Capstone Project 1 - Milestone Report

Project Scope

The scope of this project is to predict home prices in Ames, Iowa. This project is based on a competition on Kaggle, **House Prices: Advanced Regression Techniques**. The goal of this project is to understand and apply the right feature engineering and also practice regression techniques like random forest and gradient boosting.

What is The Problem?

Use the Ames, Iowa housing data set to successfully predict home prices using machine learning.

Who is the Client?

There are several clients.

a) Homebuyers/sellers - with the ability to predict home prices, both home buyers and home sellers are able to optimize their selling or bidding prices.
b) State - with the ability to predict home prices, the state is able to predict the property tax revenue.

Data Set and Data Wrangling

Where is the data from and what's in the data?

The training and test data is provided on Kaggle. The files include data such as sale price, lot area, lot shape, neighborhood, basement condition, year built, heating, central air etc.

In total, there are 83 features in the data set - 40 of them are numerical and 43 are categorical.

The train data set has 1460 rows of data and the test data set has 1459 rows of data.

Missing Data

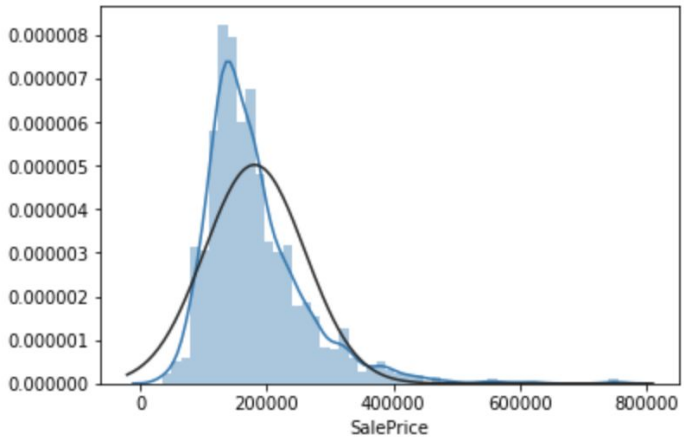
For categorical features that have missing data, fill the missing data with “None”. For numerical features that have missing data, fill the missing data with the mean value of the column. Apply these rules to both train and test data sets.

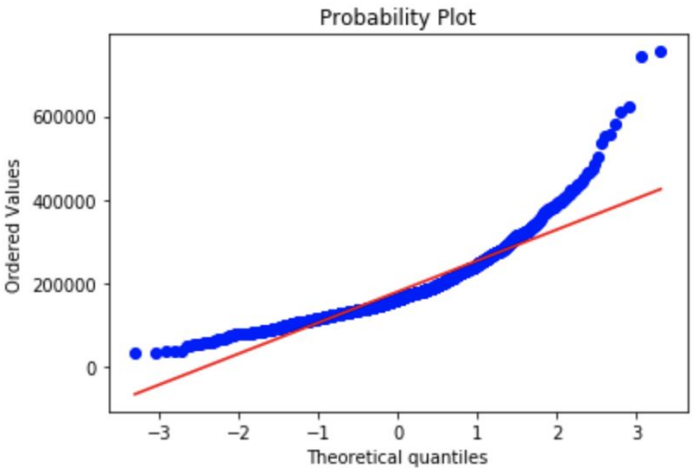
Exploratory Data Analysis

What is the Distribution of Target Feature - SalePrice?

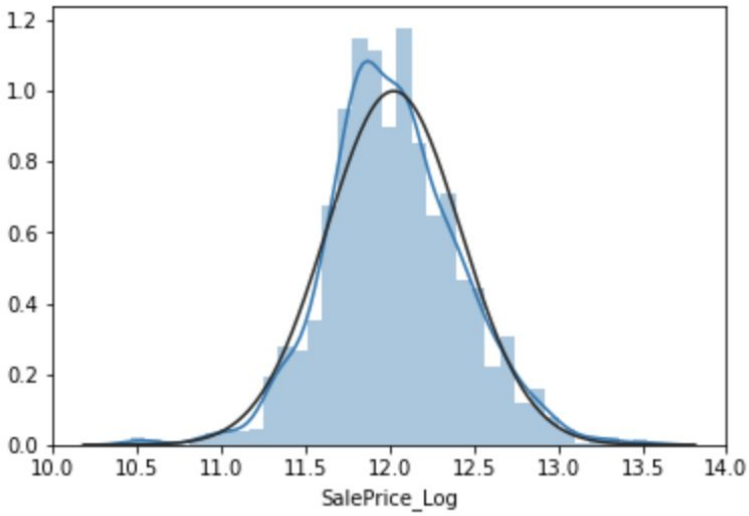
By charting the SalePrice in a histogram against a Normal Probability Plot, we can see that SalePrice is not a normal distribution. The peakness and skewness of SalePrice do not follow the diagonal line of a Normal Probability Plot.

