```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
from google.colab import files
files.upload()
```

```
Choose Files 6 files
```

- **Bangalore.csv**(text/csv) - 643851 bytes, last modified: 4/18/2022 - 100% done
- **Chennai.csv**(text/csv) - 498429 bytes, last modified: 4/18/2022 - 100% done
- **Delhi.csv**(text/csv) - 500976 bytes, last modified: 4/18/2022 - 100% done
- **Hyderabad.csv**(text/csv) - 253067 bytes, last modified: 4/18/2022 - 100% done
- **Kolkata.csv**(text/csv) - 637386 bytes, last modified: 4/18/2022 - 100% done
- **Mumbai.csv**(text/csv) - 764465 bytes, last modified: 4/18/2022 - 100% done

```
Saving Bangalore.csv to Bangalore (1).csv
Saving Chennai.csv to Chennai (1).csv
Saving Delhi.csv to Delhi (1).csv
Saving Hyderabad.csv to Hyderabad (1).csv
Saving Kolkata.csv to Kolkata (1).csv
Saving Mumbai.csv to Mumbai (1).csv
{'Bangalore.csv': b"Price,Area,Location,No. of Bedrooms,Resale,MaintenanceS
 'Chennai.csv': b"Price,Area,Location,No. of Bedrooms,Resale,MaintenanceSta
 'Delhi.csv': b"Price,Area,Location,No. of Bedrooms,Resale,MaintenanceStaff
```

```
df_banglore=pd.read_csv('Bangalore.csv')
df_chennai=pd.read_csv('Chennai.csv')
df_delhi=pd.read_csv('Delhi.csv')
df_hyderabad=pd.read_csv('Hyderabad.csv')
df_kolkata=pd.read_csv('Kolkata.csv')
df_mumbai=pd.read_csv('Mumbai.csv')
```

```
df_banglore['City']='Banglore'
df_chennai['City']='Chennai'
df_delhi['City']='Delhi'
df_hyderabad['City']='Hyderabad'
df_kolkata['City']='Kolkata'
df_mumbai['City']='Mumbai'
```

```
df=pd.concat([df_banglore, df_chennai, df_delhi, df_hyderabad, df_kolkata, df_mumbai]).reset_index(drop=True)
df.head()
```

	Price	Area	Location	No. of Bedrooms	Resale	MaintenanceStaff	Gymnas
0	30000000	3340	JP Nagar Phase 1	4	0	1	
				2	0	0	
			Kannur on				

Saved successfully!

```
df.columns
```

```
Index(['Price', 'Area', 'Location', 'No. of Bedrooms', 'Resale',
       'MaintenanceStaff', 'Gymnasium', 'SwimmingPool', 'LandscapedGardens',
       'JoggingTrack', 'RainWaterHarvesting', 'IndoorGames', 'ShoppingMall',
       'Intercom', 'SportsFacility', 'ATM', 'ClubHouse', 'School',
       '24X7Security', 'PowerBackup', 'CarParking', 'StaffQuarter',
       'Cafeteria', 'MultipurposeRoom', 'Hospital', 'WashingMachine',
       'Gasconnection', 'AC', 'Wifi', 'Children'splayarea', 'LiftAvailable',
```

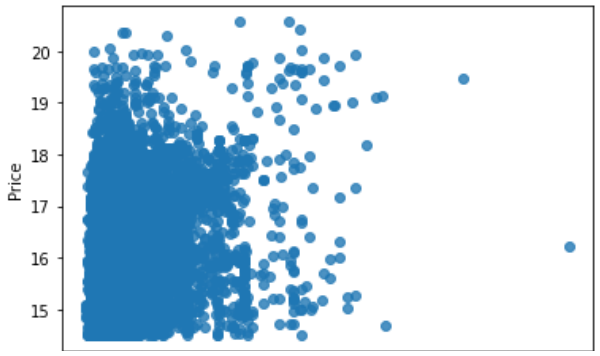
```
       'BED', 'VaastuCompliant', 'Microwave', 'GolfCourse', 'TV',
       'DiningTable', 'Sofa', 'Wardrobe', 'Refrigerator', 'City'],
      dtype='object')
```

```
df.rename(columns={"No. of Bedrooms":"Bedrooms", "24X7Security":"Security","Children'splayarea":"Playarea"}, inplac
```

```
y1=np.log(df["Price"])
```

```
sns.regplot(x="Area",y=y1,data=df,fit_reg=False)
```

