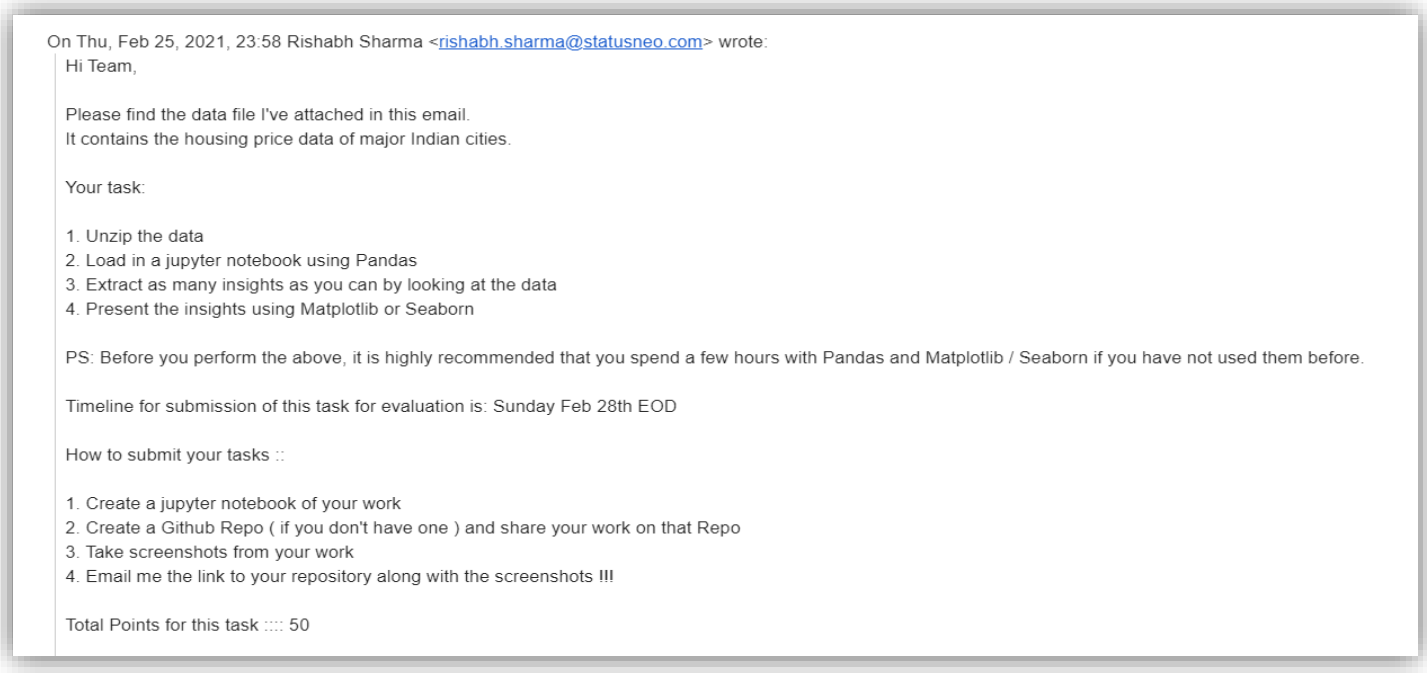
STATUSNEO Task-2



On Thu, Feb 25, 2021, 23:58 Rishabh Sharma <rishabh.sharma@statusneo.com> wrote:

Hi Team,

Please find the data file I've attached in this email.
It contains the housing price data of major Indian cities.

Your task:

1. Unzip the data
2. Load in a jupyter notebook using Pandas
3. Extract as many insights as you can by looking at the data
4. Present the insights using Matplotlib or Seaborn

PS: Before you perform the above, it is highly recommended that you spend a few hours with Pandas and Matplotlib / Seaborn if you have not used them before.

Timeline for submission of this task for evaluation is: Sunday Feb 28th EOD

How to submit your tasks ::

1. Create a jupyter notebook of your work
2. Create a Github Repo (if you don't have one) and share your work on that Repo
3. Take screenshots from your work
4. Email me the link to your repository along with the screenshots !!!

Total Points for this task :::: 50

On Thursday, February 25, 2021, you received a mail from StatusNeo which had a task you have to perform and submit within three days. That mail confused you a bit because it was full of technical words which you were not at all aware of. That mail was like a nightmare to you. Thoughts like “I am not from a technical background then why I got this task?”, “How will I be able to perform this task?”, “I should talk to sir about this and change the assigned task.” were making you uncomfortable. Minutes ago you were watching Netflix peacefully and you were “Lost in Space” and just after few minutes you were reading the mail and panicking.

