

EXPERIMENT 5

ADITHIYA.V

241501010

A PYTHON PROGRAM TO IMPLEMENT MULILAYER PERCEPTION WITH BACK PROPAGATION

AIM:

*TO IMPLEMENT A PYTHON PROGRAM WITH MULILAYER PERCEPTION
WITH BACK PROPAGATION*

CODE:

```
import pandas as pd
import numpy as np
bnotes = pd.read_csv('/content/BankNote_Authentication.csv')
bnotes.head(10)

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
#train_test ratio = 0.2
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
from sklearn.neural_network import MLPClassifier
# activation function : relu
mlp = MLPClassifier(max_iter=500,activation='relu')
mlp.fit(x_train,y_train)
MLPClassifier(max_iter=500)
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))
```

```
# activation function : logistic
mlp = MLPClassifier(max_iter=500,activation='logistic')
mlp.fit(x_train,y_train)
```

```
MLPClassifier(activation='logistic', max_iter=500)
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='tanh')
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))
```

```
# activation function : identity
mlp = MLPClassifier(max_iter=500,activation='identity')
mlp.fit(x_train,y_train)
MLPClassifier(activation='identity', max_iter=500)
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))

#train_test ratio = 0.3
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
from sklearn.neural_network import MLPClassifier
# activation function : relu
mlp = MLPClassifier(max_iter=500,activation='relu')
mlp.fit(x_train,y_train)
MLPClassifier(max_iter=500)
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))

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

# activation function : tanh
mlp = MLPClassifier(max_iter=500,activation='tanh')
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))
```

```

# activation function : identity
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))

```

OUTPUT:

```

IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: C:/Users/itzdi/OneDrive/Documents/ML_Codes/Exp_5.py
    variance  skewness  kurtosis  entropy
0      3.6216     8.6661   -2.8073  -0.44699
1      4.5459     8.1674   -2.4586  -1.46210
0      0
1      0
Name: class, dtype: int64
[0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 0 1 0 1 0
 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0
 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0
 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 1 1 1
 0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0
 0 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 1
 1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0
 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0]

      precision    recall   f1-score   support
0         1.00     1.00     1.00      142
1         1.00     1.00     1.00      133

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

```

[0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0
1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0
0 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 1 0 0
0 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0
1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0
1 0 0 0 0 1 1 1 0 0 1 1 0 1 0]
precision recall f1-score support
0 1.00 1.00 1.00 142
1 1.00 1.00 1.00 133
accuracy 0.99
macro avg 1.00 1.00 1.00 275
weighted avg 1.00 1.00 1.00 275
[0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0
1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0
0 1 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 0
0 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 1 0 0
1 0 1 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 1 1 0 0 1 0
1 0 0 0 0 1 1 1 0 0 1 1 0 1 0]
precision recall f1-score support
0 1.00 0.99 0.99 142
1 0.99 1.00 0.99 133
accuracy 0.99
macro avg 0.99 0.99 0.99 275
weighted avg 0.99 0.99 0.99 275
[0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0
1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 1 1
1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 0
0 1 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0
1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0
1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0]
precision recall f1-score support
0 1.00 0.99 0.99 142
1 0.99 1.00 0.99 133
accuracy 0.99
macro avg 0.99 0.99 0.99 275
weighted avg 0.99 0.99 0.99 275
[0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0
1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0
0 1 1 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0
1 0 1 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0
1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0]
precision recall f1-score support
0 1.00 1.00 1.00 142
1 1.00 1.00 1.00 133
accuracy 1.00
macro avg 1.00 1.00 1.00 275
weighted avg 1.00 1.00 1.00 275

	precision	recall	f1-score	support
0	1.00	0.99	0.99	142
1	0.99	1.00	0.99	133

	precision	recall	f1-score	support
accuracy			0.99	275
macro avg	0.99	0.99	0.99	275
weighted avg	0.99	0.99	0.99	275

	precision	recall	f1-score	support
0	1.00	1.00	1.00	221
1	1.00	1.00	1.00	191

	precision	recall	f1-score	support
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	412

	precision	recall	f1-score	support
0	1.00	1.00	1.00	221
1	1.00	1.00	1.00	191

	precision	recall	f1-score	support
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	412

	precision	recall	f1-score	support
0	1.00	1.00	1.00	221
1	1.00	0.99	1.00	191

	precision	recall	f1-score	support
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	412

```
[0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0
1 0 0 1 1 0 0 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0
1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0
0 0 0 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 0
0 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0
0 1 0 1 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1
1 0 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1
0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1
0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1
1 1 1 0 0]
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	221
1	0.98	0.98	0.98	191
accuracy			0.99	412
macro avg	0.99	0.99	0.99	412
weighted avg	0.99	0.99	0.99	412

RESULT:

A PYTHON PROGRAM TO IMPLEMENT MULILAYER PERCEPTION WITH BACK PROPAGATION AS BEEN ANALYSED AND VERIFIED