In [3]:

```
import pandas as pd
df = pd.read_csv("C:\\Users\\P.Navya Sree\\OneDrive\\Documents\\ML\\titanic.csv")
df.head()
```

Out[3]:

	PassengerId	Name	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	1	Braund, Mr. Owen Harris	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	Heikkinen, Miss. Laina	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	female	35.0	1	0	113803	53.1000	C123	
4	5	Allen, Mr. William Henry	3	male	35.0	0	0	373450	8.0500	NaN	

In [5]:

```
df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis='co
df.head()
```

Out[5]:

	Pclass	Sex	Age	Fare	Survived
0	3	male	22.0	7.2500	0
1	1	female	38.0	71.2833	1
2	3	female	26.0	7.9250	1
3	1	female	35.0	53.1000	1
4	3	male	35.0	8.0500	0

In [6]:

```
target = df.Survived
inputs = df.drop('Survived',axis='columns')
```

In [10]:

```
dummies = pd.get_dummies(inputs.Sex,)
dummies = dummies.astype(int)
dummies.head(3)
```

Out[10]:

	female	male
0	0	1
1	1	0
2	1	0

In [11]:
```
inputs.drop('Sex',axis='columns',inplace=True)
inputs.head(3)
```

Out[11]:

	Pclass	Age	Fare
0	3	22.0	7.2500
1	1	38.0	71.2833
2	3	26.0	7.9250

In [13]:
```
inputs = pd.concat([inputs,dummies],axis='columns')
inputs.head(3)
```

Out[13]:

	Pclass	Age	Fare	female	male
0	3	22.0	7.2500	0	1
1	1	38.0	71.2833	1	0
2	3	26.0	7.9250	1	0

In [14]:
```
inputs.columns[inputs.isna().any()]
```

Out[14]:
```
Index(['Age'], dtype='object')
```

In [15]:
```
inputs.Age[:10]
```

Out[15]:
```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

In [16]:
```
inputs.Age = inputs.Age.fillna(inputs.Age.mean())
inputs.head(6)
```

Out[16]:

	Pclass	Age	Fare	female	male
0	3	22.000000	7.2500	0	1
1	1	38.000000	71.2833	1	0
2	3	26.000000	7.9250	1	0
3	1	35.000000	53.1000	1	0
4	3	35.000000	8.0500	0	1
5	3	29.699118	8.4583	0	1

In [21]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(inputs,target,test_size=0.4)
```

In [22]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

In [23]:
```
model.fit(X_train,y_train)
```

Out[23]:
```
▾ GaussianNB ⓘ ?
GaussianNB()
```

In [24]:
```
model.score(X_test,y_test)
```

Out[24]:
```
0.7787114845938375
```

In [25]:
```
X_test[:10]
```

Out[25]:

	Pclass	Age	Fare	female	male
91	3	20.000000	7.8542	0	1
856	1	45.000000	164.8667	1	0
57	3	28.500000	7.2292	0	1
303	2	29.699118	12.3500	1	0
239	2	33.000000	12.2750	0	1
644	3	0.750000	19.2583	1	0
497	3	29.699118	15.1000	0	1
293	3	24.000000	8.8500	1	0
555	1	62.000000	26.5500	0	1
774	2	54.000000	23.0000	1	0

In [27]:
```
y_test[:10]
```

Out[27]:
```
91     0
856    1
57     0
303    1
239    0
644    1
497    0
293    0
555    0
774    1
Name: Survived, dtype: int64
```

In [28]:
```
model.predict(X_test[:10])
```

Out[28]:
```
array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1], dtype=int64)
```

In [29]:
```
model.predict_proba(X_test[:10])
```

Out[29]:
```
array([[9.87402467e-01, 1.25975332e-02],
       [7.92054427e-07, 9.99999208e-01],
       [9.88883057e-01, 1.11169427e-02],
       [2.54087103e-02, 9.74591290e-01],
       [9.78541469e-01, 2.14585314e-02],
       [2.85999618e-02, 9.71400038e-01],
       [9.89353419e-01, 1.06465809e-02],
       [4.75681607e-02, 9.52431839e-01],
       [9.23709724e-01, 7.62902757e-02],
       [2.71965041e-02, 9.72803496e-01]])
```

In [30]:
```
#EMAIL SPAM DETECTION USING NAIVE BAYIES ALGORITHM !!
```

In [31]:
```
import pandas as pd
df = pd.read_csv("C:\\Users\\P.Navya Sree\\OneDrive\\Documents\\ML\\spam.csv")
df.head()
```

Out[31]:

	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...

