naive_bayes_2_email_spam_

naive_bayes_2_email_spam_

```
import pandas as pd


df=pd.read_csv("/content/spam.csv")
df.head()
```

	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...

+ Code + Text

```
df.groupby('Category').describe()
```

	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

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

	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

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df.Message,df.spam)


#Count Vectorizer:This is a method for converting
#text documents into a matrix of token counts
from sklearn.feature_extraction.text import CountVectorizer
v =  CountVectorizer()
X_train_count=v.fit_transform(X_train.values)
#trying to acess the first two rows
X_train_count.toarray()[:2]
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
#MultinomialNB: This is a class implementing the
#Multinomial Naive Bayes algorithm,
#which is suitable for classification with discrete features
#eg.,word counts for text classification
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)
```

MultinomialNB
MultinomialNB()

```
emails=[
    'Hey mohan, can we grt together to watch football game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you.Dont miss this reward!'
]
emails_count=v.transform(emails)
model.predict(emails_count)
```

```
array([0, 1])
```

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

```
0.9863603732950467
```

```
#Sklearn Pipline
#it sequentially applies a list of transforms and a final estimator.
#Each step in the pipeline, except for the last
from sklearn.pipeline import Pipeline
clf=Pipeline([
    ('vectorizer',CountVectorizer()),
    ('nb',MultinomialNB())
])
```

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

Pipeline
CountVectorizer
MultinomialNB

```
clf.score(X_test,y_test)
```

```
0.9863603732950467
```

```
clf.predict(emails)
```

```
array([0, 1])
```