DL Pac 1

In [ ]:
```
import numpy as np
import pandas as pd
```

In [ ]:
```
from sklearn.datasets import load_boston
```

In [ ]:
```
boston = load_boston()
```

In [ ]:
```
pip install --upgrade numpy==1.24.0 scipy==1.24.0
```

In [5]:
```
pip install --upgrade numpy scipy
```

```
Requirement already satisfied: numpy in c:\users\rushi\anaconda3\lib\site-
packages (1.26.4)
Requirement already satisfied: scipy in c:\users\rushi\anaconda3\lib\site-
packages (1.9.1)
Collecting scipy
  Downloading scipy-1.12.0-cp39-cp39-win_amd64.whl (46.2 MB)
     ---------------------------------------- 46.2/46.2 MB 3.5 MB/s eta 0:
00:00
Installing collected packages: scipy
  Attempting uninstall: scipy
    Found existing installation: scipy 1.9.1
    Uninstalling scipy-1.9.1:
      Successfully uninstalled scipy-1.9.1
Successfully installed scipy-1.12.0
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
from sklearn.datasets import load_boston
```

In [7]:
```
boston = load_boston()
```

In [8]:
```
#Converting the data into pandas dataframe
data = pd.DataFrame(boston.data)
#First look at the data
data.head()
```

Out[8]:

	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

In [9]:
```
#Adding the feature names to the dataframe
data.columns = boston.feature_names
#Adding the target variable to the dataset
data['PRICE'] = boston.target
#Looking at the data with names and target variable
data.head(n=10)
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12
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
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
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
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

In [10]:
```
#Shape of the data
print(data.shape)
#Checking the null values in the dataset
data.isnull().sum()
```

```
(506, 14)
```

Out[10]:
```
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
```

In [11]:
```
#Checking the statistics of the data
data.describe()
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	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
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

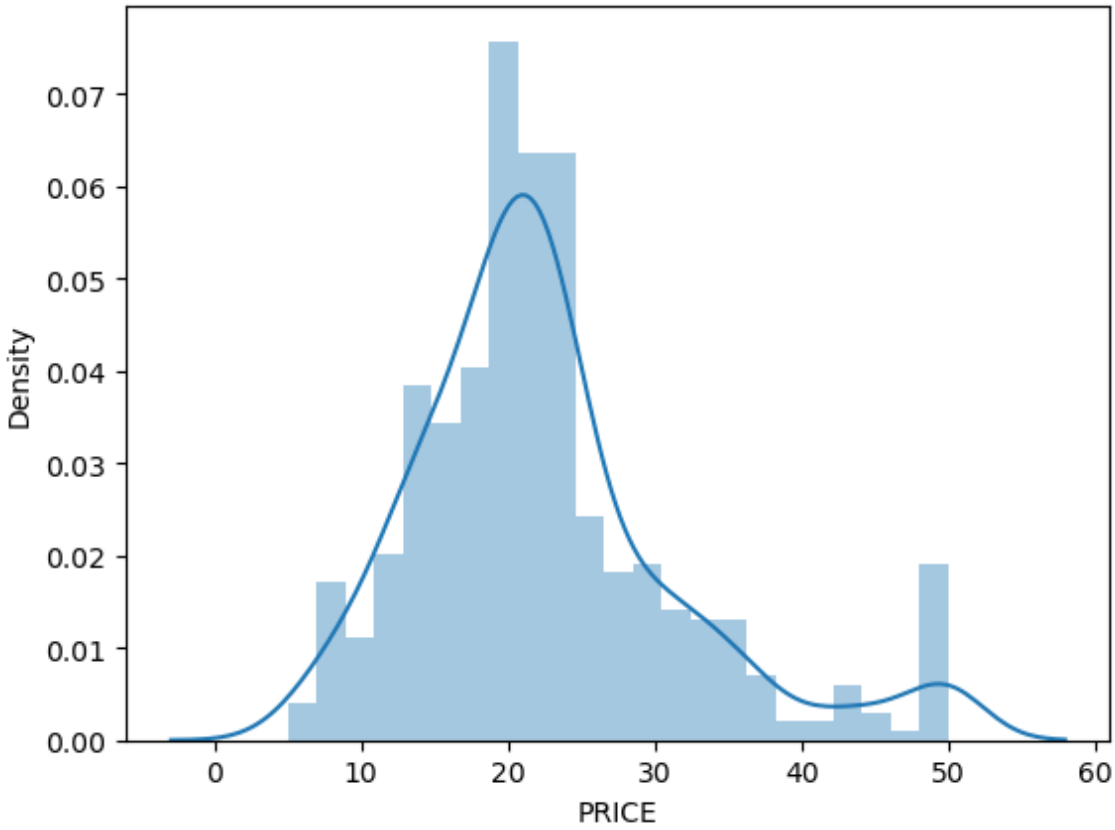
In [12]:
```
data.info()
```

```
<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
```

