Start coding or generate with AI.

PREDICTING THE PRICE OF A HOUSE

IMPORTING LIBRARIES AND LOADING DATA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Machine Learning libraries:
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
df=pd.read_csv('/content/data.csv')
df
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_baseme
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	2
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	10
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	8
...	...	...	...	...	...	...	...	...	...	...	...	
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	10
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	

4600 rows × 18 columns

DATA INFORMATIONS

```
df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	

```
df.tail()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basem
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	4	1510	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	3	1460	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	3	3010	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	3	1070	1
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	4	1490	

```
df.columns
```

```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

```
df.isna().sum()
```

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

```
df.dtypes
```

```
date              object
price            float64
bedrooms         float64
bathrooms        float64
```

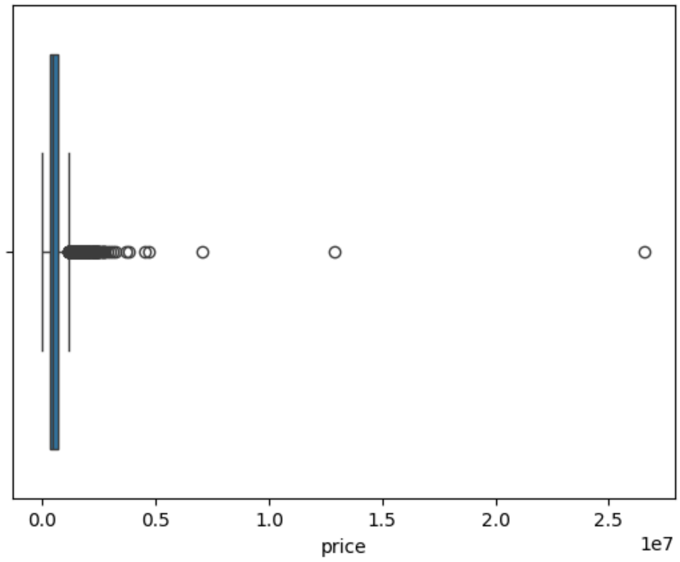
```
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
lab=LabelEncoder()
df['date']=lab.fit_transform(df['date'])
df['street']=lab.fit_transform(df['street'])
df['city']=lab.fit_transform(df['city'])
df['statezip']=lab.fit_transform(df['statezip'])
df['country']=lab.fit_transform(df['country'])
df.dtypes
```

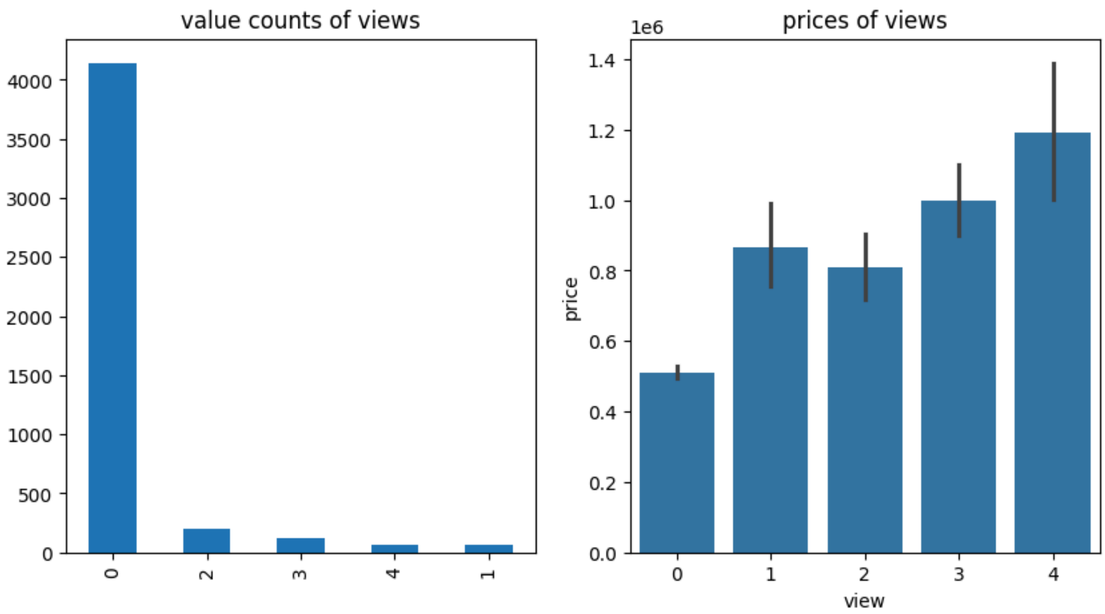
```
date               int64
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street             int64
city               int64
statezip           int64
country            int64
dtype: object
```

```
# looking for outliers
sns.boxplot(data=df,x="price")
```

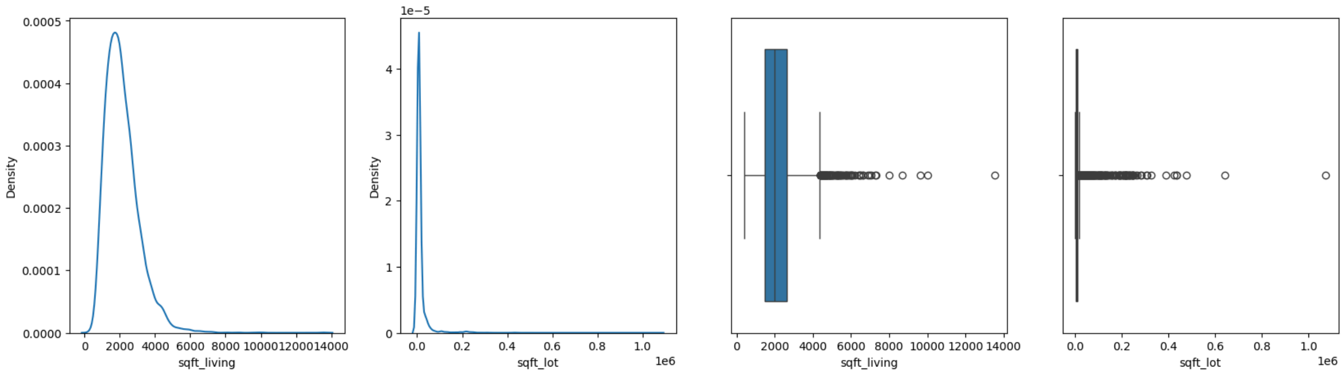
```
<Axes: xlabel='price'>
```



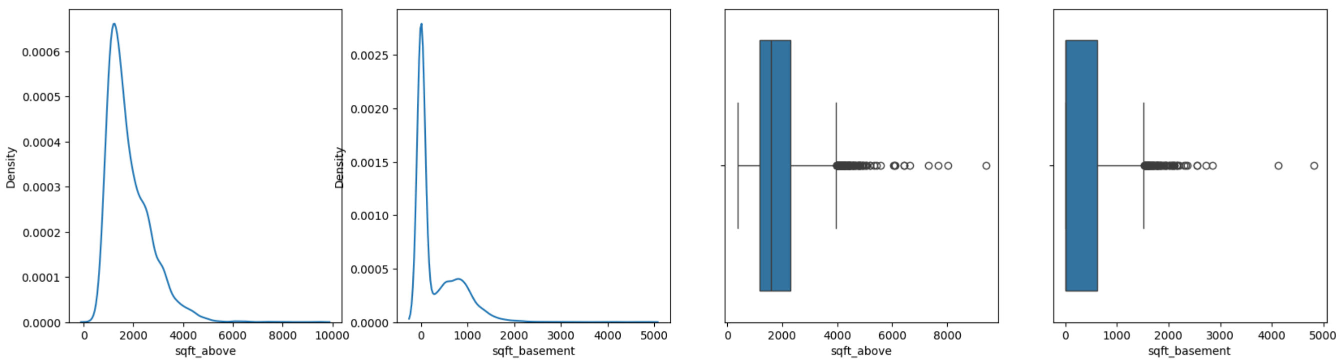
```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
plt.title("value counts of views")
df["view"].value_counts().plot(kind="bar")

plt.subplot(1,2,2)
plt.title("prices of views")
sns.barplot(df,x="view",y="price")

plt.show()
```




```
plt.figure(figsize=(20,5))

plt.subplot(1,4,1)
sns.kdeplot(df["sqft_living"])

