```
import pandas as pd
from nltk.corpus import stopwords
1.
  df= pd.read_csv("SMSSpamCollection.csv")
  sms = df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']
                        , axis=1)
  sms.
2) len(sms)
3)  a = sms.groupby('label').count()
    a
4)
  X = sms.text
  Y = sms.label
  from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test= train_test_split
                    (X, y, train_size=0.8, test_size=0.2)
5)
  def process_text(msg):
       punctuations = '''!()-[]{};:'"\,<>./?
                         @#$%^&*_~'''
     nopunc =[char for char in msg if char not in
                         punctuations]
     nopunc ='' .join(nopunc)
     return [word for word in nopunc.split()
     if word.lower() not in stopwords.words('english')]
```

Scanned with CamScanner

Natural Language Processing Lab
Lab6. Spam Filtering using Multinomial NB

In this lab, you will build Naïve Bayes classifier using SMS data to classify a SMS into spam or not. Once the model is built, it can be used to classify an unknown SMS into spam or ham.

STEPS

1. Open "SMSSpamCollection" file and load into DataFrame. It contains two columns "label" and "text".

2. How many sms messages are there?

3. How many "ham" and "spam" messages?. You need to groupby() label column.

4. Split the dataset into training set and test set (Use 20% of data for testing).

5. Create a function that will remove all punctuation characters and stop words, as below

```
def process_text(msg):
      nopunc =[char for char in msg if char not in string.punctuation]
      nopunc=''.join(nopunc)
      return [word for word in nopunc.split()
            if word.lower() not in stopwords.words('english')]
```

6. Create TfIdfVectorizer as below and perform vectorization on X_train, using fit_perform() method.

```
TfidfVectorizer(use_idf=True,
      analyzer=process_text,
      ngram_range=(1,3),
      min_df = 1,
      stop_words = 'english')
```

7. Create MultinomialNB model and perform training on X_train and y_train using fit() method

8. Predict labels on the test set, using predict() method

9. Print confusion_matrix and classification_report

10. Modify ngram_range=(1,2) and perform Steps 7 to 9.

6)

```
from sklearn.feature_extraction.text import
                                   TfidfVectorizer.
tf2 = TfidfVectorizer(use_idf = True, analyzer = process_text,
                    ngram_range = (1,3), min_df =
tf2.
  m2 = tf2.fit_transform(x_train)
  my2 = tf2.transform(x_test)
```

```
mz . shape

my2 . shape .
```

7)

```
X_train , X_test , Y_train , Y_test = train_test_split( X,y,train_size
                                                          =0.8,

from sklearn.naive_bayes import MultinomialNB .

  clf = MultinomialNB()

  clf . fit ( m2 , y_train)
```

8)

```
y_pred = clf . predict(my2)
y_pred
```

9)

```
from sklearn. metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
from sklearn. metrics import classification_report.
target_names = [ 'Class 0', 'Class 1' ]
print ( classification_report ( y_test , y_pred , target_names
                                               = target_names))
```

NOTES

```
10) Modify ngram range = (1,2)
tf3 = Tfidfvectorizer (use idf = True, analyzer = process_text,
                        ngram_range(1,2),
tf3.
m3 = tf3. fit_transform (X_train)
my3 = tf3. transform (X_test)
m3. shape.
my3. shape.
clf. fit (m3, y_train)
y_pred3 = clf. predict (my3)
y_pred3.
confusion_matrix (y_test, y_pred3)
target_names = ['class 0', 'class 1']
print (classification_report (y_test, y_pred3, target_
                                names = target_names))
```