Assignment No- 01

Title: Linear regression by using Deep Neural network: Implement Boston housing price.prediction problem by linear regression using Deep Neural network. Use Boston House price prediction dataset.

import numpy as npimport

pandas as pd

from sklearn.datasets import load_boston

boston = load_boston()

data = pd.DataFrame(boston.data)

data.head()

data.columns = boston.feature_names

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

data['PRICE'] = boston.target

data.head(n=10)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10	18.9

print(data.shape)

data.isnull().sum()

CRIM	0
ZN	0
INDUS	0
CHAS	0
NOX	0
RM	0
AGE	0
DIS	0
RAD	0
TAX	0
PTRATIO	0
B	0
LSTAT	0
PRICE	0

dtype: int64

data.describe()

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534
std	8.601545	23.322453	6.860353	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%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 506 entries, 0 to 505 Data columns

(total 14 columns):

#	Column	Non-Null Count	Dtype
0	CRIM	506 non-null	float64
1	ZN	506 non-null	float64
2	INDUS	506 non-null	float64
3	CHAS	506 non-null	float64
4	NOX	506 non-null	float64
5	RM	506 non-null	float64
6	AGE	506 non-null	float64
7	DIS	506 non-null	float64
8	RAD	506 non-null	float64
9	TAX	506 non-null	float64
10	PTRATIO	506 non-null	float64
11	B	506 non-null	float64
12	LSTAT	506 non-null	float64
13	PRICE	506 non-null	float64

dtypes: float64(14)

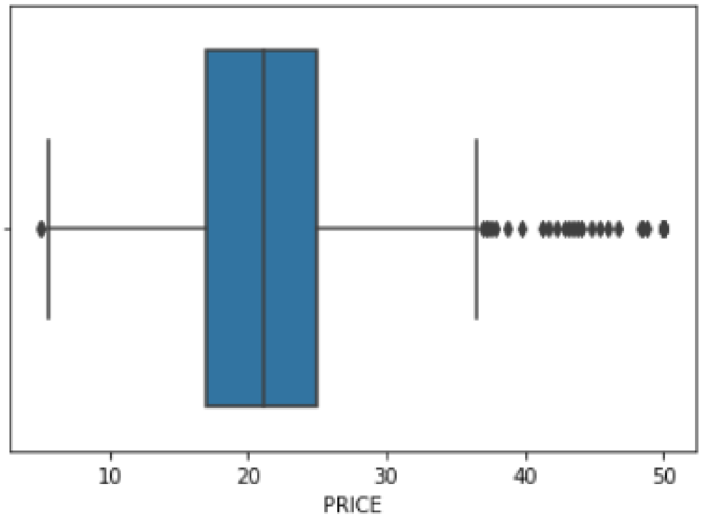
memory usage: 55.5 KB

import seaborn as sns

sns.distplot(data.PRICE)

sns.boxplot(data.PRICE)

