**CLOUD PROJECT - TEAM 2**

**(EDA with Python)**

**Submitted by:**

**Movva Sri Sai Haravali**

**Gourav Meenia**

**Siddharth**

**Tanya Mishra**

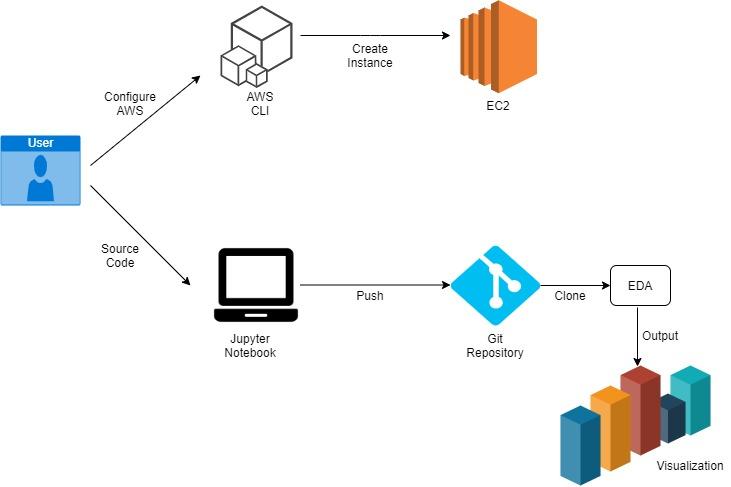
**Ambika Sharma**

**Yashvi Malhan**

**Introduction:**

Housing prices are an important reflection of the economy. Housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to visualize the given numerical data using EDA techniques with python.

**Architecture Diagram:**

****

# **Launching an EC2 instance using AWS CLI:**

### **Creating a VPC**

### aws ec2 create-vpc --cidr-block 10.0.0.0/16

### **Creating Subnets**

aws ec2 create-subnet --vpc-id <vpcId> --cidr-block 10.0.1.0/24

aws ec2 create-subnet --vpc-id <vpcId> --cidr-block 10.0.0.0/24

### **Creating Internet Gateway**

aws ec2 create-internet-gateway

aws ec2 attach-internet-gateway --vpc-id <vpcId> --internet-gateway-id <InternetGatewayId>

### **Viewing the Route Table and Subnets**

aws ec2 describe-route-tables --route-table-id <RouteTableId>

aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpcId>"

--query "Subnets[\*].{ID:SubnetId,CIDR:CidrBlock}"

### **Associating Route Table and modifying subnet**

aws ec2 associate-route-table --subnet-id <SubnetId> --route-table-id <RouteTableId>

aws ec2 modify-subnet-attribute --subnet-id <SubnetId> --map-public-ip-on-launch

### 

### 

### **Creating Key Pair and Security Group**

aws ec2 create-key-pair --key-name AWS-Keypair --query "KeyMaterial"

--output text > "C:\AWS\AWS\_Keypair.pem"

aws ec2 create-security-group --group-name <security-group-name> --description "<description>"

--vpc-id <vpcId>

aws ec2 authorize-security-group-ingress --group-id <GroupId>

--protocol tcp --port 22 --cidr 0.0.0.0/0

### **Running the EC2 Instance**

aws ec2 run-instances --image-id <ami-id> --count 1 --instance-type t2.micro

--key-name <Keypair-name> --security-group-ids <SecurityGroupId>

--subnet-id <SubnetId>

### **Viewing the Instance**

aws ec2 describe-instances --instance-id <InstanceId>

**Methodology:**

TOOLS USED:

* Python 3.7
* Pandas (Library)
* Numpy (Library)
* Matplotlib (Library)
* Seaborn(Library)
* Jupyter Notebook
* AWS CLI
* Git Bash

**GIT Commands :**

GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features.

