Project: House Rent Prediction

Submitted By

Gayathri S (EBEON1222631830)

Abstract

ABSTRACT

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more. In this paper, we calculate accuracy of machine learning algorithms for predicting house price. The algorithms used in this paper are k-nearest neighbor, decision tree, Logistic Regression and Random Forest by using Kaggle repository dataset. For implementation of Python programming Anaconda(jupyter) notebook is best tool, which have many type of library, header file, that make the work more accurate and precise. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

Contents

Table of Contents

Chapter-I

Chapter I
INTRODUCTION

Housing in India varies from palaces of earlier times to modern apartment buildings in big cities to tiny huts in far-flung villages. There has been tremendous growth in India's housing sector as incomes have risen. The Human Rights Measurement Initiative finds that India is doing 60.9% of what should be possible at its level of income for the right to housing.

Renting, also known as hiring or letting, is an agreement where a payment is made for the temporary use of a good, service, or property owned by another. A gross lease is when the tenant pays a flat rental amount and the landlord pays for all property charges regularly incurred by the ownership. Renting can be an example of the sharing economy.

Scenario:

In this Dataset, we have information on almost 4700+ Houses/Apartments/Flats Available for Rent with different parameters like BHK, Rent, Size, No. of Floors, Area Type, Area Locality, City, Furnishing Status, Type of Tenant Preferred, No. of Bathrooms, Point of Contact.

Goal:

- Predict the house price
- Experiment with various Classification Models & see which yields greatest accuracy.
- Examine trends & correlations within our data
- Determine which features are most important

`

Chapter-II

Features & Predictor

Our Predictor (Y, Positive or Negative diagnosis of House Prediction) is determined by 11 features (X):

1) **BHK:** Number of Bedrooms, Hall, Kitchen.
2) **Rent:** Price of the Houses/Apartments/Flats.
3) **Size:** Size of the Houses/Apartments/Flats in Square Feet.
4) **Floor:** Houses/Apartments/Flats situated in which Floor and Total Number of Floors (Example: Ground out of 2, 3 out of 5, etc.)
5) **Area Type:** Size of the Houses/Apartments/Flats calculated on either Super Area or Carpet Area or Build Area.
6) **Area Locality:** Locality of the Houses/Apartments/Flats.
7) **City:** City where the Houses/Apartments/Flats are Located.
8) **Furnishing Status:** Furnishing Status of the Houses/Apartments/Flats, either it is Furnished or Semi-Furnished or Unfurnished.
9) **Tenant Preferred:** Type of Tenant Preferred by the Owner or Agent.
10) **Bathroom:** Number of Bathrooms.
11) **Point of Contact:** Whom should you contact for more information regarding the Houses/Apartments/Flats.

Note: Our data has 3 types of data:

Continuous: Which is quantitative data that can be measured

Ordinal Data: Categorical data that has a order to it (0,1,2,3,etc)

Binary Data: Data whose unit can take on only two possible states(0&1)

Chapter-III

Chapter:III
Methodology

Data Cleaning and Preprocessing:

The datasets which were collected from Kaggle website contain unfiltered data which must be filtered before the final data set can be used to train the model. Also, data has some categorical variables which must be modified into numerical values for which we used Pandas library of Python. In data cleaning step, first we checked whether there are any missing or junk values in the dataset for which we used the isnull() function. Then for handling categorical variables we converted them into numerical variables

Machine Learning Algorithms:

1) Random Forest :

Random Forest is the most famous and it is considered as the best algorithm for machine learning. It is a supervised learning algorithm. To achieve more accurate and consistent prediction, random forest creates several decision trees and combines them together. The major benefit of using it is its ability to solve both regression and classification issues. When building each individual tree, it employs bagging and feature randomness in order to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree's prediction. Bagging enhances accuracy of machine learning methods by grouping them together. In this algorithm, during the splitting of nodes it takes only random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results into getting better accuracy. It efficiently deals with the huge datasets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then for each result which was predicted, it'll perform voting and through voting it will select the best predicted result.

2) Logistic Regression :

Logistic regression is often used a lot of times in machine learning for predicting the likelihood of response attributes when a set of explanatory independent attributes are given. It is used when the target attribute is also known as a dependent variable having categorical values like yes/no or true/false, etc. It's widely used for solving classification problems. It falls under the category of supervised machine learning. It efficiently solves linear and binary classification problems. It is one of the most commonly used and easy to implement algorithms. It's a statistical technique to predict classes which are binary. When the target variable has two possible classes in that case it predicts the likelihood of occurrence of the event. In our dataset the target variable is categorical as it has only two classes-yes/no.

3) Decision Tree:

It is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
The decisions or the test are performed on the basis of features of the given dataset.
It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
A decision tree can contain categorical data (YES/NO) as well as numeric data.

4) K Nearest Neighbor (KNN) :

KNN is a supervised machine learning algorithm. It assumes similar objects are nearer to one another. When the parameters are continuous in that case knn is preferred. In this algorithm it classifies objects by predicting their nearest neighbor. It's simple and easy to implement and also has high speed because of which it is preferred over the other algorithms when it comes to solving classification problems. The algorithm classifies whether or not the patient has disease by taking the heart disease dataset as an input. It takes input parameters like age, sex, chol, etc and classify person with heart disease. Algorithm takes following steps :-

Step 1: Select the value for K.

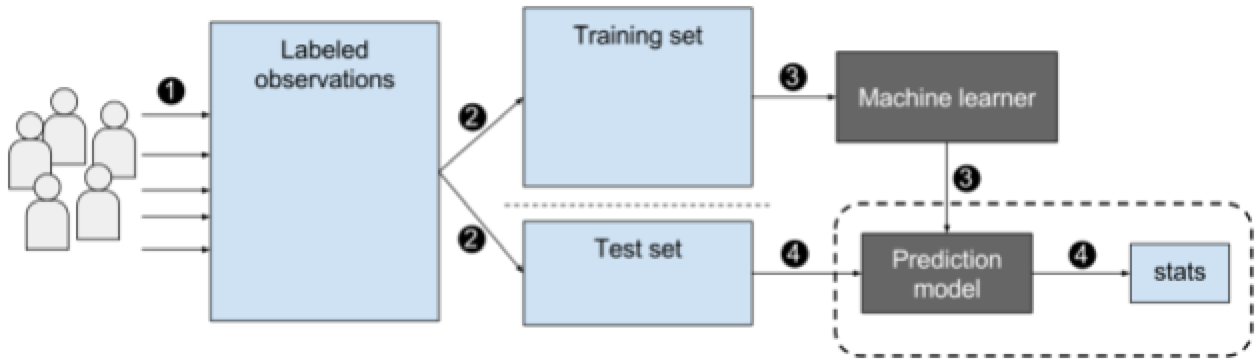
Step 2 : Find the Euclidean distance of K no. of neighbors.

Step 3 : Based on calculated distance, select the K nearest neighbors in the training data which are nearest to unknown data points.

Step 4 : Calculate no. of data points in each category among these K neighbors.

Step 5 : Assign new data points to the category which has the maximum no. of neighbors.

Step 6 : Stop.



Implementation Steps:

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps :

1) Dataset collection.
2) Importing Libraries : Numpy, Pandas, Scikit-learn, Matplotlib and Seaborn libraries were used.
3) Exploratory data analysis : For getting more insights about data.
4) Data cleaning and preprocessing : Checked for null and junk values using isnull() and isna().sum() functions of python.In Preprocessing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using function of Pandas library. Both our datasets contains some categorical variables
5) Feature Scaling : In this step, we normalize our data by applying Standardization by using StandardScalar() and fit_transform() functions of scikit-learn library.
6) Model selection : We first separated X's from y's. X's are features or input variables of our datasets and y's are dependent or target variables which are crucial for predicting disease. Then using by the importing model_selection function of the sklearn library, we splitted our X's and y's into train and test split using train_test_split() function

of sklearn. We splitted 70% of our data for training and 30% for testing.

7) Applied ML models and created a confusion matrix of all models.

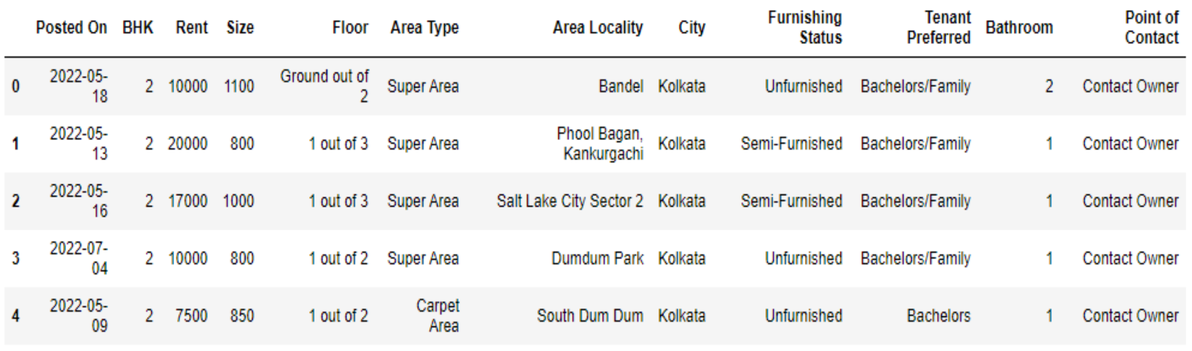
8) Deployment of the model which gave the best accuracy.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import make_scorer, accuracy_score, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

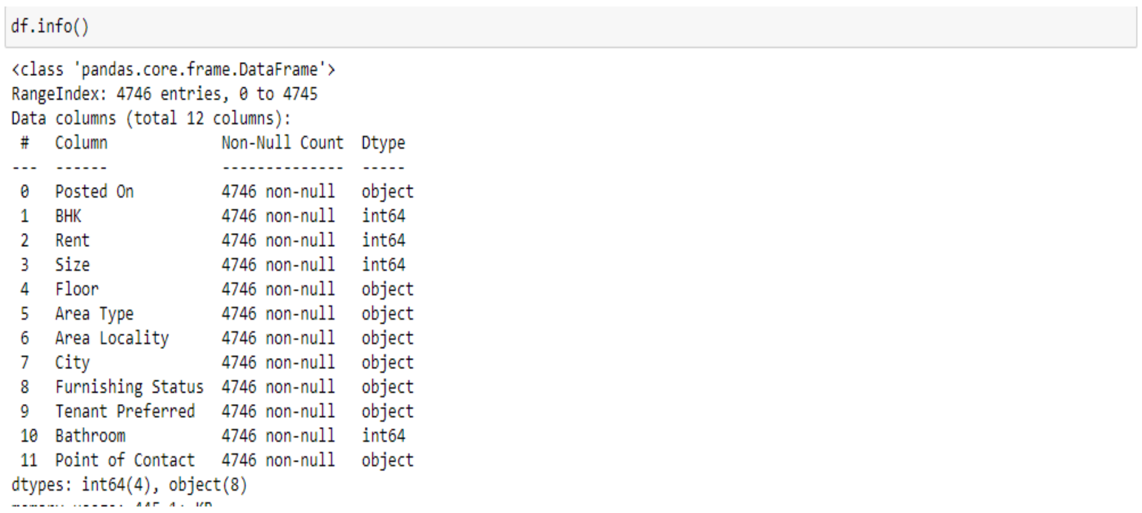
Data Wrangling

```
df=pd.read_csv("house_rent.csv")
df.head()
```



