SMS SPAM CLASSIFIER

A Project Report Submitted
in Partial Fulfillment of the Requirements

International Institute of Information Technology, Bhubaneswar
2ND YEAR B. Tech

In

COMPUTER ENGINEERING

By:

1. **APURBA RANJAN** **ROLL NO: B522008**
2. **DEBASHISH SAHOO** **ROLL NO: B522019**
3. **PRIYARANJAN KUMAR** **ROLL NO: B522046**
4. **RANJAN KUMAR** **ROLL NO: B522048**
5. **SWATA SWAYAM DAS** **ROLL NO: B522060**

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY, BHUBANESWAR

[APRIL-2024]

UNDERTAKING

We, the undersigned members of the group undertaking the project titled "SMS Spam Classifier" for our B. Tech in Computer Engineering , 2nd Year, at the International Institute of Information Technology, Bhubaneswar, hereby declare that the project report submitted by us is our original work and has not been submitted elsewhere for any other purpose.

We further declare that:

1. The project work was carried out under the guidance of Dwibik Patra , Python Professor , Computer Science.

2. All sources of information used have been duly acknowledged through proper citations and references.

3. No part of this project violates or infringes upon the rights of any third party, including but not limited to copyright, trademark, privacy, or other personal or proprietary rights.

4. Any data, code, or materials obtained or used from external sources have been appropriately credited and referenced.

We understand that any violation of the above declarations may result in disciplinary action as per the institute's rules and regulations.

Date: 13 April 2024

Signatures:

1. Apurba Ranjan - Roll No: B522008

2. Debashish Sahoo - Roll No: B522019

3. Priyaranjan Kumar - Roll No: B522046

4. Ranjan Kumar - Roll No: B522048

5. Swata Swayam Das - Roll No: B522060

Acknowledgements

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project.

First and foremost, we would like to thank Dwibik Patra, Python Professor , Computer Science for their invaluable guidance, support, and encouragement throughout the duration of this project. Their expertise and insights have been instrumental in shaping our approach and guiding us through the various stages of the project.

We would also like to extend our thanks to the faculty members of the Department of Computer Science at the International Institute of Information Technology, Bhubaneswar, for their valuable inputs and feedback.

We are grateful to [Name of Dataset Provider/Source] for providing the dataset used in this project. Their contribution has been vital to the success of our research.

Our sincere thanks to our classmates and friends for their support and encouragement throughout this endeavor.

Lastly, we would like to express our heartfelt gratitude to our families for their unwavering support, understanding, and patience during the course of this project.

1. APURBA RANJAN ROLL NO: B522008
2. DEBASHISH SAHOO ROLL NO: B522019
3. PRIYARANJAN KUMAR ROLL NO: B522046
4. RANJAN KUMAR ROLL NO: B522048
5. SWATA SWAYAM DAS ROLL NO: B522060

Table of Contents

Introduction of the Project

Our team of B. Tech 2nd year students at the International Institute of Information Technology, Bhubaneswar, has embarked on a project to develop an SMS spam classifier. The objective of this project is to create a machine learning model capable of accurately identifying spam messages in SMS communication.

We express our gratitude to our subject professor , Mr. Dwibik Patra, for his guidance and support throughout the project. Additionally, we thank our peers for their collaboration and assistance.

The project contains-

1. Data Cleaning
2. Exploratory Data Analysis (EDA)
3. Text Preprocessing
4. Model Building
5. Evaluation
6. Improvement
7. Website Development
8. Deployment

Our project roadmap encompasses key stages, including data cleaning, exploratory data analysis (EDA), text preprocessing, model building, evaluation, improvement, website development, and deployment. Each stage plays a crucial role in the development and refinement of our SMS spam classifier, ultimately leading to a robust and effective solution.

Throughout this project, we are grateful for the guidance and support of our subject professor, Mr. Dwibik Patra, whose expertise has been instrumental in navigating the complexities of machine learning and natural language processing. Additionally, we extend our thanks to our peers for their collaboration and assistance, as well as to the academic community for providing valuable resources and insights.

As we embark on this journey, we are excited about the opportunities this project presents to deepen our understanding of machine learning and contribute to the development of innovative solutions in the field of

communication technology. We look forward to sharing our progress and findings with our peers, educators, and the wider community.

PROCESS

Our project follows a systematic approach, encompassing several key stages to ensure the successful development and deployment of our SMS spam classifier. Below is an outline of the process we will undertake:

Data Cleaning:

We begin by cleaning the SMS dataset to ensure consistency and remove any inconsistencies or errors that may affect the performance of our model. This involves handling missing values, removing duplicates, and formatting the data for further analysis.

Exploratory Data Analysis (EDA):

With the cleaned dataset, we conduct exploratory data analysis to gain insights into the distribution and characteristics of spam and non-spam messages. This helps us understand the underlying patterns and features that distinguish spam from legitimate messages.

Text Preprocessing:

Before building our model, we preprocess the text data to prepare it for analysis. This involves converting text to lowercase, tokenizing, removing special characters, stopwords, and stemming or lemmatizing words to reduce dimensionality and improve model performance.

Model Building:

Using the preprocessed data, we train various machine learning models to classify SMS messages as spam or non-spam. We experiment with different algorithms such as Naive Bayes, Support Vector Machines (SVM), and ensemble methods to identify the most effective model for our task.

Evaluation:

Once we have trained our models, we evaluate their performance using appropriate metrics such as accuracy, precision, recall, and F1-score. This allows us to assess the effectiveness of each model and select the best-performing one for deployment.
Improvement:

Based on the evaluation results, we iteratively refine our model by fine-tuning hyperparameters, experimenting with different feature engineering techniques, and exploring alternative algorithms. This continuous improvement process helps us optimize the performance of our classifier.

Website Development:

To make our SMS spam classifier accessible to users, we develop a user-friendly website where users can input text messages and receive instant feedback on whether they are spam or not. The website will also provide information about the project and how the classifier works.

Deployment:

Finally, we deploy our trained model and website to a production environment, making it available for real-world use. This involves setting up servers, deploying the website, and ensuring that the classifier performs reliably and efficiently in a live environment.

By following this structured process, we aim to develop a highly accurate and reliable SMS spam classifier that can help users protect themselves from unwanted and potentially harmful messages.

System Requirements of the Project

Recommended System Requirements

Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz.

Disk space: 2 to 4 GB.

Operating systems: Windows® 10, MACOS, and UBUNTU.

Python Versions: 3.X.X or Higher.

Minimum System Requirements

Processors: Intel Atom® processor or Intel® Core™ i5 processor.

Disk space: 4 GB.

Operating systems: Windows 10 or later, MACOS, and UBUNTU.

Python Versions: 2.7.X, 3.6.X.

Prerequisites before installing MySQL Connector Python

You need root or administrator privileges to perform the installation process.

Python must be installed on your machine.

Note: – MySQL Connector Python requires python to be in the system's PATH. Installation fails if it doesn't find Python.

On Windows, If Python doesn't exist in the system's PATH, please manually add the directory containing python.exe yourself.

Source Code:-

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('spam.csv', encoding='ISO-8859-1')
```

In [3]:
```
df.head()
```

Out[3]:

	v1	v2	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
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [4]:
```
df.shape
```

Out[4]:
```
(5572, 5)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
```

DATA CLEANING

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

TO RENAME COLUMN NAMES

In [7]:
```
df.rename(columns={'v1':'Res','v2':'Text'},inplace=True)
```

In [8]:
```
df.head()
```

Out[8]:

	Res	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 [9]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

In [10]:
```
df['Res']=label_encoder.fit_transform(df['Res'])
```

In [11]:
```
df.sample(4)
```

Out[11]:

	Res	Text
4848	0	either way works for me. I am <#> year...
337	0	Just sleeping..and surfing

In [11]:
```
df.sample(4)
```

Out[11]:

	Res	Text
4848	0	either way works for me. I am <#> year...
337	0	Just sleeping..and surfing
4319	0	Hey mr and I are going to the sea view and ha...
3773	0	Ok... But bag again..

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
403
```

Removing Duplicates

In [13]:
```
df=df.drop_duplicates(keep='first')
```

In [14]:
```
df.shape
```

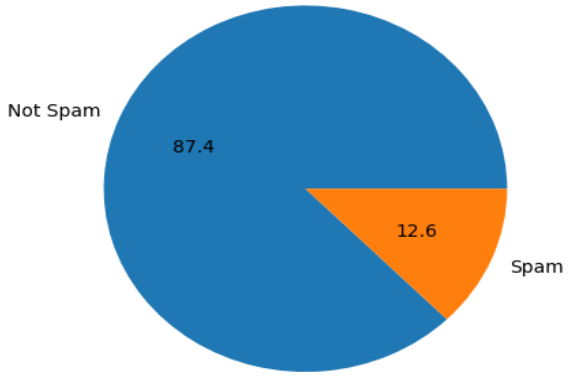
Out[14]:
```
(5169, 2)
```

EXPLORATORY DATA ANLYSIS

In [15]:
```
df['Res'].value_counts()
```

Out[15]:
```
Res
0    4516
1     653
Name: count, dtype: int64
```

In [16]:
```
plt.pie(df['Res'].value_counts(),labels=['Not Spam','Spam'],autopct="%0.1f")
plt.show()
```



MAKING THREE NEW COLUMNS FOR NO> OF CHARACTERS , WORDS AND SENTENCES USING NLTK LIBRARY OF PYTHON

In [17]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\debas\anaconda3\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\users\debas\anaconda3\lib\site-packages (from nltk) (8.0.4)
```

In [20]:
```
df['chars']=df['Text'].apply(len)
```

In [21]:
```
df.sample(3)
```

Out[21]:

	Res	Text	chars
3725	0	No chikku nt yet.. Ya i'm free	30
1781	0	;-(oh well, c u later	22
2303	0	Should I tell my friend not to come round til ...	66

In [22]:
```
df['words']=df['Text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [23]:
```
df.sample(3)
```

Out[23]:

	Res	Text	chars	words
4695	0	A guy who gets used but is too dumb to realize...	50	13
3146	0	Oh thats late! Well have a good night and i wi...	113	27
2006	0	Shopping lor. Them raining mah hard 2 leave or...	52	11

In [24]:
```
df['sen']=df['Text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [25]:
```
df.sample(3)
```

Out[25]:

	Res	Text	chars	words	sen
477	0	Tension ah?what machi?any problem?	34	9	1
4978	0	Spending new years with my brother and his fam...	111	26	4
915	0	I could ask carlos if we could get more if any...	67	15	1

In [26]:
```
df.describe()
```

Out[26]:

	Res	chars	words	sen
count	5169.000000	5169.000000	5169.000000	5169.000000
mean	0.126330	78.977945	18.455794	1.965564
std	0.332253	58.236293	13.324758	1.448541
min	0.000000	2.000000	1.000000	1.000000
25%	0.000000	36.000000	9.000000	1.000000
50%	0.000000	60.000000	15.000000	1.000000
75%	0.000000	117.000000	26.000000	2.000000
max	1.000000	910.000000	220.000000	38.000000

In [27]:
```
