PSL Project 1 Report

* Team member 1: Sushanta Panda (net Id = panda5)
  + Developed the code end-to-end and submitted the code and the report.

Introduction to Ames Housing Dataset

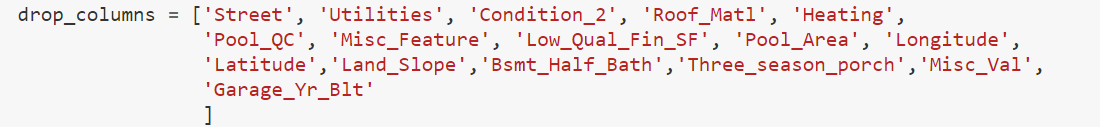
Goal

Goal of the Project is to predict the “Sale\_Price” of the Ames Housing Dataset where we need to create 2 separate model, 1st is from the Linear Regression Category (Lasso / Ridge or Elastic net) and 2nd is from the boosting (Random Forest / Xgboost or Light GBM). The metric to measure the accuracy would be the Root Mean Square Error (RMSE), where the ‘Sale\_Price’ would be converted in the log scale. We have given 10 folder sets (Train.csv + Test.csv), where after the model is being trained the test accuracy of the 1st 5 folder should be below < 0.135 and for the rest 5 folder would be < 0.125.

Data Processing

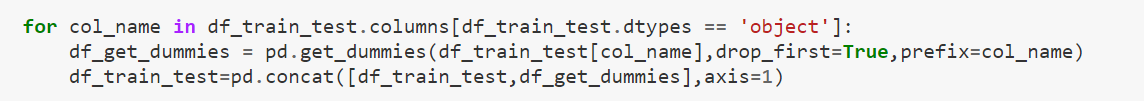
As part of the data processing, I have done the following transformation / feature engineering.

* **Drop Feature which has lots of “NA” value** – Removed the column ‘‘Garage\_Yr\_Blt’ which has lots of ‘NA’ value. Previously have tried to impute from ‘NA’ to ‘0’, however it seems dropping of the feature in its entirety has improved the test accuracy by 0.5
* **Drop Feature** – Removed the following Feature.



All the above features have been dropped because the only 1 value of the features holds > 95% of the data, which kind of noise and impact the accuracy. Also verified, by removing these variables, improve the accuracy of the model.

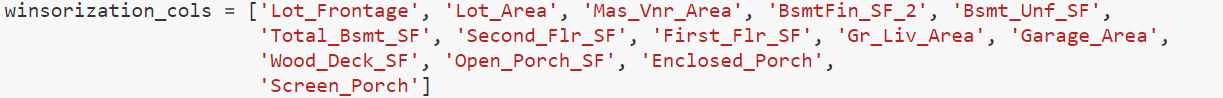
* **Converting Features into Dummy Variable** – Converting all the Categorical Features into dummy variable. If a feature as “N” levels, created “N-1” columns for the same features with prefix as the column name. Used the Pandas function “get\_dummies” to achieve this, below is the code snippet.



While converting into the dummy variable, the train and test set have been merged before the “get\_dummies” implemented into the data frame.



* **Winsorization** – Winsorized the following continuous variables to cap the value to 0.95 quantile.



* Cross Validation – Used 10 K Fold Cross Validation to validate the model and test out the best hyper parameter to gain highest accuracy. Below is the screen shot for the Cross Validation for the 2 model, Lasso & Xgboost

|  |  |
| --- | --- |
| Lasso | Xgboost |
|  |  |

Fitting Prediction Model

Following are the 2 models created as part of the Project

1. **Lasso (Linear Model)**
   1. Created the Lasso model with alpha = 0.0001. The value of the alpha comes after the CV runs on different value of alpha (for e.g., 0.1, 0.01, 0.001 etc). It seems the Validation and test error is lowest for the alpha = 0.001
   2. Total Time taken for processing a single train.csv & test.csv is around **2 second**
   3. One Hot Encoding, where the categorical variable is converted into one hot and 1 value per Feature’s value (and dropping 1 feature from the overall) helps the accuracy jumps around from 0.17 to 0.12 (around 0.06 RMSE)
   4. Winsorization helps to improves the accuracy around 0.002 for some of the fold.
2. **Xgboost (Boosting Model)**
   1. Created the Xgboost model with the same feature set which is used for the Lasso
   2. Used the hyperparameter colsample\_bytreee = 0.1, learning\_rate = 0.04, max\_depth = 25, alpha = 1 and n\_estimators = 1000
      1. Decrease the learning\_rate to 0.001 or 0.0001 with increase in n\_estimators = 10000 will not help to increase the accuracy. Learning\_rate = 0.04 helps to gives better accuracy.
      2. Increasing the “alpha” value will decrease the accuracy of the model
      3. Decreasing “max\_depth” decrease the accuracy. However, after increasing from certain value like 25 or 30, the accuracy doesn’t improve. Hence, keeping the “max\_depth” to “25”
   3. Xgboost seems taking more time around **40 – 50** seconds to train and predict for a given folder.
   4. One Hot Encoding, where the categorical variable is converted into one hot and 1 value per Feature’s value (and dropping 1 feature from the overall) helps the accuracy jumps around from 0.17 to 0.12 (around 0.06 RMSE)

Observation / Conclusion

1. Xgboost and Lasso seems to be performed in same with respect to accuracy in the Validation / Test Set
2. Lasso seems to be 20/30 times faster as compare to Xgboost, because of how boosting works with tree methods

Accuracy Table

|  |  |  |
| --- | --- | --- |
| Folder | Lasso | Xgboost |
| 1 | 0.124 | 0.126 |
| 2 | 0.114 | 0.114 |
| 3 | 0.126 | 0.126 |
| 4 | 0.132 | 0.132 |
| 5 | 0.131 | 0.131 |
| 6 | 0.124 | 0.124 |
| 7 | 0.114 | 0.114 |
| 8 | 0.126 | 0.126 |
| 9 | 0.132 | 0.132 |
| 10 | 0.131 | 0.131 |

Hardware Configuration

* iMac Pro (2017) 3.2 Ghz Intel Xeon W, 32 GB 2666 MHz DDR4
* Jupyter Notebook 6.0.3, Python 3.7.8

Running Time

|  |  |
| --- | --- |
| Total Time Taken | 4.7 Seconds |
| Lasso Training + Prediction | 0.13 Seconds |
| Xgboost Training + Prediction | 4.28 Seconds |

References