A SMS unsolicited mail (every now and then known as cell smartphone junk mail) is any junk message brought to a cellular phone as textual content messaging via the Short Message Service (SMS). Use probabilistic approach (Naive Bayes Classifier / Bayesian Network) to implement SMS Spam Filtering system. SMS messages are categorized as SPAM or HAM using features like length of message, word depend, unique keywords etc. Download Data -Set from : http://archive.ics.uci.edu/ml/datasets/sms+spam+collection This dataset is composed by just one text file, where each line has the correct class followed by the raw message. a. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary b. Perform data-preparation (Train-Test Split) c. Apply at least two Machine Learning Algorithms and Evaluate Models d. Apply Cross-Validation and Evaluate Models and compare performance. e. Apply Hyper parameter tuning and evaluate models and compare performance.

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('/content/spam.csv',encoding='ISO-8859-1')
df.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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

```
df.groupby('v1').describe()
```

v2	Unnamed: 2	Unnamed: 3

```
df['spam'] = df['v1'].apply(lambda x:1 if x=='spam' else 0)
```

```
df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	spam
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN	0
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN	1
		U dun sav so earlv hor... U c alreadv				

```
new_df = df[['v1','v2','spam']]
```

```
new_df.head()
```

	v1	v2	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
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
from sklearn.model_selection import train_test_split as tts
x_train,x_test,y_train,y_test=tts(df.v2,df.spam)
```

```
from sklearn.feature_extraction.text import CountVectorizer

v=CountVectorizer()
x_train_count=v.fit_transform(x_train.values)
x_train_count.toarray()[:2]
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
from sklearn.naive_bayes import MultinomialNB
model=MultinomialNB()
model.fit(x_train_count,y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
emails=["How are you brother?", "Free entry"]

email_count=v.transform(emails)
model.predict(email_count)
```

```
array([0, 1])
```

```
x_test_count=v.transform(x_test)
model.score(x_test_count,y_test)
```

```
0.9885139985642498
```


✓ 0s completed at 9:05 AM