In [13]:
```
#checking the distribution of the target variable
import seaborn as sns
sns.distplot(data.PRICE)
```

```
C:\Users\rushi\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

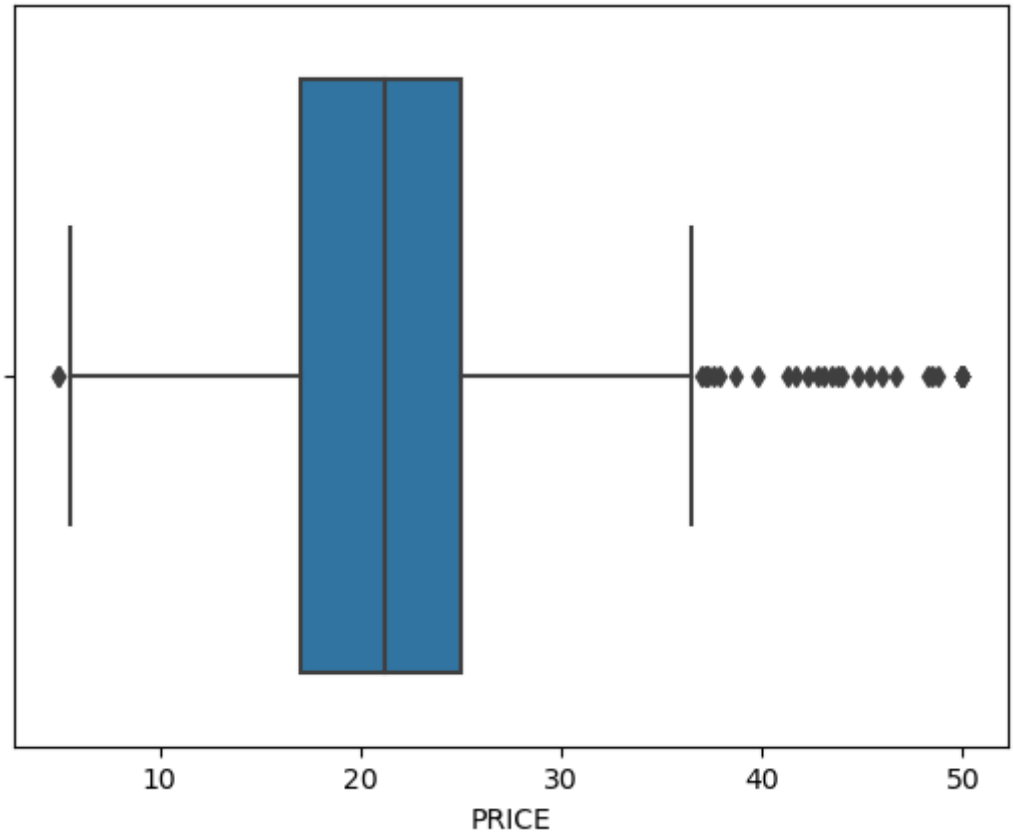
Out[13]:
```
<AxesSubplot:xlabel='PRICE', ylabel='Density'>
```



In [14]:
```
sns.boxplot(data.PRICE)
```

```
C:\Users\rushi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[14]:
```
<AxesSubplot:xlabel='PRICE'>
```



In [15]:
```
correlation = data.corr()
correlation.loc['PRICE']
```

Out[15]:
```
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
```

In [19]:
```
# plotting the heatmap
import matplotlib.pyplot as plt
```

In [20]:
```
fig,axes = plt.subplots(figsize=(15,12))
sns.heatmap(correlation,square = True,annot = True)
```

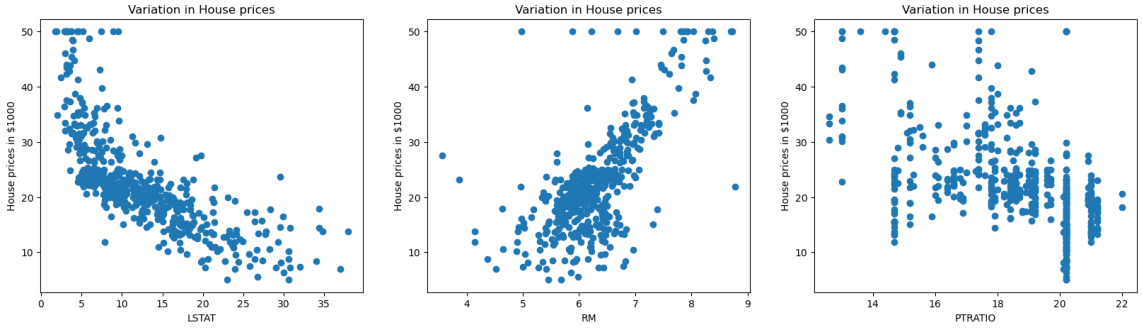
Out[20]: <AxesSubplot:>

In [26]:
```
import matplotlib.pyplot as plt

# Checking the scatter plot with the most correlated features
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")

plt.show()
```



In [27]:
```
# Splitting the dependent feature and independent feature
#X = data[['LSTAT','RM','PTRATIO']]
X = data.iloc[:,:-1]
y= data.PRICE
```

