Digital Empowerment Pakistan

"Project"

Intern Name	Muhammad Attaullah
DEP Number	DEP935
Batch	July 2024

Topic:

“Predicting House Price”

Introduction:

Predicting house prices using machine learning involves building a model that can estimate the value of a property based on various features. In this project we predict the price of house base on secondary data. There are many steps that are used to make data clean and then we train model base on the clean data.

1. EDA and Preprocessing:

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process where analysts use statistical graphics and other data visualization methods to examine datasets and summarize their main characteristics, often with the help of visual tools.

Preprocessing is a critical step in the data analysis and machine learning pipeline. It involves preparing and transforming raw data into a clean and usable format. Effective preprocessing improves the performance of machine learning models by ensuring that the data is consistent, relevant, and free of errors or biases.

House_price_prediction.ipynb

House_price_prediction.ipynb > data.shape

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[11] ✓ 0.0s Python

```python
data=pd.read_csv('kc_house_data.csv')
```

[12] ✓ 0.0s Python

```python
data.head()
```

[13] ✓ 0.0s Python

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade
0	7129300520	20141013T000000	221900.00000	3	1.00000	1180	5650	1.00000	0	0	...	7
1	6414100192	20141209T000000	538000.00000	3	2.25000	2570	7242	2.00000	0	0	...	7
2	5631500400	20150225T000000	180000.00000	2	1.00000	770	10000	1.00000	0	0	...	6
3	2487200875	20141209T000000	604000.00000	4	3.00000	1960	5000	1.00000	0	0	...	7
4	1954400510	20150218T000000	510000.00000	3	2.00000	1680	8080	1.00000	0	0	...	8

5 rows × 21 columns

House_price_prediction.ipynb

House_price_prediction.ipynb > data.shape

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```python
data.shape
```

[14] ✓ 0.0s Python

```
(21613, 21)
```

```python
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

[15] ✓ 0.0s Python

```python
data.describe()
```

[16] ✓ 0.0s Python

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
count	21613.00000	21613.00000	21613.00000	21613.00000	21613.00000	21613.00000	21613.00000	21613.00000	21613.000
mean	4580301520.86499	540088.14177	3.37084	2.11476	2079.89974	15106.96757	1.49431	0.00754	0.234
std	2876565571.31206	367127.19648	0.93006	0.77016	918.44090	41420.51152	0.53999	0.08652	0.766
min	1000102.00000	75000.00000	0.00000	0.00000	290.00000	520.00000	1.00000	0.00000	0.000
25%	2123049194.00000	321950.00000	3.00000	1.75000	1427.00000	5040.00000	1.00000	0.00000	0.000
50%	3904930410.00000	450000.00000	3.00000	2.25000	1910.00000	7618.00000	1.50000	0.00000	0.000
75%	7308900445.00000	645000.00000	4.00000	2.50000	2550.00000	10688.00000	2.00000	0.00000	0.000
max	9900000190.00000	7700000.00000	33.00000	8.00000	13540.00000	1651359.00000	3.50000	1.00000	4.000

House_price_prediction.ipynb

House_price_prediction.ipynb > data.shape

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```python
data.info()
```

[17] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
```

House_price_prediction.ipynb

House_price_prediction.ipynb > data.shape

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```python
data.isnull().sum()
```

[18] ✓ 0.0s Python

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

Explain Code Comment Code Find Bugs Code Chat Search Error Ln 4, Col 20 Spaces: 4 Spaces: 4 CRLF Cell 4 of 50 Go Live AI Code Chat

House_price_prediction.ipynb

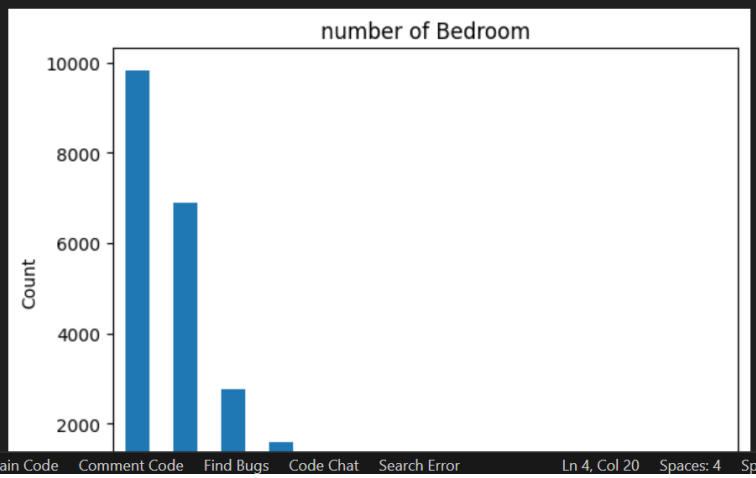
House_price_prediction.ipynb > data.shape

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```python
data['bedrooms'].value_counts().plot(kind='bar')
plt.title('number of Bedroom')
plt.xlabel('Bedrooms')
plt.ylabel('Count')
sns.despine
```

