



Optimizing Spam Filtering with Machine Learning

DONE BY

S.Sangeetha

A.Sangayi

P.Revathi

S.Rubiga

Optimizing Spam Filtering with Machine Learning

1. INTRODUCTION

1.1 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. 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.

Specify the business problem

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 is growing day by day. 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.

Business requirements

A business requirement for an SMS spam classification system would include the ability to accurately identify and flag spam messages, protect customers from unwanted or harmful messages, and comply with industry regulations and laws regarding spam messaging. Additionally, the system should be able to handle a high volume of messages, integrate with existing systems and databases, and provide reporting and analysis capabilities to

track performance and improve the system over time. The system should also have an easy-to-use interface and be easy to maintain

Literature Survey(Student Will Write)

Project would involve researching and analysing existing studies, papers, and articles on the topic to gain a thorough understanding of the current state of SMS spam classification and to identify potential areas for improvement and future research. The survey would include looking at different methods and techniques used for identifying and flagging spam messages, such as machine learning algorithms, natural language processing, and rule-based systems. It would also involve evaluating the performance and effectiveness of these methods, as well as their limitations and challenges. Additionally, the literature survey would review the current state of SMS spam and trends in the industry, as well as any existing laws and regulations related to spam messaging. The survey would also investigate the datasets and feature representations used in previous studies, which would.

Social Impact

It can help protect individuals from unwanted and potentially harmful messages. Spam messages can include phishing attempts, scams, and fraud, which can have serious financial and personal consequences for recipients. By accurately identifying and flagging spam messages, the system can help prevent these types of attacks and protect individuals from falling victim to them. Business Model/Impact:- it can help protect their customers and improve their reputation. Spam messages can harm a business's reputation

and lead to customer complaints and lost business. By accurately identifying and flagging spam messages, the system .

Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Collect the dataset

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset. As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques. There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```
import numpy as np # scientific computation
import pandas as pd # loading dataset file
import matplotlib.pyplot as plt # visulization
import nltk # preprocessing our text
from nltk.corpus import stopwords # removing all the stop words
from nltk.stem.porter import porterstemmer # stemming of words
```

Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called

read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.<https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

```
#Load our dataset
```

```
df = pd.read_csv("spam.csv",encoding="latin")  
df.head()
```

	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

Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- * Handling missing values
- * Handling categorical data
- * Handling Imbalance

Data These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

1.2.purpose

Spam filters are designed to identify emails that attackers or marketers use

to send unwanted or dangerous content. They use specific filtering methods to identify the content of emails or their senders and then flag the email as spam. The email can then be automatically deleted instantly or after a period of time.

Handling missing values

Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.

```
#Give concise summary of a DataFrame  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5572 entries, 0 to 5571  
Data columns (total 5 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   v1          5572 non-null    object  
1   v2          5572 non-null    object  
2   Unnamed: 2   50 non-null      object  
3   Unnamed: 3   12 non-null      object  
4   Unnamed: 4   6 non-null       object  
dtypes: object(5)  
memory usage: 217.8+ KB
```

For checking the null values, `df.isnull()` function is used. To sum those null values we use `.sum()` function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
i]: #Returns the sum fo all na values  
df.isna().sum()
```

```
i]: v1          0  
    v2          0  
    Unnamed: 2   5522  
    Unnamed: 3   5560  
    Unnamed: 4   5566  
    dtype: int64
```

From the above code of analysis, we can infer that columns such as V1 and v2 are not having missing columns,unnamed columns are not required for analysis.

```
df.rename({"v1":"label", "v2":"text"},inplace=True,axis=1)

# bottom 5 rows of the dataframe
df.tail()
```

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will i_b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, ' was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

Handling Categorical Value

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension. In our project,we have text column so we will be using natural language processing for processing the data. Output column is having classes we Converting into 0 and 1 by applying label encoding.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