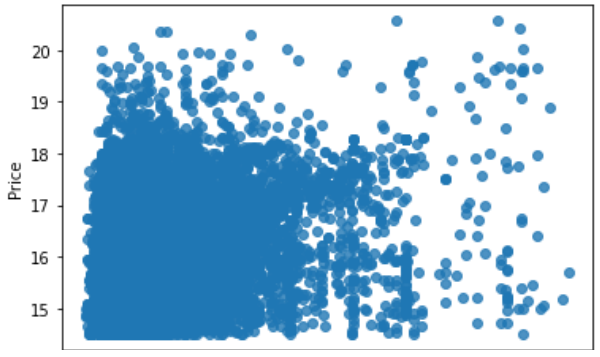
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0f52094650>
```




```
df.drop(df[df["Area"]>=8000].index,inplace=True)
```

```
y1=np.log(df["Price"])
sns.regplot(x="Area",y=y1,data=df,fit_reg=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0f5202a050>
```




```
df.columns
```

```
Index(['Price', 'Area', 'Location', 'Bedrooms', 'Resale', 'MaintenanceStaff',
       'Gymnasium', 'SwimmingPool', 'LandscapedGardens', 'JoggingTrack',
       'RainWaterHarvesting', 'IndoorGames', 'ShoppingMall', 'Intercom',
       'SportsFacility', 'ATM', 'ClubHouse', 'School', 'Security',
       'PowerBackup', 'CarParking', 'StaffQuarter', 'Cafeteria',
       'MultipurposeRoom', 'Hospital', 'WashingMachine', 'Gasconnection', 'AC',
       'Wifi', 'Playarea', 'LiftAvailable', 'BED', 'VaastuCompliant',
       'Microwave', 'GolfCourse', 'TV', 'DiningTable', 'Sofa', 'Wardrobe',
       'Refrigerator', 'City'],
      dtype='object')
```

Saved successfully!

```
col=['Location', 'Bedrooms', 'Resale', 'MaintenanceStaff',
       'Gymnasium', 'SwimmingPool', 'LandscapedGardens', 'JoggingTrack',
       'RainWaterHarvesting', 'IndoorGames', 'ShoppingMall', 'Intercom',
       'SportsFacility', 'ATM', 'ClubHouse', 'School', 'Security',
       'PowerBackup', 'CarParking', 'StaffQuarter', 'Cafeteria',
       'MultipurposeRoom', 'Hospital', 'WashingMachine', 'Gasconnection', 'AC',
```

```
       'Wifi', 'Playarea', 'LiftAvailable', 'BED', 'VaastuCompliant',
       'Microwave', 'GolfCourse', 'TV', 'DiningTable', 'Sofa', 'Wardrobe',
       'Refrigerator', 'City']
for column in col:
    df[column] = df[column].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32941 entries, 0 to 32962
Data columns (total 41 columns):
 #   Column               Non-Null Count  Dtype   
---  ------               --------------  -----   
 0   Price                32941 non-null  int64   
 1   Area                 32941 non-null  int64   
 2   Location             32941 non-null  category
 3   Bedrooms             32941 non-null  category
 4   Resale               32941 non-null  category
 5   MaintenanceStaff     32941 non-null  category
 6   Gymnasium            32941 non-null  category
 7   SwimmingPool         32941 non-null  category
 8   LandscapedGardens    32941 non-null  category
 9   JoggingTrack         32941 non-null  category
 10  RainWaterHarvesting  32941 non-null  category
 11  IndoorGames          32941 non-null  category
 12  ShoppingMall         32941 non-null  category
 13  Intercom             32941 non-null  category
 14  SportsFacility       32941 non-null  category
 15  ATM                  32941 non-null  category
 16  ClubHouse            32941 non-null  category
 17  School               32941 non-null  category
 18  Security             32941 non-null  category
 19  PowerBackup          32941 non-null  category
 20  CarParking           32941 non-null  category
 21  StaffQuarter         32941 non-null  category
 22  Cafeteria            32941 non-null  category
 23  MultipurposeRoom     32941 non-null  category
 24  Hospital             32941 non-null  category
 25  WashingMachine       32941 non-null  category
 26  Gasconnection        32941 non-null  category
 27  AC                   32941 non-null  category
 28  Wifi                 32941 non-null  category
 29  Playarea             32941 non-null  category
 30  LiftAvailable        32941 non-null  category
 31  BED                  32941 non-null  category
 32  VaastuCompliant      32941 non-null  category
 33  Microwave            32941 non-null  category
 34  GolfCourse           32941 non-null  category
 35  TV                   32941 non-null  category
 36  DiningTable          32941 non-null  category
 37  Sofa                 32941 non-null  category
 38  Wardrobe             32941 non-null  category
 39  Refrigerator         32941 non-null  category
 40  City                 32941 non-null  category
dtypes: category(39), int64(2)
memory usage: 2.1 MB
```

Saved successfully!