Just after this unsettling moment, you decided that you should go to sleep and should start the work from the next day (so after this decision you completed an entire season of “Lost in space” and then slept well).

Next morning, you woke up and started working on the task. You pressed the power button of your laptop and then plugged in your laptop’s charger. After all this you opened your Google chrome browser and you typed “How to complete the task-1 of status neo?” and first link which appeared on the screen was just like a blessing to you. It was a step by step guide to perform the task which was written By Sarthak Saraiya, a B.tech Computer Science student from UPES Dehradun.

So without wasting time you started following each and every steps of that particular blog

STEP-1 (Analysing and Understanding the Task)

From the mail, download the zip file. We have to unzip the file to access the dataset provided to us.
To unzip (extract) files or folders from a zipped folder

1. Locate the zipped folder that you want to unzip (extract) files or folders from.
2. Do one of the following:
3. To unzip a single file or folder, open the zipped folder, then drag the file or folder from the zipped folder to a new location.
4. To unzip all the contents of the zipped folder, press and hold (or right-click) the folder, select **Extract All**, and then follow the instructions.

After successfully extracting the files, study the files provided to you to have a deeper understanding of the DATA.

STEP-2 (Studying a Bit about Python Language)

- **Python** is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
  Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

- Understanding the basics of any programming language is a very important task. To study python there are many resources available online. Few sites from which you can learn the key concepts of python are listed below.
  1) w3schools
  2) GeekForGeeks
  3) Tutorial point
  4) Python.org
  5) Codeacademy
  6) Programiz.

STEP-3 (Installing & Getting friendly with the Jupyter Notebook)

Installing the classic Jupyter Notebook interface

This section includes instructions on how to get started with **Jupyter Notebook**. But there are multiple Jupyter user interfaces one can use, based on their needs. Please checkout the list and links below for additional information and instructions about how to get started with each of them.

This information explains how to install the Jupyter Notebook and the IPython kernel.

Prerequisite: Python

While Jupyter runs code in many programming languages, **Python** is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook.

We recommend using the Anaconda distribution to install Python and Jupyter. We'll go through its installation in the next section.

Installing Jupyter using Anaconda and conda

For new users, we **highly recommend** installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook. To run the notebook:

```
jupyter notebook
```

Alternative for experienced Python users: Installing Jupyter with pip

Important

Jupyter installation requires Python 3.3 or greater, or Python 2.7. IPython 1.x, which included the parts that later became Jupyter, was the last version to support Python 3.2 and 2.6.

As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Then install the Jupyter Notebook using:

```
pip3 install jupyter
```

(Use `pip` if using legacy Python 2.)

STEP-4 (Executing the Task)

- Import all the important libraries/modules of python which will be useful to complete the task. The syntax to import libraries is:

  PANDAS:

  ```
  Import pandas as pd
  ```

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science. (Here pd is the short form which we will be using throughout the code instead of writing the complete word 'Pandas')

NUMPY:

```
Import numpy as np
```

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

MATPLOTLIB:

```
Import matplotlib.pyplot as plt
```

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

SEABORN:

```
Import seaborn as sns
```

Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

- After installing the necessary libraries/modules in python, our next job is to import the **CSV** files (A Comma-Separated Values **file** is a text **file** that has a specific **format** which allows data

```
data_bangalore = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Bangalore.csv')
data_chennai = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Chennai.csv')
data_delhi = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Delhi.csv')
data_hyderabad = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Hyderabad.csv')
data_kolkata = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Kolkata.csv')
data_mumbai = pd.read_csv (r'C:\\Users\\hp\\Downloads\\data\\data\\Mumbai.csv')
```

to be saved in a table structured **format**) which was provided to you in the zip file format through the mail.

Here 'C:\\Users\\hp\\Downloads\\data\\data\\Bangalore.csv' is the location of the file whose file name is 'Bangalore.csv'. With the help of "pd.read_csv" the CSV file is imported in our code for performing the tasks and is stored in the dataframe (**DataFrame** is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used **pandas** object.) "data_bangalore". Same process is performed to import other CSV files as well.

