ABSTRACT

ABSTRACT

The number of people using mobile devices increasing day by day. SMS (short message service) is a text message service available in smartphones as well as basic phones. So, the traffic of SMS increased drastically. The spam messages also increased. The spammers try to send spam messages for their financial or business benefits like market growth, lottery ticket information, credit card information, etc. So, spam classification has special attention. In our project we use different machine learning algorithm such as SVM, KNN, Random Forest and Decision Tree etc. Finally the results will be compared and the best algorithm suited for spam filtering is determine. We also use dataset from UCI machine learning repository to build our project.

INTRODUCTION

CHAPTER 1

INTRODUCTION

Spam is unsolicited and unwanted messages sent electronically and whose content may be malicious. Email spam is sent/received over the Internet while SMS spam is typically transmitted over a mobile network. We'll refer to user that sent spam as 'spammers'. SMS messages are usually very cheap (if not free) for the user to send, making it appealing for unrightful exploitation. This is further aggravated by the fact that SMS is usually regarded by the user as a safer, more trustworthy form of communication than other sources, e. g., emails sms .The dangers of spam messages for the users are many: undesired advertisement, exposure of private 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 result in an increased cost in operations.

One of the main challenges with combating spam is that spammers are constantly developing new and more sophisticated tactics to avoid detection and reach their intended targets. This can include using techniques like spoofing and phishing to make their messages appear more legitimate, as well as exploiting vulnerabilities in software and systems to gain access to sensitive information.

SYSTEM STUDY

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM:

Most existing approaches to combating SMS spam were exported from successful email anti-spam solutions. The basic idea of spam filtering Required System work High configuration. The existing is working with logistics classification algorithm it which is giving the 56 percent accuracy. To the current doesn't integrated with webservices

DISADVANTAGE

- High system configuration.
- Low accuracy.
- Imported solutions.
- Limited integration.

2.2 PORPOSED SYSTEM

We are going to build a classification algorithm with more accuracy rate it will work with low configuration as well. We are proposing webpage as a user interface to show the model output

ADVANTAGES

- Improved accuracy.
- Low system configuration.
- User-friendly interface.
- Flexibility and integration.

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Processer : intel duel-core (or) higher
- RAM : 4GB (or) higher
- STORAGE SPACE : 100GB

3.2 SOFTWARE REQUIREMENTS

- Language Used : Python, Flask, Html, CSS
- Platform : Anaconda – Jupyter, Visual Studio

PROJECT DECRIPTION

CHAPTER 4

4.1 PROJECT DECRIPTION

The goal of this project is to develop a web application that uses machine learning to detect SMS spam messages. The application will be built using Flask, a Python web framework, and will be hosted on a web server. The application will allow users to enter SMS messages and will classify them as spam or not spam using a machine learning model. The model will be trained on a dataset of SMS messages that have been labeled as spam or not spam.

4.2 MODULE DESCRIPTION

The "SMS SPAM DETECTION USING MACHINE LEARNING" system will be developed using Python as the backend programming language. The application will have several modules, each with a specific function are.

- Training phase.
- Detection phase.
- Testing phase.
- Data Module.
- Model Module.
- Web Module.

4.2.1 TRAINING PHASE

- In training phase the SMS are analyzed and features are extracted, from the features extracted model is trained with different dataset.
- For training we will use the classification machine learning algorithm such as Random forest.

4.2.2 DETECTION PHASE

- In detection phase by using unknown data samples or sets.
- we will analysis the model features and generate the feature vectors.

4.2.3 TESTING PHASE

- In testing phase the model is test and we can find whether the SMS is SPAM or HAM

4.2.4 DATA MODULE

This module will handle the collection, preprocessing, and feature extraction of the SMS messages dataset. It will include functions to load the dataset, clean and preprocess the messages, and extract relevant features such as the frequency of certain words or phrases, the length of the message, or other characteristics.

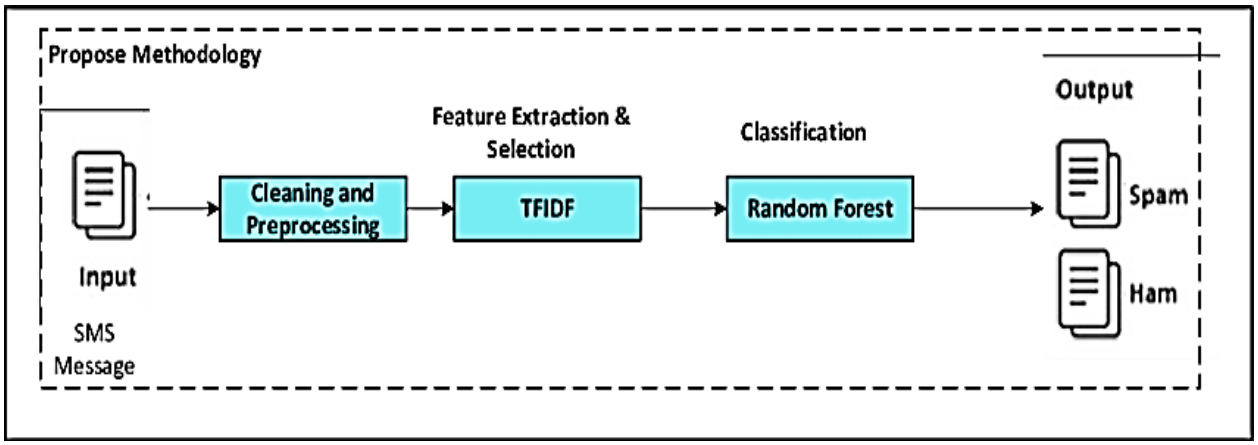
4.2.5 MODEL MODULE

This module will handle the selection, training, and evaluation of the machine learning model. It will include functions to evaluate different models and select the best performing one. The selected model will then be trained on the preprocessed dataset and evaluated on a testing set to ensure accuracy.

4.2.6 WEB MODULE

This module will handle the development of the Flask web application. It will include functions to create the web application routes, render templates, and interact with the user. It will also call functions from the Data and Model modules to preprocess the user input and classify it as spam or not spam using the trained machine learning model.

4.3 PROCESS OF SPAM



ABOUT LANGUAGE

CHAPTER 5

ABOUT LANGUAGE

5.1 INTRODUCTION TO PYTHON

Python is a popular programming language that is widely used for data analysis, machine learning, and artificial intelligence applications. It is known for its simplicity, readability, and ease of use, making it a popular choice among programmers of all levels of expertise.

In the context of the "“SMS SPAM DETECTION USING MACHINE LEARNING” project, Python can be used for a variety of tasks, such as data preprocessing, feature extraction, model training, and model evaluation. Python provides a range of libraries and frameworks for machine learning, such as scikit-learn, TensorFlow, and PyTorch, that can be used to build and train machine learning models. Python is also well-suited for data analysis tasks, such as exploring and visualizing data. Libraries such as Pandas and NumPy provide powerful tools for data manipulation and analysis, while Matplotlib and Seaborn provide a range of visualization tools for creating charts, graphs, and plots.

Python is highly extensible, and it has a large and active community that provides a range of third-party libraries and packages. This allows developers to easily access a wide range of tools and resources, making it easier to build complex applications quickly and efficiently.

5.2 PYTHON PACKAGES USED

5.2.1 NUMPY:

NumPy is a fundamental package for scientific computing with Python. It provides powerful tools for array manipulation, linear algebra, and mathematical operations. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, NumPy can be used for array manipulation and mathematical operations on the data.

5.2.2 PANDAS:

Pandas is a popular data manipulation library that provides tools for working with tabular data, including reading and writing data from various sources, filtering, sorting, and aggregating data, and handling missing values. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Pandas can be used for data preprocessing and cleaning.

5.2.3 MATPLOTLIB:

Matplotlib is a data visualization library that provides tools for creating various types of charts, graphs, and plots. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Matplotlib can be used for visualizing the data and analyzing the results.

5.2.4 SEABORN:

Seaborn is a data visualization library that builds on top of Matplotlib and provides additional tools for creating more complex and aesthetically pleasing visualizations. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Seaborn can be used for creating more advanced visualizations and analyzing the results.

5.2.5 SCIKIT-LEARN:

Scikit-learn is a machine learning library that provides tools for data preprocessing, model selection, model training, and model evaluation. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Scikit-learn can be used for building and training machine learning models.

5.2.6 COUNTVECTORIZER:

CountVectorizer is a feature extraction method used for text data that converts text into a matrix of token counts. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, CountVectorizer can be used for extracting features from the text messages.

5.2.7 ACCURACY_SCORE:

Accuracy_score is a function provided by Scikit-learn that computes the accuracy of a classification model by comparing the predicted labels with the true labels. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Accuracy_score can be used for evaluating the performance of the machine learning model.

5.2.8 CLASSIFICATION_REPORT:

Classification_report is a function provided by Scikit-learn that generates a report on the classification performance of a model, including precision, recall, and F1-score. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Classification_report can be used for analyzing the performance of the machine learning model.

5.2.9 CONFUSION_MATRIX:

Confusion_matrix is a function provided by Scikit-learn that computes the confusion matrix of a classification model, which shows the true positives, true negatives, false positives, and false negatives. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Confusion_matrix can be used for evaluating the performance of the machine learning model.

5.2.10 MULTINOMIALNB:

MultinomialNB is a classification algorithm used for text data that is based on the Naive Bayes theorem. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, MultinomialNB can be used for building and training the machine learning model.

5.2.11 PICKLE:

pickle is a Python library used for serializing and deserializing Python objects. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, pickle can be used for saving and loading the trained machine learning model.

5.2.12 FLASK:

Flask is a lightweight web application framework for Python that provides tools for building web applications and APIs. In the ""“SMS SPAM DETECTION USING MACHINE LEARNING”" project, Flask can be used for building and deploying the web application that predicts whether a given text message is spam or not.

5.3 INTRODUCTION TO HTML

HTML stands for HyperText Markup Language, and it is the standard markup language used to create web pages. HTML is used to structure content on the web and to add semantic meaning to text and images.

For the project report “SMS SPAM DETECTION USING MACHINE LEARNING,”HTML can be used to create a web-based user interface for the “SMS SPAM DETECTION USING MACHINE LEARNING” system. The user interface can be used to display the output of the machine learning algorithm and to allow users to interact with the system.

HTML provides a range of tags and attributes that can be used to structure content on the web. For example, the <head> tag can be used to specify the title of the web page, while the <body> tag can be used to define the main content of the page. Other tags can be used to add images, links, and other types of content to the web page.

5.4 INTRODUCTION TO CSS

CSS stands for Cascading Style Sheets, and it is a language used for describing the presentation of web pages. In the context of the project “SMS SPAM DETECTION USING MACHINE LEARNING,”CSS can be used to style the web interface that is being proposed for displaying the output of the classification algorithm.

The proposed web interface can be designed using HTML, and CSS can be used to define the layout, colors, fonts, and other visual elements of the interface. CSS allows for a separation of content and presentation, which makes it easier to modify the visual style of the interface without having to modify the underlying HTML code.

SOFTWARE DECRIPTION

CHAPTER 6

SOFTWARE DECRIPTION

6.1 INTRODUCTION TO ANCONODA

Anaconda is a software distribution and management platform that is used for data science and scientific computing. It provides a collection of over 7,500 open-source packages, including Python and R programming languages, along with libraries, tools, and frameworks for data analysis, machine learning, and artificial intelligence. Anaconda allows users to create and manage virtual environments for different projects, which helps to isolate dependencies and maintain consistency across different projects. It also includes a package and environment management system called "conda" that enables users to easily install, update, and uninstall packages and dependencies.

