SMS/Email Spam Classifier

Project Overview:

The SMS/Email Spam Classifier is a machine learning project designed to classify text messages and emails as either spam or non-spam. The project utilizes a dataset from Kaggle for training and employs the Naive Bayes algorithm for its excellent performance on textual datasets. The goal is to build a robust and accurate model and convert it into a website with deployment that can effectively identify and filter out spam messages.

Problem Statement:

The increasing volume of spam messages poses a significant challenge in modern communication systems. Identifying and filtering out spam is crucial to maintaining a clean and efficient communication environment. The SMS/Email Spam Classifier aims to address this problem by leveraging machine learning techniques to automatically classify incoming messages as spam or legitimate.

Project Phases:

1. Data Exploration and Preprocessing

In the initial phase, a Jupyter Notebook was used to perform exploratory data analysis (EDA) on the Kaggle dataset. This involved loading, cleaning, and manipulating the data to gain insights. Various preprocessing techniques such as normalization and word tokenization were applied to ensure the data's uniformity and enhance the model's performance.

2. Model Building and Training

The Naive Bayes algorithm, known for its effectiveness with textual datasets, was chosen to build the spam classifier model. The dataset was split into training and testing sets, and the model was trained using the training set. The evaluation metrics, including accuracy and precision, indicated a high-performance model with a 97% accuracy score and almost 100% precision score.

3. Web Application Development

Moving to PyCharm, several libraries were imported to develop a web application. Streamlit, an open-source framework, was employed to create an intuitive and user-friendly interface for the SMS/Email Spam Classifier. The application allows users to input text messages or emails and receive instant predictions on whether they are spam or not.

4. Deployment on Cloud Platform

The final phase involves deploying the web application project on a cloud platform for accessibility. This step ensures that users can utilize the spam classifier from anywhere, making it a practical and scalable solution. The choice of a cloud platform for deployment is based on individual preferences and requirements.

In [124…
```
import pandas as pd
import numpy as np
import sklearn
```

In [125…
```
data = pd.read_csv('spam.csv', encoding="ISO-8859-1")
```

In [126…
```
data.sample(8)
```

Out[126]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3828	ham	Sure, I'll see if I can come by in a bit	NaN	NaN	NaN
3229	ham	I feel like a dick because I keep sleeping thr...	NaN	NaN	NaN
2552	ham	Oh fine, I'll be by tonight	NaN	NaN	NaN
35	ham	Yup... Ok i go home look at the timings then i...	NaN	NaN	NaN
4983	spam	goldviking (29/M) is inviting you to be his fr...	NaN	NaN	NaN
2591	ham	Still work going on:)it is very small house.	NaN	NaN	NaN
2946	ham	Leave it. U will always be ignorant.	NaN	NaN	NaN
5478	ham	No probably <#> %.	NaN	NaN	NaN

In [127…
```
data.shape
```

Out[127]:
```
(5572, 5)
```

Steps:

1.Data Cleaning

2.EDA(Exploratory Data Analysis)

3.Text Preprocessing

4.Model Building

5.Model Evaluation

6.Imrovments

7.Converting into Website

8.Deployment on Heroku

1. Data Cleaning

In [128…
```
# Dropping unnessesary data:

data.drop(columns= ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace= True)
data.head()
```

Out[128]:

	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...

In [129…
```
# Renaming Columns:

data.rename(columns= {'v1': 'Target', 'v2': 'Text'}, inplace= True)
data.head()
```

Out[129]:

	Target	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...

In [130…
```
# Missing Values

data.isnull().sum()
```

Out[130]:
```
Target    0
Text      0
dtype: int64
```

In [131…
```
# Checking Duplicate Values

data.duplicated().sum()
```

Out[131]:
```
403
```

In [132…
```
# Removing Duplicate Values

data = data.drop_duplicates(keep='first')
```

In [133…
```
data.shape
```

Out[133]:
```
(5169, 2)
```

2. EDA (Exploratory Data Analysis)

In [134…
```
data.head()
```

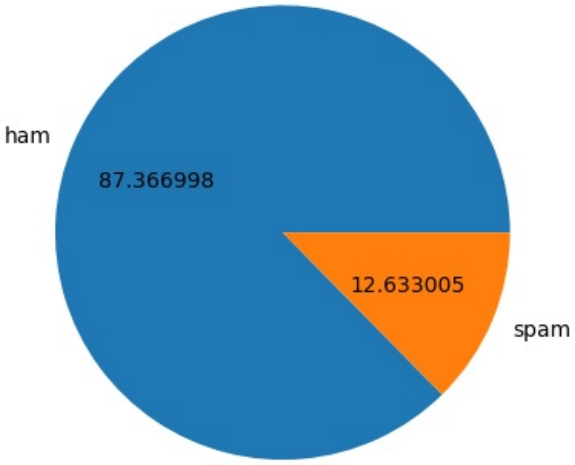
Out[134]:

	Target	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...