- After successfully importing the CSV files and storing it as a dataframe, it's time for us to perform basic operations on the data frame. These operations will help you to display the dataset on your python notebook. The basics operations which we have to perform on the dataset includes:

HEAD():

```
data_bangalore.head()
```

	Price	Area	Location	No. of Bedrooms	Resale	MaintenanceStaff	Gymnasium	SwimmingPool	LandscapedGardens	JoggingTrack	...	LiftAvailable	BED
0	30000000	3340	JP Nagar Phase 1	4	0	1	1	1	1	1	...	1	0
1	7888000	1045	Dasarahalli on Tumkur Road	2	0	0	1	1	1	1	...	1	0
2	4866000	1179	Kannur on Thanisandra Main Road	2	0	0	1	1	1	1	...	1	0
3	8358000	1675	Doddanekundi	3	0	0	0	0	0	0	...	1	0
4	6845000	1670	Kengeri	3	0	1	1	1	1	1	...	1	0

5 rows × 40 columns

`data_bangalore.head()` will display first five rows and all the columns of the bangalore dataset. "5 rows x 40 columns" means that there are five rows displayed here and total 40 columns.

VALUE_COUNTS():

```
data_bangalore['Location'].value_counts()
```

```
Electronic City Phase 2     232
RR Nagar                    217
Begur                       186
Varthur                     168
Kumaraswamy Layout          154
                           ...
Dodda Banaswadi               1
K P Main Road                 1
Sanjay Nagar                  1
Nagavara                      1
Nirman Layout                 1
Name: Location, Length: 302, dtype: int64
```

`data_bangalore['Location'].value_counts()` will select the location column and will display the frequency of the locations which are present in the dataset. For ex- 'Electronic City Phase 2' has appeared for a total of 232 times in the location column.

DTYPES():

```
data_bangalore.dtypes
```

```
Price                 int64
Area                  int64
Location             object
No. of Bedrooms       int64
Resale                int64
MaintenanceStaff      int64
Gymnasium             int64
```

`data_bangalore.dtypes` will give the datatype of all the columns. For ex- 'Price' is of int64 type, 'Location' is of object type etc.

SHAPE():

```
data_bangalore.shape
```

```
(6207, 40)
```

`data_bangalore.shape` will give the shape of the dataframe. (6207,40) here 6207 is the number of rows and 40 is the total number of columns present in the dataframe.

DESCRIBE():

```
data_bangalore.describe()
```

	Price	Area	No. of Bedrooms	Resale	MaintenanceStaff	Gymnasium	SwimmingPool	LandscapedGardens	JoggingTrack	RainWaterHarv
count	6.207000e+03	6207.000000	6207.000000	6207.000000	6207.000000	6207.000000	6207.000000	6207.000000	6207.000000	6207.0
mean	1.058510e+07	1526.094248	2.556952	0.078782	6.208797	6.461576	6.436121	6.382471	6.396649	6.3
std	1.410943e+07	764.845609	0.694300	0.269420	4.126883	3.752421	3.792567	3.875271	3.853661	3.8
min	2.000000e+06	415.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	5.000000e+06	1110.000000	2.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.0
50%	7.368000e+06	1340.000000	3.000000	0.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.0
75%	1.070000e+07	1662.500000	3.000000	0.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.0
max	3.000000e+08	9900.000000	7.000000	1.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.0

`data_bangalore.describe()` here **describe**() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

ISNULL():

```
data_bangalore.isnull().sum()
```

```
Price                0
Area                 0
Location             0
No. of Bedrooms      0
Resale               0
MaintenanceStaff     0
Gymnasium            0
SwimmingPool         0
LandscapedGardens    0
```

`data_bangalore.isnull.sum()` returns the number of missing values in the data set.

TOFRAME ():

```
data_bangalore['No. of Bedrooms'].value_counts().to_frame()
```

	No. of Bedrooms
3	2799
2	2750
4	402
1	230
5	22
7	2
6	2

`data_bangalore['No. of Bedrooms'],value_counts().to_frame()` function to convert the given series object to a dataframe. Here it is also showing (with the help of value_counts) that how many 3 bhk flats (2799) or 2 bhk flats (2750) are there in the dataframe.

- Now as we have performed the basic operations on the dataframe, it is time for us to visualize the data and find some insights from the data. To find some insights and plot them we should study about the functions that we will be using.

SEABORN:

Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics. Here we have imported seaborn as **sns**.

