

Decision Tree Model for Email Classification
Major Project Report Submitted to
SRI PADMAVATI MAHILA VISVAVIDYALAYAM

In Partial fulfilment of the requirement for the
MASTER OF COMPUTER APPLICATIONS
IV SEMESTER

By

K.MANJU BHARGAVI

(2021MCA16057)

Under the guidance of

Prof. P. Venkata Krishna



DEPARTMENT OF COMPUTER SCIENCE

SRI PADMAVATI MAHILA VISVAVIDYALAYAM (Women's University)

Accredited with A+ Grade by NAAC

Tirupati-517502(A.P), Andhra Pradesh, India

September-2023

DEPARTMENT OF COMPUTER SCIENCE
SRI PADMAVATI MAHILA VISVAVIDYALAYAM
(Women's University)
Tirupati, Andhra Pradesh-517502, India
Accredited with **A+** Grade by NAAC



CERTIFICATE

This is to certify that the project work entitled **Decision Tree Model for Email Classification**” is a bonafide record of work carried out by **K.MANJUBHARGAVI (2021MCA16057)** in the **Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupati** in partial fulfilment of the requirements of IV Semester of **MASTER OF COMPUTER APPLICATIONS**. The content of the Project Report has not been submitted to any other University / Institute for the award of any degree.

SIGNATURE OF THE GUIDE

SIGNATURE
Head of the Department

DECLARATION

I here by declare that MCA IV Semester major project entitled “ **Decision Tree Model for Email Classification** ” was done at the **Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam**, Tirupati, in the year 2023 under the guidance of **Prof. P. Venkata krishna** in partial fulfilment of requirements of MCA IV Semester.

I also declare that this project is my original contribution of the best of my knowledge and belief. I further declare that this work has not been submitted either in full or part for the award of any other degree of this or any other university.

Signature of the Student

ACKNOWLEDGEMENT

I am greatly indebted to our guide **Prof. P. Venkata Krishna** for taking keen interest on my project work and providing valuable suggestions in all the possible areas of improvement.

I express my sincere thanks to the teaching staff of the Department of Computer Science for extending support and encouragement to me in all the stages of the project work.

I gratefully acknowledge and express my gratitude to the non-teaching staff of the Computer Science Department who supported us in preparing the project report.

Signature of the Student

INDEX

SL.No.	Content	Page No.
	ABSTRACT	
1.	INTRODUCTION	1-4
	1.1.Organization profile (for major project only)	
	1.2. University profile	
2.	PROBLEM DEFINITION	5-7
	2.1. Aim	
	2.2. Problem Definition	
	2.2.1. Existing System	
	2.2.2. Proposed System	
	2.3. Objectives	
3.	SYSTEM ANALYSIS	8-18
	3.1. Software Requirement Specifications	
	3.2. System Requirement	
	3.2.1. Hardware Requirements	
	3.2.2. Software Requirements	
	3.3. Feasibility Study	
	3.3.1. Operational Feasibility	
	3.3.2. Technical Feasibility	
	3.3.3. Economical Feasibility	
	3.4. Modeling Approaches	
	(Based on the selected project)	
	3.4.1. UML diagrams	
	3.4.1.1. Use Case Diagrams	
	3.4.1.2. State Diagrams	
	3.4.1.3. Sequence Diagrams	
	3.4.2. Data flow Diagrams	

SL.No.	Content	Page No.
4.	SYSTEM TESTING	19-22
	4.1. Testing Schemes	
	4.1.1.	
	4.1.2.	
	:	
	4.2. Test cases	
5.	IMPLEMENTATION	23-24
6.	CONCLUSION	25
	6.1. Performance of Proposed System	
	6.2. Limitation	
	6.3. Future Enhancements	
	BIBILOGRAPHY	26-27
	APPENDICES	28-59
	APPENDIX A: Screens	
	• Screen Shots	
	APPENDIX C: Source code	

ABSTRACT

In addition to the undeniable benefits, the development of the Internet has led to many undesirable security effects. Spam emails are one of the most challenging issues faced by the Internet users. Spam refers to all emails of unsolicited content that arrive in a user's email box. Spam can often lead to network congestion and blocking or even damage to the system for receiving and sending electronic messages. Thus, appropriate classification of spam email from legitimate email has become quite important. This paper presents a new approach for feature selection and Iterative Dichotomiser 3 (ID3) algorithm designed to generate the decision tree for email classification. The experimental results indicate that the proposed model achieves very high accuracy.

.

1.INTRODUCTION

1.1 ORGANIZATION PROFILE:

Datapoint IT & Hardware Tech Pvt. Ltd.,

Business Proposition

Datapoint is incepted by young and ambitious team of Professional in the Industry with the Idea & motto of *“Simplifying Solutions & opportunities”*. Datapoint is into IT Training (Corporate/Individual), Project assistance, Software Development and Placements. Datapoint is one among the very few companies in Hyderabad, which are spread across all the areas and technologies. **Datapoint has been actively in the profession of sourcing IT professionals from the year 2001.** We have since placed scores of candidates from different skill sets, with varying levels of experience.

Datapoint started its journey initially as a Consulting Company and as a successful Placement Consultants as per the clients requirements we also emerged as a Corporate Training. Of-late we found that many engineering graduates are not being able to find jobs for themselves, despite increasing demand for IT professionals & Even our clients couldn't able to find the suitable and potential candidates even in the freshers.

At this crucial point we found the gap which needed to be filled by Datapoint to improve our client satisfaction levels. The very decision of *“Training (IT & Non-IT aspects) & providing Project assistance”* to the freshers made Datapoint as a significant player in the market. Datapoint is assisting many colleges and Organizations in Training & Recruiting freshers.

At Datapoint, unlike other training institutes we know the Industry requisites and what an Organization expects from a candidate and henceforth we train our students accordingly so that they can get in to the market with more confidence. Datapoint as we already mentioned not only trains extensively on technologies but also on soft skills. **Datapoint also motivates the students to implement the projects on their own, which gives them a real time exposure towards the same.**

Datapoint endeavors to be a pioneer in Recruiting and manpower consulting thanks to strategic alliances with leading multinational companies in India and US of America. **Our technically competent, experienced, and certified consultants will help our clientele to get the right manpower at the right time.** We take pride in having top-notch companies who make enable us to have faith in the future through maintaining high quality in screening, hiring and management.

Datapoint has identified a number of areas of thrust in the emerging and ever growing IT industry and virtue of which, we would focus all our energies to get on to the fast track in the shortest possible period. We pursue requirements from leading Corporate in India and abroad.

Mission

“Our mission is to identify, recruit and facilitate quality manpower who are technically strong, dynamic and determined, as we are, for the future belongs to those who think and prove global.”

Why to choose Datapoint for Academic Projects/Internship?

Our Project training is based on JAVA, DOTNET, EMBEDDED SYSTEMS, VLSI, MATLAB DSP/DIP, MATLAB SIMULINK as per industry expectations and we will allow the students to do project in real time environment under the guidance of industrial experts. We afford quality training to student which is evinced by the fact that several colleges recommended their students for our extensive project training

Why to choose Datapoint for Placements?

We guarantee **reliable and productive candidates**, which is evinced by the fact that over **94% of our clientele** have done repeat businesses with us. We provide candidates at competitive bill rates, guaranteed not to change during the life of your assignment.

Once your positions are filled, they are there to stay.

“Simplifying Opportunities”: It’s more than just our slogan. Our entire business is geared toward helping our clients to successfully complete and implement their critical I.T initiatives in a timely and cost efficient manner. Our job is to provide you with “Excellent manpower which makes the organization” so that you can be more successful and continue to be the leader in the industry.

Services:

Our Corporate Training Division has identified the following as thrust areas:

- IT Solutions
- Recruitment and Staffing
- Corporate Training

- Academic Projects (B.Tech/B.E (Mini & Major)/M.Tech/M.E/MCA), Polytechnic
- Internship
- Online Training
- CRT & Soft skills
- Workshops
- Established in 2001.
- 22 Years of Expertise in Recruitments & Staffing.
- Having Own Software Development Division.
- Had a Clientele of about 20 MNCs.
- The Company, which helps you to meet the Industry expectations.
- Excellent track record in placing the Candidates of various levels.
- A company with the Coding standards of CMM level companies.

1.2 UNIVERSITY PROFILE:

Sri Padmavati Mahila VisvaVidyalayam (university for women) was founded in the year 1983 by N.T.Rama Rao, the Chief Minister of Andhra Pradesh, with the fervent desire to train women students as better builders of nation and to include skills of leadership in all aspects of life. The University was established under the Sri Padmavati Mahila Visvavidyalayam Act of 1983, which has come in to force on 14th of April 1983, it was started with ten faculties and 300 students and 20 staff members. In pursuance of objectives of university is awarded “A Grade” by NAAC.

The campus of Sri Padmavati Mahila Visvavidyalayam is spread out in lush green area of 138.43 acres. The university is situated at a distance of 3 kilometres from railway and bus stations of Tirupati. The campus has the necessary buildings to run its academic programs and administrative machinery. There are separate Buildings for humanities and science, university’s Administration, Central Library, University Auditorium, Sericulture complex and school of Pharmaceutical Sciences and also an independent building for Computer Science, Computer Centre and Examination hall.

The Objectives of the University:

- Emancipation of women's through Education.
- Acquisition of sound updated academic knowledge.
- Acquisition of specific vocational skills and competence for gainful occupation.
- Developing awareness of major issues and problems being faced by the society and by the women in particular.
- Acquisition of evaluation skills and training to function as better builders of the home and society.
- Acquisition of skills of leadership and entrepreneurship in all aspects of life. Keeping in view of these objectives, the teaching and research works are carried out in the university.

2. PROBLEM DEFINITION:

2.1 Aim :

Decision Tree Model for Email Classification

2.2 Problem Definition:

In addition to the undeniable benefits, the development of the Internet has led to many undesirable security effects. Spam emails are one of the most challenging issues faced by the Internet users. Spam refers to all emails of unsolicited content that arrive in a user's email box. Spam can often lead to network congestion and blocking or even damage to the system for receiving and sending electronic messages. Thus, appropriate classification of spam email from legitimate email has become quite important. This paper presents a new approach for feature selection and Iterative Dichotomiser 3 (ID3) algorithm designed to generate the decision tree for email classification. The experimental results indicate that the proposed model achieves very high accuracy.

2.2.1 EXISTING SYSTEM:

Unfortunately, the continuous rise of email users has led to a massive increase of spam emails. Whether it is commercial in nature or not, spam emails can cause serious problems in electronic communication. Spam emails produce huge amount of unsolicited data and thus affect the network bandwidth and storage capacity. Due to the large number of spam emails to users of email services it is difficult to distinguish useful

Unfortunately, the continuous rise of email users has led to a massive increase of spam emails. Whether it is commercial in nature or not, spam emails can cause serious problems in electronic communication. Spam emails produce huge amount of unsolicited data and thus affect the network bandwidth and storage capacity. Due to the large number of spam emails to users of email services it is difficult to distinguish useful from unsolicited emails. Thus, managing and filtering emails is an important challenge. The filtering purpose is to detect and isolate spam emails.

from unsolicited emails. Thus, managing and filtering emails is an important challenge. The filtering purpose is to detect and isolate spam emails.

DISADVANTAGES OF EXISTING SYSTEM:

- Spam emails are usually sent in bulk and do not target individual recipient.
- The aim is to preserve the most important features and to reduce computations demand.

Algorithm: Linear Discriminant Analysis, Classification.

2.2.2 PROPOSED SYSTEM:

In the model development, the dataset is consistently split into train and test set of 80% and 20%. Train set has 400 profiles and test set has 100 profiles. The dataset used for modelling looks like this. Preprocessing is a crucial step in method. The aim is to clean the data and prepare it for use in a prediction algorithm. Few improvements are required for the data obtained from Occidental College in order to make it suitable for the proposed machine learning algorithms. Determining how to deal with missing data is a common problem in data cleaning. Since the function in question could be a good predictor of the algorithm's outcome, it's critical to find missing entries, locate them, and apply a treatment based on the variable form that enables us to use the data in the model. The data was pre-processed and split into two classes at random: a training set and a testing set. We selected 80 percent of the 7976 entries in our dataset as our training collection.

ADVANTAGES OF PROPOSED SYSTEM:

- The proposed approach is evaluated using accuracy, precision and recall.
- In the near future, it is planned to incorporate other classifiers and to compare their performances with the proposed approach.

Algorithm: Machine learning, ID3 algorithm, classification algorithm.

2.3 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

3.SYSTEM ANALYSIS

3.1 Software Requirement Specifications

REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

REQUIREMENT SPECIFICATION:

Functional Requirements

- Graphical User interface with the User.

Software Requirements

For developing the application the following are the Software Requirements:

1. Python
2. Django

Operating Systems supported

1. Windows 10 64 bit OS

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Intel i9
- RAM: 32 GB
- Space on Hard Disk: minimum 1 TB

3.2 System Requirements

3.2.1 Hardware Requirements

❖ System	: Intel i5 6 core.
❖ Hard Disk	: 500 GB SSD.
❖ Monitor	: 15'' LED
❖ Input Devices	: Keyboard, Mouse
❖ Ram	: 32 GB.

3.2.2 Requirements Software

❖ Operating system	: Windows 10.
❖ Coding Language	: Python.
❖ Tool	: PyCharm, Visual Studio Code
❖ Database	: SQLite

3.3 Feasibility Study:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

3.3.1 ECONOMICAL FEASIBILITY :

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to

high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.3.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.4 Modeling Approaches:

3.4.1 UML Diagrams:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

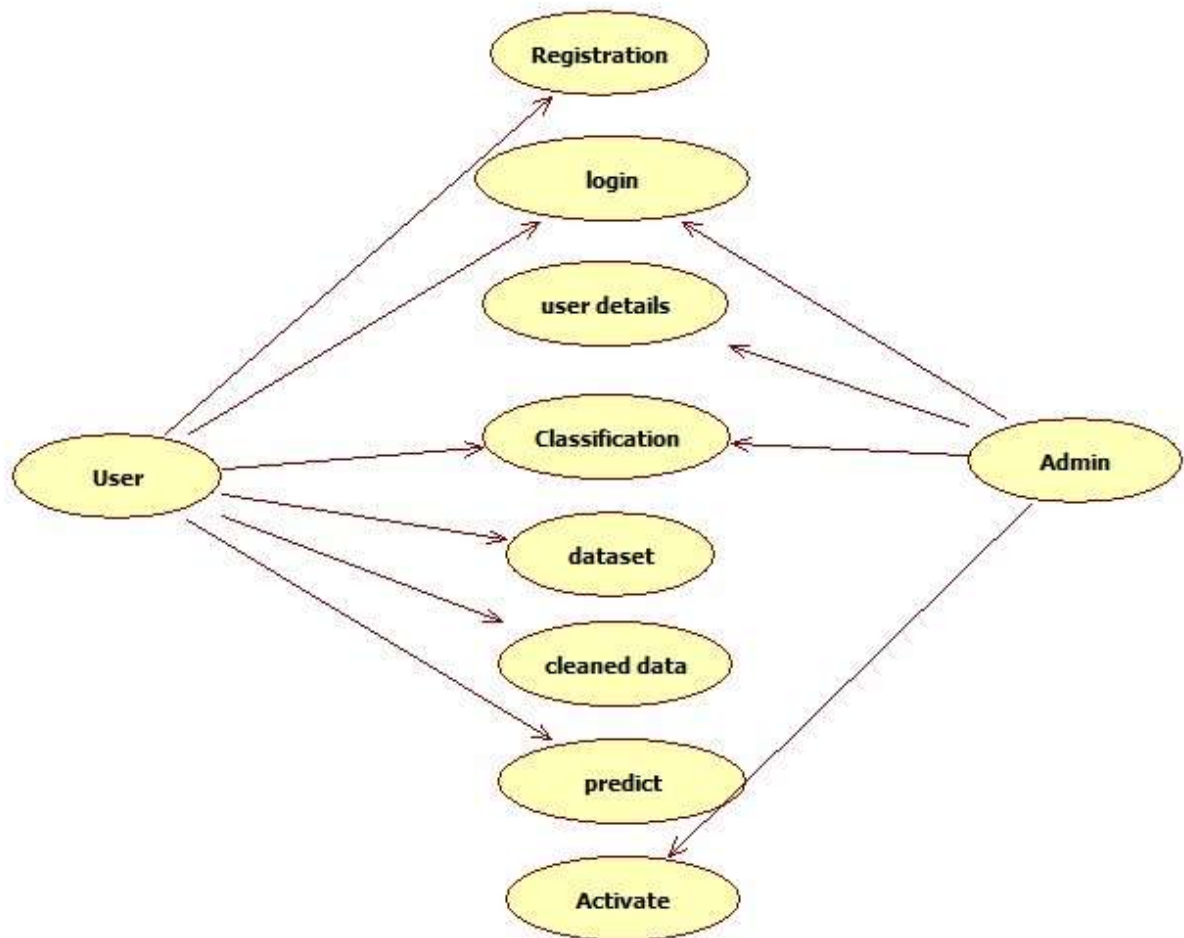
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

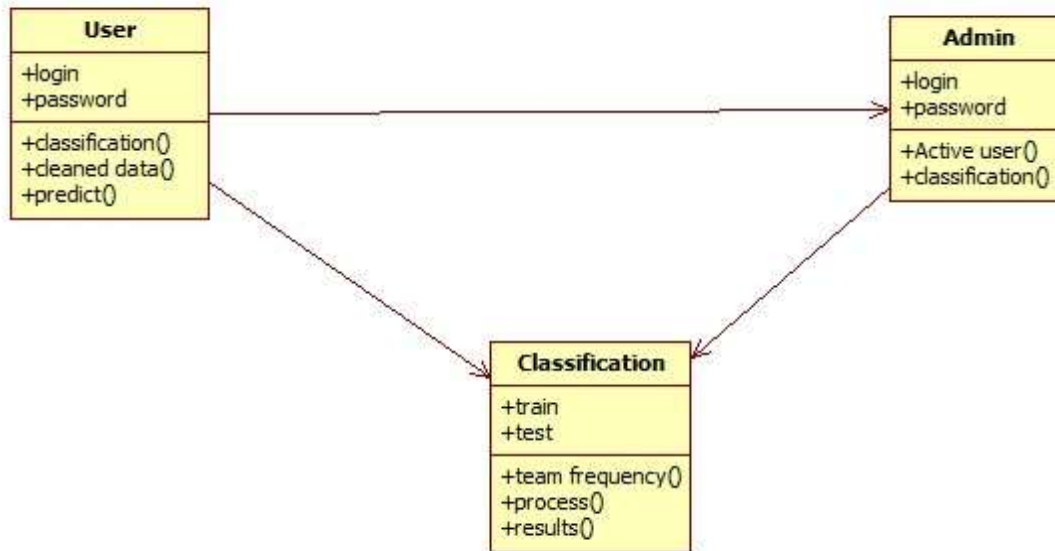
3.4.1.1 Use Case Diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



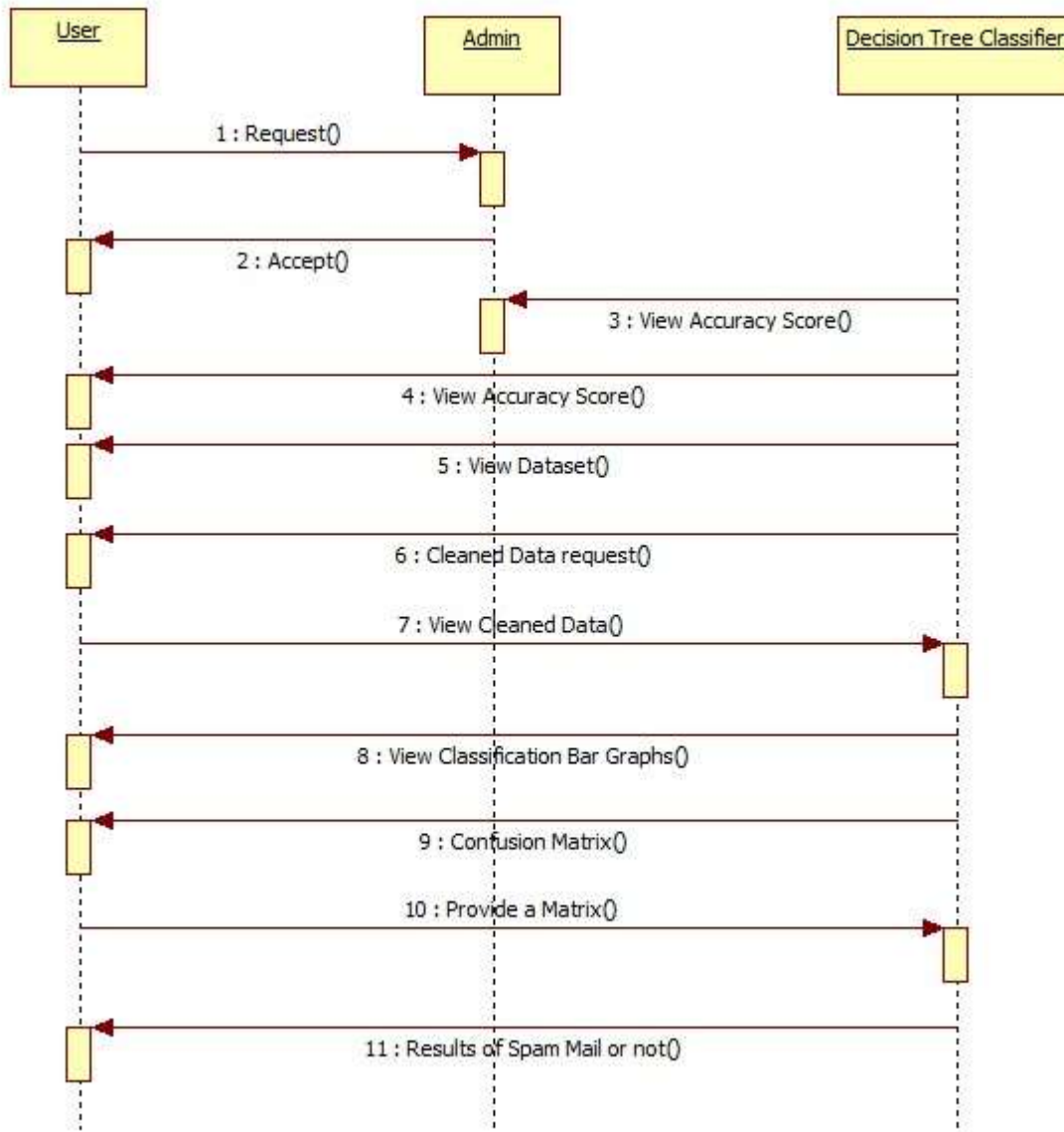
3.4.1.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



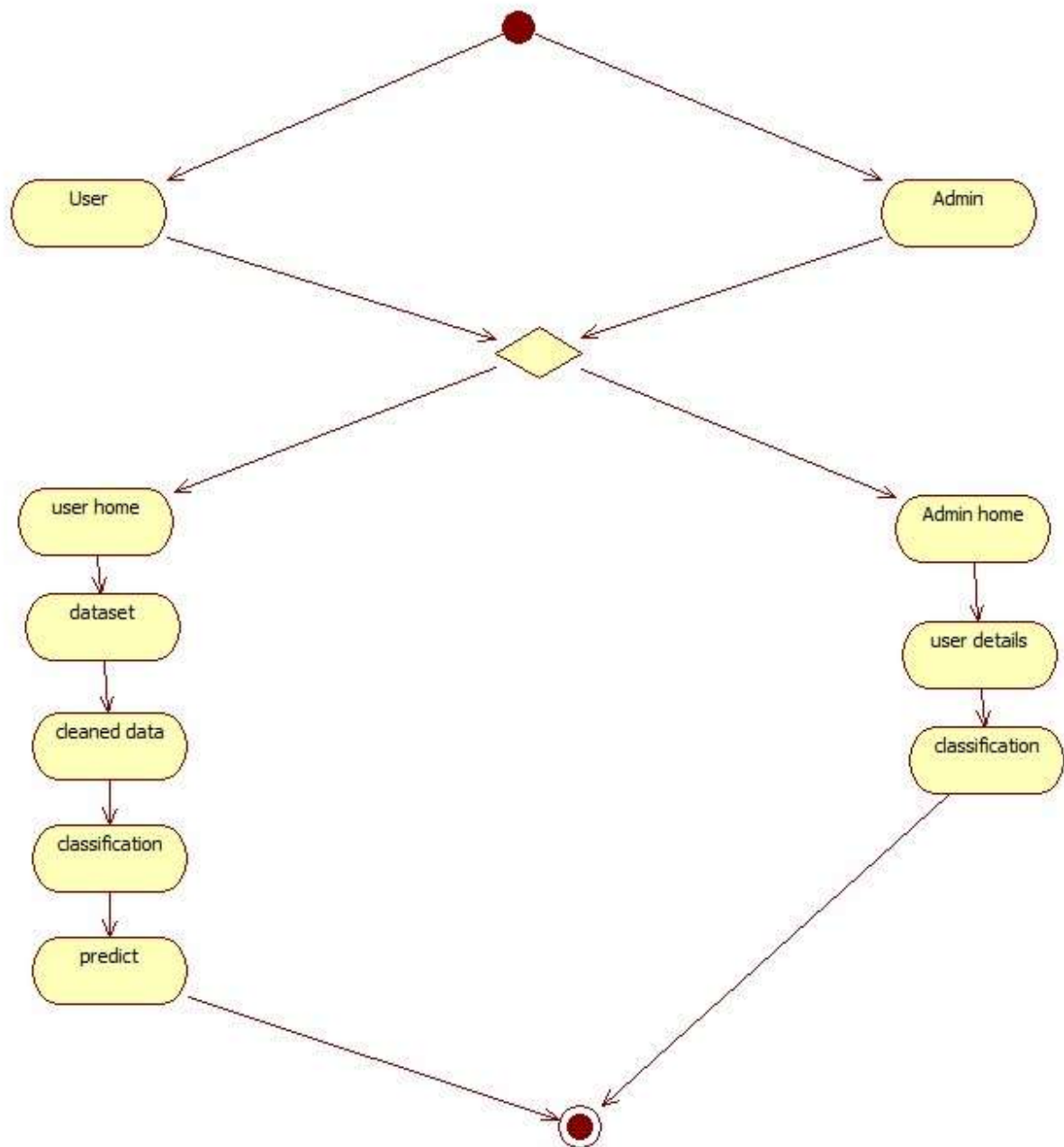
3.4.1.3SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



3.4.1.4ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

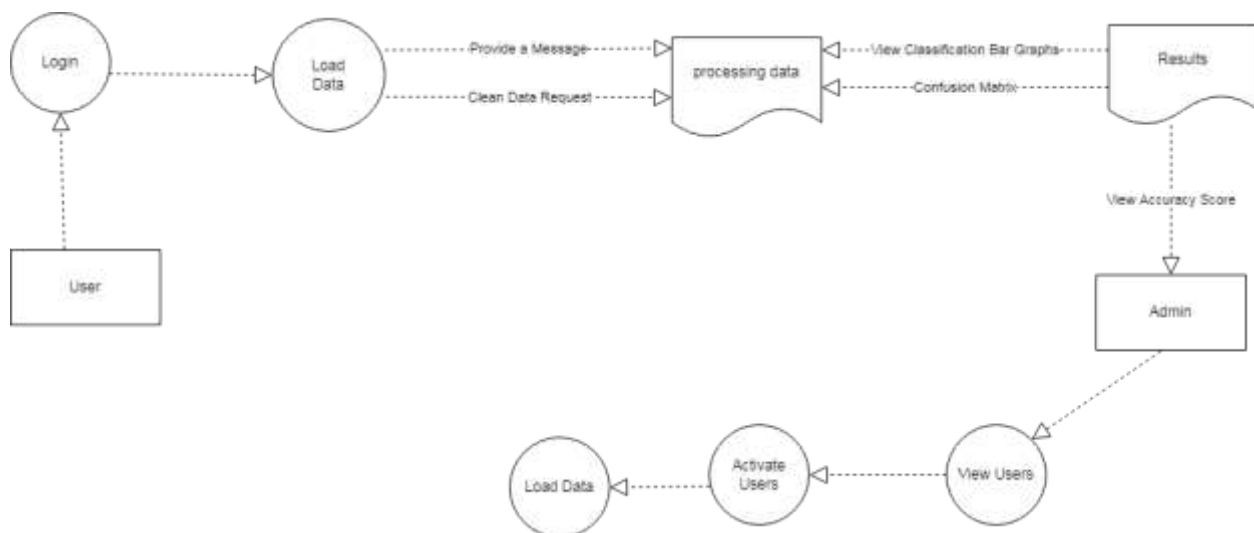


3.4.2 DATA FLOW DIAGRAM:

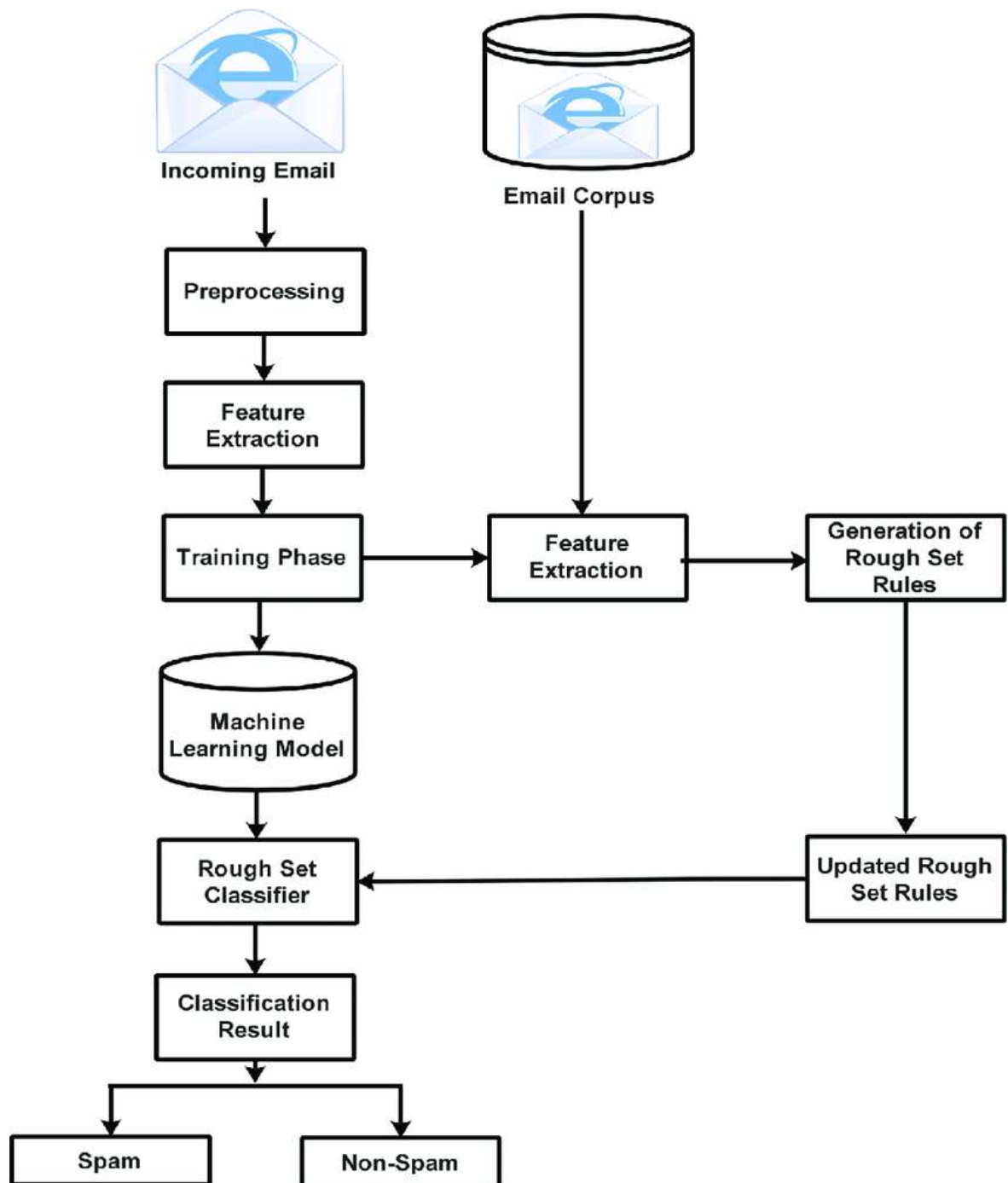
1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process,

the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



SYSTEM ARCHITECTURE:



4.SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

4.1 Testing Schemes

4.1.1 Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

4.1.2 testing Integration:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

4.1.3 Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

4.1.5 White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

4.1.6 Box Testing Black

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

4.1.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results : All the test cases mentioned above passed successfully. No defects encountered.

4.2 Test cases:

S.NO	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2.	User Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3.	User View User	Show our dataset	Pass	If Data set Not Available fail.
4.	User Machine Learning	For our Models the accuracy and Precision Score will calculated.	Pass	If Accuracy and Precision Score Not Displaying fail
5.	User Prediction	Display Review with true results	Pass	Results not True Fail
6.	User Classification	Display Visualization Results	Pass	Results not true Fail
7.	View Confusion Matrix Results	Display Confusion Matrix Results	Pass	If Results not Displayed Fail.
8.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
9.	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login
10.	Results	For our Three models the accuracy and Precision Score	Pass	If Accuracy And Precision Score Not Displayed fail

5. IMPLEMENTATION:

MODULES:

- User
- Admin
- Data Preprocessing
- Machine Learning

MODULES DESCRIPTION:

User:

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the user. Once admin activated the user then user can login into our system. User can upload the dataset based on our dataset column matched. For algorithm execution data must be in float format. Here we took Email dataset for testing purpose. User can also add the new data for existing dataset based on our Django application. User can click the Classification in the web page so that the data calculated Accuracy and Precision, Recall based on the algorithms. User can click

Prediction in the web page so that user can write the review after predict the review That will display results depends upon review like postive,negative or neutral

Admin:

Admin can login with his login details. Admin can activate the registered users. Once he activate then only the user can login into our system. Admin can view the overall data in the browser. Admin can click the Results in the web page so calculated Accuracy and Precision, Recall based on the algorithms is displayed. All algorithms execution complete then admin can see the overall accuracy in web page.

Data Preprocessing:

A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data

objects are described by a number of features that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc. Features are often called as variables, characteristics, fields, attributes, or dimensions. The data preprocessing in this forecast uses techniques like removal of noise in the data, the expulsion of missing information, modifying default values if relevant and grouping of attributes for prediction at various levels.

Machine learning:

Based on the split criterion, the cleansed data is split into 60% training and 40% test, then the dataset is subjected to machine learning classifiers such as Iterative Dichotomiser3 (ID3). The accuracy and Precision, Recall of the classifiers was calculated and displayed in my results. The classifier which bags up the highest accuracy could be determined as the best classifier.

6.CONCLUSION

In this paper, decision tree-based classification is employed for spam email detection. A novel approach for feature selection and reduction is also presented. It is shown that the system achieves high accuracy with a few features and with relatively small training dataset. In the near future is planned to incorporate other classifiers and to compare their performances with the proposed approach.

6.1 FURTHER ENHANCEMENT

In the near future, it is planned to incorporate other classifiers and to compare their performances with the proposed approach.

REFERENCES

- [1] P. Sharma and U. Bhardwaj, Machine Learning based Spam E-Mail Detection, in *International Journal of Intelligent Engineering & Systems*, vol. 11, no. 3, 2017
- [2] A. S. Rajput, J. S. Sohal, V. Athavale, "Email Header Feature Extraction using Adaptive and Collaborative approach for Email Classification", in *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, ISSN: 2278-3075, vol.8, Issue 7S, May 2019
- [3] P. Kulkarni, J.R. Saini and H. Acharya, "Effect of Header-based Features on Accuracy of Classifiers for Spam Email Classification", in: *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 3, 2020
- [4] E. G. Dada, S. B. Joseph, H. Chiroma, S. Abdulhamid, A. Adetunmbi, E. Opeyemi and Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems". in *Heliyon*, June 2019
- [5] E. M. Bahgat, S. Rady, W. Gad and I. F. Moawad, "Efficient email classification approach based on semantic methods", In: *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 3259-3269, December 2018.
- [6] F. Ruskanda, "Study on the Effect of Preprocessing Methods for Spam Email Detection", in: *Indonesian Journal on Computing (Indo-JC)*. 4. 109, March 2019.
- [7] A. Sharma, Manisha, D. Manisha and D.R. Jain, "Data Pre-Processing in Spam Detection", in: *International Journal of Science Technology & Engineering (IJSTE)*, vol. 1, Issue 11, May 2015
- [8] L. Shi, Q. Wang, X. Ma, M. Weng and H. Qiao, "Spam Email Classification Using Decision Tree Ensemble", in *Journal of Computational Information Systems*, March 2012
- [9] S. Balamurugan and R. Rajaram, "Suspicious E-mail Detection via

Decision Tree: A Data Mining Approach”, January 2007.

[10] T. A. Almeida and J.M. Gomez Hidalgo, SMS Spam Collection, *UCIMachineLearningRepository*, viewed 12 September 2020, <https://archive.ics.uci.edu/ml/datasets/sms+spam+collection>

[11] C. D. Manning, P. Raghavan and H. Schutze, “Introduction to Information Retrieval”, in *Cambridge University Press*, 2008.

[12] A. Bhowmick and S. M. Hazarika, “Machine Learning for E-mail Spam Filtering: Review, Techniques and Trends”, 2016

[13] J. Grus, “Data Science from Scratch: First Principles with Python”, *O’Reilly Media . Inc .*, April 2015

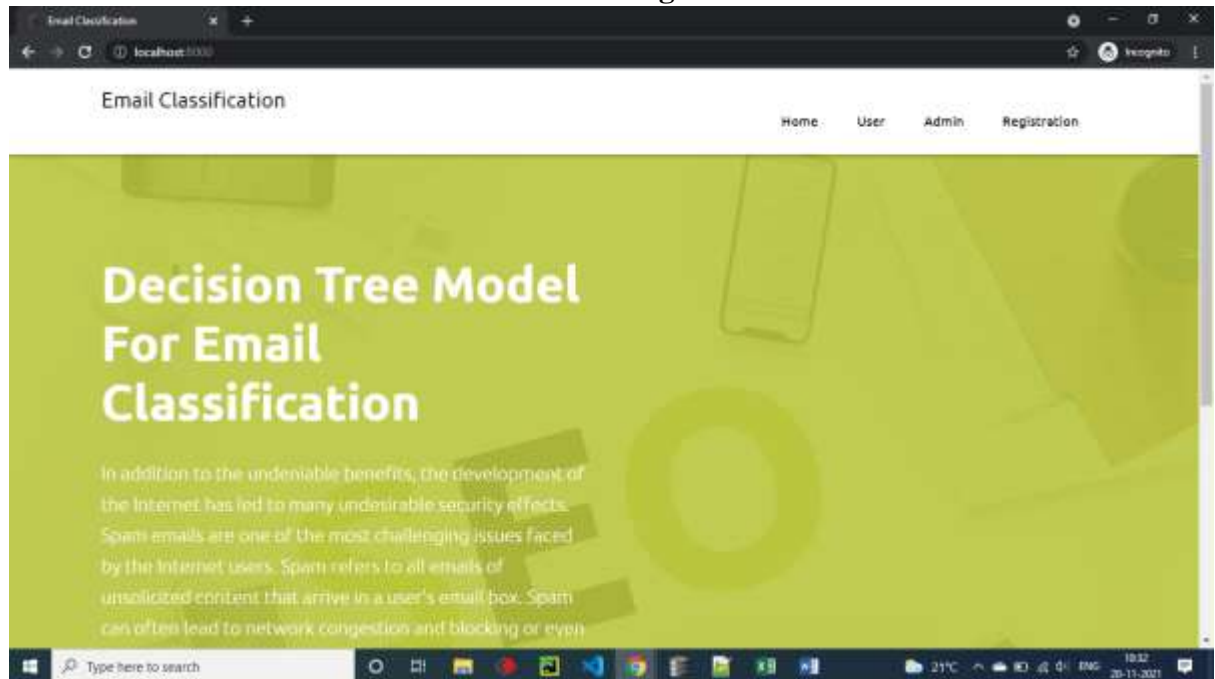
[14] I.H. Witten and E. Frank, “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, *Morgan Kaufmann*, San Francisco, 2000

[15] T. Kristensen and G. Kumar, “Entropy based disease classification of proteomic mass spectrometry data of the human serum by a support vector machine”, *Proceedings . 2005 IEEE International Joint Conference on Neural Networks*, 2005

APPENDICES:

APPENDIX A: Screens

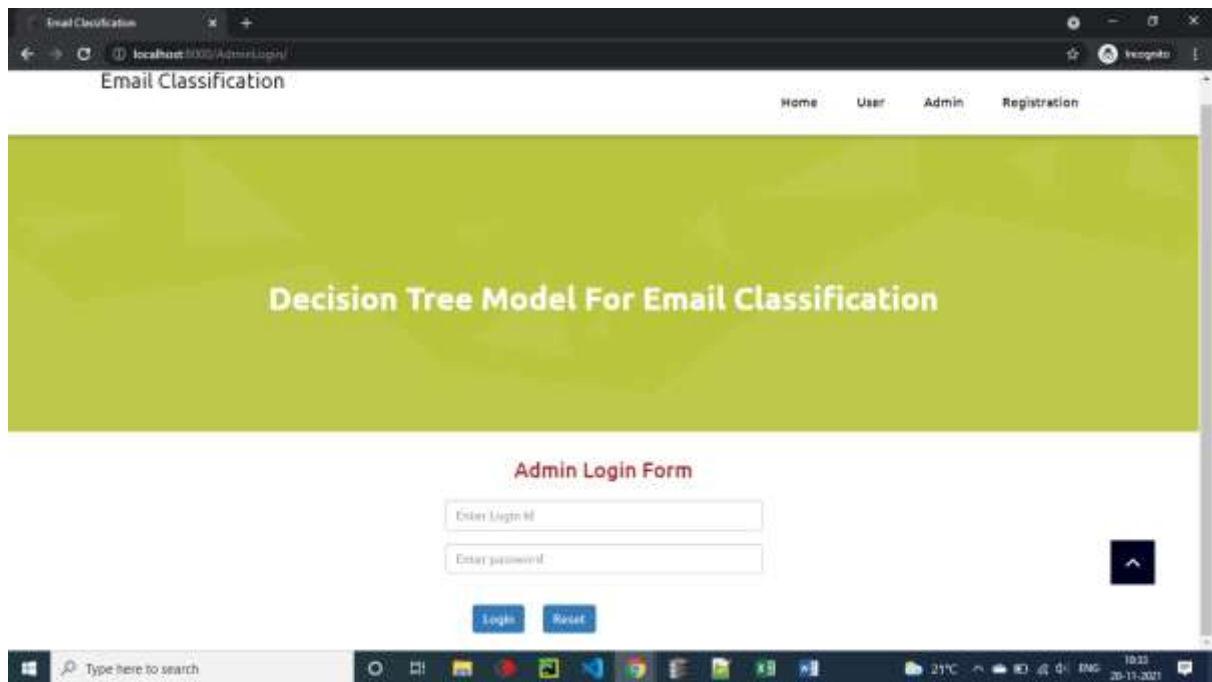
- Screen Shots
- Home Page



- User Register Form

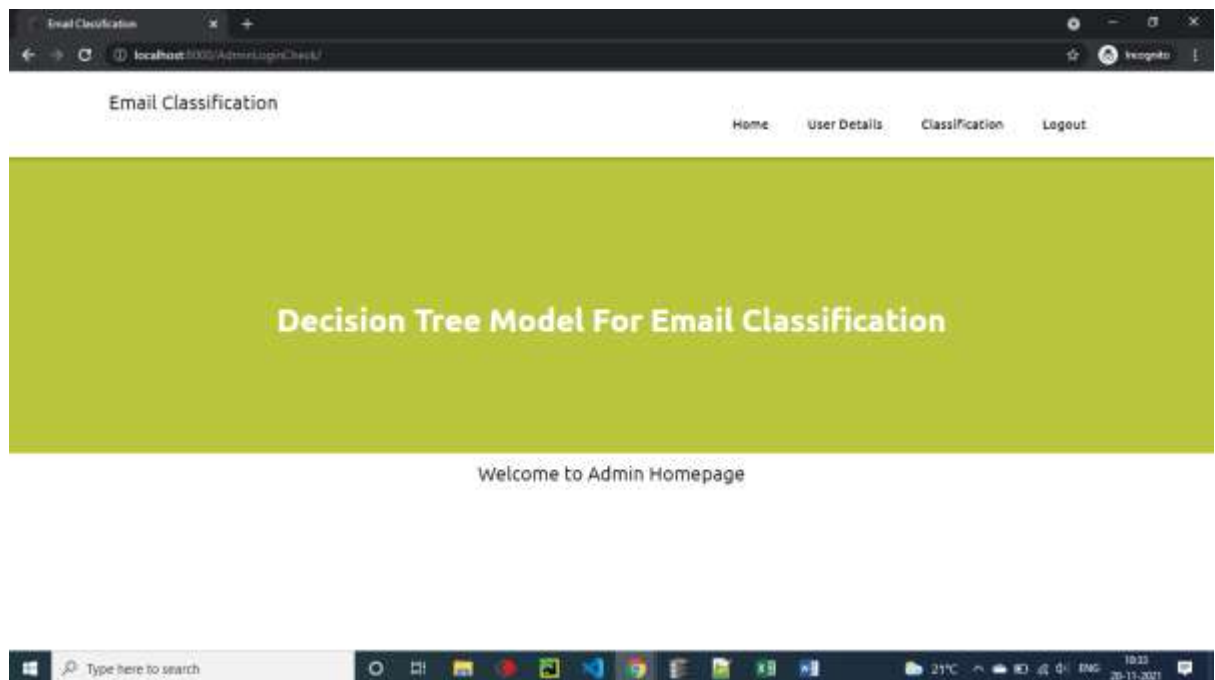
The screenshot shows a web browser window titled "Email Classification" with the URL "localhost:8000/User/register/". The page has a navigation bar with links for "Home", "User", "Admin", and "Registration". The main content area has a green header bar and the title "User Register Form". Below the title, there is a registration form with the following fields: "User Name", "Login ID", "Password", "Mobile", "email", "Locality", "Address", "City", and "State". A "Register" button is located at the bottom of the form. The Windows taskbar is visible at the bottom.

- Admin Login Form



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/AdminLogin/'. The page title is 'Email Classification'. The navigation bar includes links for 'Home', 'User', 'Admin', and 'Registration'. The main content area features a large green banner with the text 'Decision Tree Model For Email Classification'. Below the banner is the 'Admin Login Form', which consists of two input fields labeled 'Enter Login Id' and 'Enter password', followed by 'Login' and 'Reset' buttons. A Windows taskbar is visible at the bottom of the browser window.

- Admin Home page



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/AdminLoginCheck/'. The page title is 'Email Classification'. The navigation bar includes links for 'Home', 'User Details', 'Classification', and 'Logout'. The main content area features a large green banner with the text 'Decision Tree Model For Email Classification'. Below the banner, the text 'Welcome to Admin Homepage' is displayed. A Windows taskbar is visible at the bottom of the browser window.

- **Activate Users**

The screenshot shows a web browser window with the URL `localhost:8000/RegisterUserView`. The page has a navigation bar with links: Home, User Details, Classification, and Logout. Below the navigation bar is a large green banner with the text "Decision Tree Model For Email Classification". Underneath the banner is a link "View RegisterUser Details". Below this link is a table with the following data:

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	chakri	chakri	9874323432	chakri@gmail.com	hyd	activated	Activated
2	teju	teju	9874453232	teju@gmail.com	hyd	activated	Activated
3	alex	alex	9849096490	alex@gmail.com	Hyderabad	activated	Activated
4	kattamma	kattamma	9701278789	alexcodechallenge@gmail.com	Vijayawada	activated	Activated

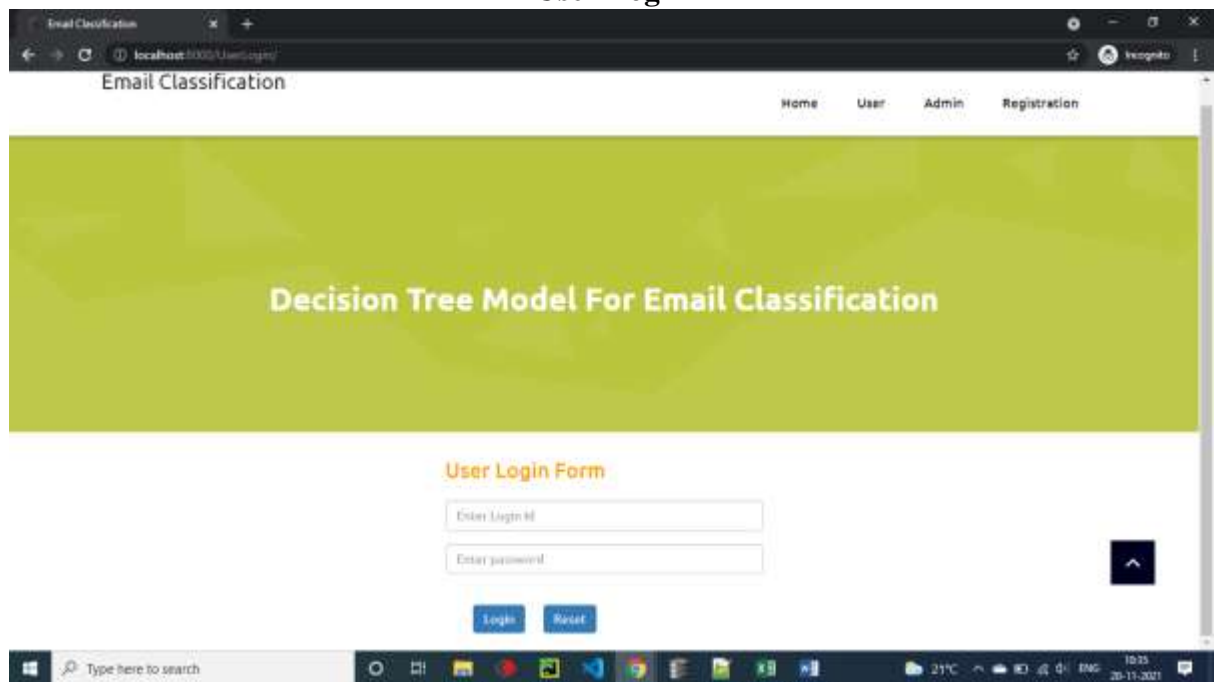
At the bottom of the table, there is a button labeled "Activate" with a blue background and a white upward-pointing arrow.

- **Admin View Results**

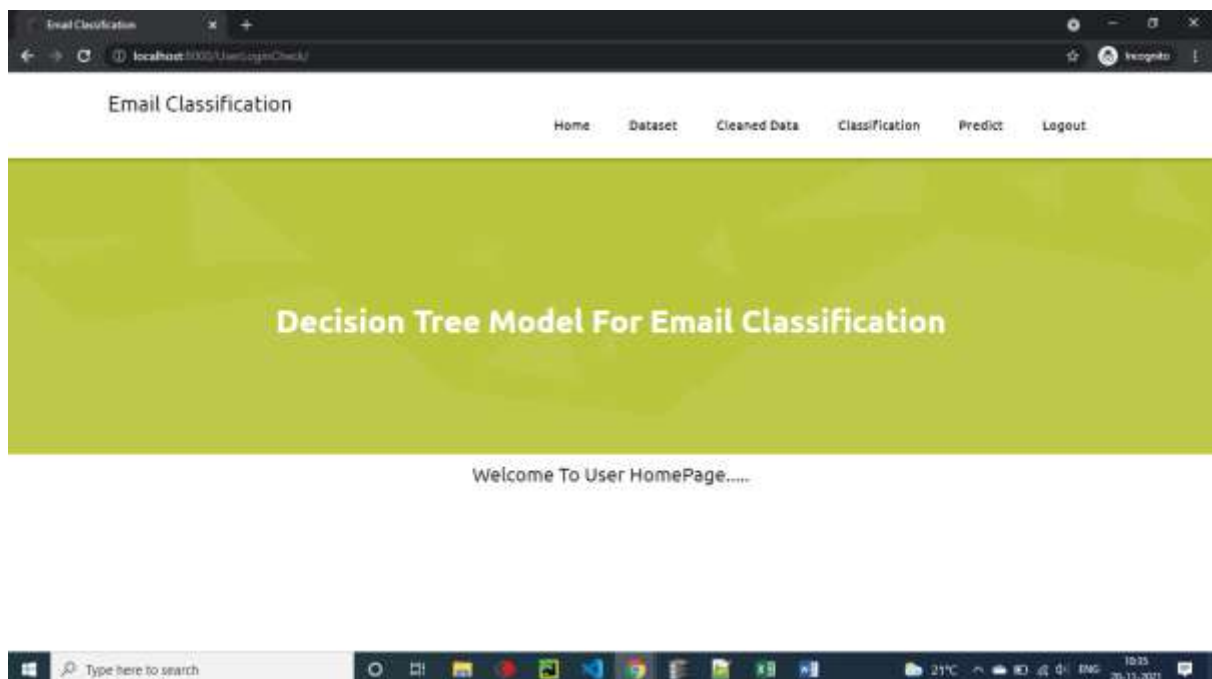
The screenshot shows a web browser window with the URL `localhost:8000/admin_classification`. The page has a navigation bar with links: Home, User Details, Classification, and Logout. Below the navigation bar is a large green banner with the text "Decision Tree Model For Email Classification". Underneath the banner is a link "Decision Tree Model Results". Below this link is a text area displaying the following results:

Accuracy 96.42857142857143
Precision 69.23076923076923
Recall 75.0

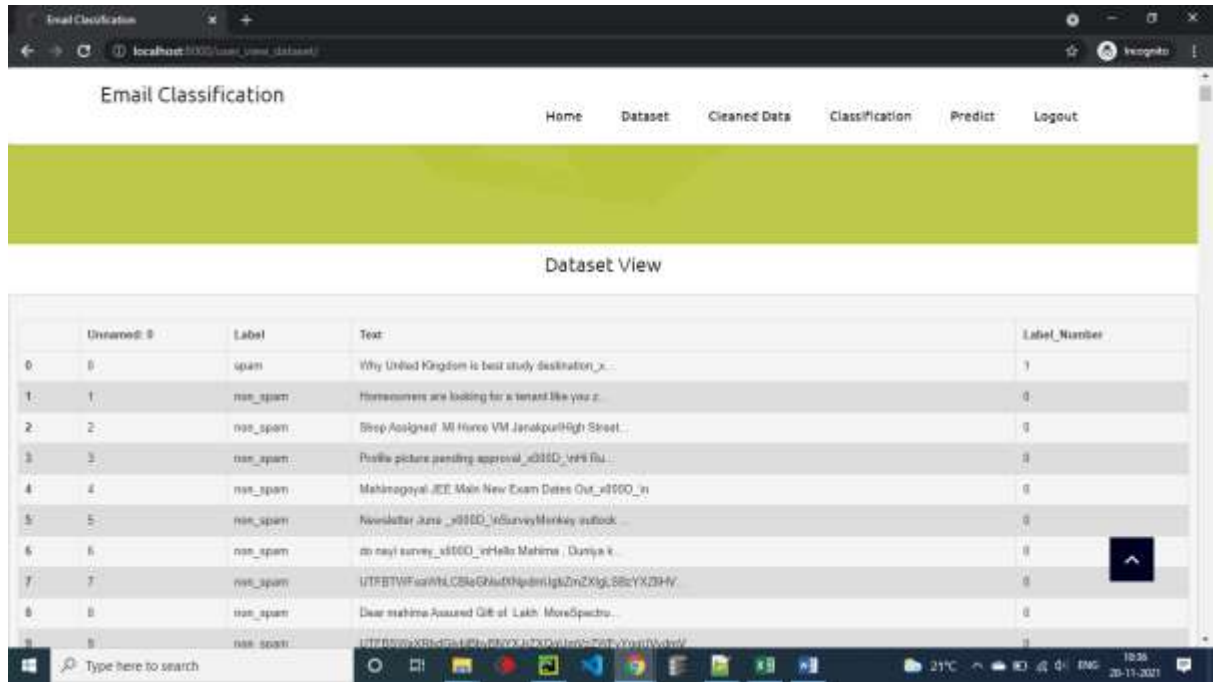
- **User Login**



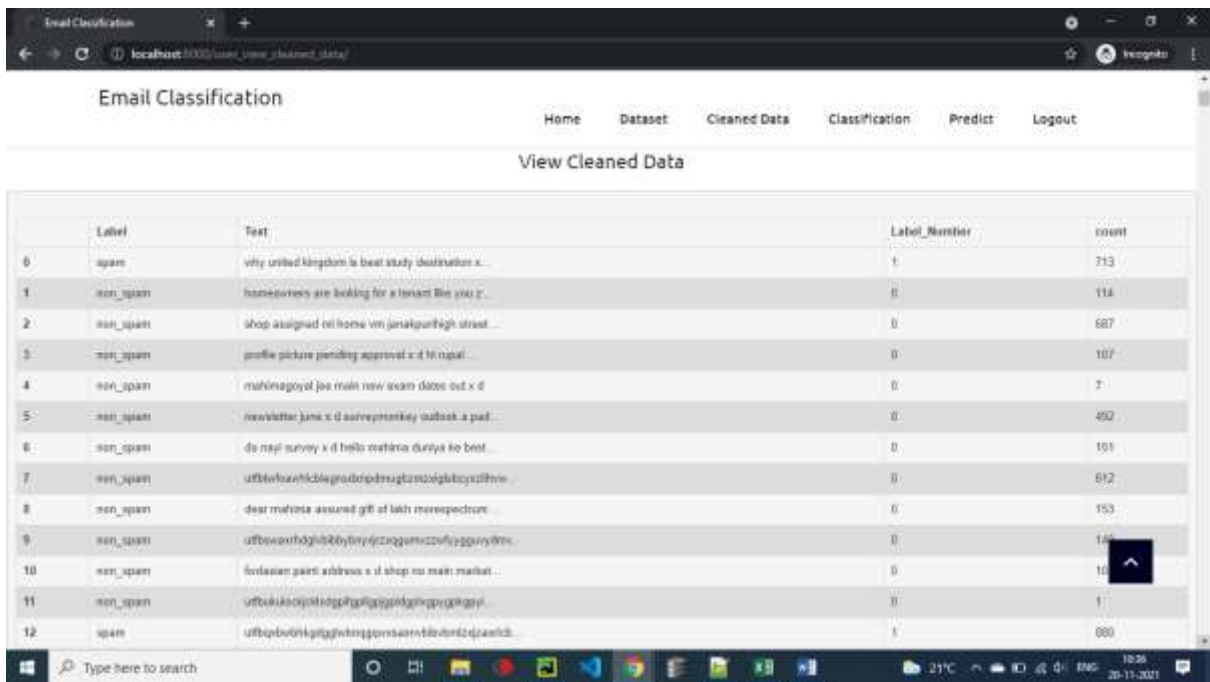
- **User Home Page**



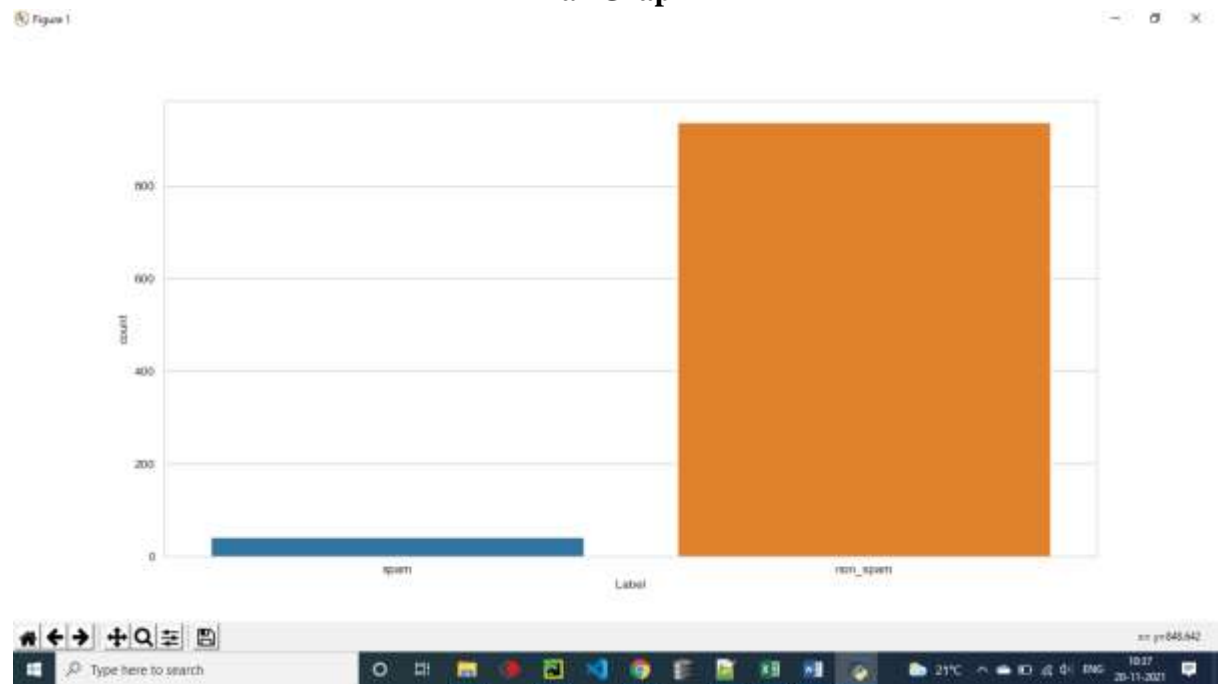
- **View Dataset**



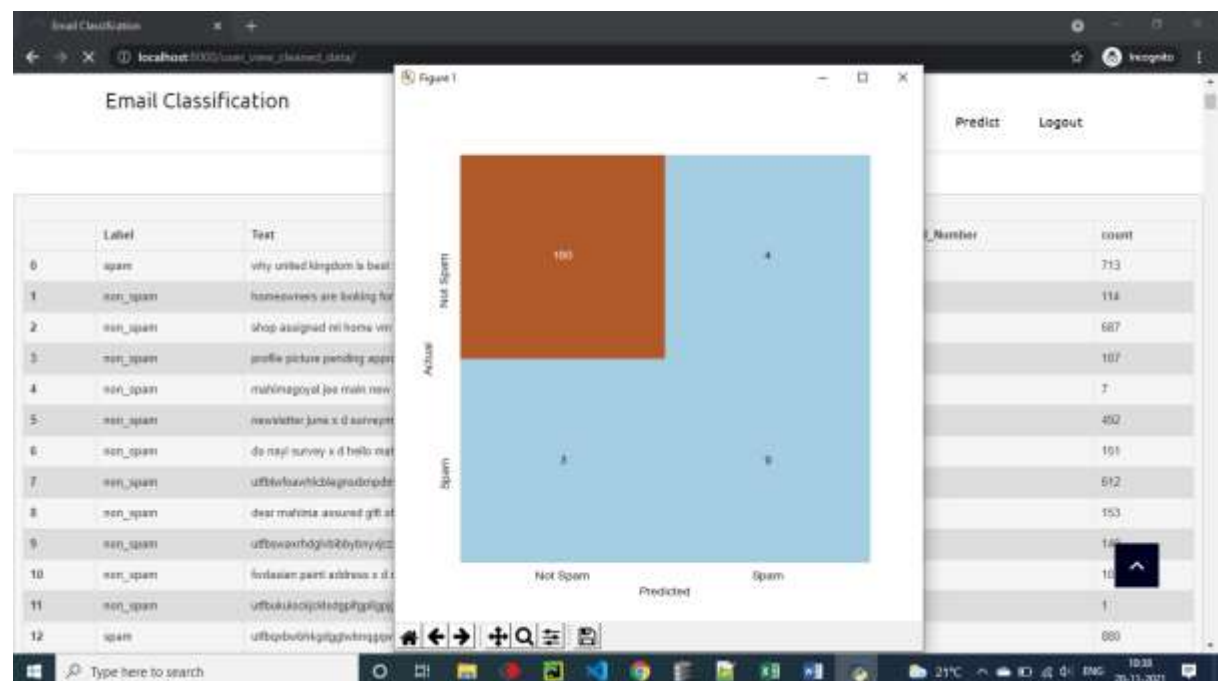
- **View Cleaned Data**



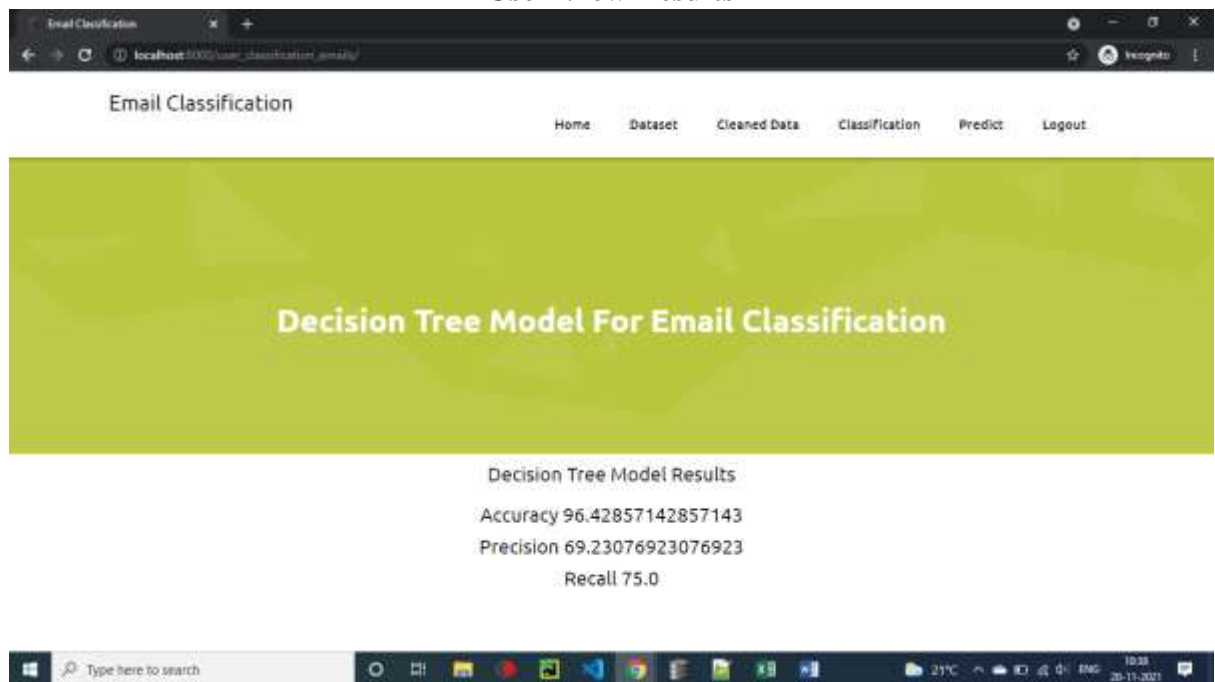
- Bar Graph



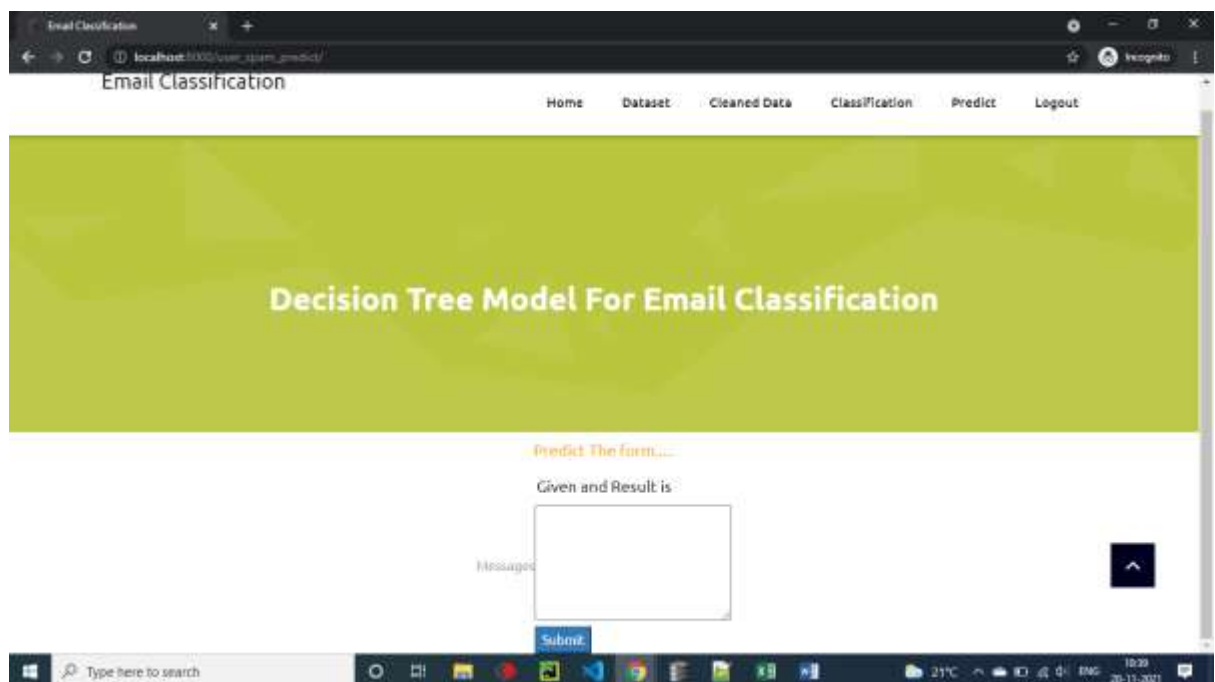
- Confusion matrix



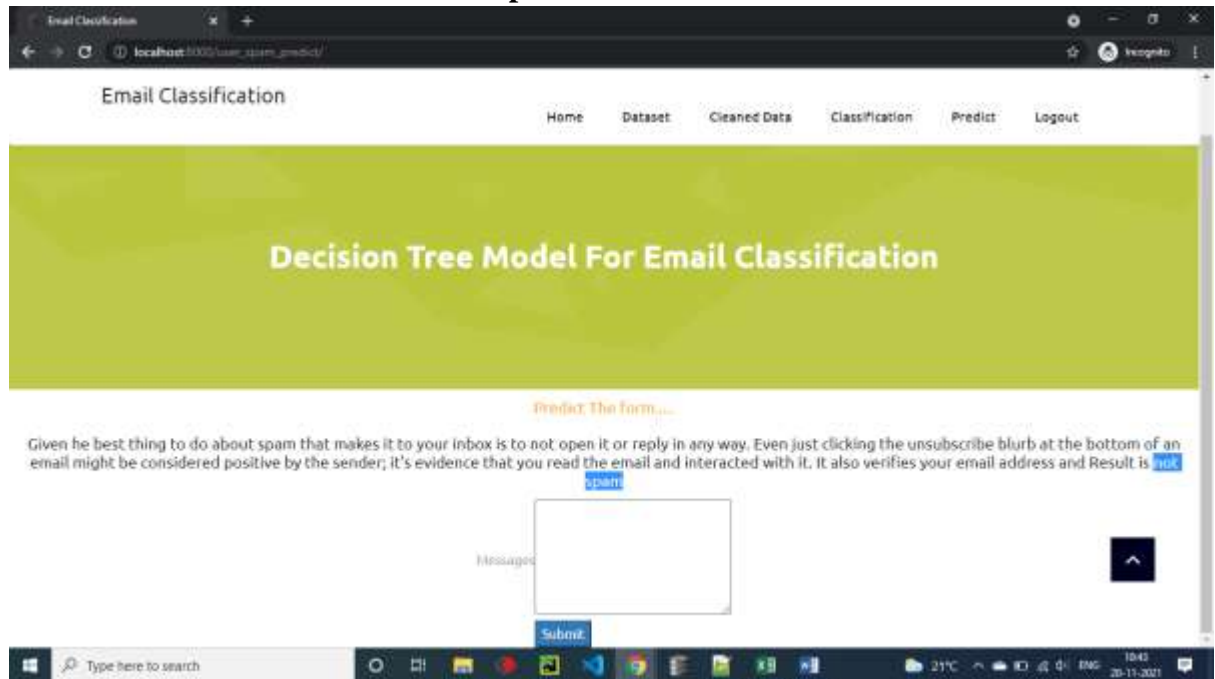
- **User View Results**



- **Prediction Form**



- Spam Test Result



APPENDIX B: Source code

User side views.py

```
from django.shortcuts import render

# Create your views here.

from django.shortcuts import render, HttpResponseRedirect
from django.contrib import messages
from .forms import UserRegistrationForm
from .models import UserRegistrationModel

# Create your views here.
def UserRegisterActions(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            print('Data is Valid')
            form.save()
            messages.success(request, 'You have been successfully registered')
            form = UserRegistrationForm()
            return render(request, 'UserRegistrations.html', {'form': form})
        else:
            messages.success(request, 'Email or Mobile Already Existed')
            print("Invalid form")
    else:
        form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
```

```

def UserLoginCheck(request):
    if request.method == "POST":
        loginid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')
        print("Login ID = ", loginid, ' Password = ', pswd)
        try:
            check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
                request.session['id'] = check.id
                request.session['loggeduser'] = check.name
                request.session['loginid'] = loginid
                request.session['email'] = check.email
                print("User id At", check.id, status)
                return render(request, 'users/UserHomePage.html', {})
            else:
                messages.success(request, 'Your Account Not at activated')
                return render(request, 'UserLogin.html')
        except Exception as e:
            print('Exception is ', str(e))
            pass
            messages.success(request, 'Invalid Login id and password')
    return render(request, 'UserLogin.html', {})

def UserHome(request):
    return render(request, 'users/UserHomePage.html', {})

```

```

def user_view_dataset(request):
    import pandas as pd
    from django.conf import settings
    path = settings.MEDIA_ROOT + "\\\" + 'All_Emails.xlsx'
    df = pd.read_excel(path)
    df = df.to_html
    return render(request, 'users/dataset_view.html', {'data': df})


def user_classification_emails(request):
    from .utility import processAlgorithms
    accuracy, precision, recall = processAlgorithms.start_process()
    return render(request, 'users/view_classification_result.html',
                  {'accuracy': accuracy, 'precision': precision, 'recall': recall})


def user_view_cleaned_data(request):
    from .utility import cleanedData
    df = cleanedData.start_cleaning_process();
    df = df.to_html
    return render(request, 'users/view_cleaned_data.html', {'data': df})


def user_spam_predict(request):
    if request.method == 'POST':
        mailBody = request.POST.get("mailbody")
        from .utility import checkSpamOrNot
        result = checkSpamOrNot.test_mail_is_spam_or_not(mailBody)
        return render(request, 'users/test_results.html', {'mail': mailBody, 'result': result})
    else:

```

```

        return render(request, 'users/test_results.html', { })

forms.py

from django import forms

from .models import UserRegistrationModel


class UserRegistrationForm(forms.ModelForm):

    name = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}), required=True,
max_length=100)

    loginid = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}), required=True,
max_length=100)

    password = forms.CharField(widget=forms.PasswordInput(attrs={'pattern': '(?=.*\d)(?=.*[a-
z])(?=.*[A-Z]).{8,}',

                                'title': 'Must contain at least one number and one uppercase
and lowercase letter, and at least 8 or more characters'})),

                                required=True, max_length=100)

    mobile = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[56789][0-9]{9}'}),
required=True,

                                max_length=100)

    email = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-z0-9._%+-]+@[a-z0-9.-
]+\.[a-z]{2,}$'}),

                                required=True, max_length=100)

    locality = forms.CharField(widget=forms.TextInput(), required=True, max_length=100)

    address = forms.CharField(widget=forms.Textarea(attrs={'rows': 4, 'cols': 22}), required=True,
max_length=250)

    city = forms.CharField(widget=forms.TextInput(

        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}),
required=True,

        max_length=100)

    state = forms.CharField(widget=forms.TextInput(

        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}),
required=True,

        max_length=100)

    status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

```

```
class Meta():  
    model = UserRegistrationModel  
    fields = '__all__'
```

```
processAlgorithm.py  
  
import matplotlib.pyplot as plt  
import pandas as pd  
import seaborn as sns  
from django.conf import settings  
  
sns.set_style('whitegrid')  
  
from nltk.tokenize import word_tokenize  
from nltk.tokenize import RegexpTokenizer  
from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score  
  
def count_words(text):  
    words = word_tokenize(text)  
    return len(words)  
  
# %%time  
def clean_str(string, reg=RegexpTokenizer(r'[a-z]+')):  
    # Clean a string with RegexpTokenizer  
    string = string.lower()  
    tokens = reg.tokenize(string)  
    return " ".join(tokens)
```

```
from nltk.stem import PorterStemmer
```

```
stemmer = PorterStemmer()
```

```
def stemming(text):
```

```
    return ".join([stemmer.stem(word) for word in text])
```

```
def start_process():
```

```
    path = settings.MEDIA_ROOT + "\\\" + 'All_Emails.xlsx'
```

```
    df = pd.read_excel(path)
```

```
    df.drop('Unnamed: 0', axis=1, inplace=True)
```

```
    df.columns = ['Label', 'Text', 'Label_Number']
```

```
    # print(df.head())
```

```
    plt.figure(figsize=(8, 6))
```

```
    sns.countplot(data=df, x='Label')
```

```
    plt.show()
```

```
    df['count'] = df['Text'].apply(count_words)
```

```
    print(df['count'])
```

```
    df.groupby('Label_Number')['count'].mean()
```

```
    print('Before cleaning:')
```

```
    print(df.head())
```

```
    print('After cleaning:')
```

```
    df['Text'] = df['Text'].apply(lambda string: clean_str(string))
```

```
    print(df.head())
```

```
    df['Text'] = df['Text'].apply(stemming)
```

```
    print(df.head())
```



```

X = df.loc[:, 'Text']
y = df.loc[:, 'Label_Number']

print(f"Shape of X: {X.shape}\nshape of y: {y.shape}")

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=11)
print(f"Training Data Shape: {X_train.shape}\nTest Data Shape: {X_test.shape}")
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
cv.fit(X_train)
print('No.of Tokens: ', len(cv.vocabulary_.keys()))
dtv = cv.transform(X_train)
type(dtv)
dtv = dtv.toarray()
print(f"Number of Observations: {dtv.shape[0]}\nTokens/Features: {dtv.shape[1]}")
dtv[1]

from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC, SVC
from time import perf_counter
import warnings
warnings.filterwarnings(action='ignore')
models = {
    "Random Forest": {"model": RandomForestClassifier(), "perf": 0},
    "MultinomialNB": {"model": MultinomialNB(), "perf": 0},
    "Logistic Regr.": {"model": LogisticRegression(solver='liblinear', penalty='l2', C=1.0), "perf":
0},

```

```

"KNN": {"model": KNeighborsClassifier(), "perf": 0},
"Decision Tree": {"model": DecisionTreeClassifier(), "perf": 0},
"SVM (Linear)": {"model": LinearSVC(), "perf": 0},
"SVM (RBF)": {"model": SVC(), "perf": 0}
}

for name, model in models.items():
    start = perf_counter()
    model['model'].fit(dtv, y_train)
    duration = perf_counter() - start
    duration = round(duration, 2)
    model["perf"] = duration
    print(f"{name:20} trained in {duration} sec")
test_dtv = cv.transform(X_test)
test_dtv = test_dtv.toarray()
print(f"Number of Observations: {test_dtv.shape[0]}\nTokens: {test_dtv.shape[1]}")
models_accuracy = []
for name, model in models.items():
    models_accuracy.append([name, model["model"].score(test_dtv, y_test), model["perf"]])
df_accuracy = pd.DataFrame(models_accuracy)
df_accuracy.columns = ['Model', 'Test Accuracy', 'Training time (sec)']
df_accuracy.sort_values(by='Test Accuracy', ascending=False, inplace=True)
df_accuracy.reset_index(drop=True, inplace=True)
print(df_accuracy)

# plt.figure(figsize=(15, 5))
# sns.barplot(x='Model', y='Test Accuracy', data=df_accuracy)
# plt.title('Accuracy on the test set\n', fontsize=15)
# plt.ylim(0.825, 1)
# plt.show()

```

```

# plt.figure(figsize=(15, 5))

# sns.barplot(x='Model', y='Training time (sec)', data=df_accuracy)

# plt.title("Training time for each model in sec", fontsize=15)

# plt.ylim(0, 1)

# plt.show()


lr = LogisticRegression(solver='liblinear', penalty='l2', C=1.0)

lr.fit(dtv, y_train)

pred = lr.predict(test_dtv)

print('Accuracy: ', accuracy_score(y_test, pred) * 100)

print(classification_report(y_test, pred))

# confusion_matrix = pd.crosstab(y_test, pred, rownames=['Actual'], colnames=['Predicted'])

# plt.figure(figsize=(6, 6))

# sns.heatmap(confusion_matrix, annot=True, cmap='Paired', cbar=False, fmt="d",
xticklabels=['Not Spam', 'Spam'],

#         yticklabels=['Not Spam', 'Spam']))

# plt.show()


# Decision Tree Classifier

dtc = DecisionTreeClassifier()

dtc.fit(dtv, y_train)

pred = dtc.predict(test_dtv)

accuracy = accuracy_score(y_test, pred) * 100

precision = precision_score(y_test, pred) * 100

recall = recall_score(y_test, pred) * 100

print(classification_report(y_test, pred))

confusion_matrix = pd.crosstab(y_test, pred, rownames=['Actual'], colnames=['Predicted'])

plt.figure(figsize=(6, 6))

sns.heatmap(confusion_matrix, annot=True, cmap='Paired', cbar=False, fmt="d",
xticklabels=['Not Spam', 'Spam'],

```

```

        yticklabels=['Not Spam', 'Spam']);

plt.show()

return accuracy,precision,recall
Cleaned_Data.py
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from django.conf import settings

sns.set_style('whitegrid')
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer

def count_words(text):
    words = word_tokenize(text)
    return len(words)

# %%time
def clean_str(string, reg=RegexpTokenizer(r'[a-z]+')):
    # Clean a string with RegexpTokenizer
    string = string.lower()
    tokens = reg.tokenize(string)
    return " ".join(tokens)

from nltk.stem import PorterStemmer

```

```
stemmer = PorterStemmer()
```

```
def stemming(text):
```

```
    return ".join([stemmer.stem(word) for word in text])
```

```
def start_cleaning_process():
```

```
    path = settings.MEDIA_ROOT + "\\\" + 'All_Emails.xlsx'
```

```
    df = pd.read_excel(path)
```

```
    df.drop('Unnamed: 0', axis=1, inplace=True)
```

```
    df.columns = ['Label', 'Text', 'Label_Number']
```

```
    # print(df.head())
```

```
    plt.figure(figsize=(8, 6))
```

```
    sns.countplot(data=df, x='Label')
```

```
    # plt.show()
```

```
    df['count'] = df['Text'].apply(count_words)
```

```
    print(df['count'])
```

```
    df.groupby('Label_Number')['count'].mean()
```

```
    print('Before cleaning:')
```

```
    print(df.head())
```

```
    print('After cleaning:')
```

```
    df['Text'] = df['Text'].apply(lambda string: clean_str(string))
```

```
    return df
```

```
Check_Spam_or_Not.py
```

```
# import the libraries
```

```
import sys
```

```
assert sys.version_info >= (3, 5)
```

```
# data manipulation
import pandas as pd
import numpy as np

# visualization
import seaborn as sns
# %matplotlib inline
from django.conf import settings
# consistent sized plot
from pylab import rcParams

rcParams['figure.figsize'] = 12, 5
rcParams['xtick.labelsize'] = 12
rcParams['ytick.labelsize'] = 12
rcParams['axes.labelsize'] = 12

# display options for dataframe
pd.options.display.max_columns = None

# text processing
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
# from fuzzywuzzy import process
# from fuzzywuzzy import fuzz

# text feature engineering
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
```

```
# models

from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import LinearSVC
```

```
# regular expressions
```

```
# string operations
import string
```

```
# compute articles similarity
from sklearn.metrics import confusion_matrix
```

```
# for file extraction
```

```
# ignore warnings
import warnings
```

```
warnings.filterwarnings(action='ignore', message='')
stemmer = PorterStemmer()
```

```
def simplify(text):
    """Function to handle the diacritics in the text"""
    import unicodedata
    try:
        text = text # unicode(text, 'utf-8')
```

```

except NameError:
    pass

text = unicodedata.normalize('NFD', text).encode('ascii', 'ignore').decode("utf-8")

return str(text)

```

```

def text_cleaning(text):
    """Function to clean the text"""
    # convert to lower case
    text = text.lower()
    # remove the email address
    text = text.replace(r'[a-zA-z0-9._]+@[a-zA-z0-9._]+', 'emailaddr')
    # replace the URL's
    text = text.replace(r'(http[s]?\\S+)((\\w+\\. [a-zA-Z]{2,4}\\S*))', 'httpaddr')
    # replace currency symbol with moneysymb
    text = text.replace(r'£\\$', 'moneysymb')
    # Replace phone numbers with 'phonenumbr'
    text = text.replace(r'\b(\\+\\d{1,2}\\s)?\\d?[\\-\\.]?\\d{3}\\)?[\\s.-]?\\d{3}[\\s.-]?\\d{4}\\b', 'phonenumbr')
    # remove the ip address
    text = text.replace(r'((2[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(\\.|$)){4}', '')
    # remove the user handles
    text = text.replace(r'@\\w+', '')
    # remove the string punctuation
    text = ''.join([txt for txt in word_tokenize(text) if txt not in string.punctuation])
    # replace the digits from the text with numbr
    text = text.replace(r'd', 'numbr')
    # remove all non alphabetical characters
    text = ''.join([txt for txt in word_tokenize(text) if txt.isalpha()])
    # replace multiple white space with a single one
    text = text.replace(r'\\s+', ' ')

```



```

# remove the leading and trailing whitespaces
text = text.replace(r'^\s+|\s+?$', '')

# remove the stop words
text = ' '.join([txt for txt in word_tokenize(text) if txt not in stopwords.words('english')])

# apply stemmer
text = ' '.join([stemmer.stem(txt) for txt in word_tokenize(text)])

# return the cleaned text
return text

```

```

def make_tidy(sample_space, train_scores, valid_scores):

    # Join train_scores and valid_scores, and label with sample_space

    messy_format = pd.DataFrame(np.stack((sample_space, train_scores.mean(axis=1),
    valid_scores.mean(axis=1)), axis=1),

                                columns=['# of training examples', 'Training set', 'Validation set'])

    # Re-structure into into tidy format

    return pd.melt(messy_format, id_vars='# of training examples', value_vars=['Training set',
    'Validation set'],

                    var_name='Scores', value_name='F1 score')

```

```

def test_mail_is_spam_or_not(mailBody):

    # load the spam collection mail file

    path = settings.MEDIA_ROOT + "\\\" + 'mailSpamCollection.csv'

    sms = pd.read_table(path, names=['label', 'message'])

    sms.head()

    sms.info()

    sms['label'].value_counts()

    # countplot of the categories of sentiments

    # sns.countplot(sms['label'])

```

```

# plt.title('Countplot of the MAIL labels (ham or spam)')
# plt.show()

sms.isna().sum()

# check for any empty string
blanks = []
for idx, rev, lab in sms.itertuples():
    if type(rev) == 'str':
        if rev.isspace():
            blanks.append(idx)
# print(blanks)
X = sms.drop('label', axis=1)
y = sms['label']
test_size = 0.3
random_state = 42
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
random_state=random_state,

                                                    stratify=sms['label'])

X_train.shape, X_test.shape
stemmer = PorterStemmer()

'''
Clean the text values in train and test
'''

preprocesses = [simplify, text_cleaning]

for preprocess in preprocesses:
    X_train['message'] = X_train['message'].apply(preprocess)
    X_test['message'] = X_test['message'].apply(preprocess)
# Construct a design matrix using an n-gram model and a tf-idf statistics
vectorizer = TfidfVectorizer(ngram_range=(1, 2))

```

```

counts = vectorizer.fit_transform(X_train['message'])
vocab = vectorizer.vocabulary_
test_counts = vectorizer.transform(X_test['message'])
# label encode the target features
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)
# Train SVM with a linear kernel on the training set
clf = LinearSVC(loss='hinge')
clf.fit(counts, y_train)
train_predictions = clf.predict(counts)
test_predictions = clf.predict(test_counts)
# print('Accuracy Score on the Train and Test Set')
# print('Train Accuracy = {}'.format(accuracy_score(y_train, train_predictions)))
# print('Test Accuracy = {}'.format(accuracy_score(y_test, test_predictions)))

# print('Classification Report on Test Set')
# print(classification_report(y_test, test_predictions))
# print('F-1 score on the test set')
# print('Test Set F1 Score = {}'.format(f1_score(y_test, test_predictions)))

# print('Confusion Matrix on the Test Set')
# print(confusion_matrix(y_test, test_predictions))
# Display a confusion matrix

pd.DataFrame(confusion_matrix(y_test, test_predictions), index=[['actual', 'actual'], ['spam',
'ham']],
              columns=[['predicted', 'predicted'], ['spam', 'ham']])

# select 10 different sizes of the entire training dataset. The test set will still be kept separate an an
unseen data

raw_text = X_train['message']

```

```

sample_space = np.linspace(500, len(raw_text) * .80, 10, dtype='int')

# Compute learning curves without regularization for the SVM model
train_size, train_scores, valid_scores = learning_curve(estimator=LinearSVC(loss='hinge',
C=1e10),

                X=counts, y=y_train,

                shuffle=True,

                train_sizes=sample_space,

                cv=StratifiedShuffleSplit(n_splits=10, test_size=0.2,

                random_state=42),

                scoring='f1',

                n_jobs=-1)

train_size
# check the train scores
train_scores
# check the test scores
valid_scores

# Initialize a FacetGrid object using the table of scores and facet on the type of score
g = sns.FacetGrid(make_tidy(sample_space, train_scores, valid_scores), hue='Scores', size=5)
# Plot the learning curves and add a legend
# g.map(plt.scatter, '# of training examples', 'F1 score')
# g.map(plt.plot, '# of training examples', 'F1 score').add_legend()
# plt.show()

''' Grid Search for the best hyperparameter '''
space = dict()
space['penalty'] = ['l1', 'l2', 'elasticnet']
space['loss'] = ['squared_hinge', 'hinge']
space['C'] = [1e10, 100]
# print(space)

```

```

clf = LinearSVC()
folds = StratifiedKFold (n_splits=10, shuffle=True, random_state=random_state)
grid_search = GridSearchCV(estimator=clf, param_grid=space, scoring='f1',
                             n_jobs=-1, cv=folds)
grid_result = grid_search.fit(counts, y_train)

grid_result.best_params_
# clf = LinearSVC(loss='hinge',penalty='l2',C=100)
clf = grid_result.best_estimator_
clf.fit(counts, y_train)
train_predictions = clf.predict(counts)
test_predictions = clf.predict(test_counts)

# print('Accuracy Score on the Train and Test Set')
# print('Train Accuracy = {}'.format(accuracy_score(y_train, train_predictions)))
# print('Test Accuracy = {}'.format(accuracy_score(y_test, test_predictions)))
# print('Classification Report on Test Set')
# print(classification_report(y_test, test_predictions))
# Display a confusion matrix

pd.DataFrame(confusion_matrix(y_test, test_predictions), index=[['actual', 'actual'], ['spam',
'ham']],
              columns=[['predicted', 'predicted'], ['spam', 'ham']])

# Display the features with the highest weights in the SVM model
pd.Series(clf.coef_.T.ravel(),
index=vectorizer.get_feature_names()).sort_values(ascending=False)[:20]

# function to decide whether a string is spam or not, and apply it on the hypothetical message from
earlier.

mail = 'Ohhh, but those are the best kind of foods'

```

```

rs = "
text = simplify(mailBody)
text_clean = text_cleaning(text)
if clf.predict(vectorizer.transform([text_clean])):
    rs = 'spam'
else:
    rs = 'not spam'
# print("Spam Results:", rs)
return rs

```

base.html

```

{% load static %}
<!DOCTYPE html>
<html lang="en">

<!-- Basic -->
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- Mobile Metas -->
<meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scale=1.0,
user-scalable=no">

<!-- Site Metas -->
<title>Email Classification</title>
<meta name="keywords" content="">
<meta name="description" content="">
<meta name="author" content="">

<!-- Site Icons -->
<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />
<link rel="apple-touch-icon" href="{% static 'images/apple-touch-icon.png' %}">

```

```

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="{ % static 'css/bootstrap.min.css' % }">

<!-- Site CSS -->
<link rel="stylesheet" href="{ % static 'style.css' % }">

<!-- Colors CSS -->
<link rel="stylesheet" href="{ % static 'css/colors.css' % }">

<!-- ALL VERSION CSS -->
<link rel="stylesheet" href="{ % static 'css/versions.css' % }">

<!-- Responsive CSS -->
<link rel="stylesheet" href="{ % static 'css/responsive.css' % }">

<!-- Custom CSS -->
<link rel="stylesheet" href="{ % static 'css/custom.css' % }">

<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->

</head>
<body class="seo_version">

  <!-- LOADER -->
  <div id="preloader">
    <div id="cupcake" class="box">
      <span class="letter">L</span>
      <div class="cupcakeCircle box">
        <div class="cupcakeInner box">
          <div class="cupcakeCore box"></div>
        </div>
      </div>
    </div>
  </div>

```

```

        </div>
        <span class="letter box">A</span>
        <span class="letter box">D</span>
        <span class="letter box">I</span>
        <span class="letter box">N</span>
        <span class="letter box">G</span>
    </div>
</div>
<!-- END LOADER -->

```

```

<header class="header header_style_01">
    <nav class="megamenu navbar navbar-default">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="{ %url 'index'% }"><h1>Email Classification</h1></a>
            </div>
            <div id="navbar" class="navbar-collapse collapse">
                <ul class="nav navbar-nav navbar-right hidden-md hidden-sm
hidden-xs">
                    <!--
                    <li><a class="btn-light btn-radius btn-brd top-btn" href="#"><i class="fa fa-
angle-double-right"></i>SEO Analysis</a></li>-->
                </ul>
                <ul class="nav navbar-nav navbar-right menu-top">
                    <li><a href="{ % url 'index' % }">Home</a></li>

```



```

        <li><a href="{ % url 'UserLogin' % }">User </a></li>
        <li><a href="{ % url 'AdminLogin' % }">Admin</a></li>
        <li><a href="{ % url 'UserRegister' % }">Registration</a></li>
<!--        <li><a href="{ % url 'index' % }">Pricing</a></li>-->
<!--        <li><a href="{ % url 'index' % }">Contact</a></li>-->
    </ul>
</div>
</div>
</nav>
</header>

<div class="all-page-title" style="background-image:url({ %static 'images/pattern-4.png'% });">
    <div class="container text-center">
        <h1>Decision Tree Model for Email Classification </h1>
    </div>

    <!--Page -->
    <div class="page-info">
        <div class="container">
            <div class="inner-container">
                <ul class="bread-crumb">
<!--        <li><a href="index.html">Home</a></li>-->
<!--        <li><span>About Company</span></li>-->

                </ul>
            </div>
        </div>
    </div>
    <!--End Page-->
</div><!-- end section -->

{ %block contents% }

```

```
<!--<div class="container" style="margin-top:-60px;">-->
<!--  <h2 align="center">Welcome to Homepage...</h2>-->
</div>
```

```
{%endblock% }
```

```
<a href="#" id="scroll-to-top" class="dmtop global-radius"><i class="fa fa-angle-up"></i></a>
```

```
<!-- ALL JS FILES -->
```

```
<script src="{ % static 'js/all.js' % }"></script>
```

```
<!-- ALL PLUGINS -->
```

```
<script src="{ % static 'js/custom.js' % }"></script>
```

```
</body>
```

```
</html>
```