	Posted On	BHK	Rent	Size	Floor	Area Type	Area Locality	City	Furnishing Status	Tenant Preferred	Bathroom	Point of Contact
0	2022-05-18	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
1	2022-05-13	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
2	2022-05-16	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	2022-07-04	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
4	2022-05-09	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner

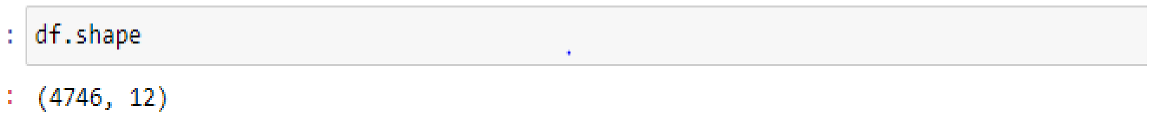
Displaying the information about data



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Posted On          4746 non-null   object
 1   BHK                4746 non-null   int64 
 2   Rent               4746 non-null   int64 
 3   Size               4746 non-null   int64 
 4   Floor              4746 non-null   object
 5   Area Type          4746 non-null   object
 6   Area Locality      4746 non-null   object
 7   City               4746 non-null   object
 8   Furnishing Status  4746 non-null   object
 9   Tenant Preferred   4746 non-null   object
 10  Bathroom           4746 non-null   int64 
 11  Point of Contact   4746 non-null   object
dtypes: int64(4), object(8)
```

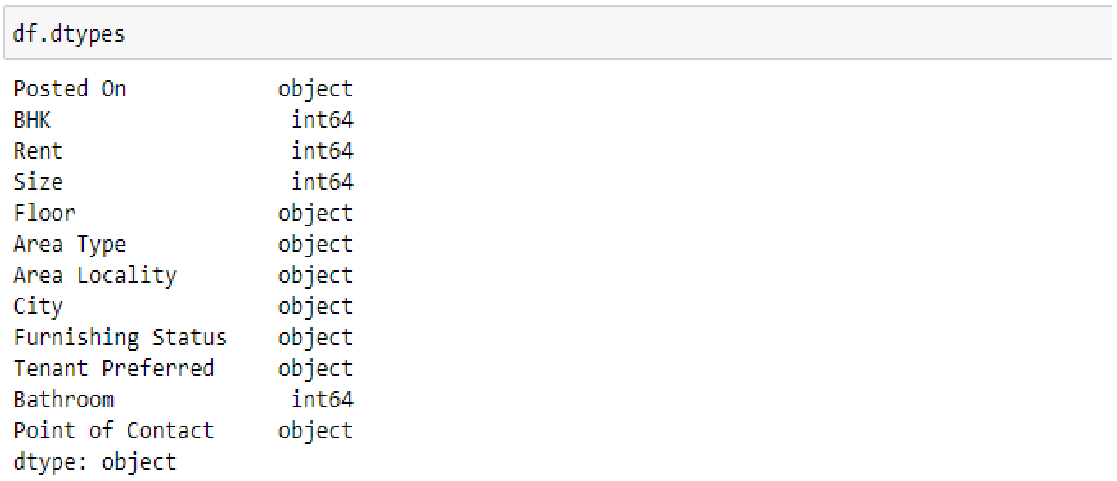
Checking the dimensions of dataset



```
: df.shape
```

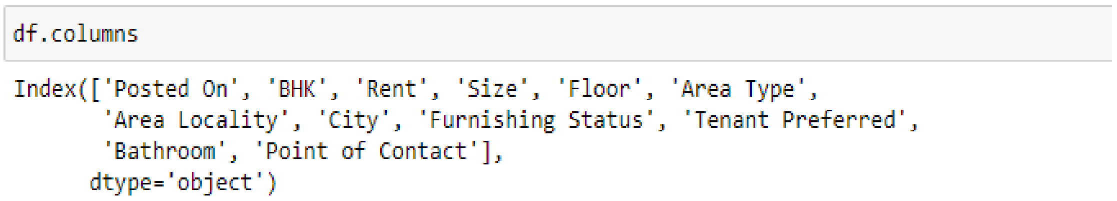
```
: (4746, 12)
```

Checking the data types and columns present in dataset



```
df.dtypes
```

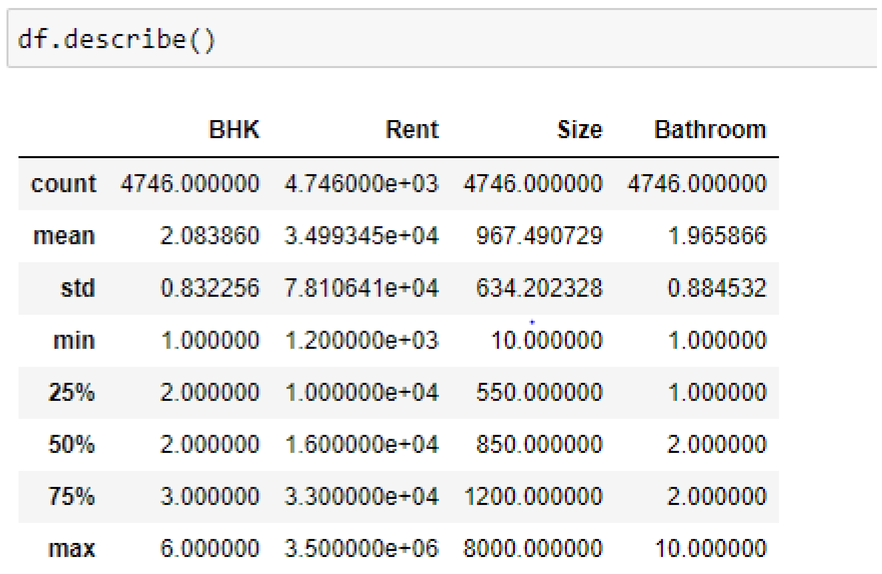
```
Posted On            object
BHK                   int64
Rent                  int64
Size                  int64
Floor                object
Area Type            object
Area Locality        object
City                 object
Furnishing Status    object
Tenant Preferred     object
Bathroom              int64
Point of Contact     object
dtype: object
```



```
df.columns
```

```
Index(['Posted On', 'BHK', 'Rent', 'Size', 'Floor', 'Area Type',
       'Area Locality', 'City', 'Furnishing Status', 'Tenant Preferred',
       'Bathroom', 'Point of Contact'],
      dtype='object')
```

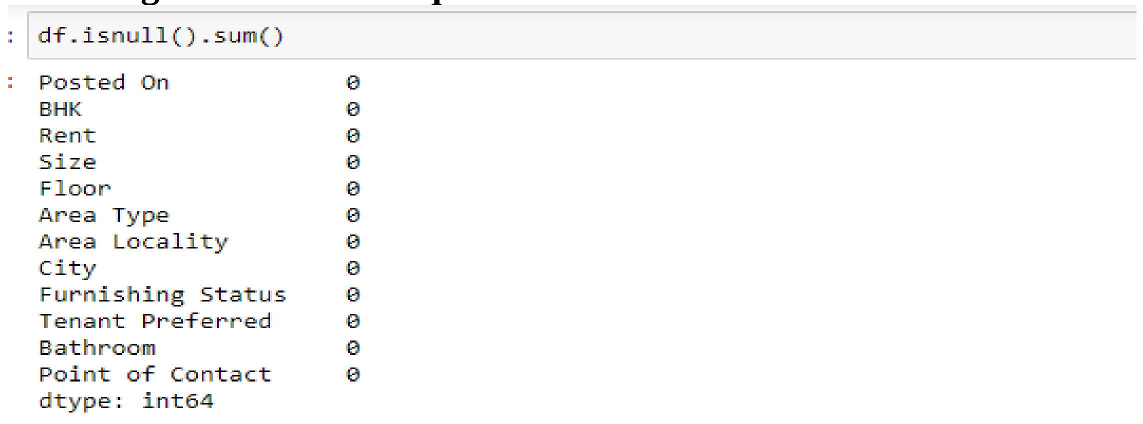
Displaying the statistical info about dataset



```
df.describe()
```

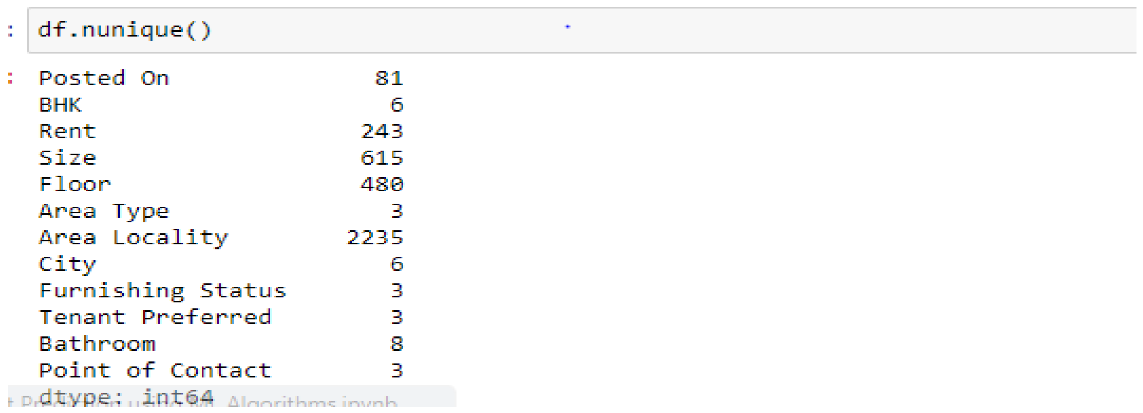
	BHK	Rent	Size	Bathroom
count	4746.000000	4.746000e+03	4746.000000	4746.000000
mean	2.083860	3.499345e+04	967.490729	1.965866
std	0.832256	7.810641e+04	634.202328	0.884532
min	1.000000	1.200000e+03	10.000000	1.000000
25%	2.000000	1.000000e+04	550.000000	1.000000
50%	2.000000	1.600000e+04	850.000000	2.000000
75%	3.000000	3.300000e+04	1200.000000	2.000000
max	6.000000	3.500000e+06	8000.000000	10.000000

Checking the null and unique values



```
: df.isnull().sum()
```

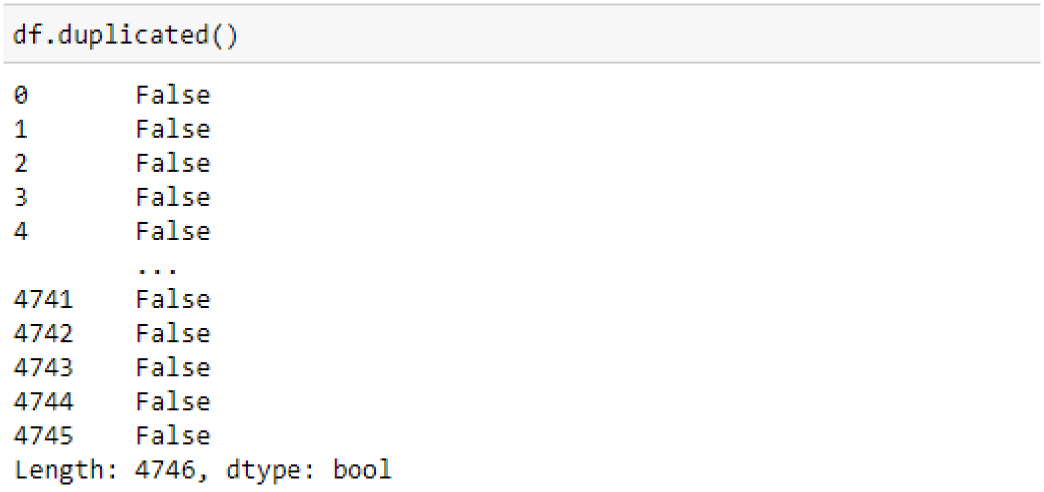
```
: Posted On            0
  BHK                  0
  Rent                 0
  Size                 0
  Floor                0
  Area Type            0
  Area Locality        0
  City                 0
  Furnishing Status    0
  Tenant Preferred     0
  Bathroom             0
  Point of Contact     0
  dtype: int64
```



```
: df.nunique()
```

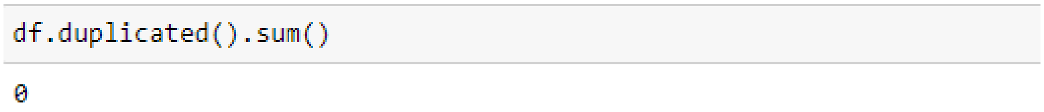
```
: Posted On              81
  BHK                     6
  Rent                  243
  Size                  615
  Floor                 480
  Area Type               3
  Area Locality        2235
  City                    6
  Furnishing Status       3
  Tenant Preferred        3
  Bathroom                8
  Point of Contact        3
  dtype: int64
```

Checking for duplicate values



```
df.duplicated()
```

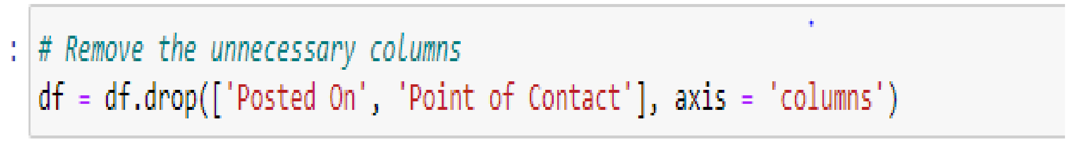
```
0       False
1       False
2       False
3       False
4       False
        ...
4741    False
4742    False
4743    False
4744    False
4745    False
Length: 4746, dtype: bool
```



```
df.duplicated().sum()
```

```
0
```

Drop the unnecessary column

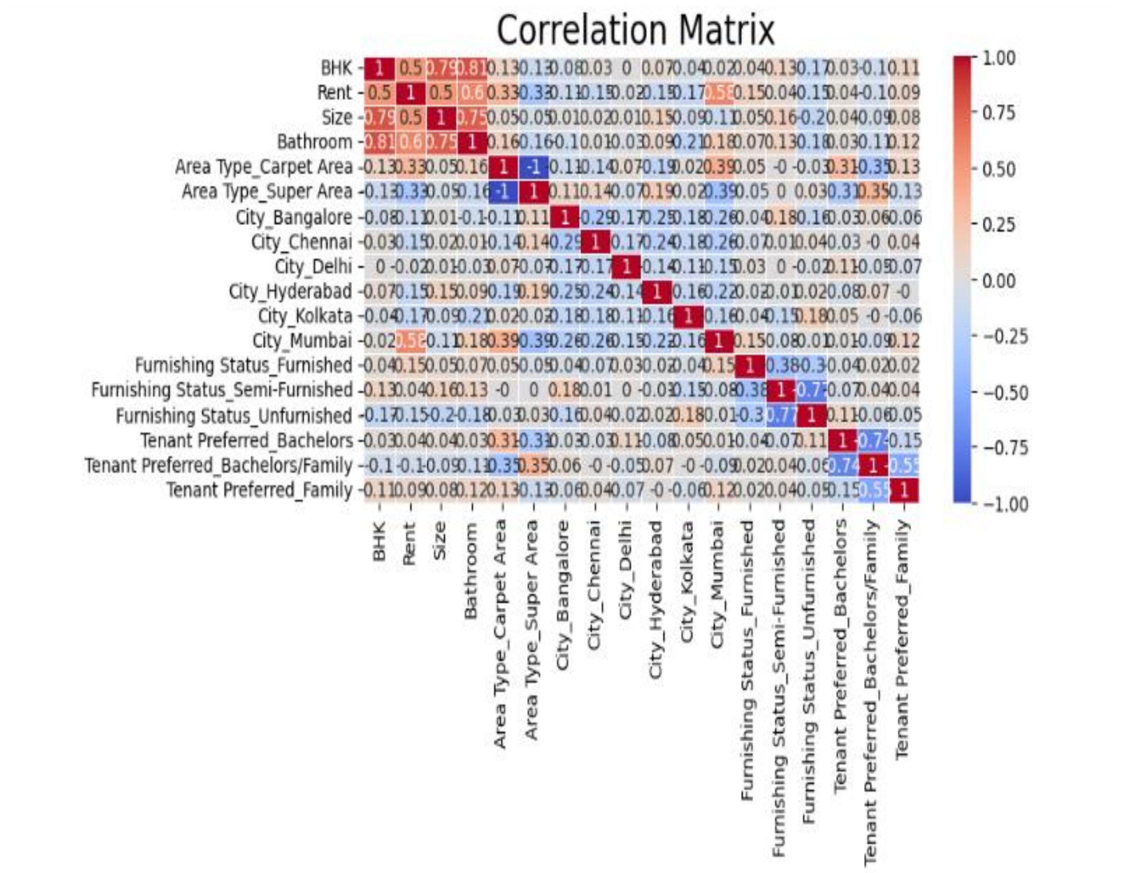


```
# Remove the unnecessary columns
df = df.drop(['Posted On', 'Point of Contact'], axis = 'columns')
```

Exploratory Data Analysis

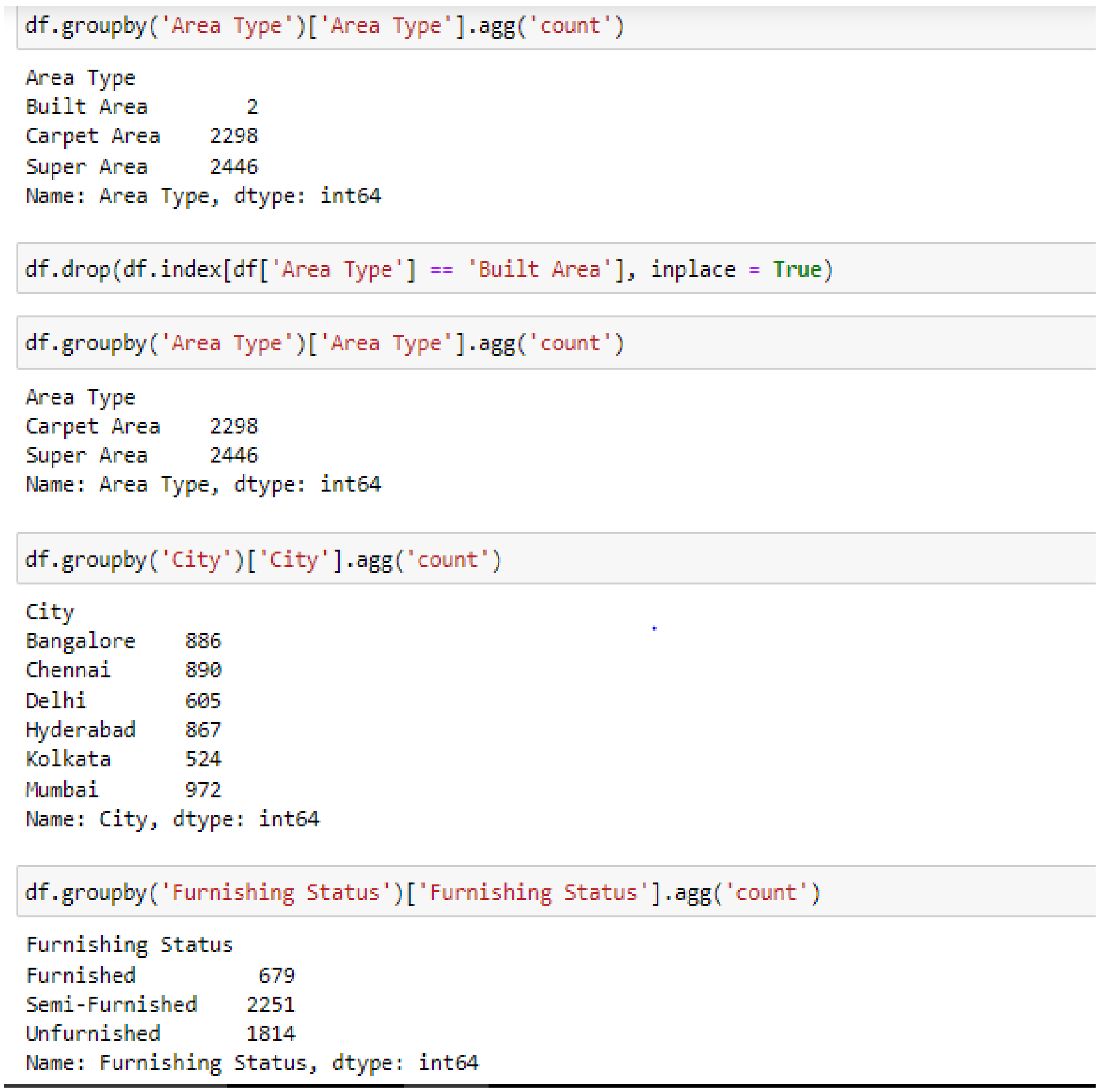
Correlation Matrix- let's see correlations between all variables. Using this we can see whether something is positively or negatively correlated with our predictor (target).

```
import seaborn as sns
#get correlations of each features in dataset
plt.figure(figsize=(10, 8))

correlation_matrix = df3.corr().round(2)

sns.heatmap(data=correlation_matrix,
annot=True, cmap='coolwarm',
linewidths=0.5, )

plt.title("Correlation Matrix ", size=20)
```



Performing univariate analysis

Univariate analysis is the technique of comparing and analyzing the dependency of a single predictor and a response variable.



```python
df.groupby('Area Type')['Area Type'].agg('count')
```

```
Area Type
Built Area         2
Carpet Area     2298
Super Area      2446
Name: Area Type, dtype: int64
```

```python
df.drop(df.index[df['Area Type'] == 'Built Area'], inplace = True)
```

```python
df.groupby('Area Type')['Area Type'].agg('count')
```

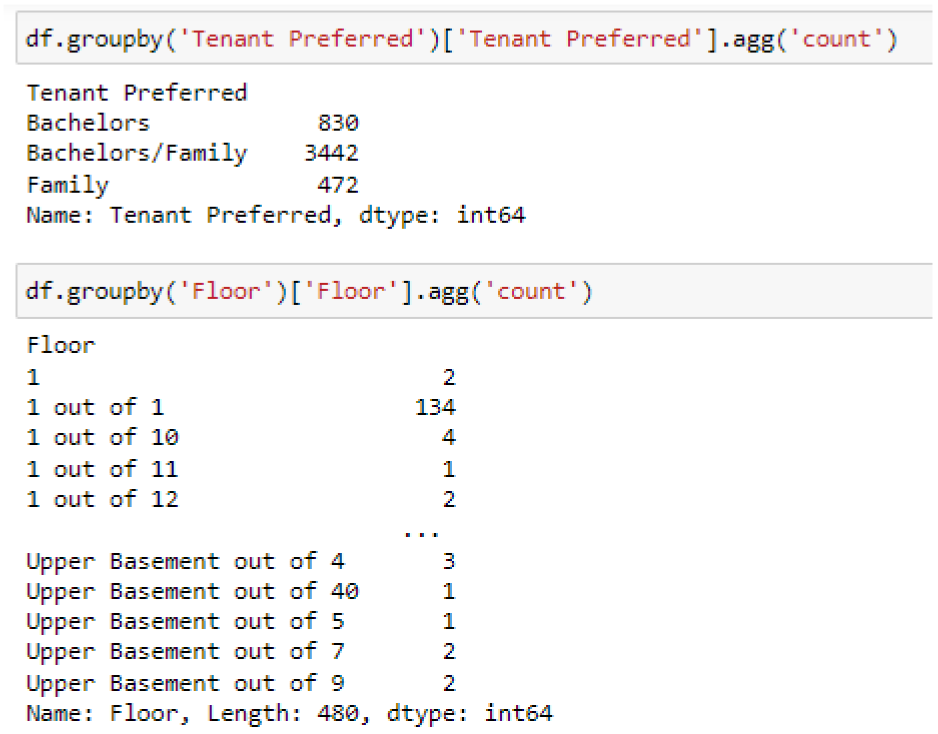
```
Area Type
Carpet Area     2298
Super Area      2446
Name: Area Type, dtype: int64
```

```python
df.groupby('City')['City'].agg('count')
```

```
City
Bangalore    886
Chennai      890
Delhi        605
Hyderabad    867
Kolkata      524
Mumbai       972
Name: City, dtype: int64
```

```python
df.groupby('Furnishing Status')['Furnishing Status'].agg('count')
```

```
Furnishing Status
Furnished          679
Semi-Furnished    2251
Unfurnished       1814
Name: Furnishing Status, dtype: int64
```



```
df.groupby('Tenant Preferred')['Tenant Preferred'].agg('count')
```

```
Tenant Preferred
Bachelors           830
Bachelors/Family   3442
Family              472
Name: Tenant Preferred, dtype: int64
```

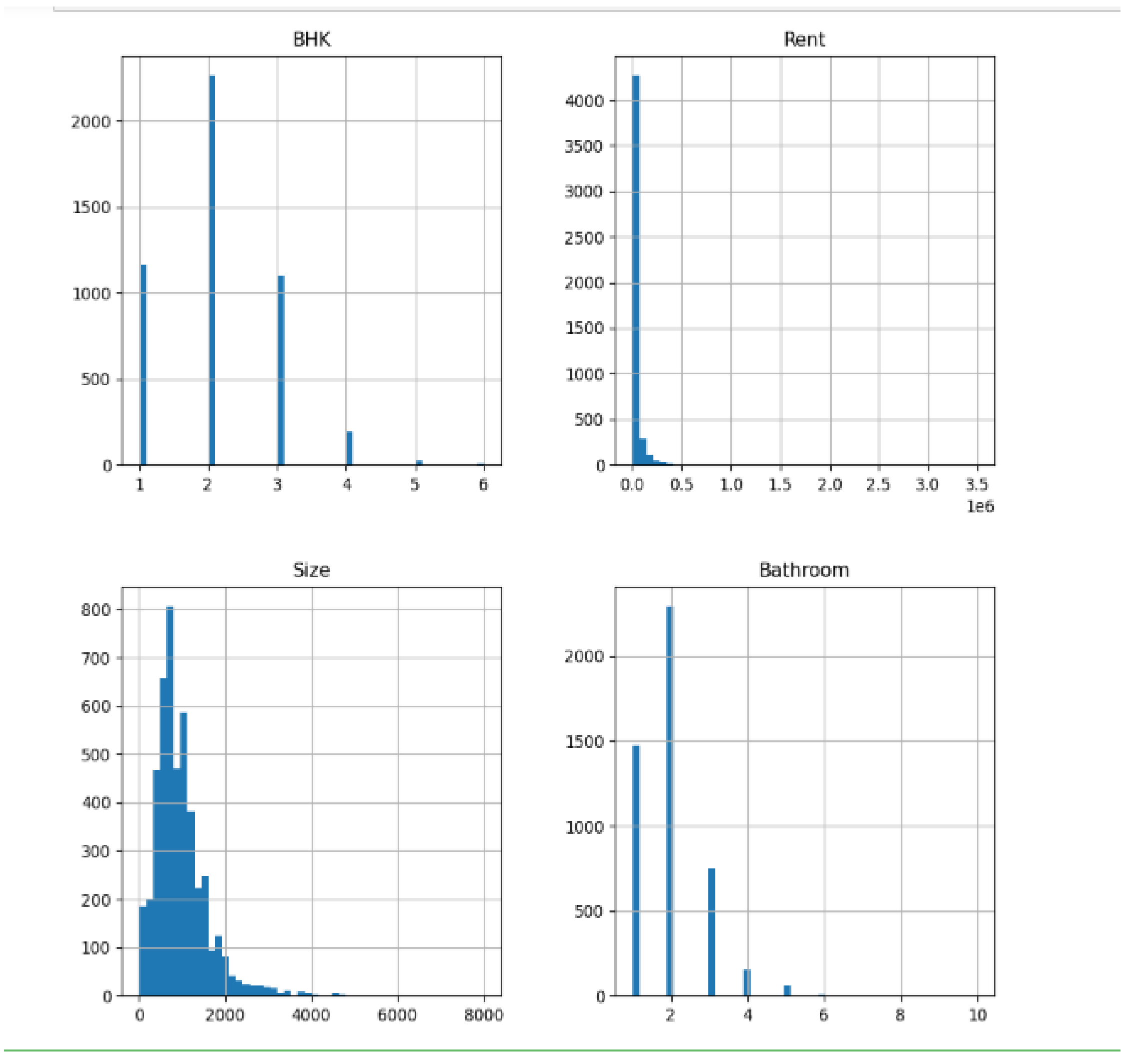
```
df.groupby('Floor')['Floor'].agg('count')
```

```
Floor
1                             2
1 out of 1                  134
1 out of 10                   4
1 out of 11                   1
1 out of 12                   2
                           ...
Upper Basement out of 4       3
Upper Basement out of 40      1
Upper Basement out of 5       1
Upper Basement out of 7       2
Upper Basement out of 9       2
Name: Floor, Length: 480, dtype: int64
```

Plotting Histogram

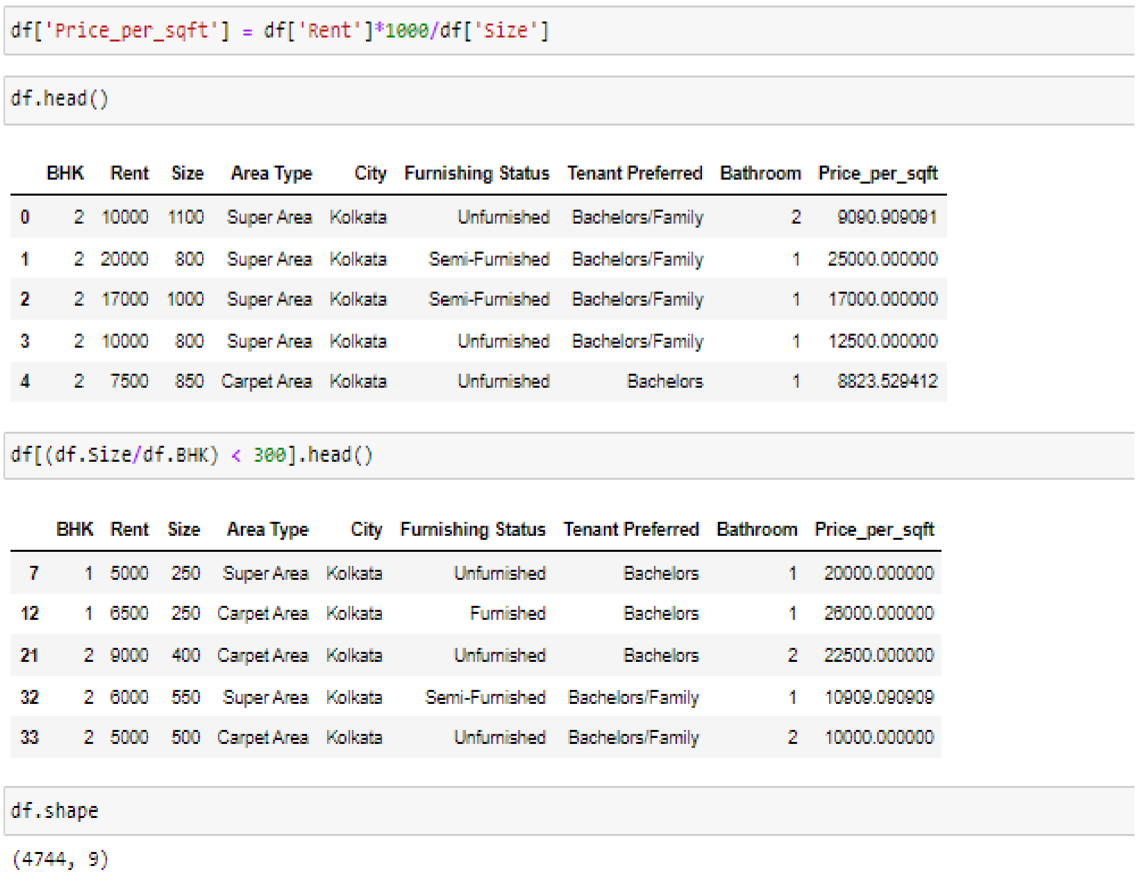
A histogram is a chart that plots the distribution of a numeric variable's values as a series of bars. Each bar typically covers a range of numeric values called a bin or class; a bar's height indicates the frequency of data points with a value within the corresponding bin.