df[df['Res']==0][['chars','words','sen']].describe()
```

Out[27]:

	chars	words	sen
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
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	38.000000

In [28]:
```
df[df['Res']==1][['chars','words','sen']].describe()
```

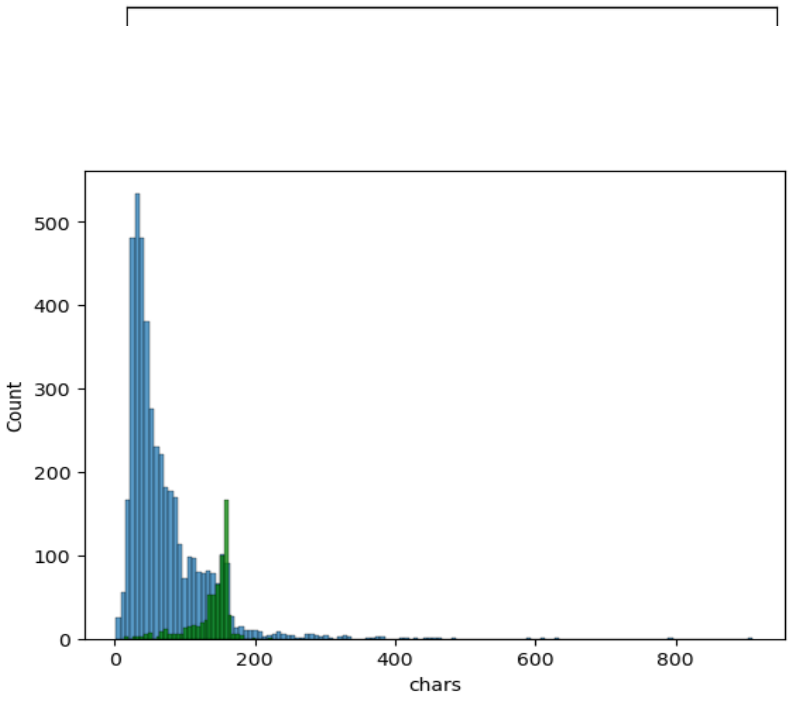
Out[28]:

	chars	words	sen
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
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	9.000000

In [29]:
```
import seaborn as sns
```

In [30]:
```
sns.histplot(df[df['Res']==0]['chars'])#FOR NOT SPAM
sns.histplot(df[df['Res']==1]['chars'],color='green')#FOR  SPAM
```

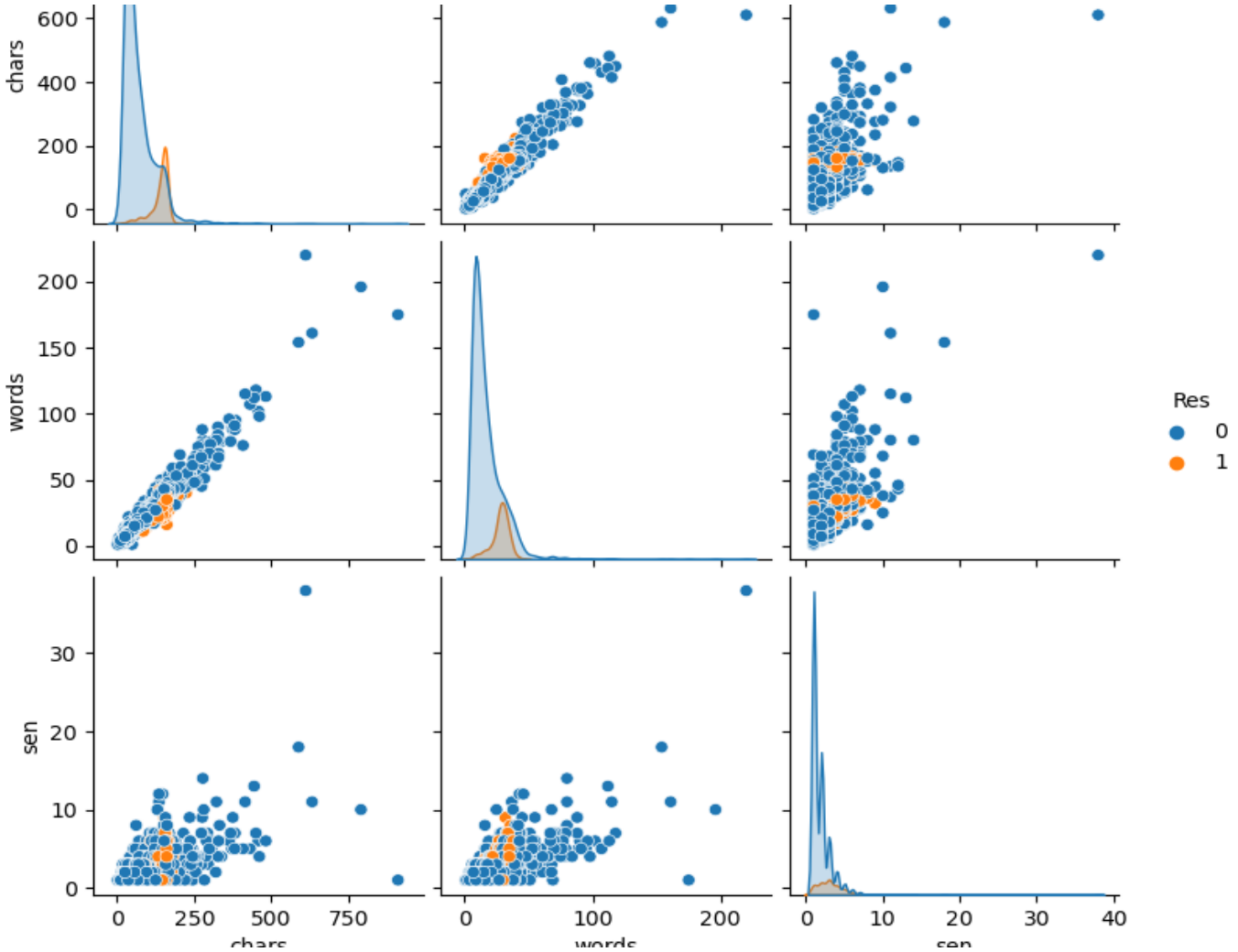
Out[30]: <Axes: xlabel='chars', ylabel='Count'>



[31]:
```
df.drop(columns=('Text')).corr()['Res']
```

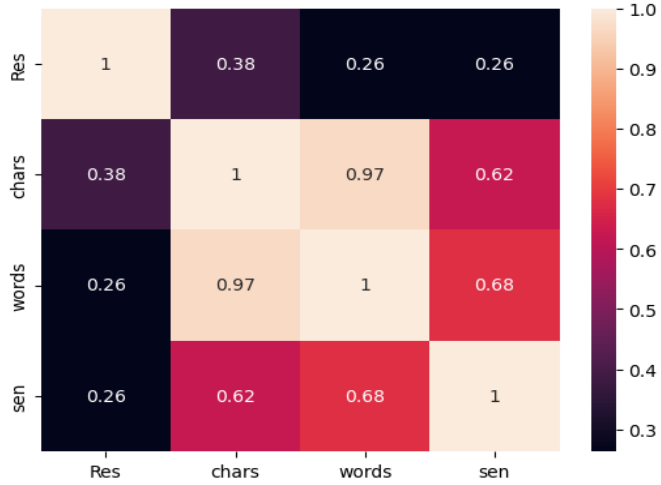
Out[31]:
```
Res      1.000000
chars    0.384717
words    0.262912
sen      0.263939
Name: Res, dtype: float64
```

[32]:
```
sns.pairplot(df,hue='Res')
```



In [33]:
```
#sns.heatmap(df.corr(),annot=True)
sns.heatmap(df.drop(columns=['Text'],inplace=False).corr(),annot=True)
```

Out[33]: <Axes: >



AS THE CHARACTER CORRELATION WITH THE RESULT IS 0.38 THATS THE MOST IMPORTANT PARAMETER IN THIS CASE HERE

DATA PREPROCESSING

In [34]:
```
#LOWER CASING THE TEXT and many other required transformations
```

In [35]:
```
nltk.download('stopwords')
```

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

Out[35]: True

In [36]:
```
import string
string .punctuation
```

Out[36]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

In [37]:
```
from nltk.corpus import stopwords
stopwords.words('english')
```

```
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
```

In [38]:
```
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
ps.stem('cooking')
```

Out[38]: 'cook'

In [39]:
```
def transform(text):
  text=text.lower()
  text=nltk.word_tokenize(text)

  y=[]#FOR REMOVING SPECIAL CHARACTERS
  for i in text:
    if i.isalnum():
      y.append(i)

  text=y[:]
  y.clear()

  for i in text:#FOR REMOVING STOP WORDS AND PUNCTUATION
    if i not in stopwords.words('english') and i not in string.punctuation:
      y.append(i)

  #FOR STEMMING THAT IS CONSIDERING WORD ASV ITS ROOT FORM LIKE COOKING TO COOK COOKS TO COOK
  text=y[:]
  y.clear()

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

  return " ".join(y)

  return y
```

In [40]:
```
transform(
    'Hi how are you? I was Thinking of going to a party tomorrow with Rohit and Krish'
)
```

In [42]:
```
df.sample(5)
```

Out[42]:

	Res	Text	chars	words	sen	Transformed_Text
5219	0	Pls she needs to dat slowly or she will vomit ...	51	12	1	pl need dat slowli vomit
804	0	K I'll be there before 4.	25	8	1	k 4
890	0	Why do you ask princess?	24	6	1	ask princess
3628	0	Should I head straight there or what	36	7	1	head straight
2855	0	Japanese Proverb: If one Can do it, U too Can ...	276	77	4	japanes proverb one u none u must indian versi...

In [ ]:
```
```

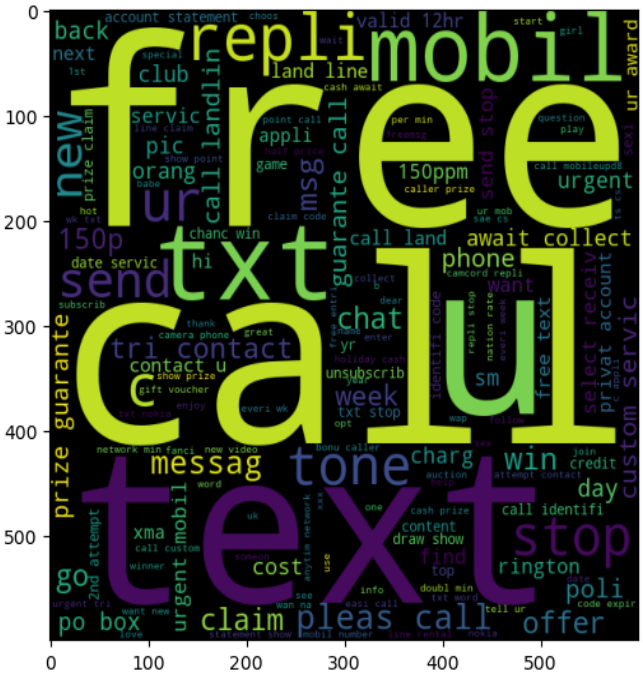
FORMING WORDCLOUD OF SPAM AND NOT SPAM USING PREINSTALLED WORDCLOUD MODULE

Type *Markdown* and LaTeX: α^2

In [43]:
```
from wordcloud import WordCloud
wc=WordCloud(width=600,height=600,min_font_size=10,background_color='black')
```

In [44]:
```
spam_wc=wc.generate(df[df['Res']==1]['Transformed_Text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

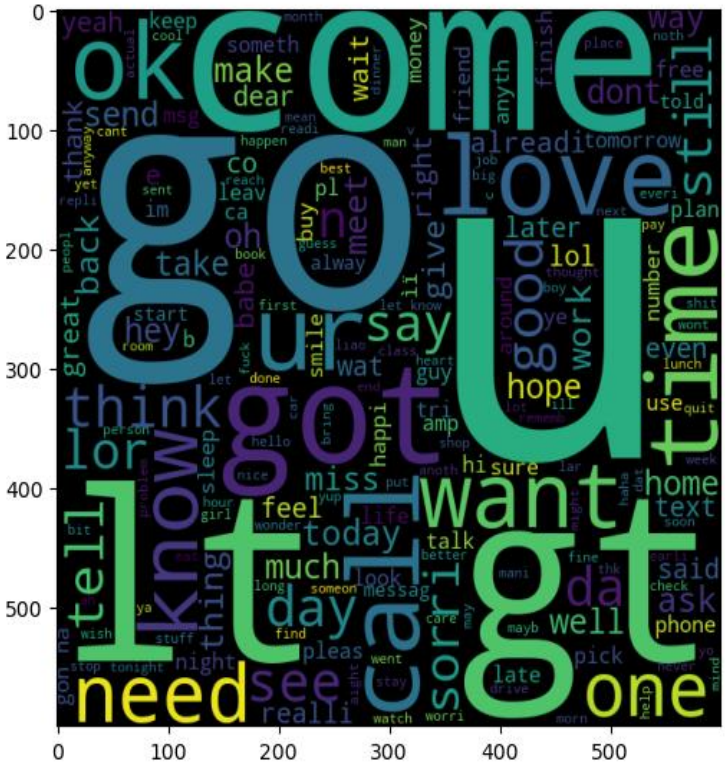
Out[44]: <matplotlib.figure.AxesImage at 0x1d3a0ab4cd0>



