Ex. No.: 5

A PYTHON PROGRAM TO IMPLEMENT MULTI LAYER PERCEPTRON WITH BACK PROPOGATION

Aim:

To implement multilayer perceptron with back propagation using python.

Algorithm:

Step 1: Import the Necessary Libraries

● Import pandas as pd.

● Import numpy as np.

Step 2: Read and Display the Dataset

● Use `pd.read_csv("dataset.csv")` to read the dataset.

● Assign the result to a variable (e.g., `data`).

● Display the first ten rows using `data.head(10)`.

Step 3: Display Dataset Dimensions

● Use the `.shape` attribute on the dataset (e.g., `data.shape`).

Step 4: Display Descriptive Statistics

● Use the `.describe()` function on the dataset (e.g., `data.describe()`).

Step 5: Import Train-Test Split Module

● Import `train_test_split` from `sklearn.model_selection`.

Step 6: Split Dataset with 80-20 Ratio

● Assign the features to a variable (e.g., `X = data.drop(columns='target')`).

● Assign the target variable to another variable (e.g., `y = data['target']`).

● Use `train_test_split` to split the dataset into training and testing sets with a ratio of 0.2.

● Assign the results to `x_train`, `x_test`, `y_train`, and `y_test`.

Step 7: Import MLPClassifier Module

● Import `MLPClassifier` from `sklearn.neural_network`.

Step 8: Initialize MLPClassifier

● Create an instance of `MLPClassifier` with `max_iter=500` and

`activation='relu'`.

● Assign the instance to a variable (e.g., `clf`).

Step 9: Fit the Classifier

● Fit the model using `clf.fit(x_train, y_train)`.

Step 10: Make Predictions

● Use the `.predict()` function on `x_test` (e.g., `pred = clf.predict(x_test)`).

● Display the predictions.

Step 11: Import Metrics Modules

● Import `confusion_matrix` from `sklearn.metrics`.

● Import `classification_report` from `sklearn.metrics`.

Step 12: Display Confusion Matrix

● Use `confusion_matrix(y_test, pred)` to generate the confusion matrix.

● Display the confusion matrix.

Step 13: Display Classification Report

● Use `classification_report(y_test, pred)` to generate the classification report.

● Display the classification report.

Step 14: Repeat Steps 9-13 with Different Activation Functions

● Initialize `MLPClassifier` with `activation='logistic'`.

● Fit the model and make predictions.

● Display the confusion matrix and classification report.

● Repeat for `activation='tanh'`.

● Repeat for `activation='identity'`.

Step 15: Repeat Steps 7-14 with 70-30 Ratio

● Use `train_test_split` to split the dataset into training and testing sets with a ratio

of 0.3.

- Assign the results to `x_train`, `x_test`, `y_train`, and `y_test`.
- Repeat Steps 7-14 with the new training and testing sets.

PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Load the uploaded CSV file
bnotes = pd.read_csv('BankNote_Authentication.csv')
print("First 10 rows of dataset:")
print(bnotes.head(10))

# Separate features and target
x = bnotes.drop('class', axis=1)
y = bnotes['class']

print("\nFeature sample:")
print(x.head(2))
print("\nTarget sample:")
print(y.head(2))

# Function to train and evaluate MLP with a given activation and test size
def evaluate_mlp(activation_fn, test_size):
    print(f"\n\n--- Activation: {activation_fn}, Test size: {test_size} --
-")
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=test_size, random_state=42)
    mlp = MLPClassifier(max_iter=500, activation=activation_fn,
random_state=42)
    mlp.fit(x_train, y_train)
    pred = mlp.predict(x_test)

    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, pred))
    print("\nClassification Report:")
    print(classification_report(y_test, pred))

# Test size = 0.2
for activation in ['relu', 'logistic', 'tanh', 'identity']:
```

```
    evaluate_mlp(activation, test_size=0.2)

# Test size = 0.3
for activation in ['relu', 'logistic', 'tanh', 'identity']:
    evaluate_mlp(activation, test_size=0.3)
```

OUTPUT :

```
First 10 rows of dataset:
   variance  skewness  curtosis  entropy  class
0   3.62160    8.6661  -2.80730 -0.44699      0
1   4.54590    8.1674  -2.45860 -1.46210      0
2   3.86600   -2.6383   1.92420  0.10645      0
3   3.45660    9.5228  -4.01120 -3.59440      0
4   0.32924   -4.4552   4.57180 -0.98880      0
5   4.36840    9.6718  -3.96060 -3.16250      0
6   3.59120    3.0129   0.72888  0.56421      0
7   2.09220   -6.8100   8.46360 -0.60216      0
8   3.20320    5.7588  -0.75345 -0.61251      0
9   1.53560    9.1772  -2.27180 -0.73535      0

Feature sample:
   variance  skewness  curtosis  entropy
0    3.6216    8.6661   -2.8073 -0.44699
1    4.5459    8.1674   -2.4586 -1.46210

Target sample:
0    0
1    0
Name: class, dtype: int64


--- Activation: relu, Test size: 0.2 ---

Confusion Matrix:
[[148   0]
 [  0 127]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       148
           1       1.00      1.00      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275
```

```
--- Activation: logistic, Test size: 0.2 ---

Confusion Matrix:
[[148   0]
 [  1 126]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       148
           1       1.00      0.99      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275



--- Activation: tanh, Test size: 0.2 ---

Confusion Matrix:
[[148   0]
 [  0 127]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       148
           1       1.00      1.00      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275



--- Activation: identity, Test size: 0.2 ---

Confusion Matrix:
[[146   2]
 [  2 125]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       148
           1       0.98      0.98      0.98       127

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275



--- Activation: relu, Test size: 0.3 ---

Confusion Matrix:
[[229   0]
 [  0 183]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      1.00      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412
```

```
--- Activation: identity, Test size: 0.2 ---

Confusion Matrix:
[[146   2]
 [  2 125]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       148
           1       0.98      0.98      0.98       127

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275



--- Activation: relu, Test size: 0.3 ---

Confusion Matrix:
[[229   0]
 [  0 183]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      1.00      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412



--- Activation: logistic, Test size: 0.3 ---

Confusion Matrix:
[[229   0]
 [  1 182]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      0.99      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412



--- Activation: tanh, Test size: 0.3 ---

Confusion Matrix:
[[229   0]
 [  0 183]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      1.00      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412
```

```
--- Activation: identity, Test size: 0.3 ---

Confusion Matrix:
[[226   3]
 [  2 181]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       229
           1       0.98      0.99      0.99       183

    accuracy                           0.99       412
   macro avg       0.99      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

RESULT:

Thus, the Python program to implement a multi-layer perceptron with back propagation on the given dataset(data set.csv) has been executed successfully, and its results have been analyzed successfully for different activation functions (relu, logistic, tanh, identity) with two different training-testing ratios ( )