Assignment No: - DL-Group1-1

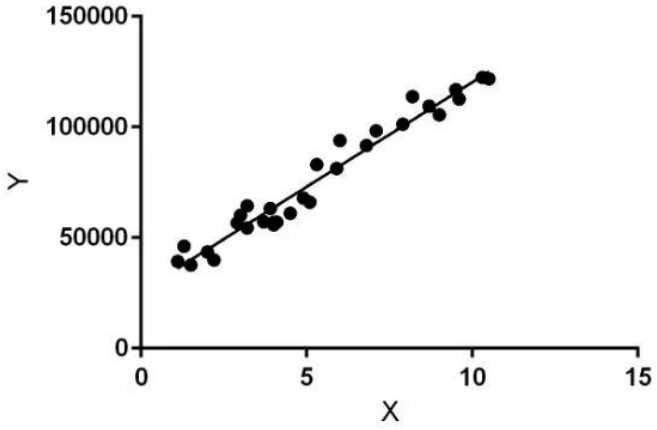
TITLE	Linear regression by using Deep Neural network
PROBLEM STATEMENT /DEFINITION	Implement Boston housing price prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.
OBJECTIVE	To build a regression model to predict the price of houses.
OUTCOME	To understand the exploratory data analysis,split the training and testing data, Model Evaluation and Prediction by the linear regression on the Boston housing dataset.
S/W PACKAGES AND HARDWARE/ APPARATUS USED	Jupyter notebook IDE, python3 PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. https://studygyaan.com/data-science-ml/linear-regression-machine-learning-project-for-house-price-prediction 2. https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155
STEPS	Installing Jupyter notebook with python3 1) import the required libraries-numpy,matplotlib.pyplot,pandas,seaborn 2) Importing Data (kaggle and scikit-learn library) and Checking out 3) Exploratory Data Analysis for House Price Prediction 4) Get Data Ready For Training a Linear Regression Model 5) Split Data into Train, Test 6) Creating and Training the LinearRegression Model 7) LinearRegression Model Evaluation 8) Predictions from our Linear Regression Model 9) Regression Evaluation Metrics.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition

	4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 9. Conclusion/Analysis

Prerequisites: Programming language, machine learning

Concepts related Theory:

Linear Regression is a Supervised Machine Learning Model for finding the relationship between independent variables and dependent variable. Linear regression performs the task to predict the response (dependent) variable value (y) based on a given (independent) explanatory variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output).



Algorithm:

Import Libraries: Install the required libraries and setup for the environment for the assignment. importing SciKit-Learn, Pandas, Seaborn, Matplotlib and Numpy.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

The purpose of "%matplotlib inline" is to add plots to your Jupyter notebook.

Importing Data and Checking out: As data is in the CSV file, we will read the CSV using pandas read_csv function and check the first 5 rows of the data frame using head().

```
HouseDF.info()
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income                5000 non-null float64
Avg. Area House Age             5000 non-null float64
Avg. Area Number of Rooms       5000 non-null float64
Avg. Area Number of Bedrooms    5000 non-null float64
Area Population                 5000 non-null float64
Price                           5000 non-null float64
Address                         5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

```
HouseDF.describe()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

```
HouseDF.columns
```

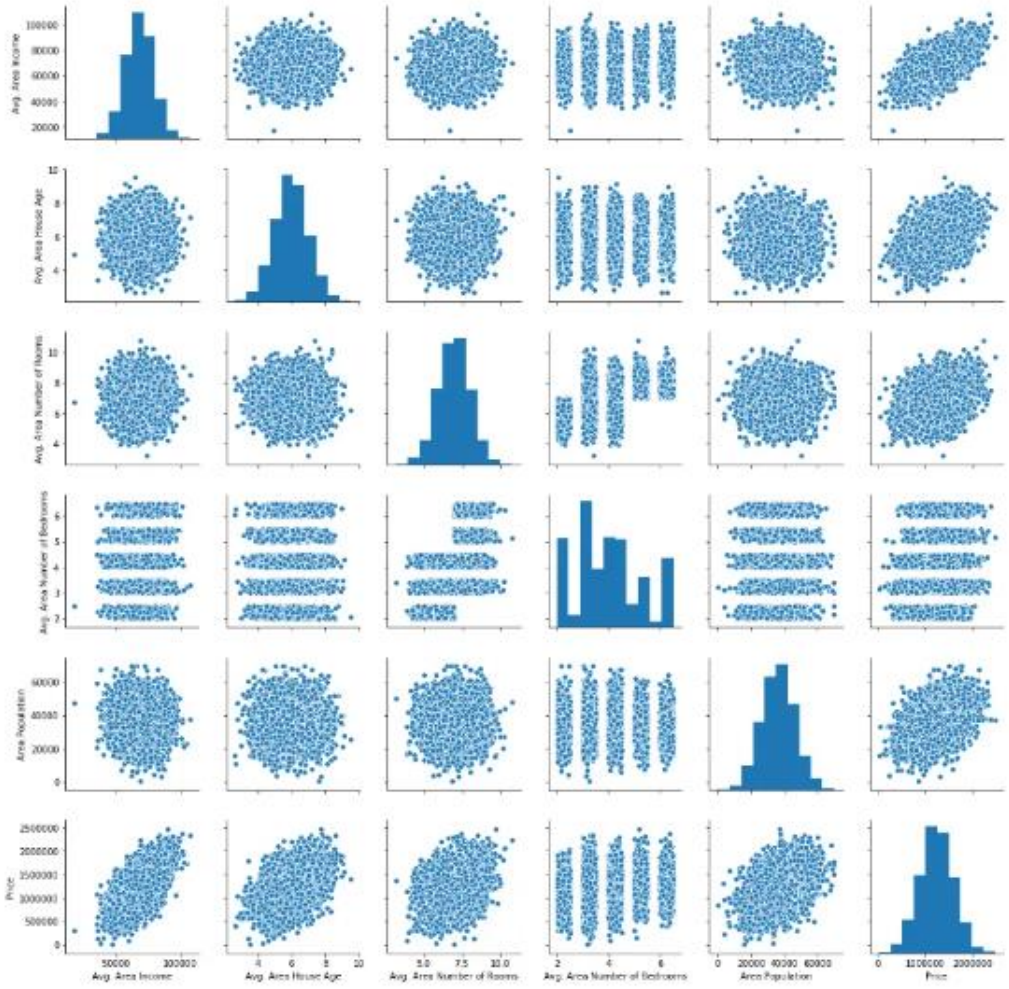
OUTPUT

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms','Avg. Area Numb
```

Exploratory Data Analysis for House Price Prediction : create some simple plot for visualizing the data.

```
sns.pairplot(HouseDF)
```

```
<seaborn.axisgrid.PairGrid at 0x67b60e8d08>
```



```
sns.heatmap(HouseDF.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x67b868c948>
```

Get Data Ready For Training a Linear Regression Model: now begin to train out the regression model. We will need to first split up our data into an X list that contains the features to train on, and a y list with the target variable, in this case, the Price column. We will ignore the Address column because it only has text which is not useful for linear regression modeling.

X and y List

```
X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
             'Avg. Area Number of Bedrooms', 'Area Population']]

y = HouseDF['Price']
```

Split Data into Train, Test: Now split our dataset into a training set and testing set using sklearn train_test_split(). The training set will be used for training the model and testing set for testing the model. We are creating a split of 40% training data and 60% of the training set.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

X_train and y_train contain data for the training model. X_test and y_test contain data for the testing model. X and y are features and target variable names.

Creating and Training the LinearRegression Model: import and create sklearn linear_model LinearRegression object and fit the training dataset in it.

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()

lm.fit(X_train,y_train)
```

```
OUTPUT

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

LinearRegression Model Evaluation: Now let's evaluate the model by checking out its coefficients and how we can interpret them

```
print(lm.intercept_)
```

```
OUTPUT

-2640159.796851911
```

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient']) coeff_df
```

	Coefficient
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

What does coefficient of data says:

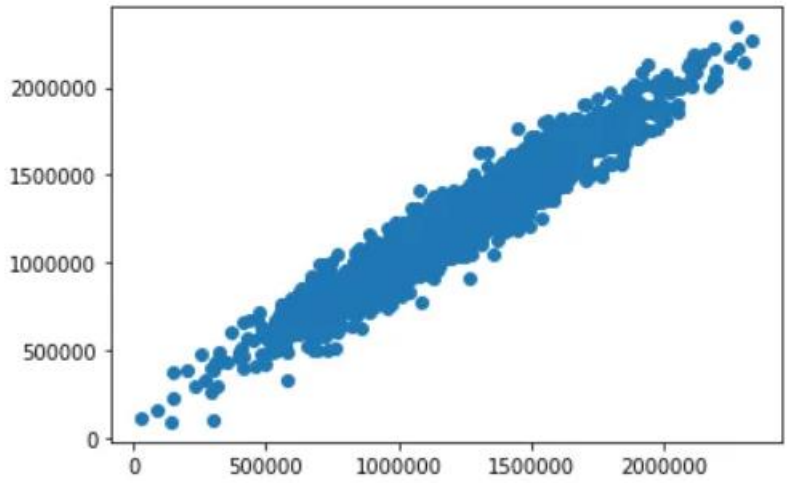
- Holding all other features fixed, a 1 unit increase in **Avg. Area Income** is associated with an **increase of $21.52** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area House Age** is associated with an **increase of $164883.28** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Rooms** is associated with an **increase of $122368.67** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Bedrooms** is associated with an **increase of $2233.80** .
- Holding all other features fixed, a 1 unit increase in **Area Population** is associated with an **increase of $15.15** .

Predictions from our Linear Regression Model: Let's find out the predictions of our test set and see how well it performs.

```
predictions = lm.predict(X_test)
```

```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x67b87ccc88>
```



In the above scatter plot, we see data is in a line form, which means our model has done good predictions.

Regression Evaluation Metrics:Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand because it's the average error.
- MSE is more popular than MAE because MSE "punishes" larger errors, which tends to be useful in the real world.
- RMSE is even more popular than MSE because RMSE is interpretable in the "y" units.

All of these are **loss functions** because we want to minimize them.

```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions)) print('MSE:', metrics.mean_squared_error(y_t
```

```
OUTPUT

MAE: 82288.22251914957
MSE: 10460958907.209501
RMSE: 102278.82922291153
```

Conclusion-We have analyzed a Linear Regression Model which we help the real estate agent for estimating the house price.

Review Questions:

1) Difference between machine learning and deep learning?
2) How is linear regression used in house price prediction?
3) What is the purpose of house price prediction?
4) What is a Linear Regression?
5) What is the primary difference between R square and adjusted R square?
6) List out the formulas to find RMSE and MSE?
7) What are the disadvantages of the linear regression model?
8) Name a possible method of improving the accuracy of a linear regression model?