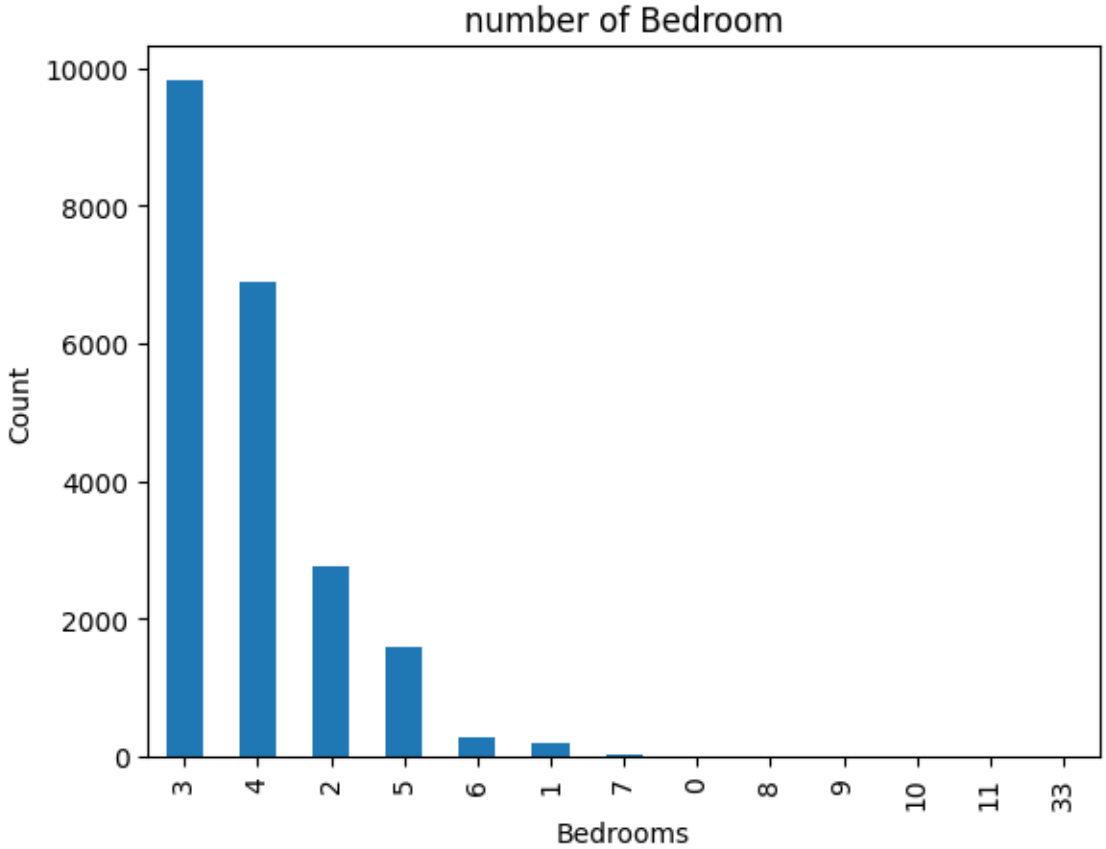
[19] ✓ 0.1s Python

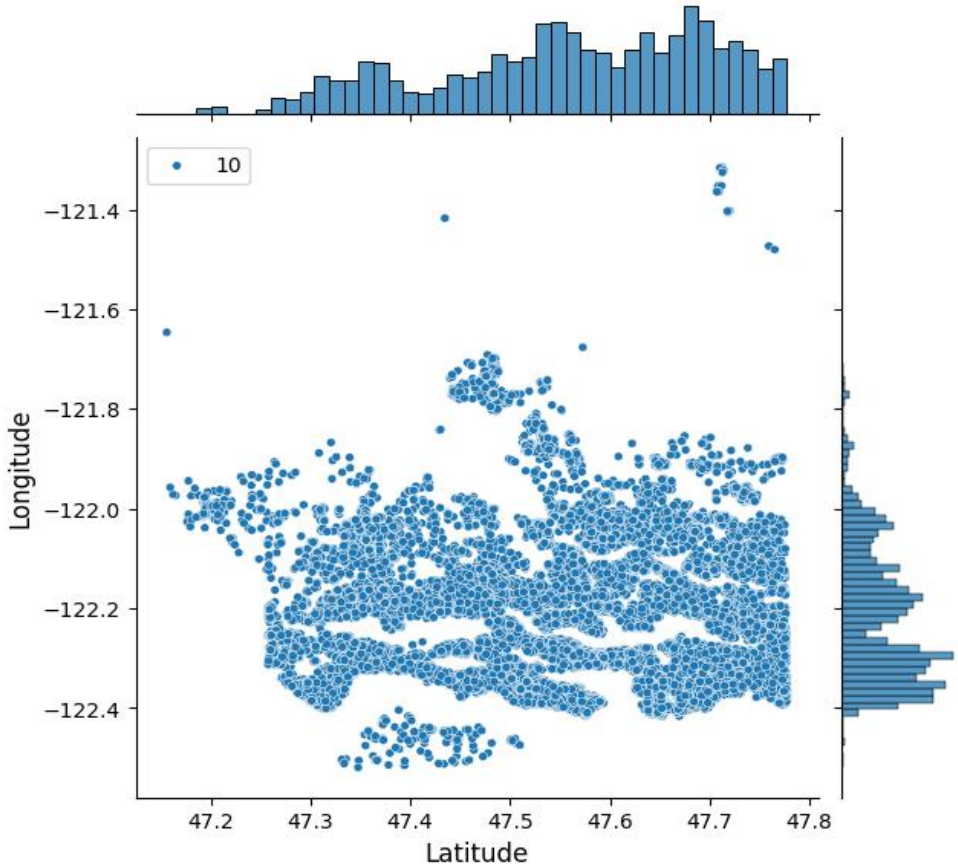
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left=False, bottom=False, offset=None, trim=False)>
```

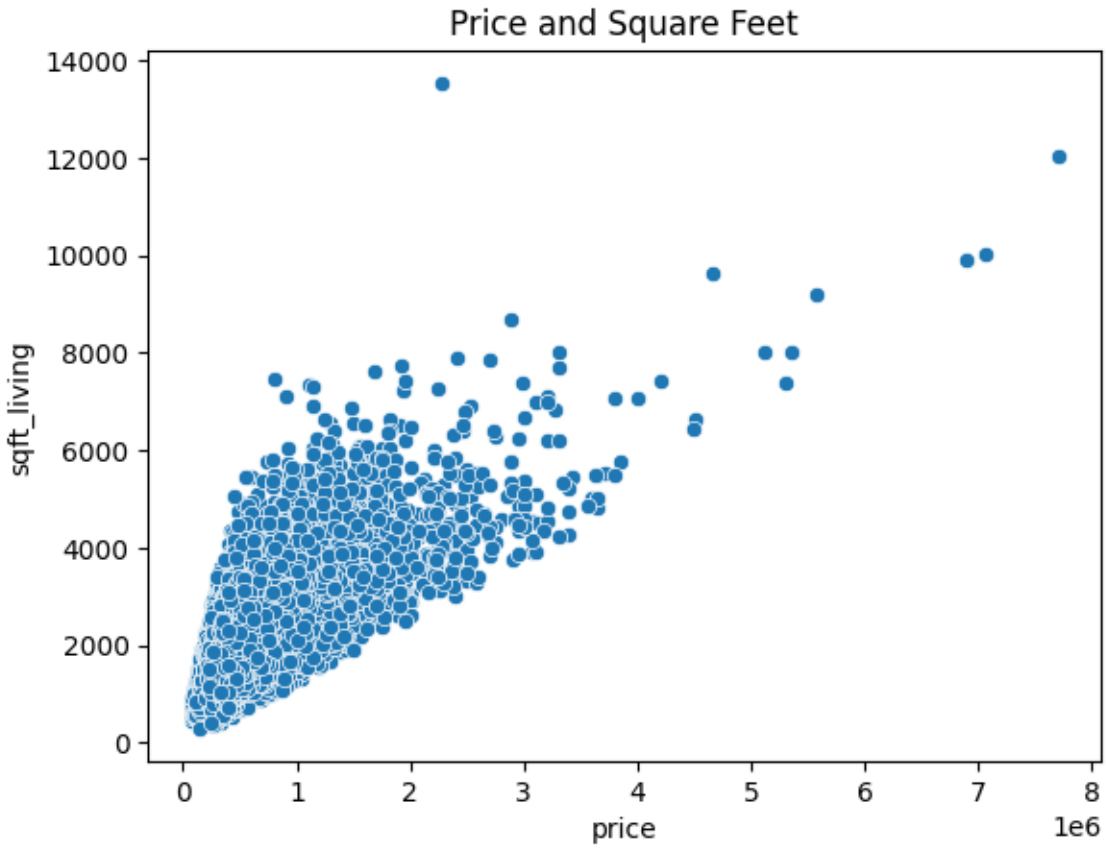


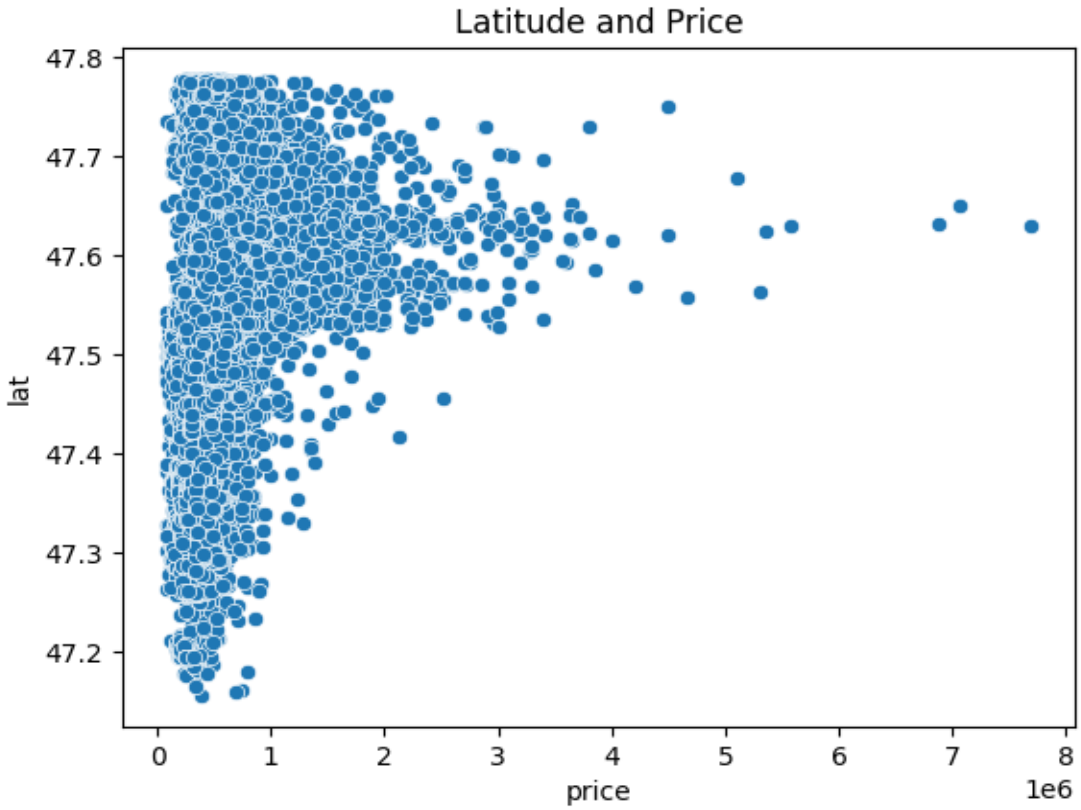
Explain Code Comment Code Find Bugs Code Chat Search Error Ln 4, Col 20 Spaces: 4 Spaces: 4 CRLF Cell 4 of 50 Go Live AI Code Chat

Graphs:


number of Bedroom
Count
10000
8000
6000
4000
2000
0
3
4
2
5
6
1
7
0
8
9
10
11
33
Bedrooms


10
−121.4
−121.6
−121.8
−122.0
−122.2
−122.4
Longitude
47.2
47.3
47.4
47.5
47.6
47.7
47.8
Latitude


Price and Square Feet
sqft_living
price
1e6
0
2000
4000
6000
8000
10000
12000
14000
0
1
2
3
4
5
6
7
8


Latitude and Price
lat
price
1e6
47.2
47.3
47.4
47.5
47.6
47.7
47.8
0
1
2
3
4
5
6
7
8

Splitting of Data:

In this we divide our data into two variables called x and y. The x variables basically that data, we give our model to learn from it and y data is that data, gives us the prediction.we also create some new variables that is easily understandable for machine is x_train,y_train,x_test,y_test. After this we apply standardition matrics on our data. Let's see in the code:

House_price_prediction.ipynb

House_price_prediction.ipynb > Exploratory Data Analysis > sns.scatterplot(x=data.price,y=data.lat)

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

+ Code + Markdown

Splitting of Data for training and testing

