VIT Applied Data Science 2023

Assignment 3

Ishaan Singh Bains
Registration Number: 20BCE2091
ishaansingh.bains2020@vitstudent.ac.in

ADS Assignment 3
Problem Statement: House Price Prediction
Description:- House price prediction is a common problem in the real estate industry and involves predicting the selling price of a house based on various features and attributes. The problem is typically approached as a regression problem, where the target variable is the price of the house, and the features are various attributes of the house
The features used in house price prediction can include both quantitative and categorical variables, such as the number of bedrooms, house area, bedrooms, furnished, nearness to main road, and various amenities such as a garage and other factors that may influence the value of the property.
Accurate predictions can help agents and appraisers price homes correctly, while homeowners can use the predictions to set a reasonable asking price for their properties. Accurate house price prediction can also be useful for buyers who are looking to make informed decisions about purchasing a property and obtaining a fair price for their investment.
Attribute Information:
Name - Description
1- Price-Prices of the houses
2- Area- Area of the houses
3- Bedrooms- No of house bedrooms
4- Bathrooms- No of bathrooms
5- Stories- No of house stories
6- Main Road- Weather connected to Main road
7- Guestroom-Weather has a guest room
8- Basement-Weather has a basement
9- Hot water heating- Weather has a hot water heater
10-Airconditioning-Weather has a air conditioner
11-Parking- No of house parking
12-Furnishing Status-Furnishing status of house

Building a Regression Model

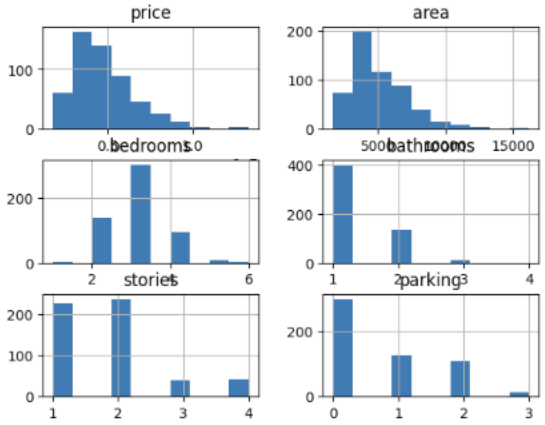
1. Download the dataset: Dataset
2. Load the dataset into the tool.

```
In [82]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score
         from sklearn.preprocessing import StandardScaler
         from sklearn.compose import ColumnTransformer
         from sklearn.preprocessing import OneHotEncoder

         # Step 2: Load the dataset
         df = pd.read_csv(r'C:\Study\SmartBridge\Assignments\CSV\housing.csv')
```

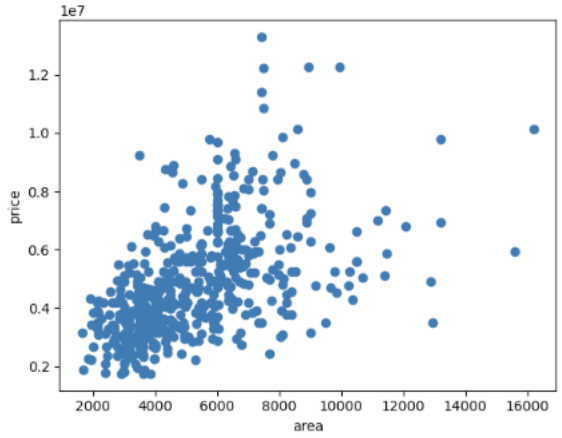
3. Perform Below Visualizations.

```
In [83]: # Step 3: Perform Visualization
         import matplotlib.pyplot as plt
         import seaborn as sns

         # Univariate Analysis
         df.hist()
         plt.show()

         # Bi-Variate Analysis
         plt.scatter(df['area'], df['price'])
         plt.xlabel('area')
         plt.ylabel('price')
         plt.show()

         # Multi-Variate Analysis
         corr_matrix = df.corr(numeric_only=True)
         plt.figure(figsize=(10, 8))
         sns.heatmap(corr_matrix, annot=True)
         plt.show()
```