```
df.hist(bins=50, figsize=(10,10))
plt.ticklabel_format(useOffset=False, style='plain')
plt.show()
```



Performing multivariate analysis

Multivariate data analysis is a type of statistical analysis that involves more than two dependent variables, resulting in a single outcome. Many problems in the world can be practical examples of multivariate equations as whatever happens in the world happens due to multiple reasons.



```
df['Price_per_sqft'] = df['Rent']*1000/df['Size']
```

```
df.head()
```

	BHK	Rent	Size	Area Type	City	Furnishing Status	Tenant Preferred	Bathroom	Price_per_sqft
0	2	10000	1100	Super Area	Kolkata	Unfurnished	Bachelors/Family	2	9090.909091
1	2	20000	800	Super Area	Kolkata	Semi-Furnished	Bachelors/Family	1	25000.000000
2	2	17000	1000	Super Area	Kolkata	Semi-Furnished	Bachelors/Family	1	17000.000000
3	2	10000	800	Super Area	Kolkata	Unfurnished	Bachelors/Family	1	12500.000000
4	2	7500	850	Carpet Area	Kolkata	Unfurnished	Bachelors	1	8823.529412

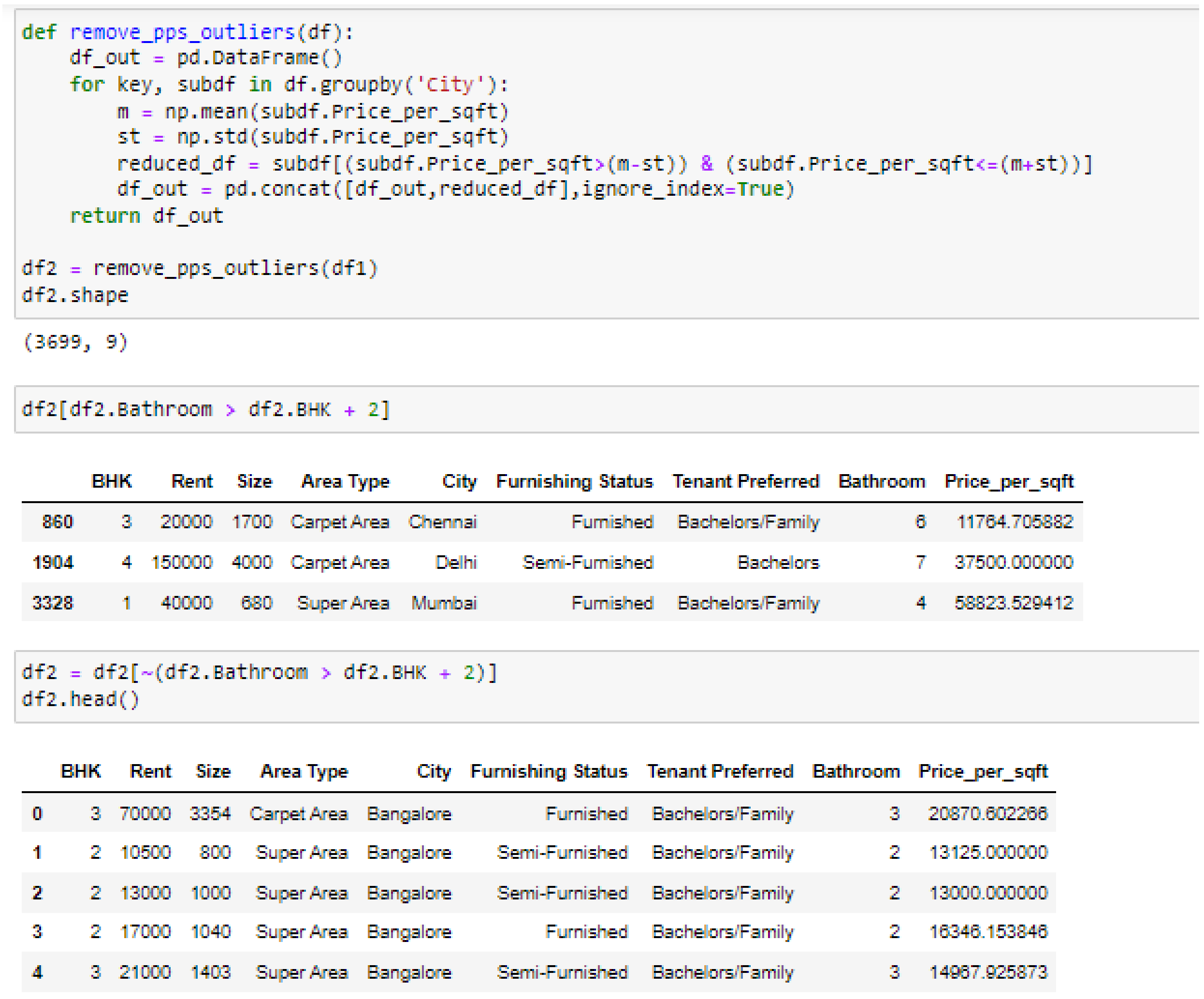
```
df[(df.Size/df.BHK) < 300].head()
```

	BHK	Rent	Size	Area Type	City	Furnishing Status	Tenant Preferred	Bathroom	Price_per_sqft
7	1	5000	250	Super Area	Kolkata	Unfurnished	Bachelors	1	20000.000000
12	1	6500	250	Carpet Area	Kolkata	Furnished	Bachelors	1	26000.000000
21	2	9000	400	Carpet Area	Kolkata	Unfurnished	Bachelors	2	22500.000000
32	2	6000	550	Super Area	Kolkata	Semi-Furnished	Bachelors/Family	1	10909.090909
33	2	5000	500	Carpet Area	Kolkata	Unfurnished	Bachelors/Family	2	10000.000000

```
df.shape
```

```
(4744, 9)
```

Removing Outliers



```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('City'):
        m = np.mean(subdf.Price_per_sqft)
        st = np.std(subdf.Price_per_sqft)
        reduced_df = subdf[(subdf.Price_per_sqft>(m-st)) & (subdf.Price_per_sqft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out

df2 = remove_pps_outliers(df1)
df2.shape
```

```
(3699, 9)
```

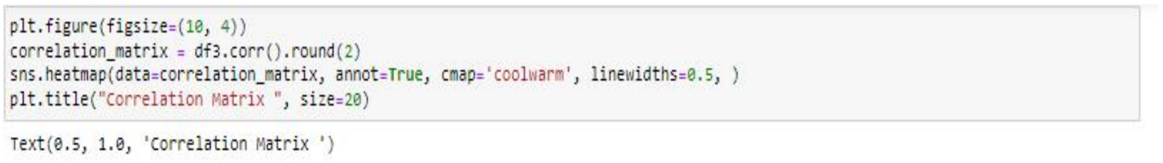
```
df2[df2.Bathroom > df2.BHK + 2]
```

	BHK	Rent	Size	Area Type	City	Furnishing Status	Tenant Preferred	Bathroom	Price_per_sqft
860	3	20000	1700	Carpet Area	Chennai	Furnished	Bachelors/Family	6	11764.705882
1904	4	150000	4000	Carpet Area	Delhi	Semi-Furnished	Bachelors	7	37500.000000
3328	1	40000	680	Super Area	Mumbai	Furnished	Bachelors/Family	4	58823.529412

```
df2 = df2[~(df2.Bathroom > df2.BHK + 2)]
df2.head()
```

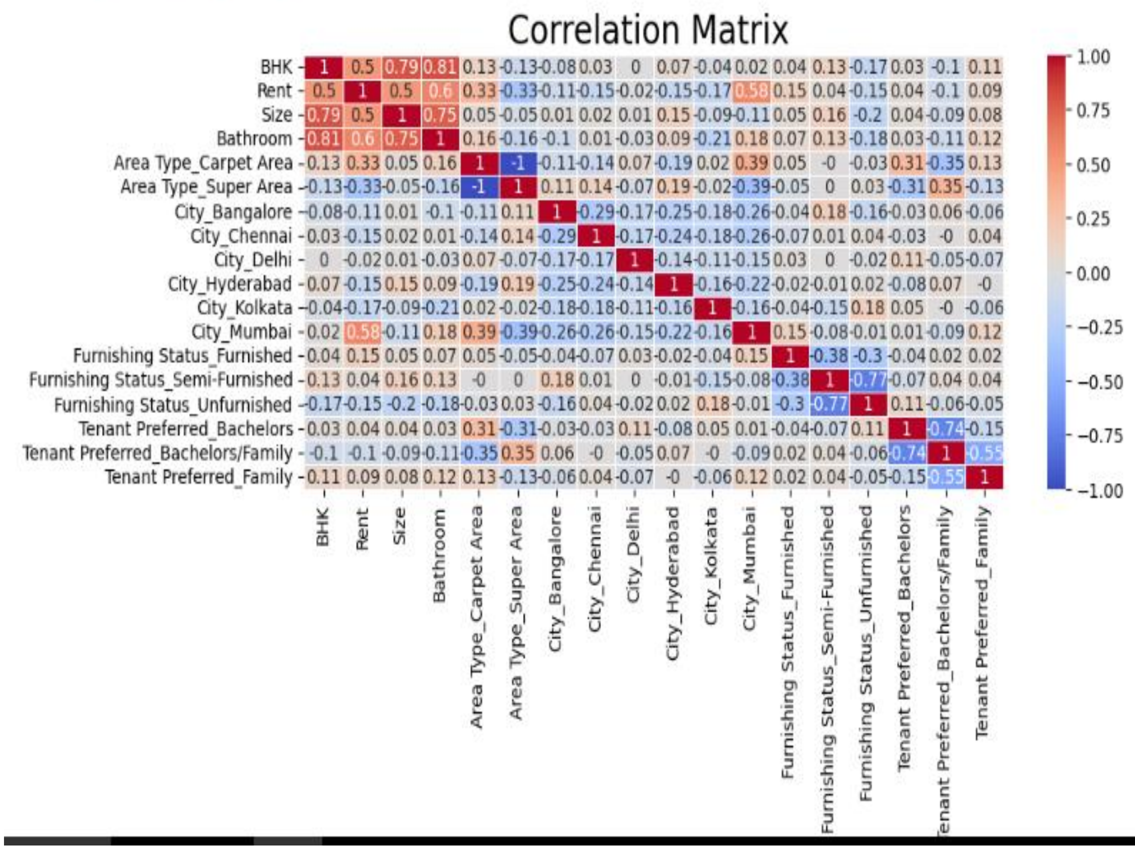
	BHK	Rent	Size	Area Type	City	Furnishing Status	Tenant Preferred	Bathroom	Price_per_sqft
0	3	70000	3354	Carpet Area	Bangalore	Furnished	Bachelors/Family	3	20870.602266
1	2	10500	800	Super Area	Bangalore	Semi-Furnished	Bachelors/Family	2	13125.000000
2	2	13000	1000	Super Area	Bangalore	Semi-Furnished	Bachelors/Family	2	13000.000000
3	2	17000	1040	Super Area	Bangalore	Furnished	Bachelors/Family	2	16346.153846
4	3	21000	1403	Super Area	Bangalore	Semi-Furnished	Bachelors/Family	3	14967.925873

Finding correlation between the variables



```
plt.figure(figsize=(10, 4))
correlation_matrix = df3.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5, )
plt.title("Correlation Matrix ", size=20)
```

```
Text(0.5, 1.0, 'Correlation Matrix ')
```



Bar Plot

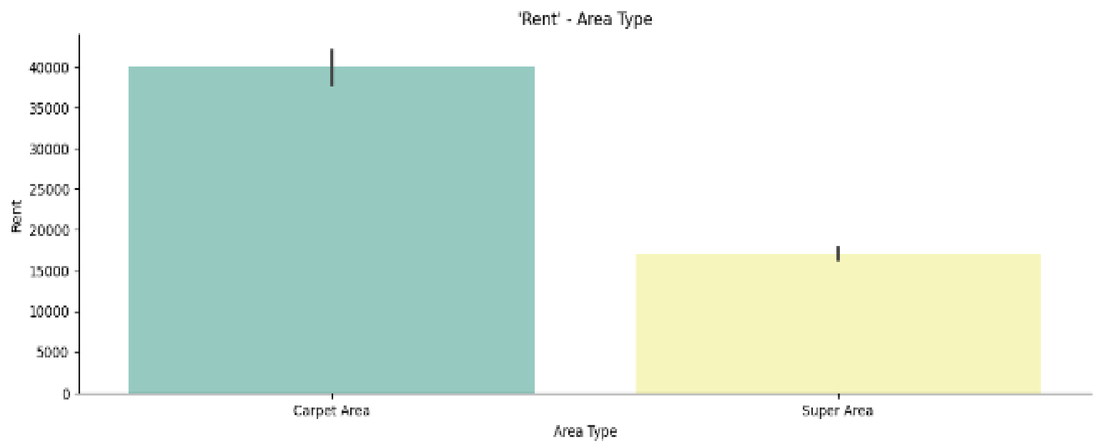
Bar plots are a type of data visualization used to represent data in the form of rectangular bars. The height of each bar represents the value of a data point, and the width of each bar represents the category of the data. The Matplotlib library in Python is widely used to create bar plots.

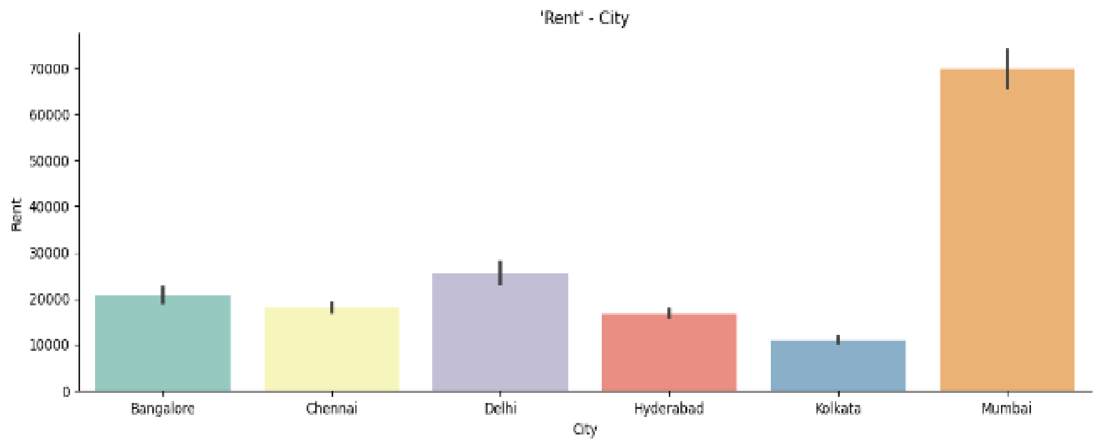
cat_features = df2.select_dtypes(include='object').columns.to_list()

for col in cat_features:

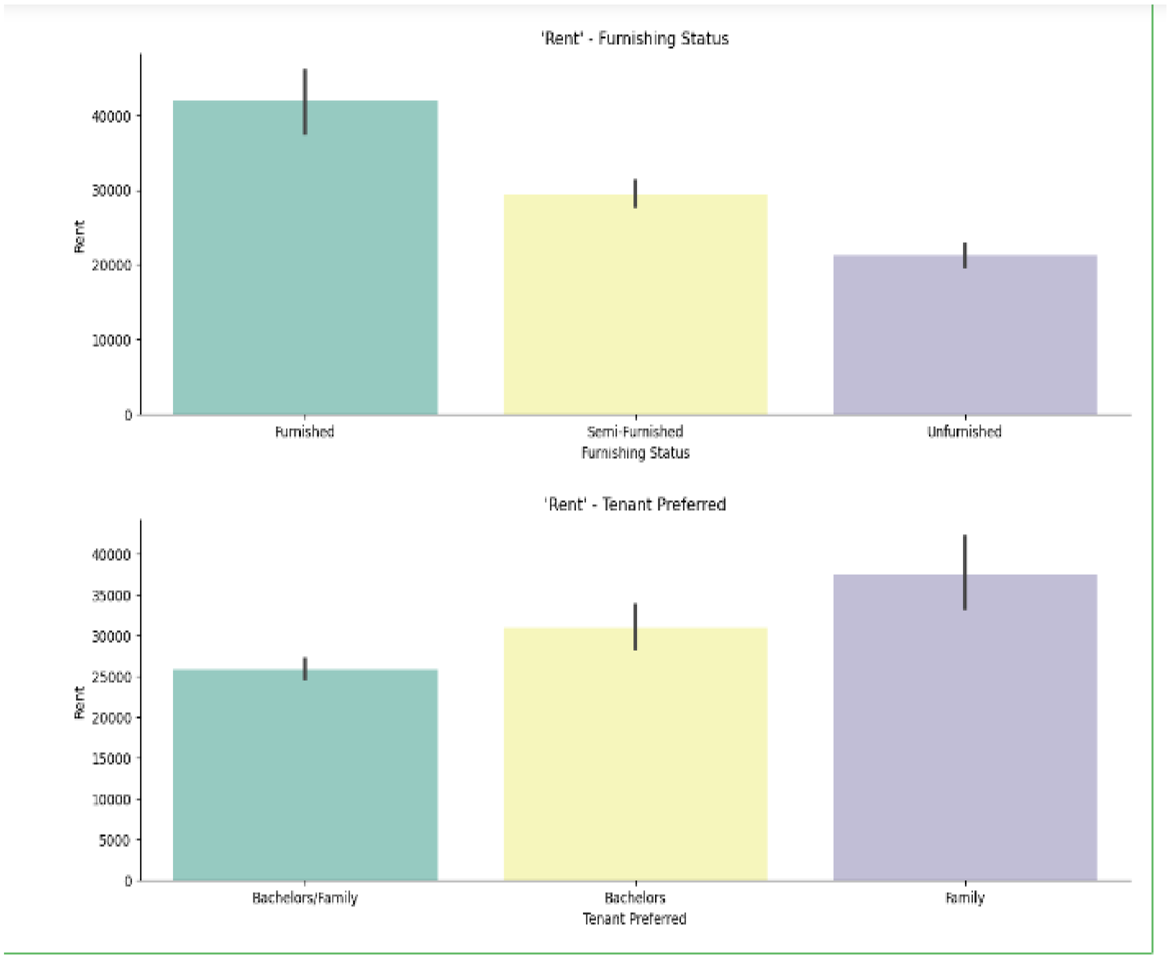
 sns.catplot(x=col, y="Rent", kind="bar", dodge=False, height = 4, aspect =3, data=df2, palette="Set3")

 plt.title("'Rent' - {}".format(col))





'Rent' - Furnishing Status



Machine learning and Predictive Analytics:

(i) Prepare the data

To prepare data for modeling, just remember **ASN (Assign,Split, Normalize).**

- **Assign** the features to X, & the last column to our classification predictor,y

  x=df3.iloc[:,:10]

  y=df3['Rent']
- **Split**: the data set into the Training set and Test set,

  from sklearn.model_selection import train_test_split

  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

(ii).Modeling /Training

We will now Train various Classification Models on the Training set & see which yields the highest accuracy. We will compare the accuracy of

Logistic Regression,

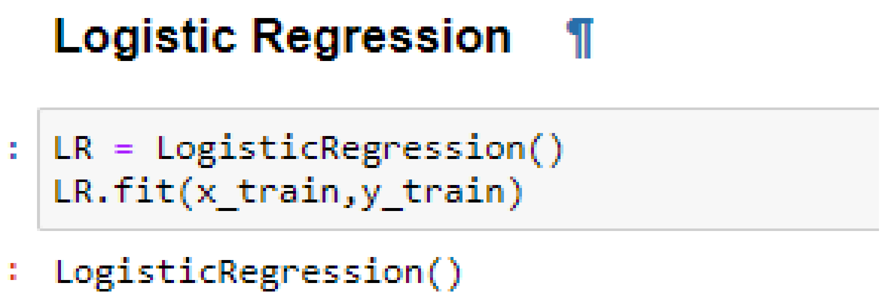
K-NN,

Decision Trees,

Random Forest.

Note: these are all supervised learning models.

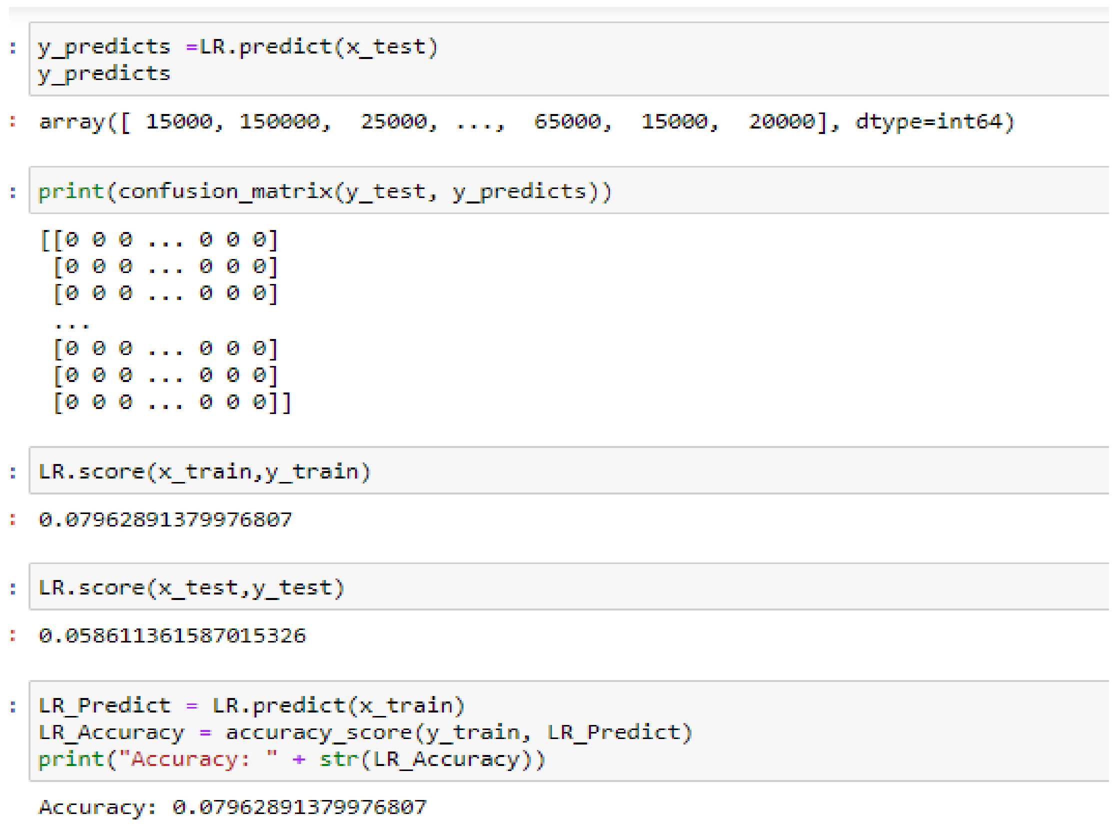
Model 1:Logistic Regression



Logistic Regression ¶

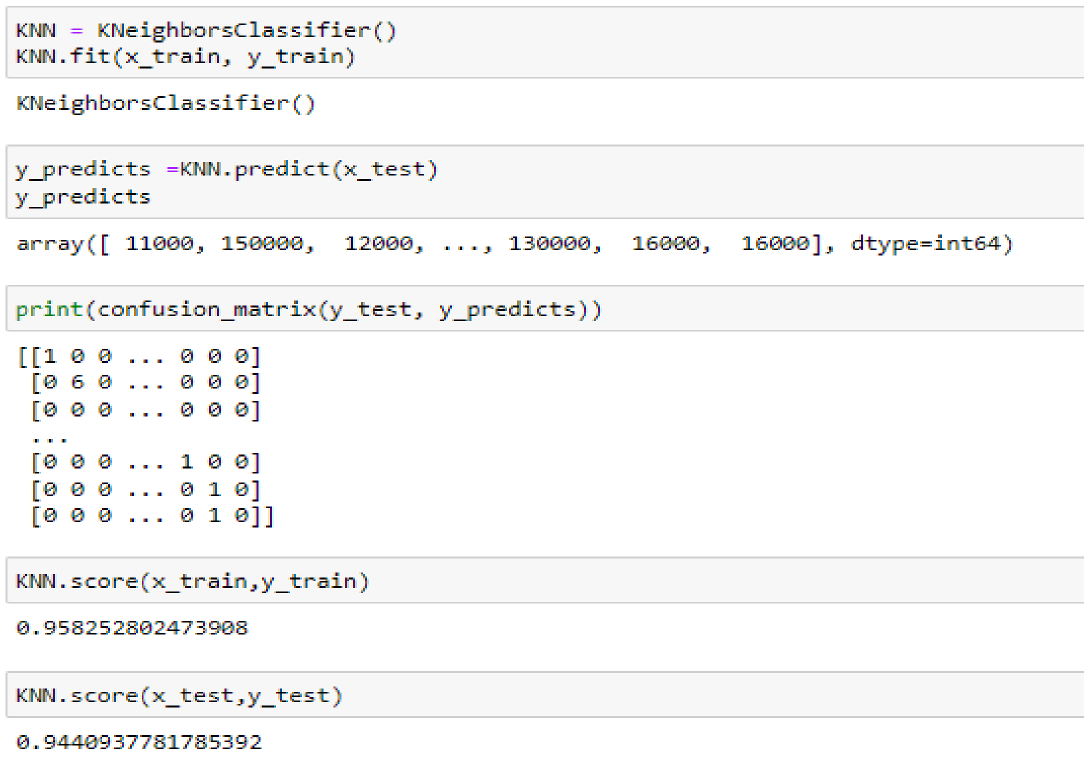
```
LR = LogisticRegression()
LR.fit(x_train,y_train)
```

