EXP 5:

A PYTHON PROGRAM TO IMPLEMENT MULTI LAYER PERCEPTRON WITH BACK PROPAGATION

Aim:
To implement multilayer perceptron with back propagation using python.

PROGRAM:

```
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:

	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

```
   variance  skewness  curtosis  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 0 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1
 0 1 0 1 1 1 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0
 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1
 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 1 1
 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 0
 0 0 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1
 1 1 1 0 0 1 1 1 0 1 0 1 0 0 0 0]
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       146
           1       1.00      1.00      1.00       129

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

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       146
           1       0.99      1.00      1.00       129

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       146
           1       1.00      1.00      1.00       129

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

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       146
           1       0.96      1.00      0.98       129

    accuracy                           0.98       275
   macro avg       0.98      0.98      0.98       275
weighted avg       0.98      0.98      0.98       275
```

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       227
           1       1.00      1.00      1.00       185

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

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

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

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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	227
1	1.00	1.00	1.00	185
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	412

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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	227
1	0.99	0.99	0.99	185
accuracy			1.00	412
macro avg	1.00	1.00	1.00	412
weighted avg	1.00	1.00	1.00	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 ( )