**House Price Prediction.**

Monish Joshy

Data Science Practicum l

Regis University

**Purpose of Analysis and Project Overview**

The COVID Pandemic has a massive impact on our lives, many of our loved ones have passed away and almost all countries have now record low GDP and sluggish economy. Investors find not so encouraging return for investment for stocks and bonds, while the record low mortgage rate entice them to invest more on housing market, but the essential building construction materials are not available, so we have very low inventory in many cities resulting in high demand for houses but low supply available, we see unprecedented housing price, many first time home buyers wonders if it is worth to buy a home now, to help the new home buyers take a knowledgeable decision, I have decided to spend time on developing the project on House price Prediction.

**Data Description and Preparation**

The dataset contains the costs and features of residential houses sold from 2006 to 2010 in Ames, Iowa, obtained from the Ames Assessor’s workplace. This dataset consists of seventy-nine house features and 1460 houses with the sold price. Although the dataset is comparatively tiny with solely 1460 examples, it contains seventy-nine features like areas of the homes, types of the floors, and numbers of bathrooms. Such massive amounts of features enable us to explore varied techniques to predict the house costs. The dataset consists of options in varied formats. it's numerical features like prices and numbers of bathrooms, bedrooms, living rooms, as well as categorical features like zone classifications for sale, which can be ‘Agricultural’, ‘Residential High Density’, ‘Residential Low Density’, ‘Residential low-density Park’, etc. in order to form this knowledge with totally different format usable for our algorithms, categorical knowledge will be converted into separated indicator data, which expands the number of features during this dataset. We will be splitting our dataset into a training and testing set with a roughly 80/20 split, with 1000 coaching examples and 460 testing examples. Besides, those features that had values of N/A, we would be replacing them with the mean of their columns so that they don’t influence the distribution. I am planning to complete the project within 5-6 weeks.

**Exploratory Data Analysis (EDA) and Visualizations**

Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data, and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, Python. Before it can conduct analysis on data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are, and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working.

In total there are 1460 training samples in our dataset and 86 different informative features in which thirty-six are quantitative and forty-three are numerical, the rest are “Id” and “SalesPrice” (target column).

The Quantitave Features are 1stFlrSF, 2ndFlrSF, 3SsnPorch, BedroomAbvGr, BsmtFinSF1, BsmtFinSF2, BsmtFullBath, BsmtHalfBath, BsmtUnfSF, EnclosedPorch, Fireplaces, FullBath, GarageArea, GarageCars, GarageYrBlt, GrLivArea, HalfBath, KitchenAbvGr, LotArea, LotFrontage, LowQualFinSF, MSSubClass, MasVnrArea, MiscVal, MoSold, OpenPorchSF, OverallCond, OverallQual, PoolArea, ScreenPorch, TotRmsAbvGrd, TotalBsmtSF, WoodDeckSF, YearBuilt, YearRemodAdd, and YrSold.

The Categorical Features are Alley, BldgType, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, BsmtQual, CentralAir, Condition1, Condition2, Electrical, ExterCond, ExterQual, Exterior1st, Exterior2nd, Fence, FireplaceQu, Foundation, Functional, GarageCond, GarageFinish, GarageQual, GarageType, Heating, HeatingQC, HouseStyle, KitchenQual, LandContour, LandSlope, LotConfig, LotShape, MSZoning, MasVnrType, MiscFeature, Neighborhood, PavedDrive, PoolQC, RoofMatl, RoofStyle, SaleCondition, SaleType, Street, and Utilities.

The table below shows a quantitative estimate of the percentage of missing values present in the column of the features of the dataset. As we can see that the features Alley, PoolQC, Fence and Miscfeature contains almost the entire column as blank so we decide to drop these features as imputing these features with any value would not make any sense.

Table

Description automatically generated with medium confidence

The graph below sums up the table fairly showing the four features as the ones having the maximum percentage of values as null values.

Chart, histogram

Description automatically generated

Fig. 1: Countplot Showing the Number of missing values in each of the features of the dataset.

Following this, I did Univariate and Bivariate Analysis of Categorical and Numerical features separately. We observe that some categories seem to more diverse with respect to SalePrice than others. The neighborhood has a big impact on house prices. The most expensive seems to be Partial SaleCondition. Having a pool on the property seems to improve the price substantially. There are also differences in variabilities between category values.

When I plotted a countplot of features against target variable, I observe that 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'Bedroom', 'Kitchen', 'KitchenQual', 'TotRmsAbvGrd', 'Fireplaces', 'GarageType', 'MiscVal', 'MoSold', 'YrSold' variables are in int64/float64 type(i.e. They were considered as quantitative features), but they can be treated as categorical. The barplots below for these mentioned features imply the reason for the same.