```
data.columns
```

[119] Python

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

```
X=data[['bedrooms', 'bathrooms', 'sqft_living',
        'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
        'sqft_above', 'sqft_basement',
         'sqft_living15', 'sqft_lot15']]
```

[120] Python

```
X
```

[121] Python

House_price_prediction.ipynb

House_price_prediction.ipynb > Exploratory Data Analysis > sns.scatterplot(x=data.price,y=data.lat)

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```
X
```

[121] Python

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_livin
0	3	1.00000	1180	5650	1.00000	0	0	3	7	1180	0	1
1	3	2.25000	2570	7242	2.00000	0	0	3	7	2170	400	1
2	2	1.00000	770	10000	1.00000	0	0	3	6	770	0	2
3	4	3.00000	1960	5000	1.00000	0	0	5	7	1050	910	1
4	3	2.00000	1680	8080	1.00000	0	0	3	8	1680	0	1
...	...	...	...	...	...	...	...	...	...	...	...	
21608	3	2.50000	1530	1131	3.00000	0	0	3	8	1530	0	1
21609	4	2.50000	2310	5813	2.00000	0	0	3	8	2310	0	1
21610	2	0.75000	1020	1350	2.00000	0	0	3	7	1020	0	1
21611	3	2.50000	1600	2388	2.00000	0	0	3	8	1600	0	1
21612	2	0.75000	1020	1076	2.00000	0	0	3	7	1020	0	1

21613 rows × 13 columns

House_price_prediction.ipynb

House_price_prediction.ipynb > Splitting of Data for training and testing > from sklearn.model_selection import train_test_split

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.11.9

[122]
```python
Y=data['price']
```
Python

[123]
```python
Y
```
Python

```
0       221900.00000
1       538000.00000
2       180000.00000
3       604000.00000
4       510000.00000
            ...
21608   360000.00000
21609   400000.00000
21610   402101.00000
21611   400000.00000
21612   325000.00000
Name: price, Length: 21613, dtype: float64
```

[125]
```python
from sklearn.model_selection import train_test_split
```
Python

House_price_prediction.ipynb

House_price_prediction.ipynb > Splitting of Data for training and testing > from sklearn.model_selection import train_test_split

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... Python 3.11.9

[126]
```python
X_train, X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.25,random_state=101)
```
Python

[127]
```python
X_train
```
Python

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_livin
19642	4	2.50000	2070	4270	2.00000	0	0	3	8	2070	0	2
8626	3	2.25000	1400	6970	2.00000	0	0	3	8	1400	0	1
6954	5	3.00000	2190	4900	2.00000	0	0	5	7	1490	700	1
20242	4	2.50000	2547	4800	2.00000	0	0	3	9	2547	0	2
10376	5	3.50000	4060	8309	2.00000	0	0	3	9	2960	1100	1
...	...	...	...	...	...	...	...	...	...	...	...	
5695	3	2.25000	1920	9672	2.00000	0	0	4	8	1920	0	1
8006	3	1.00000	1240	3600	1.50000	0	0	3	7	1240	0	1
17745	3	2.25000	1780	7332	2.00000	0	0	3	7	1780	0	1
17931	2	1.00000	1150	5000	1.00000	0	0	4	7	1050	100	1
13151	3	1.00000	1450	7930	1.00000	0	0	4	6	1150	300	1

16209 rows × 13 columns

House_price_prediction.ipynb

House_price_prediction.ipynb > Splitting of Data for training and testing > from sklearn.model_selection import train_test_split

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

[128]
```python
Y_train
```
Python

```
19642   493000.00000
8626    425000.00000
6954    588000.00000
20242   392440.00000
10376   630000.00000
            ...
5695    625000.00000
8006    390000.00000
17745   272450.00000
17931   552000.00000
13151   394000.00000
Name: price, Length: 16209, dtype: float64
```

[129]
```python
X_test
```
Python

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_livin
3834	2	1.00000	1050	6317	1.50000	0	0	4	7	1050	0	1
1348	4	2.25000	2040	9565	1.00000	0	0	3	8	1400	640	1
20366	4	2.50000	2500	4000	2.00000	0	0	3	8	2500	0	1
16617	5	2.00000	2360	19899	1.00000	0	0	4	7	2360	0	1

Explain Code Comment Code Find Bugs Code Chat Search Error Spaces: 4 Cell 21 of 50 Go Live AI Code Chat

House_price_prediction.ipynb

House_price_prediction.ipynb > Splitting of Data for training and testing > from sklearn.model_selection import train_test_split

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

2747	5	3.00000	2970	10335	2.00000	0	0	3	9	2970	0	2
15565	3	2.50000	3440	103672	2.00000	0	0	3	9	3440	0	2
21238	3	2.50000	1572	4000	2.00000	0	0	3	8	1572	0	1
2333	3	3.00000	2920	23085	1.50000	0	2	3	7	1540	1380	2

5404 rows × 13 columns

[130]
```python
Y_test
```
Python

```
3834    349950.00000
1348    450000.00000
20366   635000.00000
16617   355500.00000
20925   246950.00000
            ...
7148    738000.00000
2747    726000.00000
15565   560000.00000
21238   299000.00000
2333    555000.00000
Name: price, Length: 5404, dtype: float64
```

Explain Code Comment Code Find Bugs Code Chat Search Error Spaces: 4 Cell 21 of 50 Go Live AI Code Chat

Normalization:

In this we normalize the data by using standar normal distribution approach. Normalization is the process in which we make that clean and tidy to easily or correctly understand.

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

Normalization basically Standarization

[131]
```python
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
```
Python

[132]
```python
X_train_std=std.fit_transform(X_train)
X_test_std=std.transform(X_test)
```
Python

[133]
```python
X_train
```
Python

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_living
19642	4	2.50000	2070	4270	2.00000	0	0	3	8	2070	0	20
8626	3	2.25000	1400	6970	2.00000	0	0	3	8	1400	0	18
6954	5	3.00000	2190	4900	2.00000	0	0	5	7	1490	700	13
20242	4	2.50000	2547	4800	2.00000	0	0	3	9	2547	0	25
10376	5	3.50000	4060	8309	2.00000	0	0	3	9	2960	1100	17

Explain Code Comment Code Find Bugs Code Chat Search Error Cell 27 of 49 Go Live AI Code Chat

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

[134]
```python
X_train_std
```
Python

```
array([[ 0.69608634,  0.49894508, -0.00810266, ..., -0.65896096,
         0.12372791, -0.29371924],
       [-0.40802008,  0.173745  , -0.74457445, ..., -0.65896096,
        -0.27289927, -0.16524962],
       [ 1.80019275,  1.14934525,  0.12380274, ...,  0.93301185,
        -0.90456477, -0.29386481],
       ...,
       [-0.40802008,  0.173745  , -0.32687403, ..., -0.65896096,
        -0.69890623, -0.18399236],
       [-1.51212649, -1.45225542, -1.01937736, ..., -0.43153627,
        -0.76794874, -0.3042734 ],
       [-0.40802008, -1.45225542, -0.68961387, ...,  0.0233131 ,
        -1.38933131, -0.17980708]])
```

[135]
```python
X_test_std
```
Python

```
array([[-1.51212649, -1.45225542, -1.12929852, ..., -0.65896096,
        -0.56669717, -0.11153258],
       [ 0.69608634,  0.173745  , -0.04107901, ...,  0.79655703,
        -0.14069021, -0.14923641],
       [ 0.69608634,  0.49894508,  0.46455834, ..., -0.65896096,
```

