PROGRAM : 1

AIM: Program to implement text classification using Support vector machine.

```
import nltk
import pandas as pd


nltk.download_shell()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> l

Packages:
  [ ] abc................. Australian Broadcasting Commission 2006
  [ ] alpino.............. Alpino Dutch Treebank
  [ ] averaged_perceptron_tagger Averaged Perceptron Tagger
  [ ] averaged_perceptron_tagger_ru Averaged Perceptron Tagger (Russian)
  [ ] basque_grammars..... Grammars for Basque
  [ ] biocreative_ppi..... BioCreAtIvE (Critical Assessment of Information
                           Extraction Systems in Biology)
  [ ] bllip_wsj_no_aux.... BLLIP Parser: WSJ Model
  [ ] book_grammars....... Grammars from NLTK Book
  [ ] brown............... Brown Corpus
  [ ] brown_tei........... Brown Corpus (TEI XML Version)
  [ ] cess_cat............ CESS-CAT Treebank
  [ ] cess_esp............ CESS-ESP Treebank
  [ ] chat80.............. Chat-80 Data Files
  [ ] city_database....... City Database
  [ ] cmudict............. The Carnegie Mellon Pronouncing Dictionary (0.6)
  [ ] comparative_sentences Comparative Sentence Dataset
  [ ] comtrans............ ComTrans Corpus Sample
  [ ] conll2000........... CONLL 2000 Chunking Corpus
  [ ] conll2002........... CONLL 2002 Named Entity Recognition Corpus
Hit Enter to continue: q

---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
```

```
messages=[line.rstrip() for line in open('/content/sample_data/SMSSpamCollection')]


print(len(messages))
```

```
5574
```

```
messages[0]
```

```
'ham\tGo until jurong point, crazy.. Available only in bugis n great worl
d la e buffet... Cine there got amore wat...'
```

```
for mess_no,message in enumerate(messages[:10]):
  print(mess_no,message)
  print('/n')
```

```
0 ham   Go until jurong point, crazy.. Available only in bugis n great world la e
/n
1 ham   Ok lar... Joking wif u oni...
/n
2 spam  Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text I
/n
3 ham   U dun say so early hor... U c already then say...
/n
4 ham   Nah I don't think he goes to usf, he lives around here though
/n
5 spam  FreeMsg Hey there darling it's been 3 week's now and no word back! I'd lil
/n
6 ham   Even my brother is not like to speak with me. They treat me like aids pate
/n
7 ham   As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has
/n
8 spam  WINNER!! As a valued network customer you have been selected to receivea f
/n
9 spam  Had your mobile 11 months or more? U R entitled to Update to the latest co
/n
```

```
messages[0]
```

```
'ham\tGo until jurong point, crazy.. Available only in bugis n great worl
d la e buffet... Cine there got amore wat...'
```

```
import pandas as pd
```

```
messages=pd.read_csv('/content/sample_data/SMSSpamCollection',sep='\t',names=['label','
```

```
messages.head()
```

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...

```
#convert string into vector format
import string
```

```
#remove punctuations and stopwords
mess="sample message ! Notice: it has punctation."


string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
nopunc=[c for c in mess if c not in string.punctuation ]


nopunc
```

```
['s',
 'a',
 'm',
 'p',
 'l',
 'e',
 ' ',
 'm',
 'e',
 's',
 's',
 'a',
 'g',
 'e',
 ' ',
 ' ',
 'N',
 'o',
 't',
 'i',
 'c',
 'e',
 ' ',
 'i',
 't',
 ' ',
 'h',
 'a',
 's',
 ' ',
```

```
      'p',
      'u',
      'n',
      'c',
      't',
      'a',
      't',
      'i',
      'o',
      'n']


nopunc=''.join(nopunc)


nopunc
```

```
     'sample message  Notice it has punctation'
```

```
#remove stopwords
from nltk.corpus import stopwords


import nltk
nltk.download('stopwords')
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True
```

```
nopunc.split()
```

```
     ['sample', 'message', 'Notice', 'it', 'has', 'punctation']
```

```
clean_mess=[word for word in nopunc.split() if word.lower() not in stopwords.words("eng


clean_mess
```

```
     ['sample', 'message', 'Notice', 'punctation']
```

```
#apply to actual dataset
def text_process(mess):
  nopunc=[char for char in mess if char not in string.punctuation]
  nopunc="".join(nopunc)
  return[word for word in nopunc.split() if word.lower() not in stopwords.words("englis


messages.head()
```