<matplotlib.axes._subplots.AxesSubplot at 0x7f44d077ed60>



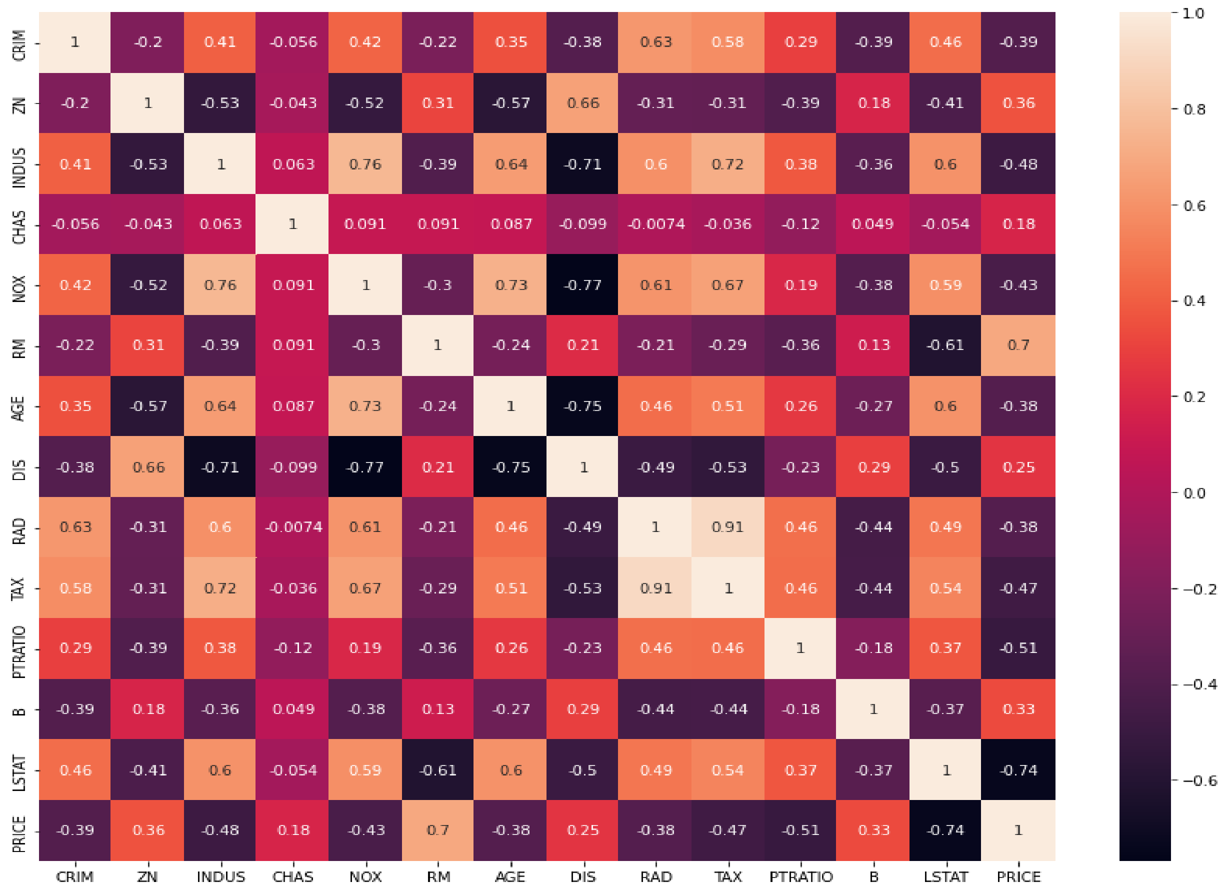
correlation = data.corr()

correlation.loc['PRICE']

CRIM	-0.388305
ZN	0.360445
INDUS	-0.483725
CHAS	0.175260
NOX	-0.427321
RM	0.695360
AGE	-0.376955
DIS	0.249929
RAD	-0.381626
TAX	-0.468536
PTRATIO	-0.507787
B	0.333461
LSTAT	-0.737663
PRICE	1.000000

Name: PRICE, dtype: float64

```
import matplotlib.pyplot as plt fig,axes =
plt.subplots(figsize=(15,12))
sns.heatmap(correlation,square = True,annot = True)
```



plt.figure(figsize = (20,5))

features = ['LSTAT','RM','PTRATIO']

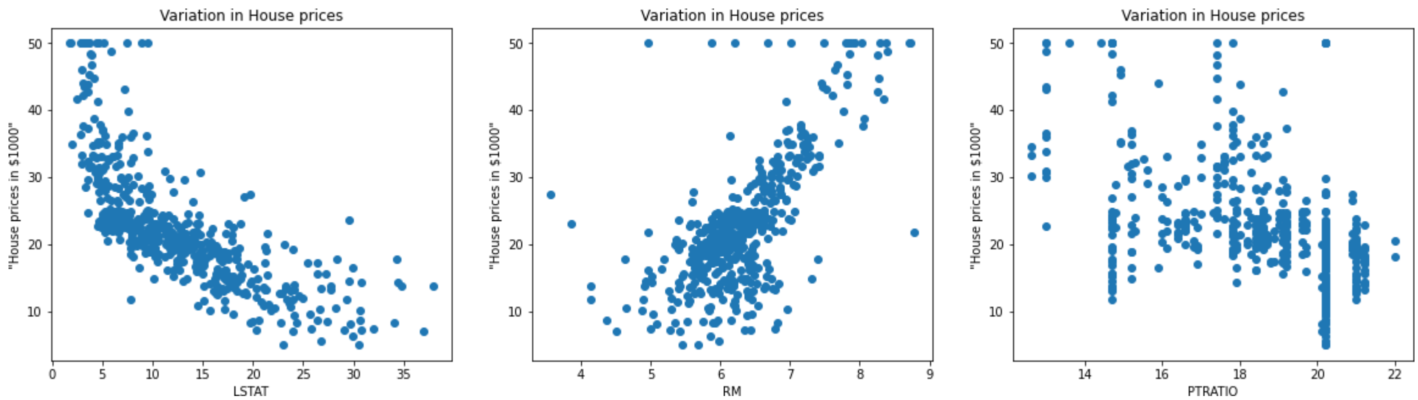
for i, col in enumerate(features): plt.subplot(1,