* Git commands to upload files in git repository using GitBash:

1. git init
2. git add .
3. git commit -m "first commit"
4. git branch -M main
5. git remote add origin

<https://github.com/HaravaliMovva/Cloud-group2-project.git>

1. git push -u origin main

* Git commands to clone the repository :

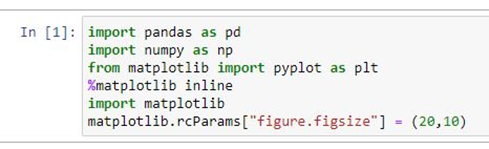
1. git clone <https://github.com/HaravaliMovva/Cloud-group2-project.git>
2. cd “projectname”/
3. git pull

**EDA with python(Visualisation) :**

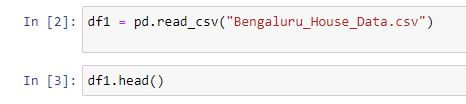
**STEPS OF IMPLEMENTATION:**

In this we will load Bangalore home prices data into pandas dataframe and then handle NA values. We will then remove some unnecessary features and also normalize property size. We will convert the range of property size (such as 2100-3250) into an average of min and max.

**a) IMPORT**

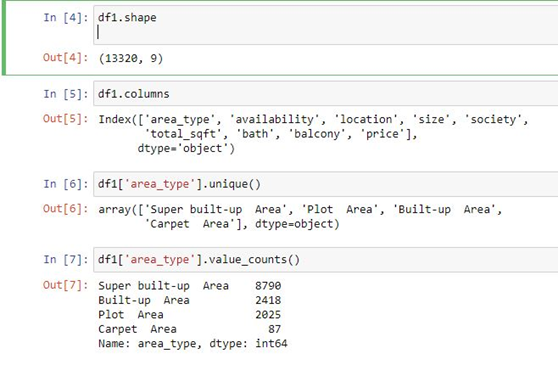


**b) READ THE DATASET:**

****

****

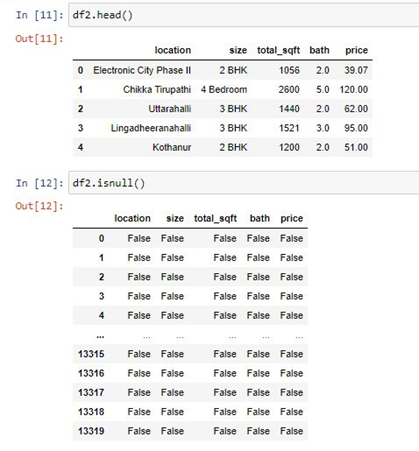
**c) ANALYZE THE DATA FRAME:**

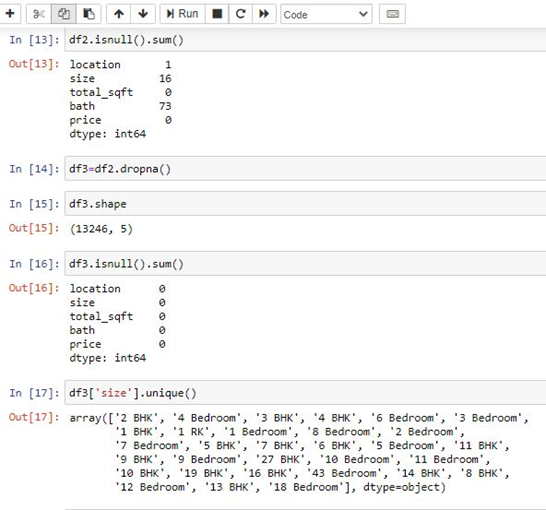
****

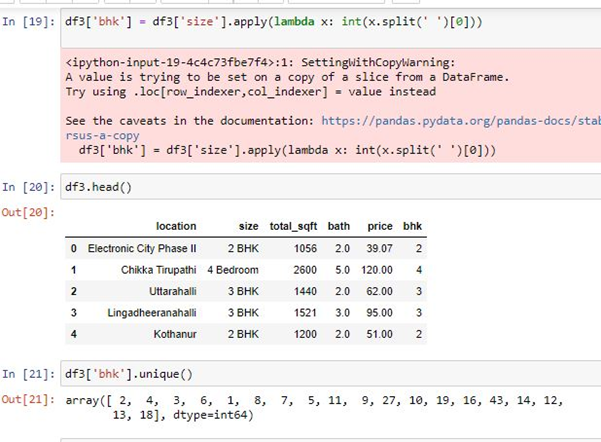
**d) DROP UNWANTED COLUMNS**

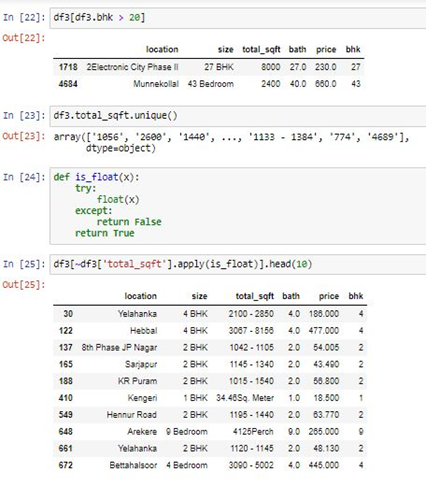
****

**e) FURTHER DATA CLEANING**

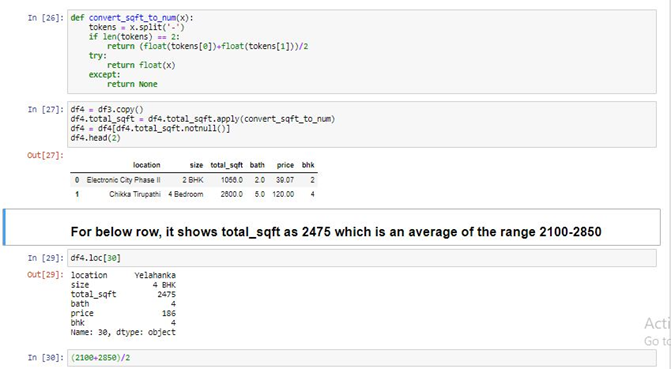
****





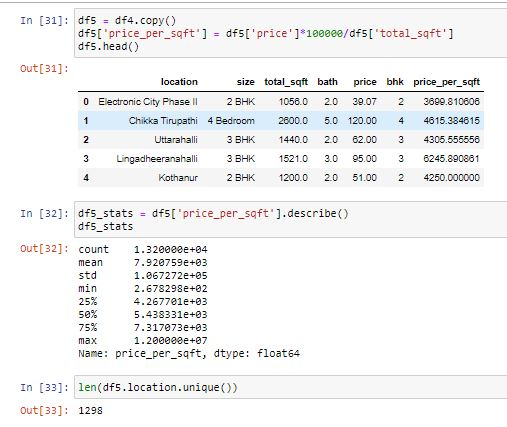
****

#### **Above shows that total\_sqft can be a range (e.g. 2100-2850). For such case we can just take average of min and max value in the range. There are other cases such as 34.46Sq. Meter which one can convert to square ft using unit conversion. We just dropped such corner cases to keep things simple**

