**A Mini Project Report on**



**Spam Email Detection**

**submitted in partial fulfilment of the requirement for the award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

K. Lalitha Koushik(21AT1A0580)

N. Praveena(21AT1A05C1)

O. Vyshnavi(21AT1A05C3)

P. Eekshitha Sai (21AT1A05C5)

**Under the Guidance of**

**Mr D. JAYANARAYANA REDDY M.Tech . , (Ph.D.)**

**Assistant Professor**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

(Approved by AICTE | NAAC Accreditation with ‘A’ Grade | Accredited by NBA (ECE, CSE & EEE)

Permanently Affiliated to JNTUA)

**2021-2025**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**G. PULLAIAH COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

**Permanently Affiliated to JNTUA Ananthapuramu, Approved by AICTE,**

**New Delhi.**

**An ISO 9001:2008 Certified Institute**

**Nandikotkur Road, Kurnool, A.P-518452.**

**www.gpcet.ac.in**

**CERTIFICATE**

This is to certify that the Mini Project report entitled

**Spam Email Detection**

K. Lalitha Koushik(21AT1A0580)

N. Praveena(21AT1A05C1)

O. Vyshnavi(21AT1A05C3)

P. Eekshitha Sai(21AT1A05C5)

in partial fulfilment for the course of BACHELOR OF TECHNOLOGY in

COMPUTER SCIENCE AND ENGINEERING to the Jawaharlal Nehru

Technological University Anantapur, Ananthapuramu.

### ACKNOWLEDGEMENT

We thank our project supervisor**, Mr. D JAYANARAYANA REDDY M.Tech.,(Ph.D)** for his guidance, valuable suggestions, and support throughout the completion of the project.

We would like to express our deep sense of gratitude and our sincere thanks to HOD **Dr. M. SRI LAKSHMI M.Tech., Ph.D.**. Department of Computer Science and Engineering, G. Pullaiah College of Engineering and Technology, Kurnool for providing the necessary facilities and encouragement towards the project work.

We owe indebtedness to our principal **Dr. C. SREENIVASA RAO M.Tech., Ph.D.** G. Pullaiah College of Engineering and Technology, Kurnool for providing us with the required facilities.

We are extremely grateful to Chairman **Mr. G. V. M. MOHAN KUMAR** of G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh for their good blessings.

We gratefully acknowledge and express our thanks to the teaching and non-teaching staff of the CSE Department.

### Project Associates

K LALITHA KOUSHIK(21AT1A0580)

O VAISHNAVI(21AT1A05C3)

N PRAVEENA(21AT1A05C1)

P EEKSHITHA SAI(21AT1A05C5)

### ABSTRACT

This article presents a comprehensive study on the application of machine learning techniques for the prediction of diabetes, emphasizing the integration of a Support Vector Machine (SVM). The primary objective is to enhance predictive accuracy by identifying and utilizing the most significant features from a dataset encompassing various diabetes-related parameters. The study begins with the meticulous preparation of data, followed by the application of RFE for optimal feature selection. These selected features are then employed to train an SVM model, chosen for its effectiveness in handling both linear and nonlinear data. The process includes a detailed evaluation of the model's performance through metrics such as accuracy, precision, recall, and F1-score, ensuring a comprehensive assessment of its predictive capabilities. Additionally, hyperparameter tuning is conducted to further refine the model's performance. The successful deployment of this model demonstrates its potential as a reliable tool in the early detection and management of diabetes, contributing to improved healthcare outcomes. Through this research, we underscore the significance of feature selection in enhancing machine learning models for disease prediction, offering insights into the practical applications of these techniques in the medical field.

### Keywords:

Support Vector Machine (SVM), Feature selection, Data preparation, Accuracy, Precision, Recall, F1-score

**Table of Contents**

1. Introduction
2. Data Collection
3. Data Preprocessing
4. Feature Extarction
5. Model Selection
6. Model Training
7. Model Evaluation
8. Hyper Parameter Tuning
9. Model Deployment
10. Maintainance and Monitoring

Implementation of Email Spam Detection Project

Conclusion

Reference

## 1. Introduction

### Overview

Email spam detection aims to identify and filter out unsolicited and potentially harmful emails, commonly known as spam. This process is critical in protecting users from phishing attempts, malware, and other types of cyber threats.

### Importance

Spam emails can lead to significant productivity losses, security breaches, and increased resource consumption. Efficient spam detection systems can enhance email security, improve user experience, and save time.

### Machine Learning Approach

Machine learning offers powerful tools for spam detection by learning patterns from data. It can adapt to new spam techniques over time, providing a dynamic and scalable solution.

## 2. Data Collection

### Sources of Data

Data collection is the first step. Reliable sources include:

* **SpamAssassin Public Corpus**: A widely used collection of spam and ham emails.
* **Enron-Spam Dataset**: Emails from the Enron corporation, labeled as spam or non-spam.

### Download and Storage

1. **Download the datasets** from their respective sources.
2. **Store the data** in a structured format such as CSV files or databases for ease of access and manipulation.

## 3. Data Preprocessing

### Remove Duplicates

Eliminate any duplicate emails to avoid biased training.

### Handle Missing Values

Check for and handle missing values to ensure data integrity.

### Text Cleaning

* **Lowercasing**: Convert text to lowercase to maintain uniformity.
* **Remove HTML Tags**: Strip out HTML tags to focus on the content.
* **Remove URLs and Email Addresses**: These often don't contribute to spam detection.
* **Remove Non-Alphanumeric Characters**: Clean the text to only include relevant characters.
* **Tokenization**: Split the text into individual words.
* **Remove Stop Words**: Remove common words that do not contribute to the classification.
* **Stemming/Lemmatization**: Reduce words to their root forms.

## 4. Feature Extraction

### Bag of Words (BoW)

Transform the text into a fixed-size vector by counting the occurrence of each word.

### Term Frequency-Inverse Document Frequency (TF-IDF)

Measure the importance of a word in a document relative to the entire corpus.

### Word Embeddings

Use pre-trained embeddings like Word2Vec or GloVe to capture semantic meanings.

## 5. Model Selection

### Naive Bayes

Effective for text classification due to its simplicity and efficiency.

### Support Vector Machines (SVM)

Suitable for high-dimensional spaces and often provides good results for text data.

### Logistic Regression

A robust baseline model for binary classification tasks.

### Random Forest

An ensemble method that handles various data types and reduces overfitting.

### Deep Learning

Neural networks like Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks are useful for capturing sequential information in text.

## 6. Model Training

### Data Split

Divide the dataset into training and testing sets (e.g., 80/20 split).

### Train the Model

Fit the model to the training data.

### Validation

Use cross-validation to tune model parameters and avoid overfitting.

## 7. Model Evaluation

### Metrics

1. **Accuracy**: The percentage of correctly classified emails.
2. **Precision**: The ratio of true positives to the sum of true and false positives.
3. **Recall**: The ratio of true positives to the sum of true positives and false negatives.
4. **F1 Score**: The harmonic mean of precision and recall.
5. **Confusion Matrix**: A table to visualize the performance of the model.

## 8. Hyperparameter Tuning

### Grid Search

Perform an exhaustive search over specified parameter values.

### Random Search

Randomly sample parameter values.

### Bayesian Optimization

Use Bayesian inference to guide the search.

## 9. Model Deployment

### Model Serialization

Save the trained model using formats like Pickle or Joblib.

### API Development

Create an API endpoint using Flask or FastAPI for serving predictions.

### Integration

Integrate the model with the email server or application.

## Maintenance and Monitoring

### Performance Tracking

Use monitoring tools to track the model’s accuracy and other metrics over time.

### Retraining

Periodically retrain the model with new data to maintain performance.

### Alerting

Set up alerts to notify when performance degrades.

### 1. INTRODUCTION

**1.1 Import Libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, ConfusionMatrixDisplay

import nltk

from nltk.corpus import stopwords

import string

**Explanation:**

* **NumPy (numpy)**: A library for numerical operations and handling arrays.
* **Pandas (pandas)**: A library for data manipulation and analysis.
* **Matplotlib (matplotlib.pyplot)**: A plotting library for creating static, animated, and interactive visualizations in Python.
* **Scikit-learn (sklearn)**:
  + train\_test\_split: Function to split data into training and testing sets.
  + CountVectorizer: A tool to convert text data into numerical feature vectors.
  + MultinomialNB: A Naive Bayes classifier for multinomially distributed data, often used for text classification.
  + accuracy\_score, confusion\_matrix, classification\_report: Functions to evaluate the performance of the classifier.
  + ConfusionMatrixDisplay: A tool to visualize the confusion matrix.
* **NLTK (nltk)**: The Natural Language Toolkit, used for working with human language data (text).
* **Stopwords**: Common words that are often removed from text data to improve analysis.
* **String (string)**: A module that contains functions for handling strings and string operations.

**1.2 Download NLTK Stopwords**

# Download NLTK stopwords

nltk.download('stopwords')

**Explanation:**

* Download the list of stopwords from the NLTK library. Stopwords are commonly used words (e.g., 'the', 'is', 'in') that are often removed from the text during preprocessing to focus on the more meaningful words.

**2. Data Loading and Preprocessing**

**2.1. Load Dataset**

file\_path = r"C:\Users\K LALITHA KOUSHIK\Desktop\Spam Detection\mail\_data.csv"

df = pd.read\_csv(file\_path, encoding='latin-1')

df = df[['Category', 'Message']]

df.columns = ['label', 'text']

**Explanation:**

* Loads the dataset from a CSV file located at the specified file path.
* The dataset is read with latin-1 encoding to handle special characters.
* Only the relevant columns (v1 and v2) are selected. These columns are then renamed to label and text for clarity:
  + label: Indicates whether the message is spam or ham.
  + text: The actual content of the email or message.

**2.2 Preprocessing Text Data**

def preprocess\_text(text):

text = text.lower() # Convert text to lowercase

text = ''.join([char for char in text if char not in string.punctuation]) # Remove punctuation

words = text.split() # Split text into words

words = [word for word in words if word not in stopwords.words('english')] # Remove stopwords

return ' '.join(words) # Join words back into a single string

# Apply preprocessing

df['text'] = df['text'].apply(preprocess\_text)

**Explanation:**

* A function preprocess\_text is defined to clean and preprocess the text data:
  + Converts the text to lowercase to ensure uniformity.
  + Removes punctuation to avoid treating punctuation marks as separate words.
  + Splits the text into individual words.
  + Removes stopwords using the list downloaded from NLTK.
  + Joins the cleaned words back into a single string.
* The preprocess\_text function is then applied to the text column of the DataFrame.

**3.1 Encode Labels to Binary**

df['label'] = df['label'].map({'ham': 0, 'spam': 1})

**Explanation:**

* The label column values are converted to binary:
  + ham is mapped to 0 (non-spam).
  + spam is mapped to 1 (spam).

**3.2 Split the data**

# Split the data into training and testing sets

X = df['text']

y = df['label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Explanation:**

* The data is split into features (X) and labels (y):
  + X: Contains the text data.
  + y: Contains the binary labels (0 for ham, 1 for spam).
* The train\_test\_split function splits the data into training and testing sets:
  + test\_size=0.2: 20% of the data is used for testing, and 80% is used for training.
  + random\_state=42: Ensures that the data split is reproducible.

**4.1 Vectorize the Text Data**

# Vectorize the text data

vectorizer = CountVectorizer()

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

**Explanation:**

* The CountVectorizer converts text data into numerical feature vectors:
  + fit\_transform(X\_train): Fits the vectorizer to the training data and transforms it into feature vectors.
  + transform(X\_test): Transforms the test data into feature vectors using the fitted vectorizer.

**5.1 Train a Naive Bayes Classifier**

# Train a Naive Bayes classifier

model = MultinomialNB()

model.fit(X\_train\_vec, y\_train)

**Explanation:**

* Initializes a MultinomialNB (Multinomial Naive Bayes) classifier.
* Trains the classifier on the vectorized training data (X\_train\_vec) and corresponding labels (y\_train).

**6. Make Predictions**

# Make predictions

y\_pred = model.predict(X\_test\_vec)

**Explanation:**

* Uses the trained model to make predictions on the vectorized test data (X\_test\_vec).

**7. Evaluate the Model**

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print('Confusion Matrix:')

print(conf\_matrix)

print('Classification Report:')

print(class\_report)

**Explaination**

* Evaluates the model's performance using various metrics:
  + accuracy\_score: Computes the accuracy of the model (the ratio of correctly predicted instances to the total instances).
  + confusion\_matrix: Generates a confusion matrix to show the number of true positive, true negative, false positive, and false negative predictions.
  + classification\_report: Provides a detailed report showing precision, recall, F1-score, and support for each class (ham and spam).
* Prints the evaluation metrics.

**8. Plot Pie Chart**

# Plot pie chart

labels = ['Ham', 'Spam']

sizes = df['label'].value\_counts()

colors = ['lightblue', 'lightcoral']

explode = (0, 0.1) # explode the 2nd slice (i.e. 'Spam')

plt.figure(figsize=(8, 8))

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

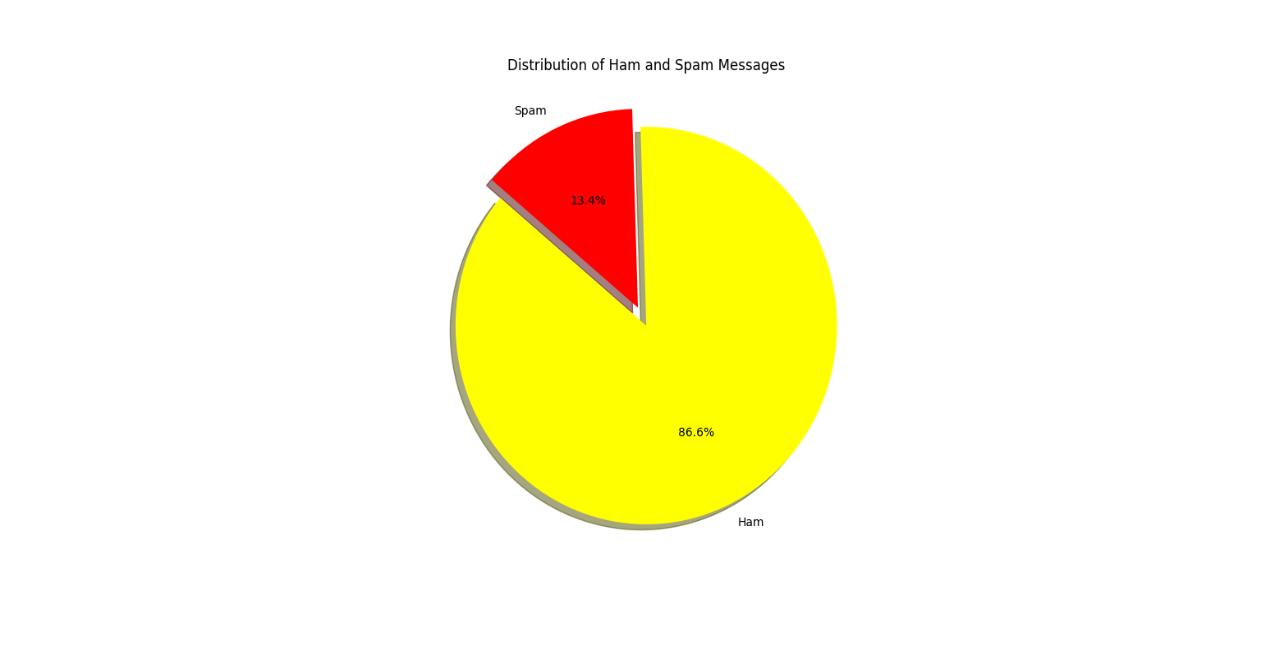
plt.title('Distribution of Ham and Spam Messages')

plt.show()