In [135…
```
data['Target'].value_counts()
```

Out[135]:
```
ham     4516
spam     653
Name: Target, dtype: int64
```

In [136…
```
# Presentation

import matplotlib.pyplot as plt
plt.pie(data['Target'].value_counts(), labels=['ham', 'spam'], autopct = "%0.6f")
plt.show()
```



Data is "Imbalanced"

In [137…
```
import nltk
```

In [138…
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\7mentor\newanaconda\lib\site-packages (3.7)
Requirement already satisfied: click in c:\7mentor\newanaconda\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in c:\7mentor\newanaconda\lib\site-packages (from nltk) (2022.7.
9)
Requirement already satisfied: joblib in c:\7mentor\newanaconda\lib\site-packages (from nltk) (1.3.2)
Requirement already satisfied: tqdm in c:\7mentor\newanaconda\lib\site-packages (from nltk) (4.64.1)
Requirement already satisfied: colorama in c:\7mentor\newanaconda\lib\site-packages (from click->nltk) (0.4.5)
```

In [139…
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[139]:
```
True
```

In [140…
```
data['Text'].apply(len)
```

Out[140]:
```
0       111
1        29
2       155
3        49
4        61
       ...
5567    161
5568     37
5569     57
5570    125
5571     26
Name: Text, Length: 5169, dtype: int64
```

In [141…
```
# Fetching number of characters

data['no_characters'] = data['Text'].apply(len)
```

In [142…
```
data.head()
```

Out[142]:

	Target	Text	no_characters
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

In [143…
```
# Fetching number of words

data['no_words'] = data['Text'].apply(lambda x : len(nltk.word_tokenize(x)))
```

In [144…
```
data.head()
```

Out[144]:

	Target	Text	no_characters	no_words
0	ham	Go until jurong point, crazy.. Available only ...	111	24
1	ham	Ok lar... Joking wif u oni...	29	8
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	ham	U dun say so early hor... U c already then say...	49	13
4	ham	Nah I don't think he goes to usf, he lives aro...	61	15

In [145…
```
# Fetching number of sentences

data['no_sentences'] = data['Text'].apply(lambda x : len(nltk.sent_tokenize(x)))
```

In [146…
```
data[['no_characters', 'no_words', 'no_sentences']].describe()
```

Out[146]:

	no_characters	no_words	no_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.453279	1.947185
std	58.236293	13.324793	1.362406
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	28.000000

In [147…
```
data['Target'] = data['Target'].map({'spam': 1, 'ham': 0})
```

In [148…
```
# For ham

data[data['Target'] == 0][['no_characters', 'no_words', 'no_sentences']].describe()
```

Out[148]:

	no_characters	no_words	no_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.120903	1.799601
std	56.358207	13.493725	1.278465
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	28.000000

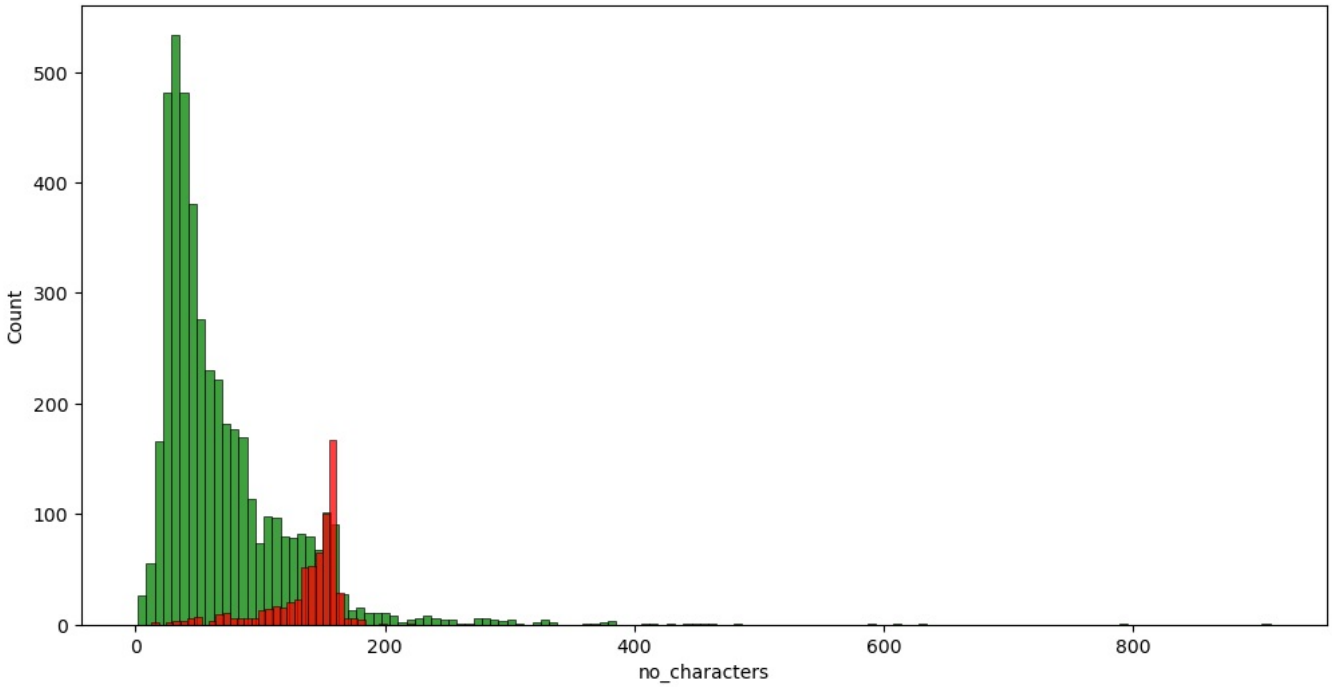
In [149…
```
# For spam