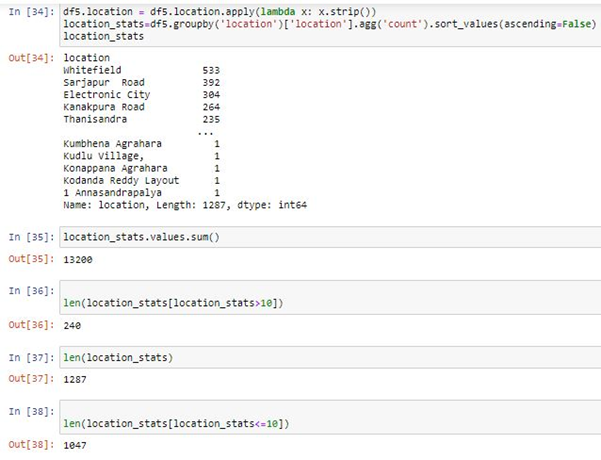


**f) FEATURE ENGINEERING**

*Add new feature called price per square feet*

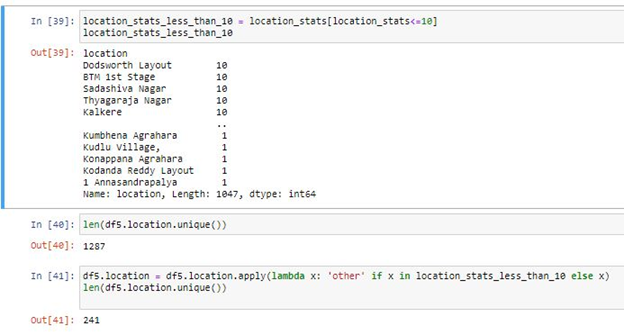


***Exploring the Location Column***

******

***Dimensionality Reduction***

***Any location having less than 10 data points should be tagged as "other" location. This way the number of categories can be reduced by a huge amount. Later on when we do one hot encoding, it will help us with having fewer dummy columns.***

******

******

***Outlier Removal Using Business Logic***

*As a data scientist when you have a conversation with your business manager (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 a 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 thresold per bhk to be 300 sqft.*

*df5[df5.total\_sqft/df5.bhk<300].head()*

| *location* | *size* | *total\_sqft* | *bath* | *price* | *bhk* | *price\_per\_sqft* |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *9* | *other* | *6 Bedroom* | *1020.0* | *6.0* | *370.0* | *6* | *36274.509804* |
| *45* | *HSR Layout* | *8 Bedroom* | *600.0* | *9.0* | *200.0* | *8* | *33333.333333* |
| *58* | *Murugeshpalya* | *6 Bedroom* | *1407.0* | *4.0* | *150.0* | *6* | *10660.980810* |
| *68* | *Devarachikkanahalli* | *8 Bedroom* | *1350.0* | *7.0* | *85.0* | *8* | *6296.296296* |
| *70* | *other* | *3 Bedroom* | *500.0* | *3.0* | *100.0* | *3* | *20000.000000* |

*Df5.shape*

*(13200, 7)*

*df6 = df5[~(df5.total\_sqft/df5.bhk<300)]*

*df6.shape*

*12456, 7)*

*df6.price\_per\_sqft.describe()*

*count 12456.000000*

*mean 6308.502826*

*std 4168.127339*

*min 267.829813*

*25% 4210.526316*

*50% 5294.117647*

*75% 6916.666667*

*max 176470.588235*

*Name: price\_per\_sqft, dtype: float64*

#### *Here we find that min price per sqft is 267 rs/sqft whereas max is 12000000, this shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation*

def remove\_pps\_outliers(df):

df\_out = pd.DataFrame()

for key, subdf in df.groupby('location'):

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

df7 = remove\_pps\_outliers(df6)

df7.shape

(10242, 7)

def plot\_scatter\_chart(df,location):

bhk2 = df[(df.location==location) & (df.bhk==2)]

bhk3 = df[(df.location==location) & (df.bhk==3)]

matplotlib.rcParams['figure.figsize'] = (15,10)

plt.scatter(bhk2.total\_sqft,bhk2.price,color='blue',label='2 BHK', s=50)

plt.scatter(bhk3.total\_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)

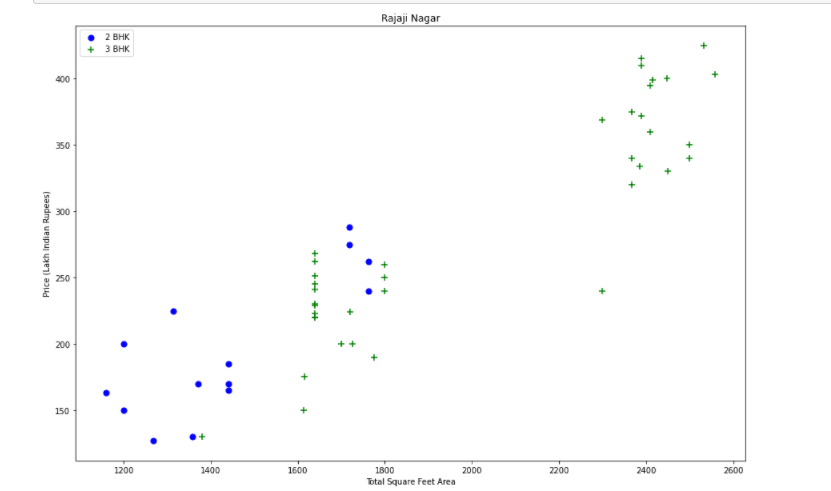
plt.xlabel("Total Square Feet Area")

plt.ylabel("Price (Lakh Indian Rupees)")

