | just now | 0 seconds | 0 seconds | 0.12295 |
| Complete | | | | |
| Jump to your position on the leaderboard ▼ | | | | |