sd-1

May 5, 2024

1 SPAM SMS DETECTION

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')
import re
import sklearn
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\gagan\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[2]:
```
sms = pd.read_csv(r'C:/Users/gagan/Downloads/archive (1)/spam.
 ↪csv',encoding='latin1')
sms.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
sms.head(5)
```

[2]:

	v1	v2
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…

[3]:
```
sms.head()
```

[3]:

	v1	v2
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…
```

[4]:
```
sms.shape
```

[4]:
```
(5572, 2)
```

[5]:
```
sms.drop_duplicates(inplace=True)
```

[6]:
```
sms.reset_index(drop=True, inplace=True)
```

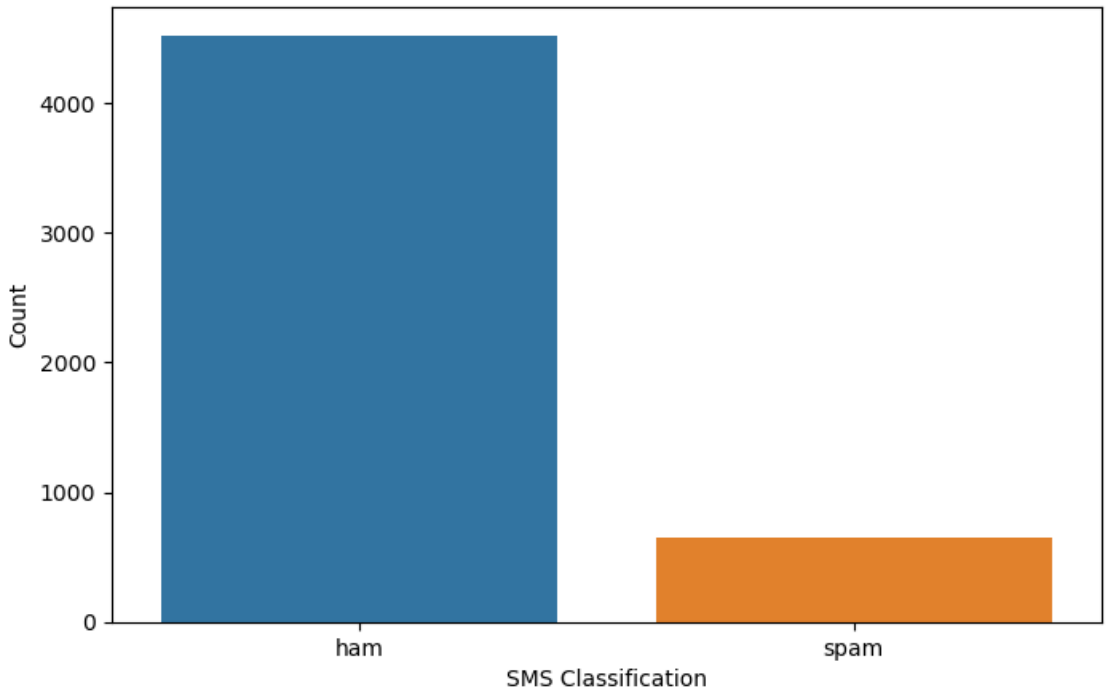
[7]:
```
sms.shape
```

[7]:
```
(5169, 2)
```

[9]:
```
sms['v1'].value_counts()
```

[9]:
```
v1
ham     4516
spam     653
Name: count, dtype: int64
```

[11]:
```
plt.figure(figsize=(8,5))
sns.countplot(x='v1', data=sms)
plt.xlabel('SMS Classification')
plt.ylabel('Count')
plt.show()
```



1.1 Cleaning the messages

[13]:
```
corpus = []
ps = PorterStemmer()

for i in range(0,sms.shape[0]):
    message = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sms.v2[i])
    message = message.lower()
    words = message.split()
    words = [word for word in words if word not in set(stopwords.
 ↪words('english'))]
    words = [ps.stem(word) for word in words]
    message = ' '.join(words)
    corpus.append(message)
```

1.2 Creating the Bag of Words model

[14]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2500)
X = cv.fit_transform(corpus).toarray()
```