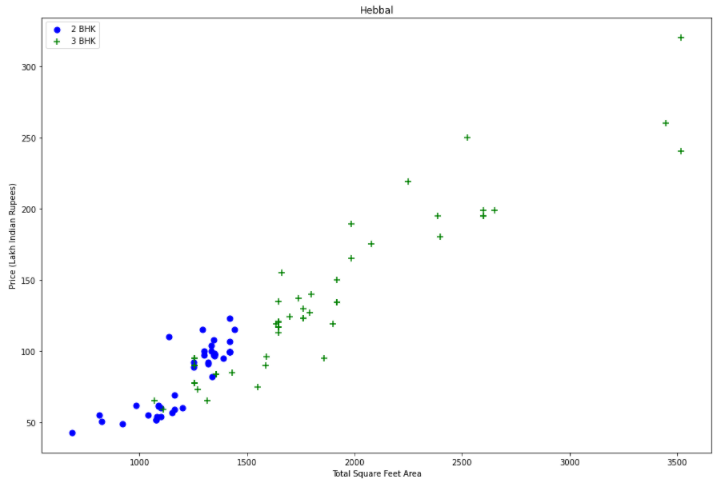
plt.title(location)

plt.legend()

plot\_scatter\_chart(df7,"Rajaji Nagar")



plot\_scatter\_chart(df7,"Hebbal")



### Now we can remove those 2 BHK apartments whose price\_per\_sqft is less than mean price\_per\_sqft of 1 BHK apartment

def remove\_bhk\_outliers(df):

exclude\_indices = np.array([])

for location, location\_df in df.groupby('location'):

bhk\_stats = {}

for bhk, bhk\_df in location\_df.groupby('bhk'):

bhk\_stats[bhk] = {

'mean': np.mean(bhk\_df.price\_per\_sqft),

'std': np.std(bhk\_df.price\_per\_sqft),

'count': bhk\_df.shape[0]

}

for bhk, bhk\_df in location\_df.groupby('bhk'):

stats = bhk\_stats.get(bhk-1)

if stats and stats['count']>5:

exclude\_indices = np.append(exclude\_indices, bhk\_df[bhk\_df.price\_per\_sqft<(stats['mean'])].index.values)

return df.drop(exclude\_indices,axis='index')

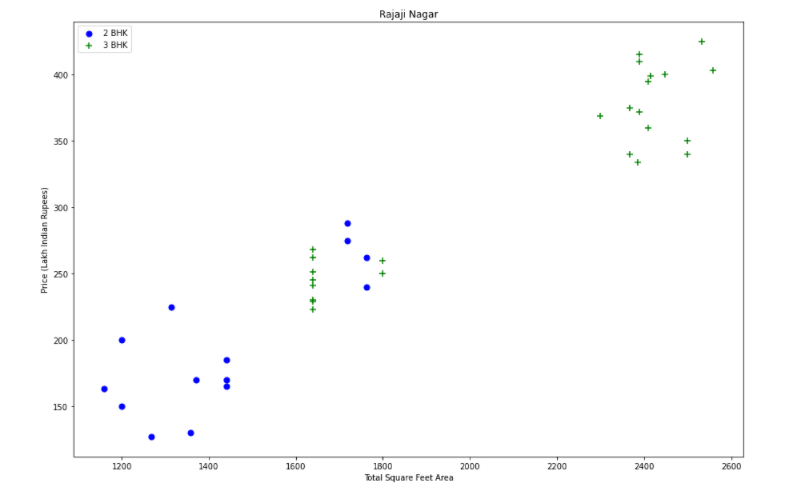
df8 = remove\_bhk\_outliers(df7)