In [28]:
```
mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train = (X_train - mean) / std
X_test = (X_test - mean) / std
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_14092\626873035.py in <module>
----> 1 mean = X_train.mean(axis=0)
      2 std = X_train.std(axis=0)
      3 X_train = (X_train - mean) / std
      4 X_test = (X_test - mean) / std

NameError: name 'X_train' is not defined
```

In [30]:
```
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

# Assuming you have your dataset loaded and features (X) and target variabl
# X and y should be your features and target variable respectively

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Standardize the features
mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train = (X_train - mean) / std
X_test = (X_test - mean) / std

# Linear Regression
regressor = LinearRegression()

# Fitting the model
regressor.fit(X_train, y_train)

# Model Evaluation
# Prediction on the test dataset
y_pred = regressor.predict(X_test)

# Calculate RMSE (Root Mean Squared Error)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", rmse)

# Calculate R-squared (R2) score
r2 = r2_score(y_test, y_pred)
print("R-squared (R2) Score:", r2)
```

```
RMSE: 4.928602182665336
R-squared (R2) Score: 0.6687594935356321
```

In [31]:
```
# Neural Networks
#Scaling the dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [38]:
```
!pip install graphviz
!pip install ann_visualizer
```

```
Collecting graphviz
  Downloading graphviz-0.20.1-py3-none-any.whl (47 kB)
     ---------------------------------------- 47.0/47.0 kB 2.3 MB/s eta 0:
00:00
Installing collected packages: graphviz
Successfully installed graphviz-0.20.1
Requirement already satisfied: ann_visualizer in c:\users\rushi\anaconda3
\lib\site-packages (2.5)
```

In [41]:
```
# Importing necessary libraries
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from ann_visualizer.visualize import ann_viz
import plotly.subplots as sp
import plotly.graph_objects as go

# Creating the neural network model
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))

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

In [48]:
```
from ann_visualizer.visualize import ann_viz
```

In [53]:
```
pip install keras matplotlib plotly
```

```
Requirement already satisfied: keras in c:\users\rushi\anaconda3\lib\site-
packages (2.15.0)
Requirement already satisfied: matplotlib in c:\users\rushi\anaconda3\lib
\site-packages (3.5.2)
Requirement already satisfied: plotly in c:\users\rushi\anaconda3\lib\site
-packages (5.9.0)
Requirement already satisfied: cycler>=0.10 in c:\users\rushi\anaconda3\li
b\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\users\rushi\anaconda3\lib
\site-packages (from matplotlib) (1.26.4)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\rushi\anaconda
3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\rushi\anac
onda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\rushi\anaconda3\l
ib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: packaging>=20.0 in c:\users\rushi\anaconda3
\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\rushi\anacond
a3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\rushi\anacond
a3\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\rushi\anaconda3
\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six>=1.5 in c:\users\rushi\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [55]:
```
# Training the model
history = model.fit(X_train, y_train, epochs=100, validation_split=0.05, ca

from plotly.subplots import make_subplots
import plotly.graph_objects as go

# Plotting the training and validation loss
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='
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='
fig.show()
```

```
Epoch 34/100
12/12 [==============================] - 0s 6ms/step - loss: 9.4964 - m
ae: 2.2495 - val_loss: 30.8409 - val_mae: 3.2702
Epoch 35/100
12/12 [==============================] - 0s 5ms/step - loss: 9.0812 - m
ae: 2.1893 - val_loss: 32.3939 - val_mae: 3.4359
Epoch 36/100
12/12 [==============================] - 0s 5ms/step - loss: 9.2167 - m
ae: 2.2004 - val_loss: 31.2514 - val_mae: 3.2967
Epoch 37/100
12/12 [==============================] - 0s 5ms/step - loss: 9.1979 - m
ae: 2.2267 - val_loss: 30.7725 - val_mae: 3.1621
Epoch 38/100
12/12 [==============================] - 0s 5ms/step - loss: 8.9466 - m
ae: 2.1851 - val_loss: 32.7635 - val_mae: 3.4431
Epoch 39/100
12/12 [==============================] - 0s 5ms/step - loss: 8.7044 - m
ae: 2.1368 - val_loss: 29.1210 - val_mae: 3.1826
```

In [56]:
```
#Evaluation of the model
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 5ms/step
4/4 [==============================] - 0s 9ms/step - loss: 10.4311 - mae:
2.0978
Mean squared error on test data:  10.431135177612305
Mean absolute error on test data:  2.097837448120117
```

In [58]:
```
#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)
```

```
Mean squared error on test data:  24.291119474973502
Mean absolute error on test data:  3.1890919658878474
0.857758107757423
```

In [59]:
```
# 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)
```

```
3.229726936140448
```

In [60]:
```
# Make predictions on new data
import sklearn
new_data = sklearn. preprocessing.StandardScaler().fit_transform(([[0.1, 10
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 42ms/step
Predicted house price: [[10.27167]]
```