1. Impoting Dependencies :

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
```

2. Data Collcetion and Analysis:

A- Loading dataset :

In [ ]:
```
data_email = pd.read_csv("C:/Machine_learning Python/projets/spamEmail/mail_data
```

B- Head of the data :

In [ ]:
```
data_email.head()
```

Out[ ]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

C-Number of row & columns:

In [ ]:
```
data_email.shape
```

Out[ ]:
```
(5572, 2)
```

2. Statisctical measures :

A- Genral Statisc:

In [ ]:
```
data_email.describe()
```

Out[ ]:

	Category	Message
count	5572	5572
unique	2	5157
top	ham	Sorry, I'll call later
freq	4825	30

B- Information about the data:

In [ ]:
```
data_email.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

C- Number of missing value in each column;

In [ ]:
```
data_email.isnull().sum()
```

Out[ ]:
```
Category    0
Message     0
dtype: int64
```

3. Label Encoding:

A- Replace the null value with a nul string:

In [ ]:
```
data_email = data_email.where((pd.notnull(data_email)),'')
```

B- Replace the spam by 0 , and ham by 1:

In [ ]:
```
data_email.loc[   data_email["Category"] == 'spam' ,'Category', ] = 0
data_email.loc[   data_email["Category"] == 'ham' ,'Category', ] = 1
```

In [ ]:
```
data_email.head()
```

Out[ ]:

	Category	Message
0	1	Go until jurong point, crazy.. Available only ...
1	1	Ok lar... Joking wif u oni...
2	0	Free entry in 2 a wkly comp to win FA Cup fina...
3	1	U dun say so early hor... U c already then say...
4	1	Nah I don't think he goes to usf, he lives aro...

4. Train test split:

A- Separating a data & label

In [ ]:
```
X = data_email["Message"]
Y = data_email ["Category"]
```

In [ ]:
```
print(X.shape)
print(Y.shape)
```

```
(5572,)
(5572,)
```

B- Test Split:

In [ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X , Y , test_size=0.2, randc
```

In [ ]:
```
print(X.shape,X_train.shape, X_test.shape)
```

```
(5572,) (4457,) (1115,)
```

4. Feature extraction:

A- Tronsfrom the text data to features vectros:

In [ ]:
```
extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)
```

In [ ]:
```
X_train_features = extraction.fit_transform(X_train)
X_test_features = extraction.transform(X_test)
```

B- Change the tyepe of Y to int:

In [ ]:
```
Y_train = Y_train.astype(int)
Y_test = Y_test.astype(int)
```

5. Model Training:

A- Loading the model:

In [ ]:
```
model = LogisticRegression()
```

B- Training the model:

In [ ]:
```
model.fit(X_train_features , Y_train)
```

Out[ ]:
```
▾ LogisticRegression
LogisticRegression()
```

6. Evaluate the model:

A. Accuracy Score of training:

```
In [ ]: X_train_predicition = model.predict(X_train_features)
        training_data_accuracy = accuracy_score(X_train_predicition, Y_train)
        print('Accuracy score of the training data:', training_data_accuracy)
```

```
Accuracy score of the training data: 0.9670181736594121
```

B. Accuracy Score of testing:

```
In [ ]: X_test_predicition = model.predict(X_test_features)
        training_data_accuracy = accuracy_score(X_test_predicition, Y_test)
        print('Accuracy score of the training data:', training_data_accuracy)
```

```
Accuracy score of the training data: 0.9659192825112107
```

C. Exemple

```
In [ ]: def  prediction_email(Message):
            input_data = [Message]
            #Reshape the data:
            dataEncoding = extraction.transform(input_data)
            prediction = model.predict(dataEncoding)
            if prediction[0] == 1:
                print('The type of your email is a normal email (not spam)')
            else:
                print('The type of your email is a spam email')



        print("Welcome to our prediction email")
        Message = input("Enter your email: ")

        prediction_email(Message)
```

```
Welcome to our prediction email
The type of your email is a spam email
```