PSL Project 1 Report

Sushanta Panda (net Id = panda5)

Introduction to Ames Housing Dataset

Ames Dataset is the housing dataset, where the features are about a house. The dataset contains 2930 rows and around 83 Features (Including **Primary Key** for each Property 'Property ID' and **Target Variable** – 'Sale_Price'). The dataset contains **46** Categorical variable and **37** continuous variables (including PID and 'Sale_Price).

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.

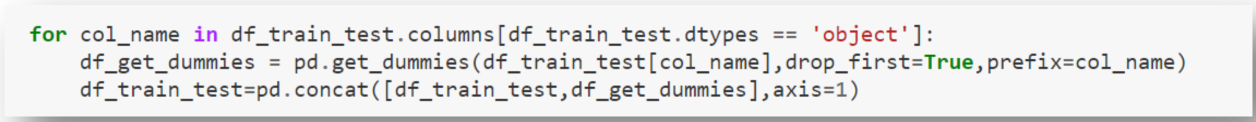
- **Drop the Feature "Garage_Yr_Blt" which has lots of 'Na'**– Dropped the feature which has lots of 'na'. Previously impute the value, where 'na' is being filled by "0". However, dropping the feature seems increase the test accuracy to "0.004".

- **Drop Feature** – Removed the following Feature.

```
drop_columns = ['Street', 'Utilities', 'Condition_2', 'Roof_Matl', 'Heating',
                'Pool_QC', 'Misc_Feature', 'Low_Qual_Fin_SF', 'Pool_Area', 'Longitude',
                'Latitude','Land_Slope','Bsmt_Half_Bath','Three_season_porch','Misc_Val',
                'Garage_Yr_Blt']
```

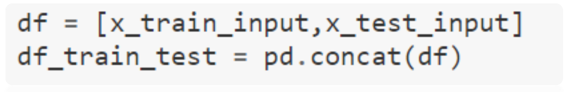
  All the above features have been dropped because the only 1 value of the features holds > 95% of the data, which kind of noise and impact the accuracy. After removing these variables, the test accuracy improves.

- **Converting Features into Dummy Variable** – Converting all the Categorical Features into dummy variable. If a feature as "N" levels, created "N-1" features for the same features with prefix as the column name. Used the Pandas function "get_dummies" to achieve this. This improves the accuracy 0.17 to 0.12 (around 0.06 RMSE). Below is the code snippet



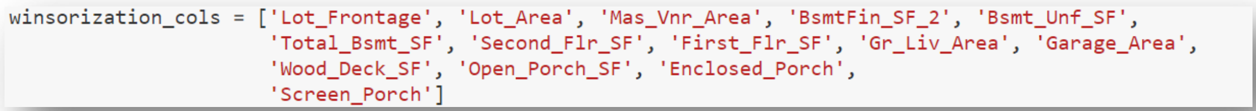
```
for col_name in df_train_test.columns[df_train_test.dtypes == 'object']:
    df_get_dummies = pd.get_dummies(df_train_test[col_name],drop_first=True,prefix=col_name)
    df_train_test=pd.concat([df_train_test,df_get_dummies],axis=1)
```

While converting into the dummy variable, the train and test set have been merged before the "get_dummies" implemented into the data frame, as the code snippet below.



```
df = [x_train_input,x_test_input]
df_train_test = pd.concat(df)
```

- **Winsorization** – Winsorized the following continuous variables to cap the value to 0.95 quantile, for the following feature set.



```
winsorization_cols = ['Lot_Frontage', 'Lot_Area', 'Mas_Vnr_Area', 'BsmtFin_SF_2', 'Bsmt_Unf_SF',
                      'Total_Bsmt_SF', 'Second_Flr_SF', 'First_Flr_SF', 'Gr_Liv_Area', 'Garage_Area',
                      'Wood_Deck_SF', 'Open_Porch_SF', 'Enclosed_Porch',
                      'Screen_Porch']
```

- **Cross Validation** – Used 10 K Fold Cross Validation to validate the model and test out the best hyper parameter to gain good accuracy.

Fitting Prediction Model

Following are the 2 models created as part of the Project

1. **Lasso (Linear Model)**
    a. 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. This is the L1 regularization term, higher the value of alpha, the, model will be more conservative and control the overfitting. Found the value of alpha = 0.0001 which gives the best accuracy on the validation test as well as on the test set.
2. **Xgboost (Boosting Model)**
    a. Created the Xgboost model with the same feature set which is used for the Lasso
    b. Used the hyperparameter colsample_bytreee = 0.1, learning_rate = 0.04, max_depth = 25, alpha = 1 and n_estimators = 1000
        i. Learning rate is the step size which helps to reach the optimum in join with the n_estimators (that many times, needs to use to update the parameters). Decrease the learning_rate from 0.04 to 0.01 / 0.001 / 0.0001, with increase in n_estimators = 10000 will not help to increase the accuracy. Learning_rate = 0.04 helps to gives better accuracy. This is the boosting step, small the number, n_estimator is needed to reach the optimum, where I have chosen the n_estimators = 1000.

ii. Increasing the "alpha" (L1 regularization on weights) value will decrease the accuracy of the model. Because the higher the value of alpha, model will be more conservative, and will not be able to predict the unknown/unseen data.

iii. 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". This is the maximum depth of the tree. Increasing the value will make the model more complex, as deeper branch will be created form the feature split. More the model will complex, little flexibility to predict the new / unknown data. Hence, the optimum value I have choose is around "25"

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 (Test Error)	Meet Benchmark	Lasso – Running Time (In Seconds)	Xgboost (Test Error)	Meet Benchmark	Xgboost – Running Time (In Seconds)
1	0.124	Yes	0.13	0.126	Yes	4.27
2	0.114	Yes	0.25	0.117	Yes	4.22
3	0.126	Yes	0.13	0.126	Yes	4.32
4	0.132	Yes	0.22	0.133	Yes	4.22
5	0.131	Yes	0.13	0.131	Yes	4.31
6	0.124	Yes	0.14	0.126	No	4.30
7	0.114	Yes	0.22	0.116	Yes	4.18
8	0.126	No	0.13	0.126	No	4.31
9	0.132	No	0.23	0.132	No	4.29
10	0.131	No	0.12	0.131	No	4.40

Note: The Running time is only for the Training and Prediction, not included the Loading of the Libraries / data (train.csv, test.csv) and pre-processing (One Hot, Winsorization, Dropping of variables)

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

References

- Winsorization - https://piazza.com/class/kjvsp15j2g07ac?cid=191