data[data['Target'] == 1][['no_characters', 'no_words', 'no_sentences']].describe()
```

Out[149]:

	no_characters	no_words	no_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.967841
std	30.137753	7.008418	1.483201
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	8.000000

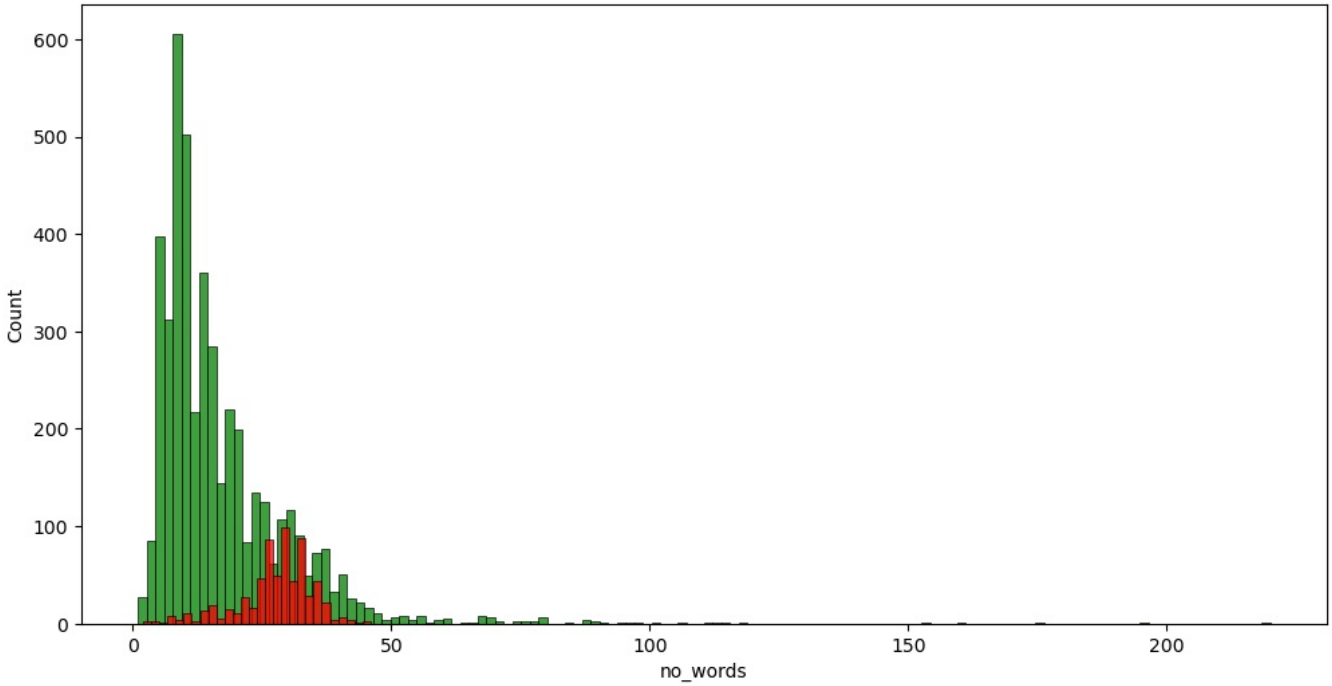
In [150…
```
# Visualization for number of characters in ham and spam messages.