In [45]:
```
nspam_wc=wc.generate(df[df['Res']==0]['Transformed_Text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(nspam_wc)
```

Out[45]: <matplotlib.figure.AxesImage at 0x1d3a0ae70d0>

Out[45]: <matplotlib.image.AxesImage at 0x1b5a0ae7d60>



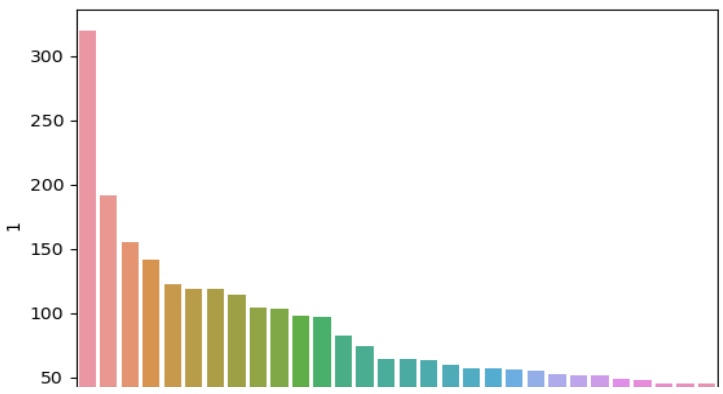
FINDING THE TOP 30 WORDS OF SPAM AND NOT SPAM

In [46]:
```
spam_w = []
for msg in df[df['Res'] == 1]['Transformed_Text'].tolist():
    for word in msg.split():
        spam_w.append(word)
```

In [47]:
```
len(spam_w)
```

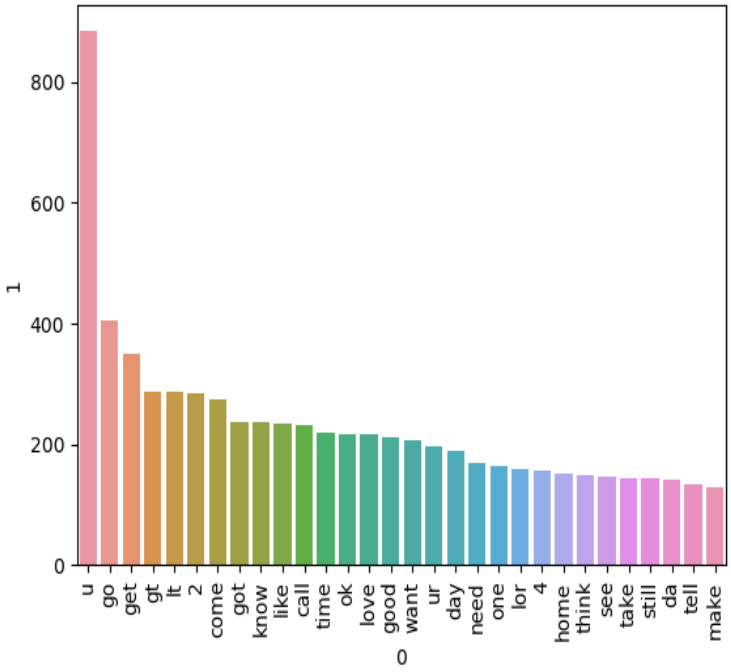
Out[47]: 9939

In [48]:
```
from collections import Counter
sns.barplot(x=pd.DataFrame(Counter(spam_w).most_common(30))[0],y=pd.DataFrame(Counter(spam_w).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```



In [49]:
```
nspam_w = []
for msg in df[df['Res'] == 0]['Transformed_Text'].tolist():
    for word in msg.split():
        nspam_w.append(word)
```

In [50]:
```
from collections import Counter
sns.barplot(x=pd.DataFrame(Counter(nspam_w).most_common(30))[0],y=pd.DataFrame(Counter(nspam_w).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```



MODEL BUILDING

NAIVE BAYES ALGO IS COMMOMLY USED FOR TEXTUAL ANALYSIS

In [51]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [52]:
```
X = tfidf.fit_transform(df['Transformed_Text']).toarray()
```