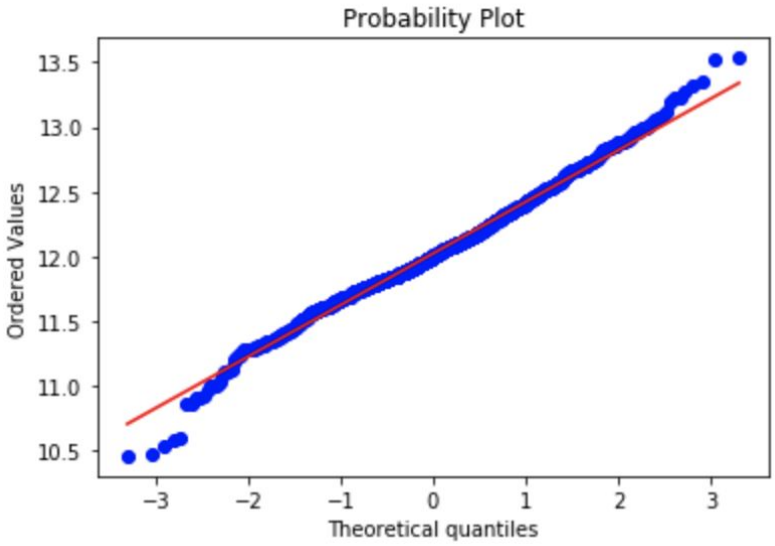
```
sns.distplot(df_train['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['SalePrice'], plot=plt)
```





Since certain ML regression models assume normal distribution, we need to make a log transformation to SalePrice. Once that's done, we plot the Log SalePrice histogram against the Normal Probability Plot again to make sure it looks like a normal distribution. Based on the charts below, the log SalePrice is now a normal distribution.