# df8 = df7.copy()

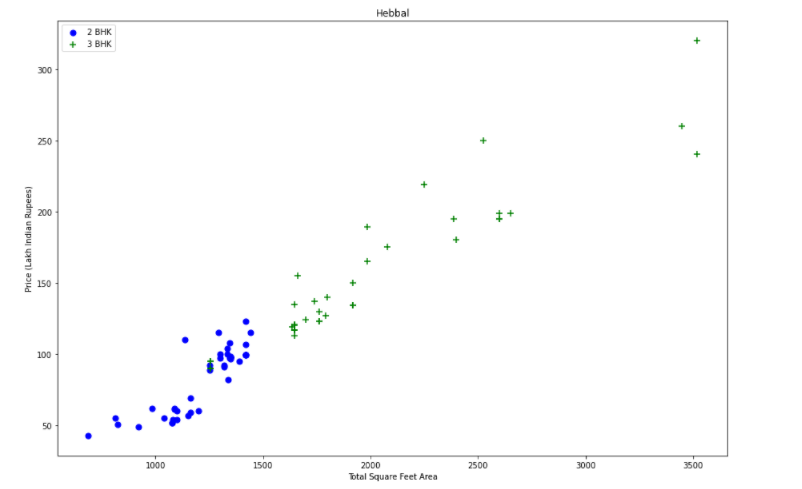
df8.shape

(7317, 7)

plot\_scatter\_chart(df8,"Rajaji Nagar")

******

***plot\_scatter\_chart(df8,"Hebbal")***

******

import matplotlib

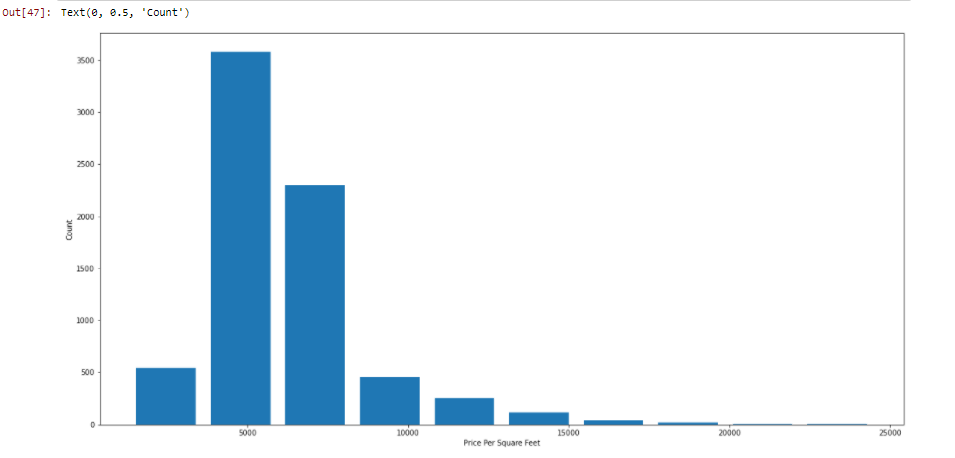
matplotlib.rcParams["figure.figsize"] = (20,10)

plt.hist(df8.price\_per\_sqft,rwidth=0.8)

plt.xlabel("Price Per Square Feet")

plt.ylabel("Count")

Text(0, 0.5, 'Count')



## **Outlier Removal Using Bathrooms Feature**

df8.bath.unique()

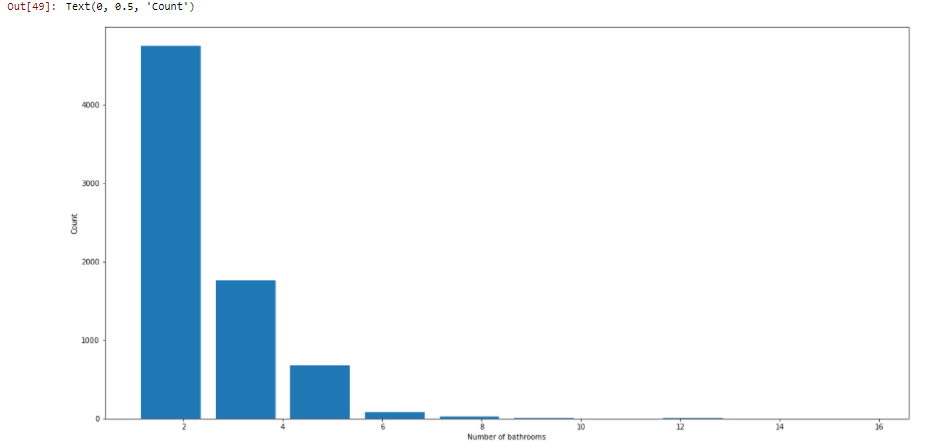
array([ 4., 3., 2., 5., 8., 1., 6., 7., 9., 12., 16., 13.])