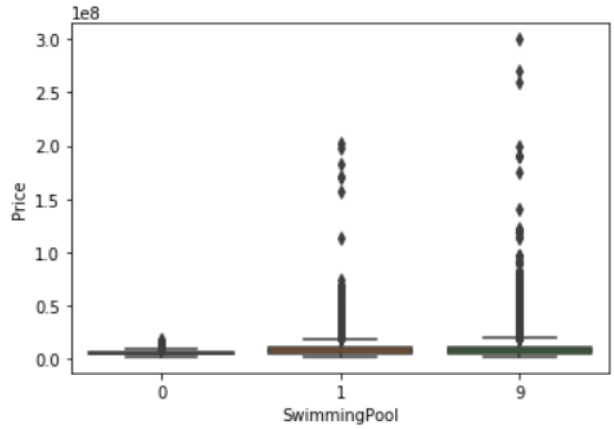
BOXPLOT:

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

Now with the help of seaborn library, we will plot a BOXPLOT to determine whether houses with a swimming pool or without a swimming pool have more price outliers (We performed same steps to compare 'Price' with Gymnasium and PowerBackup in the later stage of the code)

```
sns.boxplot(x='SwimmingPool', y='Price', data=data_bangalore)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13602ff7dc8>
```

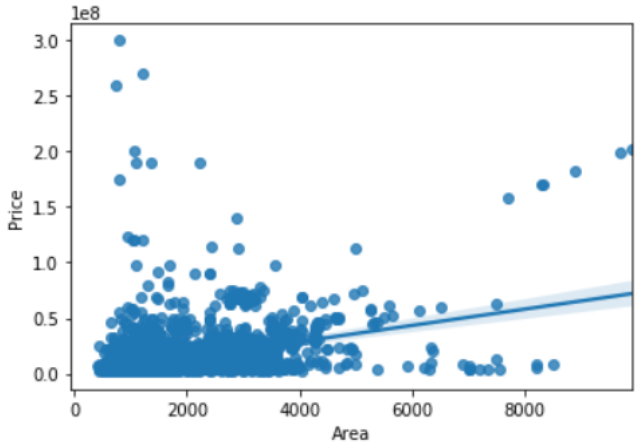


REGPLOT:

This method is used to plot data and a linear regression model fit. There are a number of mutually exclusive options for estimating the regression model.

```
sns.regplot(x='Area', y='Price', data=data_bangalore)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13603e53dc8>
```



Here using regplot()in the seaborn library to determine if the feature ‘Area’ is negatively or positively correlated with price in data_bangalore dataset. (We performed the same steps to check that if ‘Resale’ is positively or negatively related to ‘Price’)

CORR:

Pandas `dataframe.corr()` is used to find the pairwise correlation of all columns in the dataframe. Any na values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.

```
data_bangalore.corr()['Price'].sort_values()
```

```
Resale               -0.038024
School                0.028669
LiftAvailable         0.028814
ShoppingMall          0.029271
VaastuCompliant       0.029598
LandscapedGardens     0.029744
Hospital              0.029807
ATM                   0.029886
MultipurposeRoom      0.030135
Wifi                  0.030335
Wardrobe              0.030335
MaintenanceStaff      0.030497
Intercom              0.030951
```

Here we are using the Pandas method `corr()` to find the feature other than price that is most correlated with price.

LINEAR REGRESSION:

Linear Regression is usually the first machine learning algorithm that every data scientist comes across. It is a simple model but everyone needs to master it as it lays the foundation for other machine learning algorithms.

Where can Linear Regression be used?

It is a very powerful technique and can be used to understand the factors that influence profitability. It can be used to forecast sales in the coming months by analyzing the sales data for previous months. It can also be used to gain various insights about customer behaviour.

What is Linear Regression?

The objective of a linear regression model is to find a relationship between one or more features (independent variables) and a continuous target variable (dependent variable). When there is only feature it is called *Uni-variate* Linear Regression and if there are multiple features, it is called *Multi*ple Linear Regression.