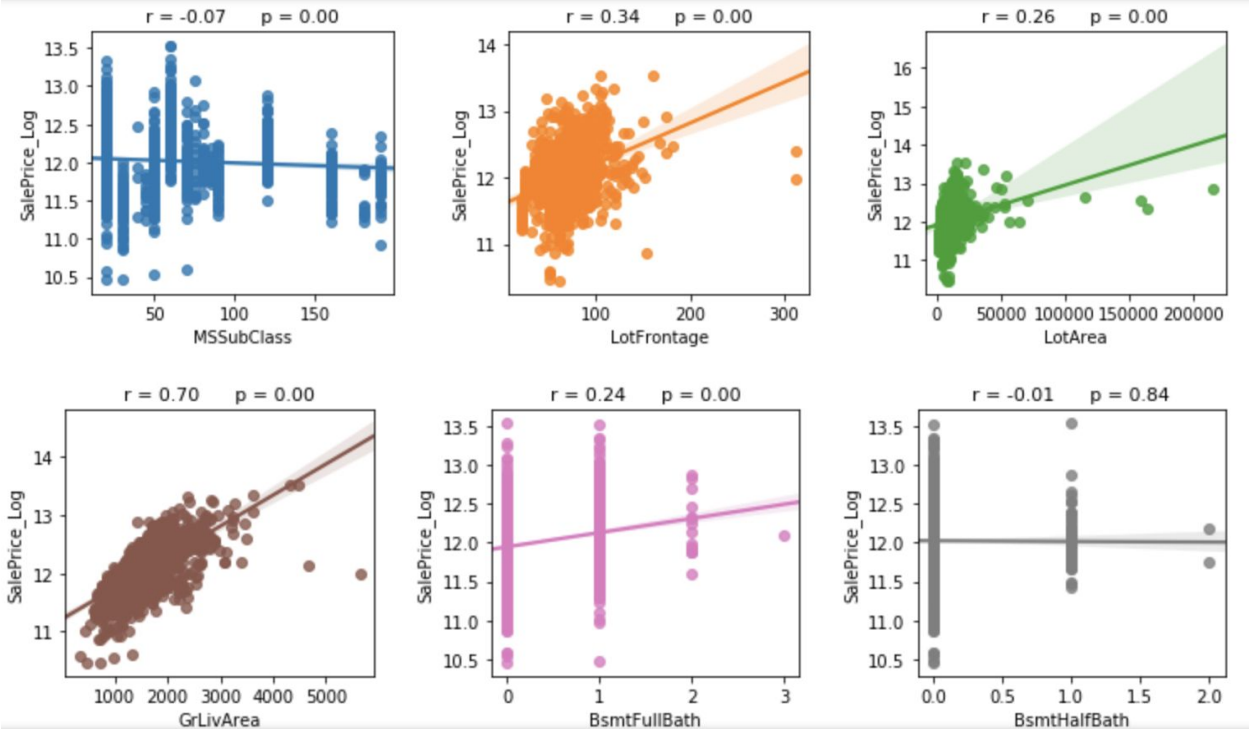




Relationships Between Numerical Features and SalePrice

Visualize The Relationships

Utilize scatter plots to visualize the relationship between SalePrice and each of the numerical features. Use the Pearson correlation coefficient to measure the linear relationship between two datasets. An **r score** is between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply an exact linear relationship. Positive correlations imply that as x increases, so does y. Negative correlations imply that as x increases, y decreases.

