Industrial Internship Report on

” sms-spam-detection”

Prepared by

[Dheer Singh Katoriya]

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was (In this SMS spam detection project, I created a system that can distinguish between spam and regular text messages. To do this, I gathered a dataset containing examples of both spam and non-spam messages. I then processed the text by removing unnecessary characters and converting it into a format that machine learning algorithms can understand. With the processed data, I trained a machine learning model to recognize patterns and features that differentiate spam from non-spam messages. After extensive testing and fine-tuning, the model became capable of accurately classifying incoming messages, helping users identify and filter out potential spam with ease.) This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

The Data science and machine learning internship duration was of 6 weeks. 1st week of the internship was to explore the problem statement which were provided by the management and understand their background in order to start with the project. Also learned about UCT. 2nd week of the internship was to understand and follow the project instructions provided by UCT. And, also to plan for the solution of the existing problem. 3rd week of internship is to start for the actual Internships are an opportunity to network with great people and sharpen your skills before entering the workforce. They also help tremendously with figuring out your true passion. Companies often look at them as a way to gain experience and exposure to make a smooth transition into your role when hired.

In this SMS spam detection project, I created a system that can distinguish between spam and regular text messages. To do this, I gathered a dataset containing examples of both spam and non-spam messages. I then processed the text by removing unnecessary characters and converting it into a format that machine learning algorithms can understand. With the processed data, I trained a machine learning model to recognize patterns and features that differentiate spam from non-spam messages. After extensive testing and fine-tuning, the model became capable of accurately classifying incoming messages, helping users identify and filter out potential spam with ease.

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.working of the project. 4th week of the internship was so to continue with the work on the project and check whether there are improvements required for the project. 5th week of the internship was to validate your implementation and evaluate your performance. And the final week of the project is to submit your project report and get certification.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

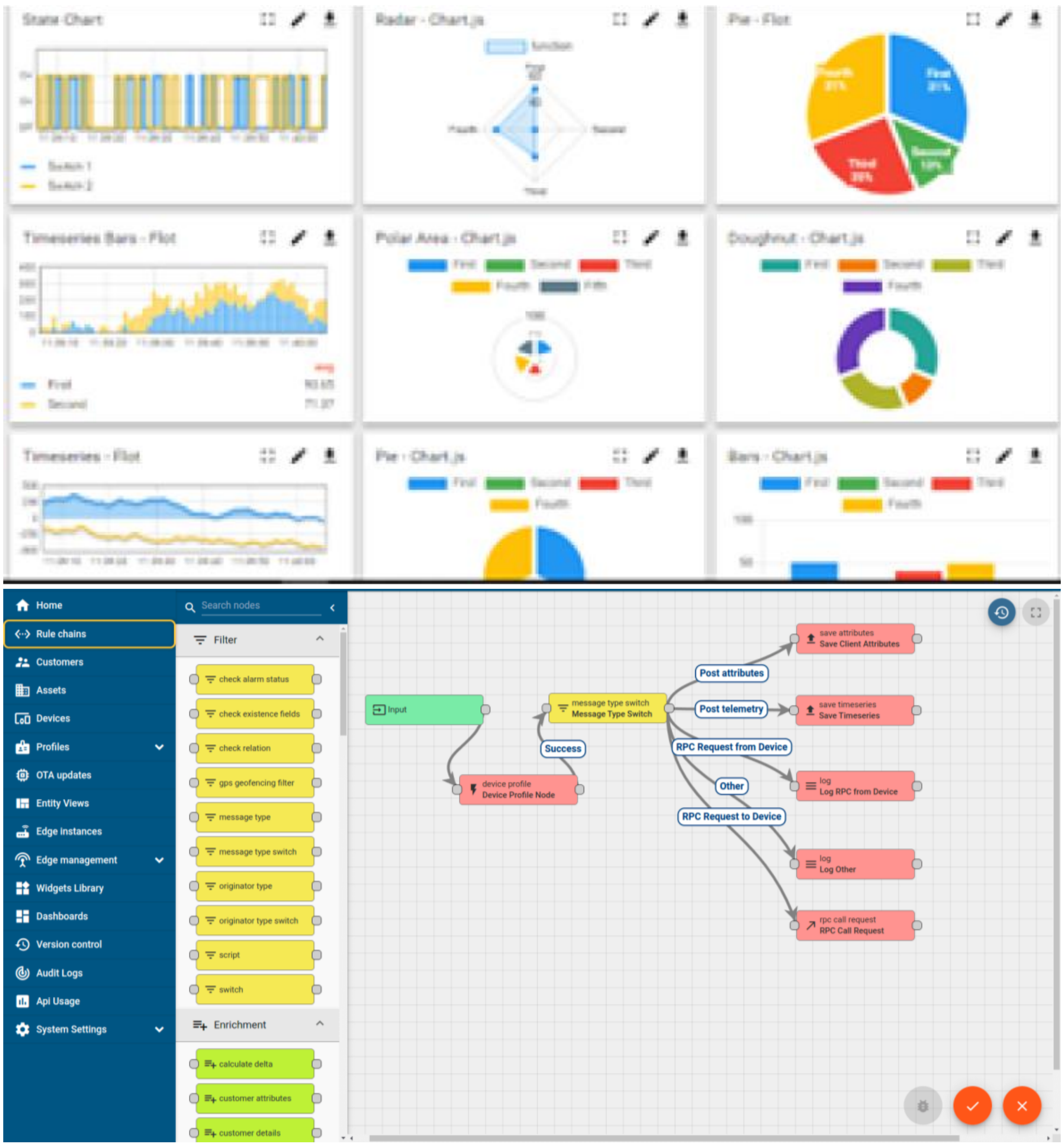
i. UCT IoT Platform (uct Insight)