plt.figure(figsize=(12,6))
sns.histplot(data[data['Target'] == 0]['no_characters'], color = 'green')   # Ham
sns.histplot(data[data['Target'] == 1]['no_characters'], color = 'red')   # Spam
```

Out[150]:
```
<AxesSubplot:xlabel='no_characters', ylabel='Count'>
```



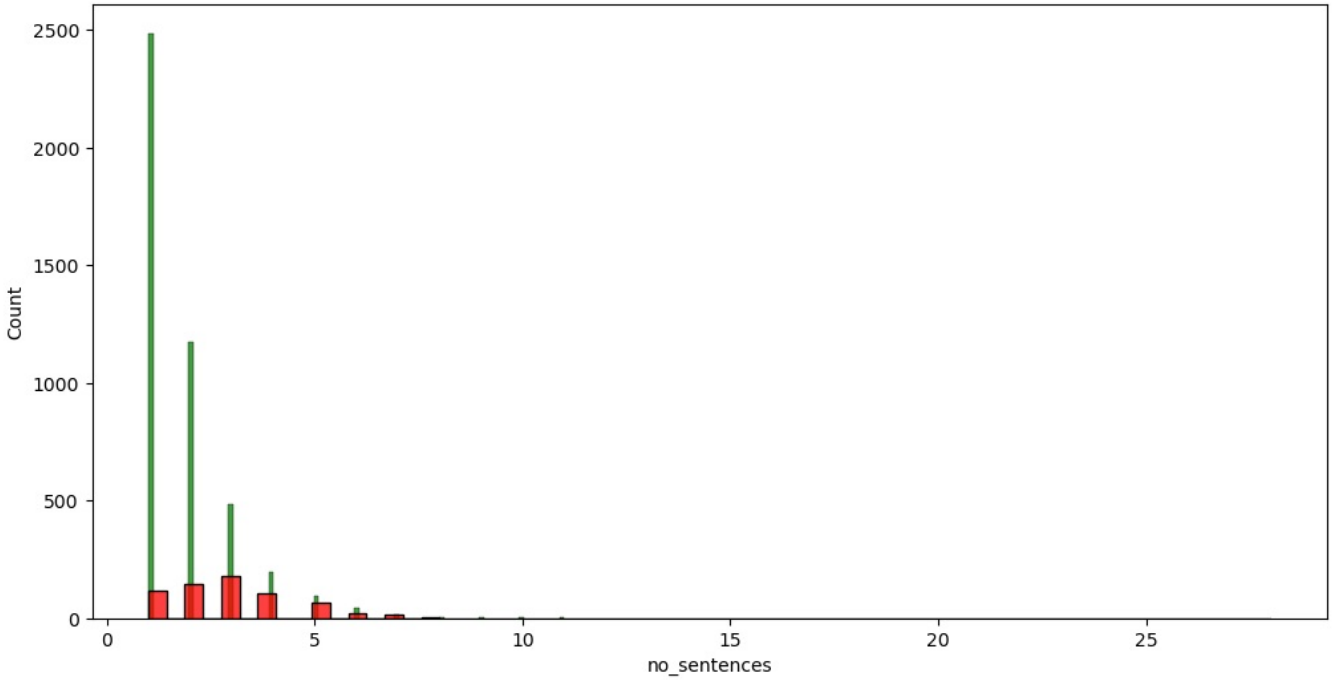
In [151…
```
# Visualization for number of words in ham and spam messages.

plt.figure(figsize=(12,6))
sns.histplot(data[data['Target'] == 0]['no_words'], color = 'green')   # Ham
sns.histplot(data[data['Target'] == 1]['no_words'], color = 'red')   # Spam
```

Out[151]: <AxesSubplot:xlabel='no_words', ylabel='Count'>



In [152…
```
# Visualization for number of sentences in ham and spam messages.

plt.figure(figsize=(12,6))
sns.histplot(data[data['Target'] == 0]['no_sentences'], color = 'green')   # Ham
sns.histplot(data[data['Target'] == 1]['no_sentences'], color = 'red')   # Spam
```

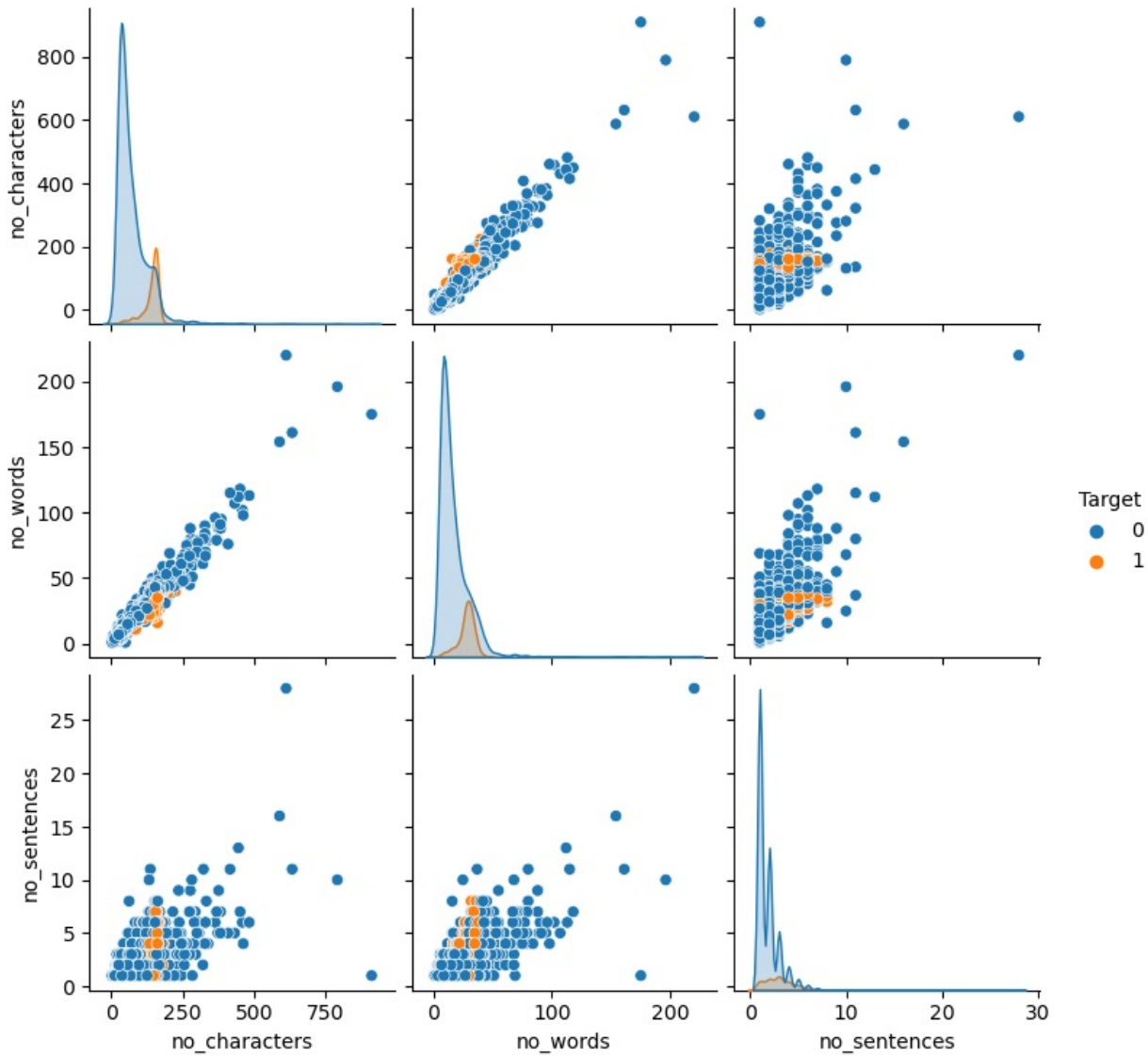
Out[152]: <AxesSubplot:xlabel='no_sentences', ylabel='Count'>



In [153…

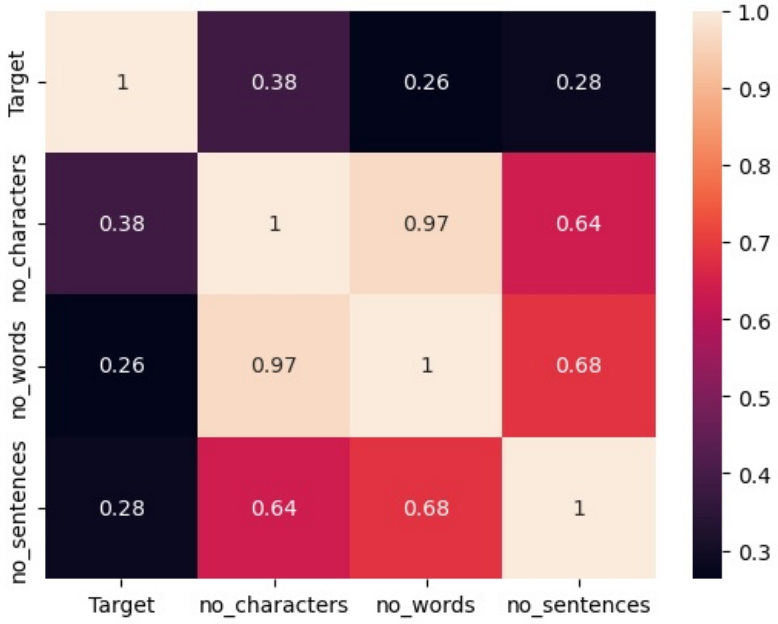
```
# Plotting relationship between number of words, sentences and characters