A picture containing crossword puzzle

Description automatically generated

Fig. 2: Countplots of features which were considered quantitative but can be converted to Categorical.

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

The three figures on the left show that the target column which is the SalesPrice does not follow a normal distribution, so before performing regression it has to be transformed. While log transformation does pretty good job, best fit is unbounded Johnson distribution.

Chart, histogram

Description automatically generated

On the left is the graph of Anova Test P values which gives a quick estimate of influence of categorical variables on SalePrice.

For each variable SalePrices are partitioned to distinct sets based on category values. Then check with ANOVA test if sets have similar distributions. If variable has minor impact then set means should be equal. Decreasing pval is sign of increasing diversity in partitions.

Fig. 4 - Anova test P-value graph

Chart

Description automatically generated

Fig. 5: Spearman Correlation values plot of each feature with the “SalesPrice” target feature.

Spearman correlation is better to work with in this case because it picks up relationships between variables even when they are nonlinear. OverallQual is main criterion in establishing house price. Neighborhood has big influence, partially it has some intrisinc value in itself, but also houses in certain regions tend to share same characteristics (confunding) what causes similar valuations.

Chart, box and whisker chart

Description automatically generated

Fig. 6: Relationship between overall quality of the house and its sales price.Chart

Description automatically generated

Fig. 7: Relationship between the year in which the house was built and its sales price.

'OverallQual' and 'YearBuilt' also seem to be related with 'SalePrice'. The relationship seems to be stronger in the case of 'OverallQual', where the box plot shows how sales prices increase with the overall quality.

A screen shot of a computer

Description automatically generated with low confidence

Fig.8: Correlation Heatmap Plot

At first sight, there are two red-colored squares that get my attention. The first one refers to the 'TotalBsmtSF' and '1stFlrSF' variables, and the second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. Actually, this correlation is so strong that it can indicate a situation of multicollinearity. If we think about these variables, we can conclude that they give almost the same information, so multicollinearity really occurs. Heatmaps are great to detect this kind of situations and in problems dominated by feature selection, like ours, they are an essential tool.

Some points to note from the Correlation Plot:

* 'OverallQual', 'GrLivArea' and 'TotalBsmtSF' are strongly correlated with 'SalePrice'.
* 'GarageCars' and 'GarageArea' are also some of the most strongly correlated variables. However, as we discussed in the last sub-point, the number of cars that fit into the garage is a consequence of the garage area. 'GarageCars' and 'GarageArea' are like twin brothers. You'll never be able to distinguish them. Therefore, we just need one of these variables in our analysis (we can keep 'GarageCars' since its correlation with 'SalePrice' is higher).

Scatter plots between 'SalePrice' and correlated variables

Fig. 9: Scatterplot of Numerical features vs SalesPrice VariableA picture containing table

Description automatically generated

This mega scatter plot gives us a reasonable idea about variables relationships.

One of the figures we may find interesting is the one between 'TotalBsmtSF' and 'GrLiveArea'. In this figure, we can see the dots drawing a linear line, which almost acts like a border. It totally makes sense that the majority of the dots stay below that line. Basement areas can be equal to the above-ground living area, but it is not expected a basement area bigger than the above-ground living area.

The plot concerning 'SalePrice' and 'YearBuilt' can also make us think. At the bottom of the 'dots cloud', we see what almost appears to be a shy exponential function (be creative). We can also see this same tendency in the upper limit of the 'dots cloud'. Also, notice how the set of dots regarding the last years tend to stay above this limit.

**Data Preparation Methodology**

From the Feature Description file, we could estimate what we want to fill the ‘nan’ values with, so the following points explain how I have filled each feature and their nan samples with:

* The feature “Functional”’s nan values were filled with “Typ” string
* The feature “Electrical”’s nan values were filled with “SBrkr” string
* The feature “KitchenQual”’s nan values were filled with “TA” string
* The feature “Exterior1st”’s nan values were filled with its mode value
* The feature “Exterior2nd”’s nan values were filled with its mode value
* The feature “SaleType”’s nan values were filled with its mode value
* The feature “PoolQC”’s nan values were filled with “None” value
* The feature “Alley”’s nan values were filled with “None” value
* The feature “FirePlaceQu”’s nan values were filled with “None” value
* The feature “Fence”’s nan values were filled with “None” value
* The feature “MiscFeature”’s nan values were filled with “None” value
* The feature “GarageArea”’s nan values were filled with the value zero (0)
* The feature “GarageCars”’s nan values were filled with the value zero (0)
* The feature “GarageType”’s nan values were filled with “None” value
* The feature “GarageFinish”’s nan values were filled with “None” value
* The feature “GarageQual”’s nan values were filled with “None” value
* The feature “GarageCond”’s nan values were filled with “None” value
* The feature “BsmtQual”’s nan values were filled with “None” value
* The feature “BsmtCond”’s nan values were filled with “None” value
* The feature “BsmtExposure”’s nan values were filled with “None” value
* The feature “BsmtFinType1”’s nan values were filled with “None” value
* The feature “BsmtFinType2”’s nan values were filled with “None” value

## Log Transformation

For those features that are highly skewed, they are normalized using Log transformation.