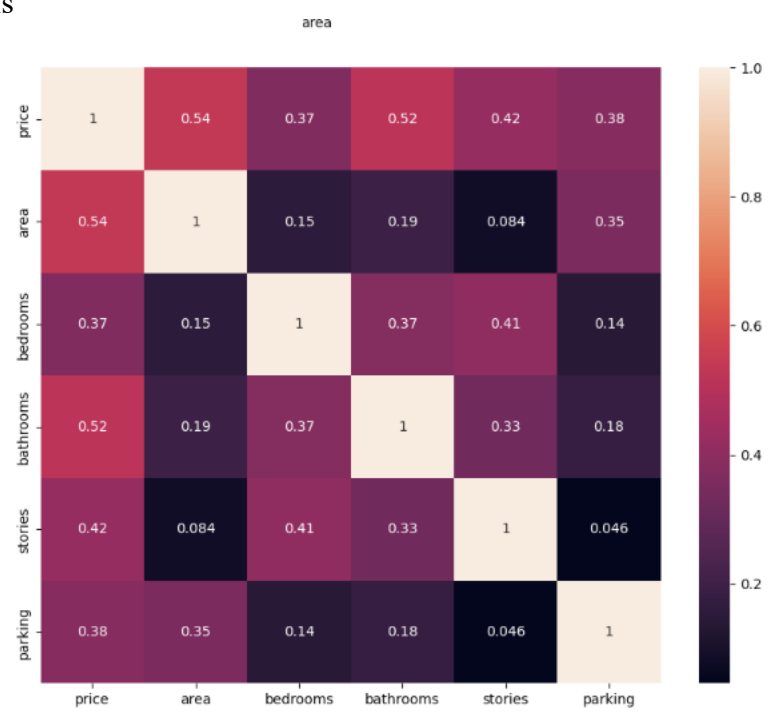
- Univariate Analysis



- Bi-Variate Analysis



- Multi-Variate Analysis



4. Perform descriptive statistics on the dataset.

```
In [84]: # Step 4: Perform descriptive statistics
         print("Descriptive Statistics:")
         print(df.describe())
```

```
Descriptive Statistics:
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000
mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505
std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492
min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000
25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000
50%    4.340000e+06   4600.000000    3.000000    1.000000    2.000000
75%    5.740000e+06   6360.000000    3.000000    2.000000    2.000000
max    1.330000e+07  16200.000000    6.000000    4.000000    4.000000

          parking
count  545.000000
mean     0.693578
std      0.861586
min      0.000000
25%      0.000000
50%      0.000000
75%      1.000000
max      3.000000
```

5. Check for Missing values and deal with them.

```
In [85]: # Step 5: Check for missing values
         print("Missing Values:")
         print(df.isnull().sum())
```

```
Missing Values:
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
furnishingstatus  0
dtype: int64
```

6. Find the outliers and replace them outliers

```
In [86]: # Step 6: Find the outliers and replace them
         from scipy.stats import zscore
         z_scores = df.select_dtypes(include=np.number).apply(zscore)
         threshold = 3
         df_no_outliers = df[(z_scores < threshold).all(axis=1)]
```

7. Check for Categorical columns and perform encoding.

```
In [87]: from sklearn.preprocessing import LabelEncoder, StandardScaler
         # Step 7: Check for Categorical columns and perform encoding
         # Identify categorical columns
         categorical_cols = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'furnishingstatus']

         for col in categorical_cols:
             df[col] = df[col].map({'yes': 1, 'no': 0})
             label_encoder = LabelEncoder()
         df['furnishingstatus'] = label_encoder.fit_transform(df['furnishingstatus'])
```

8. Split the data into dependent and independent variables.

```
In [88]: # Step 8: Split the data into dependent and independent variables
         X = df.drop(['price'], axis=1)
         y = df['price']
```

9. Scale the independent
variables
10. Split the data into training and testing
11. Build the Model
12. Train the Model
13. Test the Model
14. Measure the performance using Metrics.

In [90]:
```
# Step 10: Split the data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

In [91]:
```
# Step 11: Build the Model
model = LinearRegression()
```

In [92]:
```
# Step 12: Train the Model
model.fit(X_train, y_train)
```

Out[92]:
```
LinearRegression()
```

In [93]:
```
# Step 13: Test the Model
y_pred = model.predict(X_test)
```

In [94]:
```
# Step 14: Measure the performance using Metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 1886508447723.2888
R-squared: 0.6267717420522667
```