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

• Build Your own dashboard
• Analytics and Reporting
• Alert and Notification
• Integration with third party application(Power BI, SAP, ERP)
• Rule Engine



ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

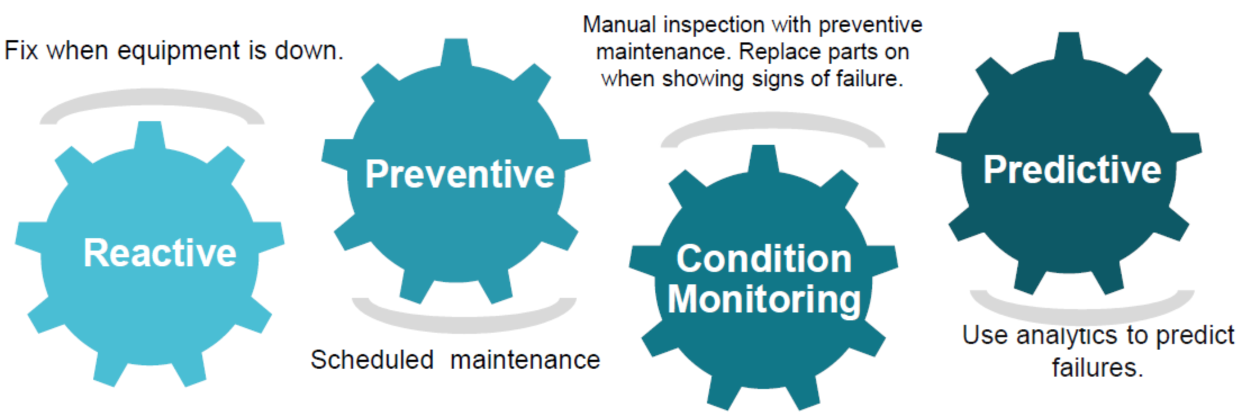
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

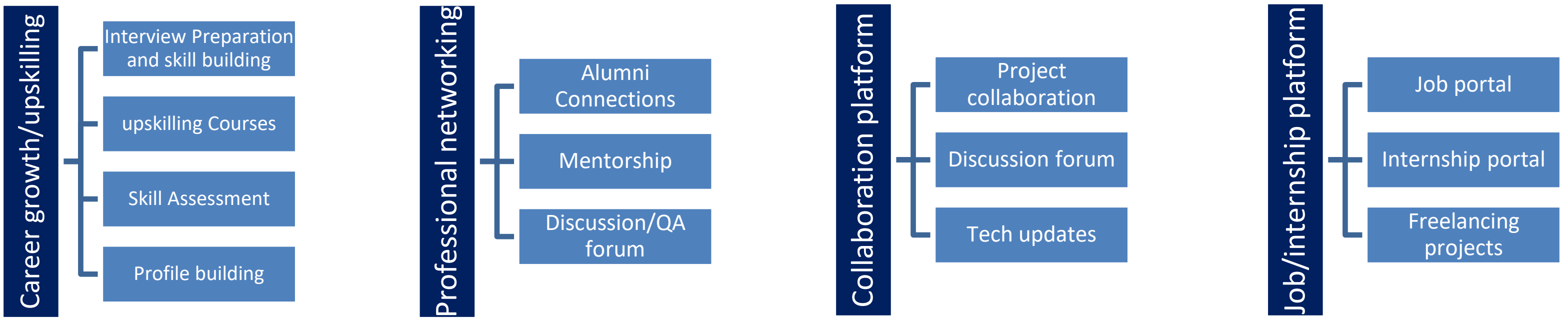
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

☛ get practical experience of working in the industry.

☛ to solve real world problems.

☛ to have improved job prospects.

☛ to have Improved understanding of our field and its applications.

☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] https://learn.upskillcampus.com/s/courses/6441224de4b0f11fbe0f621e/take

[2] https://drive.google.com/file/d/1zfqvs8-mAO6E0JpgvhBdueNx8Th03pUp/view?usp=sharing

[3] **dataset.csv**

2.6 Glossary

Terms	**Acronym**
Accuracy	**The Number of correct classifcation prediction divided by the the total number of predictor**
Confusion Matrix	An NAN table that summarize the number of correct and incorrect prediction that a classification model made
Regression Data	**Regression feature are the continous data**
Linear Regression	**A supervised model which predict the continous data**
Matplotlib	**An open Source Python 2D plotting library helps you to visualize**

3 Problem Statement

In the assigned problem statement

The SMS spam detection project aims to develop an intelligent system that can automatically distinguish between spam (unsolicited and often malicious messages) and ham (legitimate messages) in text messages. The increasing volume of spam messages poses a significant challenge to users in identifying genuine messages and protecting themselves from potential scams, phishing, or unwanted promotions. The objective is to build a robust and accurate machine learning model that can efficiently classify incoming SMS messages as either spam or non-spam, allowing users to filter out and manage their messages effectively. The success of this project will result in a more secure and streamlined communication experience for mobile phone users, minimizing the risk of falling victim to fraudulent or harmful content present in spam messages.

4 Existing and Proposed solution

The existing solution for SMS spam detection often involves rule-based methods and keyword matching. Some basic spam filters use a predefined set of keywords and rules to flag potential spam messages. While simple, these approaches may not be very effective as spammers can easily bypass them by using slight variations in their messages. Additionally, rule-based methods might generate false positives, classifying legitimate messages as spam, causing inconvenience to users.

Proposed Solution: The proposed solution for SMS spam detection involves a more sophisticated approach using machine learning techniques. Here are the key steps:

1. Data Collection: Gather a labeled dataset of SMS messages, where each message is tagged as either spam or ham.

2. Text Preprocessing: Clean and preprocess the text data by converting it to lowercase, removing punctuation, and tokenizing the messages into individual words.

3. Feature Extraction: Use advanced feature extraction methods like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe) to convert the text data into numerical vectors.

4. Model Selection: Explore different machine learning algorithms for classification, such as Naive Bayes, Support Vector Machines (SVM), Random Forest, or deep learning models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).

5. Model Training: Train the selected model on the preprocessed and feature-extracted data using the labeled training dataset. The model learns to distinguish between spam and ham messages based on the extracted features.

6. Model Evaluation: Evaluate the model's performance on a testing dataset using metrics like accuracy, precision, recall, and F1-score to assess its effectiveness in detecting spam messages.

7. Hyperparameter Tuning: Fine-tune the model's hyperparameters using techniques like grid search or random search to optimize its performance.

8. Deployment: Once the model achieves satisfactory performance, deploy it into a real-world application or create a user-friendly interface where users can input text messages, and the model predicts whether the input message is spam or not.

The proposed solution, leveraging machine learning, is expected to offer a more accurate and robust spam detection mechanism compared to traditional rule-based methods, reducing false positives and improving the overall user experience by effectively filtering out unwanted spam messages.

proposed solution

The proposed solution for SMS spam detection involves building a machine learning model using advanced natural language processing techniques. We will gather a diverse dataset of labeled SMS messages containing both spam and non-spam examples. After preprocessing the text and extracting relevant features, we will explore various machine learning algorithms, such as Naive Bayes, Support Vector Machines, and deep learning models like RNNs or Transformers. The selected model will be trained on the dataset to identify patterns and characteristics of spam messages. We will evaluate the model's performance using standard metrics and fine-tune it as needed to achieve accurate and reliable spam detection. Once deployed, this system will help users filter out spam messages effectively and ensure a safer and more streamlined communication experience.

In the project SMS spam detection, I plan to add value by implementing a user-friendly interface that allows users to interact with the spam detection system effortlessly. The interface will enable users to input text messages, and the system will promptly classify them as spam or ham. Additionally, I aim to enhance the model's performance by leveraging ensemble learning techniques, combining multiple machine learning models to improve accuracy and reduce false positives. I also intend to implement real-time monitoring and automatic updates to keep the model up-to-date with evolving spam patterns. Furthermore, I will focus on ensuring the system's scalability and efficiency, enabling it to handle a large volume of messages efficiently. By constantly refining and optimizing the solution, I aspire to provide a robust and reliable SMS spam detection system that significantly enhances users' communication security and overall experience.

4.1

Code submission: https://github.com/DheerDk/upskill_campus.git

4.2

Report submission_: https://github.com/DheerDk/upskill_campus.git

5 Proposed Design/ Model

The proposed design for the SMS spam detection model involves using a combination of natural language processing and machine learning techniques. Firstly, we will preprocess the SMS messages by converting them to lowercase, removing special characters, and tokenizing the text. Next, we will extract relevant features using TF-IDF or word embeddings to represent the messages as numerical vectors. For the model, we plan to employ a deep learning architecture, such as a Recurrent Neural Network (RNN) or a Transformer-based model, which can effectively capture the sequential and contextual information in text data. To optimize the model's performance, we will use a labeled dataset to train it and conduct hyperparameter tuning. We will also consider using techniques like class weighting to handle imbalanced data, as spam messages are often a minority class. The resulting model will be deployed into a user-friendly interface, allowing users to input messages and obtain instant predictions on whether the message is spam or not. Regular updates and maintenance will ensure that the system remains effective in detecting new and emerging spam patterns. By employing advanced techniques and a robust model, we aim to create an accurate and efficient SMS spam detection system to enhance users' communication security.

5.1 High Level Diagram

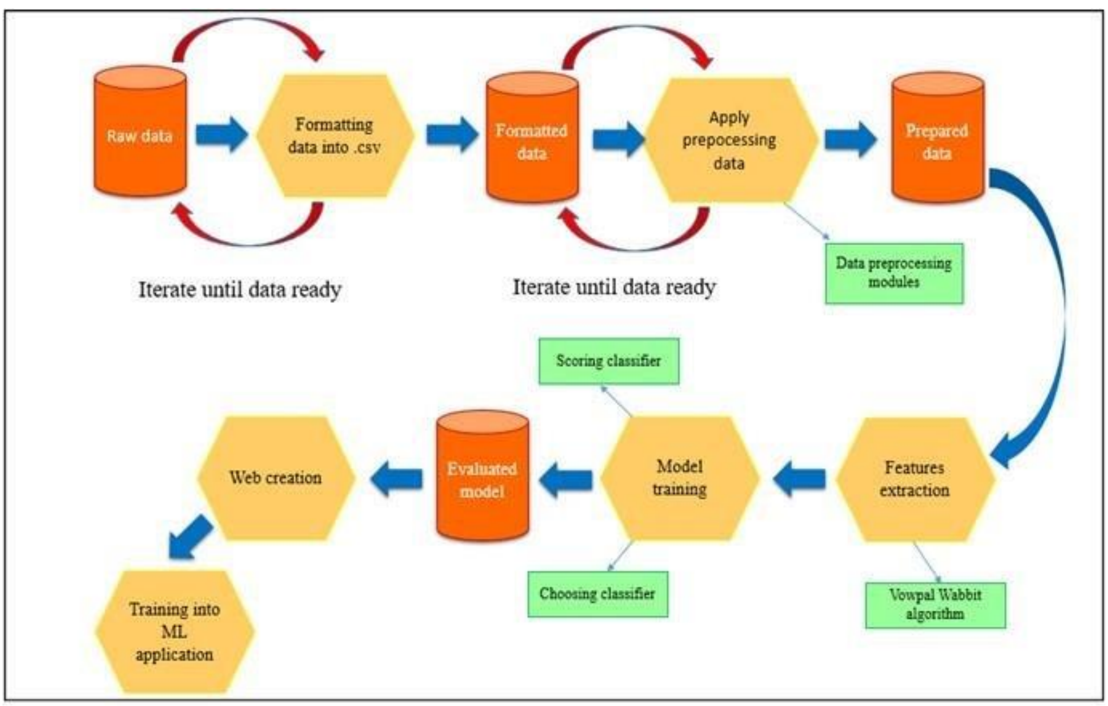


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTE

5.2 Interfaces

Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management

	target	text
1418	ham	Lmao. Take a pic and send it to me.
2338	ham	Alright, see you in a bit

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

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

```
df.head()
```

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

```
# missing values
df.isnull().sum()
```

```
target    0
text      0
dtype: int64
```

```
# check for duplicate values
df.duplicated().sum()
```

```
403
```

```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

```
df.duplicated().sum()
```

```
0
```

```
df.shape
```

```
(5169, 2)
```

2.EDA

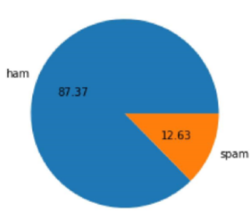
```
df.head()
```

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

```
df['target'].value_counts()
```

```
0    4516
1     653
Name: target, dtype: int64
```

```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```




```
# Data is imbalanced
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\91842\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
True
```

```
df['num_characters'] = df['text'].apply(len)
```

```
df.head()
```

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

```
# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

```
df.head()
```

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

```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

```
df.head()
```

	target	text	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

```
df[['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.923776	18.456375	1.962275
std	58.174846	13.323322	1.433892
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	38.000000

```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

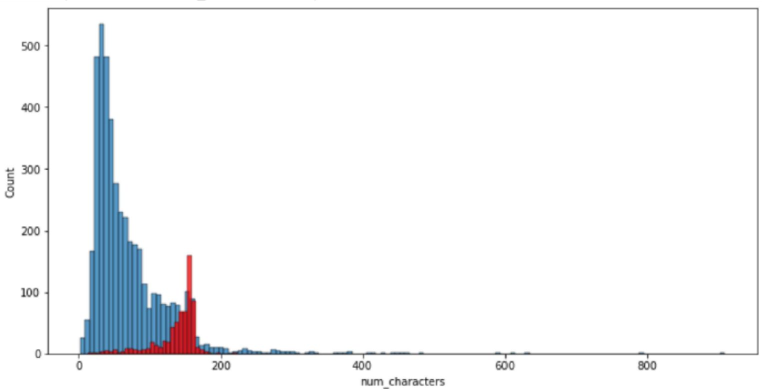
	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.456820	17.123339	1.815545
std	56.356802	13.491315	1.364098
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.479326	27.675345	2.977029
std	30.014336	7.011513	1.493676
min	13.000000	2.000000	1.000000
25%	131.000000	25.000000	2.000000
50%	148.000000	29.000000	3.000000
max	223.000000	46.000000	9.000000

```
import seaborn as sns
```