plt.subplot(1,4,2)
sns.kdeplot(df["sqft_lot"])

plt.subplot(1,4,3)
sns.boxplot(data=df,x="sqft_living")

plt.subplot(1,4,4)
sns.boxplot(data=df,x="sqft_lot")

plt.show()
```



```
plt.figure(figsize=(20,5))

plt.subplot(1,4,1)
sns.kdeplot(df["sqft_above"])

plt.subplot(1,4,2)
sns.kdeplot(df["sqft_basement"])

plt.subplot(1,4,3)
sns.boxplot(data=df,x="sqft_above")

plt.subplot(1,4,4)
sns.boxplot(data=df,x="sqft_basement")

plt.show()
```



DROP UNWANTED COLUMNS

```
df.drop(columns=["date","street","country","waterfront"],axis=1,inplace=True)
```

SEPARATING INPUT AND OUTPUT AS X AND Y

```
x=df.drop(['price'],axis=1)
x
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	city	s
0	3.0	1.50	1340	7912	1.5	0	3	1340	0	1955	2005	36	
1	5.0	2.50	3650	9050	2.0	4	5	3370	280	1921	0	35	
2	3.0	2.00	1930	11947	1.0	0	4	1930	0	1966	0	18	
3	3.0	2.25	2000	8030	1.0	0	4	1000	1000	1963	0	3	
4	4.0	2.50	1940	10500	1.0	0	4	1140	800	1976	1992	31	
...	...	...	...	...	...	...	...	...	...	...	...	...	
4595	3.0	1.75	1510	6360	1.0	0	4	1510	0	1954	1979	35	
4596	3.0	2.50	1460	7573	2.0	0	3	1460	0	1983	2009	3	
4597	3.0	2.50	3010	7014	2.0	0	3	3010	0	2009	0	32	
4598	4.0	2.00	2090	6630	1.0	0	3	1070	1020	1974	0	35	
4599	3.0	2.50	1490	8102	2.0	0	4	1490	0	1990	0	9	

4600 rows × 13 columns

```
y=df['price']
y
```

```
0       3.130000e+05
1       2.384000e+06
2       3.420000e+05
3       4.200000e+05
4       5.500000e+05
            ...
4595    3.081667e+05
4596    5.343333e+05
4597    4.169042e+05
4598    2.034000e+05
```

```
4599    2.206000e+05
Name: price, Length: 4600, dtype: float64
```

Train Test Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

Model Evaluation

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array([ 293449.20358154,  288253.47828724, 1119160.92873146, ...,
        320508.80664454,  337785.61439408,  589797.47266801])
```

```
df1=pd.DataFrame({'Actual_value':y_test,'predicted_value':y_pred,'difference':y_test-y_pred})
df1
```

	Actual_value	predicted_value	difference
3683	544000.0	2.934492e+05	250550.796418
4411	0.0	2.882535e+05	-288253.478287
2584	1712500.0	1.119161e+06	593339.071269
69	365000.0	5.616038e+05	-196603.824763
1844	275000.0	3.770401e+05	-102040.096645
...	...	...	...
3437	620000.0	7.898406e+05	-169840.641753
3340	770000.0	9.828890e+05	-212888.952401
1289	255000.0	3.205088e+05	-65508.806645
449	336900.0	3.377856e+05	-885.614394
3774	620000.0	5.897975e+05	30202.527332

1380 rows × 3 columns

```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error,mean_squared_error,r2_score
print("MAE is",mean_absolute_error(y_test,y_pred))
print("MAPE is",mean_absolute_percentage_error(y_test,y_pred))
msq=mean_squared_error(y_test,y_pred)
sq=np.sqrt(msq)
print("MSE is",msq)
print("RMSE is",sq)
print("R2_SCORE is",r2_score(y_test,y_pred))
```

```
MAE is 193842.7586600735
MAPE is 3.1999946139646595e+19
MSE is 678535679486.3732
RMSE is 823732.7718904797
R2_SCORE is 0.06078785193592806
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
d_model=DecisionTreeRegressor(max_depth=7,min_samples_split=4,min_samples_leaf=1)
r_model=RandomForestRegressor(n_estimators=120,max_depth=16,max_features=10)
model=[d_model,r_model]
```