Anaconda comes with a user-friendly graphical user interface (GUI) called "Anaconda Navigator" that allows users to browse and launch various tools, such as Jupyter Notebook, Spyder, and RStudio. It also includes an easy-to-use package manager that allows users to easily install, manage, and update packages and dependencies required for their projects. It also includes a command-line interface (CLI) that allows advanced users to perform tasks using the terminal. One of the key benefits of Anaconda is its cross-platform compatibility, which allows it to be used on Windows, macOS, and Linux operating systems. Additionally, Anaconda is a popular choice in the data science and scientific computing communities due to its extensive package collection, ease of use, and robust documentation and community support.

6.2 THE JUPYTER NOTEBOOK

Jupyter Notebook is a web-based interactive computing environment that allows users to create and share documents that combine live code, equations, visualizations, and narrative text. It is an open-source tool that supports a variety of programming languages, including Python, R, Julia, and more. The Jupyter Notebook interface consists of a web application that runs on a local or remote server and can be accessed through a web browser. Users can create, edit, and run code cells, which are the basic units of computation in Jupyter Notebook. These cells can contain code, text, or equations, and users can switch between various input modes such as code, markdown, or raw text.

Jupyter Notebook provides a flexible and collaborative environment for data exploration, analysis, and visualization. Users can easily import and manipulate data using Python libraries such as Pandas, Numpy, and Matplotlib. They can also create interactive visualizations using tools such as Bokeh, Plotly, and Seaborn. Jupyter Notebook is widely used in various fields such as data science, machine learning, research, and education. It allows users to document their code and analysis in a reproducible and shareable way, making it easy to collaborate with others and reproduce their results. Jupyter Notebook is an essential tool for data scientists and researchers who need a flexible and powerful environment for exploring, analyzing, and communicating their findings.

The important features are

- Interactive computing.
- Multi-language support.
- Collaboration.
- Extensions and plugins.

6.3 MANAGE CODE

Jupyter Notebook allows users to manage their code in a flexible and interactive way. The notebook interface allows users to organize their code into cells, which can be run and edited independently. This makes it easy to test and refine individual sections of code without affecting the rest of the notebook.

6.4 MANAGE DATA

Jupyter Notebook is a powerful tool for data management and analysis. The notebook interface allows users to import and manipulate data using a wide range of Python libraries, including Pandas, Numpy, and Matplotlib. Users can import data from various sources, such as CSV files, Excel spreadsheets, SQL databases, and more. They can then use Python libraries to clean, transform, and analyze the data. Jupyter Notebook also allows users to create interactive visualizations of the data using tools such as Bokeh, Plotly, and Seaborn.

Jupyter Notebook provides a flexible and interactive environment for data exploration and analysis. Users can create and modify data analysis workflows in real-time, making it easy to experiment with different techniques and methods. They can also document their analysis and findings using Markdown cells, making it easy to share and collaborate with others.

6.5 COMMAN LANGUAGE SPECIFICATION

Jupyter Notebook is designed to support multiple programming languages and is built around a common language specification called the Jupyter Notebook Format. This format allows notebooks to be stored and shared in a standard way, making it easy to collaborate with others and work across different computing environments. The Jupyter Notebook Format is based on JSON (JavaScript Object Notation) and includes metadata and cell types that define the content of the notebook. The notebook consists of a sequence of cells, each of which can contain either code or Markdown text.

The code cells can be executed individually, allowing users to test and refine their code incrementally. The output of each code cell is displayed directly below the code, making it easy to see the results of the code in real-time. The Markdown cells can be used to provide explanations, documentation, or visualizations of the data. The Jupyter Notebook Format is supported by a wide range of tools and platforms, including GitHub, nbviewer, and JupyterLab. This makes it easy to share and collaborate on notebooks with colleagues, students, or the wider community.

6.6 LANGUAGE SUPPORTED BY JUPYTER NOTEBOOK

Jupyter Notebook supports multiple programming languages, including Python, R, Julia, MATLAB, Scala, Perl, Ruby, C++, Fortran, Bash. Python is the most commonly used language in Jupyter Notebook, and the majority of the available libraries and extensions are designed for Python.

Users can select their preferred programming language when they create a new notebook. Each notebook is associated with a specific kernel, which is responsible for executing the code in the notebook. Each kernel is specific to a particular programming language and can be easily switched or updated as needed.

6.7 ARCHITECHURE OF JUPYTER NOTEBOOK

Jupyter Notebook has a client-server architecture, which allows users to interact with the notebook through a web browser.

The architecture consists of three main components:

6.7.1 NOTEBOOK SERVER:

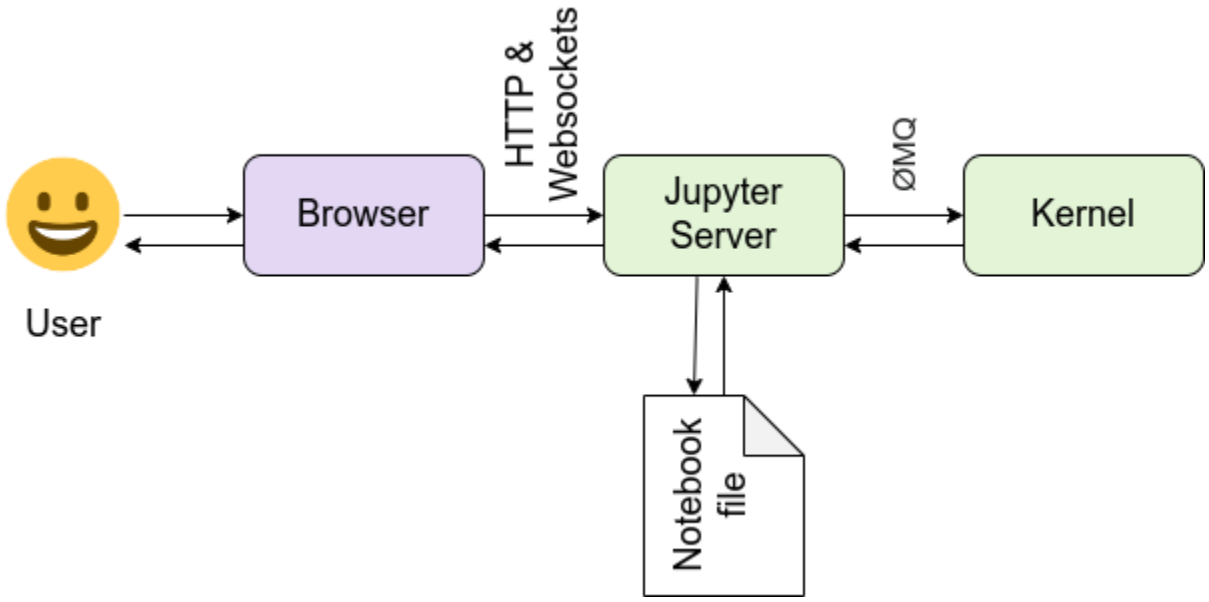
The Notebook Server is the core of the Jupyter Notebook architecture. It is responsible for managing notebooks, kernels, and other resources. The Notebook Server exposes a web interface that allows users to interact with their notebooks through a web browser.

6.7.2 KERNELS:

Kernels are separate processes that run the code in a notebook's code cells. Each notebook can have one or more kernels associated with it, depending on the programming languages used in the notebook. Kernels provide a secure and isolated environment for running code and communicating with the Notebook Server.

6.7.3 WEB BROWSER:

The web browser is the interface through which users interact with the notebook. Users can create, edit, and run code cells, as well as view the output of the cells. The web interface is also used to manage notebooks, kernels, and other resources.



6.8 KERNAL IN JUPYTER NOTEBOOK

In Jupyter Notebook, a kernel is a separate process that runs the code in a notebook's code cells. Each notebook can have one or more kernels associated with it, depending on the programming languages used in the notebook.

When a user executes a code cell in a notebook, the notebook sends the code to the associated kernel for execution. The kernel then runs the code and sends the results back to the notebook for display. Kernels provide a secure and isolated environment for running code, which helps to prevent conflicts between different notebooks and ensures that the code runs as intended.

Jupyter Notebook supports a wide range of programming languages, and each language has its own kernel. The kernel provides an interface between the notebook and the programming language, allowing users to interact with the language in a way that is consistent with the Jupyter Notebook interface.

Kernels are managed by the Jupyter Notebook Server, which creates and manages the kernels as needed. Kernels can be started and stopped independently of the notebook, allowing users to switch between different kernels and programming languages as needed.

6.9 JUPYTER NOTEBOOK SERVER

The Jupyter Notebook Server is the core component of the Jupyter Notebook architecture. It is responsible for managing notebooks, kernels, and other resources. The Notebook Server exposes a web interface that allows users to interact with their notebooks through a web browser.

When a user starts a Jupyter Notebook Server, they specify a directory where their notebooks will be stored. The Notebook Server monitors this directory for changes and automatically updates the list of available notebooks in the web interface. Users can create, edit, and delete notebooks from the web interface, as well as manage the associated kernels.

The Notebook Server also provides a range of configuration options, allowing users to customize the behavior of the server to suit their needs. For example, users can configure the Notebook Server to run on a specific port or to use SSL encryption for added security.

6.10 DATA ANALYSIS AND MANIPULATION

Data analysis and manipulation are key tasks in many fields, including business, finance, science, and engineering. CSV files are a popular format for storing data, and they can be easily manipulated and analyzed using a variety of tools and techniques.

One of the most common tools used for data analysis and manipulation is Python. Python provides a range of libraries and modules that can be used to read, write, and manipulate CSV files. For example, the Pandas library provides a range of functions for reading and writing CSV files, as well as tools for filtering, sorting, and aggregating data within the files.

Other popular tools for data analysis and manipulation include Microsoft Excel, R, and SQL. These tools provide a range of functions for working with CSV files, such as importing data, filtering, sorting, and visualizing data.

Data analysis and manipulation typically involve a range of tasks, such as cleaning and preprocessing data, transforming data into different formats, and performing statistical analysis on the data. CSV files provide a flexible and accessible format for storing and exchanging data, making them an essential tool for data analysis and manipulation tasks.

SYSTEM TESTING

CHAPTER 7

7.1 SYSTEM TESTING:

System testing is an important phase in the development of the "SMS Spam Detection Using Machine Learning" project. It involves testing the entire system, including the classification algorithm and the user interface, to ensure that it meets the requirements and specifications outlined in the project plan.

7.2 TYPES OF TESTS:

- Unit Testing.
- Integration Testing.
- Functional Testing.
- System Testing.
- Acceptance Testing.
- Regression Testing.
- Performance Testing.
- Security Testing.

7.2.1 UNIT TESTING:

Unit testing is a type of testing that focuses on testing individual components or modules of the software application. The purpose of unit testing is to ensure that each component or module is functioning as expected and that any defects or issues are identified and addressed as early as possible in the development process.

7.2.2 INTEGRATION TESTING:

Integration testing is a type of testing that is conducted to evaluate the behavior of the software application's components when integrated with other components. The objective of integration testing is to ensure that the software application's individual components are integrated correctly and work together seamlessly to meet the software application's requirements.

7.2.3 FUNCTIONAL TESTING:

Functional testing is a type of testing that focuses on testing the software application's functions and features to ensure that they are working as expected, according to the defined requirements and specifications. Functional testing is typically conducted using test cases that are designed to test each function or feature of the software application.

7.2.4 SYSTEM TESTING:

System testing is a type of testing that focuses on testing the entire software system, including all components and modules, to ensure that the system works as expected and meets all the requirements. System testing is typically conducted after all the individual components and modules have been tested and integrated into the system.

7.2.5 ACCEPTANCE TESTING:

Acceptance testing is a type of testing that is conducted to determine whether the software application meets the acceptance criteria and requirements of the end-users or stakeholders. Acceptance testing is typically conducted after the software application has undergone functional and system testing.

7.2.6 REGRESSION TESTING:

Regression testing is a type of testing that is conducted to ensure that changes or modifications made to the software application do not affect the existing functionality of the software. Regression testing is typically conducted after modifications have been made to the software application, such as bug fixes or new feature implementations.

7.2.7 PERFORMANCE TESTING:

Performance testing is a type of testing that is conducted to evaluate the software application's performance under different load and stress conditions. The objective of performance testing is to identify the system's maximum capacity, response time, scalability, and stability under different workloads and stress conditions.

7.2.8 SECURITY TESTING:

Security testing is a type of testing that is conducted to identify vulnerabilities, threats, and risks in the software application's security architecture. The objective of security testing is to ensure that the software application is protected against potential security threats, such as unauthorized access, data breaches, and cyber-attacks.

SYSTEM IMPLEMENTATION

CHAPTER 8

SYSTEM IMPLEMENTATION

8.1 SYSTEM IMPLEMENTATION

System implementation refers to the process of putting a system into operation, including installation, configuration, testing, and deployment. In the context of the project "SMS SPAM DETECTION USING MACHINE LEARNING," system implementation involves implementing the classification algorithm and the proposed web interface for displaying the output of the algorithm. The implementation process typically begins with installing and configuring the necessary software tools and libraries, such as Python, Pandas, Flask, and other dependencies. The classification algorithm can then be developed using machine learning techniques, such as supervised learning or unsupervised learning, depending on the specific requirements of the project. Once the classification algorithm has been developed and tested, it can be integrated into the proposed web interface. This involves designing the user interface using HTML and CSS, and using a web development framework such as Flask to create the backend of the web application.

The web interface can then be tested and deployed on a server or hosting platform, such as Amazon Web Services or Heroku, to make it accessible to users. The deployment process may involve configuring the server environment, setting up the database, and ensuring that the web application is secure and scalable. Throughout the implementation process, it is important to conduct testing and quality assurance to ensure that the system is functioning correctly and meeting the requirements of the project. This may involve testing the classification algorithm with sample data sets, conducting user testing of the web interface, and performing stress testing and load testing to ensure that the system can handle high volumes of traffic.

SYSTEM MAINTENANCE

CHAPTER 9

SYSTEM MAINTENANCE

9.1 SYSTEM MAINTENANCE

System maintenance is a critical process that ensures the smooth operation and longevity of a system, including hardware, software, and network infrastructure. It involves a range of tasks, including monitoring, troubleshooting, updating, and optimizing the system to ensure that it meets the desired performance and security standards. In the context of the "“SMS SPAM DETECTION USING MACHINE LEARNING” USING MACHINE LEARNING" project, system maintenance is essential to ensure the reliable operation of the classification algorithm and the web interface. This includes tasks such as regularly monitoring the system to detect any issues, such as hardware failures, network problems, or software bugs, and taking steps to resolve them promptly.

Another critical aspect of system maintenance is updating the system software and components to ensure that they are up-to-date and compatible with the latest security and performance standards. This includes updating the operating system, software libraries, and any other components used in the system. System optimization is also an important part of system maintenance, as it helps to improve the performance and efficiency of the system. This can involve tasks such as tuning the system configuration, optimizing database performance, and implementing caching mechanisms to reduce the load on the system. Regular backups of the system data and configuration are also an essential aspect of system maintenance, as they help to ensure that the system can be easily restored in case of a catastrophic failure or data loss.

CONCLUSION & FUTURE ENHANCEMENT

CHAPTER10

CONCLUSION & FUTURE ENHANCEMEN

10.1 CONCLUSION

In conclusion, the SMS spam classification project aims to classify a input SMS messages as either spam or not spam. The model is trained using a dataset of labeled SMS messages and uses machine learning techniques to learn the patterns in the data. The final model should be able to accurately classify new input SMS messages as spam or not spam. The performance of the model can be evaluated using metrics such as accuracy, precision, recall, and F1-score. The project could be improved by using more data and fine-tuning the model's parameters. Additionally, techniques such as ensemble methods and machine learning can be used to improve the performance of the model.

10.2 FUTURE ENHANCEMENTS

- Integration with additional data sources.
- Integration with other machine learning techniques.
- Integration with additional user interfaces.
- Integration with natural language processing.

APPENDIX

CHAPTER 11

APPENDIX

11.1 SOURCE CODING

BACKEND CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.naive_bayes import MultinomialNB
import pickle
from flask import Flask
# Load the dataset
df = pd.read_csv('spamraw.csv')
# Extract SPAM messages
spam_messages = df[df["type"]=="spam"]
# Find count and unique messages count of SPAM messages.
spam_messages.describe()
# Plot the counts of HAM (non SPAM) vs SPAM
sns.countplot(data=df, x=df["type"]).set_title("Amount of spam and no-spam
messages")
plt.show()
# Create a new dataframe with balanced classes
df2 = df[df['type'] == 'ham'][0:746]
df3 = df[df['type'] == 'spam']
df4 = pd.concat([df2, df3])
# Plot the counts of HAM (non SPAM) vs SPAM
sns.countplot(data=df4, x=df4["type"]).set_title("Amount of spam and no-spam
messages")
plt.show()
# Save the new dataframe to a CSV file
df4.to_csv("new_spamraw.csv", index=False)
# Split the dataset into train and test sets
```

```
data_train, data_test, labels_train, labels_test = train_test_split(df4.text,
df4.type, test_size=0.2, random_state=0)
# Fit and transform the training data
vectorizer = CountVectorizer()
data_train_count = vectorizer.fit_transform(data_train)
# Pickle the vectorizer for later use
pickle.dump(vectorizer, open('vectorizer.pkl', 'wb'))
# Fit the Naive Bayes classifier to the training data
clf = MultinomialNB()
clf.fit(data_train_count, labels_train)
# Pickle the classifier for later use
pickle.dump(clf, open('model.pkl', 'wb'))
# Transform the test data using the vectorizer
data_test_count = vectorizer.transform(data_test)
# Make predictions on the test data
predictions = clf.predict(data_test_count)
# Evaluate the accuracy of the model
print("accuracy_score : ", accuracy_score(labels_test, predictions))
print("confusion_matrix : \n", confusion_matrix(labels_test, predictions))
print(classification_report(labels_test, predictions))
# Load the pickled model and vectorizer
pickled_model = pickle.load(open('model.pkl', 'rb'))
pickled_vec = pickle.load(open('vectorizer.pkl', 'rb'))
# Define a Flask app
app = Flask(__name__)
# Define the API endpoint
@app.route("/spam")
def spam_detection():
    # Get the input text
    input_text = ["okmail: Dear Dave this is your final notice to collect your "]
    # Transform the input text using the vectorizer
    prediction_vector_count = pickled_vec.transform(input_text)
    # Predict the label
    response = pickled_model.predict(prediction_vector_count)
    return response[0]
# Run the Flask app
if __name__ == '__main__':
```

```
    app
import pickle
from flask import Flask, render_template, request
# Load the pickled model and vectorizer
pickled_model = pickle.load(open('model.pkl', 'rb'))
pickled_vec = pickle.load(open('vectorizer.pkl', 'rb'))
# Create a new message to predict
print("enter your message to predict")
new_message = [input()]
# Transform the new message using the vectorizer
new_message_count = pickled_vec.transform(new_message)
# Make a prediction using the pickled model
prediction = pickled_model.predict(new_message_count)
# Print the prediction
print(prediction)"""
import pickle
from flask import Flask, render_template, request
# Load the pickled model and vectorizer
pickled_model = pickle.load(open('model.pkl', 'rb'))
pickled_vec = pickle.load(open('vectorizer.pkl', 'rb'))
# Define a Flask app
app = Flask(__name__)
# Define the home page
@app.route('/')
def home():
    return render_template('index.html')
# Define the prediction endpoint
@app.route('/predict', methods=['POST'])
def predict():
    # Get the input text from the form
    message = request.form['message']
    # Transform the input text using the vectorizer
    message_count = pickled_vec.transform([message])
    # Predict the label using the pickled model
        prediction = pickled_model.predict(message_count)
    # Return the prediction to the user
    return render_template('index.html', prediction=prediction[0])
```

```
# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```

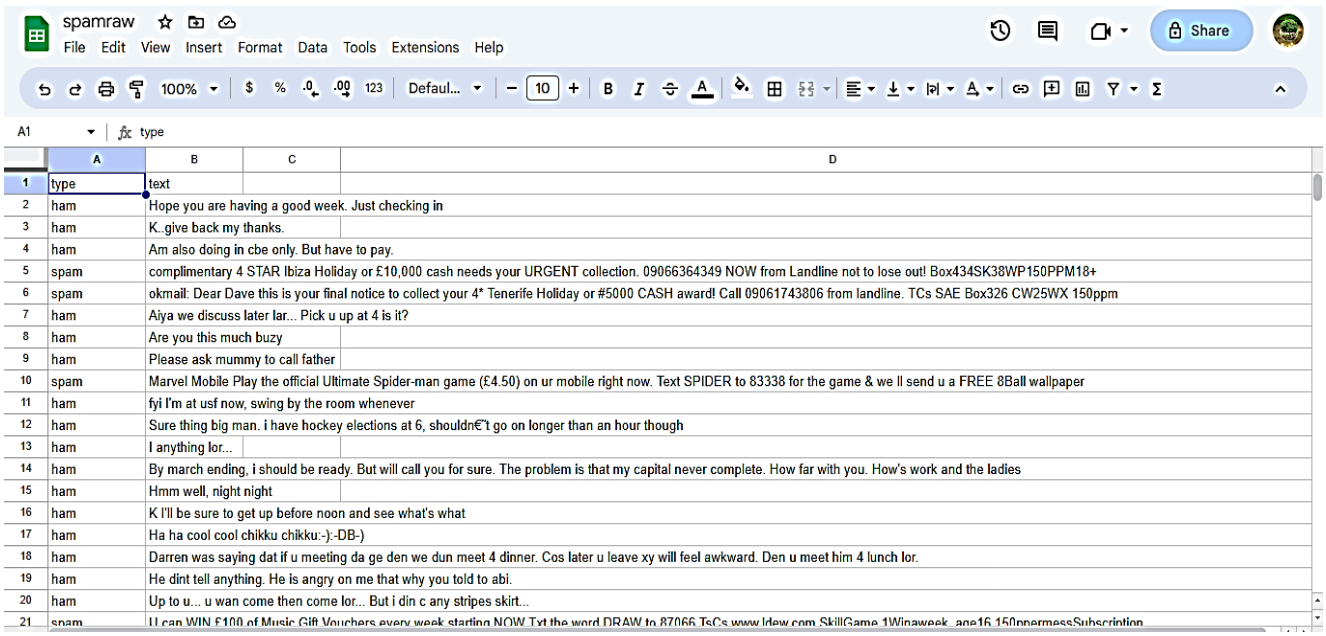
FRONTEND CODE

```
<html>
   <head>
 <title> spam detection</title>
   </head>
<style>
body{
background: rgb(36,0,0);
background: linear-gradient(90deg, rgba(36,0,0,1) 0%, rgba(0,192,223,1) 0%,
rgba(55,0,255,1) 100%);
}
h1{
color:yellow;
}
button {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 5px 12px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 11px;
  margin: 4px 2px;
    box-shadow: rgba(0, 0, 0, 0.19) 0px 10px 20px, rgba(0, 0, 0, 0.30) 0px 6px
6px;

  font-family: 'Poppins', sans-serif;
  cursor: pointer;
}
lable{
color:white;
}
```

```
h3{
color:white;
}
p{
color:#fd1d1d;


</style>
   <body align="center">
      <h1>Spam Prediction</h1>
<div class="box">
      <form method="POST" action="/predict">
        <h3>  <label for="message">Enter a message:</label><br></h3>
          <textarea id="message" name="message" rows="4"
cols="50"></textarea><br>
</div>
<br>
<br>
          <button><input type="submit" value="Predict"></button>
      </form>
      {% if prediction %}
      <p>Prediction: {{ prediction }}</p>
      {% endif %}
   </body>
</html>
```

11.2 OUTPUT SCREENSHOT

CSV FILE



spamraw

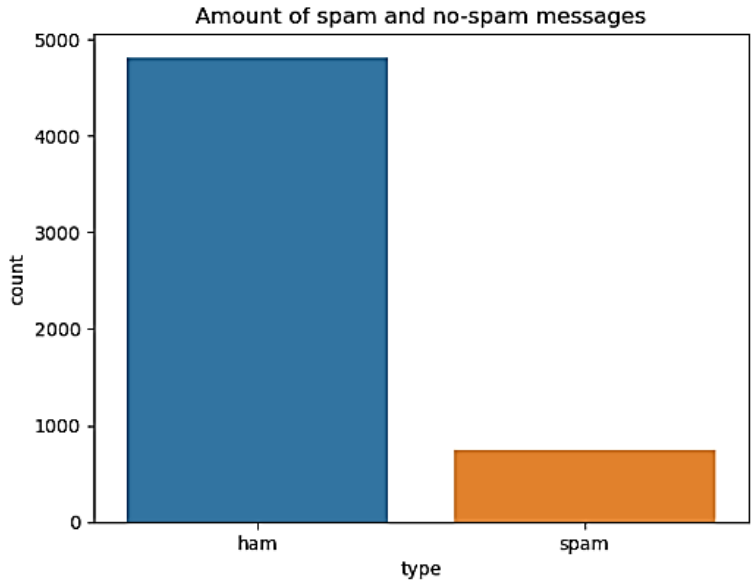
File Edit View Insert Format Data Tools Extensions Help

Share

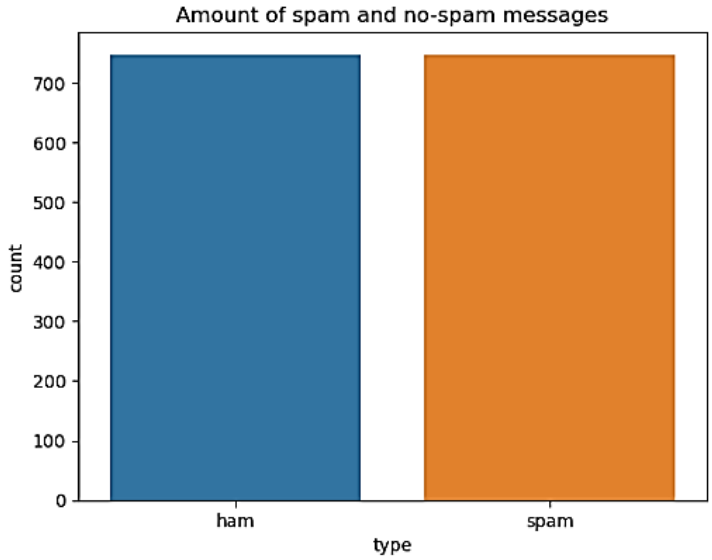
A1 | type

	A	B	C	D
1	type	text		
2	ham	Hope you are having a good week. Just checking in		
3	ham	K..give back my thanks.		
4	ham	Am also doing in cbe only. But have to pay.		
5	spam	complimentary 4 STAR Ibiza Holiday or £10,000 cash needs your URGENT collection. 09066364349 NOW from Landline not to lose out! Box434SK38WP150PPM18+		
6	spam	okmail: Dear Dave this is your final notice to collect your 4* Tenerife Holiday or #5000 CASH award! Call 09061743806 from landline. TCs SAE Box326 CW25WX 150ppm		
7	ham	Aiya we discuss later lar... Pick u up at 4 is it?		
8	ham	Are you this much buzy		
9	ham	Please ask mummy to call father		
10	spam	Marvel Mobile Play the official Ultimate Spider-man game (£4.50) on ur mobile right now. Text SPIDER to 83338 for the game & we ll send u a FREE 8Ball wallpaper		
11	ham	fyi I'm at usf now, swing by the room whenever		
12	ham	Sure thing big man. i have hockey elections at 6, shouldn€˜t go on longer than an hour though		
13	ham	I anything lor...		
14	ham	By march ending, i should be ready. But will call you for sure. The problem is that my capital never complete. How far with you. How's work and the ladies		
15	ham	Hmm well, night night		
16	ham	K I'll be sure to get up before noon and see what's what		
17	ham	Ha ha cool cool chikku chikku:-)-DB-)		
18	ham	Darren was saying dat if u meeting da ge den we dun meet 4 dinner. Cos later u leave xy will feel awkward. Den u meet him 4 lunch lor.		
19	ham	He dint tell anything. He is angry on me that why you told to abi.		
20	ham	Up to u... u wan come then come lor... But i din c any stripes skirt...		
21	spam	U can WIN £100 of Music Gift Vouchers every week starting NOW Txt the word DRAW to 87066 TsCs www.ldew.com SkillGame 1Winaweek, age16 150ppermessSubscription		

IMBALANCE DATASET

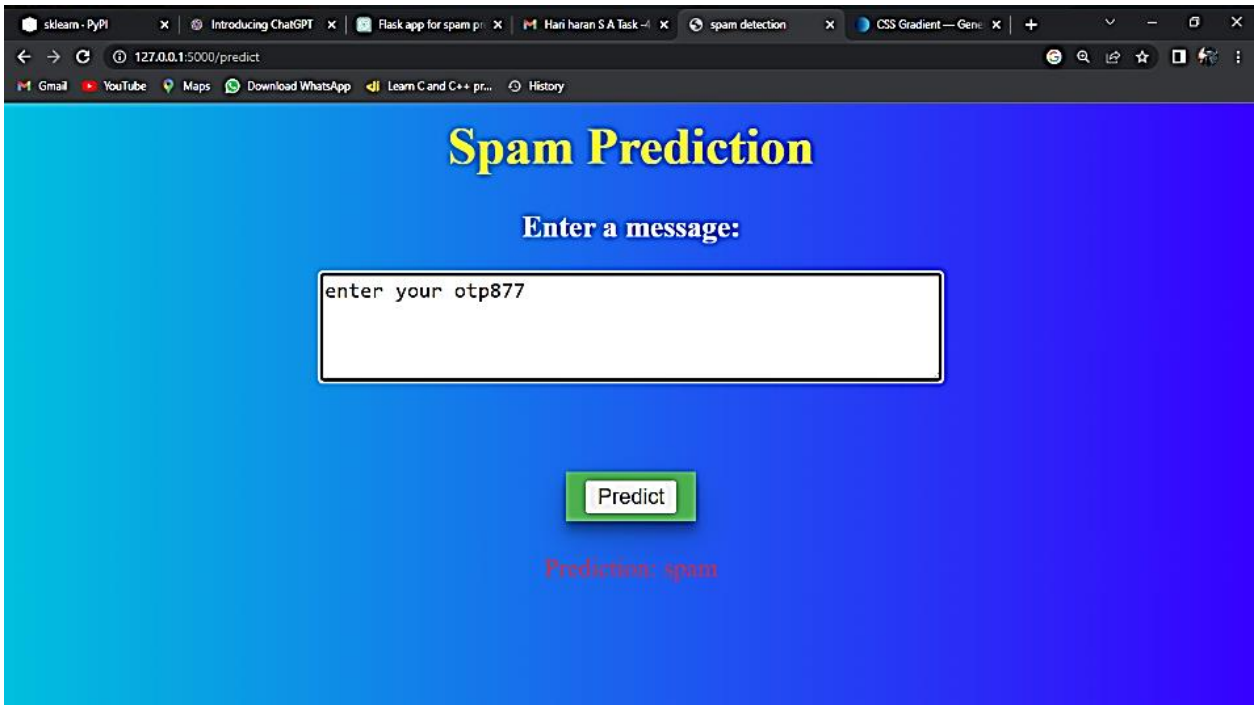


BALANCED DATASET

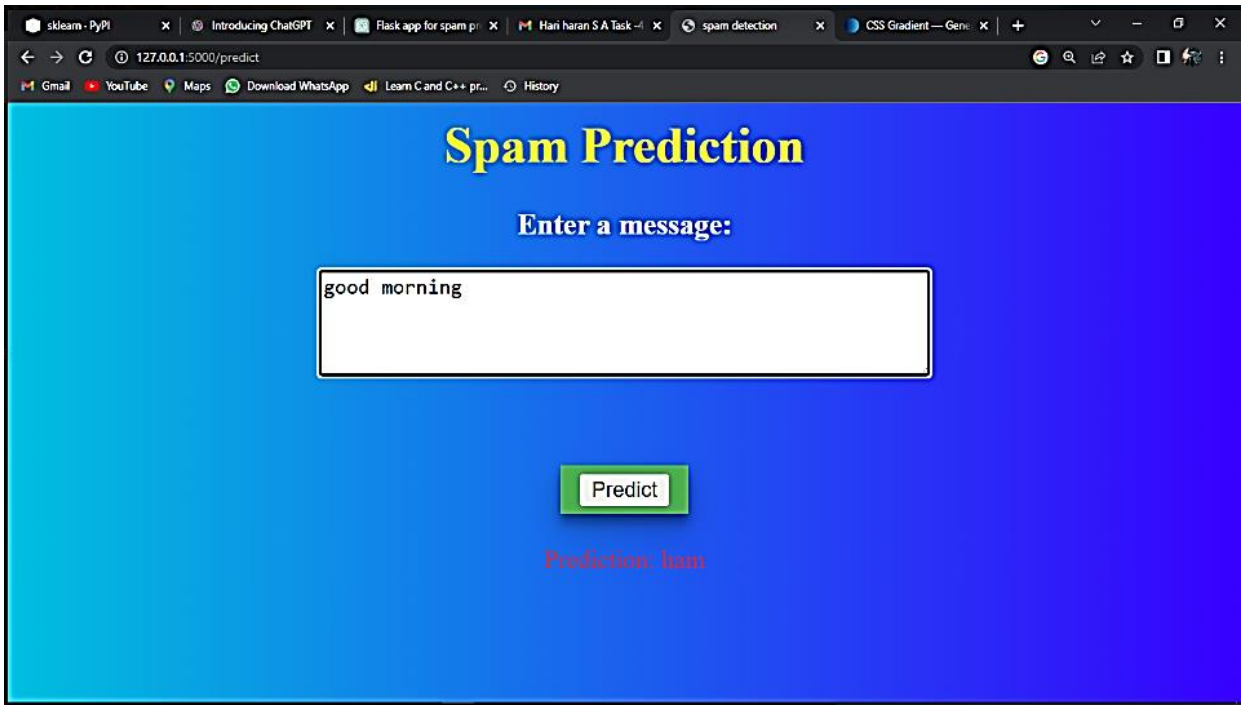


PREDICTING SPAM OR HAM MESSAGE

SPAM MESSAGE



HAM MESSAGE


sklearn · PyPI
Introducing ChatGPT
Flask app for spam pr
Hari haran S A Task -4
spam detection
CSS Gradient — Gene
127.0.0.1:5000/predict
Gmail
YouTube
Maps
Download WhatsApp
Learn C and C++ pr...
History
Spam Prediction
Enter a message:
good morning
Predict
Prediction: ham

REFERENCES

CHAPTER 12

REFERENCES

1. https://www.kaggle.com/
2. Gupta, M., Bakliwal, A., Agarwal, S., Mehndiratta, P.: A comparative study of spam SMS detection using machine learning classifiers. In: 2018 Eleventh International Conference on Contemporary Computing (IC3).
3. Popovac, M., Karanovic, M., Sladojevic, S., Arsenovic, M., Anderla, A.: Convolutional neural network based SMS spam detection. In: 26th Telecommunications Forum TELFOR 2018
4. http://en.wikipedia.org/wiki/Short Message Service
5. Scikit-learn Ensemble Documentation, ”http://scikit-learn.org/stable/ modules/ensemble.html”