```
LogisticRegression()
```

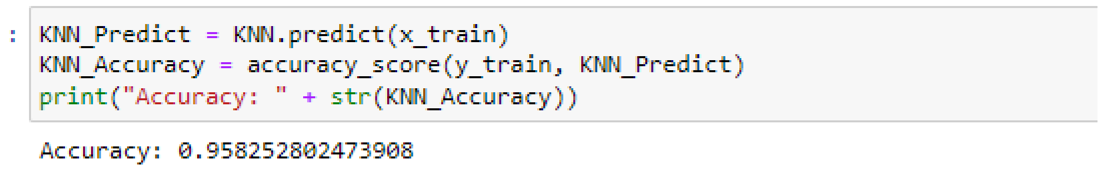


```
: y_predicts =LR.predict(x_test)
  y_predicts

: array([ 15000, 150000,  25000, ...,  65000,  15000,  20000], dtype=int64)

: print(confusion_matrix(y_test, y_predicts))

  [[0 0 0 ... 0 0 0]
   [0 0 0 ... 0 0 0]
   [0 0 0 ... 0 0 0]
   ...
   [0 0 0 ... 0 0 0]
   [0 0 0 ... 0 0 0]
   [0 0 0 ... 0 0 0]]

: LR.score(x_train,y_train)

: 0.07962891379976807

: LR.score(x_test,y_test)

: 0.058611361587015326

: LR_Predict = LR.predict(x_train)
  LR_Accuracy = accuracy_score(y_train, LR_Predict)
  print("Accuracy: " + str(LR_Accuracy))

  Accuracy: 0.07962891379976807
```

Model2: K-NN (K-Nearest Neighbors)

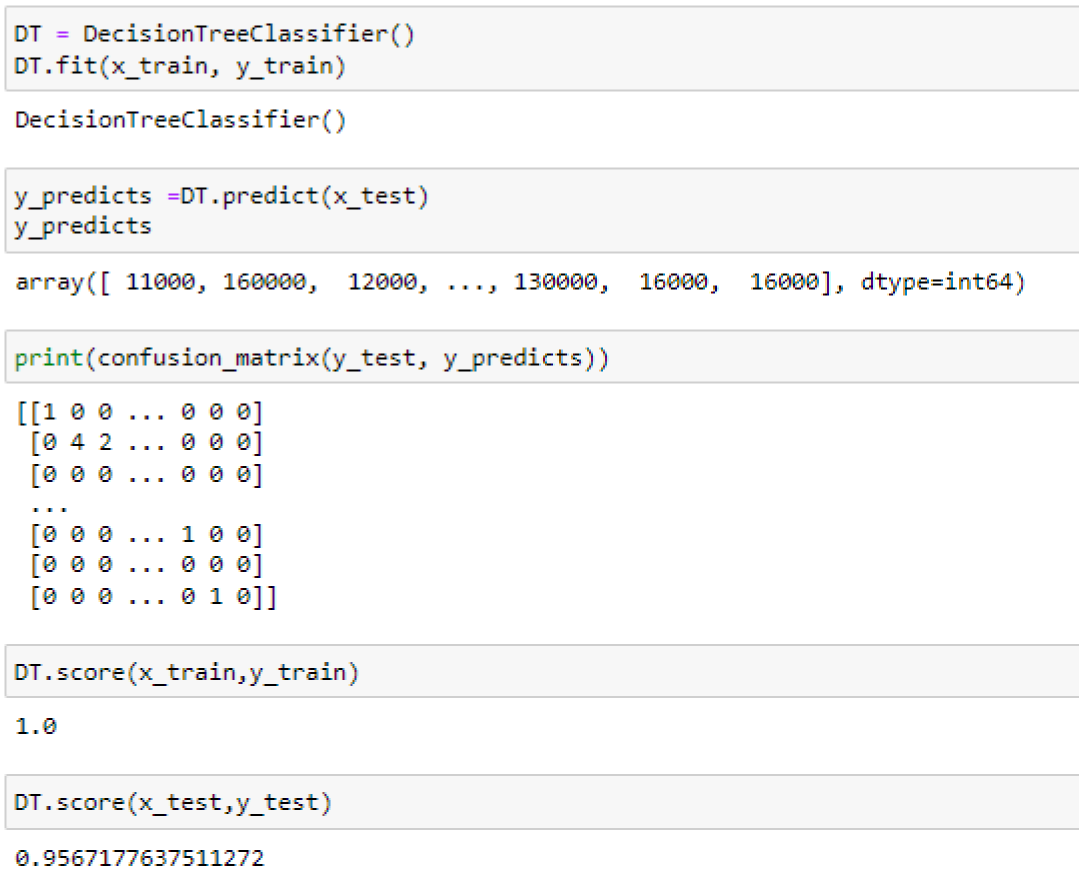


```
KNN = KNeighborsClassifier()
KNN.fit(x_train, y_train)

KNeighborsClassifier()

y_predicts =KNN.predict(x_test)
y_predicts

array([ 11000, 150000,  12000, ..., 130000,  16000,  16000], dtype=int64)

print(confusion_matrix(y_test, y_predicts))

[[1 0 0 ... 0 0 0]
 [0 6 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 1 0]]

KNN.score(x_train,y_train)

0.958252802473908

KNN.score(x_test,y_test)

0.9440937781785392
```

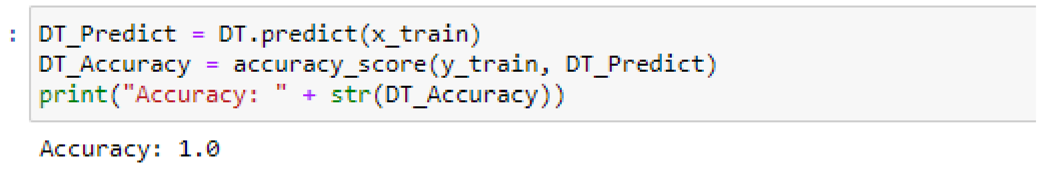


```
: KNN_Predict = KNN.predict(x_train)
  KNN_Accuracy = accuracy_score(y_train, KNN_Predict)
  print("Accuracy: " + str(KNN_Accuracy))

