SMS SPAM DETECTION USING IBM WATSON STUDIO

1. INTRODUCTION

Overview

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion-dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS, they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So, Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

Proposed System

To avoid such Spam SMS, people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem, it is needful to use a smarter technique which correctly identifies Spam SMS. Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

2. LITERATURE SURVEY

2.1 Existing Problem

Spam is unsolicited and unwanted messages sent electronically and whose content could also be malicious. Email spam is sent/received over the net while SMS spam is usually transmitted over a mobile network. We'll call the user that sent spam as spammers. SMS messages are usually the bottom (if not free) for the user to send, making it appealing for unrightful exploitation. this can be further aggravated by the very fact that SMS is sometimes regarded by the user as a safer, more trustworthy style of communication than other sources, e. g., emails.

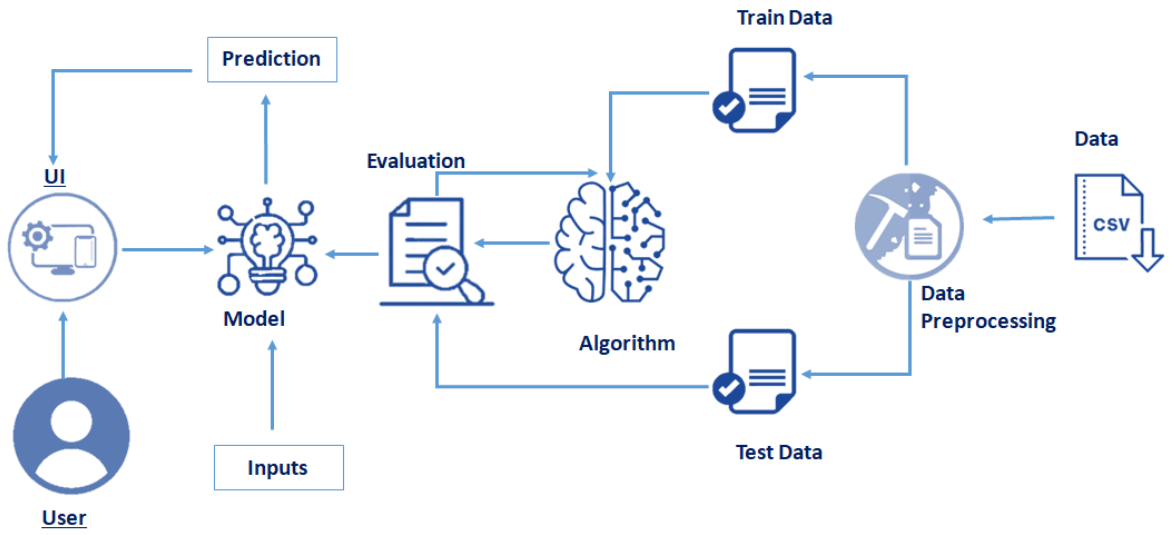
The dangers of spam messages for the users are many: undesired advertisement, exposure of personal information, becoming a victim of a fraud or financial scheme, being lured into malware and phishing websites, involuntary exposition to inappropriate content, etc. For the network operator, spam messages end in an increased cost in operations.

2.2 Proposed system

To avoid such Spam SMS, people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem, it is needful to use a smarter technique which correctly identifies Spam SMS. Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

Hardware requirements

- 2 GB ram or above
- Dual core processor or above
- Internet connection

Software requirements

- Anaconda Navigator
- Python packages
- IBM Watson Studio

4. EXPERIMENTAL INVESTIGATION

Here we are going to build a machine learning model that predicts whether the given message is a spam or not, with the help of natural language processing. Natural Language processing or NLP is a subset of Artificial Intelligence (AI), where it is basically responsible for the understanding of human language by a machine. It is done by creating a corpus to store spam and ham messages, pre-processing the data and splitting them to test and train sets. From this a model is created then our messages are tested with this model for possible spams.

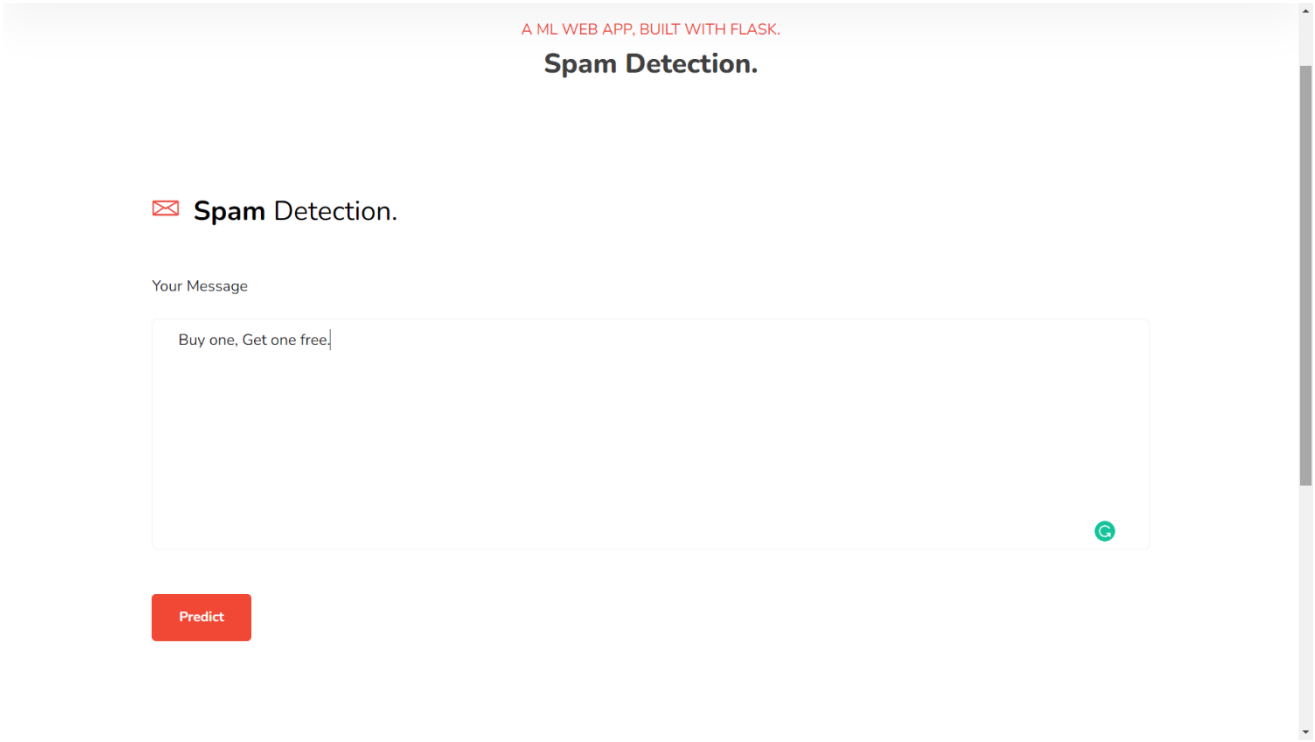
5. FLOWCHART

- User interacts with the UI (User Interface) to type their message as input.
- Message input is analysed by the model which is integrated.
- Once model analyses the given message, the prediction is showcased on the UI

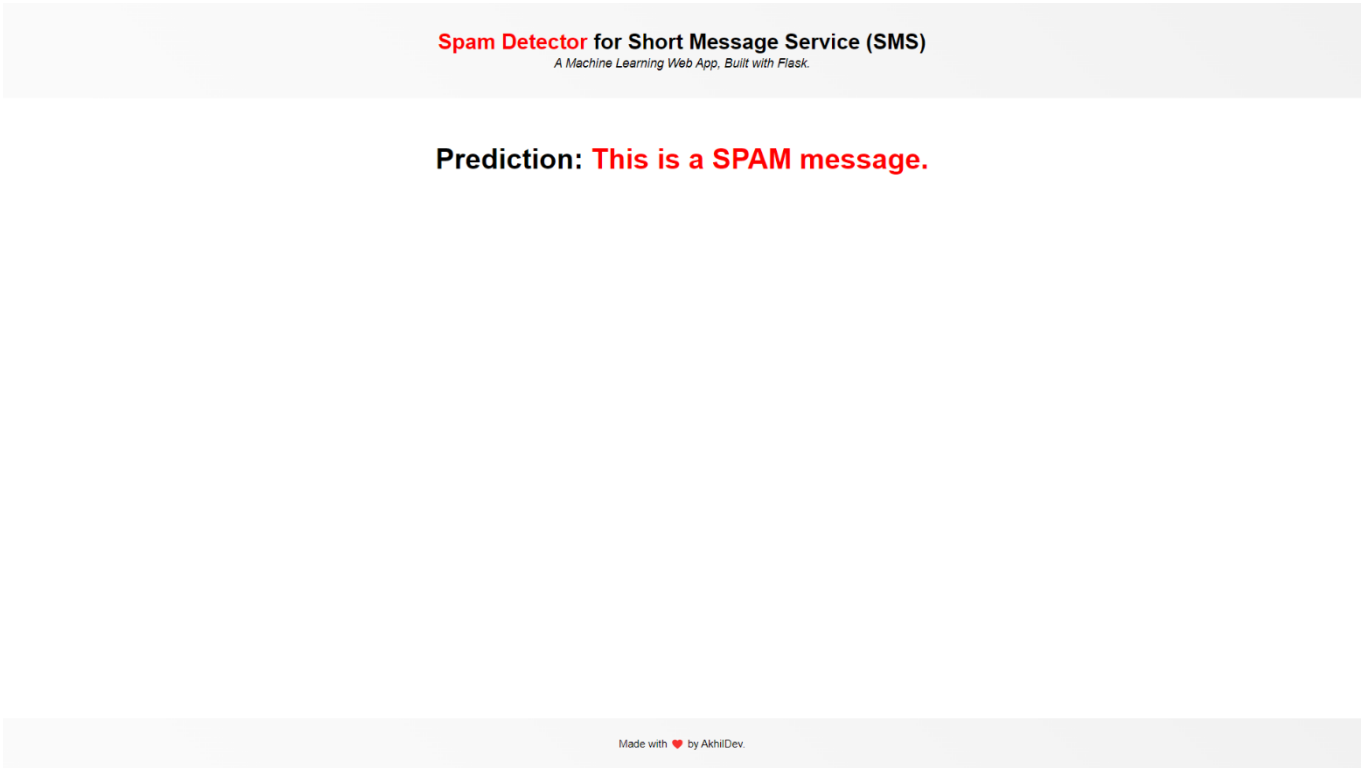
1. Data Collection
    a. Collect the dataset or create the dataset
2. Understanding the data
    a. Importing the required libraries
    b. Reading the Dataset
    c. EDA on Dataset
    d. Take care of missing data
    e. Data Visualization
    f. Cleaning The Text
    g. Building count vectors with scikit-learn Count-Vectorizer for text classification
    h. Splitting Data into Train and Test
3. Model Building
    a. Training and testing the model
    b. Evaluation of Model
    c. Saving the model
4. Application Building
    a. Create an HTML file
    b. Build Python Code
5. Final UI
    a. Dashboard Of the flask app

6. RESULT


AkhilDev
Home
About
Spam
SMS SPAM Detection
Using NLP.


A ML WEB APP, BUILT WITH FLASK.
Spam Detection.
Spam Detection.
Your Message
Buy one, Get one free.
Predict

Prediction:


Spam Detector for Short Message Service (SMS)
A Machine Learning Web App, Built with Flask.
Prediction: This is a SPAM message.
Made with ♥ by AkhilDev.

7. ADVANTAGES AND DISADVANTAGES

Advantages

- Efficient program for spam detection
- Accurate output is produced
- Will predict spam with extreme accuracy
- Relatively inexpensive and fast

Disadvantages

- Naive bayes is based on the conditional independence of features assumption
- sometimes oversimplifies the problem

8. APPLICATION

- SMS
- E-mails

9. CONCLUSION

Using this project, we can predict if the message is spam or not. This program will split each word of the message then check for usual words seen in spam messages with the help of the model. When there are too much spam keywords present in the message, the message is marked as spam message.

10. FUTURE SCOPE

This program allows users to predict if the coming message is a spam or not. By the help of this prediction, unwanted messages and emails are removed from the inbox, to a separate section. It helps reduce the stress on user identifying important mails and messages easily, since the spam are not shown in the inbox.

11. BIBLIOGRAPHY

- https://towardsdatascience.com/nlp-spam-detection-in-sms-text-data-using-deep-learning-b8632db85cc8#:~:text=NLP%20%2C%20a%20branch%20of%20Artificial,language%20translations%20and%20document%20classification.
- https://towardsdatascience.com/how-to-identify-spam-using-natural-language-processing-nlp-af91f4170113

12. APPENDIX

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import os, types

import pandas as pd

from botocore.client import Config

import ibm_boto3


def __iter__(self): return 0


client_d0ee6f8d4b2949c5a86728363528f9bc = ibm_boto3.client(service_name='s3',

    ibm_api_key_id='Gu6IIHmIOCbwoaMUrfV3oGR9qnfhORdxKcKd6b4Nns9L',

    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",

    config=Config(signature_version='oauth'),

    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')


body = client_d0ee6f8d4b2949c5a86728363528f9bc.get_object(Bucket='smsspamdetection-
donotdelete-pr-dzi7ojkxswt4w7',Key='spam_ham_dataset.csv')['Body']

if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )


df = pd.read_csv(body)

df.shape
```

```
df.ndim

df.size

df.isna().sum()

df.info()

df.head()

df.tail()

df["label_num"].value_counts().plot(kind="bar",figsize=(12,6))

plt.xticks(np.arange(2), ('Non spam', 'spam'),rotation=0)

import re

corpus = []

length = len(df)

import nltk

nltk.download('stopwords')

for i in range(0,1000):

    text = re.sub("[^a-zA-Z0-9]"," ",df["text"][i])

    text = text.lower()

    text = text.split()

    pe = PorterStemmer()

    stopword = stopwords.words("english")

    text = [pe.stem(word) for word in text if not word in set(stopword)]

    text = " ".join(text)

    corpus.append(text)

corpus

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=3500)

X = cv.fit_transform(corpus).toarray()

y = pd.get_dummies(df['label'])
```

```
y = y.iloc[:, 1].values

import pickle

pickle.dump(cv, open('cv.pkl', 'wb'))

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()

model.fit(X_train, y_train)

y_pred=model.predict(X_test)

y_pred

from sklearn.metrics import confusion_matrix,accuracy_score

cm = confusion_matrix(y_test, y_pred)

score = accuracy_score(y_test,y_pred)

print(cm)

print('Accuracy Score Is:- ' ,score*100)

import pickle

pickle.dump(model, open("spam.pkl", "wb"))

a="spam.pkl"

!tar -zcvf spam.tgz spam.pkl

b="cv.pkl"

!tar -zcvf spam1.tgz cv.pkl

import ibm_watson_machine_learning

from ibm_watson_machine_learning import APIClient

import json

wml_credentials = {

   "apikey":"jHmqhPkx75nDgJ8JxevI89h6ol_-kv2ZbhLWHFwGGKPS",

   "url" : "https://us-south.ml.cloud.ibm.com"
```

```
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
wml_client.set.default_space(space_id)
software_spec_uid = wml_client.software_specifications.get_id_by_name("default_py3.8")
software_spec_uid
model_details = wml_client.repository.store_model(model='spam.tgz',
                                  meta_props= {
                                     wml_client.repository.ModelMetaNames.NAME:"NLP",
                                     wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",

wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
                                  })
model_id = wml_client.repository.get_model_uid(model_details)
model_uid = wml_client.repository.get_model_id(model_details)
model_details1 = wml_client.repository.store_model(model='spam1.tgz',
                                  meta_props= {
                                     wml_client.repository.ModelMetaNames.NAME:"NLP",
                                     wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",

wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
                                  })
model_id1 = wml_client.repository.get_model_uid(model_details1)
model_uid1 = wml_client.repository.get_model_id(model_details1)




pip install ibm_watson_machine_learning
```

```
from ibm_watson_machine_learning import APIClient

wml_credentials = {

   "apikey":"jHmqhPkx75nDgJ8JxevI89h6ol_-kv2ZbhLWHFwGGKPS",

   "url" : "https://us-south.ml.cloud.ibm.com"

}

wml_client = APIClient(wml_credentials)

wml_client.spaces.list()

def guide_from_space_name(client,space_name):

   space = client.spaces.get_details()

   return(next(item for item in space['resources']if item['entity']["name"]==
space_name)['metadata']['id'])

space_id="5dd26f3a-4af0-44bb-afe7-e3480e26e4ee"

wml_client.set.default_space(space_id)

software_spec_uid = wml_client.software_specifications.get_id_by_name("default_py3.8")

software_spec_uid

wml_client.repository.download("9f11aa50-fb02-4e2d-acd8-eadcf31852dc","Ibm_model.tar.gz")

wml_client.repository.download("ffd687fb-02bc-472e-b520-a1d2e53d8ffe","Ibm_model1.tar.gz")
```