PRACTICAL NO.:01

Name: Imam Mahamad Shaikh

Batch:C2

Roll no.:B23025

Problem Statement:

Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months. Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc. Link to the Kaggle project: https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling
Perform following steps:

1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and test sets.
3. Normalize the train and test data.

4. Initialize and build the model. Identify the points of improvement and implement the same.

5. Print the accuracy score and confusion matrix (5 points).

```
import pandas as pd
import numpy as np

df = pd.read_csv("Churn_Modelling.csv")

df
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Gender
Age  \
0             1    15634602   Hargrave          619    France  Female
42
1             2    15647311       Hill          608     Spain  Female
41
2             3    15619304       Onio          502    France  Female
```

```
42
3             4    15701354       Boni           699    France  Female
39
4             5    15737888   Mitchell           850     Spain  Female
43
...         ...         ...        ...           ...       ...     ...
...
9995       9996    15606229   Obijiaku           771    France    Male
39
9996       9997    15569892  Johnstone           516    France    Male
35
9997       9998    15584532        Liu           709    France  Female
36
9998       9999    15682355  Sabbatini           772   Germany    Male
42
9999      10000    15628319     Walker           792    France  Female
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1
1          1   83807.86              1          0               1
2          8  159660.80              3          1               0
3          1       0.00              2          0               0
4          2  125510.82              1          1               1
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]

df.head()

   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age
\
```

```
0          1    15634602  Hargrave          619    France  Female   42

1          2    15647311      Hill          608     Spain  Female   41

2          3    15619304      Onio          502    France  Female   42

3          4    15701354      Boni          699    France  Female   39

4          5    15737888  Mitchell          850     Spain  Female   43


   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0

df.tail()

      RowNumber  CustomerId    Surname  CreditScore Geography  Gender
Age  \
9995       9996    15606229   Obijiaku          771    France    Male
39
9996       9997    15569892  Johnstone          516    France    Male
35
9997       9998    15584532        Liu          709    France  Female
36
9998       9999    15682355  Sabbatini          772   Germany    Male
42
9999      10000    15628319     Walker          792    France  Female
28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
```

```
9998         92888.52       1
9999         38190.78       0
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
df.describe()
```

```
         RowNumber    CustomerId   CreditScore           Age
Tenure  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000
10000.000000
mean    5000.50000  1.569094e+07    650.528800     38.921800
5.012800
std     2886.89568  7.193619e+04     96.653299     10.487806
2.892174
min        1.00000  1.556570e+07    350.000000     18.000000
0.000000
25%     2500.75000  1.562853e+07    584.000000     32.000000
3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000
5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000
7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000
10.000000

             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
```

```
mean    76485.889288       1.530200      0.70550         0.515100
std     62397.405202       0.581654      0.45584         0.499797
min         0.000000       1.000000      0.00000         0.000000
25%         0.000000       1.000000      0.00000         0.000000
50%     97198.540000       1.000000      1.00000         1.000000
75%    127644.240000       2.000000      1.00000         1.000000
max    250898.090000       4.000000      1.00000         1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

```
df.dtypes
```

```
RowNumber             int64
CustomerId            int64
Surname              object
CreditScore           int64
Geography            object
Gender               object
Age                   int64
Tenure                int64
Balance             float64
NumOfProducts         int64
HasCrCard             int64
IsActiveMember        int64
EstimatedSalary     float64
Exited                int64
dtype: object
```

```
df.isnull()
```

```
      RowNumber  CustomerId  Surname  CreditScore  Geography  Gender
Age  \
0         False       False    False        False      False   False
False
1         False       False    False        False      False   False
False
2         False       False    False        False      False   False
False
3         False       False    False        False      False   False
False
4         False       False    False        False      False   False
False
```

```
...          ...          ...      ...           ...         ...     ...
...
9995      False       False    False         False      False   False
False
9996      False       False    False         False      False   False
False
9997      False       False    False         False      False   False
False
9998      False       False    False         False      False   False
False
9999      False       False    False         False      False   False
False

      Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0      False    False          False      False           False
1      False    False          False      False           False
2      False    False          False      False           False
3      False    False          False      False           False
4      False    False          False      False           False
...      ...      ...            ...        ...             ...
9995   False    False          False      False           False
9996   False    False          False      False           False
9997   False    False          False      False           False
9998   False    False          False      False           False
9999   False    False          False      False           False

      EstimatedSalary  Exited
0               False   False
1               False   False
2               False   False
3               False   False
4               False   False
...               ...     ...
9995            False   False
9996            False   False
9997            False   False
9998            False   False
9999            False   False

[10000 rows x 14 columns]
```

```
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
```

```
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
x = df.drop(columns=["RowNumber","CustomerId","Surname","Exited"])
y = df["Exited"]

x
```

```
      CreditScore Geography  Gender  Age  Tenure    Balance
NumOfProducts  \
0             619    France  Female   42       2       0.00
1
1             608     Spain  Female   41       1   83807.86
1
2             502    France  Female   42       8  159660.80
3
3             699    France  Female   39       1       0.00
2
4             850     Spain  Female   43       2  125510.82
1
...           ...       ...     ...  ...     ...        ...
...
9995          771    France    Male   39       5       0.00
2
9996          516    France    Male   35      10   57369.61
1
9997          709    France  Female   36       7       0.00
1
9998          772   Germany    Male   42       3   75075.31
2
9999          792    France  Female   28       4  130142.79
1

      HasCrCard  IsActiveMember  EstimatedSalary
0             1               1        101348.88
1             0               1        112542.58
2             1               0        113931.57
3             0               0         93826.63
4             1               1         79084.10
...         ...             ...              ...
9995          1               0         96270.64
9996          1               1        101699.77
9997          0               1         42085.58
9998          1               0         92888.52
```

```
9999          1               0         38190.78

[10000 rows x 10 columns]
```

```
y
```

```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler

X = pd.get_dummies(x, drop_first=True)

X=X.values

X
```

```
array([[619, 42, 2, ..., False, False, False],
       [608, 41, 1, ..., False, True, False],
       [502, 42, 8, ..., False, False, False],
       ...,
       [709, 36, 7, ..., False, False, False],
       [772, 42, 3, ..., True, False, True],
       [792, 28, 4, ..., False, False, False]], dtype=object)
```

```
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, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(32, 16), max_iter=200,
random_state=42)
mlp.fit(X_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic
```

```
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier(hidden_layer_sizes=(32, 16), random_state=42)
```

```
y_pred = mlp.predict(X_test)
```

```
y_pred
```

```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.861
Confusion Matrix:
 [[1519   88]
 [ 190  203]]
```