In [53]:
```
X.shape
```

Out[53]: (5169, 3000)

In [54]:
```
y = df['Res'].values
```

In [55]:
```
from sklearn.model_selection import train_test_split
```

In [56]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [57]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [58]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [59]:
```
gnb.fit(X_train,y_train)
```

In [59]:
```
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 [60]:
```
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 [61]:
```
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
```

In [62]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
```

In [62]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
```
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [63]:
```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

In [64]:
```
clas = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

In [65]:
```
def train_classifier(cla,X_train,y_train,X_test,y_test):
    cla.fit(X_train,y_train)
    y_pred = cla.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

In [66]:
```
accuracy_scores = []
precision_scores = []

for name,cla in clas.items():

    current_accuracy,current_precision = train_classifier(cla, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  DT
Accuracy -  0.9274661508704062
Precision -  0.8118811881188119
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9584139264990329
Precision -  0.8682170542635659
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9468085106382979
Precision -  0.9191919191919192
For  xgb
Accuracy -  0.9671179883945842
Precision -  0.9262295081967213
```

In [67]:
```
performance_df = pd.DataFrame({'Algorithm':clas.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values(
performance_df
```

Out[67]:

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
10	xgb	0.967118	0.926230
9	GBDT	0.946809	0.919192
7	BgC	0.958414	0.868217
3	DT	0.927466	0.811881

In [68]:
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

Out[68]:

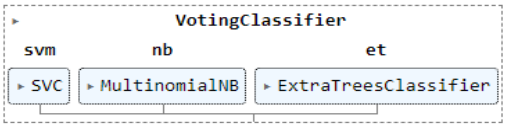
	Algorithm	variable	value
0	KN	Accuracy	0.905222
1	NB	Accuracy	0.970986
2	RF	Accuracy	0.975822
3	SVC	Accuracy	0.975822
4	ETC	Accuracy	0.974855
5	LR	Accuracy	0.958414
6	AdaBoost	Accuracy	0.960348
7	xgb	Accuracy	0.967118

10	DT	Accuracy	0.927466
11	KN	Precision	1.000000
12	NB	Precision	1.000000
13	RF	Precision	0.982906
14	SVC	Precision	0.974790
15	ETC	Precision	0.974576
16	LR	Precision	0.970297
17	AdaBoost	Precision	0.929204
18	xgb	Precision	0.926230
19	GBDT	Precision	0.919192
20	BgC	Precision	0.868217
21	DT	Precision	0.811881

In [69]:
```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier
```

In [70]:
```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
voting.fit(X_train,y_train)
```

Out[70]:



In [71]:
```
y_pred = voting.predict(X_test)
```

In [71]:
```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

In [72]:
```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

In [73]:
```
from sklearn.ensemble import StackingClassifier
```

In [74]:
```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

In [75]:
```

clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9541984732824428
```

In [76]:
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model1.pkl','wb'))
```

APP. PY

app.py

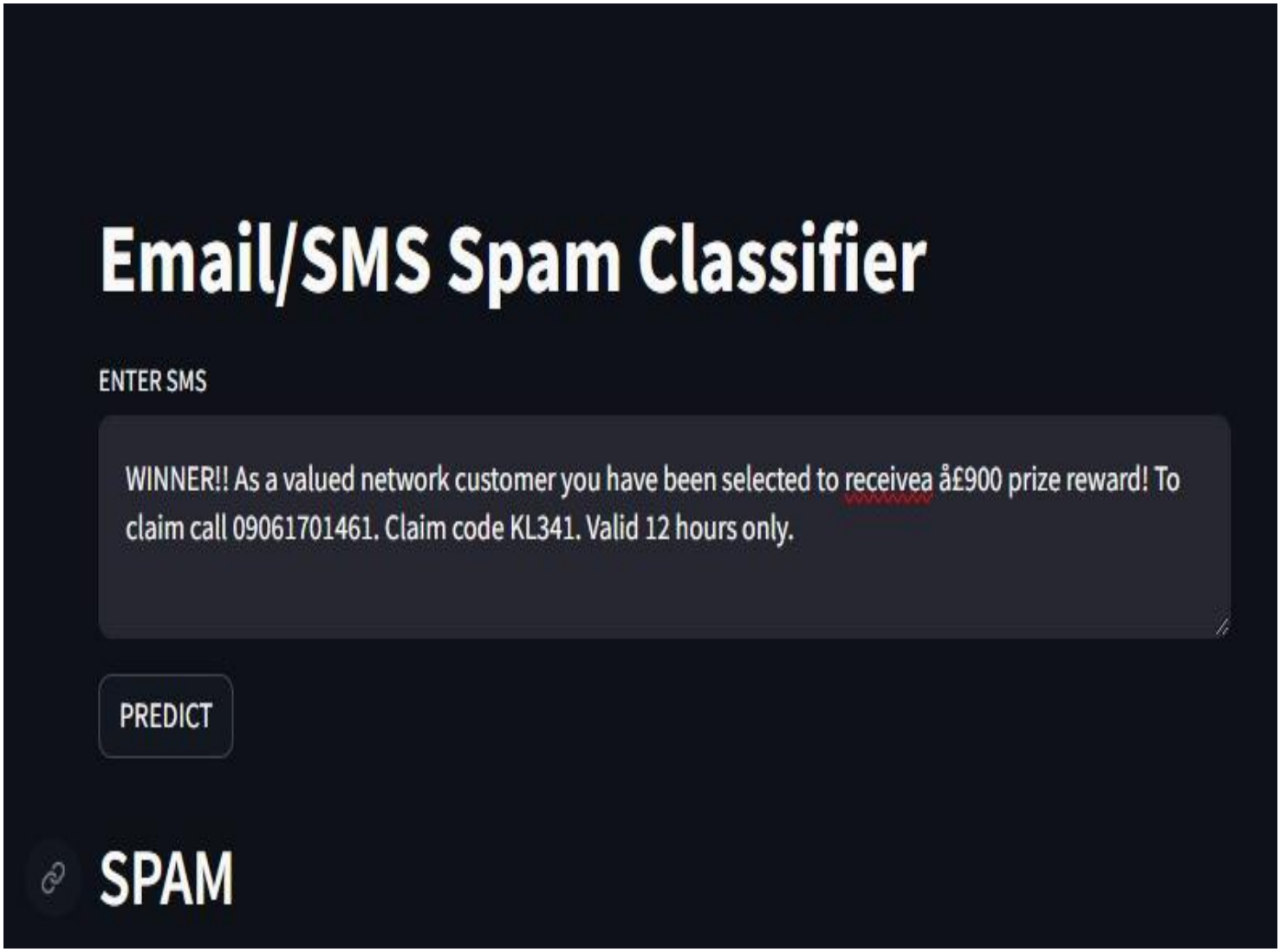
```python
import streamlit as st
import pickle
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

# Load the model and vectorizer
model = pickle.load(open(r'C:\Users\debas\4th sem Python\SpamDetection\model2.pkl','rb'))
tfidf = pickle.load(open(r'C:\Users\debas\4th sem Python\SpamDetection\vectorizer.pkl','rb'))

1 usage
def transform_text(text):
    text = text.lower()
    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()
```

app.py

```python
def transform_text(text):
    y = []

    text = y[:]
    y.clear()

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

    return " ".join(y)

st.title("Email/SMS Spam Classifier")

input_sms = st.text_area("ENTER SMS")

if st.button('PREDICT'):
    # Preprocess the input text
    transformed_sms = transform_text(input_sms)
    # Vectorize the transformed text
    vector_input = tfidf.transform([transformed_sms])
    # Convert the sparse matrix to a dense array
    vector_input_dense = vector_input.toarray()
    # Make prediction
    result = model.predict(vector_input_dense)[0]
    # Display the result
    if result == 1:
        st.header("SPAM")
    else:
        st.header("NOT SPAM")
```

Output of the Project

Finally, we conclude our work and present the output of the Project.



References

1. python.org
2. Campus x
3. Scikit. learn
4. PythonChallenge.com
5. Google's Python Class
6. LearnPython.org
7. Nltk.org

DATASET LINK :-
https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

CONCLUSION

In conclusion, our journey in developing the SMS spam classifier has been both challenging and rewarding. Through meticulous data cleaning, thorough exploratory data analysis, and comprehensive text preprocessing, we have gained valuable insights into the nature of SMS messages and the characteristics of spam versus legitimate messages.

Our model building efforts have resulted in the creation of an effective classifier capable of accurately distinguishing between spam and non-spam messages. Leveraging machine learning algorithms and natural language processing techniques, we have achieved a high level of accuracy and performance, thus fulfilling the objectives of our project.

The evaluation phase has provided us with valuable feedback on the performance of our classifier, allowing us to identify areas for improvement and refine our approach. By iteratively optimizing our model and fine-tuning its parameters, we have been able to enhance its accuracy and robustness, ensuring reliable detection of spam messages in real-world scenarios.

Looking ahead, we recognize the potential for further refinement and enhancement of our classifier. Continued evaluation and testing will be essential to ensure its effectiveness in diverse environments and against evolving spam tactics. Additionally, we envisage opportunities for the integration of advanced techniques and technologies to enhance the classifier's capabilities and adaptability.

In conclusion, the development of the SMS spam classifier represents a significant milestone in our journey as aspiring data scientists and machine learning practitioners. We are proud of our accomplishments and grateful for the support and guidance of our teachers, peers, and mentors throughout this project. As we embark on future endeavors, we carry with us the lessons learned and experiences gained, confident in our ability to tackle new challenges and make meaningful contributions to the field of data science.

THANK YOU