import seaborn as sns

sns.pairplot(data, hue = 'Target')
```

Out[153]: <seaborn.axisgrid.PairGrid at 0x2ebe3bb7520>



In [154]:
```
sns.heatmap(data.corr(), annot=True)
```

Out[154]: <AxesSubplot:>



3. Data Preprocessing

Lower Case

Tokenization

Removing special characters

Removing stop words and punctuations

Steamming

In [155…
```
import nltk
nltk.download('stopwords')
```

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

Out[155]: True

In [156…
```
from nltk.corpus import stopwords
import string

# Your code that uses stopwords
stop_words = set(stopwords.words('english'))
```

In [157…
```
def transform_text(Text):
    Text = Text.lower()            # Making text lowercase
    Text = nltk.word_tokenize(Text) #

    y = []
    for i in Text:
        if i.isalnum():
            y.append(i)

    Text = y[:]
    y.clear()


    for i in Text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    Text = y[:]
    y.clear()

    for i in Text:
        y.append(ps.stem(i))


    return " ".join(y)
```

In [158…]
```
transform_text('Did you like my presentation on ML')
```

Out[158]:
```
'like present ml'
```

In [159…]
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[159]:
```
'love'
```

In [160…]
```
transform_text('I loved the YT lectures on Machine Learning')
```

Out[160]:
```
'love yt lectur machin learn'
```

In [161…]
```
data['transformed_text'] = data['Text'].apply(transform_text)
```

In [162…]
```
data.head()
```

Out[162]:

	Target	Text	no_characters	no_words	no_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

4. Model Building

We are starting with naive based algorithm as it considered the best fit algorithm for textbased ML problem.

In [163…]
```
pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\7mentor\newanaconda\lib\site-packages (1.3.2)
Requirement already satisfied: joblib>=1.1.1 in c:\7mentor\newanaconda\lib\site-packages (from scikit-learn) (1
.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\7mentor\newanaconda\lib\site-packages (from scikit-lear
n) (1.21.6)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\7mentor\newanaconda\lib\site-packages (from scikit-le
arn) (2.2.0)
Requirement already satisfied: scipy>=1.5.0 in c:\7mentor\newanaconda\lib\site-packages (from scikit-learn) (1.
9.1)
Note: you may need to restart the kernel to use updated packages.
```

In [164…]
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features = 3000)
```

In [165…]
```
x = tfidf.fit_transform(data['transformed_text']).toarray()
```

In [166…]
```
x.shape
```

Out[166]:
```
(5169, 3000)
```

In [167…]
```
y = data['Target'].values
```

In [168…]
```
y
```

Out[168]:
```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

In [169…]
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score
```

In [170…]
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state=2)
```

In [171…]
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
```

In [172…]
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [173…]
```
gnb.fit(x_train, y_train)
y_pred1 = gnb.predict(x_test)
print(accuracy_score(y_test, y_pred1))
print(confusion_matrix(y_test, y_pred1))
```

```
print(precision_score(y_test, y_pred1))
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

In [174…
```
mnb.fit(x_train, y_train)
y_pred2 = mnb.predict(x_test)
print(accuracy_score(y_test, y_pred2))
print(confusion_matrix(y_test, y_pred2))
print(precision_score(y_test, y_pred2))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

