Spam/phishing mail detection

Dataset used:

mail_data.csv (5572 entries):

	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...

```
ham      4825
spam      747
Name: Category, dtype: int64
```

Firstly we preprocess the data, converting spam and ham(legit) to 0 and 1 and getting rid of any row that contains empty values.

In [7]:
```
data["Category"].replace(["ham", "spam"], [1, 0], inplace = True)
```

In [8]:
```
data['Category'].replace('', np.nan, inplace = True)
data['Message'].replace('', np.nan, inplace = True)

data.dropna(inplace = True)
```

In [9]:
```
data.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   int64 
 1   Message   5572 non-null   object
dtypes: int64(1), object(1)
memory usage: 87.2+ KB
```

Then, we save the message as X and category as y and we split the data into training and test sets.

```
In [14]: X = data["Message"]
         y = data["Category"]
```

```
In [15]: X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 20)
```

Because the message is a string, we will need to use TfidfVectorizer() from sklearn library to convert the text into a matrix of features before classifying.

```
In [11]: feature_extraction = TfidfVectorizer()

         X_train = feature_extraction.fit_transform(X_train_raw)
         X_test = feature_extraction.transform(X_test_raw)
```

After this, we can create the model and start training it. For the algorithm, I chose logistic regression because it's one of the most reliable methods of classification, especially when the number of classes is only 2.

```
In [12]: model = LogisticRegression()

         model.fit(X_train, y_train)
```

Here are the results:

```
In [14]: pred = model.predict(X_test)

         acc = accuracy_score(y_test, pred)
         prec = precision_score(y_test, pred)
         rec = recall_score(y_test, pred)
```

```
In [15]: [acc, prec, rec]
Out[15]: [0.967741935483871, 0.9653767820773931, 0.9978947368421053]
```

Finally, I included an example on how to test the model manually.

In [16]:
```
input_mail = ["Hello, you won a free phone! Contact us for more details."]

input_data = feature_extraction.transform(input_mail)

answer = model.predict(input_data)[0]

if answer == 0:
    print("Spam detected!")
else:
    print("Not spam.")
```

```
Spam detected!
```