House Prediction Task-1

Use a dataset that includes information about housing prices and features like square footage, number of bedrooms, etc. to train a model that can predict the price of a new house

House price prediction is a machine learning task that involves using historical data to build a model capable of estimating the prices of houses based on various features or attributes. This process is essential for real estate, financial planning, and investment decisions. Here's an explanation of the key steps involved in house price prediction:

panda,numpy,matplotlib,seaborn,sklearn are the basic libraries used in the email spam filtering natural language tool kit used to study the data which means a mail and visualized the data in the different graphical form(pictorial representation and here we are using the linear regression to predict the price of a new house

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
file_path="C:\\Users\\prade\\OneDrive\\Documents\\DATASCIENCE\\Intern DataS
Hp_df=pd.read_csv(file_path)
Hp_df
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080
...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076

21613 rows × 21 columns

In [5]:
```
Columns= Hp_df.shape[1]
Rows=Hp_df.shape[0]
print('Number of columns :',Columns)
print('Number of Rows    :',Rows)
```

```
Number of columns : 21
Number of Rows    : 21613
```

In [6]:
```
Hp_df.head()
```

Out[6]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1

5 rows × 21 columns

In [7]:
```
Hp_df.tail()
```

Out[7]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076

5 rows × 21 columns

In [8]:
```
Hp_df.info()
```

```
<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
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [9]:
```
Hp_df.dtypes
```

Out[9]:
```
id                 int64
date              object
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

In [10]:
```
Hp_df.describe(include='all').T
```

Out[10]:

	count	unique	top	freq	mean	
id	21613.0	NaN	NaN	NaN	4580301520.864988	2876565571.312
date	21613	372	20140623T000000	142	NaN	N
price	21613.0	NaN	NaN	NaN	540088.141767	367127.196
bedrooms	21613.0	NaN	NaN	NaN	3.370842	0.930
bathrooms	21613.0	NaN	NaN	NaN	2.114757	0.770
sqft_living	21613.0	NaN	NaN	NaN	2079.899736	918.440
sqft_lot	21613.0	NaN	NaN	NaN	15106.967566	41420.511
floors	21613.0	NaN	NaN	NaN	1.494309	0.539
waterfront	21613.0	NaN	NaN	NaN	0.007542	0.086
view	21613.0	NaN	NaN	NaN	0.234303	0.766
condition	21613.0	NaN	NaN	NaN	3.40943	0.650
grade	21613.0	NaN	NaN	NaN	7.656873	1.175
sqft_above	21613.0	NaN	NaN	NaN	1788.390691	828.090
sqft_basement	21613.0	NaN	NaN	NaN	291.509045	442.575
yr_built	21613.0	NaN	NaN	NaN	1971.005136	29.373
yr_renovated	21613.0	NaN	NaN	NaN	84.402258	401.679
zipcode	21613.0	NaN	NaN	NaN	98077.939805	53.505
lat	21613.0	NaN	NaN	NaN	47.560053	0.138
long	21613.0	NaN	NaN	NaN	-122.213896	0.140
sqft_living15	21613.0	NaN	NaN	NaN	1986.552492	685.391
sqft_lot15	21613.0	NaN	NaN	NaN	12768.455652	27304.179

In [13]:

```
Hp_df.drop('id',axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[13], line 1
----> 1 Hp_df.drop('id',axis=1,inplace=True)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5258, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   5110 def drop(
   5111     self,
   5112     labels: IndexLabel = None,
   (...)
   5119     errors: IgnoreRaise = "raise",
   5120 ) -> DataFrame | None:
   5121     """
   5122     Drop specified labels from rows or columns.
   5123 
   (...)
   5256             weight  1.0     0.8
   5257     """
-> 5258     return super().drop(
   5259         labels=labels,
   5260         axis=axis,
   5261         index=index,
   5262         columns=columns,
   5263         level=level,
   5264         inplace=inplace,
   5265         errors=errors,
   5266     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4549, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)
   4547 for axis, labels in axes.items():
   4548     if labels is not None:
-> 4549         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4551 if inplace:
   4552     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4591, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
   4589         new_axis = axis.drop(labels, level=level, errors=errors)
   4590     else:
-> 4591         new_axis = axis.drop(labels, errors=errors)
   4592     indexer = axis.get_indexer(new_axis)
   4594 # Case for non-unique axis
   4595 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6699, in Index.drop(self, labels, errors)
   6697 if mask.any():
   6698     if errors != "ignore":
-> 6699         raise KeyError(f"{list(labels[mask])} not found in axis")
   6700     indexer = indexer[~mask]
   6701 return self.delete(indexer)

KeyError: "['id'] not found in axis"
```

In [ ]:
```
Hp_df.head()
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le
```

In [ ]:
```
Hp_df['date']=le.fit_transform(Hp_df['date'])
Hp_df['date'].dtype
```

Exploratory Data Analysis[EDA]

count the number of houses with unique floor values.

In [ ]:
```
Hp_df['floors'].value_counts()
```

In [ ]:
```
Hp_df['floors'].value_counts().unique()
```

In [ ]:
```
Hp_df['floors'].value_counts().to_frame()
```

In [ ]:
```
Hp_df.hist(bins=40,figsize=(15,15))
plt.show()
```

Determine whether houses with a waterfront view or without a waterfront view␣ have more price outliers.

In [ ]:
```
sns.boxplot(data=Hp_df,x=Hp_df['waterfront'],y=Hp_df['price'])
plt.show()
```

Determine if the feature sqft_above is negatively or positively correlated␣ with price.

In [ ]:
```
sns.regplot(data=Hp_df,x=Hp_df['sqft_above'],y=Hp_df['price'])
plt.show()
```

In [ ]:
```
sns.boxplot(data=Hp_df,x=Hp_df['sqft_basement'],y=Hp_df['price'])
plt.show()
```

In [ ]:
```
sns.barplot(data=Hp_df,x=Hp_df['floors'],y=Hp_df['price'])
plt.show()
```

In [ ]:
```
sns.histplot(data=Hp_df,x=Hp_df['grade'],y=Hp_df['price'])
plt.show()
```

In [ ]:
```
sns.barplot(data=Hp_df,x=Hp_df['grade'],y=Hp_df['price'])
plt.show()
```

In [ ]:
```
corr_matrix= Hp_df.corr()
fig, ax = plt.subplots(figsize=(15, 10))
ax = sns.heatmap(corr_matrix,annot=True,
                 linewidths=0.5,fmt=".2f",cmap="viridis");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top- 0.5)
```

In [ ]:
```
correlation_values = Hp_df.drop('price', axis=1).corrwith(Hp_df['price'])
correlation_values.plot(kind='bar', grid=True, figsize=(10, 6), title="Corr
plt.show()
```

In [ ]:
```
Hp_df.skew()
```

splitting the data set

In [ ]:
```
from sklearn.model_selection import train_test_split

X=np.array(Hp_df.drop(columns="price"))
y=np.array(Hp_df.drop(columns='price'))
space=Hp_df["sqft_living"]
price=Hp_df["price"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_st
print(f"the shape of x_train is : {X_train.shape}")
print(f'the shape of x_test is : {X_test.shape}')
print(f'the shape of y_tain is : {y_train.shape}')
print(f'the shape of y_test is {y_test.shape}')
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_absolute_error
model3=LinearRegression()
model3.fit(X_train,y_train)
y_pred3=model3.predict(X_test)
print(f'R2 Score is : {r2_score(y_test,y_pred3)}')
print(f'Mae is : {mean_absolute_error(y_test,y_pred3)}')
```

In [ ]:
```
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, y_train, color='blue')
plt.title("visualization--")
plt.xlabel('space')
plt.ylabel('price')
plt.show()
```

In [ ]:
```
plt.scatter(X_test, y_test, label='Actual data',color='blue')
plt.plot(X_test, y_test, color='red')
plt.title("visualization")
plt.xlabel('space')
plt.ylabel('price')
plt.show()
```

In [ ]:

In [ ]:

In [ ]: