**NLP**

**MINI PROJECT**

**Spam Detection Using Machine Learning**

**Prepared By:**

**Abdul khuddus**

**Master of Computer Application (MCA)**

**28-10-2024**

**2.Contents:**

1. Title Page
2. Contents (Index)
3. Abstract
4. Objective
5. Implementation 5.1 Code
6. Results o 6.1 Output Screenshots

**3.Abstract:**

In today’s digital age, spam messages are pervasive, affecting users through email, messaging applications, and social media. This project aims to build a machine learning-based spam detection system capable of classifying messages as either spam or legitimate (ham). Using a dataset of label messages, the system applies Natural Language Processing (NLP) techniques to transform text data and a Naive Bayes classifier to categorize each message. This project leverages the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique to process text data and Naive Bayes, a commonly used algorithm for text classification due to its simplicity and effectiveness.

**4.Objective:**

The primary objective of this project is to design a simple, yet effective, spam detection system that can classify messages based on their content. The model should be able to identify patterns associated with spam messages and distinguish them from legitimate messages.

Specific objectives include:

1. Implementing text preprocessing and vectorization with TF-IDF to transform messages.
2. Training a Naive Bayes classifier on a dataset of spam and ham messages.
3. Testing the model's performance to assess accuracy and classification metrics.
4. Creating a user-friendly application for users to classify messages in real-time.

**5.Implementation:**

**Sample Dataset:**

data = {

"message": [

"Free money now!!!",

"Hey, are we meeting tomorrow?",

"Earn cash fast!",

"Lunch at 12?",

"You won a prize, claim now!",

"Can we discuss the project details?",

"Get a loan with no credit check.",

"Please review the attached document.",

"Congratulations! You've been selected for a special prize.",

"Urgent: Your account has been compromised!",

"Your subscription will expire soon. Renew now!",

"Want to earn extra cash from home?",

"Meeting at the office next week.",

"Get rich quick with this one simple trick!",

"Can you send me the report by EOD?"

],

"label": [

"spam", "ham", "spam", "ham", "spam", "ham",

"spam", "ham", "spam", "spam", "spam", "ham",

"ham", "spam", "spam", "ham"

]

}

**Backend Code (Model Training and Prediction):**

* First, we will implement the backend code that handles data preprocessing, model training, and prediction. Save this as spam\_detection\_model.py.

# spam\_detection\_model.py

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from nltk.corpus import stopwords

import nltk

# Download stopwords if not already downloaded

nltk.download('stopwords')

class SpamDetector:

def \_\_init\_\_(self):

# Sample dataset for training

data = {

"message": [

"Free money now!!!",

"Hey, are we meeting tomorrow?",

"Earn cash fast!",

"Lunch at 12?",

"You won a prize, claim now!",

"Can we discuss the project details?",

"Get a loan with no credit check.",

"Please review the attached document."

],

"label": ["spam", "ham", "spam", "ham", "spam", "ham", "spam", "ham"]

}

# Create DataFrame and split data

df = pd.DataFrame(data)

X = df['message']

y = df['label']

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

# Initialize TF-IDF Vectorizer and transform text data

self.vectorizer = TfidfVectorizer(stop\_words=stopwords.words('english'))

X\_train\_tfidf = self.vectorizer.fit\_transform(X\_train)

# Train Naive Bayes model

self.model = MultinomialNB()

self.model.fit(X\_train\_tfidf, y\_train)

def predict(self, message):

# Transform the input message

message\_tfidf = self.vectorizer.transform([message])

# Predict using the model

prediction = self.model.predict(message\_tfidf)[0]

return prediction

**Explanation of Backend Code**

1. **SpamDetector Class:** Initializes a sample spam detection model using a TF-IDF vectorizer and a Naive Bayes classifier.
2. **predict Method:** Transforms user input and returns the model's prediction of "spam" or "ham".

**Frontend Code (User Interface using tkinter)**

Now, we will create the frontend code that imports the SpamDetector class from spam\_detection\_model.py and provides a simple UI for user interaction. Save this as spam\_detection\_app.py.

# spam\_detection\_app.py

import tkinter as tk

from tkinter import messagebox

from spam\_detection\_model import SpamDetector # Import the backend class

# Initialize the spam detector model

detector = SpamDetector()

# Function to handle button click

def check\_spam():

message = entry.get() # Get the message from the entry box

result = detector.detect(message) # Detect spam

messagebox.showinfo("Result", result) # Show the result in a message box

# Set up the GUI

root = tk.Tk()

root.title("Spam Detector")

entry = tk.Entry(root, width=50)

entry.pack(pady=20)

button = tk.Button(root, text="Check Spam", command=check\_spam)

button.pack(pady=10)

root.mainloop()

**Explanation of Frontend Code:**

1. **Import Backend Model:** The frontend code imports the SpamDetector class from spam\_detection\_model.py.
2. **check\_spam Function:** Gets input, calls the backend model's predict method, and displays the prediction result.
3. **GUI Setup:** The UI includes an entry box for user input, a button to trigger the check, and a display for results.

**6.Results**

Output Screenshots







