Aim: - Implement a Spam classifier using Nave Bayes classifier.
Requirement: -Python versions 3.7, 3.8, 3.9, 3.10 or 3.11, jupyter notebook

Theory: -

1. Data collection: Obtain a labeled dataset that consists of examples of both spam and non-spam (ham) messages. This dataset will be used for training and evaluating the spam classifier.

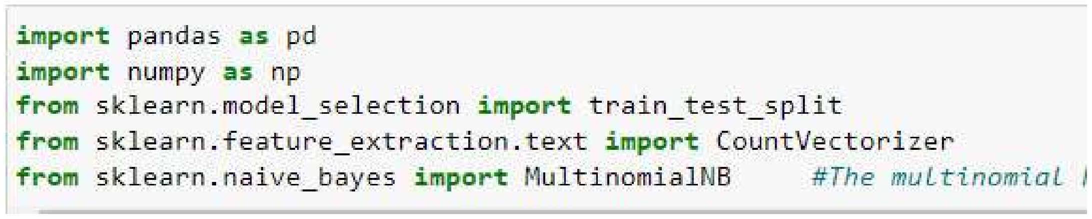
2. Data preprocessing: Preprocess the text data to clean it and convert it into a format suitable for machine learning algorithms. This may include steps such as lowercasing, removing punctuation, tokenization, removing stop words, and stemming or lemmatization.

3. Feature extraction: Transform the preprocessed text data into numerical features that can be used as input for the spam classifier. Common techniques for feature extraction in NLP include bag-of-words, TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings (e.g., Word2Vec or GloVe), or more advanced methods like BERT (Bidirectional Encoder Representations from Transformers).

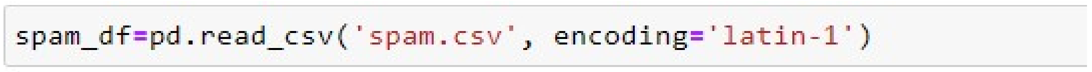
4. Training the classifier: Split the labelled dataset into training and testing sets. Use the training set to train the spam classifier using machine learning algorithms such as Naive Bayes, SVM (Support Vector Machine), or a neural network. The choice of algorithm will depend on the size of the dataset and the complexity of the problem.

5. Evaluation: Evaluate the performance of the trained spam classifier on the testing set. Common evaluation metrics for classification tasks include accuracy, precision, recall, and F1 score. Analyze the results to assess the classifier's effectiveness in identifying spam messages.

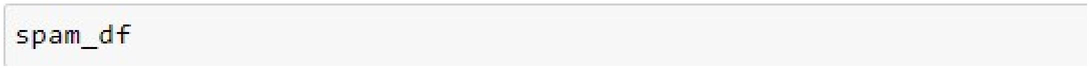
Implement a Spam classifier in Natural Language Processing.



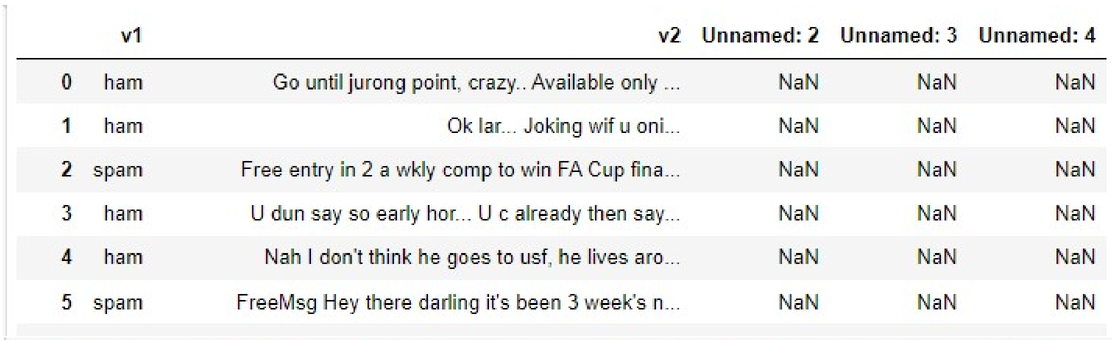
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB     #The multinomial
```



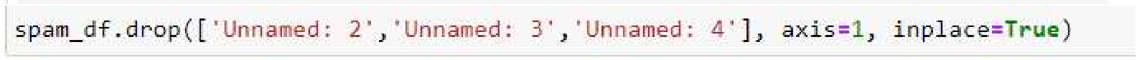
```
spam_df=pd.read_csv('spam.csv', encoding='latin-1')
```



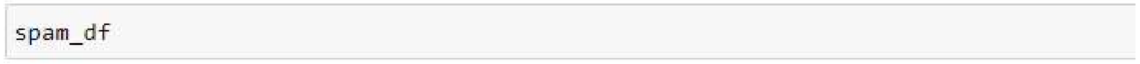
```
spam_df
```



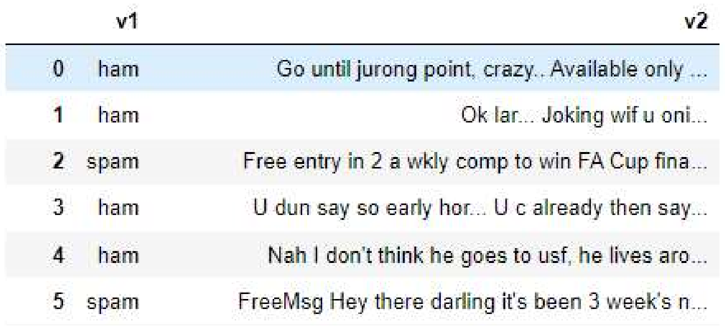
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN



```
spam_df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'], axis=1, inplace=True)
```



```
spam_df
```



	v1	v2
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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...

```python
spam_df.rename(columns={'v1':'Category','v2':'Message'},inplace=True)
```

```python
spam_df
```

	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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...

```python
spam_df.groupby('Category').describe()
```

	Message			
	count	**unique**	**top**	**freq**
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

```python
# turn spam/ham into numerical data, creating a new column called 'spam'
spam_df['spam']=spam_df['Category'].apply(lambda x:1 if x=='spam' else 0)
```

```python
spam_df
```

	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

```python
#create train/test split
x_train,x_test,y_train,y_test=train_test_split(spam_df.Message, spam_df.spam)
```

```python
x_train
```

```
1988                               Sorry, I'll call later
1216    You have 1 new voicemail. Please call 08719181...
2334                               Do you like Italian food?
1027          Are you not around or just still asleep? :V
2262                 It should take about  &lt;#&gt;  min
273                    Usf I guess, might as well take 1 car
330                                K.k:)apo k.good movie.
4427    Aiyar dun disturb u liao... Thk u have lots 2 ...
```

```
x_train.describe()
```

```
count                          4179
unique                         3929
top       Sorry, I'll call later
freq                             24
Name: Message, dtype: object
```

```
"""
CountVectorizer means breaking down a sentence or any text into words
by performing preprocessing tasks like converting all words to lowercase, thus removing special characters.
In NLP models can't understand textual data they only accept numbers, so this textual data needs to be vectorized.
"""
#find word count and store data as a matix
cv=CountVectorizer()
x_train_count=cv.fit_transform(x_train.values)
```

```
x_train_count.toarray()
```

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

```
"""
The fit() method takes the training data as arguments,
which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning.
"""
#train model
model=MultinomialNB()
model.fit(x_train_count,y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
""" cv.transform performs the matrix transformation of each element of the input array"""
#pre-test ham
email_ham=["hey wanna meet up for the game?"]
email_ham_count=cv.transform(email_ham)
model.predict(email_ham_count)
```

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

```
#pre-test spam
email_spam=["reward money click"]
email_spam_count=cv.transform(email_spam)
model.predict(email_spam_count)
```

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

```
# test model
x_test_count=cv.transform(x_test)
model.score(x_test_count,y_test)
```

```
0.9863603732950467
```