Kaggle Project: House prices – Advanced regression technique

Introduction:

The project is all about predicting the prices of real estates on the basis of some features like lot size in square feet, general shape of property and many others. The objective is to give a real valued data that is nothing but the price of the houses. This is clearly a supervised learning problem and at the same time, also a regression task. Now, the key to this is the proper study and research of input data (do some data pre-processing if required) followed by fitting some model that works well both on the training data set and test dataset sufficiently well. The key thing is to note that the model should not over-fit. Various models can be used in this regard. What we have done is that we used some very common regression technique like linear regression, Random forest regression and XGBoost to predict house prices and then we have compared the results.

The tasks involved are as following:

- Exploratory data analysis
- Feature engineering
- Feature selection
- Splitting training data in ratio of 80:20 as training data and validation data
- Building machine learning models
- Checking RMSE on validation data
- Hyperparameters tuning

Dataset:

There are total 81 columns and 1460 rows. List of columns with description is added in another text file.

1) EDA:

a) Checking missing values:

Total 18 features have null values. Here percentage of null values are printed for the features having null values.

LotFrontage has 17.74 % missing values

Alley has 93.77 % missing values

MasVnrType has 0.55 % missing values

MasVnrArea has 0.55 % missing values

BsmtQual has 2.53 % missing values

BsmtCond has 2.53% missing values

BsmtExposure has 2.6 % missing values

BsmtFinType1 has 2.53 % missing values

BsmtFinType2 has 2.6 % missing values

FireplaceQu has 47.26 % missing values

GarageType has 5.55 % missing values

GarageYrBlt has 5.55 % missing values

GarageFinish has 5.55 % missing values

GarageQual has 5.55 % missing values

GarageCond has 5.55 % missing values

PoolQC has 99.52 % missing values

Fence has 80.75 % missing values

MiscFeature has 96.3 % missing values

To check the significance of missing values, relation between missing values and median of sale price is checked. Graphs are plotted in the file with code. From the graphs, relation between missing values and dependent variable is clearly visible. So, we need to replace these nan values with something meaningful which is done in feature engineering section.

b) Checking numerical features:

There are total 38 numerical variables. These numerical variables are of three types as continuous variables, discrete variables and temporal (Date time) variables.

There are 17 discrete variables and 16 continuous variables. Relations between the numeric variables and dependent variables are studied. Distribution of numerical variables is also studied. As some features are skewed, their relation with dependent variables is checked by taking logarithmic transformation. Outliers are also found. Graphs are plotted in file with the code.

From the Dataset we have 4-year variables. We have to extract information from the datetime variables like no of years or no of days. One example in this specific scenario can be difference in years between the year the house was built and the year the house was sold. From analysis it can be seen that as year sold is increasing sale price is decreasing but in practical

scenario situation is exactly opposite. So, we need to consider effects of other features as well and do proper analysis. We will be performing this analysis in the Feature Engineering section.

c) Checking categorical features:

There are total 43 categorical variables. These categorical variables are of two types as nominal variables and ordinal variables. Here number of categories in each variable is found. Relation between categorical variables and dependent variables are studied. Graphs are plotted.

2) Feature Engineering:

a) Handling missing values

There are 18 features which have missing values. Out of which 15 are categorical features and 3 are numerical features. So, missing or NAN values in categorical features are replaced with a new category called “Missing”. Missing values in numerical features can be replaced by mean or median but in this case, there are some outliers. So, to avoid their effect, missing values are replaced by their median value. Just to keep a note of this, extra 3 columns are added which shows the places where missing values were there.

2) Handling skewness of numerical features:

In numerical features, there are many skewed features. Here we are converting only those features which don’t have 0 as a value to log normal distribution.

3) Handling rare categorical features:

We will remove categorical variables which are less than 1% of total observations. So those all categorical variables are labelled as Rare_var (i.e. one feature variable instead of all)

4) Handling temporal variables:

As it is seen that sale price is decreasing as year sold is increasing. To handle this situation, other variables are combines with year sold and then their relation with dependent variable is checked.

5) Other operations:

- drop irrelevant columns
- drop columns with more than 50 percent of null values
- One hot encoding is performed on ordinal categorical variables
- Label encoding is performed on nominal categorical variables

6) Feature scaling:

Feature scaling is done by using MinMaxScalar of Scikit-learn library

3) Feature Selection:

- Feature selection is done by using Lasso and SelectFromModel tools of scikit-learn library.

- Some features added by observation for e.g. independent features having correlation more than 0.45 and less than -0.45 with dependent variable are added.

List of various approaches used from the start to the final approach:

1) Linear Regression:

First, I split the training dataset in two groups as training and validation. 20 percent of the data was separated. Then I trained Linear Regression model on training dataset. Here I tried to minimize RMSE on validation data. To reduce RMSE, I trained model on changing features.

Eventually, I got minimum RMSE on validation data as 0.17 and on testing data as 0.25.

2) Random Forest Regressor:

First, I split the training dataset in two groups as training and validation. 20 percent of the data was separated. Then I trained Random forest regressor model on training dataset. Here I tried to minimize RMSE on validation data. To reduce RMSE, I had to find best hyperparameters. These hyperparameters are found by using RandomizedSearchCV. Then I followed the same steps by changing some features.

Eventually, I got min RMSE on validation data as 0.138 and on testing data as 0.286.

Optimized hyperparameters:

n_estimators = 1000,

min_samples_split = 2,

min_samples_leaf = 1,

max_features = 'sqrt',

max_depth = 110.

3) XGBoost Regressor:

First, I split the training dataset in two groups as training and validation. 20 percent of the data was separated. Then I trained XGBoost regressor model on training dataset. Here I tried to minimize RMSE on validation data. To reduce RMSE, I had to find best

hyperparameters. These hyperparameters are found by using RandomizedSearchCV. Then I followed the same steps by changing some features.

Eventually, I got min RMSE on validation data as 0.14 and on testing data as 0.24.

Optimized hyperparameters:

n_estimators: 100,

min_child_weight: 3,

max_depth: 10,

learning_rate: 0.1,

booster: 'gbtree'.

Results and Conclusion:

Model	**RMSE (Training)**	**RMSE (Validation)**	**RMSE (Test)**
Linear Regression	0.1412	0.1737	0.2585
Random Forest Regressor	0.0726	0.1374	0.2861
XGBoost Regressor	0.034	0.1448	0.2497

- On training dataset, RMSE values are like this:

XGBoost Regressor < Random Forest Regressor < Linear Regression

- On validation dataset, RMSE values are like this:

Random Forest Regressor < XGBoost Regressor < Linear Regression

- On test dataset, RMSE values are like this:

XGBoost Regressor < Linear Regression < Random Forest Regressor

- From above results, it can be concluded that **XGBoost** regressor gave the **minimum RMSE**. Hence, this model is accepted for the prediction purpose.