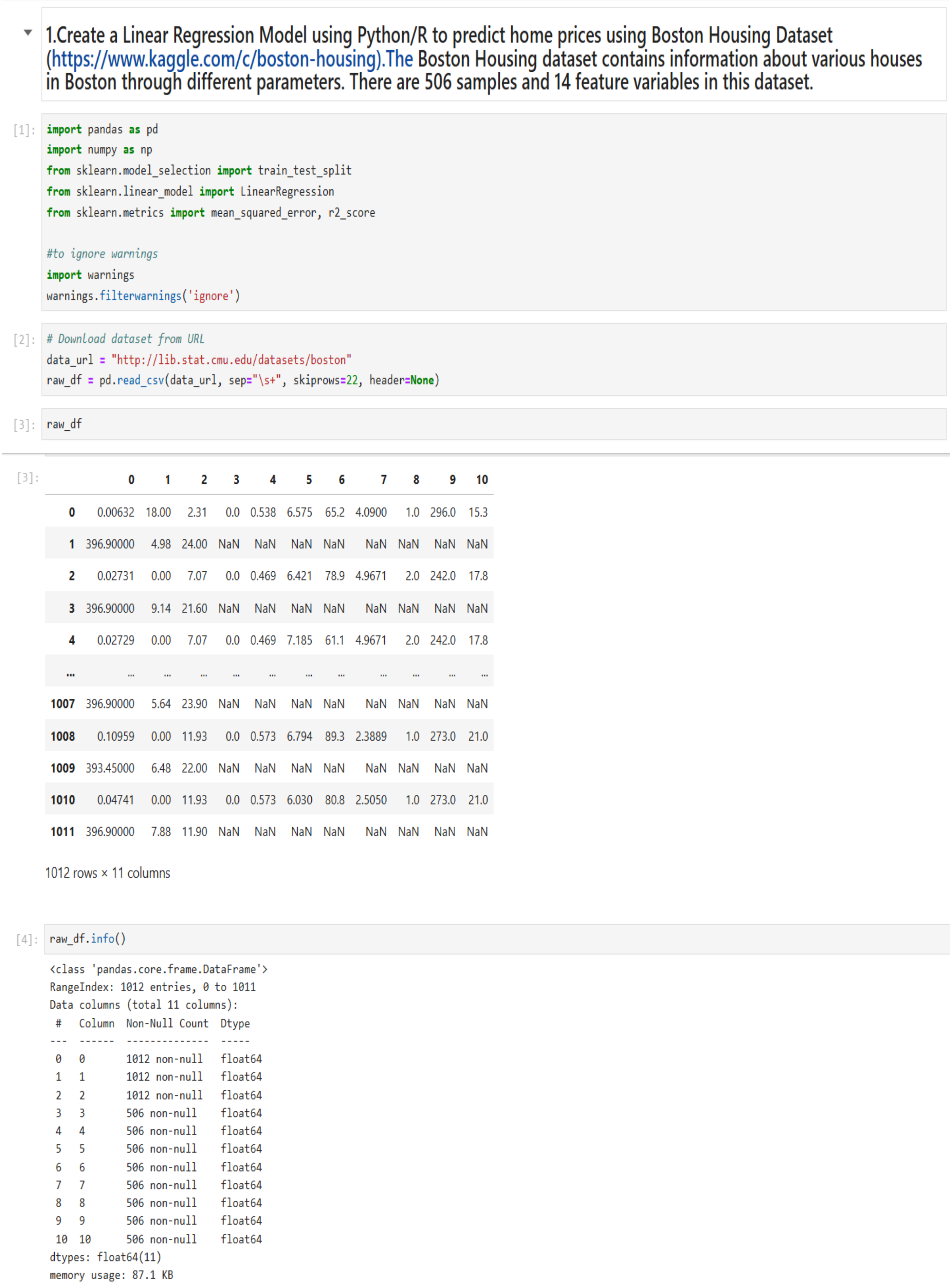
Practical No:4



1.Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing).The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

#to ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# Download dataset from URL
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
```

[3]:
```
raw_df
```

[3]:

	0	1	2	3	4	5	6	7	8	9	10
0	0.00632	18.00	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	396.90000	4.98	24.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	0.02731	0.00	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
3	396.90000	9.14	21.60	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.02729	0.00	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
...	...	...	...	...	...	...	...	...	...	...	...
1007	396.90000	5.64	23.90	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1008	0.10959	0.00	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0
1009	393.45000	6.48	22.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1010	0.04741	0.00	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0
1011	396.90000	7.88	11.90	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1012 rows × 11 columns

[4]:
```
raw_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1012 entries, 0 to 1011
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       1012 non-null   float64
 1   1       1012 non-null   float64
 2   2       1012 non-null   float64
 3   3       506 non-null    float64
 4   4       506 non-null    float64
 5   5       506 non-null    float64
 6   6       506 non-null    float64
 7   7       506 non-null    float64
 8   8       506 non-null    float64
 9   9       506 non-null    float64
 10  10      506 non-null    float64
dtypes: float64(11)
memory usage: 87.1 KB
```

[5]:
```
raw_df.isnull().sum()
```

[5]:
```
0       0
1       0
2       0
3     506
4     506
5     506
6     506
7     506
8     506
9     506
10    506
dtype: int64
```

[6]:
```
raw_df.describe()
```

[6]:

	0	1	2	3	4	5	6	7	8	9	10
count	1012.000000	1012.000000	1012.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	180.143778	12.008350	16.834792	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534
std	188.132839	17.250728	9.912616	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000
25%	0.257830	0.000000	8.375000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	24.021000	7.240000	18.100000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	391.435000	16.780000	21.890000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	396.900000	100.000000	50.000000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

[7]:
```
raw_df.dtypes
```

[7]:
```
0     float64
1     float64
2     float64
3     float64
4     float64
5     float64
6     float64
7     float64
8     float64
9     float64
10    float64
dtype: object
```

[ ]:

[8]:
```
# Preprocess dataset
X = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])  # Features
y = raw_df.values[1::2, 2]  # Target (House Prices)
```

[9]:
```
# Convert to DataFrame
columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']
df = pd.DataFrame(X, columns=columns)
df['PRICE'] = y
```

[10]:
```
# Splitting Dataset
X = df.drop('PRICE', axis=1)
y = df['PRICE']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[11]:
```
# Model Training
model = LinearRegression()
model.fit(X_train, y_train)
```

[11]:
```
▾ LinearRegression
LinearRegression()
```

[12]:
```
# Prediction
y_pred = model.predict(X_test)
```

[13]:
```
# Evaluation
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R-Squared Score:", r2_score(y_test, y_pred))
print("Predicted Prices:\n", y_pred[:5])
```

```
Mean Squared Error: 24.291119474973478
R-Squared Score: 0.6687594935356326
Predicted Prices:
 [28.99672362 36.02556534 14.81694405 25.03197915 18.76987992]
```