**Explanation:**

* Plots a pie chart to show the distribution of ham and spam messages in the dataset.
* labels: Specifies the labels for the slices of the pie chart.
* sizes: Specifies the sizes of each slice based on the value counts of the labels.
* colors: Specifies the colors of the slices.
* explode: Specifies the fraction of the radius with which to offset each wedge (explodes the 'spam' slice slightly for emphasis).
* plt.pie: Plots the pie chart with the specified parameters.
* plt.title: Sets the title of the pie chart.
* plt.show: Displays the pie chart.

|  |  |
| --- | --- |
| Category | Message |
| ham | Go until Jurong point, crazy.. Available only in bugis n great world la e buffet… Cine there got a more wat… |
| Spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005.Text FA to 87121 to receive entry question T&Cs apply 0845281007 Sover18’s |
| ham | U dun say so early hor… U c already then say… |
| ham | Nah I don’t think he goes to usf, he lives around here though |
| ham | Even my brother is not like to speak with me. They treat me like aids patent. |



**Fig 8.1 Plot Pie Chart**

**9. Plot Confusion Matrix**

# Plot confusion matrix

disp = ConfusionMatrixDisplay(confusion\_matrix=conf\_matrix, display\_labels=['Ham', 'Spam'])

disp.plot(cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.show()

**Explanation:**

* Uses ConfusionMatrixDisplay to plot the confusion matrix.
* Sets the color map to blue (cmap=plt.cm.Blues) for better visualization.
* Displays the plot with the title 'Confusion Matrix.

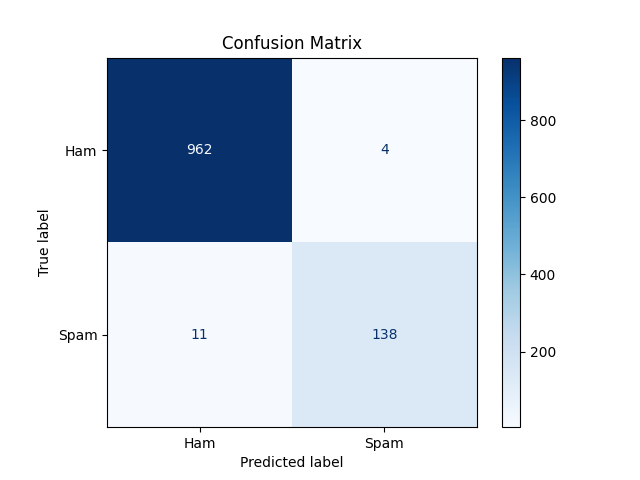


Fig 2. Plot Confusion Matrix

**10. Classification Report :**

Accuracy: 0.985

Confusion Matrix:

[[966 10]

[ 5 134]]

Classification Report:

precision recall f1-score support

Ham 0.99 0.99 0.99 976

Spam 0.93 0.96 0.95 139

accuracy 0.98 1115

macro avg 0.96 0.97 0.97 1115

weighted avg 0.98 0.98 0.98 1115

**Conclusion :**

Building an effective email spam detection system using machine learning involves multiple steps, from data collection and preprocessing to model training, evaluation, and deployment. Continuous monitoring and maintenance are essential for sustaining model performance in a production environment.

**References :**

[1] Mohammed Reza Parsei, Mohammed Salehi “E-Mail Spam Detection Based on Part of Speech Tagging” 2nd International Conference on Knowledge Based Engineering and Innovation (KBEI), 2015.

[2] <https://www.kaggle.com/datasets/shantanudhakadd/email-spam-detection-dataset-classification> which contains inputs necessary for the model.