DOCUMENTATION

Predicting House Prices using Linear Regression Model

Understanding the data:

I have downloaded the data source from the link that is provided in the task document link for the dataset.

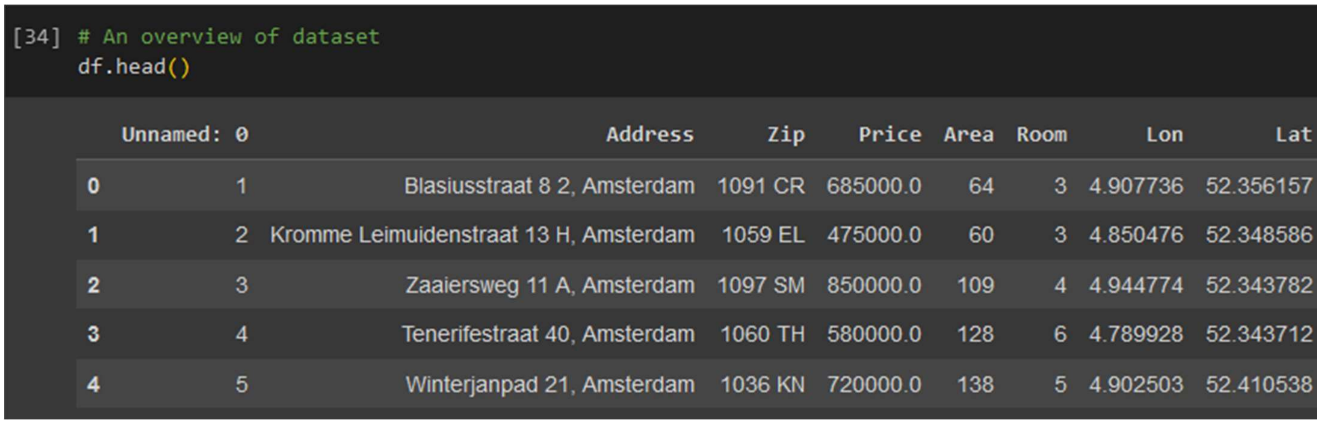
Firstly I have downloaded the .CSV file and took an overview of the no. of entries and attributes and their nature.

Checked for id or unique data columns which are not specifically important for the analysis.

Because it is a Price prediction problem I looked into the columns that are most important for the data analysis.

Data Exploration:

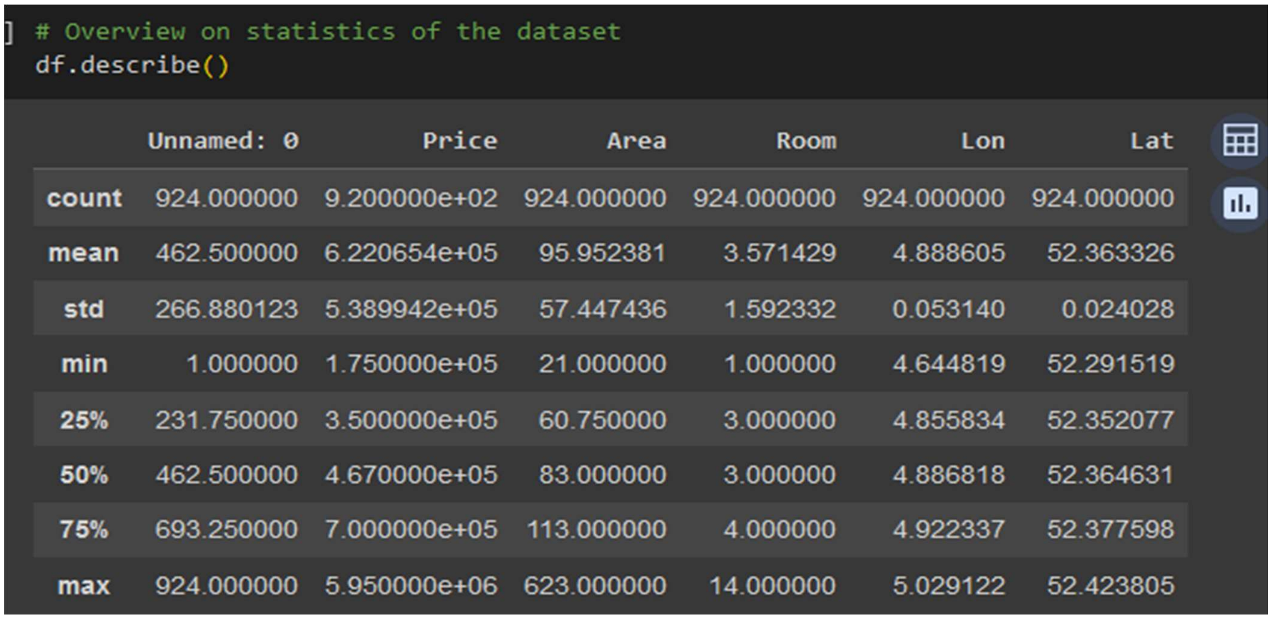
1. Importing libraries – import python libraries like pandas, matplotlib, seaborn, train_test_split and Linear regression model from sklearn and Model Evaluation packages required for the task.
2. Data from the .CSV file is loaded and converted into a Dataframe.
3. Path of the listing data is taken as a variable listing_data and data from this path is loaded into a Dataframe.
4. Use head( ) function to get an overview of the data that we are working with.



```
[34] # An overview of dataset
     df.head()
```

