BUILDING A DYSLEXIA DETECTION WEB APP WITH BOOSTING AND FLASK

A PROJECT REPORT

Submitted by

ABDUR RAZAQ A [211420104003]

KARTHIKEYAN P [211420104124]

KIRUBHANIDHI M [211420104133]

in partial fulfillment for the award of the degreeof

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"BUILDING A DYSLEXIA DETECTION WEB APP WITH BOOSTING AND FLASK"** is the bonafide work of **"ABDUR RAZAQ A [211420104003], KARTHIKEYAN P[211420104124], KIRUBHANIDHI M [211420104133]"** who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with the date**
DR.L. JABASHEELA M.E. Ph.D.,	**Dr.S.HARIHARAN,M.E,M.TECH,Ph.D.,**
PROFESSOR	**ASSOCIATE PROFESSOR**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Panimalar Engineering College,	Panimalar Engineering College,
Chennai - 123	Chennai - 123

Certified that the above candidate(s) was examined in the End Semester Project Viva-Voce Examination held on ....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We ABDUR RAZAQ A [211420104003] , KARTHIKEYAN P [211420104124] , KIRUBHANIDHI M [211420104133] hereby declare that this project report titled **"BUILDING A DYSLEXIA DETECTION WEB APP WITH BOOSTING AND FLASK" , under the guidance of Dr.S.HARIHARAN,M.E,M.TECH,Ph.D is the original work done by us and we havenot plagiarized or submitted to any other degree in any university by us.**

ABDUR RAZAQ A [211420104003]
KARTHIKEYAN P [211420104124]
KIRUBHANIDHI M [211420104133]

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI M.A. Ph.D**. for his fervent encouragement. His inspirationalsupport proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We wish to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR M.E. Ph.D. an Dr. SARANYASREE SAKTHIKUMAR B.E. M.B.A. Ph.D.** for graciously affording us the essential resources and facilities for undertaking this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI M.E. Ph.D.** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA M.E. Ph.D.** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of the project.

We would like to thank the **Project Coordinator Dr. G. SENTHIL KUMAR, M.C.A., M.Phil., M.E., M.B.A., Ph.D.,** and **Project Guide Dr. S.HARIHARAN, M.E, M.TECH, Ph.D.,** all the facultymembers of the Department of CSE for their unwavering support for the successful completion of the project.

ABDUR RAZAQ A [211420104003]
KARTHIKEYAN P [211420104124]
KIRUBHANIDHI M [211420104133]

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

CIN: U80902TN2021PTC141464

www.pantechelearning.com

20/03/2024

COMPLETION CERTIFICATE

This is to acknowledge that student from **“PANIMALAR ENGINEERING COLLEGE”** has completed **Project** on the **title** of **“BUILDING A DYSLEXIA DETECTION WEB APP WITH BOOSTING AND FLASK”** at our concern from **DEC 2023** to **MAR 2024.**

1. **ABDUR RAZAQ A**
2. **KARTHIKEYAN P**
3. **KIRUBHANIDHI M**

For Pantech e learning.,

[illegible]

PANTECH E LEARNING ★ CHENNAI-17 ★

Authorized Signatory

Pantech eLearning Pvt Ltd.,
II Floor, Kotta Srinivasiah Charities Building,
Thanjavur Street, Near Duraisamy Subway, T.Nagar, Chennai – 600017
Phone: 91 44 42606470 | hr@pantechmail.com

ABSTRACT

This guide presents a step-by-step approach to developing a Dyslexia Detection Web App utilizing Boosting algorithms and Flask, offering a comprehensive resource for both novice and experienced developers.

The abstract encapsulates a dual focus: leveraging advanced Boosting techniques for accurate dyslexia identification and harnessing Flask's versatility for seamless web app deployment.

By delving into the intricacies of dyslexia detection and providing practical insights into algorithm implementation, this guide seeks to empower readers to create a sophisticated tool for aiding individuals with dyslexia. The abstract highlights the guide's commitment to fostering inclusivity through technology, promoting accessibility, and advancing the field of assistive applications.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

CHAPTER 1 INTRODUCTION

This project focuses on leveraging a boosting algorithm for the development of a dyslexia detection model. Dyslexia, a widespread learning disorder, can benefit from machine learning solutions that employ boosting techniques.

The project aims to create an effective model capable of accurately identifying dyslexia based on relevant features. The subsequent integration of this model into a Flask web application enhances accessibility, enabling users to receive real-time predictions and potentially contributing to early dyslexia identification.

1.1DOMAIN OVERVIEW

Machine Learning

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

Machine Learning vs. Traditional Programming:

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

1.2 APPLICATION OF MACHINE LEARNING

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry.
- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

1.3 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term mayalso be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed tothe natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions thathumans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the abilityto move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence.

AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous withAI, but a few, including Python, R, and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life-like exchanges with people, or an image recognition tool can learn toidentify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

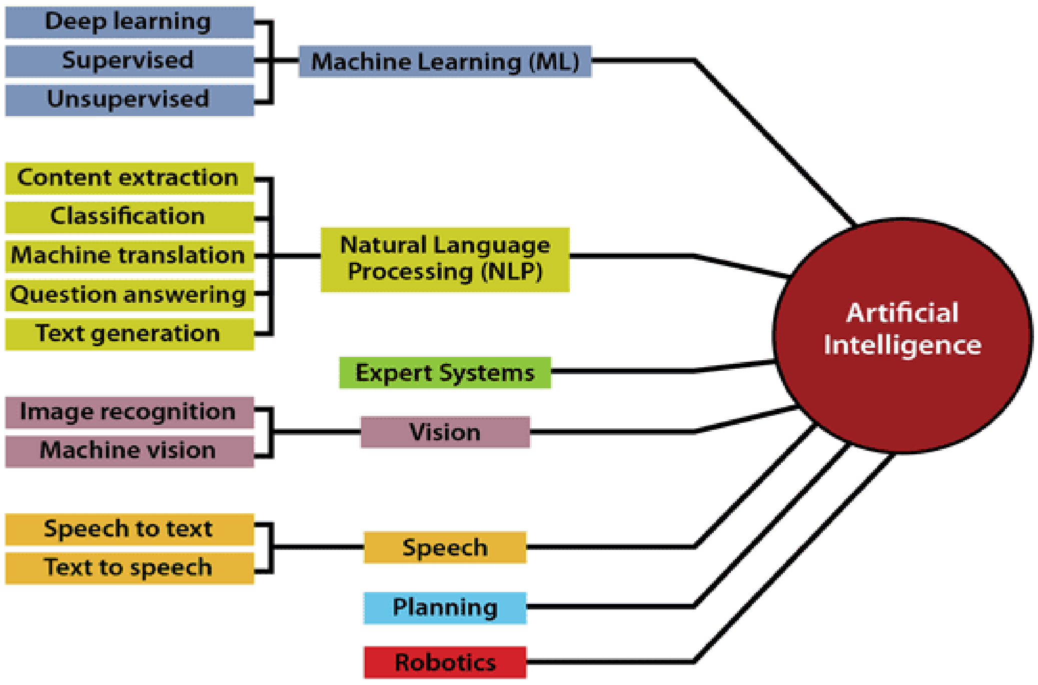


Fig 1.1

CHAPTER 2
LIRTERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1REVIEW OF LITERATURE SURVEY:

1.Title: Advance Machine Learning Methods for Dyslexia Biomarker Author: OPEYEMI LATEEF USMAN, RAVIE CHANDREN MUNIYANDI, MAZLY FARINA MOHAMAD,KHAIRUDDIN OMAR

Year: 2021

Dyslexia is a neurological disorder that is characterized by imprecise comprehension of words and generally poor reading performance. It affects a significant population of school-age children, with more occurrences in males, thus, putting them at risk of poor academic performance and low self-esteem for a lifetime. The long-term hope is to have a dyslexia diagnostic method that is informed by neural-biomarkers. In this regard, large numbers of machine learning methods and, more recently, deep learning methods have been implemented across various types of dataset with the above-chance classification accuracy. However, attainment of clinical acceptability of these state-of-the-art methods is bedeviled by certain challenges including lack of biologically-interpretable biomarkers, privacy of dataset and classifiers, hyper-parameter selection/optimization, and overfitting problem among others. This review paper critically analyzes recent machine learning methods for detecting dyslexia and its biomarkers and discusses challenges that require proper attentions from the users of deep learning methods in order to enable them to attain clinically relevance and acceptable level. The review is conducted within the premise of implementation and experimental outcomes for each of the 22 selected articles using the Preferred Reporting Items for Systematic review and Meta-Analyses (PRISMA) protocol, with a view to outlining some critical challenges for achieving high accuracy and reliability of the state-of-the-art machine.

2.Title: **Cosmic Sounds: A Game to Support Phonological Awareness Skills for Children With Dyslexia**

Author: Attracta brennan,Tara McDonagh,mary Dempsey,John Mcavoy

Year: 2022

Studies show that reduced literacy skills can negatively influence a child's self-esteem and future career opportunities. Literacy is significantly affected when problems exist in understanding the phonological component or sound structure of language, i.e., phonological awareness. Children with dyslexia in particular experience difficulties in spelling and reading accuracy due to a deficit in this phonological component of language. To support children with dyslexia and reduced literacy skills, intervention programs that focus on phonological awareness elements are recommended. Studies show that game-based learning interventions can enhance learning for children with dyslexia. The purpose of this pilot study was to partner with children with dyslexia aged between 9 and 12 years, to develop a game toolkit called Cosmic Sounds, and to support the teaching of phonological awareness skills. The content for the Cosmic Sounds games was informed by a pedagogical expert in dyslexia. This pilot study addressed the following: "Can a toolkit of games, codesigned by children with dyslexia improve the teaching of phonological awareness skills?" Our findings showed that by including children and their teacher as part of the design team, they were more invested in using the games for learning. Furthermore, when children with dyslexia played Cosmic Sounds, there was a positive impact on their phonological awareness skills progress while their engagement in learning also increased.

