Practice exercise 4
Multilayer Perceptron Regression Exercise: Predicting House Prices

Problem Statement: Given a dataset of houses with features like square footage, number of bedrooms, and location, predict the house price using a Multilayer Perceptron (MLP) model.

Dataset: You can use a popular dataset like the Boston Housing dataset, which is available in scikit-learn.

Task:

1. **Import Necessary Libraries.**
   import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   from sklearn.model_selection import train_test_split
   from sklearn.preprocessing import StandardScaler
   import tensorflow as tf
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Dense

2. **Load and Preprocess the Data:**
   - Load the Boston Housing dataset.
   - Split the data into features (X) and target variable (y).
   - Scale the features using StandardScaler.
   - Split the data into training and testing sets.
3. **Create a Multilayer Perceptron Model:**
   - Define a Sequential model.
   - Add Dense layers with appropriate activation functions (e.g., ReLU for hidden layers and linear for the output layer).
   - Compile the model with an optimizer (e.g., Adam) and loss function (e.g., Mean Squared Error mse).
4. **Train the Model:**
   - Fit the model to the training data.
   - Monitor the training process using metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).
5. **Evaluate the Model:**
   - Evaluate the model on the testing set using the same metrics.
   - Make predictions on the testing set and compare them to the actual value

Code Implementation:

```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

# Load the Boston Housing dataset
boston = load_boston()
```

```
X = boston.data
y = boston.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,   random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create  the MLP model
model = Sequential([ Dense(32, activation='relu', input_dim=X_train.shape[1]), Dense(16,
activation='relu'), Dense(1) ])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train_scaled,  y_train,  epochs=100, batch_size=32,
validation_data=(X_test_scaled, y_test))

# Evaluate the model
test_loss = model.evaluate(X_test_scaled, y_test)
print('Test Loss:', test_loss)

# Make predictions
y_pred = model.predict(X_test_scaled)

# Visualize the predictions
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual  vs. Predicted House Prices')
 plt.show()
```

6. **Compare the performance metrics and analyze the results.**
    - Predict the house price using various regression models: Linear Regression, Decision Tree Regression, Random Forest Regression on the testing set
    - Calculate MSE and R-squared scores for each model.
    - Compare the performance metrics with obtained predicted results of MLP model and analyze the results.