assignment4

March 11, 2024

1 Consider the Bangalore House Price Data. Perform following operations.

a) Find and replace null values in the data using appropriate technique.

b) Transform the 'Size' column to numerical values. For Example: 2 BHK to be converted as 2

c) Transform the 'total_sqft' column to contain numerical values on same scale. If the range is given average value of the range to be taken.

d) Calculate and add one more column as 'Price_Per_Sqft'

e) Remove the outliers from Price_Per_Sqft and BHK Size column if any.

f) Apply the Linear Regression model to the data and display the training and testing performance measures as Mean Squared Error and Accuracy

[2]:

```
import pandas as pd
import numpy as np
```

[3]:

```
df = pd.read_csv('/content/Banglore Housing Prices.csv')
```

[4]:

```
df.head()
```

[4]:

```
                    location       size total_sqft  bath   price
0  Electronic City Phase II      2 BHK       1056   2.0   39.07
1          Chikka Tirupathi  4 Bedroom       2600   5.0  120.00
2               Uttarahalli      3 BHK       1440   2.0   62.00
3        Lingadheeranahalli      3 BHK       1521   3.0   95.00
4                  Kothanur      2 BHK       1200   2.0   51.00
```

[5]:

```
df.dtypes
```

[5]:

```
location       object
size           object
total_sqft     object
bath          float64
price         float64
dtype: object
```

2 a) Find and replace null values in the data using appropriate technique.

```
[6]: df.isnull().sum()
```

```
[6]: location       1
     size          16
     total_sqft     0
     bath          73
     price          0
     dtype: int64
```

```
[7]: df.dropna(axis =0, subset= ['location'],inplace = True)
```

```
[8]: df['bath'].fillna(df['bath'].median(), inplace = True)
```

```
[9]: most_freq = df['size'].value_counts().idxmax()
     print(most_freq)
```

```
2 BHK
```

```
[10]: df['size'].fillna(value='2 BHK', inplace=True)
```

```
[11]: df.isnull().sum()
```

```
[11]: location      0
      size          0
      total_sqft    0
      bath          0
      price         0
      dtype: int64
```

3 b) Transform the 'Size' column to numerical values. For Example: 2 BHK to be converted as 2

```
[12]: df.dtypes
```

```
[12]: location        object
      size            object
      total_sqft      object
      bath           float64
      price          float64
      dtype: object
```

```
[13]: df['BHK'] = df['size'].apply(lambda x: int(x.split(' ')[0]))
```

[14]:

```
df = df.drop('size', axis=1)
```

[35]:

```
df.head()
```

[35]:

```
                    location  total_sqft  bath   price  BHK  Price_Per_Sqft
0  Electronic City Phase II      1056.0   2.0   39.07    2        0.036998
1          Chikka Tirupathi      2600.0   5.0  120.00    4        0.046154
2               Uttarahalli      1440.0   2.0   62.00    3        0.043056
3        Lingadheeranahalli      1521.0   3.0   95.00    3        0.062459
4                  Kothanur      1200.0   2.0   51.00    2        0.042500
```

4 c) Transform the 'total_sqft' column to contain numerical values on same scale. If the range is given average value of the range to be taken.

[15]:

```
df.dtypes
```

[15]:

```
location       object
total_sqft     object
bath          float64
price         float64
BHK             int64
dtype: object
```

[16]:

```
import re

def convert_sqft_to_num(sqft):
        if '-' in sqft:
            tokens = sqft.split('-')
            return (float(tokens[0]) + float(tokens[1])) / 2
        else:
            numeric_part = re.search(r'\d+\.\d+|\d+', sqft).group()  # Extract␣
 ↪numeric part using regular expression
            return float(numeric_part)

df['total_sqft'] = df['total_sqft'].apply(convert_sqft_to_num)
```

[33]:

```
df['total_sqft'].dtypes
```

[33]:

```
dtype('float64')
```

5 d) Calculate and add one more column as 'Price_Per_Sqft'

[18]:

```
df['Price_Per_Sqft'] = df['price'] / df['total_sqft']
```

[19]:

```
df.dtypes
```

[19]:

```
location           object
total_sqft        float64
bath              float64
price             float64
BHK                 int64
Price_Per_Sqft    float64
dtype: object
```

[34]:

```
df.head()
```

[34]:

```
                  location  total_sqft  bath   price  BHK  Price_Per_Sqft
0  Electronic City Phase II      1056.0   2.0   39.07    2        0.036998
1          Chikka Tirupathi      2600.0   5.0  120.00    4        0.046154
2               Uttarahalli      1440.0   2.0   62.00    3        0.043056
3        Lingadheeranahalli      1521.0   3.0   95.00    3        0.062459
4                  Kothanur      1200.0   2.0   51.00    2        0.042500
```

6 e) Remove the outliers from Price_Per_Sqft and BHK Size column if any.

[20]:

```
z_score_price_sqft = np.abs((df['Price_Per_Sqft'] - df['Price_Per_Sqft'].
 ↪mean()) / df['Price_Per_Sqft'].std())

z_score_bhk = np.abs((df['BHK'] - df['BHK'].mean()) / df['BHK'].std())
```

[21]:

```
df_no_outliers = df[(z_score_price_sqft < 3) & (z_score_bhk < 3)]
```

[22]:

```
df_no_outliers.head()
```

[22]:

```
                  location  total_sqft  bath   price  BHK  Price_Per_Sqft
0  Electronic City Phase II      1056.0   2.0   39.07    2        0.036998
1          Chikka Tirupathi      2600.0   5.0  120.00    4        0.046154
2               Uttarahalli      1440.0   2.0   62.00    3        0.043056
3        Lingadheeranahalli      1521.0   3.0   95.00    3        0.062459
4                  Kothanur      1200.0   2.0   51.00    2        0.042500
```

7 f) Apply the Linear Regression model to the data and display the training and testing performance measures as Mean Squared Error and Accuracy

[23]:

```
x = df_no_outliers[['BHK', 'total_sqft', 'bath']]
y = df_no_outliers['price']
```

[27]:

```
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=42)
```

[28]:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(x_train, y_train)
```

[28]:

```
LinearRegression()
```

[31]:

```
from sklearn.metrics import mean_squared_error, accuracy_score

# Training performance
train_preds = model.predict(x_train)
train_mse = mean_squared_error(y_train, train_preds)

train_r_squared = model.score(x_train, y_train)

print(f'Training Mean Squared Error: {train_mse}')
print(f'Training R-squared: {train_r_squared}')
```

```
Training Mean Squared Error: 10598.854485807771
Training R-squared: 0.43895420489846004
```

[32]:

```
# Testing performance
test_preds = model.predict(x_test)
test_mse = mean_squared_error(y_test, test_preds)

test_r_squared = model.score(x_test, y_test)

print(f'Testing Mean Squared Error: {test_mse}')
print(f'Testing R-squared: {test_r_squared}')
```

```
Testing Mean Squared Error: 14806.822930693757
Testing R-squared: 0.4017790711088002
```

[ ]: