Title Page

SMS SPAM CLASSIFIER

Name : Kartavya Master Divyakant

Project Overview

1. Enhance communication efficiency by filtering out spam messages, ensuring users receive only relevant and legitimate content.
2. Mitigate security risks associated with spam, such as phishing attacks and malware dissemination, safeguarding users' personal information and privacy.
3. Improve overall user experience and trust in SMS communication channels by providing a reliable spam detection mechanism.

Steps Performed in Project

1. Data Cleaning
2. EDA
3. Text Preprocessing
4. Model Building
5. Improvement

Libraries used : Pandas
Matplotlib , Seaborn
numpy
NLTK
scikit learn

Title Page | Reporters | 1 .Data cleaning

Removing Duplicates and Null values

1. Data overview

```
In [3]: df.sample(5)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
2919	ham	Thanx 4 the time we��ve spent 2geva, its bin m...	NaN	NaN	NaN
1491	spam	Your account has been credited with 500 FREE T...	NaN	NaN	NaN
1203	ham	Thanks for understanding. I've been trying to ...	NaN	NaN	NaN
5041	spam	Jamster! To get your free wallpaper text HEART...	NaN	NaN	NaN
1741	ham	I can do that! I want to please you both insid...	NaN	NaN	NaN

2. Removing the unnecessary columns

```
In [6]: df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

```
In [7]: df.sample(4)
```

Out[7]:

	v1	v2
1056	ham	Then u drive lor.
913	ham	Ok lor but not too early. Me still having proj...
3980	ham	Huh i cant thk of more oredi how many pages do...
4800	ham	The guy at the car shop who was flirting with ...

3. Removing the Null rows as number of NULL rows are less

```
In [11]: #Checking for duplicate value and data has no missing value anymore...
         df.duplicated().sum()
```

Out[11]: 403

```
In [12]: df.drop_duplicates(inplace=True)
```

```
In [13]: df.shape
```

Out[13]: (5169, 2)

4. Labeling Column

```
In [8]: # We will change the column name for better understanding...

        df=df.rename(columns={'v1':'target','v2':'sms'})
```

5. Labeling Output Column

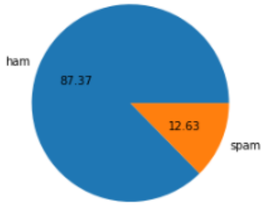
```
In [9]: #Chaning the target column values into numeric form

        from sklearn.preprocessing import LabelEncoder
```

```
In [10]: lb = LabelEncoder()

         df['target'] = lb.fit_transform(df['target'])
```

Visualizing the dataset and exploring Insights

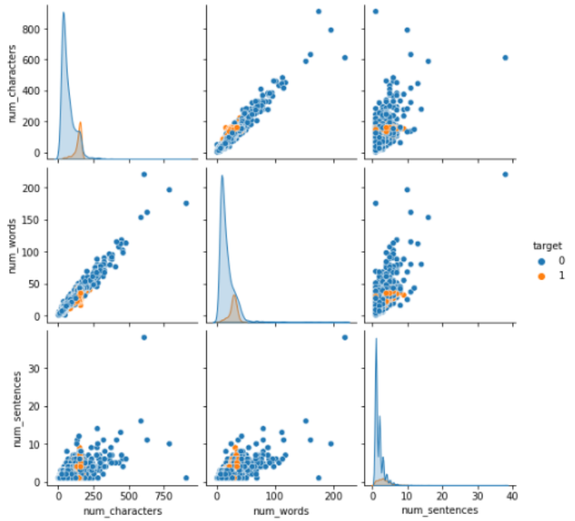
1. Ratio of Target values

```
In [17]: plt.pie(df['target'].value_counts(),labels=['ham','spam'],autopct='%.2f')
         plt.show()

         #DATA is imbalance
```



87.37% msgs are not Spam

```
In [34]: sns.pairplot(df,hue='target')
Out[34]: <seaborn.axisgrid.PairGrid at 0x2040666f3d0>
```



2. tokenizing the word , numbers for further process

Out[23]:

	target	sms	num_characters
691	0	Sorry to trouble u again. Can buy 4d for my da...	109
72	0	HI BABE IM AT HOME NOW WANNA DO SOMETHING? XX	45

```
In [24]: df['num_words'] = df['sms'].apply(lambda x : nltk.word_tokenize(x)).apply(len)
In [25]: df['num_sentences'] = df['sms'].apply(lambda x : nltk.sent_tokenize(x)).apply(len)
In [26]: df
```

Out[26]:

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

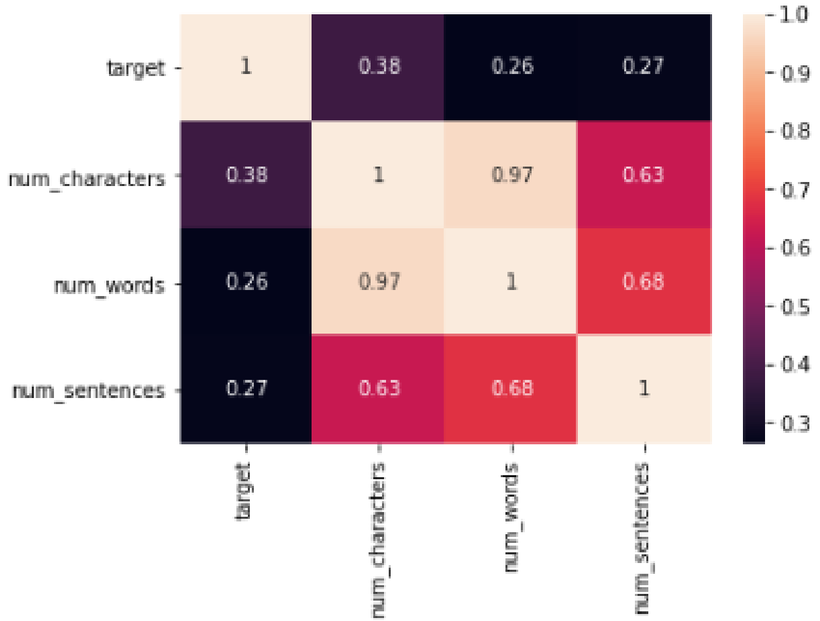
3. Pairplot of 3 columns (num_char, Num_words, num_sentences)

- here we can notice that num_characters are giving little linear observation.

Visualizing the dataset and exploring Insights

4. Heatmap

```
In [35]: sns.heatmap(df.corr(),annot=True)

         # Through this heatmap we can assume that num_character gives us more clear scnerio for the target column and has
         # good co-relation with other column.

Out[35]: <AxesSubplot:>
```



- The variable "num_characters" exhibits a correlation of 0.38 with the target column.
- It also demonstrates a strong correlation with other columns.

Working to data to build model

1. Reviewing **punctuation** in order to eliminate **unnecessary instances** from the text.

```
In [36]: !pip install string

ERROR: Could not find a version that satisfies the requirement string
ERROR: No matching distribution found for string

In [37]: import string
         from string import punctuation
         punctuation

Out[37]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

2. Detecting **stop words** to eliminate unnecessary words that **do not contribute** to model development.

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "yo
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herse
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'th
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'f
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up
```

3. **Stemming** the words to reduce words to their root form. (Shown in Ex.)

```
In [39]: from nltk.stem import PorterStemmer

         ps = PorterStemmer()
         ps.stem('rising')

Out[39]: 'rise'
```

Working to data to build model

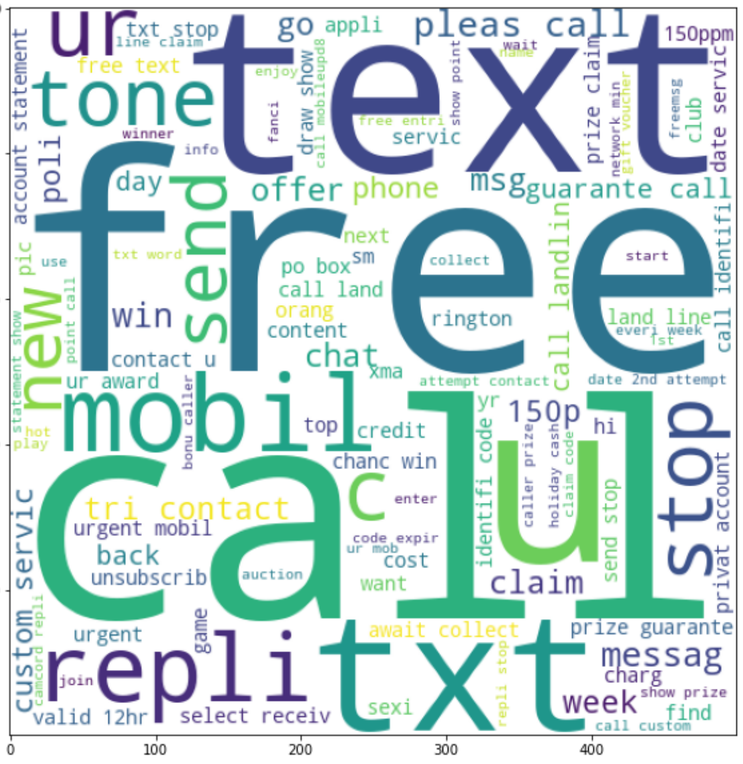
4. Creating a **Data Preprocessing function** with the role of transforming text to lowercase, tokenizing, removing special characters, eliminating stop words, and stemming the sentences.

In [184]:
```
def Datapreprocessing(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y=[]

    for i in text:
        if i.isalnum():
            y.append(i)
    text = y.copy()
    y.clear()

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

    text = y.copy()
    y.clear()

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

    text = y[:]
    y.clear()

    text = " ".join(text)
    return text
```

Working to data to build model

5. Creating **WordCloud** to detect frequently used word in **Spam** and **Ham** messages.

```
In [51]: wc_spam = wc.generate(df[df['target'] == 1]['transformed_sms'].str.cat(sep=" "))

In [52]: plt.figure(figsize=(10,10))
         plt.imshow(wc_spam)

Out[52]: <matplotlib.image.AxesImage at 0x204082ef2e0>
```



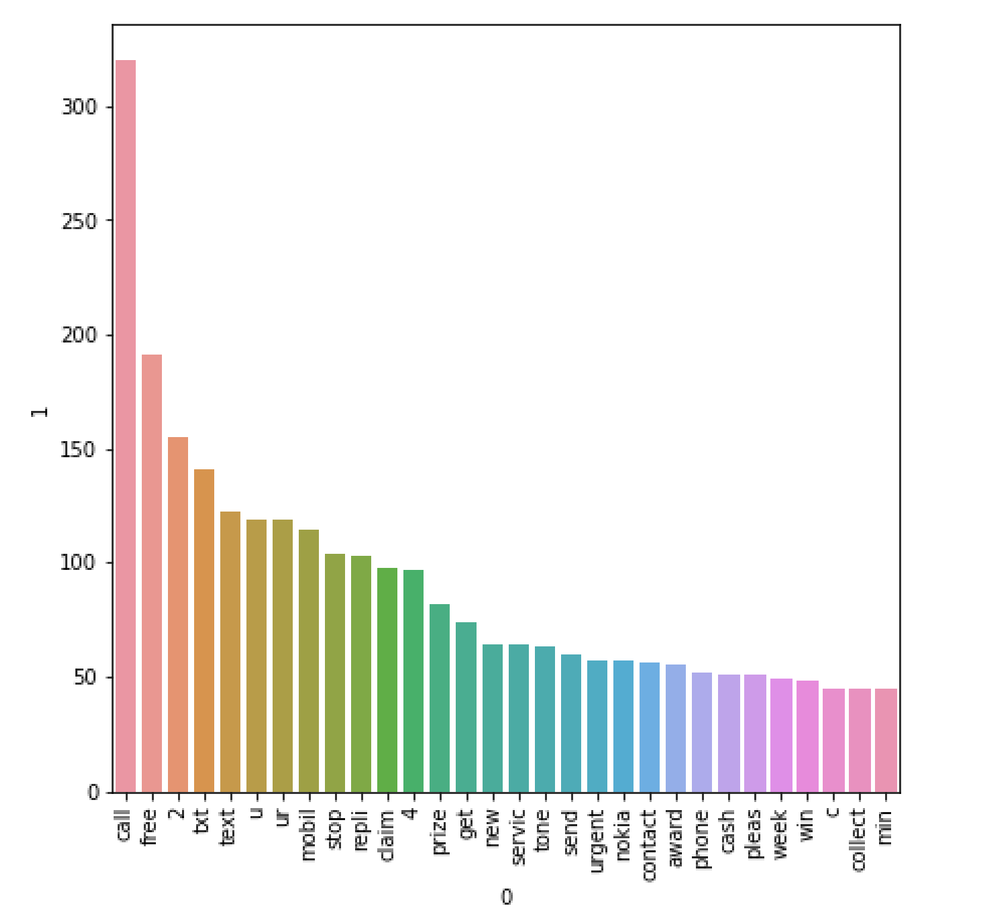
SPAM

```
In [53]: wc_ham = wc.generate(df[df['target'] == 0]['transformed_sms'].str.cat(sep=" "))

In [54]: plt.figure(figsize=(10,10))
         plt.imshow(wc_ham)

Out[54]: <matplotlib.image.AxesImage at 0x204063db4f0>
```

