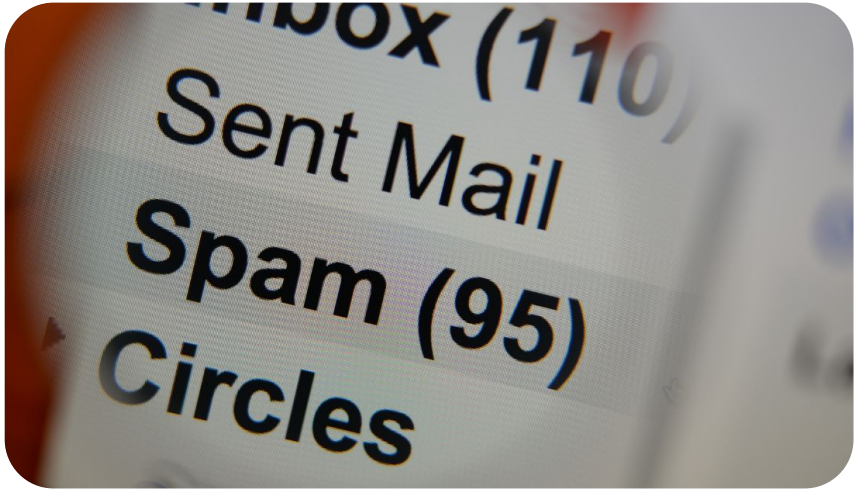
SMS Spam Classification



Presented by Alfiyyah Ajeng Nurardita - SI602003

Table of contents

Introduction

Problem Identification

E-commerce platforms often face spam in product reviews, user messages, and customer support tickets.

This leads to poor **user experience**, reduced **customer trust**, and potential security risks such as **phishing** and **scams**.

Solution

A machine learning–based **spam classification system** can automatically distinguish between spam and legitimate content, ensuring only relevant and safe information reaches users.

Objective

The project aims to:

1. Develop a reliable spam vs. ham classification model.
2. Apply effective text preprocessing to improve accuracy.
3. Evaluate the model using precision, recall, and F1-score.

Methodology

Dataset

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

EDA

Checking class imbalance, text length, and word cloud

Text Preprocessing

Cleaning and transforming raw text data into a structured format

Model Building

Comparing 4 algorithm (2 Dense Layer, LSTM, Bi-LSTM, and RNN) to build classification model

Dataset

No	v1	v2
1	ham	Go until jurong point, crazy.. Available only ...
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup fina...
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives aro...

Data Cleaning

1. Rename columns name
v1 -> Target
v2 -> Text

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

2. Checking missing values

```
#checking missing values
df.isna().sum()
```

```
Target    0
Text      0
dtype: int64
```

3. Checking and drop duplicates values

```
#checking duplicate values
df.duplicated().sum()
```

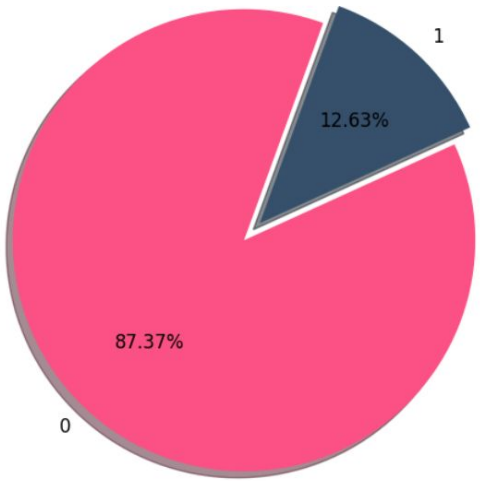
```
403
```

```
#drop duplicate values
df = df.drop_duplicates()
```

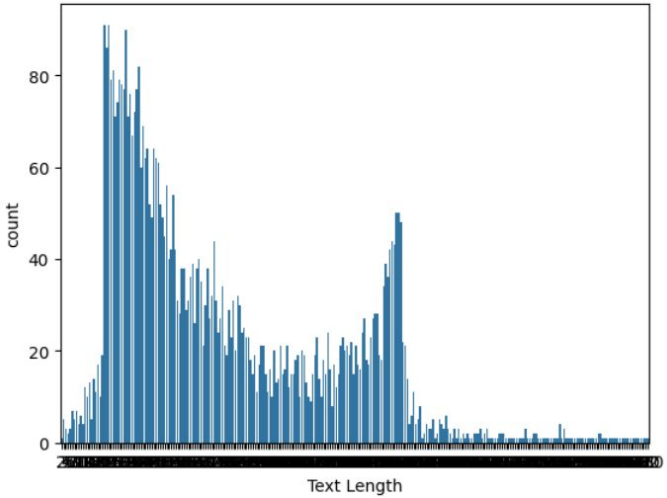
4. Encoding (ham -> 0, spam -> 1)

```
#encoding target column
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['Target'] = encoder.fit_transform(df['Target'])
```

Exploratory Data Analysis (EDA)

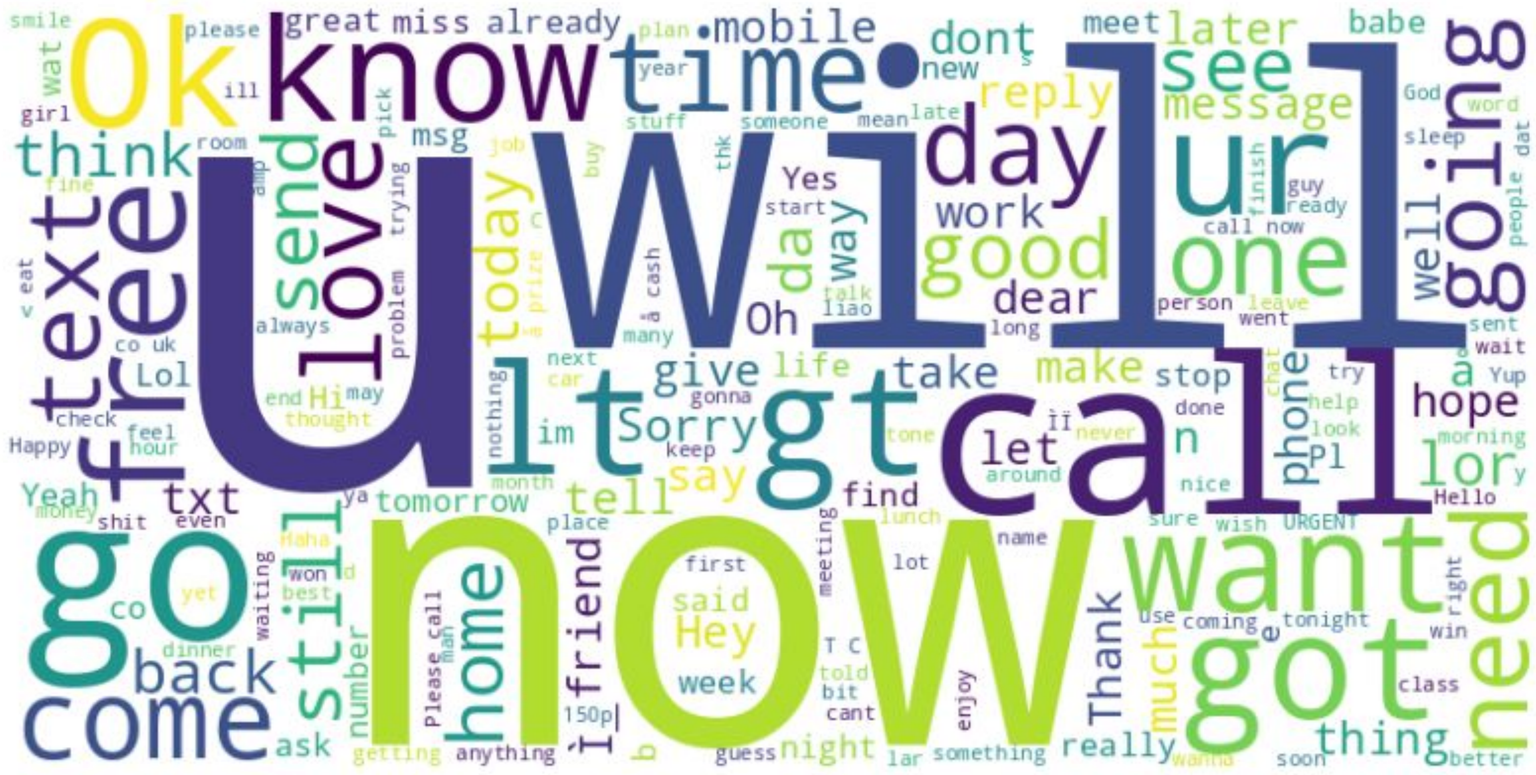


Data points 0 (ham) are 6.9 times more than points with 1 (spam) labels, which leads to **Imbalanced dataset**



We have 408237 words in our Dataframe the average word count in every sentence is 78

Word Cloud



u
will
now
call
go
Ok
know
time
free
text
day
ur
going
want
got
need
come
love
send
lt
gt
one
good
see
message
think
home
still
back
dont
later
mobile
great
miss
already
meet
babe
reply
today
friend
Sorry
give
take
make
hope
tell
say
txt
Yeah
Hey
said
week
night
really
thing
tomorrow
Thank
much
phone
dear
work
way
da
Oh
Yes
Lol
lor
let
im
find
stop
life
smile
please
plan
new
year
ill
wat
girl
msg
room
stuff
someone
mean
late
God
word
sleep
fine
amp
job
buy
thk
start
guy
ready
finish
call now
people
eat
pick
trying
always
problem
prize
cash
many
talk
liao
long
person
leave
went
sent
wait
Yup
co uk
next
car
end
Hi
may
gonna
check
thought
feel
hour
Happy
nothing
keep
never
tone
done
help
look
morning
try
chat
around
nice
month
ya
Hello
money
shit
even
place
lunch
sure
wish
URGENT
name
meeting
Haha
won
best
first
lot
right
waiting
yet
dinner
coming
tonight
use
win
number
Please call
told
bit
class
enjoy
cant
150p
ask
getting
anything
guess
lar
something
wanna
soon
better

Text Preprocessing

```
# Case folding
text = text.lower()

# Mention removal
text = re.sub("@[A-Za-z0-9_]+", " ", text)

# Hashtags removal
text = re.sub("#[A-Za-z0-9_]+", " ", text)

# Newline removal (\n)
text = re.sub(r"\\n", " ",text)

# Whitespace removal
text = text.strip()

# URL removal
text = re.sub(r"http\S+", " ", text)
text = re.sub(r"www.\S+", " ", text)

# Non-letter removal (such as emoticon, symbol (like μ, $, 兀), etc
text = re.sub("[^A-Za-z\s']", " ", text)

# Tokenization
tokens = word_tokenize(text)

# Stopwords removal
tokens = [word for word in tokens if word not in stpwds_id]

# Stemming
tokens = [stemmer.stem(word) for word in tokens]

# Combining Tokens
text = ' '.join(tokens)
```

1. **text.lower** → Converts all text to lowercase to maintain consistency.
2. **Removal (mention, hashtag, newline, whitespace, URL, non-letter, stopwords)** → Cleans the text by removing symbols, links, common words, and irrelevant characters.
3. **Tokenization** → Splits text into smaller units (tokens/words) for analysis.
4. **Stemming** → Reduces words to their root form by removing affixes.

Model Building

2 Dense Layers

The model uses a TextVectorization + Embedding layer, followed by GlobalAveragePooling and Flatten. It has two Dense layers: one hidden layer with 32 ReLU units and an output layer with 1 sigmoid unit for binary classification. Compiled with Adam optimizer and Binary Crossentropy loss.

LSTM

The model uses an Embedding layer (100 dimensions, input length 100), followed by an LSTM layer with 64 units, and a Dense output layer with 1 sigmoid unit for binary classification. It is compiled with Adam optimizer and Binary Crossentropy loss, trained for 5 epochs with batch size 32.

Bi-LSTM

The model uses a TextVectorization + Embedding layer, followed by two Bidirectional LSTM layers (first with return_sequences=True, both with 64 units, tanh activation). The output is passed through Flatten → Dropout (0.1) → Dense(32, ReLU) → Dense(1, Sigmoid) for binary classification. It is compiled with Adam optimizer and Binary Crossentropy loss, trained for 5 epochs.

LSTM

The model consists of an Embedding layer (64 dimensions, input length 100), followed by a SimpleRNN layer with 64 units, and a Dense output layer with 1 sigmoid unit for binary classification. It is compiled with Adam optimizer and Binary Crossentropy loss, trained for 5 epochs with batch size 32.

Model Evaluation

Model	Loss	Accuracy
2 Dense Layer	0.571600	0.979691
LSTM	0.347461	0.890716
Bi-LSTM	0.067345	0.983559
Simple RNN	0.354138	0.890716

Conclusion

- **metrics** : from all models, Bi-LSTM have a greatest accuracy which 98%.
- **problem** : the models use imbalanced dataset, most of data points have the label of ham. For this case accuracy can't be a good metrics

Thank You !

Full Code : colab

Lets Connect


https://www.linkedin.com/in/ajeng-nurardita/


https://github.com/alfiyyahajeng