REPORT

The dataset is picked from kaggle. Name of the dataset is spam-~~ham~~-dataset.csv.
It initially contained four columns, ~~we~~ two columns were removed, only the column containing the email texts and spam - non-spam label are used. (0/1 binary label values).

Stop words :- These are those words which occur very frequently in texts or emails. So they do not add a lot of meaning to the message. It is not necessary to take them in account for the classifier. So, these words should be removed.

Many words refer to the same activity/moment. for eg. Analyse, Analysis, Analysing etc. they all refer to the same activity.
So instead of ~~using~~ adding these different words we can include a single word accounting for all related words. Thus simplifying the process for further steps.
This removal or accounting one for all is known as stemming.

Porter stemmer is a very famous stemming algorithm and we will be using it stem words.

→ Also every text should be first converted to lower case or capital case. because PANKAJ and 'pankaj', mean the same but they will be treated differently while parsing. So they should be either converted to lower case or capital case. We have converted all to lower case.

Sometimes it so happens that group of words means different from single words in the text. 'not' 'winner' means very different from 'not winner'. We will be storing two words in single entry.

Spliting the dataset into training & testing data :-

The dataset is split into training data and testing dataset.
80% of the randomly selected data is used for training and the remaining is used for testing.

Preprocessing -

Preprocessing includes tokenizing of words in the mails.
Eliminating stop words and storing the remaing to a list of words.
Stemming is then applied on that list.

```
def Preprocess(email):
    email = email.lower()
    words = word_tokenize(email)
    wrds = []
    for i in words:
        if len(i) > 2:
            words.append(i)

    sw = stopwords.words('english')
    words1 = []
    for i in words:
        if i not in sw:
            words1.append(i)

    stemmer = PorterStemmer()
    words2 = []
    for i in words1:
        words2.append(stemmer.stem(i))
```

Algorithm:-

Bayes algorithm is used to implement spam-ham classifier.

A vocabulary of unique words is maintained.

Frequency of all the words in the vocabulary is calculated.

Occurence of the word in spam mail and non-spam mail is calculated.

Probability of word is calculated.

$$\text{Prob}(w) = \frac{\text{No. of times 'word' occured in dataset}}{\text{total number of words}}$$

$$P(w/\text{spam}) = \text{probability of word in a spam mail}$$

$$= \frac{\text{count of } w \text{ in spam mails}}{\text{total (spam mails)}}$$

Inverse document frequency (IDF) is also calculated to determine how rare or how common the given word is.

If a word is seen a lot then that word does not give much information / not much useful to us. IDF of a word is calculated as:

$$IDF(w) = \log \frac{\text{total (mails)}}{\text{total no. of mails having the word } w}$$

let each word has a score = (freq. of word) * IDF(w).

$$P(w) = \frac{\text{score}(w)}{\text{sum of each words score}}$$

$$P(w/\text{spam}) = \frac{(\text{freq. of a word in a spam mail}) \times IDF(w)}{\left(\text{sum of } (\text{freq. of word/spam mai}) \times IDF(w) \text{ for each word.}\right)}$$

If we encounter a word which is not in the data then $P(w) = 0$.

To overcome this we add $\alpha = 1$ to the numerator and add α times number of words in the denominator.

This is know as additive smoothing, particularly Lapalacian smoothing

Classification -

1. find $P(w/\text{spam})$
2. If word does not exist in the dataset then freq $(w) = 0$ & find $P(w/\text{spam})$ using additive smoothing.
3. $P(\text{spam/email}) = \Sigma P(\text{spam}) \times P(w/\text{spam})$.
   similarly find $P(\text{ham/email})$.
4. If P.Ham > Pspam → 0 (ham)
   else 1 (spam mail)

→ functions in class classifier:-

① TrainAlgo( ) → calling key function
② WordFrequency( ) → calculating ~~freq~~ frequency
③ CalcProb( ) → calculating probabilities
④ classify(self, ProcessedEmails):
⑤ Predict(self, test).

→ Main() method initializes class classifier and trains algorithm on training dataset.

It calculates Prediction on Test dataset.

Also Calculates Accuracy.