Ex No: 5

A Python Program to implement multilayer Perceptron with Back Propagation

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("banknote.csv") to read the dataset
* Assign the result to a variable (eg. data)
* Display the first ten rows using data.head(10)
Step 3: Display the Data Sets Dimensions
* Use the shape attribute on the dataset (eg. data.shape)
Step 4: Import Train-Test Split Module
* Import train_test_split from 'sklearn.model_selection'
Step 5: Split Dataset with 80-20 Ratio.
* Assign the features to a variable (eg. x = data.drop(columns = 'target'))
* Assign the target variable to another variable (eg. 'y = data['target']')
* Use 'train_test_split' to split the dataset

	Image.Var	Image.Skew	Image.Curt	Entropy	Class
0	3.62160	8.6661	-2.8073	-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

* Use train_test_split to split the dataset into training and testing sets with a ratio of 0.2.
* Assign the result to 'x_train', 'x_test', 'y_train', and 'y_test'

Step 7: Import MLPClassifier Module
* Import 'MLP Classifier from sklearn.neural_network'.

Step 8: Initialize MLP Classifier
* Create an instance of 'MLP Classifier' with activation = 'relu'.
* Assign the instance to a variable (eg. 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'
(eg, pred = clf.predict(x_test))
* Display the prediction

Step 10: Import Metrices Modules
* Import 'confusion_matrix' from sklearn.metrics'
* Import 'classification_report' from sklearn.metrics'.

Step 11: 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 step 9-13 with Different Activation Function
* Initialize MLP Classifier with activation = "logistic"
* Fill the model and make prediction
* Display the confusion matrix and classification report
* Repeat for activation = 'tanh'
* Repeat for activation = 'identify'.

Step 15: Reapeat steps 7-14 with 70-30 Ratio.
* Use 'trainSplit' to split the dataset into training and testing sets with a ratio of 0.3
* Assign the results to 'X_train', '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.neutral_network import MLPClas
-ifier
from sklearn.metrics import classification
report, confusion_matrix
# Load dataset
bnotes = pd.read_csv('./input/banknotes-dataset/
bank_note_data.csv')
print(bnotes.head(10)).
x = bnotes.drop("Class", axis=1)
y = bnotes['Class']
print(x.head(2))
print(y.head(2))
x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size = 0.2)
# activation function: relu
mlp = MLPClassifier(max_iter=500, activation
='relu')
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)
print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
# activation function : logistic
mlp = MLPClassifier(max_iter=500,
activation='logistic')
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)
print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
# activation function : tanh
mlp = MLPClassifier(max_iter=500, activation
='identity')
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)
print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))

# ---- train_test ratio = 0.3 ----
x_train, x_test, y_train, y_test =
  split(x, y, test_size=0.3)
# Activation function : relu
mlp = MLPClassifier(max_iter=500,
activation='logistic')
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)
print=(mlp.predict(x_test)
print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
# activation function: identity
mlp = MLP Classifier(max_iter = 500,
 activation = 'identity')
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)
print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

Result:-

Thus the python program to implement multi-layer perceptron with back propogation on the given dataset (bankNotes.csv) has been executed successfully and its results have been analysed successfully for different activation functions (relu, logistic, tanh, identity) with two different training-testing ratio(0.2 and 0.3)