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.11.9

[136]
```python
Y_train
```

```
19642   493000.00000
8626    425000.00000
6954    588000.00000
20242   392440.00000
10376   630000.00000
            ...
5695    625000.00000
8006    390000.00000
17745   272450.00000
17931   552000.00000
13151   394000.00000
Name: price, Length: 16209, dtype: float64
```

[137]
```python
Y_test
```

```
3834    349950.00000
1348    450000.00000
20366   635000.00000
16617   355500.00000
20925   246950.00000
            ...
```

Model:

Model basically use to predict our outcomes based on our label data.In this we are going to create our model base on our data. Let’s start:

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.11.9

[138]
```python
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

[139]
```python
lr.fit(X_train_std,Y_train)
```

```
LinearRegression
LinearRegression()
```

[140]
```python
Y_pred=lr.predict(X_test_std)
```

[141]
```python
Y_pred
```

```
array([356776.02589275, 511927.33428643, 573736.52389151, ...,
       824101.44697874, 437013.51415353, 731915.82170729])
```

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.11.9

[142]
```python
Y_test
```
Python

```
3834    349950.00000
1348    450000.00000
20366   635000.00000
16617   355500.00000
20925   246950.00000
            ...
7148    738000.00000
2747    726000.00000
15565   560000.00000
21238   299000.00000
2333    555000.00000
Name: price, Length: 5404, dtype: float64
```

[143]
```python
from sklearn.metrics import mean_absolute_error,r2_score
```
Python

[144]
```python
mean_absolute_error(Y_test,Y_pred)
```
Python

```
151500.5360109019
```

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

Code | Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ... | Python 3.11.9

[145]
```python
X_test
```
Python

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_livin
3834	2	1.00000	1050	6317	1.50000	0	0	4	7	1050	0	1
1348	4	2.25000	2040	9565	1.00000	0	0	3	8	1400	640	1
20366	4	2.50000	2500	4000	2.00000	0	0	3	8	2500	0	1
16617	5	2.00000	2360	19899	1.00000	0	0	4	7	2360	0	1
20925	3	3.00000	1670	4440	1.00000	0	0	3	7	1670	0	1
...	...	...	...	...	...	...	...	...	...	...	...	
7148	3	1.75000	1520	5500	1.50000	0	0	5	7	1520	0	2
2747	5	3.00000	2970	10335	2.00000	0	0	3	9	2970	0	2
15565	3	2.50000	3440	103672	2.00000	0	0	3	9	3440	0	2
21238	3	2.50000	1572	4000	2.00000	0	0	3	8	1572	0	1
2333	3	3.00000	2920	23085	1.50000	0	2	3	7	1540	1380	2

5404 rows × 13 columns

```python
r2_score(Y_test,Y_pred)
```

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

[146]
```python
r2_score(Y_test,Y_pred)
```
Python

```
0.6211653492243892
```

Prediction

[147]
```python
new_house=[[3,1,1120,5000,1.5,0,0,5,7,1620,0,2010,5000]]
```
Python

[148]
```python
new_house_std=std.transform(new_house)
```
Python

```
C:\Users\PMYLS\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\s
  warnings.warn(
```

[149]
```python
new_house_std
```
Python

House_price_prediction.ipynb

House_price_prediction.ipynb > Normalization basically Standarization

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ··· Python 3.11.9

```
  warnings.warn(
```

[149]
```python
new_house_std
```
Python

```
array([[-0.40802008, -1.45225542, -1.05235371, -0.23898601,  0.00725634,
        -0.08921717, -0.30859645,  2.45576999, -0.55954731, -0.20383709,
        -0.65896096,  0.03558853, -0.27952571]])
```

[150]
```python
int(lr.predict(new_house_std))
```
Python

```
C:\Users\PMYLS\AppData\Local\Temp\ipykernel_25028\2175366686.py:1: DeprecationWarning: Conversion of an array with ndim > 0 to
  int(lr.predict(new_house_std))
```

```
444708
```

Conclusion:

In this project we create a model based on data that we get from Kaggle link is: https://www.kaggle.com/datasets/shivachandel/kc-house-data I get the secondary that of house prices based on different variables like bedrooms, bathrooms etc, I perform some analysis on it normalize data after I train the model base on the train data that I have in the code.ss