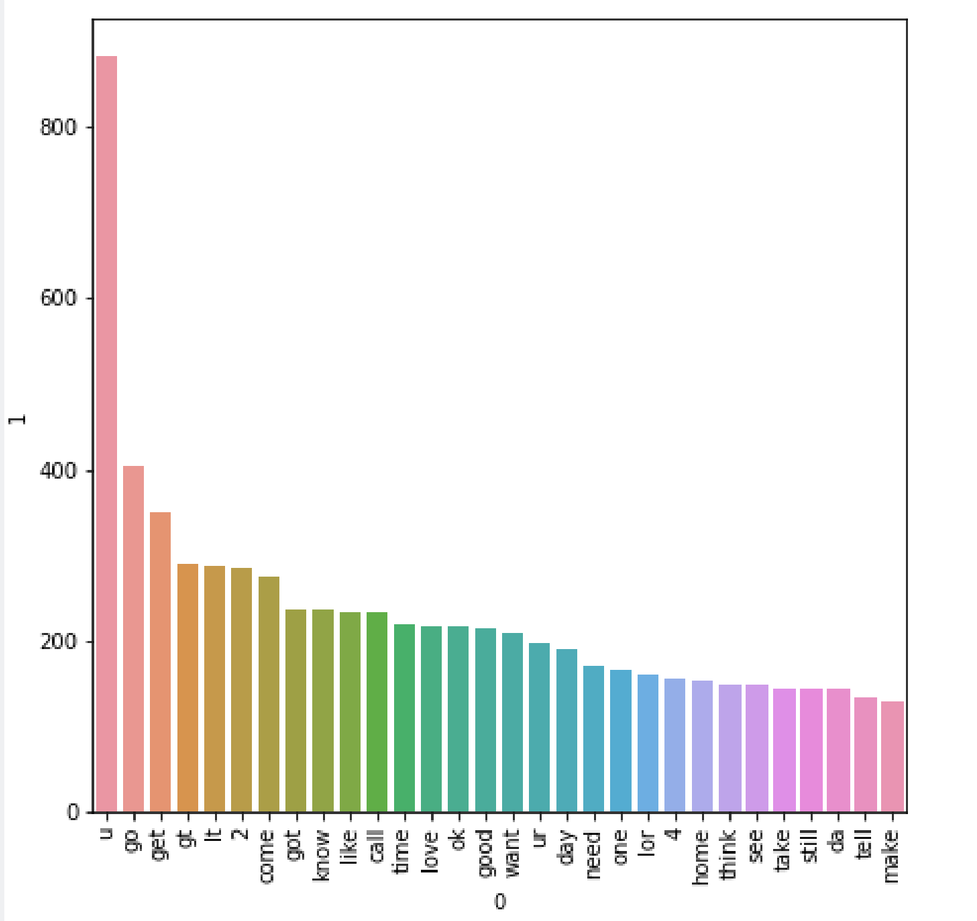
HAM

Working to data to build model

6. **Visualizing** the number of times word accrued in **Spam** and **Ham** messages.



SPAM



HAM

Reviewing the data across various models.

1. Applied multiple Algorithmd on dataset and record the **Precision score** and **Accuracy score.**
ex.(GaussianNB,MultinomialNB,BernoulliNB)

```
gnb.fit(X_train,y_train)
y_pred_gnb = gnb.predict(X_test)
print('Accuracy Score : ',accuracy_score(y_test,y_pred_gnb))
print('Accuracy Score : ',confusion_matrix(y_test,y_pred_gnb))
print('Precision Score : ',precision_score(y_test,y_pred_gnb))
```

```
Accuracy Score :  0.8704061895551257
Accuracy Score :  [[788 108]
 [ 26 112]]
Precision Score :  0.509090909090909
```

```
mnb.fit(X_train,y_train)
y_pred_mnb = mnb.predict(X_test)
print('Accuracy Score : ',accuracy_score(y_test,y_pred_mnb))
print('Accuracy Score : ',confusion_matrix(y_test,y_pred_mnb))
print('Precision Score : ',precision_score(y_test,y_pred_mnb))
```

```
Accuracy Score :  0.971953578336557
Accuracy Score :  [[896   0]
 [ 29 109]]
Precision Score :  1.0
```

```
bnb.fit(X_train,y_train)
y_pred_bnb = bnb.predict(X_test)
print('Accuracy Score : ',accuracy_score(y_test,y_pred_bnb))
print('Accuracy Score : ',confusion_matrix(y_test,y_pred_bnb))
print('Precision Score : ',precision_score(y_test,y_pred_bnb))
```

```
Accuracy Score :  0.9835589941972921
Accuracy Score :  [[895   1]
 [ 16 122]]
Precision Score :  0.991869918699187
```

- The **Bernoulli Naive Bayes** model showcases exceptional performance
- Evidenced by an impressive Accuracy Score of 0.98 and a Precision Score of 0.99.

Engaged in refining algorithm parameters.

2. Implemented additional **renowned algorithms** on the dataset and **recorded** both the precision and accuracy scores.

```
performance_df
```

	Algorithm	Accuracy	Precision
1	KN	0.900387	1.000000
2	NB	0.959381	1.000000
8	ETC	0.977756	0.991453
5	RF	0.970019	0.990826
0	SVC	0.972921	0.974138
10	xgb	0.971954	0.957983
6	AdaBoost	0.962282	0.954128
4	LR	0.951644	0.940000
9	GBDT	0.951644	0.931373
7	BgC	0.957447	0.861538
3	DT	0.934236	0.830189

3. At the **final stage, hyperparameter** tuning is essential in determining the **optimal algorithm** with well-suited hyperparameters.

```
new_df
```

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000	Precision_max_ft_3000
0	KN	0.900387	1.000000	0.905222	1.000000
1	NB	0.959381	1.000000	0.971954	1.000000
2	ETC	0.977756	0.991453	0.979691	0.975610
3	RF	0.970019	0.990826	0.975822	0.982906
4	SVC	0.972921	0.974138	0.974855	0.974576
5	xgb	0.971954	0.957983	0.968085	0.941176
6	AdaBoost	0.962282	0.954128	0.961315	0.945455
7	LR	0.951644	0.940000	0.956480	0.969697
8	GBDT	0.951644	0.931373	0.946809	0.927835
9	BgC	0.957447	0.861538	0.959381	0.869231
10	DT	0.934236	0.830189	0.930368	0.823529

- In this context, NB (Naive Bayes) serves as the optimal model.

Engaged in refining algorithm parameters.

4. Using **Voting Classifier** to Combine the multiple models and checking the Accuracy and Precision Score

```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
rf = RandomForestClassifier
kn = KNeighborsClassifier()
from sklearn.ensemble import VotingClassifier
```

```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc),('kn',kn)],voting='soft')
```

```
voting.fit(X_train,y_train)
```

```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2)),
                             ('kn', KNeighborsClassifier())],
                 voting='soft')
```