`from sklearn.linear_model import LinearRegression` From the sklearn library we imported LinearRegression which will help us in finding the insights.

```
U = data_bangalore[['Area']]
V = data_bangalore['Price']
lm = LinearRegression()
lm
lm.fit(U,V)
lm.score(U,V)
```

```
0.1581063952035402
```

In ‘U’ we stored ‘Area’ and in ‘V’ we stored ‘Price’ and Fit a linear regression model to predict the 'Price' using the feature 'Area'

Later we used fit() function to fit U & V and score() Returns the coefficient of determination (R^2). A measure of how well observed outcomes are replicated by the model, as the proportion of total variation of outcomes explained by the model.

```
features =["Resale", "LiftAvailable","No. of Bedrooms" ,"LandscapedGardens" ,"MaintenanceStaff" ,"24X7Security" ,"CarParking"
X = data_bangalore[features]
Y = data_bangalore['Price']
lm.fit(X,Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
lm.score(X,Y)
```

```
0.06646153856864256
```

Later we fit a linear regression model to predict the 'price' using the list of features: Resale, LiftAvailable, No. of Bedrooms, LandscapedGardens, MaintainanceStaff, 24x7Security, Carparking etc.

MODEL EVALUATION:

Model evaluation aims to estimate the generalization accuracy of a **model** on future (unseen/out-of-sample) data. Methods for **evaluating** a **model's** performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the **model**) to **evaluate model** performance.

```
features =["Resale", "LiftAvailable","No. of Bedrooms" ,"LandscapedGardens" ,"MaintenanceStaff" ,"24X7Security" ,"CarParking"
X = data_bangalore[features]
Y = data_bangalore['Price']

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.15, random_state=1)


print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 932
number of training samples: 5275
```

We will split the data into training and testing set with the help of `train_test_split()` function.

NOTE- Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the the data set into two sets: a training set and a testing set. 80% for training, and 20% for testing. You *train* the model using the training set. You *test* the model using the testing set. *Train* the model means *create* the model. *Test* the model means test the accuracy of the model.

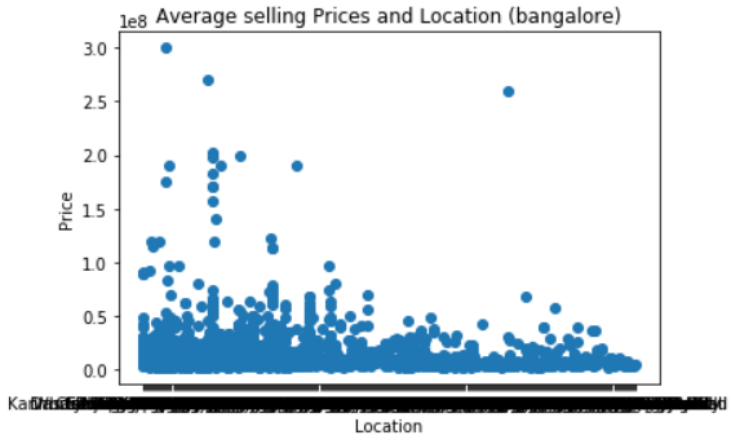
DATA EXPLORATION:

```
# TODO: Minimum price of the data
minimum_price = np.min(data_bangalore['Price'])

# TODO: Maximum price of the data
maximum_price = np.max(data_bangalore['Price'])

# TODO: Mean price of the data
mean_price = np.mean(data_bangalore['Price'])

# TODO: Median price of the data
median_price = np.median(data_bangalore['Price'])

# TODO: Standard deviation of prices of the data
std_price = np.std(data_bangalore['Price'])

# Show the calculated statistics
print("Statistics for Bangalore housing dataset:\n")
print("Minimum price: {}".format(minimum_price))
print("Maximum price: {}".format(maximum_price))
print("Mean price: {}".format(mean_price))
print("Median price {}".format(median_price))
print("Standard deviation of prices: {}".format(std_price))
```

```
Statistics for Bangalore housing dataset:

Minimum price: 2000000
Maximum price: 300000000
Mean price: 10585104.825519575
Median price 7368000.0
Standard deviation of prices: 14108297.374997266
```

Later we performed the data exploration task on the data_bangalore dataframe and found the minimum price, maximum price, mean price, median price and standard price of the data. We performed the same steps on Mumbai, Kolkata and delhi dataset to find out the insights.

```
fig=plt.figure()
ax=fig.add_subplot(1, 1, 1)
ax.scatter(data_bangalore['Location'], data_bangalore['Price'])
#Lables & Title
plt.title("Average selling Prices and Location (bangalore)")
plt.xlabel("Location")
plt.ylabel("Price")
plt.show()
```



We later plotted a scatter plot comparing the Average Selling price and the Location of the property.