In [32]:
```
df.groupby('Category').describe()
```

Out[32]:

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

In [33]:
```
df['spam']=df['Category'].apply(lambda x: 1 if x == 'spam' else 0)
df.head()
```

Out[33]:

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

In [35]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.Message,df.spam,test_size=0.2)
```

In [ ]:
```
"""
A CountVectorizer is a tool used in natural language processing (NLP) to convert a

How CountVectorizer Works:
```

```
Tokenization: It splits the text into individual words (tokens).
Vocabulary Building: It creates a vocabulary of all the unique words in the entire
Counting: For each document, it counts how many times each word from the vocabulary
Example:
Suppose you have the following two sentences:

"I love machine learning"
"Machine learning is fun"
When you apply CountVectorizer, it would perform the following steps:

Tokenization:

Sentence 1: "I", "love", "machine", "learning"
Sentence 2: "Machine", "learning", "is", "fun"
Vocabulary Building:

The vocabulary would include all unique words: ["I", "love", "machine", "learning",
Counting:

Sentence 1: [1, 1, 1, 1, 0, 0] (1 occurrence each of "I", "love", "machine", "learn
Sentence 2: [0, 0, 1, 1, 1, 1] (1 occurrence each of "machine", "learning", "is", "
Output:
The output is a matrix where each row corresponds to a document and each column cor

Why Use CountVectorizer?
Text Representation: It provides a simple and interpretable way to represent text d
Feature Extraction: It helps in extracting features from text for tasks like text c
"""
```

In [36]:

```
from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_count = v.fit_transform(X_train.values)
X_train_count.toarray()[:3]
```

Out[36]:

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

In [ ]:

```
#Naive Bayes has thre types of calssifiers :
#1. Bernoulli Naive Bayes - for 0 and 1 values to be checked
#2. Multinomial Naive Bayes - for discrete dat(rating of movie)
#3. Gausssian Naive Bayes - for continuous data , which can't be that easily differ
```

In [37]:

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)
```

Out[37]:

```
▾ MultinomialNB
MultinomialNB()
```

In [ ]:

```
email = [
     'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this rew
]
email_count = v.transform(email)
model.predict(email_count)

spam_labels = ['Spam', 'Not Spam']
result = [spam_labels[pred] for pred in predictions]

# Output the results
```

```
for msg, label in zip(email, result):
    print(f"Email: '{msg}' => Classification: {label}")
```

In [41]:

```
X_test_count = v.transform(X_test)
model.score(X_test_count,y_test)
```

Out[41]:

```
0.9865470852017937
```

In [42]:

```
#this is another method without countvectorizer
```

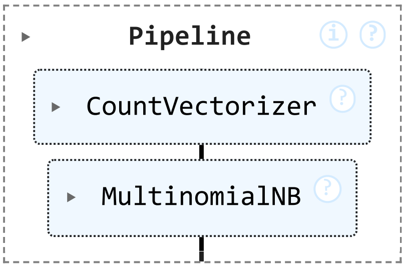
In [45]:

```
from sklearn.pipeline import Pipeline
clf = Pipeline([
    ('vectorizer',CountVectorizer()),
                ('nb',MultinomialNB())
])
```

In [46]:

```
clf.fit(X_train,y_train)
```

Out[46]:



In [48]:

```
clf.predict(email)
```

Out[48]:

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

In [49]:

```
#Pipeline is used to simply the above code!!
```

In [ ]: