Practical No. 10

Submitted By:		
Name: Mahak Mor	**Roll No.:** 20	**Section:** A

Aim: To implement a Feed-Forward Neural Network (FNN) based estimation using Scikit-learn.

Theory: A Feedforward Neural Network (FNN) is a type of artificial neural network where information flows in one direction—from input nodes, through hidden nodes (if any), to output nodes. Unlike recurrent networks, it has no cycles or loops.

Key Concepts in Feedforward Neural Networks:

1. Structure of FNN:
    - Input Layer: Receives input data.
    - Hidden Layers: Layers of neurons that transform the input into more abstract representations. Each neuron applies a weighted sum of inputs followed by a non-linear activation function.
    - Output Layer: Provides the final prediction. In regression, the output layer typically has a single node (for predicting continuous values).
2. Activation Functions:
    - Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh. ReLU is often used in hidden layers, while no activation function is applied in the output layer for regression tasks.
3. Backpropagation and Optimization:
    - The network uses backpropagation to adjust weights based on the error between predicted and actual values. The error is minimized using an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam.
4. Loss Function:
    - For regression tasks, the Mean Squared Error (MSE) is commonly used as the loss function, which measures the average squared difference between predicted and actual values.

Feedforward Neural Networks are powerful for modeling non-linear relationships and can outperform simpler models like linear regression, particularly on complex datasets.

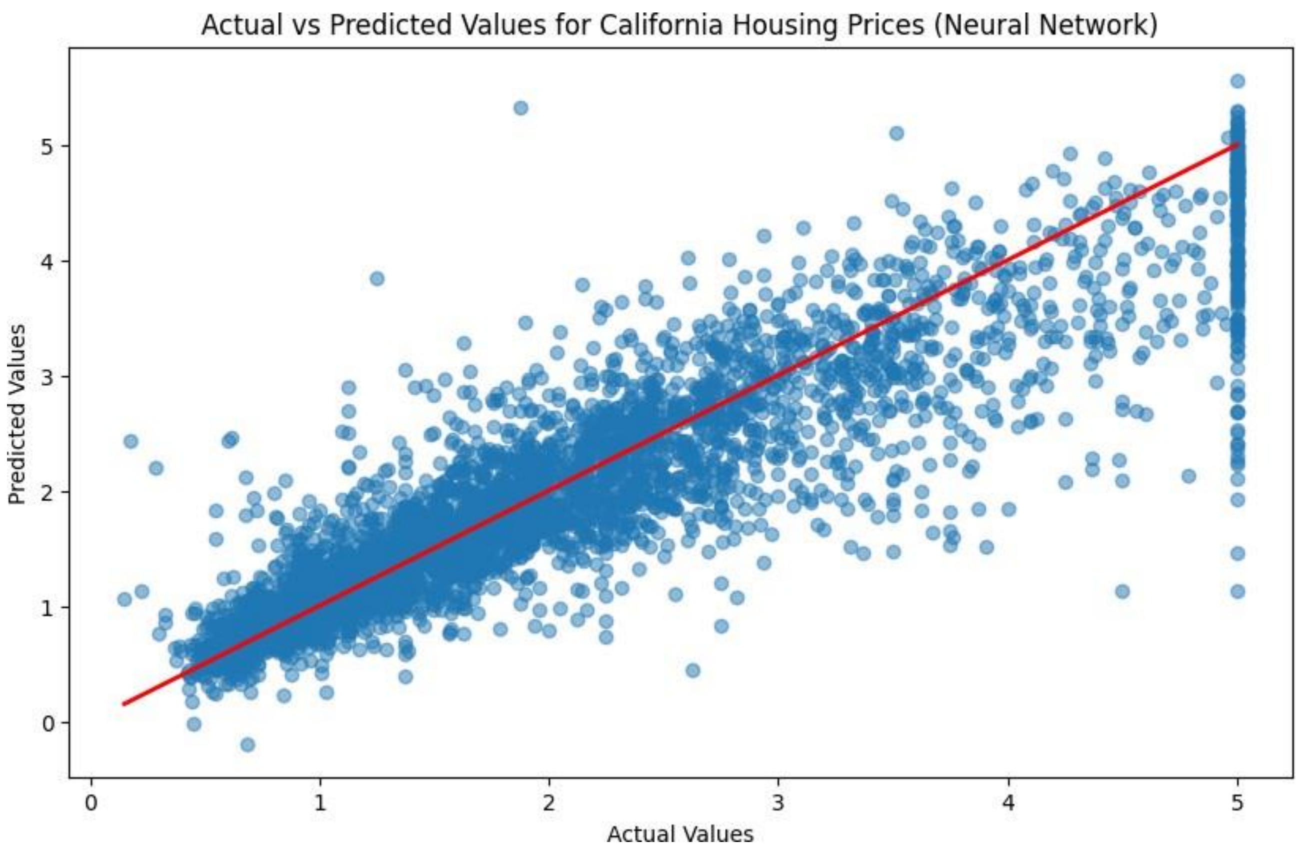
Code and Output:

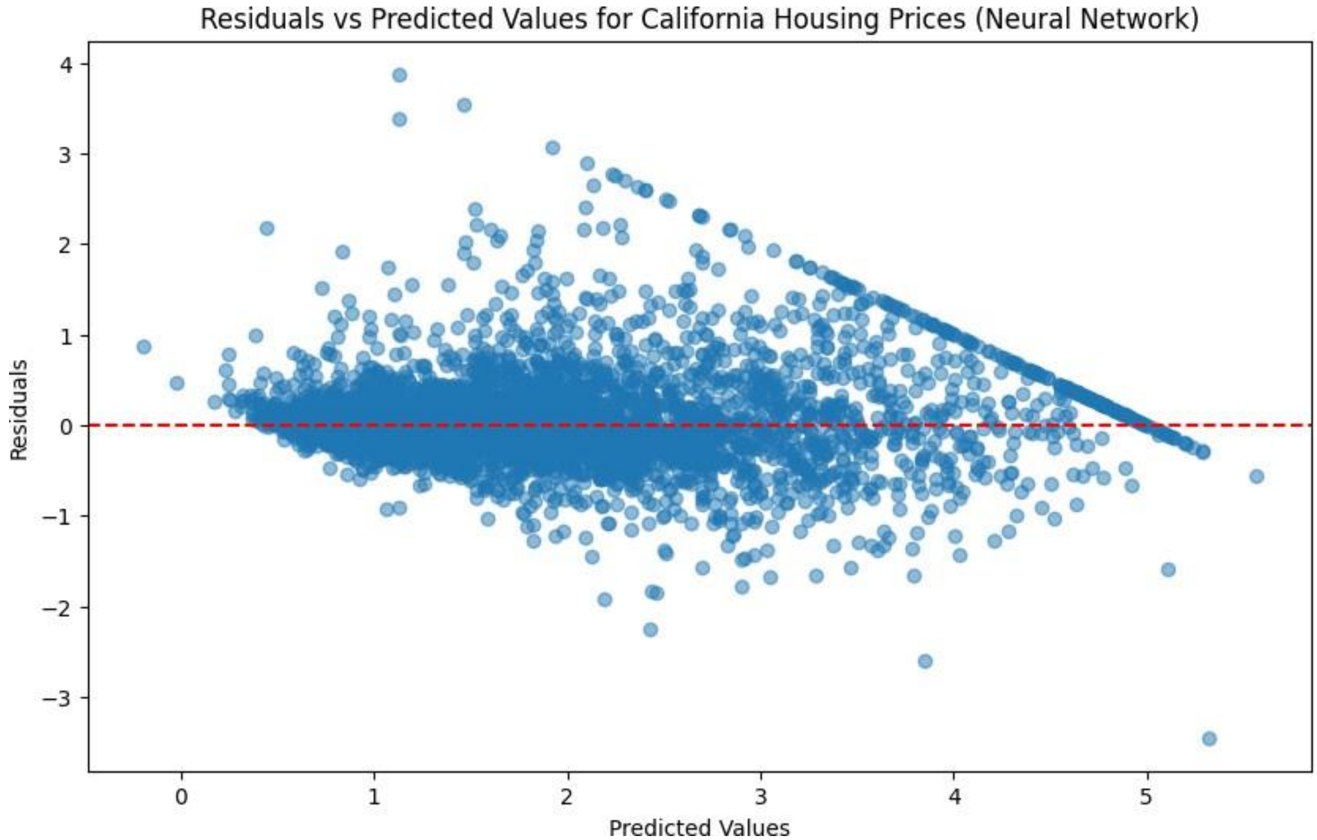
```
# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.neural_network import MLPRegressor

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.datasets import fetch_california_housing

from sklearn.preprocessing import StandardScaler

# Load and standardize the dataset

housing = fetch_california_housing()

X = pd.DataFrame(housing.data, columns=housing.feature_names)

y = pd.Series(housing.target)

# Splitting dataset 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)

# Standardize features for better neural network performance

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Initialize and train the Feedforward Neural Network (MLPRegressor)

model = MLPRegressor(hidden_layer_sizes=(64, 32), activation='relu',
solver='adam', max_iter=1000, random_state=42)

model.fit(X_train_scaled, y_train)

# Make predictions

y_pred = model.predict(X_test_scaled)
```

```python
# Model Evaluation

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

# Display results

print("Mean Squared Error:", mse)

print("R-squared Score:", r2)


# Importing visualization libraries

import matplotlib.pyplot as plt

# Scatter plot of Actual vs Predicted values

plt.figure(figsize=(10, 6))

plt.scatter(y_test, y_pred, alpha=0.5)

plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 
color='red', linewidth=2)  # Line of perfect prediction

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.title("Actual vs Predicted Values for California Housing Prices (Neural 
Network)")

plt.show()

# Plotting residuals

residuals = y_test - y_pred

plt.figure(figsize=(10, 6))

plt.scatter(y_pred, residuals, alpha=0.5)

plt.axhline(y=0, color='red', linestyle='--')

plt.xlabel("Predicted Values")

plt.ylabel("Residuals")
```

```
plt.title("Residuals vs Predicted Values for California Housing Prices (Neural Network)")

plt.show()
```

```
Mean Squared Error: 0.2742889195569183
R-squared Score: 0.7906844930770199
```





Conclusion: In this practical, a Feed-Forward Neural Network (FNN) was implemented using Scikit-learn's MLPRegressor to perform regression on the California Housing dataset. The neural network performed well with an R^2 score of 0.79, showing that it was able to capture the underlying patterns in the data. The MSE value further supported the accuracy of the predictions. FNNs are powerful tools for regression tasks, but they require careful tuning of hyperparameters such as the number of hidden layers, activation functions, and learning rate to achieve optimal performance.