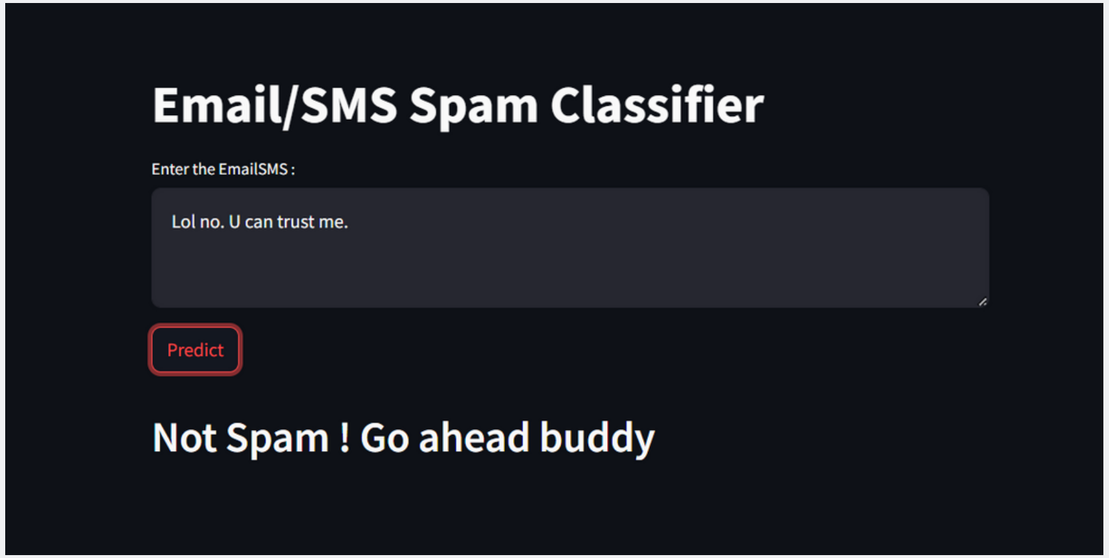
```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

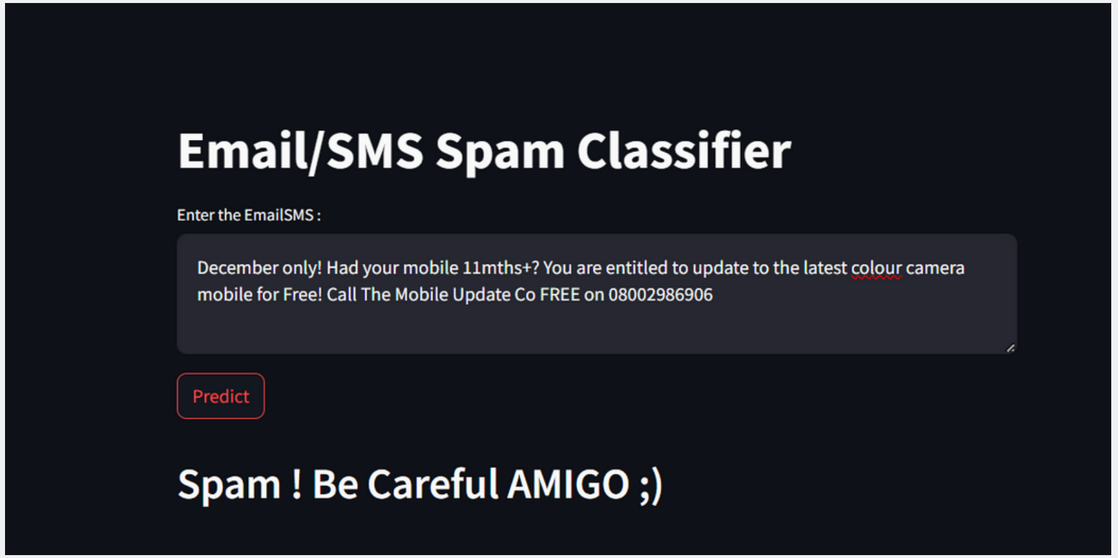
```
Accuracy 0.9738878143133463
Precision 1.0
```

- After **tuning** the training of the **Voting classifier**, we have achieved an Accuracy Score of 0.97 and a Precision Score of 1.0. Consequently, we are opting to utilize the Voting Classifier, comprising **SVM, NB, ET,** and **KN** models.

The model is prepared.

Final project sneak peek!





All necessary project files have been uploaded to the GitHub repository.


Thank you
www.thankyou.com
THANK
YOU!