3.Title: Eye-Tracking Image Encoding: Autoencoders for the Crossing of Language Boundaries in Developmental Dyslexia Detection

Author: Ivan A.Vajs,Goran S.Kvascev,Tamara M.Papic,Milica M.Jankovic

Year : 2023

The negative influence of developmental dyslexia on academic performance is a well-documented and researched topic. Although the research focused on developmental dyslexia detection and evaluation is plentiful, the study designs vary to a great degree, making the exchange of obtained knowledge often difficult. This paper focuses on bridging the gap between different study designs by developing a machine learning based pipeline that was evaluated on two completely different eye-tracking datasets (training on one, testing on the other, and vice versa). One dataset included 30 subjects who read text written in Serbian on different color configurations and were tracked with a remote eye-tracker. The second dataset included 185 subjects who read text written in Swedish and recorded eye-tracking data using a goggle-based system. The data from both datasets were converted to grayscale images, using various time window configurations to parse the signals, and to plot the data in a 2D plane. The train images were used to train an Autoencoder neural network, and the images' reconstruction error was used to create features that describe each instance of both the training and test sets. The train feature set was used to train various machine learning algorithms, which were then evaluated on the testing feature dataset. A classification accuracy of 85.6% was obtained when testing on Serbian readers' data and 82.9% when testing on Swedish readers. The proposed pipeline was shown to be transferable between the datasets, despite many differences in the experiment design, showing potential in combining various eye-tracking dyslexia studies.

4.Title: Model for Profiling Users With Disabilities on e-Learning Platforms

Author: SANDRA SANCHEZ-GORDON, CARMEN AGUILAR-MAYANQUER, TANIA CALLE-JIMENEZ

Year : 2021

There are millions of people worldwide using e-Learning platforms. However, people with disabilities still have problems using these platforms due to a lack of accessibility. In the present study, a model for the generation of the profiles of users with disabilities is proposed. This model enables e-Learning platforms to have information on users' accessibility needs as input for automated adaptation of its interfaces. This would enable equal access for all learners with and without disabilities. In this context, different studies related to the profiling of users on e-Learning platforms were identified and analyzed. The model presented considers the Web Content Accessibility Guidelines (WCAG) that can be implemented in the interfaces of the e-Learning platforms and the metadata that represent the accessibility needs of users based on Schema.org. The researchers used Unified Modelling Language diagrams to design the model and define a description of the interaction between users, user interfaces, WCAG, Schema.org., and the eXtensible Markup Language (XML) profile generated. The testing of the prototype was carried out using WAVE and ARC Toolkit. The validation with the support of forty-four users was conducted using the System Usability Scale (SUS). The most outstanding results of this study are the identification of WCAG success criteria that are automatically implementable, the inclusion of two special categories for combined accessibility needs related to the elderly and linguistics, and the automatic generation of the profile as an XML file containing the metadata needed to enable the adaptation of e-Learning platform interfaces

5.Title: TestGraphia, a Software System for the Early Diagnosis of Dysgraphia

Author: Douglas Teoh

Year: 2014

Dysgraphia, which is known as a writing disorder, is a specific disorder of writing regarding the reproduction of alphabetical and numerical signs. Dysgraphia may be related to dyspraxia, which is secondary to incomplete lateralization and characterized by a difficulty to reproduce alphabetical and numerical signs. Since the causes of dysgraphia are unknown, the rapid detection of symptoms is very important. In academic and clinical uses, the most common tool for detecting dysgraphia is an evaluation of the quality of writing on paper sheets. A writing analysis is based on rules for scoring the writing quality. In this paper, we discuss TestGraphia, which is a software system that can support doctors in making diagnoses and monitoring patients with dysgraphia in an objective manner. The system is based on known document analysis algorithms and modified or specially designed algorithms. Based on this software, a forms analysis requires considerably less time than that needed by traditional methods, enabling large screening activities and reducing time and cost. Potential dynamic changes in dysgraphia screening can be assessed by monitoring the quality of writing in a non-invasive way with reduced costs, both in the laboratory and the patient's home, and the appropriate frequency. In the system that we will describe, the mean time to execute a diagnosis is nearly ten times faster with trustworthy results.