	Unnamed: 0	Address	Zip	Price	Area	Room	Lon	Lat
0	1	Blasiusstraat 8 2, Amsterdam	1091 CR	685000.0	64	3	4.907736	52.356157
1	2	Kromme Leimuidenstraat 13 H, Amsterdam	1059 EL	475000.0	60	3	4.850476	52.348586
2	3	Zaaiersweg 11 A, Amsterdam	1097 SM	850000.0	109	4	4.944774	52.343782
3	4	Tenerifestraat 40, Amsterdam	1060 TH	580000.0	128	6	4.789928	52.343712
4	5	Winterjanpad 21, Amsterdam	1036 KN	720000.0	138	5	4.902503	52.410538

5. Use shape( ) function to get the no. of entries and attributes in the dataset.
6. In this task the shape of the dataset is (924,8) i.e, it has 924 rows and 8 columns or attributes.
7. Used info( ) function to get the data-types of the attributes and also the number of non-null entries in each column.
8. Used describe( ) function to get basic statistical details like mean, standard deviation, percentile etc.

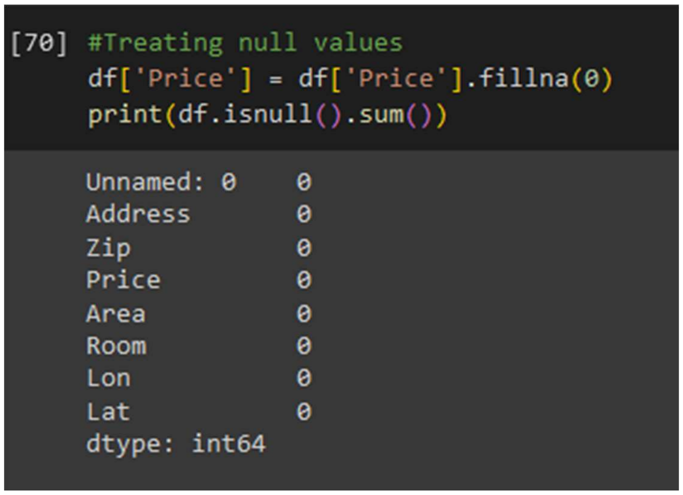


```
] # Overview on statistics of the dataset
  df.describe()
```

	Unnamed: 0	Price	Area	Room	Lon	Lat
count	924.000000	9.200000e+02	924.000000	924.000000	924.000000	924.000000
mean	462.500000	6.220654e+05	95.952381	3.571429	4.888605	52.363326
std	266.880123	5.389942e+05	57.447436	1.592332	0.053140	0.024028
min	1.000000	1.750000e+05	21.000000	1.000000	4.644819	52.291519
25%	231.750000	3.500000e+05	60.750000	3.000000	4.855834	52.352077
50%	462.500000	4.670000e+05	83.000000	3.000000	4.886818	52.364631
75%	693.250000	7.000000e+05	113.000000	4.000000	4.922337	52.377598
max	924.000000	5.950000e+06	623.000000	14.000000	5.029122	52.423805

Data Cleaning:

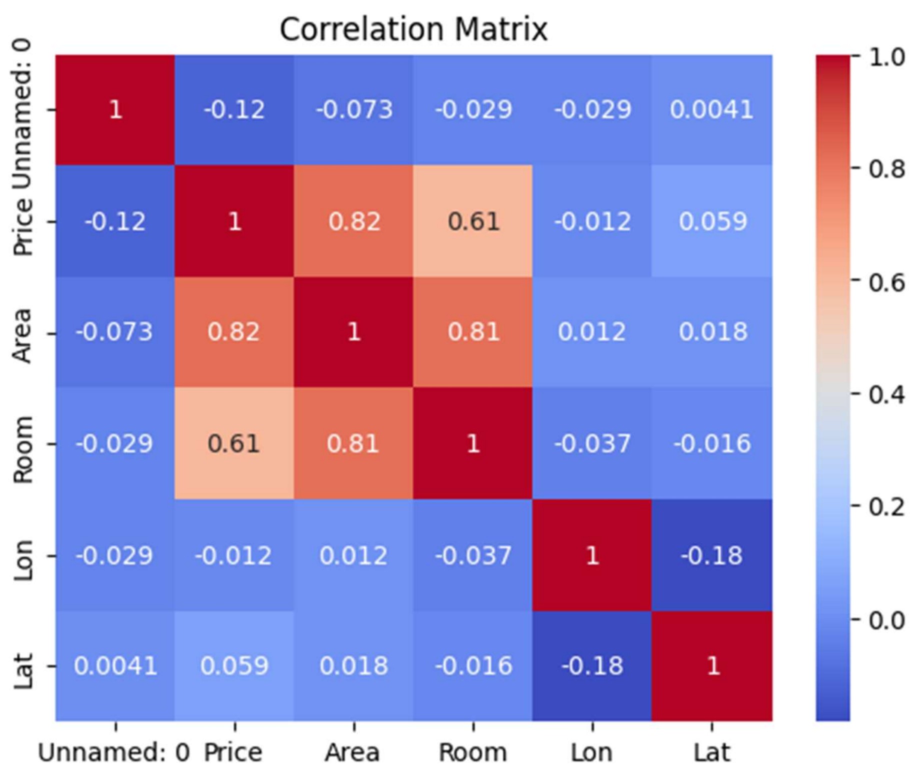
1. Firstly, I have checked for columns having minimum one missing value using isnull( )and sum( ) functions.
2. Dropped a Unnamed column which is not useful for the data analysis.
3. Column ‘Price’ have null values. So, replaced the null values with zeroes.



```
[70] #Treating null values
     df['Price'] = df['Price'].fillna(0)
     print(df.isnull().sum())

     Unnamed: 0    0
     Address       0
     Zip           0
     Price         0
     Area          0
     Room          0
     Lon           0
     Lat           0
     dtype: int64
```

Correlation Analysis:

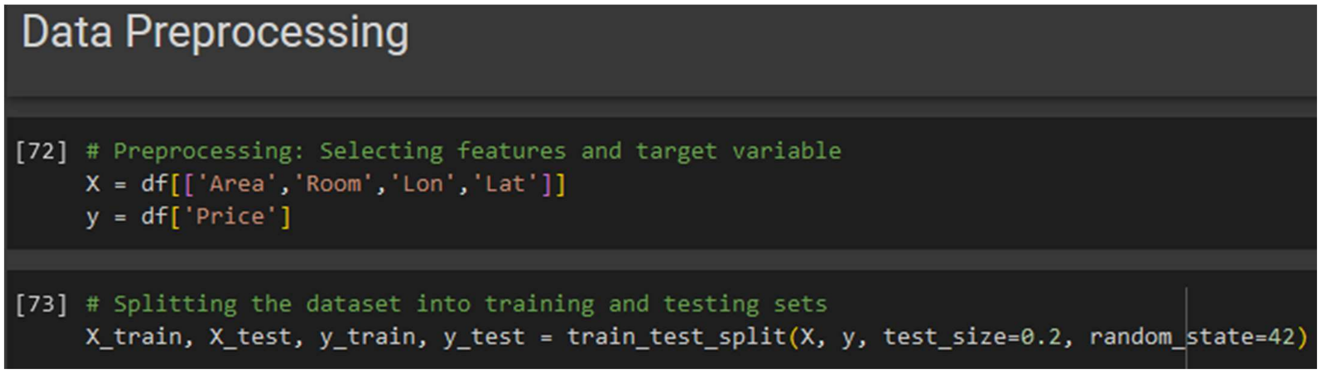
1. After doing the necessary cleaning in the data I’ve created a correlation matrix to check the relation between the attributes.
2. For a careful observation, a heat-map is plotted to clearly check for the correlation between the attributes.



3. From the above correlation analysis we can observe that Area is positively correlated to Price Room is positively correlated to Area, Area and no. of rooms is positively correlated to Price. Lon and Lat columns doesn't have any effect on price.

