**PROJECT 1: PREDICT THE HOUSING PRICES IN AMES**

For this project we used the Ames housing data that was provided to us. The aim was to build a regression model that can predict the price of a house given its set of attributes.

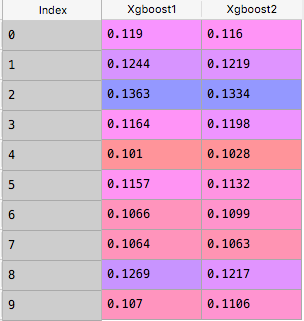
I started off by doing some exploratory data analysis to get a first-hand feel of how the data looks like and what pre-processing would be helpful.

**Following are the steps that I followed:**

1. Through the EDA I found **1 variable** “Garage\_Yr\_Blt”, having **missing** **value** so I planned to drop that.
2. In addition to that I dropped “﻿Condition\_2”, “﻿Utilities”, “﻿Roof\_Matl”, “﻿Latitude”, “﻿Longitude”
3. Since the target variable was **skewed**, I decided to take a **log** of it to reduce the skewness.
4. Segregating the **Numerical** and **Categorical** variable and further filtered them down to **Discrete**, **Continuous**, **Ordinal**, **Nominal**.
5. Read the variable description page which mentioned the levels that were used in the Ordinal variable, used that to perform **Integer** **Encoding**.
6. As suggested on the instructions page, performed **discretization** on “Year\_Built” and “Year\_Remod\_Add” variable.
7. Next, I considered the Nominal data and I found that there were many variables which had some 3-5 **frequent** **levels** and other had a low cardinality so performed grouping of **low** **cardinality** levels and binned them into one level.
8. From the EDA, I found very prominent **outlier** in the “Gr\_Liv\_Area” and “Total\_Bsmt\_SF” variable, so **removed** **3** outlier observations from the data.
9. Since other variables were also prone to outliers I used the **interquartile** **range** and identified the upper and lower range for every variable, those having **extreme** values were **replaced** by their respective limits.
10. To fix **collinearity**, I filtered out columns that were corelated with each other and those that had correlation larger than a threshold (70%) were dropped.
11. Also, there existed some predictors with **0 variance**(constant) and some **qadi constant** predictors (very small variance), so I filtered out those predictors and **removed** them.
12. Next, I perform **feature scaling**, to ensure better performance of the machine learning model.
13. I tried out with **Lasso**, **GradinetBoosting**, **LightGbm**, **ElasticNet**, **Xgboost**, **RandomForestRegressor** to get a feel of how well each of them do, and after doing lots of testing and model tuning, I ended up with using 2 different version of Xgboost.
14. For every model I calculated the **logarithmic** prediction, **transformed** them back by taking the **exponent** and saved them to a text file.

**Model Evaluation Rmse for different splits:**

**Part 1**



For Project 1 **Part 2** I found the **rmse** = ﻿0.11597, using the provided split.

**Running Time:**

For Project 1 **Part 1** on an average the script took **40 seconds** to execute.

For Project 1 **Part 2** I found the running time to be **8.99 seconds**.

**System Specifications:**

Model Name: MacBook Air

Model Identifier: MacBookAir7,2

Processor Name: Intel Core i5

Processor Speed: 1.6 GHz

Number of Processors: 1

Total Number of Cores: 2

L2 Cache (per Core): 256 KB

L3 Cache: 3 MB

Memory: 8 GB

**Acknowledgements:**

I went through many Kaggle kernels and found some useful pre-processing steps , experimented with them on my jupyter notebook and used some of them that were found useful for my model.

**Libraries:**

Name: pandas Version: 0.23.4

Name: numpy Version: 1.15.2

Name: xgboost Version: 0.80

Name: scikit-learn Version: 0.20.0