

# Lab:6. Spam Filtering using Multinomial NB

## Annapoornima S(225229101)

### Step-1

```
In [1]: import pandas as pd
```

```
In [2]: data = pd.read_csv("SMSSpamCollection.csv",encoding='latin-1')
data.head()
```

```
Out[2]:
```

	label	text	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

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

```
In [4]: data.head()
```

```
Out[4]:
```

	label	text
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...

### Step-2

```
In [6]: data['text'].value_counts().sum()
```

```
Out[6]: 5572
```

### Step-3

```
In [7]: data.groupby(['label']).count()
```

```
Out[7]:
```

	text
label	
ham	4825
spam	747

### Step-4

```
In [8]: X = data['text']
        y = data['label']
```

```
In [9]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

### Step-5

```
In [10]: from nltk.corpus import stopwords
        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')]
```

```
In [11]: import nltk
        nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\1mscdsa17\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\stopwords.zip.
```

```
Out[11]: True
```

## Step-6

```
In [12]: from sklearn.feature_extraction.text import TfidfVectorizer
df1 = TfidfVectorizer(use_idf=True,
                      analyzer = process_text,
                      ngram_range=(1,3),
                      min_df=1,
                      stop_words = 'english')

df1
```

```
Out[12]: TfidfVectorizer(analyzer=<function process_text at 0x000001CC42812F70>,
                        ngram_range=(1, 3), stop_words='english')
```

```
In [13]: a = df1.fit_transform(X_train)
```

```
In [14]: a1 = df1.transform(X_test)
```

## Step-7

```
In [15]: from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(a,y_train)
```

```
Out[15]: MultinomialNB()
```

## Step-8

```
In [16]: y_pred = clf.predict(a1)
y_pred
```

```
Out[16]: array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'spam'], dtype='<U4')
```

```
In [17]: from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

```
Out[17]: array([[965,  0],
                [ 39, 111]], dtype=int64)
```

## Step-9

```
In [18]: from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	965
spam	1.00	0.74	0.85	150
accuracy			0.97	1115
macro avg	0.98	0.87	0.92	1115
weighted avg	0.97	0.97	0.96	1115

## Step-10

```
In [19]: from sklearn.feature_extraction.text import TfidfVectorizer
df2 = TfidfVectorizer(use_idf=True,
                      analyzer = process_text,
                      ngram_range=(1,2),
                      min_df=1,
                      stop_words = 'english')

df2
```

```
Out[19]: TfidfVectorizer(analyzer=<function process_text at 0x000001CC42812F70>,
                        ngram_range=(1, 2), stop_words='english')
```

```
In [20]: b = df2.fit_transform(X_train)
b1= df2.transform(X_test)
```

```
In [21]: from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(b,y_train)
```

```
Out[21]: MultinomialNB()
```

```
In [22]: y1_pred = clf.predict(b1)
y1_pred
```

```
Out[22]: array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'spam'], dtype='<U4')
```

```
In [23]: confusion_matrix(y_test,y1_pred)
```

```
Out[23]: array([[965,  0],
                [ 39, 111]], dtype=int64)
```

```
In [24]: print(classification_report(y_test,y1_pred))
```

	precision	recall	f1-score	support
ham	0.96	1.00	0.98	965
spam	1.00	0.74	0.85	150
accuracy			0.97	1115
macro avg	0.98	0.87	0.92	1115
weighted avg	0.97	0.97	0.96	1115

```
In [ ]:
```