	label	message
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...

```
messages['message'].head(5).apply(text_process)
```

```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: message, dtype: object
```

```
#CONVERTING into vectors
from sklearn.feature_extraction.text import CountVectorizer
```

```
bow_transformer=CountVectorizer(analyzer=text_process).fit(messages['message'])
```

```
print(len(bow_transformer.vocabulary_))
```

```
11425
```

```
mess4=messages['message'][6]
```

```
print(mess4)
```

```
Even my brother is not like to speak with me. They treat me like aids patent.
```

```
bow4=bow_transformer.transform([mess4])
```

```
print(bow4)
```

```
  (0, 1802)     1
  (0, 4590)     1
  (0, 5193)     1
  (0, 7800)     2
  (0, 8761)     1
  (0, 9971)     1
  (0, 10629)    1
```

```
bow_transformer.get_feature_names()[7800]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Fu
  warnings.warn(msg, category=FutureWarning)
'like'
```

```
#transform whole msg
messages_bow=bow_transformer.transform(messages['message'])
```

```
print('shape of sparse matrix:',messages_bow.shape)
```

```
shape of sparse matrix: (5572, 11425)
```

```
messages_bow.nnz
```

```
50548
```

```
#term frequency, inverse document frequency
from sklearn.feature_extraction.text import TfidfTransformer
```

```
tfidf_transformer=TfidfTransformer().fit(messages_bow)
```

```
tfidf4=tfidf_transformer.transform(bow4)
```

```
print(tfidf4)
```

```
  (0, 10629)    0.3352766696931058
  (0, 9971)     0.3268691780062757
  (0, 8761)     0.43700993321905807
  (0, 7800)     0.41453906826037096
  (0, 5193)     0.33843411088434017
  (0, 4590)     0.43700993321905807
  (0, 1802)     0.3352766696931058
```

```
messages_tfidf=tfidf_transformer.transform(messages_bow)
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
spam_detect_model=MultinomialNB().fit(messages_tfidf,messages['label'])
```

```
all_pred=spam_detect_model.predict(messages_tfidf)
```

```
all_pred
```

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

```
from sklearn.model_selection import train_test_split
```

```
msg_train,msg_test,label_train,label_test=train_test_split(messages['message'],messages
```

```
spam_detect_model=MultinomialNB().fit(messages_tfidf,messages['label'])
```

```
predict=spam_detect_model.predict(messages_tfidf)
```

```
from sklearn.metrics import classification_report
print(classification_report(messages['label'],predict))
```

```
                  precision    recall  f1-score   support

             ham       0.98      1.00      0.99      4825
            spam       1.00      0.85      0.92       747

        accuracy                           0.98      5572
       macro avg       0.99      0.92      0.95      5572
    weighted avg       0.98      0.98      0.98      5572
```

```
#train test split
from sklearn.model_selection import train_test_split
msg_train,msg_test,label_test,label_train = \
train_test_split(messages['message'],messages['label'],test_size=0.2)
```

```
#pipeline
from sklearn.pipeline import Pipeline
pipeline = Pipeline([('bow',CountVectorizer(analyzer=text_process)),
                     ('tfidf',TfidfTransformer()),
                     ('classifier',MultinomialNB()),
                     ])
```

```
pipeline.fit(msg_test,label_train)
```

```
Pipeline(steps=[('bow',
                 CountVectorizer(analyzer=<function text_process at 0x7fa5719314d0
                ('tfidf', TfidfTransformer()),
                ('classifier', MultinomialNB())])
```

```
predictions=pipeline.predict(msg_test)
```

```
print(classification_report(predictions,label_train))
```

```
                  precision    recall  f1-score   support

             ham       1.00      0.96      0.98      1011
            spam       0.70      1.00      0.83       104

        accuracy                           0.96      1115
       macro avg       0.85      0.98      0.90      1115
    weighted avg       0.97      0.96      0.96      1115
```

```
#svm classifier
from sklearn import model_selection,naive_bayes, svm
from sklearn.metrics import accuracy_score
```

```
pipeline1 = Pipeline([('bow',CountVectorizer(analyzer=text_process)),
                      ('tfidf',TfidfTransformer()),
                      ('classifier',svm.SVC(C=1.0,kernel='linear',degree=3,gamma='auto')
                      ])
```

```
pipeline1.fit(msg_test,label_train)
```

```
    Pipeline(steps=[('bow',
                     CountVectorizer(analyzer=<function text_process at 0x7fa5719314d0
                     ('tfidf', TfidfTransformer()),
                     ('classifier', SVC(gamma='auto', kernel='linear'))])
```

```
predictions1=pipeline1.predict(msg_test)
```

```
print(classification_report(predictions1,label_train))
```

```
                  precision    recall  f1-score   support

             ham       1.00      1.00      1.00       969
            spam       0.99      1.00      0.99       146

        accuracy                           1.00      1115
       macro avg       0.99      1.00      1.00      1115
    weighted avg       1.00      1.00      1.00      1115
```