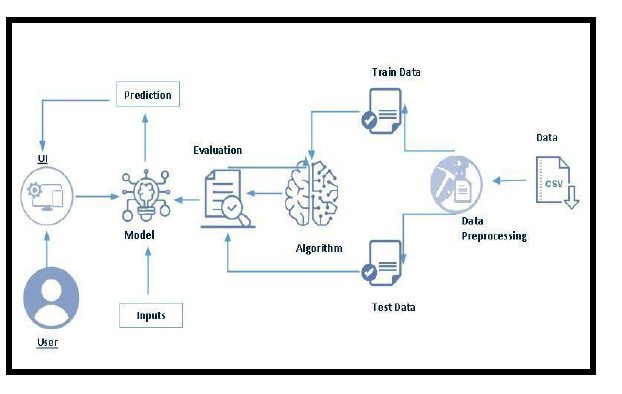
**Project TiTle**

**Optimizing Spam Filtering with Machine Learning**

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.

**Technical Architecture**



**Project Flow:**

● User interacts with the UI to enter the input.

● Entered input is analysed by the model which is integrated.

● Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

● Define Problem / Problem Understanding

○ Specify the business problem

○ Business requirements

○ Literature Survey

○ Social or Business Impact.

● Data Collection & Preparation

○ Collect the dataset

○ Data Preparation

● Exploratory Data Analysis

○ Descriptive statistical

○ Visual Analysis

● Model Building

○ Training the model in multiple algorithms

○ Testing the model

● Performance Testing & Hyperparameter Tuning

○ Testing model with multiple evaluation metrics

○ Comparing model accuracy before & after applying hyperparameter tuning

● Model Deployment

○ Save the best model

○ Integrate with Web Framework

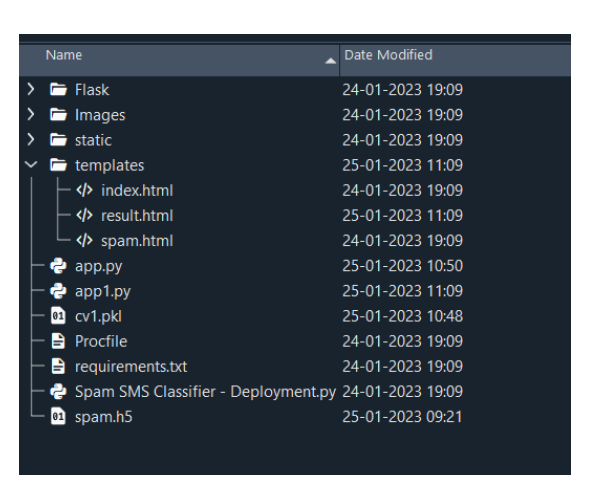
● Project Demonstration & Documentation

○ Record explanation Video for project end to end solution

○ Project Documentation-Step by step project development procedure

**Project Structure:**

Create the Project folder which contains files as shown below



* We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
* Spam.h5 is our saved model. Further we will use this model for flask integration.

**Milestone 1: Define Problem / Problem Understanding**

**Activity 1: Specify the business problem**

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.

**Activity 2: 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 and update.

**Activity 3: Literature Survey**

Project would involve researching and analyzing 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 help to determine the best approach for the current project. Furthermore, It would be important to check the pre-processing techniques used in the research to understand how to properly clean and prepare the data for the classifier

**Activity 4: Social or Business Impact.**

**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 can help protect businesses and improve their customer's trust.

**Milestone 2: 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.

**Activity 1: Collect the dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI

repository, etc.

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.

Link: <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

**Note:** 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.

**Activity 1.1: Importing the libraries**

Import the necessary libraries as shown in below

#impoerting the libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import nltk

from ntlk.corpus import stopwords

from ntlk.stem.porter import PorteStemmer

**Activity 1.2: 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.

data=pd.read\_csv("E:\\NMDS\\spam.csv",encoding="latin")

data.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 |

**Activity 2: 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.

1. Handling missing values
2. Handling categorical data
3. Handling Imbalance Data

Note: 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.

**Activity 2.1: 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.

**data.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: 108.9+ 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.

**data.isna().sum()**

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

Renaming the columns according the requirement

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

**data.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 Ì\_ 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

**Activity 2.2: Handling Categorical Values**

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()**

**data['label']=le.fit\_transform(data['label'])**

**Activity 2.3: 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 biassed 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.

**from sklearn.model\_selection import train\_test\_split**

**X\_train,X\_text,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=0)**

**Milestone 3: Exploratory Data Analysis**

**Activity 1: 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.

**data.describe()**

**label**

count 5572.000000

mean 0.134063

std 0.340751

min 0.000000

25% 0.000000

50% 0.000000

75% 0.000000

max 1.000000

**data.shape**

(5572, 5)

**Activity 2: 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.

**Activity 2.1: Univariate analysis**

In 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 class 0 than class 1

**Scaling the Data**

Scaling is one the important process, we have to perform on the dataset, because of data

measures in different ranges can leads to mislead in prediction

Models such as KNN, Logistic regression need scaled data, as they follow distance based

method and Gradient Descent concept.

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.

**Milestone 4: Model Building**

**Activity 1: 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.

**Activity 1.1: Decision tree model**

A function named decisionTree is created and train and test data are passed as the

parameters. Inside the function, DecisionTreeClassifier 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.

**rom sklearn.tree import DecisionTreeClassifier**

**model=DecisionTreeClassifier()**

**model=fit.(X\_train\_res,y\_train\_res)**

**Activity 1.2: Random forest model**

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier 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.

**Activity 1.3: 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.

**Activity 2: Testing the model**