SVM TEXT CLASSIFICATION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
```

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

	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
		U dun say so early hor... U c already then			

```
dataset.shape
```

```
(5572, 5)
```

```
dataset=dataset.replace({'label':'spam'},1)
dataset=dataset.replace({'label':'ham'},0)
```

```
dataset.head()
```

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	0	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	0	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
		U dun say so early hor... U c already then			

✓ 0s completed at 12:27 PM ● ✕

```
vectorized_data=vectorizer.fit_transform(x for x in x)

vectorized_data = pd.DataFrame(vectorized_data.toarray())
print(vectorized_data.head())

#obtaining the token names

tdfidf_tokens=vectorizer.get_feature_names()
vectorized_data=vectorized_data.set_axis(tdfidf_tokens,axis=1,inplace=False)


print(vectorized_data.head())
```

```
     0    1    2    3    4    5   ...  8398  8399  8400  8401  8402  840
0  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.
1  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.
2  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.
3  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.
4  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.

[5 rows x 8404 columns]
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
    00  000  000pes  008704050406  0089  ...  ûªve   ûï  ûïharry   ûò  ûówell
0  0.0  0.0     0.0           0.0   0.0  ...   0.0  0.0      0.0  0.0     0.0
1  0.0  0.0     0.0           0.0   0.0  ...   0.0  0.0      0.0  0.0     0.0
2  0.0  0.0     0.0           0.0   0.0  ...   0.0  0.0      0.0  0.0     0.0
3  0.0  0.0     0.0           0.0   0.0  ...   0.0  0.0      0.0  0.0     0.0
4  0.0  0.0     0.0           0.0   0.0  ...   0.0  0.0      0.0  0.0     0.0

[5 rows x 8404 columns]
```

```
x_train,x_test,y_train,y_test=train_test_split(vectorized_data,y,test_size=0.2)
```

Linear SVM

```
svc=SVC(kernel='linear')
svc.fit(x_train,y_train)
```

```
SVC(kernel='linear')
```

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       973
           1       0.99      0.90      0.94       142

    accuracy                           0.99      1115
   macro avg       0.99      0.95      0.97      1115
weighted avg       0.99      0.99      0.99      1115
```

Accuracy

```
r=accuracy_score(y_test,y_predict)
print(r*100,"%")
```

```
98.65470852017937 %
```

K SVM

RBF KERNEL

```
poly_classification=SVC(kernel='rbf',random_state=0, gamma=1, C=1)
poly_classification.fit(x_train,y_train)
```

```
SVC(C=1, gamma=1, random_state=0)
```

```
y_predict=svc.predict(x_test)
y_predict
```

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

Accuracy

```
r=accuracy_score(y_test,y_predict)
print(r*100,"%")
```

```
    98.65470852017937 %
```

POLYNIMIAL KERNEL

```
poly_classification=SVC(kernel='poly',degree=2)
poly_classification.fit(x_train,y_train)
```

```
    SVC(degree=2, kernel='poly')
```

```
y_predict=svc.predict(x_test)
y_predict
```

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

```
print(confusion_matrix(y_test,y_predict))
```

```
    [[972   1]
     [ 14 128]]
```

```
print(classification_report(y_test,y_predict))
```