len(features) , i+1)x = data[col]

y = data.PRICE plt.scatter(x, y,

marker='o')
plt.title("Variation in House prices")
plt.xlabel(col)

plt.ylabel('"House prices in $1000"')



X = data.iloc[:,:-1]y=
data.PRICE

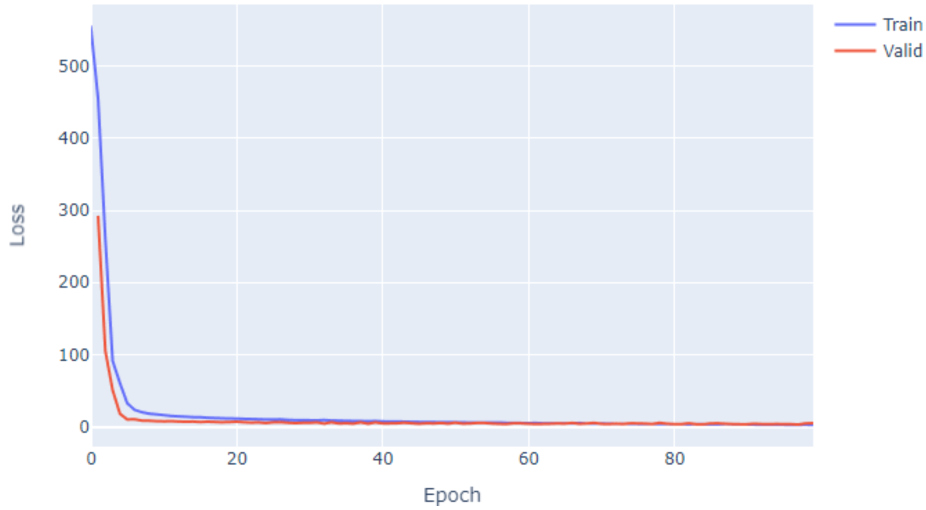
mean = X_train.mean(axis=0)std =
X_train.std(axis=0)

X_train = (X_train - mean) / stdX_test =
(X_test - mean) / std #Linear Regression