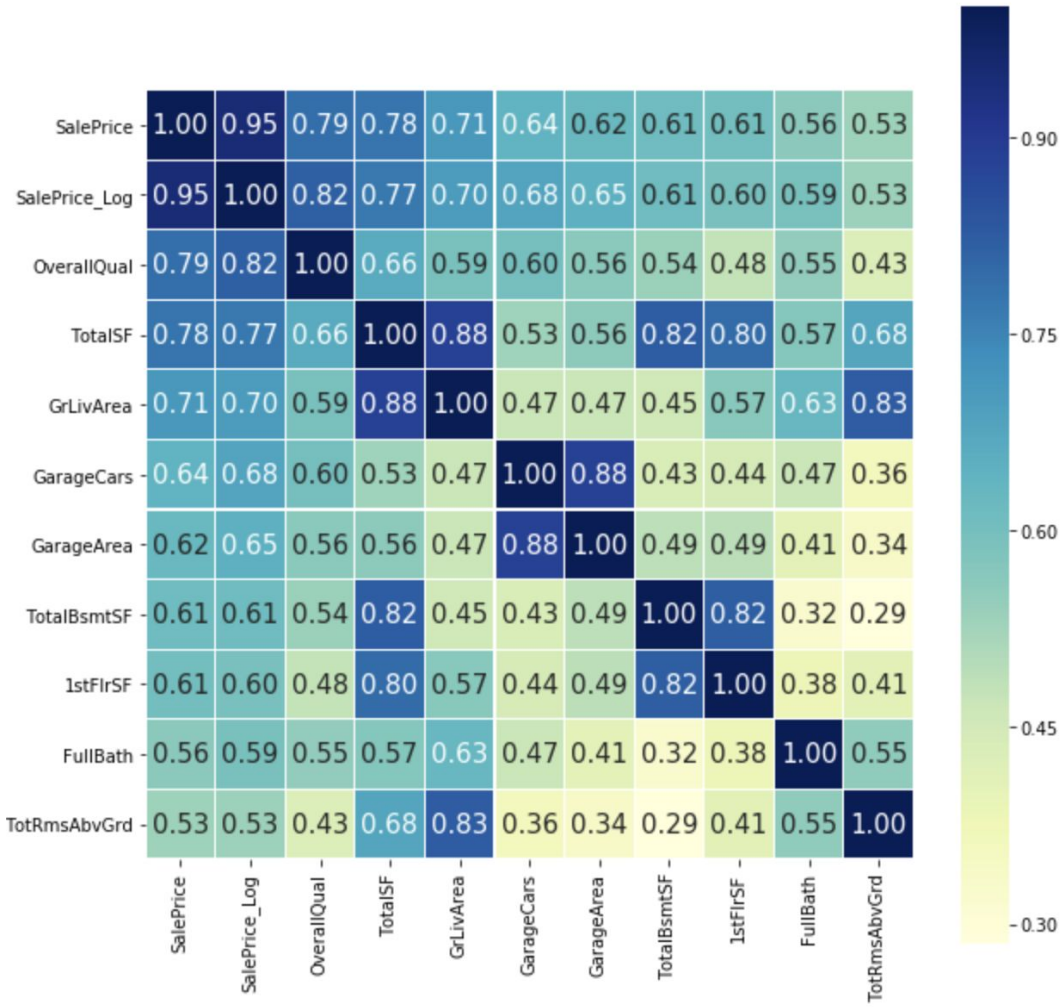


Top Numerical Features with the Highest Correlation to SalePrice

Visualization is great, however it's hard to select the top correlated features when there are so many features. Compute pairwise correlation of columns using DataFrame.corr and it returns a matrix. With this, we can easily select the top 10 features with the highest correlations to SalePrice and plot a heatmap using Seaborn.

```
#Method 1 to get the top 10 correlated variables that are related to SalePrice,
k = 11
cols = corr_matrix.nlargest(k, 'SalePrice')['SalePrice'].index
cols
```

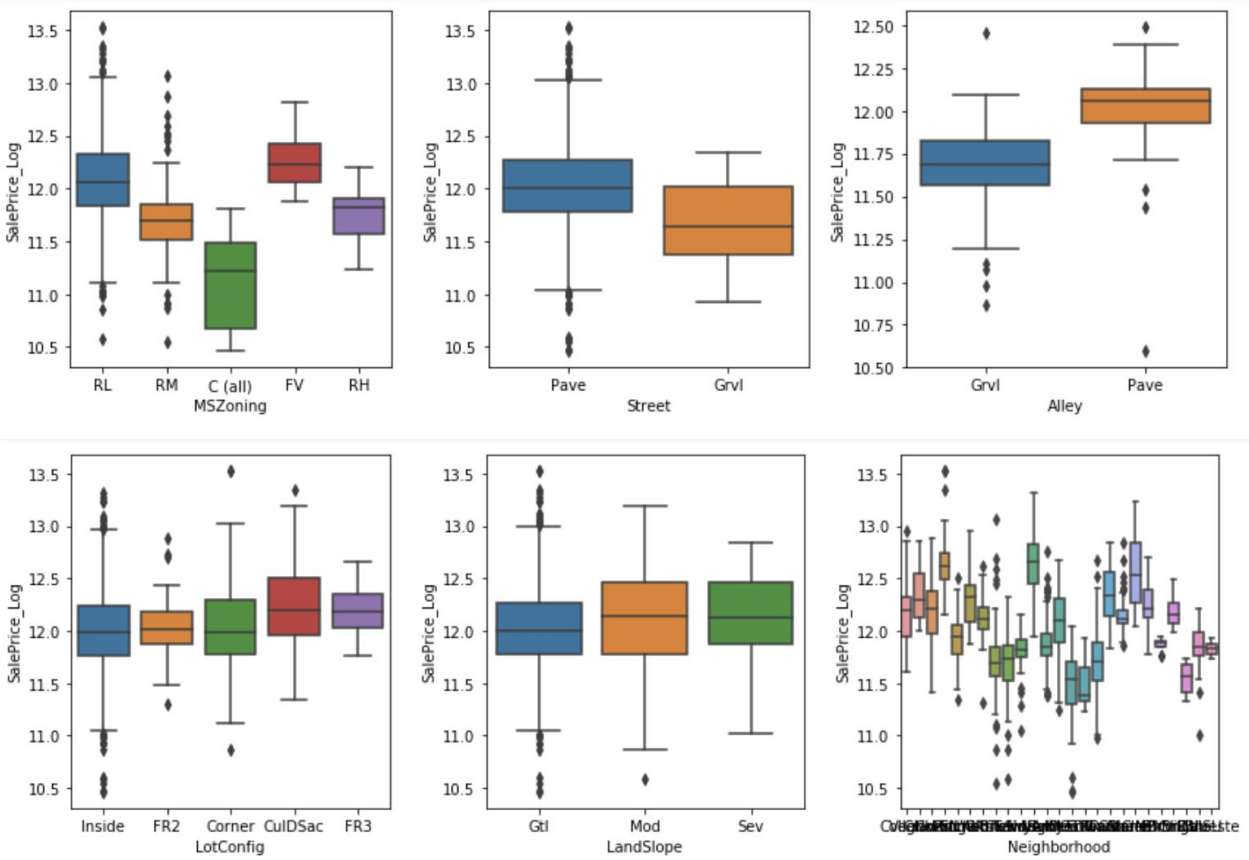
```
Index(['SalePrice', 'SalePrice_Log', 'OverallQual', 'TotalSF', 'GrLivArea',
       'GarageCars', 'GarageArea', 'TotalBsmtSF', '1stFlrSF', 'FullBath',
       'TotRmsAbvGrd'],
      dtype='object')
```



With this matrix, we can conclude that the highly correlated variables to SalePrice are - 'OverallQual', 'TotalSF', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', '1stFlrSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt'

Relationships Between Categorical Features and SalePrice

We utilize a box plots to visualize the correlations between SalePrice and all the Categorical features.