Data Pre-processing:

1. Separated both input variable X and target variable y.
2. After that split the dataset into training and testing data in 80% and 20% respectively.



Data Preprocessing

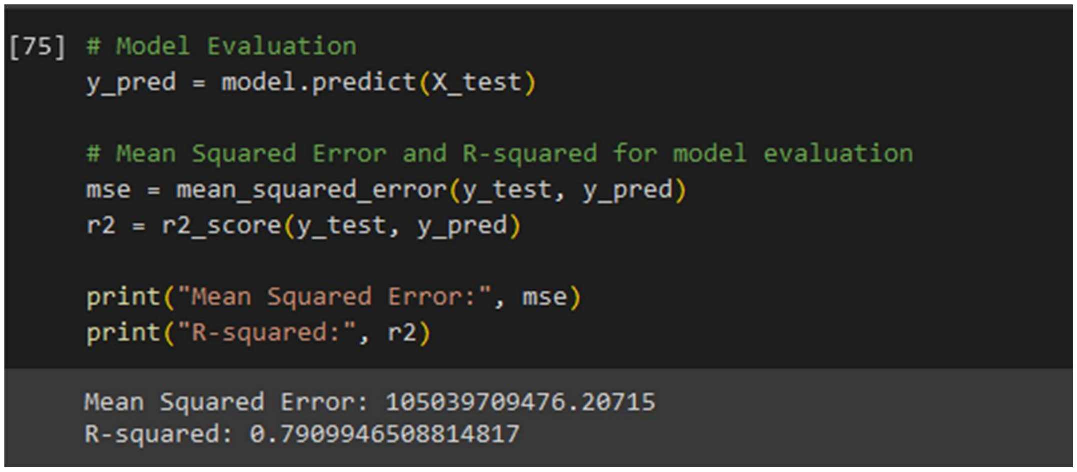
```
[72] # Preprocessing: Selecting features and target variable
     X = df[['Area','Room','Lon','Lat']]
     y = df['Price']
```

```
[73] # Splitting the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Building:

1. Created a LinearRegression Model.

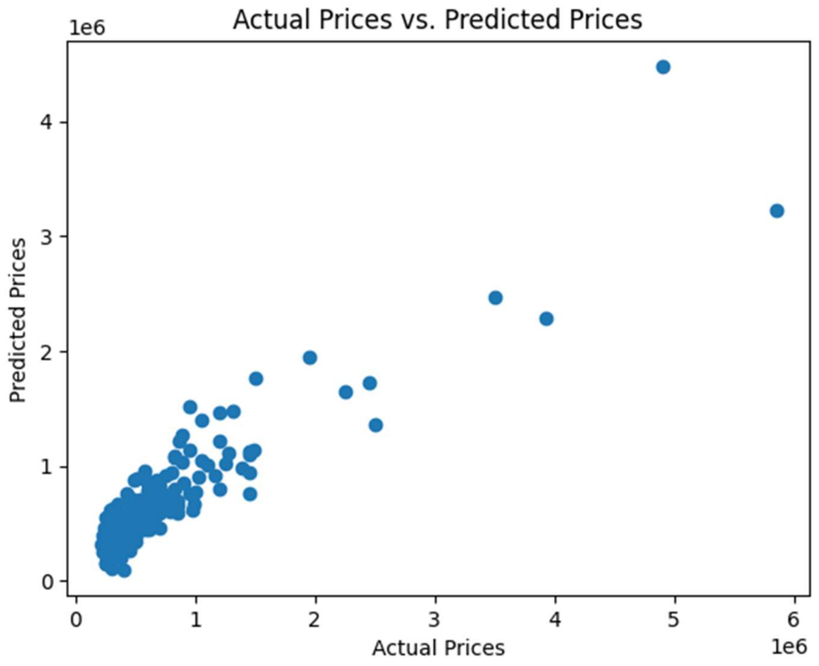
2. Calculated evaluation metrics like Mean squared erroe and R-squared error.