Handling Imbalance Data

Data Balancing is one of the most important step, which need to be

performed for classification models, because when we train our model on imbalanced dataset ,we will get biased results, which means our model is able to predict only one class element For Balancing the data we are using the SMOTE Method. SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.

```
#Splitting data into train and validation sets using train_test_split

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)

##train size 80% and test size 20%

### Given data is imbalanced one, we are balancing the data

print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {}".format(sum(y_train == 0)))

# import SMOTE module from imblearn Library
# pip install imblearn (if you don't have imblearn in your system)
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train y: {}'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))

Before OverSampling, counts of label '1': 581
Before OverSampling, counts of label '0': 3876

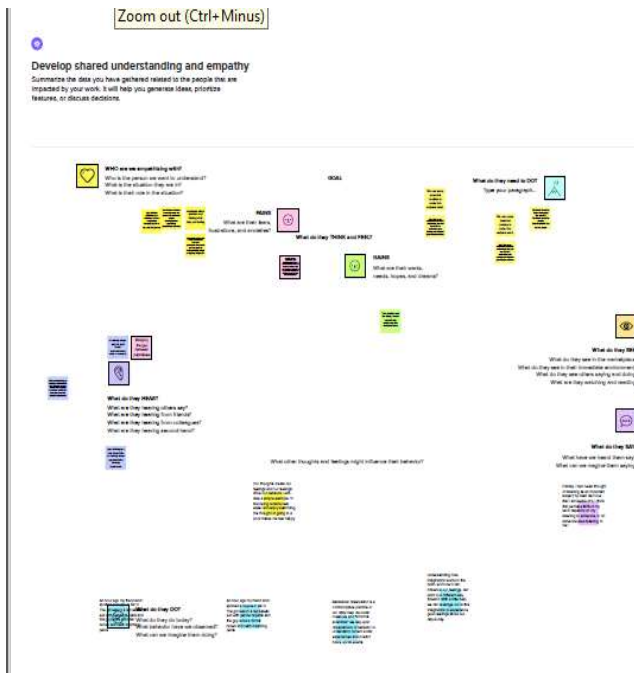
After OverSampling, the shape of train X: (7752, 7163)
After OverSampling, the shape of train y: (7752,)

After OverSampling, counts of label '1': 3876
After OverSampling, counts of label '0': 3876
```

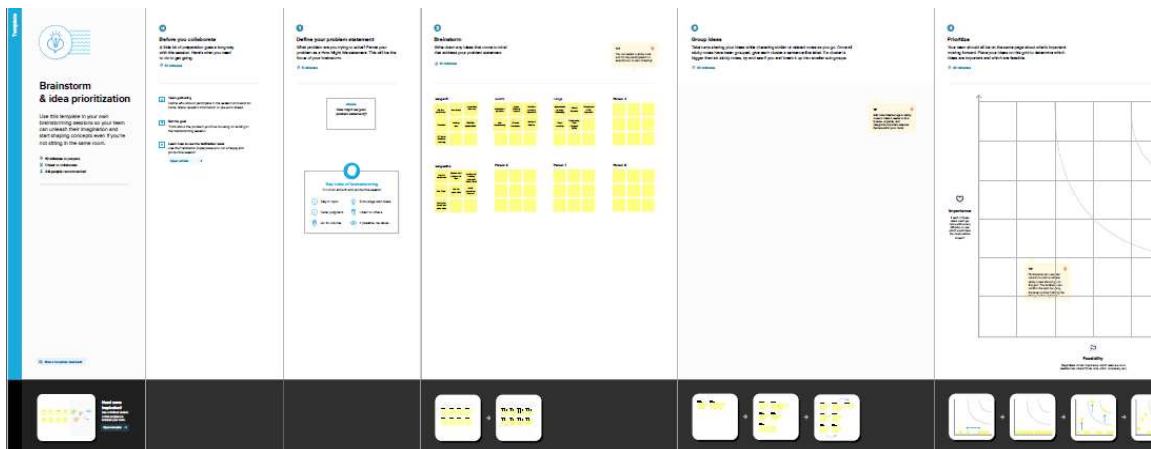
From the above picture, we can infer that ,previously our dataset had 581 class 1, and 3876 class 0 items, after applying smote technique on the dataset the size has been changed for minority class.

2. PROBLEM DEFINITION / DESIGN THINKING

2.1 Empathy Map



2.2 Brainstorming MAP



Cleaning the text data

```

nltk.download("stopwords")

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

True

import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

import re
corpus = []
length = len(df)

for i in range(0,length):
    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)

```

Text pre-processing includes

- *Removing punctuation from the text using regular expression library
- *Converting the sentence into lower case
- *Tokenization – splitting the sentence into words
- * Removing stop words from the data
- *Stemming – stemming is the process of bringing all the words into base form

```

[18]: corpus

[18]: ['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
      'ok lar joke wif u oni',
      'free entri 2 wkli comp win fa cup final tkt 21st may 2005 text fa 87121 receiv entri question std txt rate c appli 08452810 075over18',
      'u dun say earli hor u c already say',
      'nah think goe usf live around though',
      'freemsg hey darl 3 week word back like fun still tb ok xxx std chg send 1 50 rcv',
      'even brother like speak treat like aid patent',
      'per request mell mell oru minnaminungint nurungu vettam set callertun caller press 9 copi friend callertun',
      'winner valu network custom select receivea 900 prize reward claim call 09061781461 claim code kl341 valid 12 hour',
      'mobil 11 month u r entiti updat latest colour mobil camera free call mobil updat co free 08002986030',
      'gonna home soon want talk stuff anymor tonight k cri enough today',
      'six chanc win cash 100 20 000 pound txt csh11 send 87575 cost 150p day 6day 16 tsandc appli repli hl 4 info',
      'urgent 1 week free membership 100 000 prize jackpot txt word claim 81010 c www dbuk net lccltd pobox 4403ldnw1a7rw18',
      'search right word thank breather promis wont take help grant fulfil promis wonder bless time',
      'date sunday',
      'xxomobilemovieclub use credit click wap link next txt messag click http wap xxxomobilemovieclub com n qjkgighjgcbl',
      'oh k watch',
      'eh u namemb 2 spell name ye v naughti make v wet',
      '']

[19]: from sklearn.feature_extraction.text import CountVectorizer
      cv = CountVectorizer(max_features=35000)
      X = cv.fit_transform(corpus).toarray()

```

- * After applying all the above functions, we will get corpus
- * Converting the corpus into Document Term matrix using Count vectorizer

```
import pickle ## importing pickle used for dumping models
pickle.dump(cv, open('cv1.pkl', 'wb')) ## saving to into cv.pkl file
```

Exploratory Data Analysis

Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... Good night... @"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

```
df.shape
```

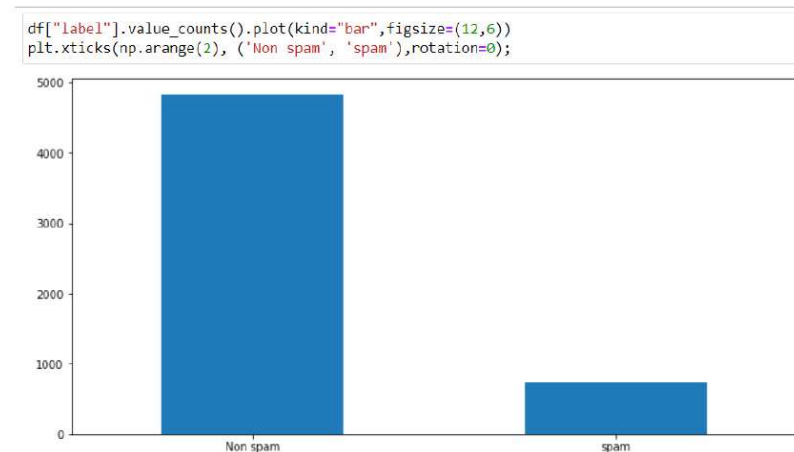
```
(5572, 5)
```

Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate analysis

Simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot. The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features.

Countplot

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot(), so you can compare counts across nested variables. From the graph we can infer that, more data belongs to class 0 than class 1.

Scaling the Data

Scaling is one of the important processes we have to perform on the dataset, because data measures in different ranges can lead to misleading predictions. Models such as KNN, Logistic regression need scaled data, as

they follow distance based method and Gradient Descent concept.

```
# performing feature Scaling operation using standard scaller on X part of the dataset beca
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)

x_bal = pd.DataFrame(x_bal,columns=names)
```

We will perform scaling only on the input values. Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

Splitting data into train and test

Now let's split the Dataset into train and test sets. Changes: first split the dataset into x and y and then split the data set. Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
#Splitting data into train and validation sets using train_test_split

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)

##train size 80% and test size 20%
```

Model Building

Training the model in multiple algorithms

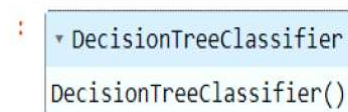
Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four

classification algorithms. The best model is saved based on its performance.

Decision tree model

A function named `decisionTree` is created and train and test data are passed as the parameters. Inside the function, `Decision TreeClassifier` algorithm is initialised and training data is passed to the model with the `.fit()` function. Test data is predicted with `.predict()` function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train_res, y_train_res)
```

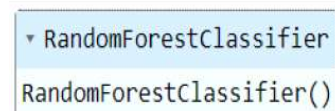


The output of the code cell shows a dropdown menu for `DecisionTreeClassifier` with the text `DecisionTreeClassifier()` below it.

Random forest model

A function named `randomForest` is created and train and test data are passed as the parameters. Inside the function, `Random ForestClassifier` algorithm is initialised and training data is passed to the model with `.fit()` function. Test data is predicted with `.predict()` function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
from sklearn.ensemble import RandomForestClassifier
model1 = RandomForestClassifier()
model1.fit(X_train_res, y_train_res)
```



The output of the code cell shows a dropdown menu for `RandomForestClassifier` with the text `RandomForestClassifier()` below it.

Naïve Bayes model

A function named MultinomialNB is created and train and test data are passed as the parameters. Inside the function, MultinomialNB algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
] : from sklearn.naive_bayes import MultinomialNB
    model = MultinomialNB()

] : #Fitting the model to the training sets
    model.fit(X_train_res, y_train_res)

] :
    ▾ MultinomialNB
    MultinomialNB()
```

ANN model

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.


```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

#Fitting the model to the training sets
model = Sequential()

X_train.shape
(4457, 7163)

model.add(Dense(units = X_train_res.shape[1],activation="relu",kernel_initializer="random_uniform"))
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
model.add(Dense(units=1,activation="sigmoid"))

model.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])

generator = model.fit(X_train_res,y_train_res,epochs=10,steps_per_epoch=len(X_train_res)//64)

generator = model.fit(X_train_res,y_train_res,epochs=10,steps_per_epoch=len(X_train_res)//64)

Epoch 1/10
121/121 [=====] - 26s 203ms/step - loss: 0.1404 - accuracy: 0.9565
Epoch 2/10
121/121 [=====] - 24s 199ms/step - loss: 0.0220 - accuracy: 0.9950
Epoch 3/10
121/121 [=====] - 23s 188ms/step - loss: 0.0155 - accuracy: 0.9985
Epoch 4/10
121/121 [=====] - 23s 194ms/step - loss: 0.0163 - accuracy: 0.9962
Epoch 5/10
121/121 [=====] - 24s 195ms/step - loss: 0.0158 - accuracy: 0.9965
Epoch 6/10
121/121 [=====] - 23s 198ms/step - loss: 0.0132 - accuracy: 0.9974
Epoch 7/10
121/121 [=====] - 23s 198ms/step - loss: 0.0130 - accuracy: 0.9972
Epoch 8/10
121/121 [=====] - 24s 196ms/step - loss: 0.0120 - accuracy: 0.9974
Epoch 9/10
121/121 [=====] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [=====] - 11s 1s - loss: 0.0120 - accuracy: 0.99724688106;tensorflow: your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least steps_per_epoch * epochs batches (in this case, 1210 batches). You may need to use the repeat() function when building your dataset.
121/121 [=====] - 23s 193ms/step - loss: 0.0128 - accuracy: 0.9972

```

Testing the model

```

[45]: y_pred=model.predict(X_test)
      y_pred

35/35 [=====] - 2s 29ms/step
[45]: array([[1.5844109e-15],
            [4.4117190e-04],
            [1.1517670e-18],
            ...,
            [2.0001259e-08],
            [3.8018154e-17],
            [1.2099350e-12]], dtype=float32)

[46]: y_pr = np.where(y_pred>0.5,1,0)

[49]: y_test

[49]: array([0, 0, 0, ..., 0, 0, 0])

[50]: from sklearn.metrics import confusion_matrix,accuracy_score
      cm = confusion_matrix(y_test,y_pr)
      score = accuracy_score(y_test,y_pr)
      print(cm)
      print('Accuracy Score is: ',score*100)

[[937 12]
 [ 16 150]]
Accuracy Score is: 97.48878923790816

```

This code defines a function named "new_review" which takes in a new_review as an input. The function then converts the input new_review from a list to a numpy array. It reshapes the new_review array as it contains only one record. Then, it applies feature scaling to the reshaped new_review array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns

the prediction of the classifier on the scaled new_review.

Performance Testing & Hyperparameter Tuning

Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score

Compare the model

For comparing the above four models, the compareModel function is defined.

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test, y_pred)
print(cm)
print('Accuracy Score Is Naive Bayes:- ', score*100)
```

```
[[935  14]
 [ 13 153]]
Accuracy Score Is:- 97.57847533632287
```

```

cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test, y_pred)
print(cm)
print('Accuracy Score Is:- ', score*100)

cm1 = confusion_matrix(y_test, y_pred1)
score1 = accuracy_score(y_test, y_pred1)
print(cm1)
print('Accuracy Score Is:- ', score1*100)

[[796 153]
 [ 17 149]]
Accuracy Score Is:- 84.75336322869956
[[855  94]
 [ 14 152]]
Accuracy Score Is:- 90.31398134529148

121/121 [=====] - 24s 196ms/step - loss: 0.0128 - accuracy: 0.9974
Epoch 9/10
121/121 [=====] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [=====>...] - ETA: 1s - loss: 0.0128 - accuracy: 0.9972WARNING:tensorflow:
|: from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test, y_pr)
print(cm)
print('Accuracy Score Is:- ', score*100)

[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816

```

After calling the function, the results of models are displayed as output. From the five models ANN is performing well. From the below image, We can see the accuracy of the model. ANN is giving the 99.72% accuracy for the training data and 97.48 for testing data.

Comparing model accuracy before & after applying hyperparameter tuning

Evaluating performance of the model From sklearn, `cross_val_score` is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by `model.save("model.h5")`

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test, y_pr)
print(cm)
print('Accuracy Score Is:- ', score*100)
```

```
[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816
```

Model Deployment

The best model

The best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

Saving our model

By comparing the all the model , we can come to a conclusion that ANN is the best model

```
] : model.save('spam.h5')
```

Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks.

- *Building HTML Pages

- * Building server side script

- * Run the web application

Building Html Pages

For this project create two HTML files namely

- * index.html

- * spam.html

- * result.html and save them in the templates folder.

Build Python code

```
# Importing essential libraries
from flask import Flask, render_template, request
import pickle
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from tensorflow.keras.models import load_model
# Load the Multinomial Naive Bayes model and CountVector
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application.

Flask constructor takes the name of the current module (`__name__`) as argument.

```
# Load the Multinomial Naive Bayes model
loaded_model = load_model('spam.h5')
cv = pickle.load(open('cv1.pkl', 'rb'))
app = Flask(__name__)
```

Render HTML page:

```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier. In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/Spam', methods=['POST', 'GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('spam.html')

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        message = request.form['message']
        data = message

        new_review = str(data)
        print(new_review)
        new_review = re.sub('[^a-zA-Z]', ' ', new_review)
        new_review = new_review.lower()
        new_review = new_review.split()
        ps = PorterStemmer()
        all_stopwords = stopwords.words('english')
        all_stopwords.remove('not')
        new_review = [ps.stem(word) for word in new_review if not word in set(all_stopwords)]
        new_review = ' '.join(new_review)
        new_corpus = [new_review]
        new_X_test = cv.transform(new_corpus).toarray()
        print(new_X_test)
        new_y_pred = loaded_model.predict(new_X_test)
        new_X_pred = np.where(new_y_pred>0.5, 1, 0)
        print(new_X_pred)
        if new_review[0][0]==1:
            return render_template('result.html', prediction="Spam")
        else:
            return render_template('result.html', prediction="Not a Spam")
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__ == "__main__":
    # app.run(host='0.0.0.0', port=8000, debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)

(base) D:\thesmartBridge\Projects\2. DrugClassification\Drug c
Serving Flask app "app" (lazy loading)
Environment: production
WARNING: This is a development server. Do not use it in a p
Use a production WSGI server instead.
Debug mode: off
Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

3. RESULT

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.
- This is the main page of Spam Detection , where you can know about the project and also
 - from this page users can click onto the spam button and they will redirect onto the spam/prediction page for providing the inputs.
 - Spam Detection About Page.

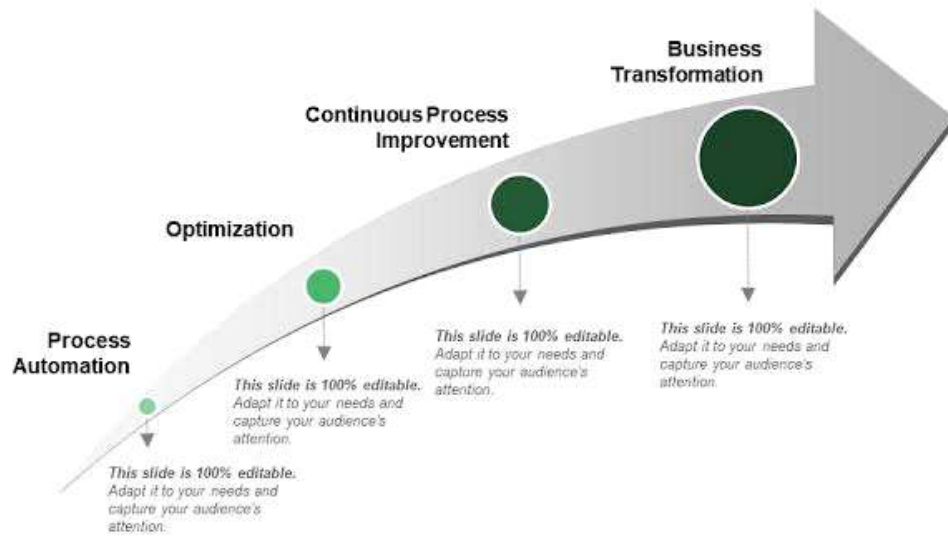
Process optimization benefits



zapier



Business Outcomes Process Automation Optimization...





4. ADVANTAGES & DISADVANTAGES

Advantages

To study the background literature on spam classification methods and working of DCA and KHO algorithm in detail. To propose the design methodology in order to apply relevant feature selection methods using KH Optimization. To identify the role of KH in DCA for optimization and to identify spam messages from the sample dataset by following risk concentration using DCA. To evaluate and compare different classifiers in terms of Recall, Precision, Accuracy and F-measure with 18 different optimization function. To measuring time complexity with combination of

DCA and KH Algorithm. Gravity filters have advantages of extreme simplicity, needing only simple accessories, low first cost and can be made of almost any material.

Disadvantages

Thousands of spam emails may reach Inboxes before a spammer's email address, IP or domain is blacklisted. Spam filtering is machine-based so there is a room for mistakes called “false positives.” Bayesian filters may be fooled by spammers, e.g. in a case of using large blocks of legitimate text. What are the disadvantages of filtering? The main disadvantages of barrier-based filtration are: Replacement and disposal costs. When this type of filter becomes blocked by waste particles, it needs to be replaced.

Over time this can be costly both in terms of purchasing replacement filters, as well as the downtime required to remove and replace used filters. Disadvantages of Filtration. Filters can only work on liquids and gasses. Autoclaving is usually cheaper than filtration since filters are expensive to replace, especially nano-filters. Glass filters are very brittle and can break easily. Membrane filters rupture easily. Over time this can be costly both in terms of purchasing replacement filters, as well as the downtime required to remove and replace used filters.

Relatively low rate of filtration. Excessive floor area needed and high labour charges If the amount of particulate matter to be removed is too small or it is finely divided, sand filter will not remove the suspended solids. In processes involving organic materials there may be danger of bacterial infection from an infected process-water supply and the sand filter cannot

remove the bacteria as such. In these cases a coagulant like ferrous sulphate or aluminium sulphate is added to the water before filtration. These are hydrolysed by the alkalinity of most normal waters with the formation of a flocculant precipitate of iron or aluminium hydroxide. This precipitate adsorbs finely divided suspended matter and even bacteria, even if added to the water in very small amounts. The resultant flocs, though fine, are removed by the sand filters.

5. APPLICATIONS

Spam detection is a constant worry for managed services providers (MSPs) and IT professionals. The simple user error of opening a malicious attachment can cause devastating effects, from compromising sensitive data to bringing down an entire network and its servers. N-able™ Mail Assure allows MSPs to provide their clients with layered security spam filtering approach that is simple and comprehensive. Cyberattacks are becoming more sophisticated as time goes on, making it more important than ever for MSPs to provide reliable and effective security solutions to their customers. Mail Assure not only blocks spam emails, it also offers malware, virus, and spear phishing prevention, which helps ensure your customers' safety and productivity.

6. CONCLUSION

In this paper, we show that an adversary can effectively disable the SpamBayes spam filter with relatively little system state information and relatively limited control over training data. Our Usenet dictionary attack causes misclassification of 36% of ham messages with only 1% control over

the training messages, rendering SpamBayes unusable. Our focused attack changes the classification of the target message 60% of the time with knowledge of only 30% of the target's tokens. We also explore two successful defenses. The RONI defense filters out dictionary attack messages with complete success. The dynamic threshold defense also mitigates the effect of the dictionary attacks. Focused attacks are especially difficult to defend against because of the attacker's extra knowledge; developing effective defenses is an important open problem. In future work, we intend to continue refining our defenses.

Our attacks and defenses should also apply to other spam filtering systems based on similar learning algorithms, such as Bogofilter and the Bayesian component of SpamAssassin although their effect may vary. Similar techniques may also be effective against other learning systems, such as worm or intrusion detection.

7. FUTURE SCOPE

* The Future of Anti-Spam Protection:- When it comes to modern computing, the future is now. But what about the future of spam protection? In this post, we're going to peer into the future of anti-spam measures.

* More Accurate Detection:- First of all, it's safe to say that the spam filters that are at work protecting today's email accounts will continue to improve, until they're operating at a near 100 per cent success rate. You'll be able to spend less time sifting through potentially spammy messages, because they simply aren't going to make it to your inbox in the first place.

* Of course, spammers' tactics are sure to evolve with the times as well,

making it difficult to catch every single attempt at spam. However, it's perhaps even more important that the spam filters of the future become better at identifying those messages that are not considered spam.

* As an example, consider that some 7 per cent of emails sent by non-profit organisations today are incorrectly identified as spam. That's not a terribly high figure, but it's unfortunate that these well-meaning institutions have to be penalised due to the unscrupulous activities of others.

* In the future, spam filters will be more intelligent than ever, correctly distinguishing safe emails from those that need to be removed from the inbox. And that's going to be possible thanks – in large part – to the ongoing work of companies like MailCleaner.

* Spam Protection Goes Virtual:-To be fair, virtual and augmented reality are sure to become a part of virtually everything that we do with our computers, mobile devices and Internet-based activities. All the same, we're already seeing hints of how our struggle against spam might look in a more virtual environment.

* As an example, take Microsoft's cutting-edge HoloLens – an augmented reality (AR) platform that is just now entering the developer's market. Microsoft recently released an AR version of Outlook Mail for the headset.

* The major innovation of this platform is that it allows you to free up your desktop by moving your emails into an augmented-reality space. You could, for example, move your spam folder to a virtual position on the wall to the side of your desk, so that you can glance over there to see what's happening

in the folder without allowing it to clutter your actual desktop space.

* Spam Detection in Real Time:- We're already seeing breakthroughs in real-time spam detection, and this is only likely to continue. A good example is the Twitter app Periscope, which allows users to broadcast events in their lives in real time so that other Periscope users can join in the fun.

* The problem is that spammers like to join trending events so that they can unleash their pesky wares on the comment section. But Periscope is putting a stop to this by allowing users to flag spam in real time, with enough red flags leading the company to shut down the offending account.

8.APPENDIX

Electronic mail has eased communication methods for many organisations as well as individuals. This method is exploited for fraudulent gain by spammers through sending unsolicited emails. This article aims to present a method for detection of spam emails with machine learning algorithms that are optimized with bio-inspired methods. A literature review is carried to explore the efficient method applied on different datasets to achieve good results. An extensive research was done to implement machine learning models using Naïve Bayes, Support Vector Machine, Random Forest, Decision Tree and Multi-Layer Perceptron on seven different email datasets, along with feature extraction and pre-processing. The bio-inspired algorithms like Particle Swarm Optimization and Genetic Algorithm were implemented to optimize the performance of classifiers. Multinomial Naïve Bayes with Genetic Algorithm performed the best overall. The comparison

of our results with other.