```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


class Bank:

    __name = "Churn_Modelling.csv"

    def load_dataset(self):
        self.dataset = pd.read_csv(self.__name)

    def show_dataset_head(self):
        display(self.dataset.head())
        print("Length of dataset: ", len(self.dataset))

    def eda(self):
        # print(self.dataset.info())
        # print("Unique values in Surname: ", self.dataset.Surname.unique())
        # print("Unique values in Geography: ", self.dataset.Geography.unique())
        # print("Unique values in Age: ", self.dataset.Age.unique())
        # print("Unique values in No. of Products: ", self.dataset.NumOfProducts.unique())
        print("Number of Exited and Non-Exited people: ", self.dataset.Exited.value_counts())

    def preprocess(self):
        self.X = self.dataset[[feature for feature in self.dataset.columns
                               if ((feature!="RowNumber") & (feature!="CustomerId") &
(feature!="Exited"))]]
        self.y = self.dataset["Exited"]

        self.le = LabelEncoder()
        self.X.Geography = self.le.fit_transform(self.X["Geography"])
        self.X.Surname = self.le.fit_transform(self.X["Surname"])
        self.X.Gender = self.le.fit_transform(self.X["Gender"])

        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(self.X,
self.y, random_state=100, test_size=0.2)

        self.MinMaxScaler = MinMaxScaler()
        self.X_train_transformed = self.MinMaxScaler.fit_transform(self.X_train)
        self.X_test_transformed = self.MinMaxScaler.transform(self.X_test)

        self.df_train = pd.DataFrame({"Index": self.y_train.keys(), "Exited":
self.y_train.values})
        self.df_test = pd.DataFrame({"Index": self.y_test.keys(), "Exited":
self.y_test.values})
        # print("Number of Exited and Non-Exited people in training: ",
self.df_train.Exited.value_counts())
        # print("Number of Exited and Non-Exited people in testing: ",
self.df_test.Exited.value_counts())


    def train(self):
```

```
        self.model = Sequential()
        self.model.add(Dense(60, input_shape=(11,), activation="relu"))
        self.model.add(Dropout(0.5))
        self.model.add(Dense(30, activation="relu"))
        self.model.add(Dropout(0.5))
        self.model.add(Dense(15, activation="relu"))
        self.model.add(Dropout(0.5))
        self.model.add(Dense(1, activation="sigmoid"))

        self.model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])

        self.model.fit(self.X_train_transformed, self.y_train, epochs=100)

    def predict(self):
        self.y_pred = self.model.predict(self.X_test)
        for i in range(len(self.y_pred)):
            if (self.y_pred[i] > 0.5):
                self.y_pred[i] = 1
            else:
                self.y_pred[i] = 0

    def evaluate(self):
        self.cm = confusion_matrix(self.y_test, self.y_pred)
        print(self.cm)
        print(classification_report(self.y_test, self.y_pred))
        _, accuracy = self.model.evaluate(self.X_test_transformed, self.y_test)
        print("Accuracy: %.2f"%(accuracy*100))



customer = Bank()
customer.load_dataset()
customer.show_dataset_head()
customer.eda()
customer.preprocess()
customer.train()
customer.predict()
customer.evaluate()
```

# Output:

| wNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Exited |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | 0.00 | 1 | 1 | 1 | 101348.88 | 1 |
| 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83807.86 | 1 | 0 | 1 | 112542.58 | 0 |
| 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159660.80 | 3 | 1 | 0 | 113931.57 | 1 |
| 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | 0.00 | 2 | 0 | 0 | 93826.63 | 0 |
| 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125510.82 | 1 | 1 | 1 | 79084.10 | 0 |

```
Length of dataset:  10000
Number of Exited and Non-Exited people:  0    7963
1    2037
```

Training the Model:
250/250 [==============================] - 1s 2ms/step - loss: 0.5540 - accuracy: 0.7789
Epoch 2/100
250/250 [==============================] - 1s 2ms/step - loss: 0.5087 - accuracy: 0.7970
Epoch 3/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4950 - accuracy: 0.7969
.
.
.
250/250 [==============================] - 1s 2ms/step - loss: 0.3752 - accuracy: 0.8432
Epoch 84/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3712 - accuracy: 0.8489
Epoch 85/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3708 - accuracy: 0.8464
Epoch 86/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3695 - accuracy: 0.8487
Epoch 87/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3716 - accuracy: 0.8465
Epoch 88/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3708 - accuracy: 0.8451
Epoch 89/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3641 - accuracy: 0.8478
Epoch 90/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3676 - accuracy: 0.8474
Epoch 91/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3691 - accuracy: 0.8471
Epoch 92/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3683 - accuracy: 0.8482
Epoch 93/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3698 - accuracy: 0.8469
Epoch 94/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3653 - accuracy: 0.8501
Epoch 95/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3681 - accuracy: 0.8474
Epoch 96/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3701 - accuracy: 0.8495
Epoch 97/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3666 - accuracy: 0.8459
Epoch 98/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3665 - accuracy: 0.8506
Epoch 99/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3626 - accuracy: 0.8503
Epoch 100/100
250/250 [==============================] - 0s 2ms/step - loss: 0.3724 - accuracy: 0.8410
63/63 [==============================] - 0s 1ms/step

Confusion Matrix:

[[ 15 1573]
 [ 2 410]]

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.88 | 0.01 | 0.02 | 1588 |
| 1 | 0.21 | 1.00 | 0.34 | 412 |
| accuracy | | | 0.21 | 2000 |
| macro avg | 0.54 | 0.50 | 0.18 | 2000 |
| weighted avg | 0.74 | 0.21 | 0.09 | 2000 |

Accuracy: 84.75