```
[75] # Model Evaluation
     y_pred = model.predict(X_test)

     # Mean Squared Error and R-squared for model evaluation
     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: 105039709476.20715
     R-squared: 0.7909946508814817
```

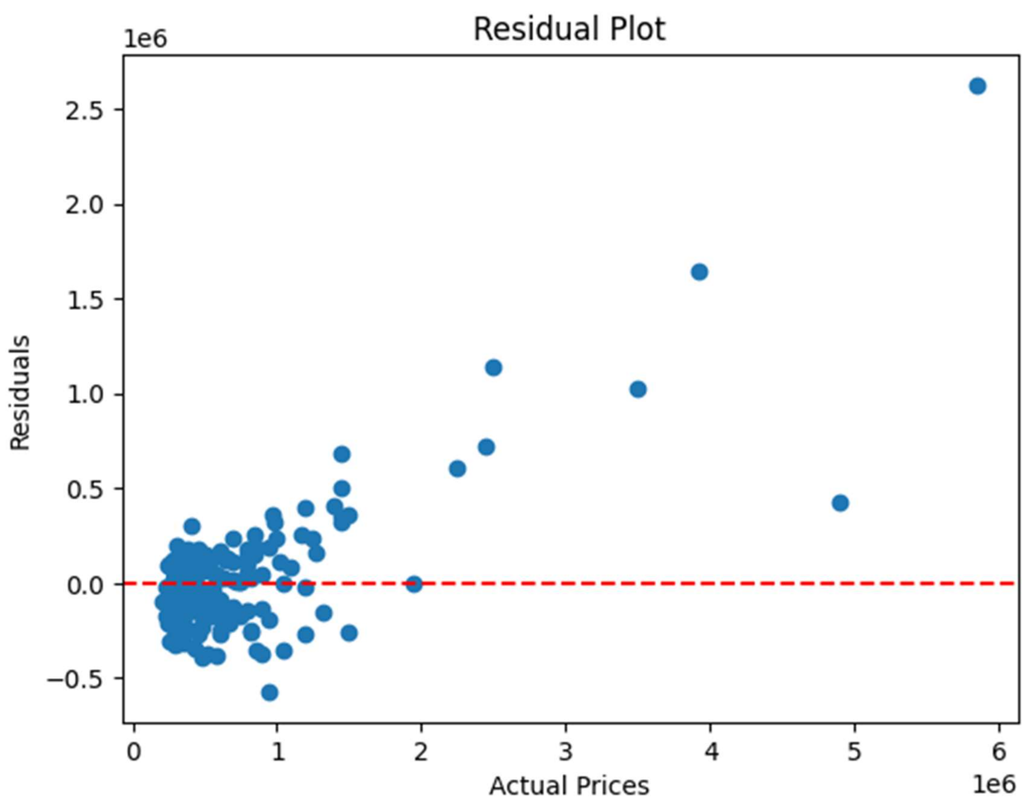
Visualization and Prediction:

1. A scatter plot is plotted against actual values and predicted values to check the accuracy.

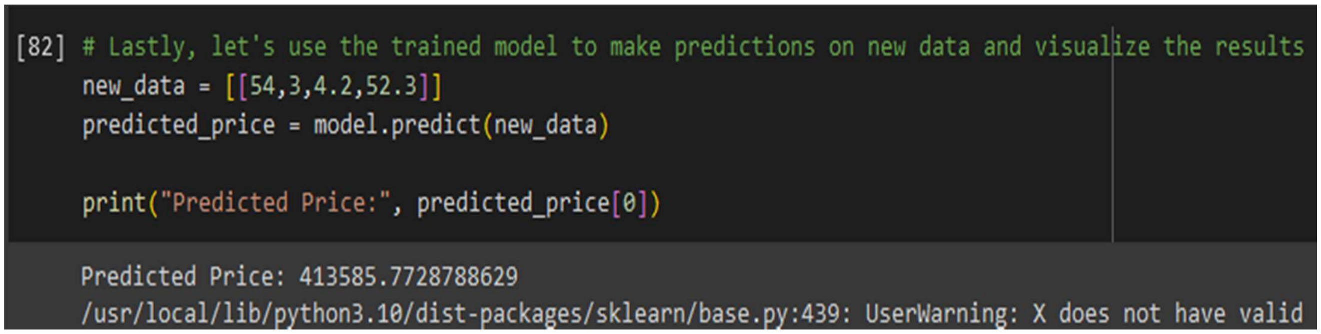


2. The above plot shows that the actual values and predicted values form a right-skewed scatter. And when we consider certain points it forms a straight line.

3. Also created a residual plot to check the model performance



4. After training the model if we give new instances the model will generate or predict the price of houses.



```
[82] # Lastly, let's use the trained model to make predictions on new data and visualize the results
     new_data = [[54,3,4.2,52.3]]
     predicted_price = model.predict(new_data)

     print("Predicted Price:", predicted_price[0])

     Predicted Price: 413585.7728788629
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid
```