import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

data\_url = "http://lib.stat.cmu.edu/datasets/boston"

raw\_df = pd.read\_csv(data\_url, sep="\s+", skiprows=22, header=None)

data = np.hstack([raw\_df.values[::2, :], raw\_df.values[1::2, :2]])

target = raw\_df.values[1::2, 2]

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

import keras

from keras.layers import Dense, Activation,Dropout

from keras.models import Sequential

import warnings

warnings.filterwarnings("ignore")

--

import pandas as pd

import numpy as np

data\_url = "http://lib.stat.cmu.edu/datasets/boston"

raw\_df = pd.read\_csv(data\_url, sep="\s+", skiprows=22, header=None)

data = np.hstack([raw\_df.values[::2, :], raw\_df.values[1::2, :2]])

target = raw\_df.values[1::2, 2]

*# Creating a DataFrame*

columns = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"]

data = pd.DataFrame(data, columns=columns)

data["PRICE"] = target

data.head()

--

*# Data Exploration*

print(data.shape)

print(data.dtypes)

print(data.isnull().sum())

print(data.describe())

--

*# Data Visualization*

sns.displot(data.PRICE)

correlation = data.corr()

correlation.loc['PRICE']

fig,axes = plt.subplots(figsize=(15,12))

sns.heatmap(correlation,square = True,annot = True)

--

*# Splitting Data into testing and training data*

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

y= data.PRICE

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.2, random\_state = 4)

*# Normalizing the data*

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

In [ ]:

*# Model Building*

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')

model.summary()

--

*# Fitting the data to the model*

model.fit(X\_train, y\_train, epochs = 100)

--

*# Splitting Data into testing and training data*

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

y= data.PRICE

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.2, random\_state = 4)

*# Normalizing the data*

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

*# Model Building*

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')

model.summary()

--

*# Fitting the data to the model*

model.fit(X\_train, y\_train, epochs = 100)

--

*# Evaluating the model*

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

rmse = (np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("R2 Score = ", r2)

print("RMSE Score = ", rmse)