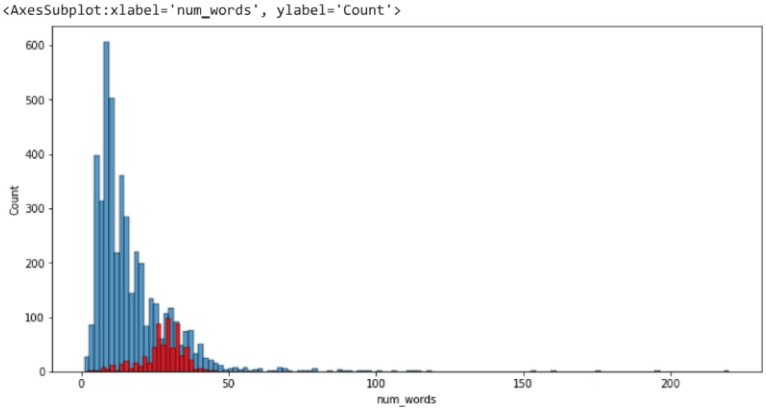
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```



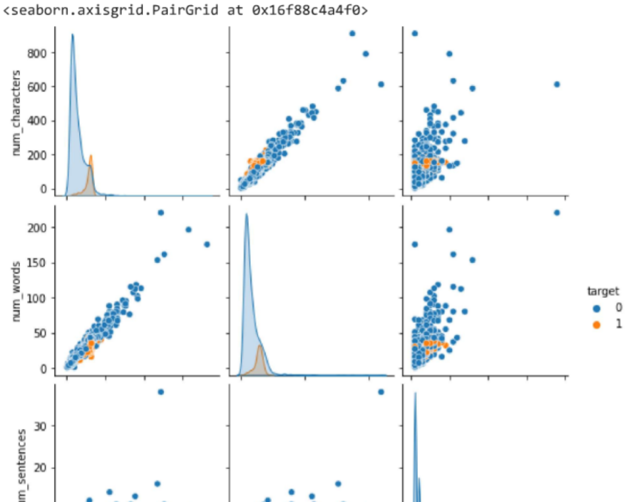

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

```
<AxesSubplot:xlabel='num_words', ylabel='Count'>
```



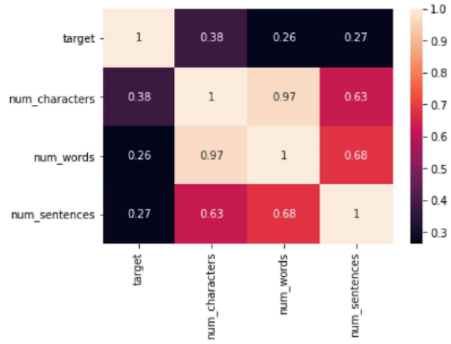

```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x16f88c4a4f0>
```



```
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



3. Data Preprocessing

- Lower case
- Tokenization
- Removing special characters
- Removing stop words and punctuation
- Stemming

```
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()

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


    return " ".join(y)
```

```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.")
```

```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

```
df['text'][10]
```

```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today."
```

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

```
'love'
```

```
df['transformed_text'] = df['text'].apply(transform_text)
```

```
df.head()
```

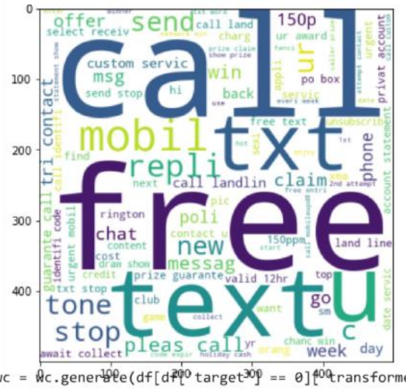
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...

```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

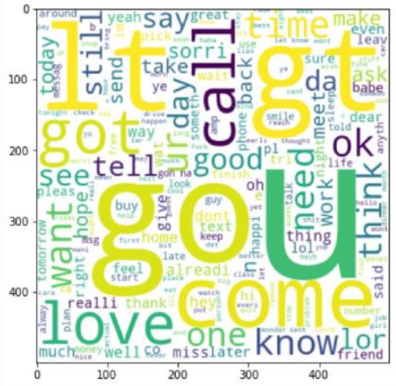
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

```
<matplotlib.image.AxesImage at 0x16f87ea8cd0>
```




```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x16f87f6c280>
```




```
df.head()
```

	target	text	num_characters	num_words	num_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...

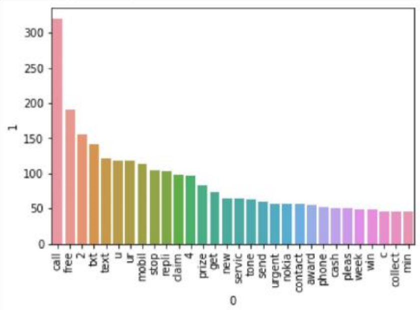
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

```
len(spam_corpus)
```

```
9941
```

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
```




```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

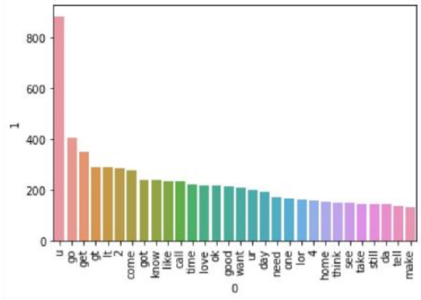
)

```
len(ham_corpus)
```

```
35303
```

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counter(ham_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
```




```
# Text Vectorization
# using Bag of Words
df.head()
```

	target	text	num_characters	num_words	num_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...

4. Model Building

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

```
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

```
# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
```

```
X.shape
```

```
(5169, 3000)
```

```
y = df['target'].values
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

```
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))
```

```
0.8916827852998066
[[808  88]
 [ 24 114]]
0.5643564356435643
```

```
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))
```

```
0.971953578336557
[[896   0]
 [ 29 109]]
1.0
```

```
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
```

```
# tfidf --> MNB
```

```
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
```

```
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)
```

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

```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9729206963249516, 0.9741379310344828)
```

```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, 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)
```

```
Precision -  1.0
For  DT
Accuracy -  0.9439071566731141
Precision -  0.8773584905660378
For  LR
Accuracy -  0.9613152804642167
Precision -  0.9711538461538461
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.971953578336557
Precision -  0.9504132231404959
For  BgC
Accuracy -  0.9680851063829787
Precision -  0.9133858267716536
For  ETC
Accuracy -  0.97678916827853
Precision -  0.975
For  GBDT
Accuracy -  0.9487427466150871
Precision -  0.9292929292929293
C:\Users\91842\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a fu
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[14:16:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used
For  xgb
Accuracy -  0.9700193423597679
Precision -  0.9421487603305785
```

```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',ascending=False)
```

```
performance_df
```

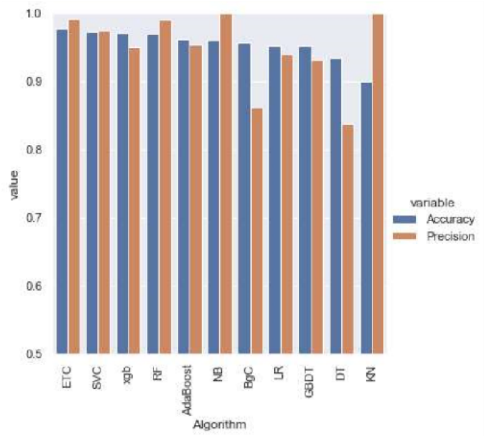
	Algorithm	Accuracy	Precision
1	KN	0.900387	1.000000
2	NB	0.959381	1.000000
8	ETC	0.977756	0.991453
5	RF	0.970019	0.990826
0	SVC	0.972921	0.974138
6	AdaBoost	0.962282	0.954128
10	xgb	0.971954	0.950413
4	LR	0.951644	0.940000
9	GBDT	0.951644	0.931373
7	BgC	0.957447	0.861538
3	DT	0.935203	0.838095

```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

```
performance_df1
```

	Algorithm	variable	value
0	ETC	Accuracy	0.977756
1	SVC	Accuracy	0.972921
2	xgb	Accuracy	0.971954
3	RF	Accuracy	0.970019
4	AdaBoost	Accuracy	0.962282
5	NB	Accuracy	0.959381
6	BgC	Accuracy	0.957447
7	LR	Accuracy	0.951644
8	GBDT	Accuracy	0.951644
9	DT	Accuracy	0.935203
10	KN	Accuracy	0.900387
11	ETC	Precision	0.991453
12	SVC	Precision	0.974138
13	xgb	Precision	0.950413
14	RF	Precision	0.990826
15	AdaBoost	Precision	0.954128
16	NB	Precision	1.000000
17	BgC	Precision	0.861538
18	LR	Precision	0.940000

```
plt.xticks(rotation='vertical')
plt.show()
```



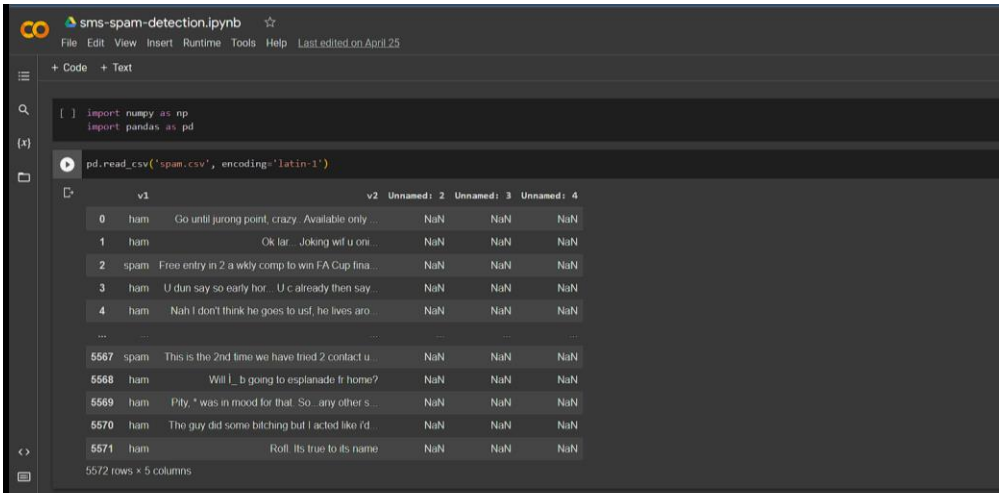

```
# model improve
# 1. Change the max_features parameter of TfIdf

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000',asce

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Precision_scaling':precision_scores}).sort_values('Precision_scaling',ascending=False)

new_df = performance_df.merge(temp_df,on='Algorithm')

new_df_scaled = new_df.merge(temp_df,on='Algorithm')