Based on the box plots, we can identify the categorical features that have strong correlation to SalePrice log are - 'MSZoning', 'Neighborhood', 'Condition2', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'ExterQual', 'BsmtQual', 'BsmtCond', 'Heating', 'CentralAir', 'KitchenQual', 'GarageType', 'GarageQual', 'PoolQC', 'MiscFeature', 'SaleType'

Missing Values & Converting Categorical Features to Numeric

For categorical features with NaN, we replace it with None in both train and test data.
For numerica features with NaN, we replace with the mean of the column.

As we are using Scikit Learn for our modeling, we need to encode all categorical features numerically using LabelEncoder. We apply this encoder to both training and test data.

Modeling

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). In this case, the target is SalePrice.

For this project, we'lll adopt several regression analysis algorithms like Linear Regression, Ridge Regression, Lasso Regression and ElasticNet Regression. We'll compare the results and select the best model for predicting SalePrice.

Linear Regression Without Log Transformation of Data

We use train_test_split to split the training data set into training and test set. Without investigating if the data is skewed in anyway.

Results

- The $R2$ is 0.87 which is really close to 1. This shows that the model is close to a perfect fit.
- The Root Mean Squared Error is 0.15 which is close to 0, which means the error is very low.
- Based on the scatter plot, we can also see that the majority of the predicted Saleprice is very close to actual Saleprice.
- It's easy to draw a conclusion here that this is a good model. However, we didn't look at the skewness of that data and we didn't standardize our data. Most models use some sort of distance to inform the prediction. If the data is too wide spread, the model will be misinformed.

Linear Regression With Log Transformation of Data

Method 1 - Identify Data That Needs to Be Standardized

Identify which features are skewed and standardize them using scipy.stats skew. Run Linear Regression again on this new data set that has been standardized.

- The $R2$ is 0.88 which is better than model using the non-standardized data.
- The Root Mean Squared Error is 0.14 which is better than model using the non-standardized data.

Method 2 - Use StandardScaler to Automatically Standardize Data

Create a pipeline with StandardScaler and LinearRegression.

- The $R2$ is 0.86
- The Root Mean Squared Error is 0.15.

Surprisingly, using StandardScaler didn't produce a better result. Method 1 where we identify the numerical features that are skewed and we log them individually resulted in a much improved results.

Linear Regression with Regularization

Other than Linear Regression, we are going to use RidgeCV, LassoCV and ElasticNet CV to see if we can get better results.

RidgeCV

Large coefficients can lead to overfitting, that's why we need to regularize to penalize large coefficients. RidgeCV comes with built in cross-validation.

LassoCV

- Can be used to select important features of a dataset
- Shrinks the coefficients of less important features to exactly 0
- Lasso linear model with iterative fitting along a regularization path
- The best model is selected by cross-validation

ElasticNetCV

- ElasticNet is another regularization model. It tends to select more features than Lasso hence leading to larger models (also more expensive to train) but also be more accurate in general.
- In particular, Lasso is very sensitive to correlation between features and might select randomly one out of 2 very correlated informative features while ElasticNet will be more likely to select both which should lead to a more stable model.

Results from Various Models

	$R2$	**Root Mean Squared Error**
Linear Regression without Standardized Data	0.86707	0.15017
Linear Regression with Standardized Data - Individually transform features that are skewed	0.15017	0.14123
Linear Regression with Standardized Data using StandardScaler()	0.86714	0.15013
RidgeCV	0.88375	0.14043
LassoCV	0.87365	0.14640
ElasticNetCV	0.88295	0.14091

Best Model

- RidgeCV performed better than Linear Regression and LassoCV.
- RidgeCV and ElasticNetCV produced very similar results. ElasticNetCV produced a slightly lower RMSE but a little lower $R2$ than RidgeCV.