  Accuracy: 0.958252802473908
```

Model 3: Decision Tree Classifier

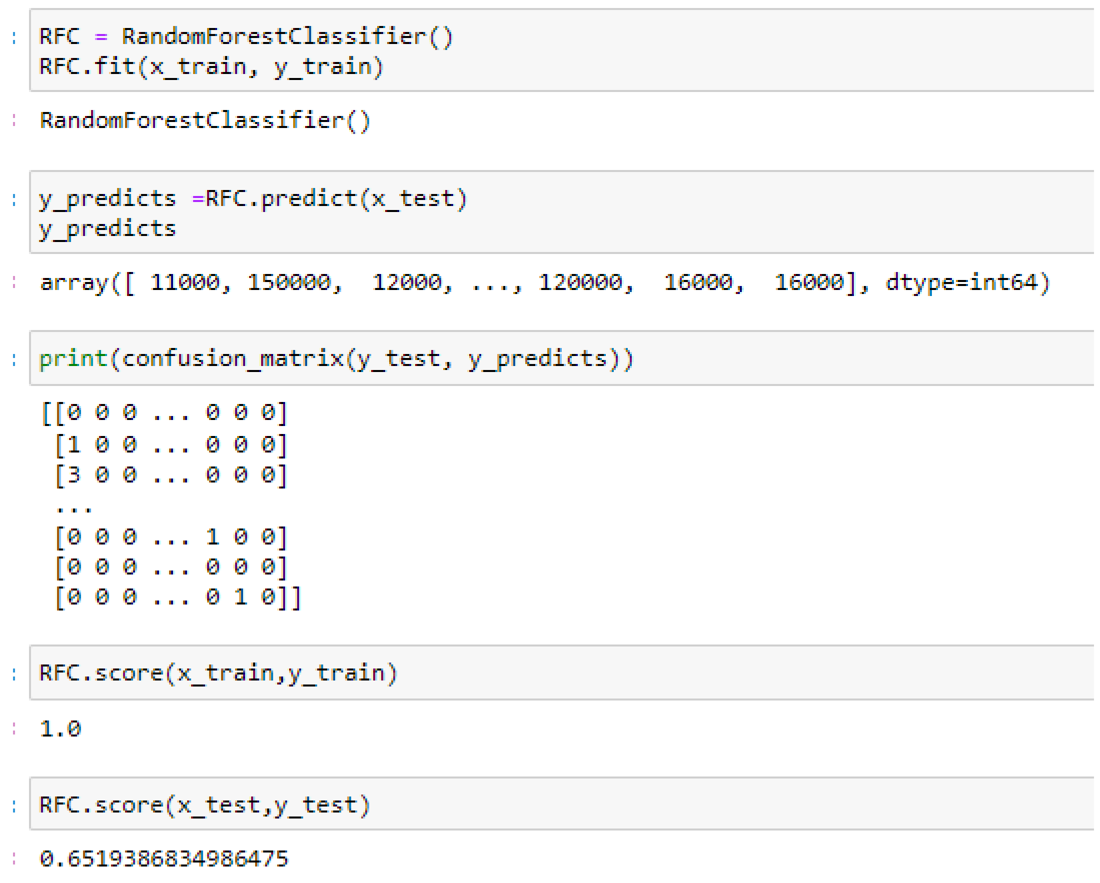


```
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)

DecisionTreeClassifier()

y_predicts =DT.predict(x_test)
y_predicts

array([ 11000, 160000,  12000, ..., 130000,  16000,  16000], dtype=int64)

print(confusion_matrix(y_test, y_predicts))

[[1 0 0 ... 0 0 0]
 [0 4 2 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]

DT.score(x_train,y_train)

1.0

DT.score(x_test,y_test)

0.9567177637511272
```



```
: DT_Predict = DT.predict(x_train)
  DT_Accuracy = accuracy_score(y_train, DT_Predict)
  print("Accuracy: " + str(DT_Accuracy))

  Accuracy: 1.0
```

Model 4: Random Forest



```
RFC = RandomForestClassifier()
RFC.fit(x_train, y_train)
```

```
RandomForestClassifier()
```

```
y_predicts =RFC.predict(x_test)
y_predicts
```

```
array([ 11000, 150000,  12000, ..., 120000,  16000,  16000], dtype=int64)
```

```
print(confusion_matrix(y_test, y_predicts))
```

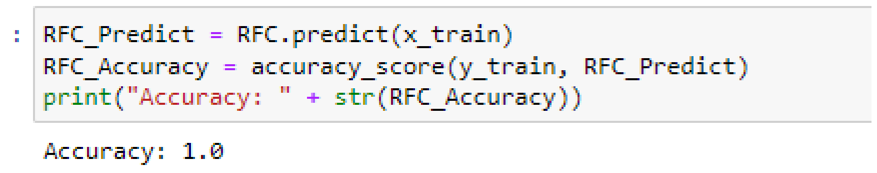
```
[[0 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [3 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]
```

```
RFC.score(x_train,y_train)
```

```
1.0
```

```
RFC.score(x_test,y_test)
```

```
0.6519386834986475
```



```
RFC_Predict = RFC.predict(x_train)
RFC_Accuracy = accuracy_score(y_train, RFC_Predict)
print("Accuracy: " + str(RFC_Accuracy))
```

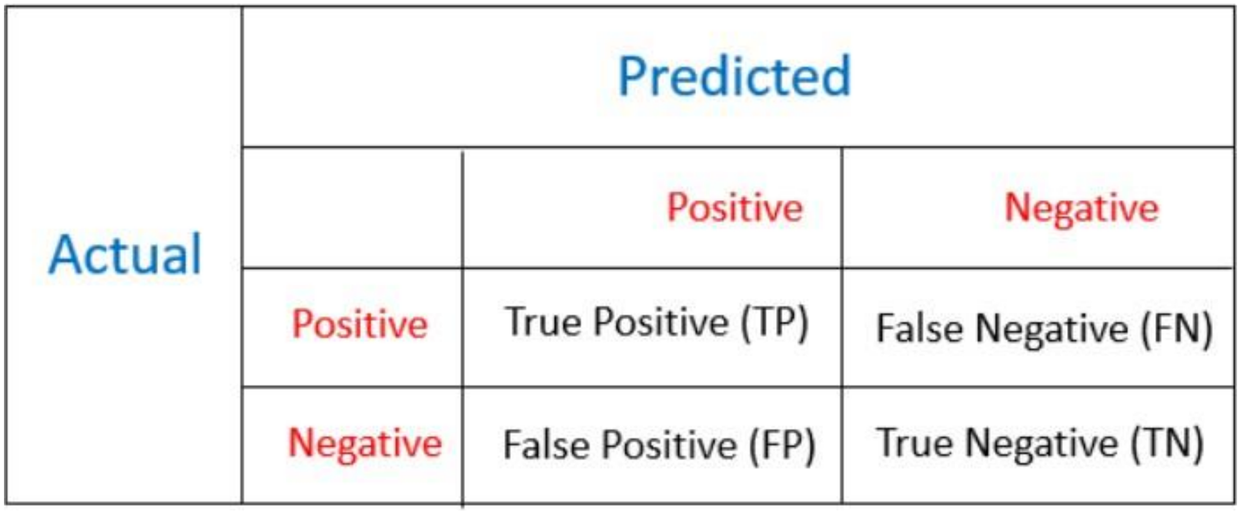
```
Accuracy: 1.0
```

On comparing the above 4 models, we can conclude that Random Forest and Decision tree yields the highest accuracy. With an accuracy of 100%.

Measuring Model Performance

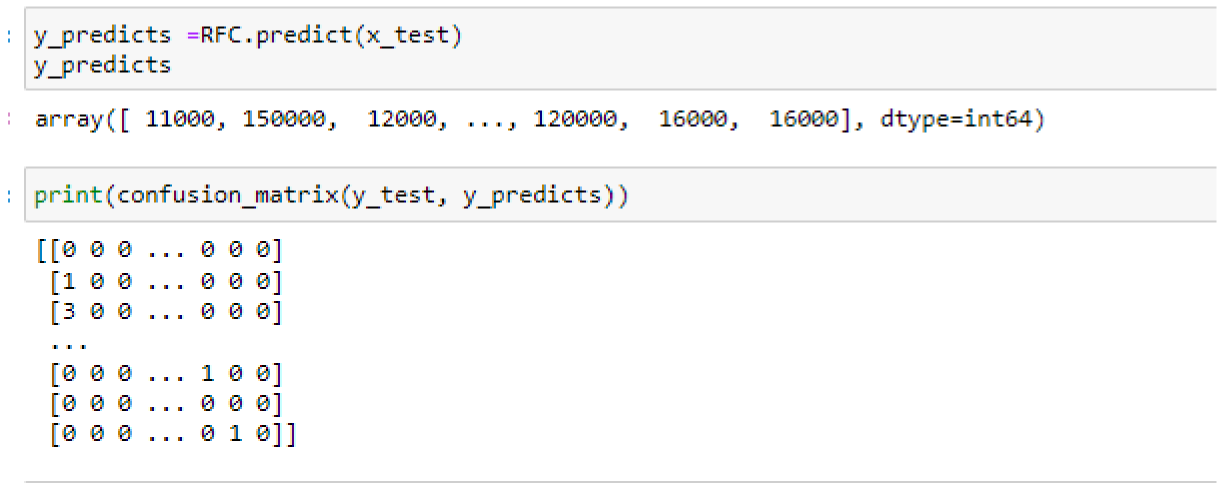
While there are other ways of measuring model performance (precision, recall, F1 Score, ROC Curve, etc), let's keep this simple and use accuracy as our metric. To do this are going to see how the model performs on new data (test set)

Accuracy is defined as: (fraction of correct predictions): correct predictions / total number of data points



		Predicted	
Actual		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Making the Confusion Matrix



```
y_predicts =RFC.predict(x_test)
y_predicts
```

```
array([ 11000, 150000,  12000, ..., 120000,  16000,  16000], dtype=int64)
```

```
print(confusion_matrix(y_test, y_predicts))
```

```
[[0 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [3 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]
```

Note: A good rule of thumb is that any accuracy above 70% is considered good, but be careful because if your accuracy is extremly high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!

Chapter-IV

Analysis of the Result

Machine Learning Models	Accuracy
Logistic Regression	74%
KNN	95%
Decision Tree	100%
Random Forest	100%

.

An accuracy of 100% in a Decision Tree Classifier and Random Forest for house price prediction sounds impressive. An accuracy score of this magnitude suggests that the classifier is performing well in correctly classifying the price of house based on the input features used.

However, it's important to note that the accuracy score alone may not provide a complete picture of the model's performance. It is essential to consider other evaluation metrics, such as precision, recall, and F1 score, to get a more comprehensive assessment of the classifier's effectiveness.

Additionally, the accuracy achieved by a model can be influenced by factors such as the quality and representativeness of the training data, the choice of features, and the preprocessing steps applied to the data. It is crucial to ensure that the dataset used for training and evaluation is representative of the real-world scenarios and that the model's performance is validated on unseen data to gauge its generalization capabilities.

Chapter-V

Conclusions

- Our Random Forest algorithm and Decision Tree classifier yields the highest accuracy, 100%. Any accuracy above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!
- Our machine learning algorithm can now classify the price of houses.

References

References

1) Housing Price Prediction via Improved Machine Learning Techniques by Quang Truong, Minh Nguyen, Hy Dang, Bo Mei, Procedia Computer Science, Volume 174, 2020, Pages 433-442.
2) House Price Prediction Using Machine Learning by AYUSH KUMAR TIWARI, International Journal of Mechanical Engineering, Vol. 7 No. 4 April, 2022.