

Bengaluru House Price Prediction Machine Learning Project

Abstract

Property Technology (PropTech) is the next big thing that is going to disrupt the real estate market. Nowadays, we see applications of Machine Learning (ML) and Artificial Intelligence (AI) in almost all the domains but for a long time the real estate industry was quite slow in adopting data science and machine learning for problem solving and improving their processes. However, things are changing quite fast as we see a lot of adoption of AI and ML in the US and European real estate markets. But the Indian real estate market has to catch-up a lot. This project proposes a machine learning approach for solving the house price prediction problem in the classified advertisements. This project focuses on the Indian real estate market. Predictive models for determining the sale price of houses in cities like Bengaluru is still remaining as more challenging and tricky task. The sale price of properties in cities like Bengaluru depends on a number of interdependent factors. Key factors that might affect the price include area of the property, location of the property and its amenities. In this research work, an analytical study has been carried out by considering the data set that remains open to the public by illustrating the available housing properties in machine hackathon platform. The data set has nine features. In this study, an attempt has been made to construct a predictive model for evaluating the price based on the factors that affect the price. Here, the attempt is to construct a predictive model for evaluating the price based on factors that affects the price.

Introduction

Modeling uses machine learning algorithms, where machine learns from the data and uses them to predict a new data. The most frequently used model for predictive analysis is regression. As we know, the proposed model for accurately predicting future outcomes has applications in economics, business, banking sector, healthcare industry, e-commerce, entertainment, sports etc. One such method used to forecast house prices are based on multiple factors. In metropolitan cities like Bengaluru, the prospective home buyer considers several factors such as location, size of the land, proximity to parks, schools, hospitals, power generation facilities, and most importantly the house price. Multiple linear regression is one of the statistical techniques for assessing the relationship between the (dependent) target variable and several independent variables. Regression techniques are widely used to build a model based on several factors to predict price. In this project, we have made an attempt to build house price prediction regression model for data set that remains accessible to the public in Kaggle platform. We have considered almost all regression model. A comparative study was carried out with evaluation metrics as well. Once we get a good fit, we can use the model to forecast monetary value of that particular housing property in Bengaluru. The project is divided into the following sections: Section 2 explains the description of the data set used, pre-processing of data, Data cleaning, Outlier removal, Feature engineering and exploratory analysis of data before regression model is built. Section 3 - Data processing presents a summary of the regression models developed in the comparison study and the

evaluation metrics is used. Section 4-Model deployment using Streamlit Section 5 concludes with the future scope of the proposed work. Section 6 lists the applicability of the model.

Data Description and Pre-Processing

Data Description

The train and test data will consist of various features that describe that property in Bengaluru. This is an actual data set that is curated over months of primary & secondary research by our team. Each row contains fixed size object of features. There are 9 features and each feature can be accessed by its name.

Features:

- Area_type – describes the area
- Availability – when it can be possessed or when it is ready (categorical and time-series)
- Location – where it is located in Bengaluru
- Price – Value of the property in lakhs (INR)
- Size – in BHK or Bedroom (1-10 or more)
- Society – to which society it belongs
- Total_sqft – size of the property in sq. ft
- Bath – No. of bathrooms
- Balcony – No. of the balcony

With 9 features available, we try to build regression models to predict house price. We predicted the price of test data set with the regression models built on train data set.

Pre-processing

Setting up Environment and importing libraries

```
!pip install pycaret
!pip install pycaret AutoViz
import pandas as pd
import numpy as np
import pycaret
import autoviz
import pandas_profiling
from pycaret.regression import *
```

The first step of any machine learning experiment in **PyCaret** is setting up the environment by importing the required module and initializing `setup()`. The module used in this project is `pycaret.regression`. **Auto Viz** is a python library which can automate the whole process of data visualization in just a single line of code. **NumPy** is the core library for scientific computing in Python. It provides a high-performance multidimensional array object and tools for working with these arrays. It is a powerful N-dimensional array object which is Linear algebra for

Python. **Pandas** is an open-source library built on top of NumPy providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. It allows for fast analysis and data cleaning and preparation. It excels in performance and productivity. **Pandas profiling** is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code.

Loading the dataset

```
df1=pd.read_csv("/content/Bengaluru_House_Data.csv")
df1
```

CSV should be uploaded in google colab. Here Bengaluru House dataset is taken from Kaggle.

Removing Duplicates

```
df2=pd.DataFrame.drop_duplicates(df1)
```

Data understanding and basic EDA

The purpose is to create a model that can estimate housing prices. We divide the set of data into functions and target variable. In this section, we will try to understand overview of original data set, with its original features and then we will make an exploratory analysis of the data set and attempt to get useful observations. The train data set consists of 11200 records with 9 explanatory variables.

Pandas-Profiling

1. Identify the target column
2. Identify the problem (Either it is regression, Classification or Clustering)
3. Using pandas profiling analyse data deeply

```
report=pandas_profiling.ProfileReport(df2)
```

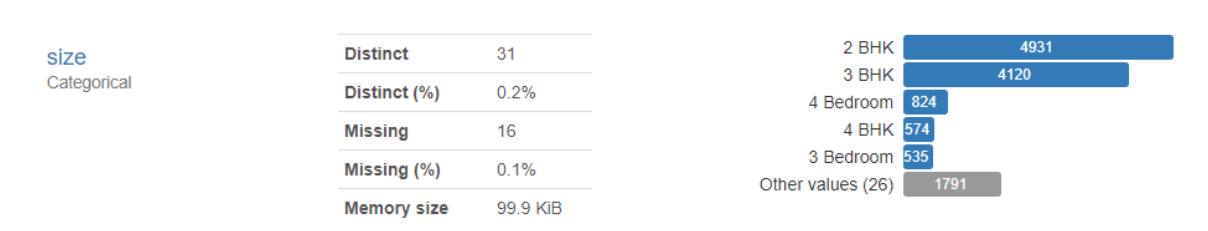
One of the strong points of the generated report are the warnings that appear at the beginning. It tells us the variables that contain missing values, variables with many zeros, categorical variables with high cardinality, etc.

Warnings

availability has a high cardinality: 81 distinct values	High cardinality
location has a high cardinality: 1305 distinct values	High cardinality
society has a high cardinality: 2688 distinct values	High cardinality
total_sqft has a high cardinality: 2117 distinct values	High cardinality
society has 5328 (41.7%) missing values	Missing
balcony has 605 (4.7%) missing values	Missing
df_index has unique values	Unique

Data Cleaning

Size



Size column has really weird values. size feature shows the number of rooms in size feature we assume that 2 BHK = 2 Bedroom == 2 RK so takes only number and remove suffix text. Add new feature(integer) for bhk (Bedrooms Hall Kitchen)

```
df2[['bhk']] = df2['size'].str[:1]
```

Total_sqft

```
df2.total_sqft.unique()

array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

Here we are seeing some alphanumeric value also. So now i am going to make a function which will take the mean when it see range and ignore the alphanumeric value. There are some range values and values in diff format like sq. meter, Perch etc fetch values which are not integers.

```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True

df2[~df2['total_sqft'].apply(is_float)].head()
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price	bhk
30	Super built-up Area	19-Dec	Yelahanka	4	LedorSa	2100 - 2850	4.0	0.0	186.000	4.0
56	Built-up Area	20-Feb	Devanahalli	4	BrereAt	3010 - 3410	NaN	NaN	192.000	4.0
81	Built-up Area	18-Oct	Hennur Road	4	Gollela	2957 - 3450	NaN	NaN	224.500	4.0

Above shows that total_sqft can be a range (e.g. 2100-2850). For such case best strategy is to convert it into number by splitting it and we can take average of min and max value in the range. There are other cases such as 34.46Sq. Meter which I am going to just drop such corner cases to keep things simple.

Function to find average

```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

```
df3 = df2.copy()
df3.total_sqft = df3.total_sqft.apply(convert_sqft_to_num)
df3 = df3[df3.total_sqft.notnull()]
df3.head(50)
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price	bhk
0	Super built-up Area	19-Dec	Electronic City Phase II	2	Coomee	1056.00	2.0	1.0	39.07	2.0
1	Plot Area	Ready To Move	Chikka Tirupathi	4	Theanmp	2600.00	5.0	3.0	120.00	4.0

Scatter plots, Pair plots & Heatmap before outlier removal

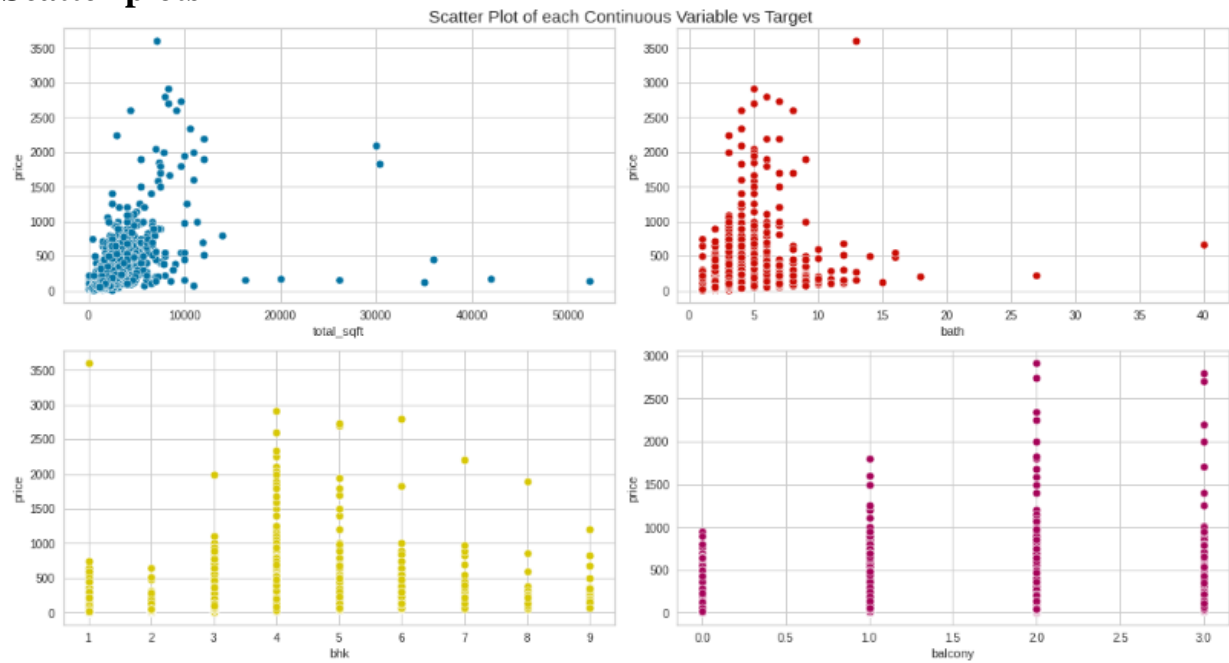
```
#AutoViz initialization
from autoviz.AutoViz_Class import AutoViz_Class
#Instantiate the AutoViz class
AV=AutoViz_Class()
```

Some other arguments which we can pass while calling AutoViz are:

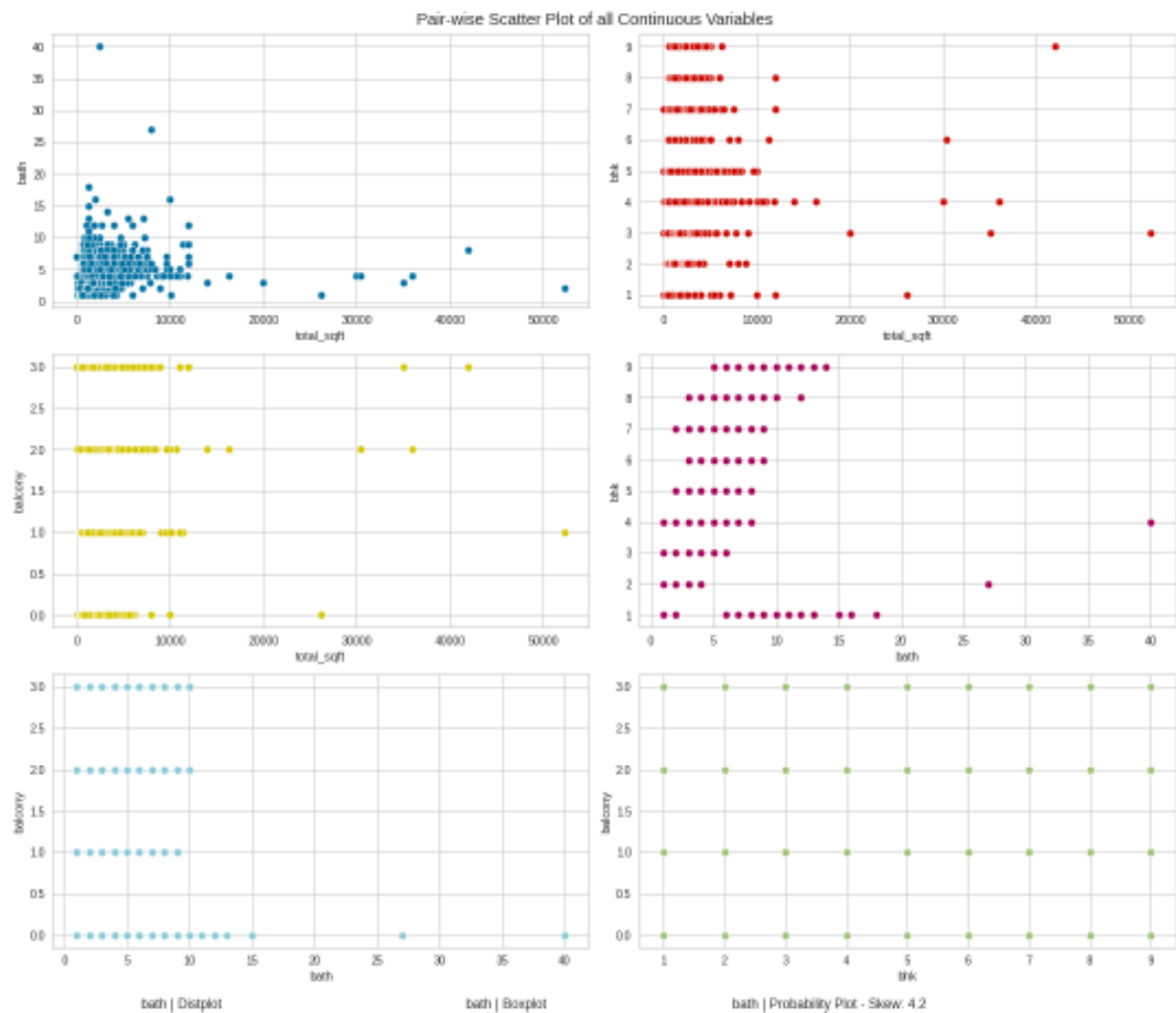
1. sep, which is the separator by which data is separated, by default it is ','.
2. target, which is the target variable in the dataset.
3. chart_format is the format of the chart displayed.
4. max_row_analyzed is used to define the number rows to be analysed
5. max_cols_analyzed is used to define the number of columns to be analysed.

```
df_auto=df4.drop(['area_type', 'availability', 'society', 'price_per_sqft', 'size'], axis=1)
AV.AutoViz(filename='', sep=',', depVar='price', dfte=df_auto)
```

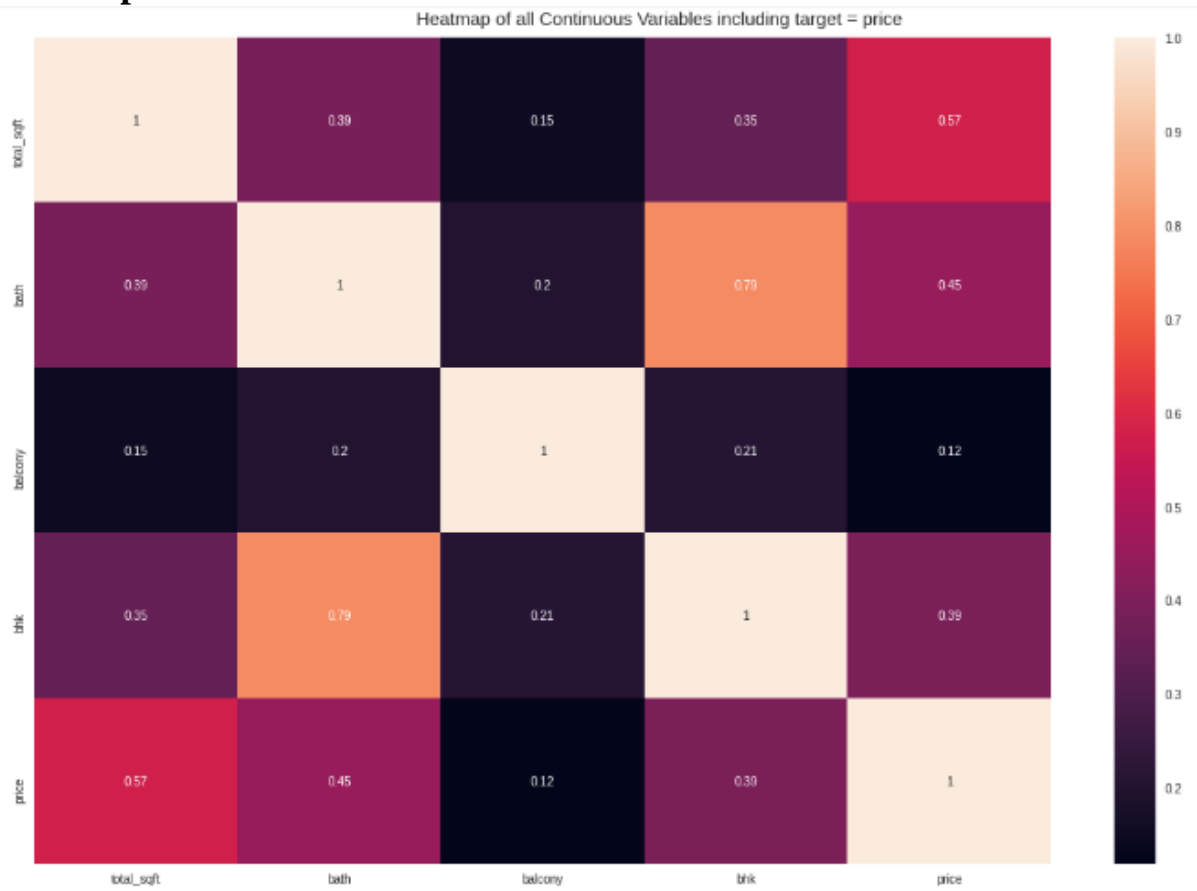
Scatter plots



Pair plots



Heatmap



Feature Engineering and Outlier Removal

Adding price per sqft to detect outliers Add new feature called price per square feet.

```
df4 = df3.copy()
df4['price_per_sqft'] = df4['price']*100000/df4['total_sqft']
df4.head()
```

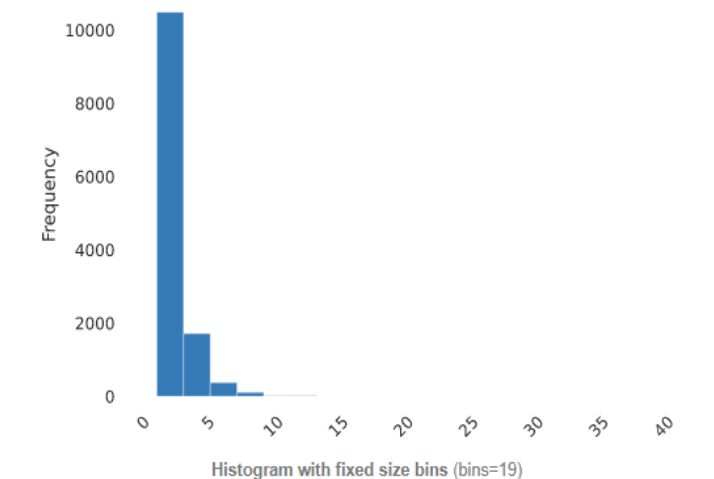
	area_type	availability	location	size	society	total_sqft	bath	balcony	price	bhk	price_per_sqft
0	Super built-up Area	19-Dec	Electronic City Phase II	2	Coomee	1056.0	2.0	1.0	39.07	2.0	3699.810606
1	Plot Area	Ready To Move	Chikka Tirupathi	4	Theanmp	2600.0	5.0	3.0	120.00	4.0	4615.384615

Location

Dimensionality Reduction Any location having less than 10 data points should be tagged as "other" location. This way number of categories can be reduced by huge amount.

```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Bath



Quantile statistics

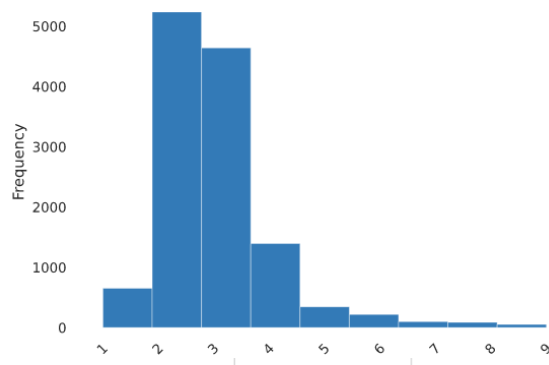
Minimum	1
5-th percentile	1
Q1	2
median	2
Q3	3
95-th percentile	5
Maximum	40
Range	39
Interquartile range (IQR)	1

It could be seen that distribution is highly skewed. It ranges from 1 to 40

Value	Count	Frequency (%)	
2	6531	51.3%	<div></div>
3	3169	24.9%	<div></div>
4	1194	9.4%	<div></div>
1	755	5.9%	<div></div>
5	517	4.1%	<div></div>
6	267	2.1%	<div></div>
7	102	0.8%	<div></div>
8	64	0.5%	<div></div>
9	40	0.3%	<div></div>
10	13	0.1%	<div></div>
Other values (9)	20	0.2%	<div></div>

Around 98% of data lies below 6

BHK



Value	Count	Frequency (%)	
2	5236	41.1%	<div></div>
3	4641	36.5%	<div></div>
4	1394	11.0%	<div></div>
1	653	5.1%	<div></div>
5	346	2.7%	<div></div>
6	220	1.7%	<div></div>
7	99	0.8%	<div></div>
8	88	0.7%	<div></div>
9	52	0.4%	<div></div>

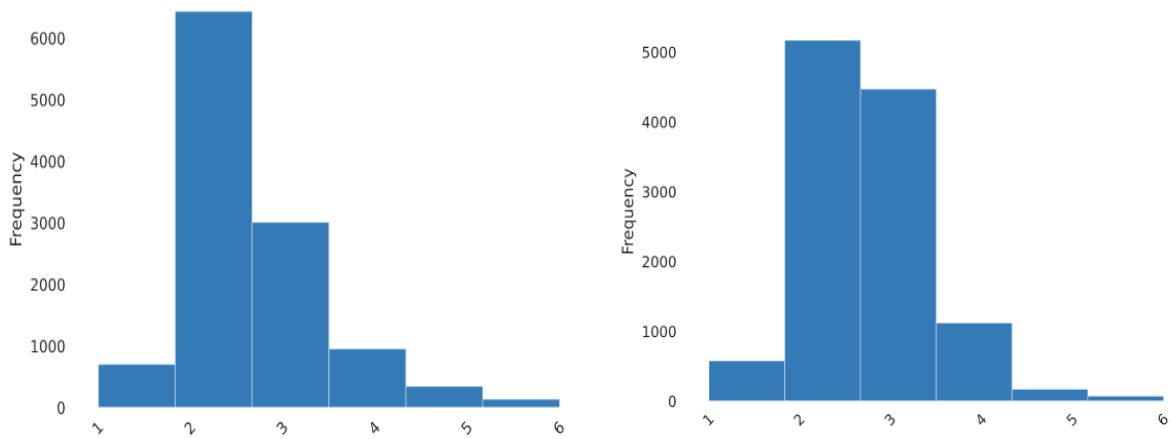
Values ranges from 1 to 9

Around 98% of data lies below 6

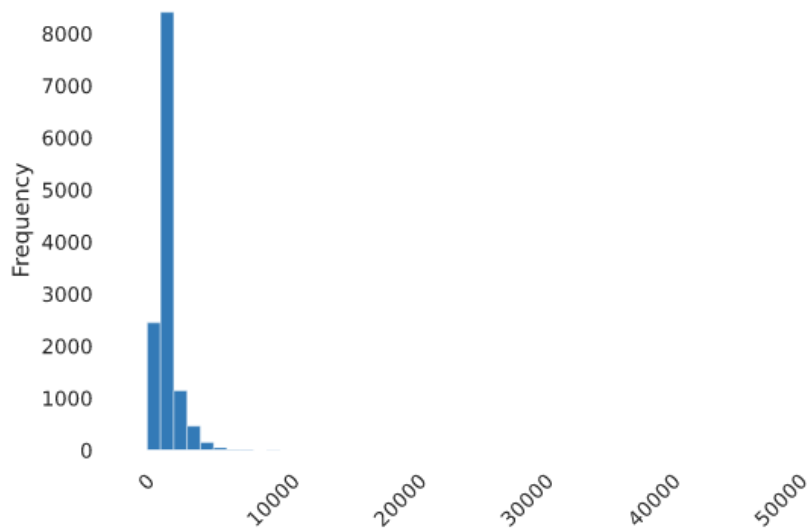
So its better to filter the data by above conditions

```
df7=df6[df6.bath<7]
df8=df7[df7.bhk<7]
```

Histogram plot after outlier removal



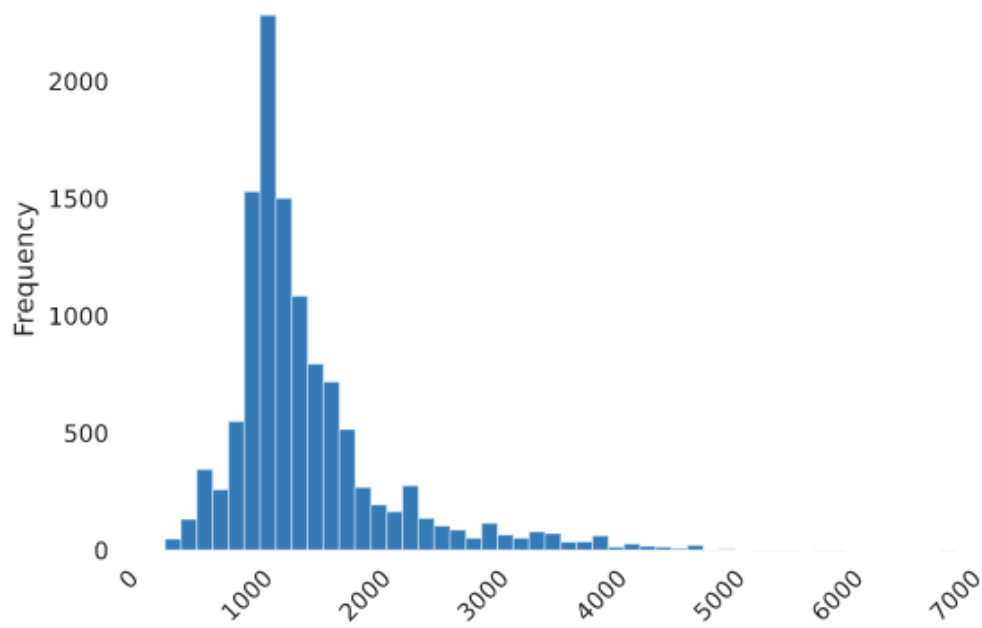
Sq. Ft



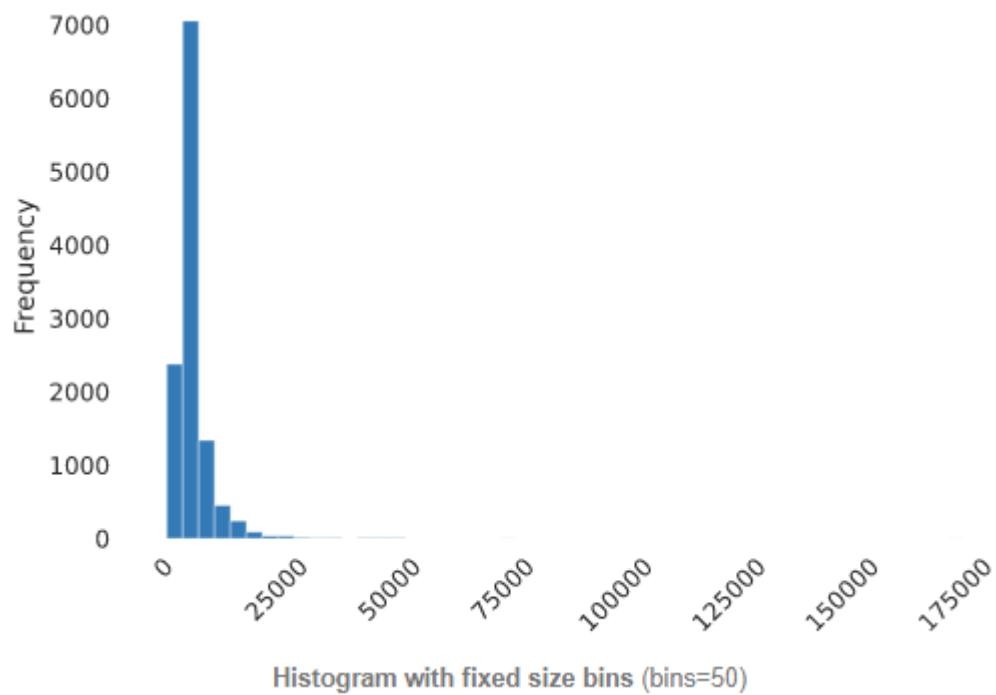
Outlier Removal Using Business Logic As a data scientist when you have a conversation with a person who has expertise in real estate, he will tell you that normally square ft per bedroom is 300 (i.e. 2 bhk apartment is minimum 600 sqft. If you have for example 400 sqft apartment with 2 bhk than that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum threshold per bhk to be 300 sqft. We can set a max sqft per bhk as 1200(4 times the min.)