The following figures show how the log transformation affects the distribution of SalesPrice values.A picture containing icon

Description automatically generated

Fig. 10: Before Log transformation, the graphs shows how the data is skewed.

Fig. 11: After Log transformation, the graphs shows how the data is perfectly normalized.A picture containing icon

Description automatically generated

# **Baseline Models**

We divide the dataset in the ratio 66.67/33.33 into train/validation sets respectively. The model is trained on train dataset and further evaluated on the validation dataset. The evaluation metric used is Root Mean Square Error. The calculation formula for the same is given below:

Shape

Description automatically generated with medium confidence

The table below represents a tabular representation of performances of numerous different regressors that I have implemented.

Graphical user interface, text

Description automatically generated

Fig. 12: Model Performance on the dataset without any feature Engineering or selection

# **Feature Engineering**

Apart from the initial features, I have created four more features that can helpn increase the performance of the model, they are as follows along with their formulas to calculate:

* SqFtPerRoom = train\_test["GrLivArea"] / (train\_test["TotRmsAbvGrd"] +

train\_test["FullBath"] +

train\_test["HalfBath"] +

train\_test["KitchenAbvGr"])

* Total\_Home\_Quality = train\_test['OverallQual'] + train\_test['OverallCond']
* Total\_Bathrooms = (train\_test['FullBath'] + (0.5 \* train\_test['HalfBath']) +

train\_test['BsmtFullBath'] + (0.5 \* train\_test['BsmtHalfBath']))

* HighQualSF = train\_test["1stFlrSF"] + train\_test["2ndFlrSF"]

# **Feature Selection**

After visualizing the baseline model performance without any feature selection or generation, nor any hyperparameter tuning, I decided to do some feature generation and generated four new features that could increase the performance of the model.

Now, in order to do feature selection, we need feature importance first to do see how the algorithm of the model is performing and which features are more important than the others.

I implemented a CatBoost Regressor and fitted it on the train dataset. Following this I could easily get the feature importance of each feature for the model. The barplot below shows a quantitative estimate of feature importance of top twenty features.

Chart, funnel chart

Description automatically generatedFig. 13 - Feature Importance plot of top twenty features w.r.t the CatBoost Regressor.

Catboost comes with a great method: get\_feature\_importance. This method can be used to find important interactions among features. This is a huge advantage because it can give us insights about possible new features to create that can improve the performance.

Graphical user interface

Description automatically generated with medium confidence

Fig. 14 - Feature Interaction Levels

The table on the right shows the interaction level between two features that can be used to generate new features or to remove certain features if they are almost equivalent to each other.

# 

# 

# 

# **Hyper Parameter Tuning and Results**

In order to achieve the best performance of the model, we need to perform HyperParameter tuning of various different hyperparameter options that are given to us by Cat Boost Regressor.

In order to do this, we do a Random Grid Search in order to get the best combination of parameters, the parameters that we tune are “iterations”, which are used to specify the number of epochs the model will iterate over the training set, “learning\_rate”, “depth” which implies the depth of decision tree that will be formed and “l2\_leaf\_reg” which is the L2 regularization term of the cost function.

The following are the best set of parameters obtained from the Grid Search

* Iterations - 6000
* Learning rate - 0.005
* Depth - 4
* L2\_leaf\_reg - 1
* Evaluation\_metric - RMSE (Root Mean Square Error)
* Verbose - 200
* Early Stopping Rounds - 200

After implementing these Hyperparameters with CatBoost Regressor along with the features generated through feature Engineering, we were successfully able to achieve an RMSE score of 0.1098 on the train set and 0.0553 on the test set. Thus, we are successfully able to build a model that accurately predicts the prices of houses provided we know its feature values.

**References**

Lantz, B. (2015) *Machine Learning with R: Discover How to Build Machine Learning Algorithms, Prepare Data, and Dig Deep into Data Prediction Techniques with R*. Packt Publishing.

Nwanganga, F., Chapple, M. (2020) *Practical Machine Learning in R*. Wiley.

Yu-Wei, C. (2015).  *Machine Learning with R Cookbook Explore over 110 Recipes to Analyze Data and Build Predictive Models with the Simple and Easy-to-Use R Code*. Packt Publishing.