In [175…
```
bnb.fit(x_train, y_train)
y_pred3 = bnb.predict(x_test)
print(accuracy_score(y_test, y_pred3))
print(confusion_matrix(y_test, y_pred3))
print(precision_score(y_test, y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

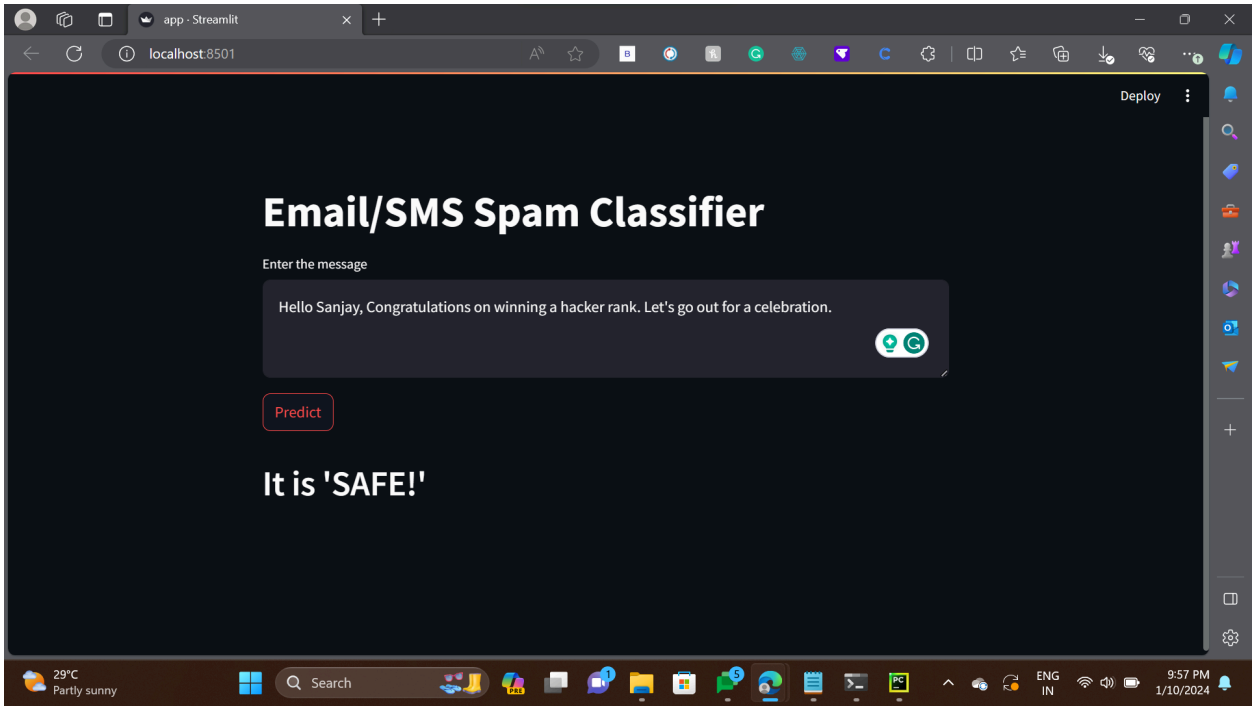
Since, Precision score is best in tfidf, which is important in this problem, We are going with tfidf

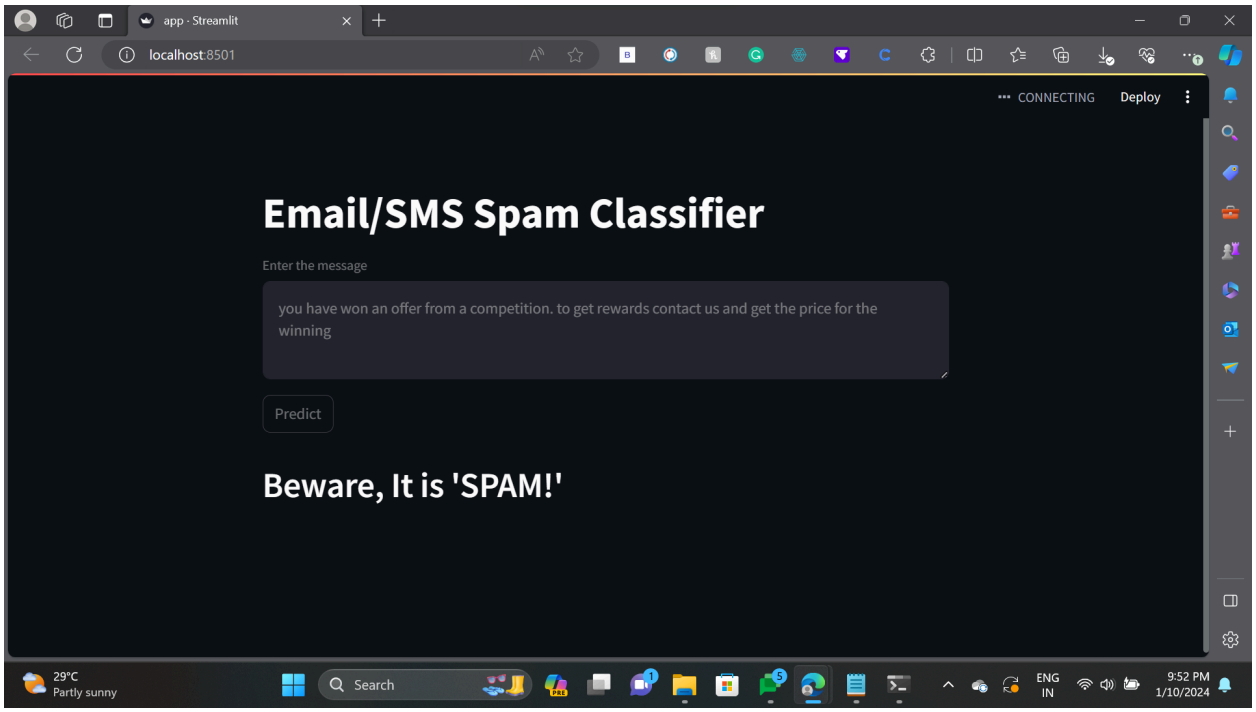
In [176…
```
import pickle
pickle.dump(tfidf, open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl', 'wb'))
```

In [ ]:

```
 1 import streamlit as st
 2 import pickle
 3 import nltk
 4 import string
 5 from nltk.corpus import stopwords
 6 from nltk.stem.porter import PorterStemmer
 7
 8 ps = PorterStemmer()
 9
10
11 def transform_text(Text):
12
13     Text = Text.lower()            # Making text lowercase
14     Text = nltk.word_tokenize(Text) #
15
16     y = []
17     for i in Text:
18         if i.isalnum():
19
20             y.append(i)
21
22     Text = y[:]
23     y.clear()
24
25
26     for i in Text:
27         if i not in stopwords.words('english') and i not in string.punctuation:
28             y.append(i)
29
30     Text = y[:]
31     y.clear()
```

```
32
33      for i in Text:
34          y.append(ps.stem(i))
35
36
37      return " ".join(y)
38
39  tfidf = pickle.load(open('vectorizer.pkl'
    , 'rb'))
40  model = pickle.load(open('model.pkl', 'rb'
    ))
41
42  st.title("Email/SMS Spam Classifier")
43
44  input_sms = st.text_area("Enter the
    message")
45
46  if st.button('Predict'):
47
48      # 1. Preprocess
49      transform_sms = transform_text(
    input_sms)
50      # 2. Vectorize
51      vector_input = tfidf.transform([
    transform_sms])
52      # 3. Predict
53      result = model.predict(vector_input)[0
    ]
54      # 4. Display
55      if result == 1:
56          st.header("Beware, It is 'SPAM!' "
    )
57      else:
```


app - Streamlit
localhost:8501
Deploy
Email/SMS Spam Classifier
Enter the message
Hello Sanjay, Congratulations on winning a hacker rank. Let's go out for a celebration.
Predict
It is 'SAFE!'
29°C
Partly sunny
Search
ENG
IN
9:57 PM
1/10/2024


app - Streamlit
localhost:8501
CONNECTING
Deploy
Email/SMS Spam Classifier
Enter the message
you have won an offer from a competition. to get rewards contact us and get the price for the winning
Predict
Beware, It is 'SPAM!'
29°C
Partly sunny
Search
ENG
IN
9:52 PM
1/10/2024

Architecture:

The architecture of the SMS/Email Spam Classifier project involves a seamless integration of various tools and technologies. The Jupyter Notebook is used for data exploration and preprocessing, PyCharm for web application development, and Streamlit for creating the user interface. The trained Naive Bayes model forms the core of the application, providing accurate predictions based on the input text.

Acknowledgements:

This project wouldn't have been possible without the support and guidance from various sources. YouTube tutorials provided valuable insights into the implementation of machine learning algorithms, while ChatGPT played a crucial role in answering queries and providing assistance throughout the project. Special thanks to experts who helped in understanding the code and resolving errors encountered during the development process.

Conclusion:

The SMS/Email Spam Classifier project successfully addresses the challenge of spam identification in text messages and emails. With a well-trained Naive Bayes model and an intuitive web application, users can now easily determine whether incoming messages are spam or legitimate. The deployment on a cloud platform enhances accessibility, making the solution practical and scalable for real-world applications. This project serves as a testament to the power of machine learning in solving everyday communication challenges.