```
df5 = df4[~(df4.total_sqft/df4.bhk<300)]  
df6 = df5[~(df5.total_sqft/df5.bhk>1200)]
```

Histogram after outlier removal



Price per sqft



Histogram with fixed size bins (bins=50)

Quantile statistics

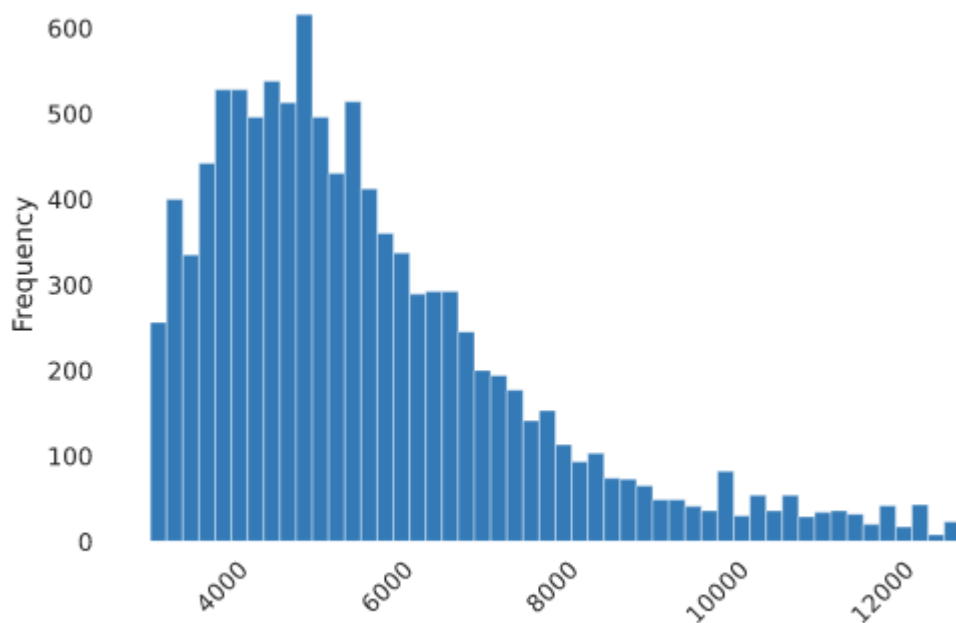
Minimum	1538.461538
5-th percentile	3119.953488
Q1	4239.24643
median	5300
Q3	6867.967674
95-th percentile	12962.96296
Maximum	176470.5882
Range	174932.1267
Interquartile range (IQR)	2628.721243

Price per sqft min is 1538.46 and max is 176470, which are impossible for generic model. Outlier Removal Using Business Logic A when you have a conversation with a real estate expert, he said normal price per square ft is 1500. If you have for example of price per sqft less than that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping a range between 5th and 95th percentile

```
df11=df10.loc[df10['price_per_sqft'] >=3119.953488]
```

```
df12=df11.loc[df11['price_per_sqft'] <=12962.96296]
```

Histogram after outlier removal



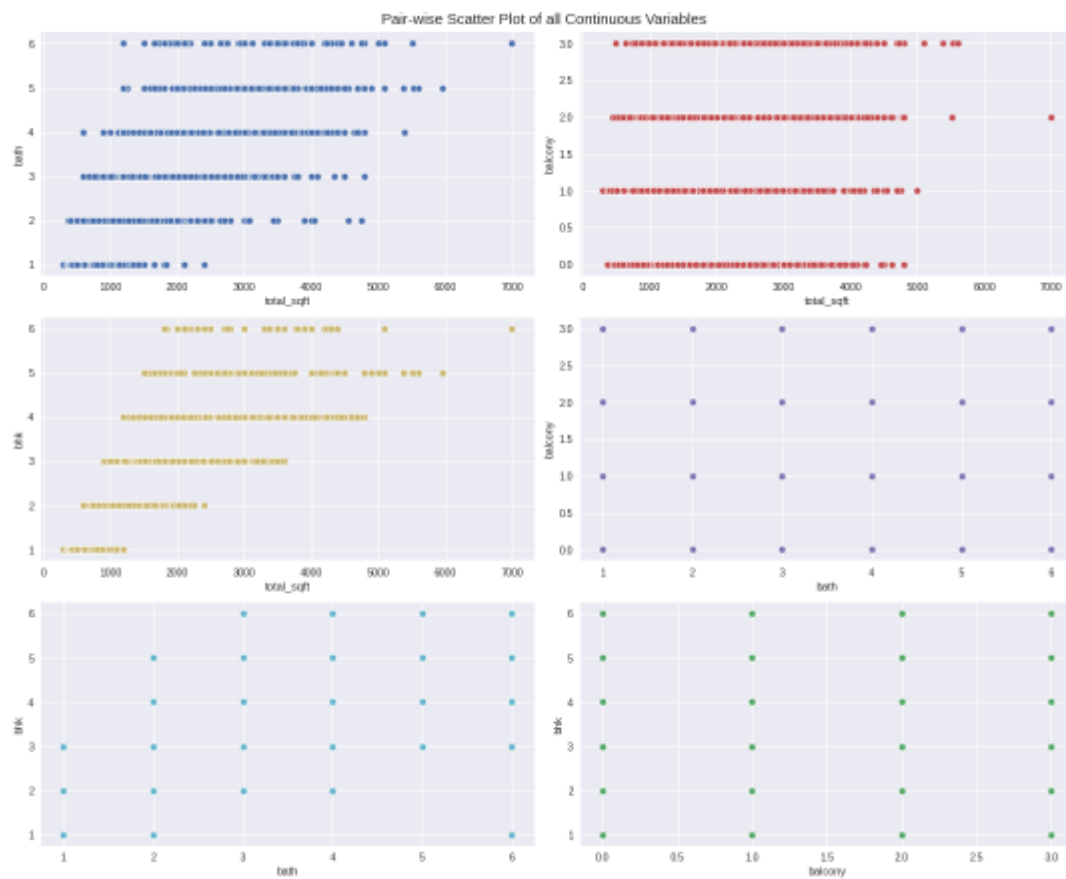
Histogram with fixed size bins (bins=50)

Scatter plots, Pair plots & Heatmap after outlier removal

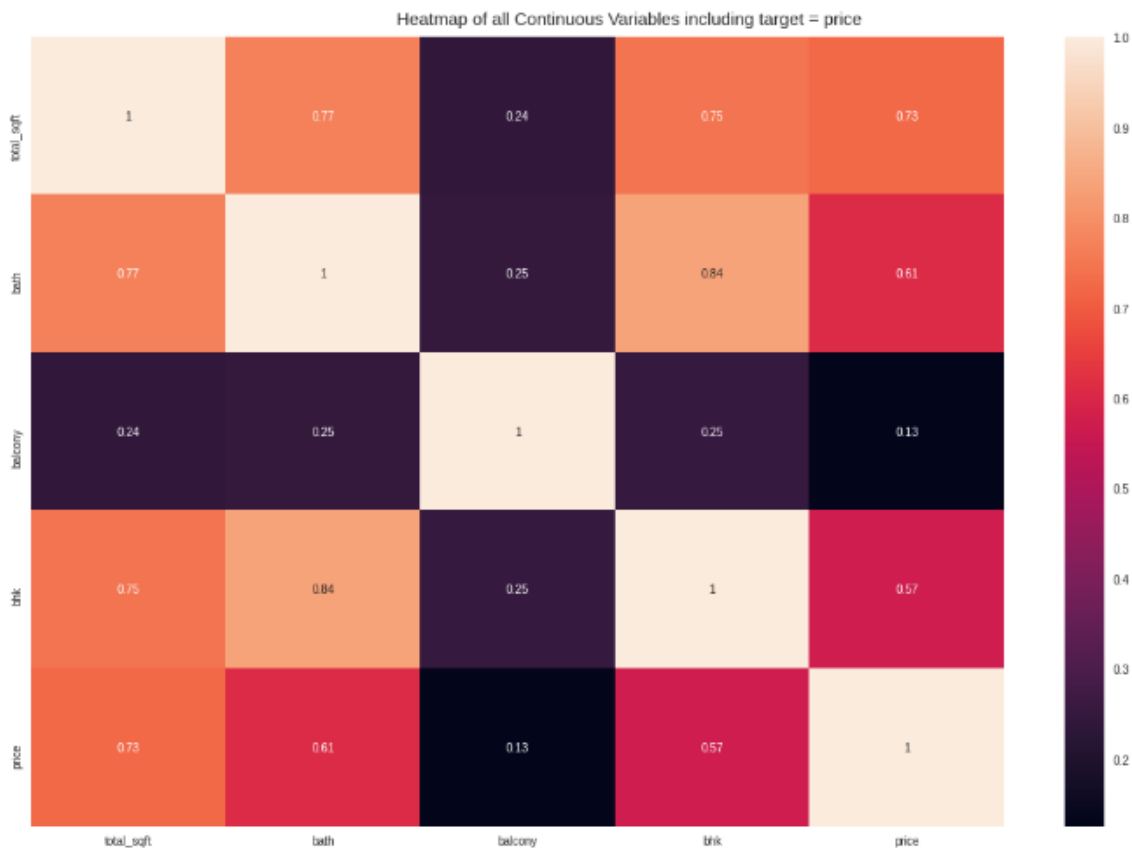
Scatter plots



Pair plots



Heatmap



From the heatmap 'balcony' feature seems to be low correlated with the 'price' and by seeking advice from real estate expert's 'balcony' seems to be irrelevant when compared to others for the price prediction.

The data pre-processing of each feature in data sets is summarized as below:

1. Around 41% of society records are missing in the data set and the features society, availability, area type is dropped the data sets as it doesn't add much to the model.
2. There are around 1298 different locations. Dimensionality Reduction Any location having less than 10 data points should be tagged as "other" location. This way number of categories can be reduced by huge amount. Since the feature location is categorical.
3. In total_sqft we are seeing some alphanumeric values, values denoting the range (with min-max). So now i am going to make a function which will take the mean when we see range and ignore the alphanumeric value. There are some range values and values in diff format like sq. meter, Perch etc fetch values which are not integers.
4. As the 98% of data in features bath and bhk lies less than 6 we just filter out the data by applying these conditions.

5. We observe that, all total-sqft records are not in square-feet in both the data sets. Some of them are in square-yards, acres, perch and grounds. Every data point with respect to total-sqft has been converted into square-feet by carrying out necessary transformations.
6. Adding price per sqft to detect outliers Add new feature called price per square feet.
7. Outlier Removal Using Business Logic As a data scientist when you have a conversation with a person who has expertise in real estate, he will tell you that normally square ft per bedroom is 300. We will remove such outliers by keeping our minimum threshold per bhk to be 300 sqft. We can set a max sqft per bhk as 1200(4 times the min.).
8. Outlier Removal Using Business Logic when you have a conversation with a real estate expert, he said normal price per square ft is 1500.

All these data pre-processing steps have been carried out in colab notebook python with necessary packages.

Data allocation for validation

After getting pre-processed data. The next step is to separate data as train and test or unseen or seen. Unseen data are taken for prediction and seen data are used for training and creating model.

```
data = df12.sample(frac=0.95, random_state=786)
data_unseen = df12.drop(data.index)
data.reset_index(inplace=True, drop=True)
data_unseen.reset_index(inplace=True, drop=True)
print('Data for Modeling: ' + str(data.shape))
print('Unseen Data For Predictions: ' + str(data_unseen.shape))
```

Data for Modeling: (9899, 11)
Unseen Data For Predictions: (521, 11)

The unseen data is saved for final prediction and is used in front end section

```
data_unseen.to_csv('/content/unseen_bengaluru.csv')
```

The data is uploaded in colab and it should be downloaded from google colab.

Initializing the setup

Common to all modules in PyCaret, setup is the first and the only mandatory step to start any machine learning experiment. Besides performing some basic processing tasks by default, PyCaret also offers wide array of pre-processing features which structurally elevates an average machine learning experiment to an advanced solution. Listed below are the essential default tasks performed by PyCaret when you initialize the setup:

We have to define the target variable for prediction, we should normalize the values in different columns to prevent biasing to only one feature, ignore some features which is non-essential for our data and enable remove outliers, high cardinality features and remove multi collinearity.

```
clf=setup(data=data,target='price(in lakhs)',normalize=True,normlize_method='minmax',remove_multicollinearity=True,ignore_features=['price_per_sqft','availability','society','area_type','balcony','size'],remove_outliers=True,high_cardinality_features=['location'])
```

Data Type Inference: Any experiment performed in PyCaret begins with determining the correct data types for all features. The setup function performs essential inferences about the data and performs several downstream tasks such as ignoring ID and Date columns, categorical encoding, missing values imputation based on the data type inferred by PyCaret's internal algorithm. Once the setup is executed a dialogue box appears with the list of all the features and their inferred data types. Data type inferences are usually correct but once the dialogue box appears, user should review the list for accuracy. If all the data types are inferred correctly you may press enter to continue or if not you may type '**quit**' to stop the experiment.

Data Type	
location	Categorical
total_sqft	Numeric
bath	Numeric
bhk	Numeric
price(in lakhs)	Label

Regression Models and Evaluation Metrics Used

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which a researcher finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For example, the method of ordinary least squares computes the unique line (or hyperplane) that minimizes the sum of squared differences between the true data and that line (or hyperplane).

Regression analysis is primarily used for two conceptually distinct purposes. First, regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Second, in some situations regression analysis can be used to infer causal relationships between the independent and dependent variables. Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using observational data.

In practice, researchers first select a model they would like to estimate and then use their chosen method (e.g., ordinary least squares) to estimate the parameters of that model. Regression models involve the following components:

- The **unknown parameters**, often denoted as a scalar or vector β .
- The **independent variables**, which are observed in data and are often denoted as vector x (where x_i denotes a row of data).
- The **dependent variable**, which are observed in data and often denoted using the scalar y .
- The **error terms**, which are not directly observed in data and are often denoted using the scalar ϵ .

In various fields of application, different terminologies are used in place of dependent and independent variables.

Most regression models propose that y is a function of x and ϵ , with ϵ representing an additive error term that may stand in for un-modelled determinants of y or random statistical noise.

The researchers' goal is to estimate the function that most closely fits the data. To carry out regression analysis, the form of the function must be specified. Sometimes the form of this function is based on knowledge about the relationship between y and x that does not rely on the data. If no such knowledge is available, a flexible or convenient form for f is chosen. For example, a simple univariate regression may propose $f(x) = \beta_0 + \beta_1 x$, suggesting that the researcher believes to be a reasonable approximation for the statistical process generating the data.

All available regression models are as follows:

models()			
ID	Name	Reference	Turbo
lr	Linear Regression	sklearn.linear_model_base.LinearRegression	True
lasso	Lasso Regression	sklearn.linear_model_coordinate_descent.Lasso	True
ridge	Ridge Regression	sklearn.linear_model_ridge.Ridge	True
en	Elastic Net	sklearn.linear_model_coordinate_descent.Elast...	True
lar	Least Angle Regression	sklearn.linear_model_least_angle.Lars	True
llar	Lasso Least Angle Regression	sklearn.linear_model_least_angle.LassoLars	True
omp	Orthogonal Matching Pursuit	sklearn.linear_model_omp.OrthogonalMatchingPu...	True
br	Bayesian Ridge	sklearn.linear_model_bayes.BayesianRidge	True
ard	Automatic Relevance Determination	sklearn.linear_model_bayes.ARDRidgeRegression	False
par	Passive Aggressive Regressor	sklearn.linear_model_passive_aggressive.Pass...	True
ransac	Random Sample Consensus	sklearn.linear_model_ransac.RANSACRegressor	False
tr	TheilSen Regressor	sklearn.linear_model_theil_sen.TheilSenRegressor	False
huber	Huber Regressor	sklearn.linear_model_huber.HuberRegressor	True
kr	Kernel Ridge	sklearn.kernel_ridge.KernelRidge	False
svm	Support Vector Regression	sklearn.svm_classes.SVR	False
knn	K Neighbors Regressor	sklearn.neighbors_regression.KNeighborsRegressor	True
dt	Decision Tree Regressor	sklearn.tree_classes.DecisionTreeRegressor	True
rf	Random Forest Regressor	sklearn.ensemble_forest.RandomForestRegressor	True
et	Extra Trees Regressor	sklearn.ensemble_forest.ExtraTreesRegressor	True
ada	AdaBoost Regressor	sklearn.ensemble_weight_boosting.AdaBoostRegr...	True
gbr	Gradient Boosting Regressor	sklearn.ensemble_gb.GradientBoostingRegressor	True
mlp	MLP Regressor	pycaret.internal.tunable.TunableMLPRegressor	False
xgboost	Extreme Gradient Boosting	xgboost.sklearn.XGBRegressor	True
lightgbm	Light Gradient Boosting Machine	lightgbm.sklearn.LGBMRegressor	True
catboost	CatBoost Regressor	catboost.core.CatBoostRegressor	True

For this study, we evaluated model's performance using metrics: the coefficient of determination MAE(Mean Absolute Error), Mean squared error(MSE), R² and adjusted R², RMSE (Root Means Square Error) and RMLSE (Root Mean Squared Logarithmic Error).

1. Mean Absolute Error (MAE)

We know that an error basically is the absolute difference between the actual or true values and the values that are predicted. Absolute difference means that if the result has a negative sign, it is ignored.

Hence, **MAE = True values – Predicted values.**

MAE takes the **average** of this error from every sample in a dataset and gives the output.

But this value might not be the relevant aspect that can be considered while dealing with a real-life situation because the data we use to build the model as well as evaluate it is the same, which means the model has no exposure to real, never-seen-before data. So, it may perform extremely well on seen data but might fail miserably when it encounters real, unseen data.

The concepts of underfitting and overfitting can be pondered over, from here:

Underfitting: The scenario when a machine learning model almost exactly matches the training data but performs very poorly when it encounters new data or validation set.

Overfitting: The scenario when a machine learning model is unable to capture the important patterns and insights from the data, which results in the model performing poorly on training data itself.

2. Mean squared error(MSE)

MSE is calculated by taking the average of the square of the difference between to original and predicted values of the data.

Hence MSE=

$$\frac{1}{N} \sum_{i=1}^n (\text{actual values} - \text{predicted values})^2$$

Here N is the total number of observations/rows in the dataset.

The sigma symbol denotes that the difference between actual and predicted values taken on every i value ranging from 1 to n.

3. RMSE