2.1 SYSTEM STUDY:

AIM:

The project scope encompasses the development of a Dyslexia Detection Web App using Boosting and Flask, aiming to create an accessible and accurate tool for identifying dyslexia

Objectives:

The objective of this comprehensive guide is to empower developers and enthusiasts to create a Dyslexia Detection Web App using Boosting techniques, offering a step-by-step roadmap for implementation. It aims to provide a hands-on experience in building a robust and effective tool for identifying dyslexia. Readers will gain insights into the intricacies of dyslexia detection, learn the fundamentals of Boosting algorithms, and acquire practical skills in using the algorithms and module packages.

Scope:

The project scope encompasses the development of a Dyslexia Detection Web App using Boosting and Flask, aiming to create an accessible and accurate tool for identifying dyslexia. The scope includes implementing advanced Boosting algorithms, integrating Flask for web deployment, and providing a comprehensive guide for developers. The project seeks to contribute to assistive technology, fostering inclusivity and support for individuals with dyslexia by offering a practical and insightful resource for building a robust dyslexia detection solution

2.2 FEASIBILITY STUDY:

Data Collection:

Gather a dataset of medical records that includes information about patients, their medical history, lifestyle factors, and whether they have had a stroke. This dataset should be large and diverse to train a robust machine learning model

Data Preprocessing:

Clean and preprocess the data. This involves handling missing values, data normalization, and converting categorical data into numerical form. Ensure data privacy and compliance with relevant regulations when dealing with sensitive.

Feature Selection and Engineering:

Identify relevant features (attributes) that can influence stroke prediction. These may include age, gender, hypertension, heart disease, smoking status, BMI, and more.

Model Training and Evaluation:

Train the selected machine learning model on the training data. The model learns to predict whether a patient is at risk of having a stroke based on the provided features.

Evaluate the model's performance on the testing set using appropriate metrics, such as accuracy, precision, recall, F1-score, and ROC-AUC, to assess its predictive accuracy and reliability.

CHAPTER 3
THEORETICAL BACKGROUND

CHAPTER 3

THEORETICALBACKGROUND

3.1 Implementation Environment:

`conda build`

Anaconda Package versions are been managed by the package managements system "Conda". The Anaconda distribution is used by over 12 million users and includesmore than 1400 popular data science packages suitable for Windows, Linux, and MacOS.So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packagescan be individually installed from the Anaconda repository with the condo install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of condo packages and in most cases they can work together. Custom packages can be made using the command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7.However, you can create new environments that include any version of Python packaged with conda.

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

- Add support for Offline Mode for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

PYTHON

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications.

It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Features of Python

A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

- Free and open source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

● Portability

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

● Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

● A high-level, interpreted language

Unlike C/C++, you don’t have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower level operations.

● Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use MySQLdb library using import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

● Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating object

Python

History and Versions:

Python is predominantly a dynamic typed programming language which was initiated by Guido van Rossum in the year 1989. The major design philosophy that was given more importance was the readability of the code and expressing an idea in fewer lines of code rather than the verbose way of expressing things as in C++ and Java [K-8][K-9]. The other design philosophy that was worth mentioning was that, there should be always a single way and a single obvious way to express a given task which is contradictory to other languages such as C++, Perl etc. [K-10]. Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The initial versions of Python were heavily inspired from lisp (for functional programming constructs). Python had heavily borrowed the module system, exception model and also keyword arguments from Modula-3 language [K-10]. Pythons' developers strive not to entertain premature optimization, even though it might increase the performance by a few basis points [K-9]. During its design, the creators had conceptualized the language as being a very extensible language, and hence they had designed the language to have a small core library which was extended by a huge standard library [K-7]. Thus as a result, python is used as a scripting language as it can be easily embedded into any application, though it can be used to develop a full-fledged application. The reference implementation of python is CPython. There are also other implementations like Jython, Iron Python which can use python syntax as well as can use any class of Java (Jython) or .Net class (Iron Python). Versions: Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.6 and the latest in 3.x series is 3.4.0. 1.5.2 Paradigms: Python supports multi-paradigms such as: Object-Oriented, Imperative, Functional, Procedural, and Reflective. In Object-Oriented Paradigm, Python supports most of the OOPs concepts such as Inheritance (It also has support for Multiple Inheritance), Polymorphism but its lack of support for encapsulation is a blatant omission as Python doesn’t have private, protected members: all class members are public [K-11]. Earlier Python 2.6 versions didn’t support some OOP’s concepts such as Abstraction through Interfaces and Abstract Classes [K-19]. It also supports Concurrent paradigm, but with Python we will not be able to make truly multitasking applications as the inbuilt threading API is limited by GIL (Global Interpreter Lock) and hence applications that use the threading API cannot run on multi-core parallelly [K-12].The only remedy is that, the user has to either use the multi-processing module which would fork processes or use Interpreters that haven’t implemented GIL such as Jython or Iron Python [K-12]. 1.5.3 Compilation, Execution and Memory Management: Compilation, Execution and Memory Management: 21 A Comparative Studies of Programming Languages (Comparative Studies of Six Programming Language) Just like the other Managed Languages, Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The reference implementation (i.e. CPython) doesn’t come with a JIT compiler because of which the execution speed is slow compared to native programming languages [K-17]. We can use PyPy interpreter as it includes a JIT compiler rather than using the Python interpreter that comes by default with the python language, if speed of execution is one of the important factors [K-18]. The Python Runtime Environment also takes care of all the allocation and deallocation of memory through the Garbage Collector. When a new object is created, the GC allocates the necessary memory, and once the object goes out of its scope, the GC doesn’t release memory immediately but instead it becomes eligible for Garbage Collection, which would eventually release the memory. Typing Strategies: Python is a strongly dynamic typed language. Python 3 also supports optional static typing [K-20]. There are a few advantages in using a dynamic typed language, the most prominent one would be that the code is more readable as there is less code (in other words has less boiler-plate code). But the main disadvantage in having python as a dynamic programming language is that there would be no way to guarantee that a particular piece of code would run successfully for all the different data-types scenarios simply because it had run successfully with one type. Basically, we don’t have any means to find out an error in the code, till the code has started running. 1.5.4 Strengths and Weaknesses and Application Areas: Python is predominantly used as a scripting language used in developing standalone applications that are being developed with Static-Typed languages, because of the flexibility it provides due to its dynamic typed nature. Python favours rapid application development, which qualifies it to be used for prototyping. To a certain extent, Python is also used in developing websites. Due to its dynamic typing and of the presence of a Virtual Machine, there is a considerable overhead which translates to way less performance when we compare with native programming languages [K-13]. And hence it is not suited.

3.2 Existing System:

Traditionally, dyslexia detection involves educational assessments and psychological evaluations. While some researchers explore machine learning for this purpose, there isn't a standardized system yet. To find the latest information, it's advised to check recent research and open-source projects in relevant fields. Ethical considerations and stakeholder involvement are essential when developing solutions for learning disorders

Demerits:

- Dependency on Human Expertise
- Limited Accessibility
- Costly Interventions
- Lack of Personalization

3.3 Proposed System:

The proposed system introduces a novel approach to dyslexia detection by incorporating a boosting algorithm, specifically designed to enhance accuracy and efficiency in comparison to conventional methods. This system leverages machine learning, utilizing the boosting algorithm's capabilities to automate the identification process. Real-time predictions are facilitated through a Flask web application, addressing subjectivity concerns and offering continuous monitoring. The system is designed to adapt to emerging technologies, ensuring a data-driven, accessible, and effective means of early dyslexia identification.

Advantage:

- Precision and Personalization.
- Efficiency and Time-Saving.
- Continuous Monitoring
- Real-Time Predictions
- Efficient Pattern Recognition

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

4.1 System Architecture

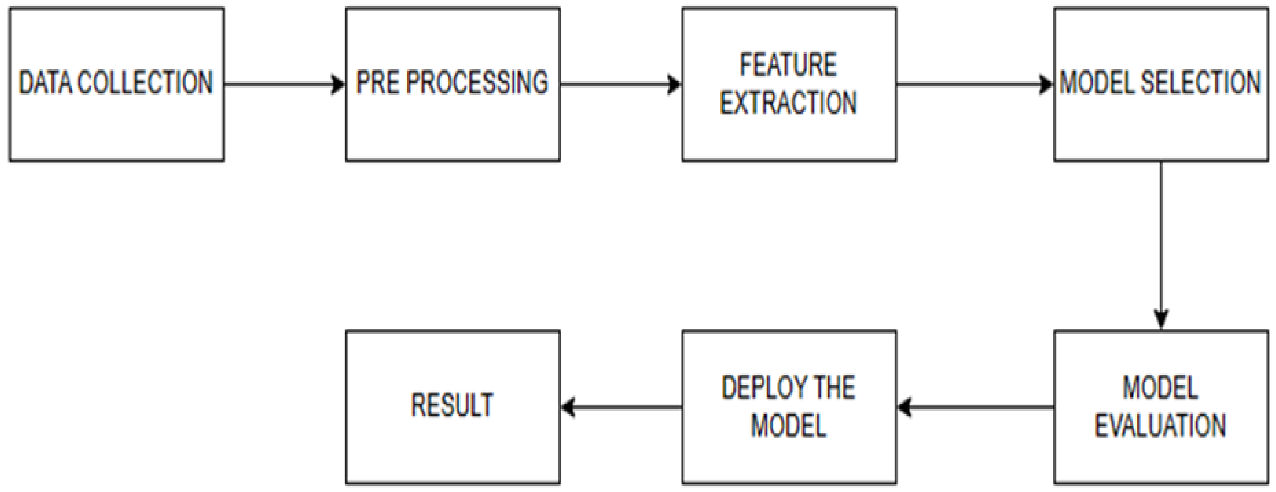


Fig 4.1

`Data Collection:

Gather a dataset of medical records that includes information about patients, their medical history, lifestyle factors, and whether they have had a stroke. This dataset should be large and diverse to train a robust machine learning model.

