Experiment-7

Aim:

To train a Feed Forward Neural Network (FFNN) using Back Propagation algorithm for a classification task.

Apparatus Required:

- Python (Anaconda/Miniconda/Standard Python Installation)
- Jupyter Notebook or any Python IDE
- Libraries: TensorFlow (Keras), NumPy, Matplotlib, scikit-learn

Theory:

Feed Forward Neural Networks (FFNN) consist of input, hidden, and output layers where information flows forward from input to output. Back Propagation is a supervised learning algorithm used to train FFNNs by minimizing the error between predicted and actual outputs. Back Propagation works by calculating the gradient of the loss function with respect to each weight through the chain rule (gradient descent). It updates the weights iteratively to reduce the loss, effectively allowing the network to learn the underlying mapping from inputs to outputs.

Code:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Generate dataset
X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

# Build FFNN model
model = Sequential()
model.add(Dense(10, input_shape=(2,), activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile model with optimizer + loss (needed for backprop)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train model (this triggers backpropagation internally)
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
                    validation_data=(X_test, y_test), verbose=0)

# Evaluate on test data
loss, acc = model.evaluate(X_test, y_test, verbose=0)
# Plot accuracy and loss curves
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
plt.grid(True)
plt.show()

print(f"\nFinal Test Accuracy: {acc:.4f}")
```

Result:

- The Feed Forward Neural Network trained successfully on the binary classification dataset.
- The accuracy and loss graphs showed progressive improvement and convergence over epochs.
- The trained model was able to classify the test data with good accuracy.