It can be defined as the standard sample deviation between the predicted values and the observed ones. It is to be noted that unit of RMSE is same as dependent variable y. The lower RMSE values are indicative of a better fit model. If the model's primary objective is prediction then RMSE is a stronger measure.

4. R-squared and Adjusted R-squared

The R-square value provides a measure of how much the model replicates the actual results, based on the ratio of total variation of outcomes as explained in the model.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_{\text{predicted}} - y_{\text{observed}})^2}{\sum_{i=1}^n (y_{\text{predicted}} - \bar{y}_{\text{observed}})^2}$$

The higher the R-squared, the better the model fits the data given. The R-squared value ranges from 0 to 1, representing the percentage of a squared correlation between the target variable's expected and real values. But in case of multiple linear regressions, R-squared value may increase with increasing features even though the model is not actually improving. A related, Adjusted R-squared statistic can be used to address this disadvantage. This measures the model's goodness and penalizes the model to use more predictors.

5. RMSLE (Root Mean Squared Logarithmic Error)

It is the root mean squared error of the logarithmic transformed predicted and log transformed actual values. The error term can be exploded to a very high value if outliers are present in case of RMSE, whereas in case of RMLSE the outliers are drastically scaled down therefore so that impact is nullified. We have split the training data into subsets of train and test data . We know that multi-collinearity is the effect of the multiple regression models having related predictors. In simple, when data set has a large number of predictors, it is possible that few of these predictors may be highly correlated. Existence of high correlation between independent variables is called multi-collinearity. Presence of multi-collinearity can destabilize the model. Apart from correlation matrix, to check this sort of relations between variables we use, variation inflation factor (VIF). It measures the magnitude of multi-collinearity. It is defined as follows:

$$VIF = \frac{1}{1 - R^2}$$

Compare model results

compare_models()								
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	16.7619	634.8514	25.1549	0.7290	0.2502	0.2091	1.601
lightgbm	Light Gradient Boosting Machine	16.9343	659.7653	25.6280	0.7189	0.2533	0.2115	0.081
xgboost	Extreme Gradient Boosting	16.8138	683.8058	26.0992	0.7089	0.2541	0.2083	0.992
gbr	Gradient Boosting Regressor	17.6530	690.1834	26.2033	0.7056	0.2607	0.2215	0.248
rf	Random Forest Regressor	17.1295	721.8918	26.8171	0.6923	0.2656	0.2136	1.098
lr	Linear Regression	19.5766	776.9876	27.8431	0.6679	0.3427	0.2569	0.117
lar	Least Angle Regression	19.5766	776.9877	27.8431	0.6679	0.3427	0.2569	0.014
br	Bayesian Ridge	19.5764	776.9875	27.8431	0.6679	0.3426	0.2569	0.014
knn	K Neighbors Regressor	18.2393	780.2982	27.9024	0.6668	0.2747	0.2251	0.069
ridge	Ridge Regression	19.5971	781.1669	27.9180	0.6663	0.3341	0.2568	0.013
omp	Orthogonal Matching Pursuit	19.8086	795.3490	28.1664	0.6602	0.3555	0.2611	0.013
huber	Huber Regressor	19.0493	808.0431	28.3895	0.6550	0.3176	0.2356	0.038
par	Passive Aggressive Regressor	19.3589	808.7038	28.3956	0.6548	0.3068	0.2451	0.020
ada	AdaBoost Regressor	21.2843	840.1534	28.9438	0.6410	0.3182	0.3005	0.096
et	Extra Trees Regressor	18.1890	856.5883	29.2267	0.6340	0.2854	0.2241	0.831
lasso	Lasso Regression	21.8390	1038.5204	32.1709	0.5580	0.3187	0.2892	0.014
dt	Decision Tree Regressor	20.1409	1087.4061	32.9090	0.5367	0.3182	0.2458	0.025
en	Elastic Net	33.2801	2232.8965	47.1978	0.0488	0.4957	0.4836	0.015
llar	Lasso Least Angle Regression	34.3570	2352.7704	48.4509	-0.0025	0.5116	0.5013	0.013

<catboost.core.CatBoostRegressor at 0x7faaa9492908>

We found that our best model is Catboost

CatBoost

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

- It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
- Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

“CatBoost” name comes from two words “**C**ategory” and “**B**oosting”.

As discussed, the library works well with multiple **C**ategories of data, such as audio, text, image including historical data.

“**B**oost” comes from gradient boosting machine learning algorithm as this library is based on gradient boosting library. Gradient boosting is a powerful machine learning algorithm that is widely applied to multiple types of business challenges like fraud detection, recommendation items, forecasting and it performs well also. It can also return very good result with relatively less data, unlike DL models that need to learn from a massive amount of data.

Advantages of CatBoost Library

- **Performance:** CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front.
- **Handling Categorical features automatically:** We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features.
- **Robust:** It reduces the need for extensive hyper-parameter tuning and lower the chances of overfitting also which leads to more generalized models. Although, CatBoost has multiple parameters to tune and it contains parameters like the number of trees, learning rate, regularization, tree depth, fold size, bagging temperature and others.
- **Easy-to-use:** You can use CatBoost from the command line, using an user-friendly API for both Python and R.
- **Great quality without parameter tuning:** Reduce time spent on parameter tuning, because CatBoost provides great results with default parameters
- **Fast prediction:** Apply your trained model quickly and efficiently even to latency-critical tasks using CatBoost's model applier
- **Fast and scalable GPU version:** Train your model on a fast implementation of gradient-boosting algorithm for GPU. Use a multi-card configuration for large datasets.

Catboost Secret Sauce

Catboost introduces two critical algorithmic advances - the implementation of **ordered boosting**, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for **processing categorical features**.

Both techniques are using random permutations of the training examples to fight the prediction shift caused by a special kind of target leakage present in all existing implementations of gradient boosting algorithms.

Creating model and cross validation

Creating a model in any module is as simple as writing **create_model**. It takes only one parameter i.e. the Model ID as a string. For supervised modules (classification and regression) this function returns a table with k-fold cross validated performance metrics along with the trained model object.

```
catboost=create_model('catboost')
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	15.8566	531.3412	23.0508	0.7759	0.2371	0.2062
1	15.8521	611.1235	24.7209	0.7212	0.2502	0.2052
2	16.6407	576.8498	24.0177	0.7415	0.2537	0.2162
3	17.3170	700.6395	26.4696	0.6908	0.2573	0.2130
4	17.3223	728.4601	26.9900	0.7135	0.2464	0.2020
5	17.4291	709.7486	26.6411	0.7326	0.2496	0.2053
6	17.9253	717.3344	26.7831	0.7177	0.2645	0.2221
7	15.7210	534.7806	23.1253	0.7104	0.2473	0.2058
8	16.9466	648.6105	25.4678	0.7295	0.2446	0.2053
9	16.6079	589.6261	24.2822	0.7570	0.2515	0.2097
Mean	16.7619	634.8514	25.1549	0.7290	0.2502	0.2091
SD	0.7228	72.3837	1.4437	0.0232	0.0070	0.0059

Hyperparameter tuning

Tuning hyperparameters of a machine learning model in any module is as simple as writing **tune_model**. It tunes the hyperparameter of the model passed as an estimator using Random grid search with pre-defined grids that are fully customizable. Optimizing the hyperparameters of a model requires an objective function which is linked to target variable automatically in supervised experiments such as Regression. The number of folds can be defined using **fold** parameter within **tune_model** function. By default, the **fold** is set to **10**. All the metrics are rounded to 4 decimals by default by can be changed using **round** parameter. Tune model function in PyCaret is a randomized grid search of a pre-defined search space hence it relies on number of iterations of search space. By default, this function performs 10 random iteration over search space which can be changed using **n_iter** parameter within **tune_model**. Increasing the **n_iter** parameter may increase the training time but often results in a highly optimized model. Metric to be optimized can be defined using **optimize** parameter. By default, Regression tasks will optimize **R^2**.

