```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.neural_network import MLPRegressor

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error


# Set a random seed for reproducibility

np.random.seed(201681)

N = 300


# Generate Input Variables

X1 = (9-2)*(np.random.random(N)+2)

X2 = np.random.chisquare(5, N)

X3 = np.random.chisquare(2, N)

E = np.random.normal(3, 3, N)

Y1 = 0.35*X1**3 + 2*X2 + 5*X3 + E


# Combine input and output variables into a DataFrame

df = np.column_stack((X1, X2, X3, Y1))

D = pd.DataFrame(df, columns=["X1", "X2", "X3", "Y1"])


# Check for Missing Values in Each Column

print(D.isna().sum(axis=0))  # Add print to see the result


# Divide the Input Data and Output Data into Training and Testing Sets

X = D.drop(["Y1"], axis=1)
```

```
Y = D["Y1"]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, train_size=0.75,
random_state=1111)


# Initializing the MinMaxScaler

Scale = MinMaxScaler()


# Scaling the training data

X_train_scaled = pd.DataFrame(Scale.fit_transform(X_train), columns=X_train.columns)

Y_train_scaled = pd.DataFrame(Scale.fit_transform(Y_train.values.reshape(-1, 1)), columns=["Y"])


# Scaling the testing data using the same scaler fitted on the training data for X and Y

X_test_scaled = pd.DataFrame(Scale.fit_transform(X_test), columns=X_test.columns)

Y_test_scaled = pd.DataFrame(Scale.fit_transform(Y_test.values.reshape(-1, 1)), columns=["Y"])


# Multi-layer Perceptron Regressor (ANN Model Fitting)

ANN = MLPRegressor(hidden_layer_sizes=(3, 2), activation="logistic", solver="lbfgs",
max_iter=99999999, random_state=1111)

ANN.fit(X_train_scaled, Y_train_scaled.values.ravel())  # Use values.ravel() to flatten the target array


# Predict on Training and Testing Data

Y_train_scaled_predict = ANN.predict(X_train_scaled)

Y_test_scaled_predict = ANN.predict(X_test_scaled)


# Rescale Predictions to Original Scale using inverse_transform

Y_train_predicted = pd.DataFrame(Scale.inverse_transform(Y_train_scaled_predict.reshape(-1, 1)))

Y_test_predicted = pd.DataFrame(Scale.inverse_transform(Y_test_scaled_predict.reshape(-1, 1)))


# Calculate Training Metrics
```

```
mse_train = mean_squared_error(Y_train, Y_train_predicted)
mae_train = mean_absolute_error(Y_train, Y_train_predicted)
mape_train = mean_absolute_percentage_error(Y_train, Y_train_predicted)
r2_train = r2_score(Y_train, Y_train_predicted)

# Calculate Testing Metrics
mse_test = mean_squared_error(Y_test, Y_test_predicted)
mae_test = mean_absolute_error(Y_test, Y_test_predicted)
mape_test = mean_absolute_percentage_error(Y_test, Y_test_predicted)
r2_test = r2_score(Y_test, Y_test_predicted)

print(f"Training Metrics:\nMSE: {mse_train}, MAE: {mae_train}, MAPE: {mape_train}, R2: {r2_train}")
print(f"Testing Metrics:\nMSE: {mse_test}, MAE: {mae_test}, MAPE: {mape_test}, R2: {r2_test}")

# Plot Actual vs Predicted
plt.figure(figsize=(6, 4))
plt.scatter(Y_train, Y_train_predicted, color="red", label="Training", s=20)
plt.scatter(Y_test, Y_test_predicted, color="blue", label="Test", s=20)
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted")
plt.legend()
plt.show()
```