```
##This block is only for access of files using google drive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

#For accessing any file from google drive, first share it for public access. Copy its id f
downloaded = drive.CreateFile({'id':"1MQ7ZftvZMoF3zlb56TJb-QwmTco5O7an"})   # replace the
downloaded.GetContentFile('spam.csv')        # replace the file name with your file


import pandas as pd
import numpy as np
import string

#import the data file
filename = 'spam.csv'

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

	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

```
#Remove the unwanted columns
df_sms = df_sms.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df_sms = df_sms.rename(columns={"v1":"label", "v2":"sms"})
df_sms.head()
```

	label	sms
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...

```
#Print number of records
l = len(df_sms)
```

```
print(L)
#Example of accessing a column in pandas dataframe
df_sms.sms
```

```
5572
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                            Rofl. Its true to its name
Name: sms, Length: 5572, dtype: object
```

```
#Define a Function to convert sms text to Lower case and remove stop words, punctuation an

def preprocess_Text(input_Text):
  input_Text = input_Text.lower();
  stopwords = ['the','what','is','a','an','of', 'that']
  querywords = input_Text.split()

  resultwords  = [word for word in querywords if word not in stopwords]
  result = ' '.join(resultwords)

  exclude = set(string.punctuation)
  result = ''.join(ch for ch in result if ch not in exclude)

  exclude = set('0123456789')
  result = ''.join(ch for ch in result if ch not in exclude)

  return result;


#Preprocess all the sms texts
L = len(df_sms)
for i in range(0,L-1):
  df_sms['sms'][i] = preprocess_Text(df_sms['sms'][i])


#Divide the dataframes into training and and testing set
from sklearn.utils import shuffle
df_sms = shuffle(df_sms)
training_Subset = df_sms.iloc[:round(len(df_sms)*0.9),:]   #90% data into training
test_Subset = df_sms.iloc[round(len(df_sms)*0.9):,:]       #10% data into testing
spam_Subset = training_Subset.query('label == "spam"')
ham_Subset = training_Subset.query('label == "ham"');


#combine all text into one large paragraph which shall be used to list unique words
L = len(training_Subset);
all_Text = ""
for i in training_Subset.index:
  all_Text = all_Text + " "+training_Subset['sms'][i];
```

```
#make a table with all unique words
allWords = all_Text.split()

row_Names = []
for i in allWords:
      if not i in row_Names:
          row_Names.append(i);
print(row_Names)
```

```
['we', 'got', 'divorce', 'lol', 'shes', 'here', 'i', 'want', 'some', 'cock', 'my', 'h
```

```
#For each word find inspam probability and in-ham probability
word = '';
inSpamCount = 0;
inHamCount = 0;
columns = ['inSpamProbability','inHamProbability']
probability_Table = pd.DataFrame(index=row_Names, columns=columns)
for word in row_Names:
  inSpamCount = 0;
  inHamCount = 0;
  for i in spam_Subset['sms']:
    if(i.find(word)==0):
      inSpamCount = inSpamCount+1;


  for i in ham_Subset['sms']:
    if(i.find(word)==0):
       inHamCount = inHamCount+1;


  probability_Table.at[word, 'inSpamProbability'] = inSpamCount/len(spam_Subset);
  probability_Table.at[word,'inHamProbability'] = inHamCount/len(ham_Subset);



probability_Table.sort_values("inSpamProbability", axis = 0, ascending = False,
                 inplace = True, na_position ='first')


probability_Table
```

	inSpamProbability	inHamProbability
u	0.1261	0.0186937
f	0.0982405	0.0131549
y	0.0982405	0.0666974
you	0.0953079	0.0207708
yo	0.0953079	0.0246942
...	...	...
did	0.00146628	0.00484653
it	0.00146628	0.0143088
an	0.00146628	0.0103854
are	0.00146628	0.00807754
life	0.00146628	0.000923148

204 rows × 2 columns

```
#drop rows wherever spam or ham is zero probability
probability_Table = probability_Table[(probability_Table[['inSpamProbability','inHamProbab
```

```
#Question_1:
def check_message(test_msg):
 test_msg = test_msg.split()
 word=""
 spam_prob=1
 ham_prob=1
 for word in test_msg:
  for words in probability_Table.index:
   if(word == words):
    spam_prob = spam_prob * probability_Table['inSpamProbability'][word]
    ham_prob = ham_prob * probability_Table['inHamProbability'][word]

 if(spam_prob >= ham_prob):
   return "spam"
 else:
   return "ham"
```

```
test_label = []
for test_inp in test_Subset['sms']:
 out_label = check_message(test_inp)
 test_label.append(out_label)
test_Subset['prediction'] = test_label
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/use
  """
```

```
test_Subset
```

```
L = len(test_Subset)
test_spam = len(test_Subset.query('label == "spam"'))
test_spam_data = test_Subset.query('label == "spam"')
test_ham = len(test_Subset.query('label == "ham"'))
test_ham_data = test_Subset.query('label == "ham"')

print(test_spam)
print(test_ham)
```

```
65
492
```

```
#Question_2:
#True Negative Rate
ptn = len(test_spam_data.query('prediction == "spam"'))
#TN Rate
TN = ptn / test_spam
print(TN)
```

```
0.9692307692307692
```

```
#True Positive Rate
ptp = len(test_ham_data.query('prediction == "ham"'))

#TP Rate
TP = ptp / test_ham
print(TP)
```

```
0.516260162601626
```

```
#False Negative Rate
pfn = len(test_spam_data.query('prediction == "ham"'))

#FN Rate
FN = pfn / test_spam
print(FN)
```

```
0.03076923076923077
```

```
#False Positive Rate
pfp = len(test_ham_data.query('prediction == "spam"'))
```

```
#FP Rate
FP = pfp / test_ham
print(FP)
```

```
0.483739837398374
```

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
print(Accuracy)
```

```
0.7427454659161976
```

```
Error_rate = (FP + FN) / (TP + TN + FP + FN)
print(Error_rate)
```

```
0.2572545340838024
```