```
    print(df[column].value_counts())
    value_counts=df[column].value_counts()
    to_remove=value_counts[value_counts<35].index
    df.replace(to_remove,np.nan,inplace=True)
```

```
1    1182
Name: Cafeteria, dtype: int64
column
0    7648
1    2438
Name: MultipurposeRoom, dtype: int64
column
0    9512
1     574
Name: Hospital, dtype: int64
```

```
column
0    9655
1     431
Name: WashingMachine, dtype: int64
column
0    8315
1    1771
Name: Gasconnection, dtype: int64
column
0    9458
1     628
Name: AC, dtype: int64
column
0    9889
1     197
Name: Wifi, dtype: int64
column
1    5177
0    4909
Name: Playarea, dtype: int64
column
1    7231
0    2855
Name: LiftAvailable, dtype: int64
column
0    9167
1     919
Name: BED, dtype: int64
column
0    7080
1    3006
Name: VaastuCompliant, dtype: int64
column
0    9595
1     491
Name: Microwave, dtype: int64
column
0    9868
1     218
Name: GolfCourse, dtype: int64
column
0    9525
1     561
Name: TV, dtype: int64
column
0    9554
1     532
Name: DiningTable, dtype: int64
column
```

```
df.dropna(axis=0,inplace=True)
from sklearn.preprocessing import LabelEncoder


for column in col:
    le=LabelEncoder()
    df[column]=le.fit_transform(df[column])
```

Saved successfully!

```
<bound method DataFrame.info of           Price  Area  Location  Bedrooms  Resale  MaintenanceStaff  \
2       4866000  1179        78         1       0                 0
4       6845000  1670        82         2       0                 1
5       6797000  1220        56         1       0                 0
6      20000000  2502       185         3       0                 0
7       7105000  1438       149         2       0                 0
...         ...   ...       ...       ...     ...               ...
26630   8009999   965       111         1       0                 0
26631   8400000   965       111         1       0                 0
26632   7430000   965       111         1       0                 0
26633  21400000  1560        26         2       0                 0
26634  25700000  1505        47         2       0                 1

       Gymnasium  SwimmingPool  LandscapedGardens  JoggingTrack  ...  BED  \
2              1             1                  1             1  ...    0
```

```
4             1            1                 1            1  ...    0
5             1            1                 1            1  ...    0
6             1            1                 1            1  ...    0
7             1            0                 0            1  ...    0
...         ...          ...               ...          ...  ...  ...
26630         1            1                 1            1  ...    0
26631         1            1                 1            1  ...    0
26632         1            1                 1            1  ...    0
26633         1            0                 0            1  ...    0
26634         1            1                 1            1  ...    0

       VaastuCompliant  Microwave  GolfCourse  TV  DiningTable  Sofa  \
2                    0          0           0   0            0     0
4                    0          0           0   0            0     0
5                    0          0           0   0            0     0
6                    1          0           0   0            0     0
7                    1          0           0   0            0     0
...                ...        ...         ...  ..          ...   ...
26630                1          0           0   0            0     0
26631                1          0           0   0            0     0
26632                1          0           0   0            0     0
26633                0          0           0   0            0     0
26634                1          0           0   0            0     0

       Wardrobe  Refrigerator  City
2             0             0     0
4             0             0     0
5             0             0     0
6             0             0     0
7             0             0     0
...         ...           ...   ...
26630         0             0     5
26631         0             0     5
26632         0             0     5
26633         0             0     5
26634         0             0     5

[7081 rows x 41 columns]>
```

```
df.to_csv('Combined.csv', index=False)
```

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
!tar -xvf spark-3.2.1-bin-hadoop3.2.tgz
!pip install -q findspark
```

```
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_boolean_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_categorical_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_complex_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_date_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_datetime_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_null_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_num_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_string_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/test_udt_ops.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/data_type_ops/testing_utils.py
n/pyspark/pandas/tests/indexes/
n/pyspark/pandas/tests/indexes/__init__.py
n/pyspark/pandas/tests/indexes/test_base.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/indexes/test_category.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/indexes/test_datetime.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/__init__.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_frame_plot.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_frame_plot_matplotlib.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_frame_plot_plotly.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_series_plot.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_series_plot_matplotlib.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/plot/test_series_plot_plotly.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_categorical.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_config.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_csv.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_dataframe.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_dataframe_conversion.py
```