plt.hist(df8.bath,rwidth=0.8)

plt.xlabel("Number of bathrooms")

plt.ylabel("Count")

Text(0, 0.5, 'Count')



df8[df8.bath>10]

|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **5277** | Neeladri Nagar | 10 BHK | 4000.0 | 12.0 | 160.0 | 10 | 4000.000000 |
| **8483** | other | 10 BHK | 12000.0 | 12.0 | 525.0 | 10 | 4375.000000 |
| **8572** | other | 16 BHK | 10000.0 | 16.0 | 550.0 | 16 | 5500.000000 |
| **9306** | other | 11 BHK | 6000.0 | 12.0 | 150.0 | 11 | 2500.000000 |
| **9637** | other | 13 BHK | 5425.0 | 13.0 | 275.0 | 13 | 5069.124424 |

## **It is unusual to have 2 more bathrooms than number of bedrooms in a home**

df8[df8.bath>df8.bhk+2]

|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1626** | Chikkabanavar | 4 Bedroom | 2460.0 | 7.0 | 80.0 | 4 | 3252.032520 |
| **5238** | Nagasandra | 4 Bedroom | 7000.0 | 8.0 | 450.0 | 4 | 6428.571429 |
| **6711** | Thanisandra | 3 BHK | 1806.0 | 6.0 | 116.0 | 3 | 6423.034330 |
| **8408** | other | 6 BHK | 11338.0 | 9.0 | 1000.0 | 6 | 8819.897689 |

### **Again the business manager has a conversation with you (i.e. a data scientist) that if you have 4 bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed**

df9 = df8[df8.bath<df8.bhk+2]

df9.shape

(7239, 7)

df9.head(2)

|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1st Block Jayanagar | 4 BHK | 2850.0 | 4.0 | 428.0 | 4 | 15017.543860 |
| **1** | 1st Block Jayanagar | 3 BHK | 1630.0 | 3.0 | 194.0 | 3 | 11901.840491 |

df10 = df9.drop(['size','price\_per\_sqft'],axis='columns')

df10.head(3)

|  | **location** | **total\_sqft** | **bath** | **price** | **bhk** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1st Block Jayanagar | 2850.0 | 4.0 | 428.0 | 4 |
| **1** | 1st Block Jayanagar | 1630.0 | 3.0 | 194.0 | 3 |
| **2** | 1st Block Jayanagar | 1875.0 | 2.0 | 235.0 | 3 |

### 

### **Use One Hot Encoding For Location**

dummies = pd.get\_dummies(df10.location)

dummies.head(20)

df11=pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')

df11.head()

df12 = df11.drop('location',axis='columns')

df12.head(2)

df12.shape

(7239, 244)

X = df12.drop(['price'],axis='columns')

X.head(3)

X.shape

(7239, 243)

y = df12.price

y.head(3)

0 428.0

1 194.0

2 235.0

Name: price, dtype: float64

len(y)

7239

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=10)

**from** **sklearn.linear\_model** **import** LinearRegression

lr\_clf = LinearRegression()

lr\_clf.fit(X\_train,y\_train)

lr\_clf.score(X\_test,y\_test)

0.8629132245229442

### **Use K Fold cross validation to measure accuracy of our LinearRegression model**

from sklearn.model\_selection import ShuffleSplit

from sklearn.model\_selection import cross\_val\_score

cv = ShuffleSplit(n\_splits=5, test\_size=0.2, random\_state=0)

cross\_val\_score(LinearRegression(), X, y, cv=cv)

array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])