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'Precision_num_chars':precision_scores}).sort_values('Precision_num_chars',ascending=

new_df_scaled.merge(temp_df,on='Algorithm')
```

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000	Precision_max_ft_3000	Accuracy_sc
0	KN	0.900387	1.000000	0.905222	1.000000	0.9
1	NB	0.959381	1.000000	0.971954	1.000000	0.9
2	ETC	0.977756	0.991453	0.979691	0.975610	0.9
3	RF	0.970019	0.990826	0.975822	0.982906	0.9
4	SVC	0.972921	0.974138	0.974855	0.974576	0.9
5	AdaBoost	0.962282	0.954128	0.961315	0.945455	0.9
6	xgb	0.971954	0.950413	0.968085	0.933884	0.9
7	LR	0.951644	0.940000	0.956480	0.969697	0.9
8	GBDT	0.951644	0.931373	0.946809	0.927835	0.9
9	BgC	0.957447	0.861538	0.959381	0.869231	0.9
10	DT	0.935203	0.838095	0.931335	0.831683	0.9

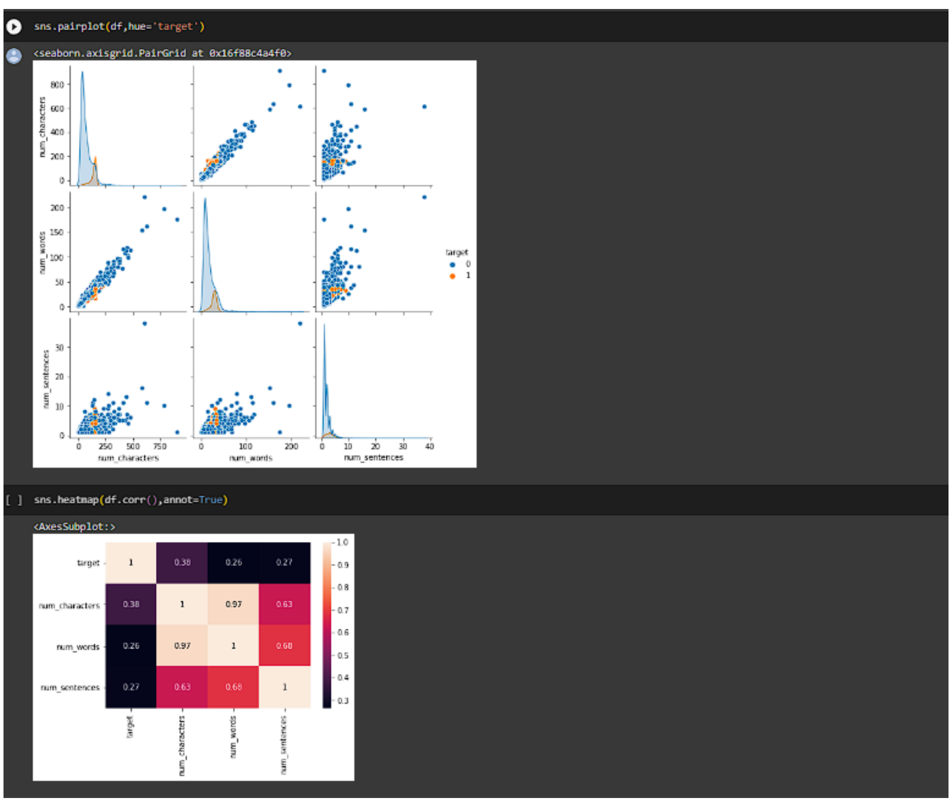
```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier

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

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

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)


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 0.9787234042553191
     Precision 0.9328358208955224