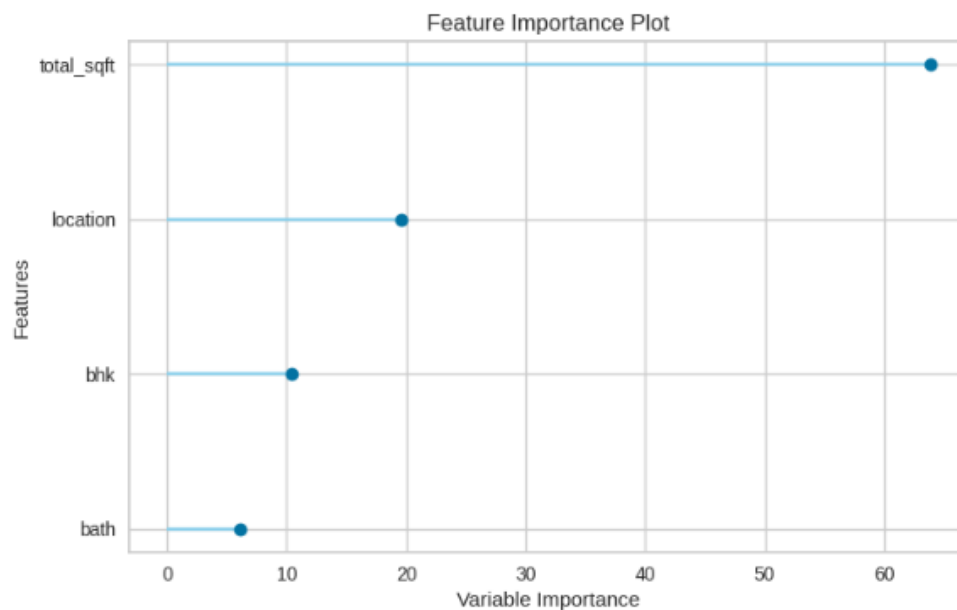
```
tuned_catboost = tune_model(catboost)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	16.4787	576.6547	24.0136	0.7568	0.2465	0.2163
1	16.5337	609.5829	24.6897	0.7219	0.2557	0.2181
2	17.1724	595.1760	24.3962	0.7333	0.2592	0.2246
3	18.2355	747.6594	27.3434	0.6700	0.2660	0.2250
4	18.0279	767.5816	27.7053	0.6981	0.2543	0.2124
5	17.9842	735.5734	27.1215	0.7229	0.2574	0.2149
6	17.9419	692.5083	26.3156	0.7274	0.2652	0.2245
7	16.3622	557.2624	23.6064	0.6983	0.2530	0.2148
8	17.3674	627.9707	25.0593	0.7381	0.2557	0.2159
9	17.0873	616.0289	24.8199	0.7461	0.2607	0.2196
Mean	17.3191	652.5998	25.5071	0.7213	0.2574	0.2186
SD	0.6700	72.5832	1.4100	0.0246	0.0055	0.0044

Plot Model

Analysing performance of trained machine learning model is an integral step in *any* machine learning workflow. analysing model performance in PyCaret is as simple as writing **plot_model**. The function takes trained model object and type of plot as string within **plot_model** function.

```
plot_model(tuned_catboost,plot='feature')
```



Finalize Model

```
final=finalize_model(tuned_catboost)
```

Finalize model is the last step in a typical supervised experiment workflow. When an experiment is started in PyCaret using `setup`, a hold-out set is created that is not being used in model training.

Predict Model

```
predict_unseen=predict_model(final,data=data_unseen)
```

```
predict_unseen.head(50)
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price(in lakhs)	bhk	price_per_sqft	Label
0	Super built-up Area	Ready To Move	Whitefield	2	DuenaTa	1170.00	2.0	1.0	38.00	2.0	3247.863248	57.334020
1	Super built-up Area	19-Dec	Binny Pete	3	She 2rk	1755.00	3.0	1.0	122.00	3.0	6951.566952	114.095146
2	Super built-up Area	Ready To Move	other	1	Ceove G	600.00	1.0	0.0	38.00	1.0	6333.333333	35.956454
3	Super built-up Area	19-Sep	Kanakpura Road	2	Soazak	1330.74	2.0	2.0	91.79	2.0	6897.665960	73.977017
4	Super built-up Area	20-Aug	Begur Road	2	Pruthg	1358.00	2.0	1.0	80.58	2.0	5933.726068	74.073889

Once a model is successfully deployed either on cloud using `deploy_model` or locally using `save_model`, it can be used to predict on unseen data using **`predict_model`** function. These functions take a trained model object and the dataset to predict. It will automatically apply the entire transformation pipeline created during the experiment.

Saving the model

```
save_model(final,'final_project_bengaluru')
```

The process of saving the model is called **`pickling`**. The model is saved in colab and is download from there. The pickled model is used for prediction.

Model Deployment of the House Price Prediction model using Streamlit

Building a Webapp

Python provides us with another robust and low code library called streamlit. Streamlit is a minimal framework that helps deploy powerful apps. It includes simple code to create designs for the dashboard and is compatible with multiple environments. For creating a dashboard in our application, all we need to do is load our pickle file and create a field to enter inputs. Create a new python file for the implementation of the dashboard.

We have installed 3 libraries here. pyngrok is a python wrapper for ngrok which helps to open secure tunnels from public URLs to localhost. This will help us to host our web app. Streamlit will be used to make our web app.

```
!pip install ipykernel==4.10 # only for colab ,and after this please restart
!pip install pycaret
!pip install --upgrade streamlit -q
!pip install pyngrok -q
```

Next, we will have to create a separate session in Streamlit for our app. In this part, we are saving the script as stream.py, and then we are loading the required libraries which are pickle to load the trained model and streamlit to build the app. Then we are loading the trained model and saving it in a variable named model.

```
#importing libraries
from pycaret.regression import load_model, predict_model
import streamlit as st
import pandas as pd
import numpy as np

# loading the trained model
model = load_model('final_project_bengaluru')
```

Now, let us write a function to predict the output for different inputs we get through the web interface. Next, we have defined the prediction function. This function will take the data provided by users as input and make the prediction using the model that we have loaded earlier. It will take the customer details like the gender, marital status, income, loan amount, and credit history as input, and then pre-process that input so that it can be feed to the model and finally, make the prediction using the model loaded as a classifier. In the end, it will return whether the loan is approved or not based on the output of the model.

```
# Making predictions
def predict(model, input_df):
    predictions_df = predict_model(estimator=model, data=input_df)
    predictions = predictions_df['Label'][0]
    return predictions
```

The below code is to display the front-end elements of our web page which include two pictures relating to our project, a select box for iterating between online or batch input and title for our web page

```
def run():
    from PIL import Image
    image = Image.open('banglore_1.jpg')
    image_office = Image.open('banglore_2.jpg')
    st.image(image, use_column_width=True)
    add_selectbox = st.sidebar.selectbox(
        "How would you like to predict?",
        ("Online", "Batch"))
    st.sidebar.info('This app is created to predict the house prices at various locations in Bengaluru')
    st.sidebar.success('https://www.pycaret.org')
    st.sidebar.image(image_office)
    st.title("Predicting House Prices")
```

The following lines create text boxes in which the user can enter the data required to make the predictions. These boxes will represent the five features on which our model is trained. The first box is for the location where u want to buy it. We had set it as a category select box. The user will have more than 200 locations in Bengaluru and rest of locations a deployed under keyword other in location, and we will have to pick one from them. We are creating a dropdown using the selectbox function of streamlit. Similarly, for bhk, bath and total sqft we had defined select boxes. Since these variables will be numeric in nature, we are using the number_input function from streamlit.

```
bhk=st.number_input('bhk' , min_value=1, max_value=6, value=1)
total_sqft =st.number_input('total_sqft',min_value=350.0, max_value=7000.0, value=50)
bath = st.number_input('bath', min_value=1, max_value=6, value=1)
...
```

The below function is creating a dictionary of the data set feature names (key) and the variable we created above for the respective feature (value). This is then loaded into a pandas Data Frame. This dictionary should be in order of how the columns are listed in the original data so that our input_df and original df match up when we concatenate.

```
output=""
input_dict={'location':location,'size':size,'total_sqft':total_sqft,'bhk':bhk}
input_df = pd.DataFrame([input_dict])
```

The below line ensures that when the button called 'Predict' is clicked, the prediction function defined above is called to make the prediction and store it in the variable result

```

if st.button("Predict"):
    output = predict(model=model, input_df=input_df)
    output = str(output)
    st.success('The output is {}'.format(output))

```

The inputs can either be online where users enter the values in fields provided and get the predictions, or it can be through a CSV file. We will implement both these features. Keep in mind when you are creating fields, it has to be in the same order as the training data. Let us now create an option for uploading CSV files as input

```

if add_selectbox == 'Batch':
    file_upload = st.file_uploader("Upload csv file for predictions", type=["csv"])
    if file_upload is not None:
        data = pd.read_csv(file_upload)
        predictions = predict_model(estimator=model, data=data)
        st.write(predictions)

```

At the end of the app, there will be a predict button and after filling in the details, users have to click that button. Once that button is clicked, the prediction function will be called and the result of the predicted price will be displayed in the app. This completes the web app creating part.