##### ***We can see that in 5 iterations we get a score above 80% all the time. This is pretty good but we want to test a few other algorithms for regression to see if we can get even better score. We will use GridSearchCV for this purpose***

## **Find best model using GridSearchCV**

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import Lasso

from sklearn.tree import DecisionTreeRegressor

def find\_best\_model\_using\_gridsearchcv(X,y):

algos = {

'linear\_regression' : {

'model': LinearRegression(),

'params': {

'normalize': [**True**, **False**]

}

},

'lasso': {

'model': Lasso(),

'params': {

'alpha': [1,2],

'selection': ['random', 'cyclic']

}

},

'decision\_tree': {

'model': DecisionTreeRegressor(),

'params': {

'criterion' : ['mse','friedman\_mse'],

'splitter': ['best','random']

}

}

}

scores = []

cv = ShuffleSplit(n\_splits=5, test\_size=0.2, random\_state=0)

**for** algo\_name, config **in** algos.items():

gs = GridSearchCV(config['model'], config['params'], cv=cv, return\_train\_score=**False**)

gs.fit(X,y)

scores.append({

'model': algo\_name,

'best\_score': gs.best\_score\_,

'best\_params': gs.best\_params\_

})

**return** pd.DataFrame(scores,columns=['model','best\_score','best\_params'])

find\_best\_model\_using\_gridsearchcv(X,y)

|  | **model** | **best\_score** | **best\_params** |
| --- | --- | --- | --- |
| **0** | linear\_regression | 0.847796 | {'normalize': False} |
| **1** | lasso | 0.726738 | {'alpha': 2, 'selection': 'cyclic'} |
| **2** | decision\_tree | 0.714629 | {'criterion': 'mse', 'splitter': 'random'} |

##### ***Based on above results we can say that LinearRegression gives the best score. Hence we will use that.***

print(df12['price'].describe())

plt.figure(figsize=(9, 8))

sns.distplot(df12['price'], color='g', bins=100, hist\_kws={'alpha': 0.4});

count 7239.000000

mean 96.546676

std 87.732734

min 10.000000

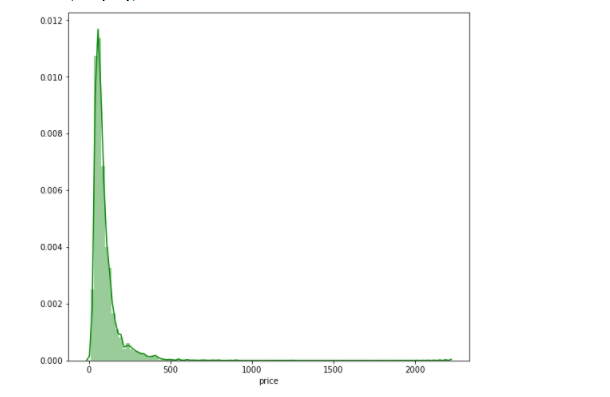
25% 50.000000

50% 72.000000

75% 110.000000

max 2200.000000

Name: price, dtype: float64



def predict\_price(location,sqft,bath,bhk):

loc\_index = np.where(X.columns==location)[0][0]

x = np.zeros(len(X.columns))

x[0] = sqft

x[1] = bath

x[2] = bhk

if loc\_index >= 0:

x[loc\_index] = 1

return lr\_clf.predict([x])[0]

predict\_price('1st Phase JP Nagar',1000, 2, 2)

83.86570258312275

predict\_price('1st Phase JP Nagar',1000, 3, 3)

86.08062284987054

predict\_price('Indira Nagar',1000, 2, 2)

193.31197733179937

predict\_price('Indira Nagar',1000, 3, 3)

195.52689759854715

**Summary:**

We have created an EC2 instance through the aws cli. We have written the source code for the data visualization with respect to pricing of houses in Bengaluru. We have cloned the repository, downloaded the required dataset ( Bangalore House Dataset ). We have then executed the EDA(.ipython) file and visualized the data.