Saved successfully!

```
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_dataframe_spark_io.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_default_index.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_expanding.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_extension.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_frame_spark.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_groupby.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_indexing.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_indexops_spark.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_internal.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_namespace.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_numpy_compat.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_ops_on_diff_frames.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_ops_on_diff_frames_groupby.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_ops_on_diff_frames_groupby_expanding.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_ops_on_diff_frames_groupby_rolling.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_repr.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_reshape.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_rolling.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_series.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_series_conversion.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_series_datetime.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_series_string.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_spark_functions.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_sql.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_stats.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_typedef.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_utils.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/tests/test_window.py
spark-3.2.1-bin-hadoop3.2/python/pyspark/pandas/typedef/
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"


import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()


from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression



dataset = spark.read.csv('Combined.csv',inferSchema=True, header =True)


dataset.printSchema()
```

```
root
 |-- Price: integer (nullable = true)
 |-- Area: integer (nullable = true)
 |-- Location: integer (nullable = true)
 |-- Bedrooms: integer (nullable = true)
 |-- SwimmingPool: integer (nullable = true)
 |-- LandscapedGardens: integer (nullable = true)
 |-- JoggingTrack: integer (nullable = true)
 |-- RainWaterHarvesting: integer (nullable = true)
 |-- IndoorGames: integer (nullable = true)
 |-- ShoppingMall: integer (nullable = true)
 |-- Intercom: integer (nullable = true)
 |-- SportsFacility: integer (nullable = true)
 |-- ATM: integer (nullable = true)
 |-- ClubHouse: integer (nullable = true)
 |-- School: integer (nullable = true)
 |-- Security: integer (nullable = true)
 |-- PowerBackup: integer (nullable = true)
 |-- CarParking: integer (nullable = true)
 |-- StaffQuarter: integer (nullable = true)
```

Saved successfully!

```
 |-- Cafeteria: integer (nullable = true)
 |-- MultipurposeRoom: integer (nullable = true)
 |-- Hospital: integer (nullable = true)
 |-- WashingMachine: integer (nullable = true)
 |-- Gasconnection: integer (nullable = true)
 |-- AC: integer (nullable = true)
 |-- Wifi: integer (nullable = true)
 |-- Playarea: integer (nullable = true)
 |-- LiftAvailable: integer (nullable = true)
 |-- BED: integer (nullable = true)
 |-- VaastuCompliant: integer (nullable = true)
 |-- Microwave: integer (nullable = true)
 |-- GolfCourse: integer (nullable = true)
 |-- TV: integer (nullable = true)
 |-- DiningTable: integer (nullable = true)
 |-- Sofa: integer (nullable = true)
 |-- Wardrobe: integer (nullable = true)
 |-- Refrigerator: integer (nullable = true)
 |-- City: integer (nullable = true)
```

```
#Input all the features in one vector column
assembler = VectorAssembler(inputCols=['Area', 'Location', 'Bedrooms', 'Resale', 'MaintenanceStaff',
       'Gymnasium', 'SwimmingPool', 'LandscapedGardens', 'JoggingTrack',
       'RainWaterHarvesting', 'IndoorGames', 'ShoppingMall', 'Intercom',
       'SportsFacility', 'ATM', 'ClubHouse', 'School', 'Security',
       'PowerBackup', 'CarParking', 'StaffQuarter', 'Cafeteria',
       'MultipurposeRoom', 'Hospital', 'WashingMachine', 'Gasconnection', 'AC',
       'Wifi', 'Playarea', 'LiftAvailable', 'BED', 'VaastuCompliant',
       'Microwave', 'GolfCourse', 'TV', 'DiningTable', 'Sofa', 'Wardrobe',
       'Refrigerator', 'City'], outputCol = 'Attributes')

output = assembler.transform(dataset)

#Input vs Output
finalized_data = output.select("Attributes","Price")

finalized_data.show()
```

```
+--------------------+--------+
|          Attributes|   Price|
+--------------------+--------+
|(40,[0,1,2,5,6,7,...| 4866000|
|(40,[0,1,2,4,5,6,...| 6845000|
|(40,[0,1,2,5,6,7,...| 6797000|
|(40,[0,1,2,5,6,7,...|20000000|
|(40,[0,1,2,5,8,9,...| 7105000|
|(40,[0,1,2,5,6,7,...| 8405000|
|(40,[0,1,4,5,6,7,...| 3506000|
|(40,[0,1,2,5,6,7,...| 7700000|
|(40,[0,1,2,5,6,7,...| 9369000|
|(40,[0,1,2,5,6,7,...| 8716000|
|(40,[0,1,2,5,6,10...| 5394000|
|(40,[0,1,2,5,6,7,...| 6367000|
|(40,[0,1,2,5,6,7,...| 5080000|
|(40,[0,1,2,5,6,7,...| 7209999|
|(40,[0,1,2,4,5,6,...| 6845000|
|(40,[0,1,2,5,6,7,...| 6797000|
|(40,[0,1,2,5,6,7,...|20000000|
|(40,[0,1,2,5,8,9,...| 7105000|
+--------------------+--------+
only showing top 20 rows
```