1.3 Extracting dependent variable from the dataset

[16]:
```
y = pd.get_dummies(sms['v1'])
y = y.iloc[:, 1].values
```

[17]:
```
y
```

[17]:
```
array([False, False,  True, …, False, False, False])
```

1.4 train_test_split

[18]:
```
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=0)
```

1.5 Naive Bayes Classifier

[19]:
```
best_accuracy = 0.0
alpha_val = 0.0
for i in np.arange(0.0,1.1,0.1):
    temp_classifier = MultinomialNB(alpha=i)
    temp_classifier.fit(X_train, y_train)
```

```
    temp_y_pred = temp_classifier.predict(X_test)
    score = accuracy_score(y_test, temp_y_pred)
    print("Accuracy score for alpha={} is: {}%".format(round(i,1),␣
  ↪round(score*100,2)))
    if score>best_accuracy:
        best_accuracy = score
        alpha_val = i
print('--------------------------------------------')
print('The best accuracy is {}% with alpha value as {}'.
  ↪format(round(best_accuracy*100, 2), round(alpha_val,1)))
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\naive_bayes.py:629:
FutureWarning: The default value for `force_alpha` will change to `True` in 1.4.
To suppress this warning, manually set the value of `force_alpha`.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\naive_bayes.py:635:
UserWarning: alpha too small will result in numeric errors, setting alpha =
1.0e-10. Use `force_alpha=True` to keep alpha unchanged.
  warnings.warn(
```

```
Accuracy score for alpha=0.0 is: 97.58%
Accuracy score for alpha=0.1 is: 98.07%
Accuracy score for alpha=0.2 is: 98.07%
Accuracy score for alpha=0.3 is: 98.16%
Accuracy score for alpha=0.4 is: 98.07%
Accuracy score for alpha=0.5 is: 98.07%
Accuracy score for alpha=0.6 is: 98.07%
Accuracy score for alpha=0.7 is: 98.07%
Accuracy score for alpha=0.8 is: 98.07%
Accuracy score for alpha=0.9 is: 98.16%
Accuracy score for alpha=1.0 is: 98.16%
--------------------------------------------
The best accuracy is 98.16% with alpha value as 0.3
```

1.6 Fitting Naive Bayes to the Training set

[20]:
```
classifier = MultinomialNB(alpha=0.1)
classifier.fit(X_train, y_train)
```

[20]:
```
MultinomialNB(alpha=0.1)
```

1.7 Predicting the Test set results

[21]:
```
y_pred = classifier.predict(X_test)
```

[22]:
```
y_pred
```

[22]:
```
array([False, False, False, …, False, False, False])
```

1.8 Accuracy Score

[23]:
```
acc_s = accuracy_score(y_test, y_pred)*100
```

[24]:
```
print("Accuracy Score {} %".format(round(acc_s,2)))
```

```
Accuracy Score 98.07 %
```

1.9 Prediction

[25]:
```
def predict_spam(sample_message):
    sample_message = re.sub(pattern='[^a-zA-Z]',repl=' ', string =␣
 ↪sample_message)
    sample_message = sample_message.lower()
    sample_message_words = sample_message.split()
    sample_message_words = [word for word in sample_message_words if not word␣
 ↪in set(stopwords.words('english'))]
    ps = PorterStemmer()
    final_message = [ps.stem(word) for word in sample_message_words]
    final_message = ' '.join(final_message)
    temp = cv.transform([final_message]).toarray()
    return classifier.predict(temp)
```

[27]:
```
result = ['Wait a minute, this is a SPAM!','Ohhh, this is a normal message.']
```

[ ]:
```
#Hi! You are pre-qulified for Premium SBI Credit Card. Also get Rs.500 worth␣
 ↪Amazon Gift Card*, 10X Rewards Point* & more. Click
#[Update] Congratulations Nile Yogesh, You account is activated for investment␣
 ↪in Stocks. Click to invest now:
```

[ ]:
```
msg = input("Enter the test message: ")
if predict_spam(msg):
    print(result[0])
else:
    print(result[1])
```