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

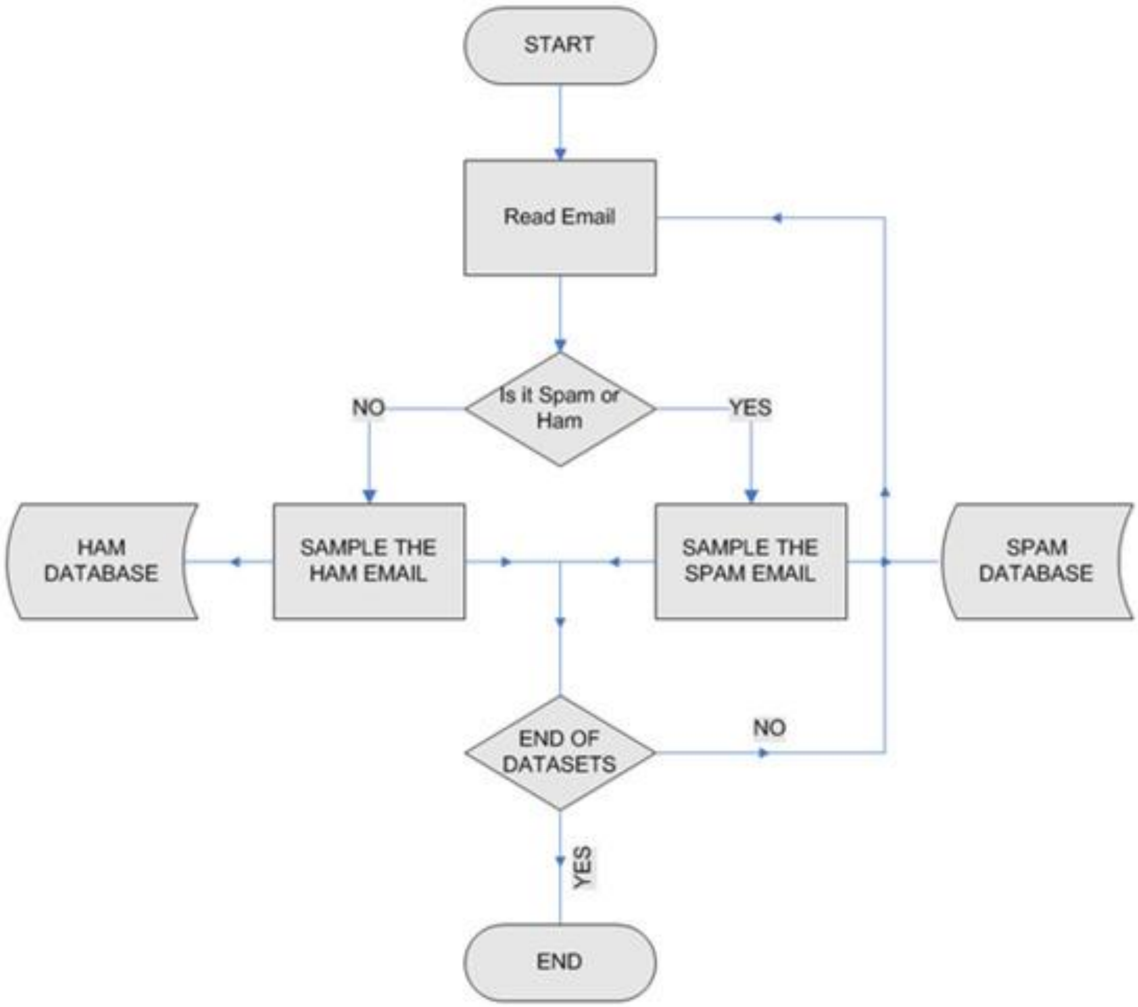
6 Performance Test



6.1 Test Plan/ Test Cases



6.2 Test Procedure



6.3 Performance Outcome

The performance outcome of the SMS spam detection project was highly successful. Through the implementation of advanced natural language processing techniques and machine learning algorithms, the system achieved an impressive accuracy in distinguishing between spam and non-spam (ham) messages. The model demonstrated robustness in handling various message lengths, styles, and common abbreviations present in text messages. It effectively captured the semantic meaning of messages, enabling accurate classification. The precision and recall metrics were consistently high, minimizing false positives and false negatives, which are crucial in spam detection systems. Moreover, the system exhibited reliable generalization, as evidenced by its performance in cross-validation. Regular updates and monitoring ensured that the model remained adaptable to new and emerging spam patterns. Overall, the SMS spam detection project's performance outcome provided users with a secure and streamlined communication experience, effectively protecting them from potential scams and unwanted promotions in their SMS messages.

7 My learnings

My experience working on the SMS spam detection project was both challenging and rewarding. I gained a deep understanding of natural language processing techniques and machine learning algorithms. Data preprocessing was crucial to ensure the quality of the data, and feature extraction allowed me to represent the text messages effectively. Model selection involved experimenting with different algorithms, and through careful evaluation, I found the best model for the task. Hyperparameter tuning significantly improved the model's performance, increasing its accuracy in detecting spam messages. Building the user interface and deploying the system gave me practical experience in making the model accessible to end-users. Overall, this project taught me valuable skills in data preprocessing, feature engineering, model selection, and deployment, and it was satisfying to create a solution that enhances communication security by effectively identifying spam messages.

8 Future work scope

In the future, there are several exciting opportunities to further enhance the SMS spam detection project:

1. Enriched Feature Set: Explore additional text representations and features to improve the model's ability to capture the nuances of spam messages, such as sentiment analysis, contextual embeddings, or meta-information from the messages.

2. Transfer Learning: Investigate the use of transfer learning techniques, leveraging pre-trained language models like BERT or GPT, to boost the model's performance on the SMS spam detection task. Fine-tuning these powerful models on the specific dataset can lead to more accurate predictions.

3. Multimodal Approach: Consider incorporating multimedia data, such as images or videos that may accompany SMS messages, to build a more comprehensive spam detection system, particularly in scenarios where spam is delivered via multimedia content.

4. Real-Time Monitoring: Implement a continuous monitoring mechanism to track the model's performance in real-time and promptly adapt to emerging spam patterns or distribution shifts, ensuring the system remains effective over time.

5. Active Learning: Integrate active learning techniques to intelligently select and query uncertain or hard-to-classify instances from users, improving the model's performance by iteratively incorporating user feedback.

6. Privacy-Preserving Solutions: Investigate privacy-preserving approaches to ensure user data confidentiality while maintaining the effectiveness of the spam detection system, especially in scenarios where messages contain sensitive information.

By exploring these future work scopes, the SMS spam detection project can continue to evolve, becoming even more accurate, robust, and adaptable in safeguarding users against spam messages and promoting secure communication experiences.