Saved successfully!

```
#Split training and testing data
train_data,test_data = finalized_data.randomSplit([0.8,0.2])
```

```
regressor = LinearRegression(featuresCol = 'Attributes', labelCol = 'Price')

#Learn to fit the model from training set
regressor = regressor.fit(train_data)

#To predict the prices on testing set
pred = regressor.evaluate(test_data)

#Predict the model
pred.predictions.show()
```

```
/content/spark-3.2.1-bin-hadoop3.2/python/pyspark/sql/context.py:127: FutureWarning: Deprecated in 3.0.0. Use
  FutureWarning
+--------------------+--------+--------------------+
|          Attributes|   Price|          prediction|
+--------------------+--------+--------------------+
|(40,[0,1,2],[1015...| 4099000|   4927469.923736397|
|(40,[0,1,2],[1080...| 5250000|  5678402.7775553875|
|(40,[0,1,2],[1175...| 4465000|    5991326.24213133|
|(40,[0,1,2],[1180...| 8100000|   6859573.392029613|
|(40,[0,1,2],[1395...| 7100000|   9376255.260443237|
|(40,[0,1,2,3],[10...| 3975000|   6696176.949918272|
|(40,[0,1,2,3,4,5,...| 3825000|     5402704.3835182|
|(40,[0,1,2,3,4,5,...|15000000| 1.852488425735092E7|
|(40,[0,1,2,3,4,5,...|15500000| 2.134256098831273E7|
|(40,[0,1,2,3,4,5,...|22000000|3.0382607166815184E7|
|(40,[0,1,2,3,4,5,...|14200000| 1.731119998211448E7|
|(40,[0,1,2,3,4,5,...|22000000|1.6068450080900425E7|
|(40,[0,1,2,3,4,5,...|16000000|  1.56987462214335E7|
|(40,[0,1,2,3,4,5,...|11000000|1.0519725591918156E7|
|(40,[0,1,2,3,4,5,...|60000000| 5.763011451661664E7|
|(40,[0,1,2,3,4,5,...|11200000| 1.629845501466327E7|
|(40,[0,1,2,3,4,5,...|15000000|2.3405269225529574E7|
|(40,[0,1,2,3,4,5,...|21000000|  2.30978006476447E7|
|(40,[0,1,2,3,4,5,...| 6850000|   8027021.521001477|
|(40,[0,1,2,3,4,5,...|12500000|   7354817.709850356|
+--------------------+--------+--------------------+
only showing top 20 rows
```

```
#coefficient of the regression model
coeff = regressor.coefficients

#X and Y intercept
intr = regressor.intercept

print ("The coefficient of the model is : %a" %coeff)
print ("The Intercept of the model is : %f" %intr)
```

```
The coefficient of the model is : DenseVector([11740.3197, -174.1132, -802406.9081, 1886110.2233, 344532.3392
The Intercept of the model is : -6182194.844949
```

Saved successfully!

```
from pyspark.ml.evaluation import RegressionEvaluator
eval = RegressionEvaluator(labelCol="Price", predictionCol="prediction", metricName="rmse")

# Root Mean Square Error
rmse = eval.evaluate(pred.predictions)
print("RMSE: %.3f" % rmse)

# Mean Square Error
mse = eval.evaluate(pred.predictions, {eval.metricName: "mse"})
print("MSE: %.3f" % mse)

# Mean Absolute Error
mae = eval.evaluate(pred.predictions, {eval.metricName: "mae"})
```

```
print("MAE: %.3f" % mae)

# r2 - coefficient of determination
r2 = eval.evaluate(pred.predictions, {eval.metricName: "r2"})
print("r2: %.3f" %r2)
```

```
/content/spark-3.2.1-bin-hadoop3.2/python/pyspark/sql/context.py:127: FutureWarning: Deprecated in 3.0.0. Use
  FutureWarning
RMSE: 6292402.235
MSE: 39594325881181.195
MAE: 3030949.924
r2: 0.561
```

✓ 2s completed at 1:01 PM

Saved successfully!