Exp 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.

Step 2: Read and display the dataset.

* Use 'pd.read_csv("banknotes.csv") to read the dataset.
* display the first ten rows using 'data.head()'.

Step 3: Display descriptive dimentions.

Step 4: Display descriptive statistics.

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 variable.
* Assign the target variable to another variable.

Step 7: Import MLP classification module

* Import MLP classifier from 'sklearn.neural network'.

Step 8: Initialize MLP classifier.

* Assign the instance to a variable.

Step 9: Fit the classifier.

* Fit the model using clf.fit(x.train, y.train)

Step 10: Make predictions.

* Display the predictions.

Step 11: Import metrics modules.

Step 12: Display confusion matrix

Step 13: Display classification metric

* Display classification report

Step 14: Repeat ~~steps~~ steps 9-13 with different activation functions.

* Repeat for activation = 'tanh'
* Repeat for activation - 'identity'.
* Repeat for with 70-30 ratio.

Step 15: Repeat steps 7-14 with

* Assign the results to 'x_train', 'x_test', 'y ~~test~~ train' and 'y_test'.
* Repeat steps 7-14 with new training and testing sets.

Program:

```
import pandas as pd
import numpy as np
bnotes = pd.read_csv("../input/banknotes_dataset/bank-note-
                                   data.csv")

bnotes.head(10)
bnotes.describe(include='all')
x= bnotes.drop('class', axis=1)
 y = bnotes ['class']
Print (x.head (2))
Print (y.head (2))
from sklearn.model_selection import train_test_split
x_train, x_test , y_train, y_test = train_test_split(x, y, test
                   _size = 0.2) from sklearn.neural_network;

import MLPclassifier
mlp = MLPclassifier(max_item=500, activation='relu')

mlp.fit(x_train, y_train)
Pred= mlp.predict (x_test)
Print(pred)
from sklearn.metrics import classification_report,
                             confusion_matrix.
```

Con fusion _matrix(y_test, pred))
Print (classification_report (y_test, pred))
mlp. fit (x_train, y_train)
Pred = mlp. predict (x_test)
Print (pred)
from Sk learn. metrices import classification_report,
confusion_matrix

Confusion - matrix (y_test, pred)
Print(classification_report (y - test, pred))
~~mlp = MLP.~~
mlp. fit (x_train, y_train)
Pred = mlp. Predict (x_test)
from sklearn. metrics import classification_report
Confusion_matrix

Confusion_matrix(y_test, pred))
Print (classification_report (y_test, pred))
mlp. fit(x_train, y_train)
MLPclassifier (activation = 'identity', max_iter =500)
Pred = mlp. predict(x_test)
Print (classification_report (y_test, pred))
x_train, x_test, y_train, y_test =
train_test_split (x, y, test_size= 0.3) from sklearn neural network.

import MLPclassifier
mlp = MLPclassifier(max_item =500, activation = 'relu')
mlp. fit(x_train, y_train)
MLPclassifier (max_item =500)
Pred = mlp. Predict (x_test)
Print (~~classif~~ Pred).

```
MLPClassifier(max_iter=500, activation='tanh')
from sklearn.metrics import classification_report,
~~class~~                     confusion_matrix

confusion_matrix(y_test, pred)
mlp.fit(x_train, y_train)
pred=mlp.predict(x_test)
print(pred)
from sklearn.metrics import classification_report,
                                 confusion_matrix

confusion_matrix(y_test, pred).
print(classification_report(y_test, pred))
mlp=MLPClassifier(max_iter=500, activation='identity')
mlp.fit(x_train, y_train)
MLPClassifier(max_iter=500, activation='identity')
pred=mlp.predict(x_test)
print(pred)
from sklearn.metrics import classification_report,
                                 confusion_matrix

confusion_matrix(y_test; pred)
print(classification_report(y_test, pred))
```