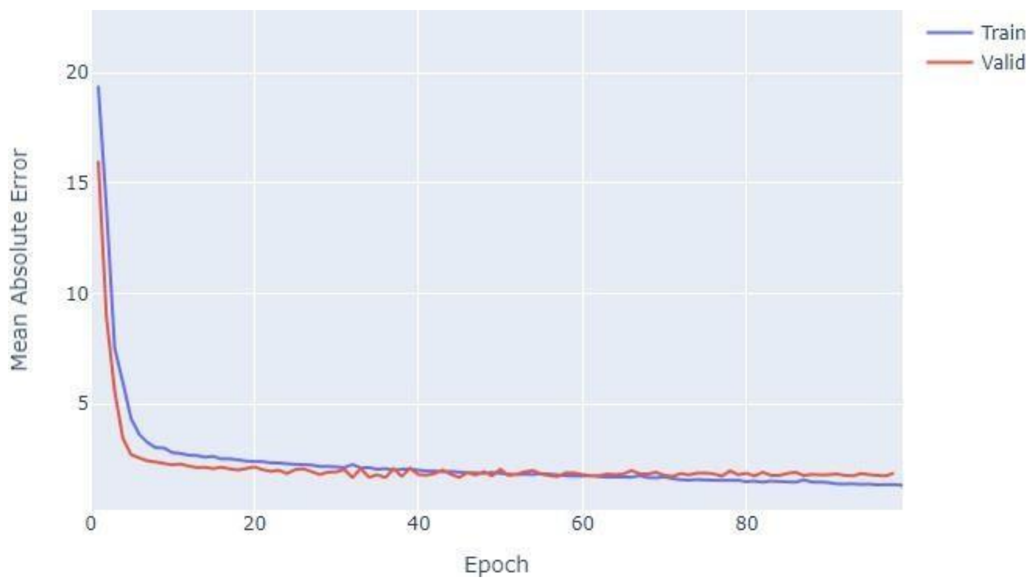
from sklearn.linear_model import LinearRegression

```
regressor = LinearRegression()
regressor.fit(X_train,y_train)

y_pred = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))print(rmse)

from sklearn.metrics import r2_scorer2 =
r2_score(y_test, y_pred) print(r2)

from sklearn.preprocessing import StandardScalersc =
StandardScaler()

X_train = sc.fit_transform(X_train)X_test =
sc.transform(X_test)

import keras

from keras.layers import Dense, Activation,Dropoutfrom
keras.models import Sequential

model = Sequential()
model.add(Dense(128,activation = 'relu',input_dim =13))model.add(Dense(64,activation   = 'relu'))
```

```
model.add(Dense(32,activation

='relu'))  model.add(Dense(16,activation
= 'relu')) model.add(Dense(1))

model.compile(optimizer = 'adam',loss
='mean_squared_error',metrics=['mae'])

!pip install ann_visualizer
!pip install graphviz
from ann_visualizer.visualize importann_viz;

ann_viz(model, title="DEMO ANN");
history = model.fit(X_train, y_train, epochs=100, validation_split=0.05)
from        plotly.subplots          import
make_subplots                        import
plotly.graph_objects as go

fig = go.Figure() fig.add_trace(go.Scattergl(y=history.history['loss'],

                    name='Train'))

fig.add_trace(go.Scattergl(y=history.history['val_loss'],
                    name='Valid'))


fig.update_layout(height=500, width=700,
                  xaxis_title='Epoch',
                  yaxis_title='Loss')
fig.show()
```




```
fig = go.Figure()

fig.add_trace(go.Scattergl(y=history.history['mae'],
                           name='Train'))

fig.add_trace(go.Scattergl(y=history.history['val_mae'],
                           name='Valid'))


fig.update_layout(height=500, width=700,
                  xaxis_title='Epoch',
                  yaxis_title='Mean AbsoluteError')

fig.show()
```



y_pred =
model.predict(X_test)

mse_nn, mae_nn = model.evaluate(X_test, y_test)

print('Mean squared error on test data: ', mse_nn) print('Mean
absolute error on test data:
', mae_nn)

4/4 [==============================] - 0s 4ms/step - loss: 10.5717 - mae: 2.2670

Mean squaíed eííoí on test data: 10.571733474731445 Mean absolute eííoí on test data: 2.2669904232025146

#Comparison with traditional
approaches

#First let's try with a simple algorithm, the Linear Regression:from sklearn.metrics
import mean_absolute_error

lr_model =
LinearRegression()
lr_model.fit(X_train,
y_train)

y_pred_lr = lr_model.predict(X_test)

mse_lr = mean_squared_error(y_test, y_pred_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)

print('Mean squared error on test data: ', mse_lr) print('Mean
absolute error on test data:
', mae_lr)from sklearn.metrics import r2_score

r2 = r2_score(y_test,
y_pred)print(r2)

0.8812832788381159

Predicting RMSE the Test set results

from sklearn.metrics import mean_squared_error

rmse = (np.sqrt(mean_squared_error(y_test,
y_pred)))print(rmse)

3.320768607496587

Make predictions on new
dataimport sklearn

new_data = sklearn.preprocessing.StandardScaler().fit_transform((([[0.1,10.0,5.0, 0, 0.4, 6.0,
50, 6.0, 1, 400, 20, 300, 10]]))

prediction = model.predict(new_data)
print("Predicted house price:", prediction)

1/1
[==============================] -
0s 70ms/step

Predicted house price: [[11.104753]]