Now host this app to a public URL using pyngrok library.

```

!streamlit run --server.port 80 stream_price.py >/dev/null

```

We are first running the python script. And then we will connect it to a public URL:

```

from pyngrok import ngrok
public_url = ngrok.connect(port='80')
print (public_url)

```

This will generate a link something like this:

```

NgrokTunnel: "http://8a34d0cd797c.ngrok.io" -> "http://localhost:80"

```

Note that the link will vary at your end. You can click on the link which will take you to the web app:


For a single data

How would you like to predict?


Online

This app is created to predict the house prices at various locations in Bengaluru

<https://www.qvcaret.org>



PREDICTING HOUSE PRICES IN BENGALURU



Predicting House Prices

location
Electronic City Phase II

bhk
2

total_sqft
600

bath
2

Predict

Price in Lakhs is 38.77845759407735


For a batch of data

How would you like to predict?


Batch

This app is created to predict the house prices at various locations in Bengaluru

<https://www.qvcaret.org>



PREDICTING HOUSE PRICES IN BENGALURU



Predicting House Prices

Upload csv file for predictions

Drag and drop file here
Limit 200MB per file • CSV

Browse Files

unseen_bengaluru.csv 48.59KB

Unnamed: 0	area_type	availability	location
0	Super built-up Area	Ready To Move	Whitefield
1	Super built-up Area	19-Dec	Sinny Pete
2	Super built-up Area	Ready To Move	other
3	Super built-up Area	19-Sep	Kanekpura Road
4	Super built-up Area	26-Aug	Begur Road
5	Super built-up Area	Ready To Move	other
6	Super built-up Area	Ready To Move	other
7	Super built-up Area	Ready To Move	Hosmaru
8	Super built-up Area	Ready To Move	Kothanur
9	Super built-up Area	Ready To Move	Mulimavu
10	Super built-up Area	Ready To Move	other

Enlarged view of the prediction

← → ↺ ⚠ Not secure | 0668da8e6760.ngrok.io 🔍 ☆ ⏴ ⏵

Click to go back, hold to see history

	Unnamed: 0	area_type	availability	location	size	society	total_sqft	bath	balcony	price(in lakhs)	bhk	price_per_sqft	Label
0	0	Super built-up Area	Ready To Move	Whitefield	2	DuenTa	1170	2	1	38	2	3,247.8632	55.1593
1	1	Super built-up Area	10-Dec	Binny Pete	3	She Zrk	1756	3	1	122	3	6,961.6670	111.2612
2	2	Super built-up Area	Ready To Move	other	1	Ceeve G	600	1	0	38	1	6,333.3333	35.6493
3	3	Super built-up Area	10-Sep	Kanekupuz Road	2	Soazak	1,330.7400	2	2	91.7900	2	6,897.6660	75.3244
4	4	Super built-up Area	20-Aug	Segur Road	2	Purvhg	1393	2	1	80.8800	2	5,933.7261	74.4998
5	5	Super built-up Area	Ready To Move	other	3	nan	1450	2	2	70	3	4,527.5852	75.0071
6	6	Super built-up Area	Ready To Move	other	2	Makesez	1007	2	2	43	2	4,270.1092	90.3471
7	7	Super built-up Area	Ready To Move	Moznavu	2	SydeKz	1500	2	1	78	2	5200	87.8379
8	8	Super built-up Area	Ready To Move	Kothanuz	3	Somunya	1828	3	nan	110	3	6,057.5055	114.2928
9	9	Super built-up Area	Ready To Move	Hulimavu	2	Pizan P	1125	2	2	80	2	4,444.4444	84.4191
10	10	Super built-up Area	Ready To Move	other	3	Hohnazi	2000	3	1	175	3	8750	135.8352
11	11	Super built-up Area	Ready To Move	Electronic City	3	Ohm E	1300	2	1	64.8000	3	4300	77.4187
12	12	Super built-up Area	Ready To Move	Malleshwazan	1	Uttitaps	705	1	2	67	1	9,503.5461	43.0964
13	13	Super built-up Area	Ready To Move	Hulimavu	2	Aniona	1242	2	3	51	2	4,106.2802	66.7114
14	14	Built-up Area	Ready To Move	other	4	nan	2000	4	1	75.5000	4	3775	150.8296
15	15	Super built-up Area	Ready To Move	other	3	MacLeK	2405	4	2	260	3	10,510.8105	199.1613
16	16	Plot Area	Ready To Move	other	2	nan	900	2	1	110	2	12,222.2222	47.2851
17	17	Super built-up Area	Ready To Move	Kaggadasapura	3	nan	1250	2	2	56	3	4375	59.7942
18	18	Super built-up Area	Ready To Move	Electronic City Phase II	2	Coash W	1369	2	2	45	2	4,209.5616	81.1995
19	19	Super built-up Area	Ready To Move	other	2	Puddeaz	1613	2	1	51	2	4,503.2102	85.1154
20	20	Super built-up Area	15-Apr	Electronic City	2	Pkashun	1128	2	1	69	2	5,762.4113	92.1093
21	21	Super built-up Area	Ready To Move	Somasundara Palya	3	nan	1600	3	3	64	3	4000	100.8166
22	22	Super built-up Area	15-May	Jalahalli	3	Shard T	1400	3	0	77	3	5500	74.6446
23	23	Built-up Area	20-Dec	other	3	nan	1629	3	2	79	3	4,849.6010	97.2278
24	24	Built-up Area	Ready To Move	other	2	VMonsea	1460	2	1	58	2	3,972.6027	88.4760
25	25	Super built-up Area	Ready To Move	Chandapura	3	Vallisaf	1209	3	1	45	3	3,725.1656	72.1934
26	26	Super built-up Area	Ready To Move	Whitefield	4	Heestt	3262	4	1	230	4	7,072.8707	236.1374
27	27	Super built-up Area	Ready To Move	Hulimavu	3	nan	1650	3	2	78	3	4,727.2727	96.1977
28	28	Plot Area	Ready To Move	Sector 7 HSR Layout	4	nan	3000	5	1	275	4	9,166.6667	227.6590
29	29	Super built-up Area	Ready To Move	Hebbal	3	Btium C	1450	5	2	260	3	7,556.2319	254.6813
30	30	Plot Area	Ready To Move	other	2	nan	1800	1	1	80	2	4,444.4444	123.9871
31	31	Super built-up Area	Ready To Move	Electronics City Phase 1	3	Galicave	1490	3	1	54	3	5,637.5839	81.8194
32	32	Super built-up Area	Ready To Move	HSR Layout	3	nan	1450	2	1	65	3	4,549.4545	73.9949
33	33	Super built-up Area	Ready To Move	Halima Agahaza	3	HancjRe	1450	2	3	55	3	5,562.0450	82.7727
34	34	Super built-up Area	Ready To Move	Raja Rajeshwar Nagar	2	Grivadoz	1173	2	1	49.0700	2	4,192.9797	90.3153
35	35	Super built-up Area	Ready To Move	Chandapura	2	nan	1450	2	1	55	2	4,527.5852	90.3471

Conclusion and Future scope

The ability to build and deploy a machine learning model in around 30 lines of code is remarkable and shows how fast the world of AI is developing. This article shows a simple and efficient way of using different python libraries to build and deploy and model. We have discussed the concept of PropTech and its applications. We worked on the use-case of proactive pricing of houses in classified advertisements in the Indian Real Estate market. We used advanced machine learning and artificial intelligence techniques to develop an algorithm that can predict housing prices based on certain input features. The business application of this algorithm is that classified websites can directly use this algorithm to predict prices of new properties that are going to be listed by taking some input variables and predicting the correct and justified price i.e. avoid taking price inputs from customers and thus keeping transparency in the system. The proposed methods can be used for other applications of PropTech as mentioned above. Since this was a proof of concept (POC), so the models were implemented on a smaller dataset. However, the error margins can be reduced further if we use much larger datasets, which we aim to work on in the future.

An optimal model does not necessarily represent a robust model. A model that frequently use a learning algorithm that is not suitable for the given data structure. Sometimes the data itself might be too noisy or it could contain too few samples to enable a model to accurately capture the target variable which implies that the model remains fit. When we observe the evaluation metrics obtained for advanced regression models, we can say both behave in a similar manner. We can choose either one for house price prediction compared to basic model. With the help of box plots, we can check for outliers. If present, we can remove outliers and check the model's performance for improvement. We can build models through advanced techniques namely random forests, neural networks, and particle swarm optimization to improve the accuracy of predictions.

Model Applicability

It is necessary to check before deciding whether the built model should or should not be used in a real-world setting. The data has been collected in 2016 and Bengaluru is growing in size and population rapidly. So, it is very much essential to look into the relevancy of data today. The characteristics present in the data set are not sufficient to describe house prices in Bengaluru. The dataset considered is quite limited and there are a lot of features, like the presence of pool or not, parking lot and others, that remain very relevant when considering a house price. The property has to be categorized either as a flat or villa or independent house. Data collected from a big urban city like Bengaluru would not be applicable in a rural city, as for equal value of feature prices, which will be comparatively higher in the urban area.

GitHub

https://github.com/johnahjohn/ml_bengaluru_house_price_prediction

Streamlit Webapp

https://share.streamlit.io/johnahjohn/ml_bengaluru_house_price_prediction/main/stream_price.py