```
import pandas as pd

df=pd.read_csv("/home/student/Downloads/Churn_Modelling.csv")

df.head
```

```
<bound method NDFrame.head of       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.describe

<bound method NDFrame.describe of       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]>
```

```
#  missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 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
```

```
Gender_Data=pd.get_dummies(df["Gender"],drop_first=True)
Gender_Data
```

```
       Male
0     False
1     False
2     False
3     False
4     False
...     ...
9995   True
9996   True
9997  False
9998   True
9999  False
```

```
[10000 rows x 1 columns]
```

```
Geo_Data=pd.get_dummies(df["Geography"],drop_first=True)
Geo_Data
```

```
      Germany  Spain
0       False  False
1       False   True
2       False  False
3       False  False
4       False   True
...       ...    ...
9995    False  False
9996    False  False
9997    False  False
9998     True  False
9999    False  False
```

```
[10000 rows x 2 columns]
```

```
df=df.drop("Gender",axis=1)
```

```
df=df.drop("Geography",axis=1)
```

```
df=df.drop("Surname",axis=1)
```

```
df=df.drop("RowNumber",axis=1)
```

```
df=df.drop("CustomerId",axis=1)
```

```
df
```

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

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

```
...               ...              ...      ...
9995                0         96270.64        0
9996                1        101699.77        0
9997                1         42085.58        1
9998                0         92888.52        1
9999                0         38190.78        0

[10000 rows x 9 columns]
```

```
df=pd.concat([df,Gender_Data,Geo_Data],axis=1)
```

```
df
```

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

      IsActiveMember  EstimatedSalary  Exited   Male  Germany  Spain
0                  1        101348.88       1  False    False  False
1                  1        112542.58       0  False    False   True
2                  0        113931.57       1  False    False  False
3                  0         93826.63       0  False    False  False
4                  1         79084.10       0  False    False   True
...              ...              ...     ...    ...      ...    ...
9995               0         96270.64       0   True    False  False
9996               1        101699.77       0   True    False  False
9997               1         42085.58       1  False    False  False
9998               0         92888.52       1   True     True  False
9999               0         38190.78       0  False    False  False

[10000 rows x 12 columns]
```

```
X = df.drop('Exited', axis=1)
y = df['Exited']
```

```
X
```

```
      CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0             619   42       2       0.00              1          1
1             608   41       1   83807.86              1          0
2             502   42       8  159660.80              3          1
3             699   39       1       0.00              2          0
```

```
4             850   43       2  125510.82              1          1
...           ...  ...     ...        ...            ...        ...
9995          771   39       5       0.00              2          1
9996          516   35      10   57369.61              1          1
9997          709   36       7       0.00              1          0
9998          772   42       3   75075.31              2          1
9999          792   28       4  130142.79              1          1

      IsActiveMember  EstimatedSalary   Male  Germany  Spain
0                  1        101348.88  False    False  False
1                  1        112542.58  False    False   True
2                  0        113931.57  False    False  False
3                  0         93826.63  False    False  False
4                  1         79084.10  False    False   True
...              ...              ...    ...      ...    ...
9995               0         96270.64   True    False  False
9996               1        101699.77   True    False  False
9997               1         42085.58  False    False  False
9998               0         92888.52   True     True  False
9999               0         38190.78  False    False  False

[10000 rows x 11 columns]
```

```
import numpy as np
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.preprocessing import StandardScaler
scaler = StandardScaler()

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

import keras
from keras.models import Sequential
from keras.layers import Dense

classifier=Sequential()

classifier.add(Dense(activation="relu",input_dim=11,units=6,kernel_ini
tializer="uniform"))
classifier.add(Dense(activation="relu",units=6,kernel_initializer="uni
form"))
classifier.add(Dense(activation="sigmoid",units=1,kernel_initializer="
uniform"))
```

```
/home/student/.local/lib/python3.10/site-packages/keras/src/layers/
core/dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
```

```
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
classifier.compile(optimizer="adam",loss="binary_crossentropy",metrics
=["accuracy"])
```

```
classifier.summary()
```

```
Model: "sequential_6"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━
━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━
━┩
│ dense_21 (Dense)                │ (None, 6)              │
72 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ dense_22 (Dense)                │ (None, 6)              │
42 │
├─────────────────────────────────┼────────────────────────┼──────────────
─┤
│ dense_23 (Dense)                │ (None, 1)              │
7 │
└─────────────────────────────────┴────────────────────────┴──────────────
─┘
```

```
 Total params: 121 (484.00 B)

 Trainable params: 121 (484.00 B)

 Non-trainable params: 0 (0.00 B)
```

```
classifier.fit(X_train,y_train,batch_size=10,epochs=50)
```

```
Epoch 1/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 886us/step - accuracy: 0.7898 - loss:
0.5811
Epoch 2/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 983us/step - accuracy: 0.8253 - loss:
0.4034
Epoch 3/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 972us/step - accuracy: 0.8340 - loss:
0.3885
Epoch 4/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8239 - loss:
0.3998
```

```
Epoch 5/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8368 - loss:
0.3728
Epoch 6/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 980us/step - accuracy: 0.8314 - loss:
0.3871
Epoch 7/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8396 - loss:
0.3823
Epoch 8/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 932us/step - accuracy: 0.8449 - loss:
0.3671
Epoch 9/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 937us/step - accuracy: 0.8512 - loss:
0.3687
Epoch 10/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 879us/step - accuracy: 0.8524 - loss:
0.3599
Epoch 11/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 955us/step - accuracy: 0.8543 - loss:
0.3598
Epoch 12/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 936us/step - accuracy: 0.8549 - loss:
0.3613
Epoch 13/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 922us/step - accuracy: 0.8417 - loss:
0.3674
Epoch 14/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 943us/step - accuracy: 0.8549 - loss:
0.3582
Epoch 15/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 968us/step - accuracy: 0.8597 - loss:
0.3492
Epoch 16/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 954us/step - accuracy: 0.8521 - loss:
0.3578
Epoch 17/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 880us/step - accuracy: 0.8494 - loss:
0.3608
Epoch 18/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 878us/step - accuracy: 0.8575 - loss:
0.3510
Epoch 19/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 924us/step - accuracy: 0.8505 - loss:
0.3592
Epoch 20/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 970us/step - accuracy: 0.8493 - loss:
0.3539
Epoch 21/50
```

```
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8506 - loss:
0.3577
Epoch 22/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8534 - loss:
0.3582
Epoch 23/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8585 - loss:
0.3531
Epoch 24/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8594 - loss:
0.3439
Epoch 25/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 997us/step - accuracy: 0.8601 - loss:
0.3559
Epoch 26/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 825us/step - accuracy: 0.8448 - loss:
0.3578
Epoch 27/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8516 - loss:
0.3574
Epoch 28/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8598 - loss:
0.3486
Epoch 29/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 846us/step - accuracy: 0.8495 - loss:
0.3583
Epoch 30/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 825us/step - accuracy: 0.8609 - loss:
0.3467
Epoch 31/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 993us/step - accuracy: 0.8566 - loss:
0.3512
Epoch 32/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8536 - loss:
0.3526
Epoch 33/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 796us/step - accuracy: 0.8552 - loss:
0.3518
Epoch 34/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 950us/step - accuracy: 0.8573 - loss:
0.3569
Epoch 35/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8575 - loss:
0.3483
Epoch 36/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8606 - loss:
0.3428
Epoch 37/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8603 - loss:
```

```
0.3470
Epoch 38/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8530 - loss:
0.3496
Epoch 39/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8617 - loss:
0.3377
Epoch 40/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 952us/step - accuracy: 0.8510 - loss:
0.3519
Epoch 41/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8589 - loss:
0.3529
Epoch 42/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 964us/step - accuracy: 0.8627 - loss:
0.3374
Epoch 43/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 805us/step - accuracy: 0.8609 - loss:
0.3412
Epoch 44/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 879us/step - accuracy: 0.8638 - loss:
0.3384
Epoch 45/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 785us/step - accuracy: 0.8532 - loss:
0.3547
Epoch 46/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 737us/step - accuracy: 0.8587 - loss:
0.3396
Epoch 47/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 782us/step - accuracy: 0.8521 - loss:
0.3490
Epoch 48/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 789us/step - accuracy: 0.8605 - loss:
0.3392
Epoch 49/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 731us/step - accuracy: 0.8648 - loss:
0.3350
Epoch 50/50
800/800 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8560 - loss:
0.3439

<keras.src.callbacks.history.History at 0x7b4ffc18f3a0>

F_pred=classifier.predict(X_test)
F_pred

63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 986us/step

array([[0.0701367 ],
       [0.13290085],
```

```
       [0.0171692 ],
       ...,
       [0.02924804],
       [0.03465875],
       [0.10427628]], dtype=float32)
```

```
y_pred=(F_pred>0.5)
y_pred
```

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

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

accuracy=accuracy_score(y_test, y_pred)

accuracy
```

```
0.879
```

```
con_matrix=
```