Data Preprocessing:

Clean and preprocess the data. This involves handling missing values, data normalization, and converting categorical data into numerical form. Ensure data privacy and compliance with relevant regulations when dealing with sensitive medical data.

Feature Selection and Engineering:

Identify relevant features (attributes) that can influence stroke prediction. These may include age, gender, hypertension, heart disease, smoking status, BMI, and more. Perform feature engineering to create new features or transform existing ones if needed.

Data Splitting:

Split the dataset into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate its performance.

Machine Learning Model Selection:

Choose an appropriate machine learning algorithm for classification. Common choices include logistic regression, decision trees, random forests, support vector machines, and neural networks.

Evaluate and compare different algorithms to select the best-performing one And We Apply AdaBoost classifier and gradient boosting classifier.

Model Training:

Train the selected machine learning model on the training data. The model learns topredict whether a patient is at risk of having a stroke based on the provided features

Model Evaluation:

Evaluate the model's performance on the testing set using appropriate metrics, such as accuracy, precision, recall, F1-score, and ROC-AUC, to assess its predictive accuracy and reliability.

4.1 Data flow diagram:

Level 0:

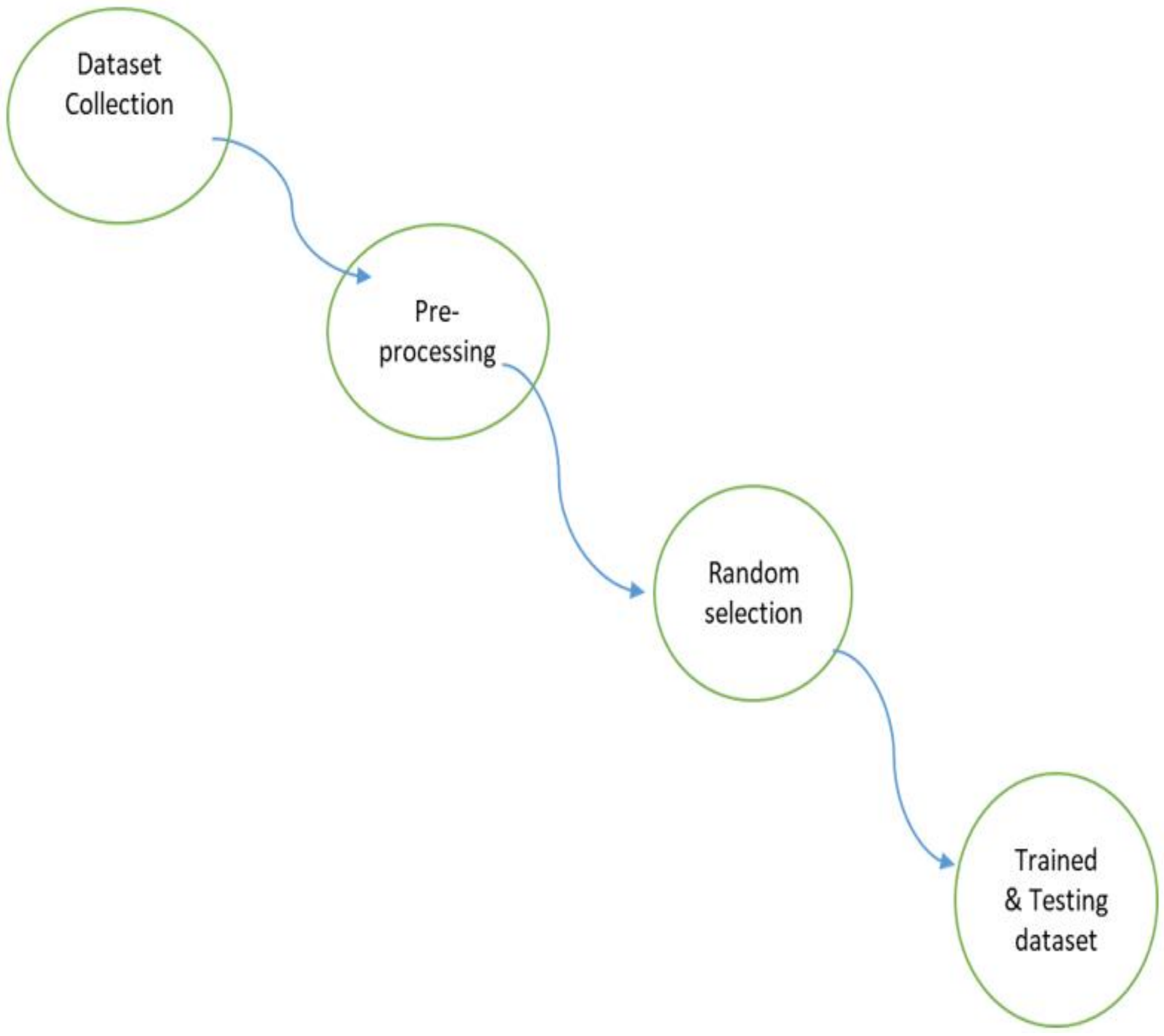


Fig 4.2

Level 1:

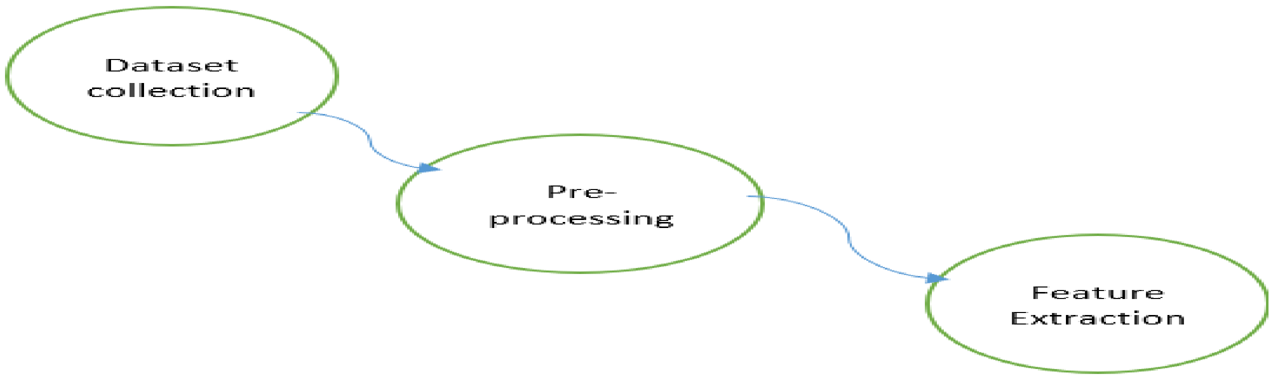


Fig 4.3

Level 2:

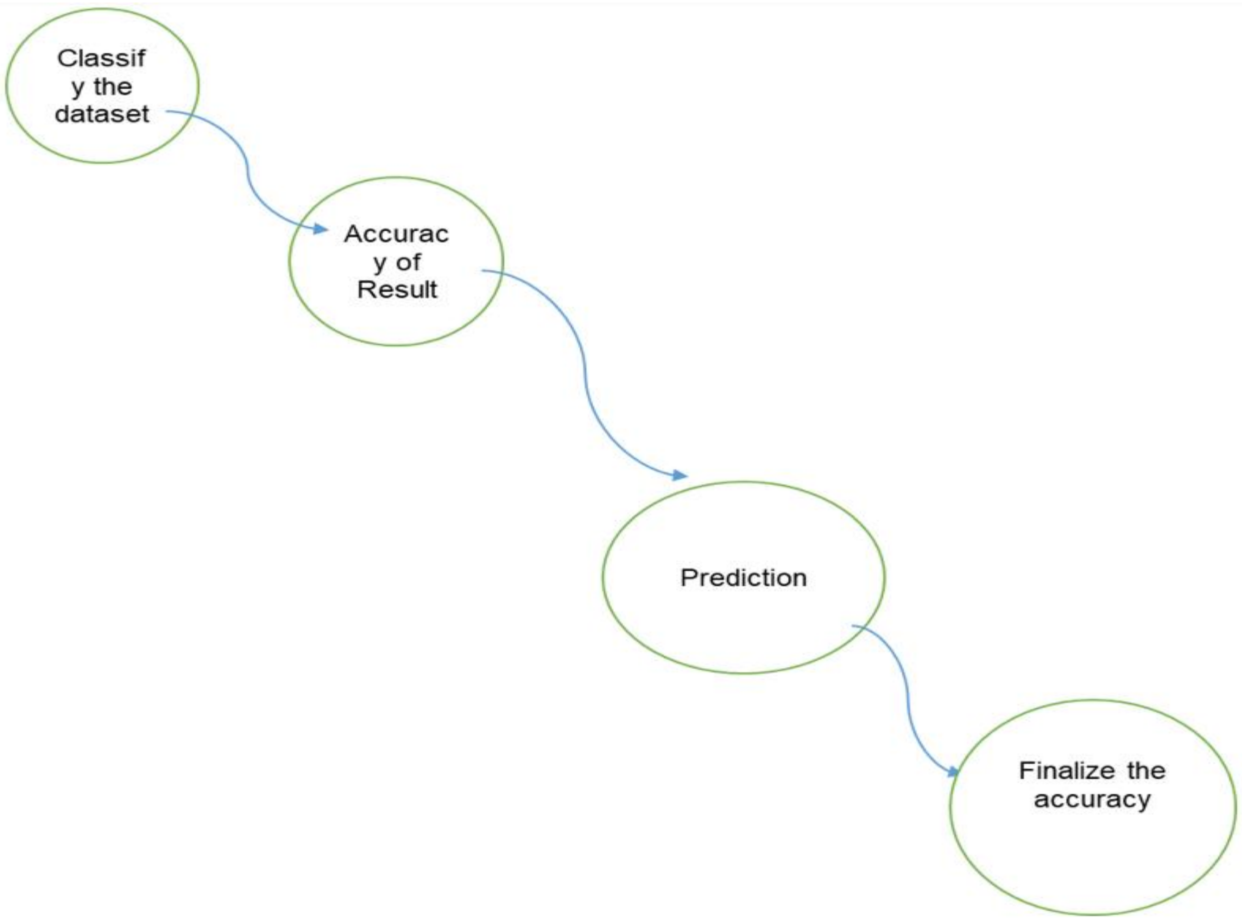


Fig 4.4

4.2 Use Case Diagram

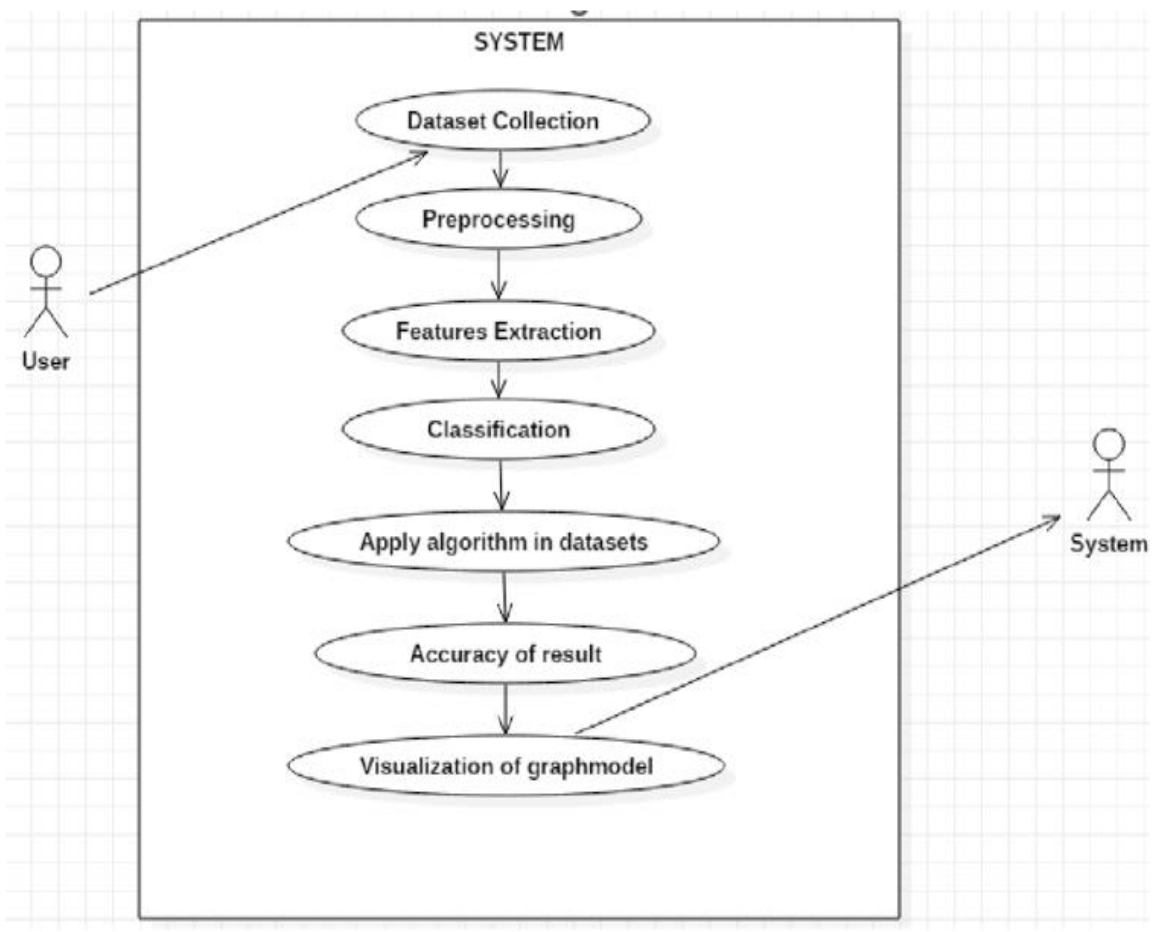


Fig 4.5

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

4.3 **Class Diagram**:

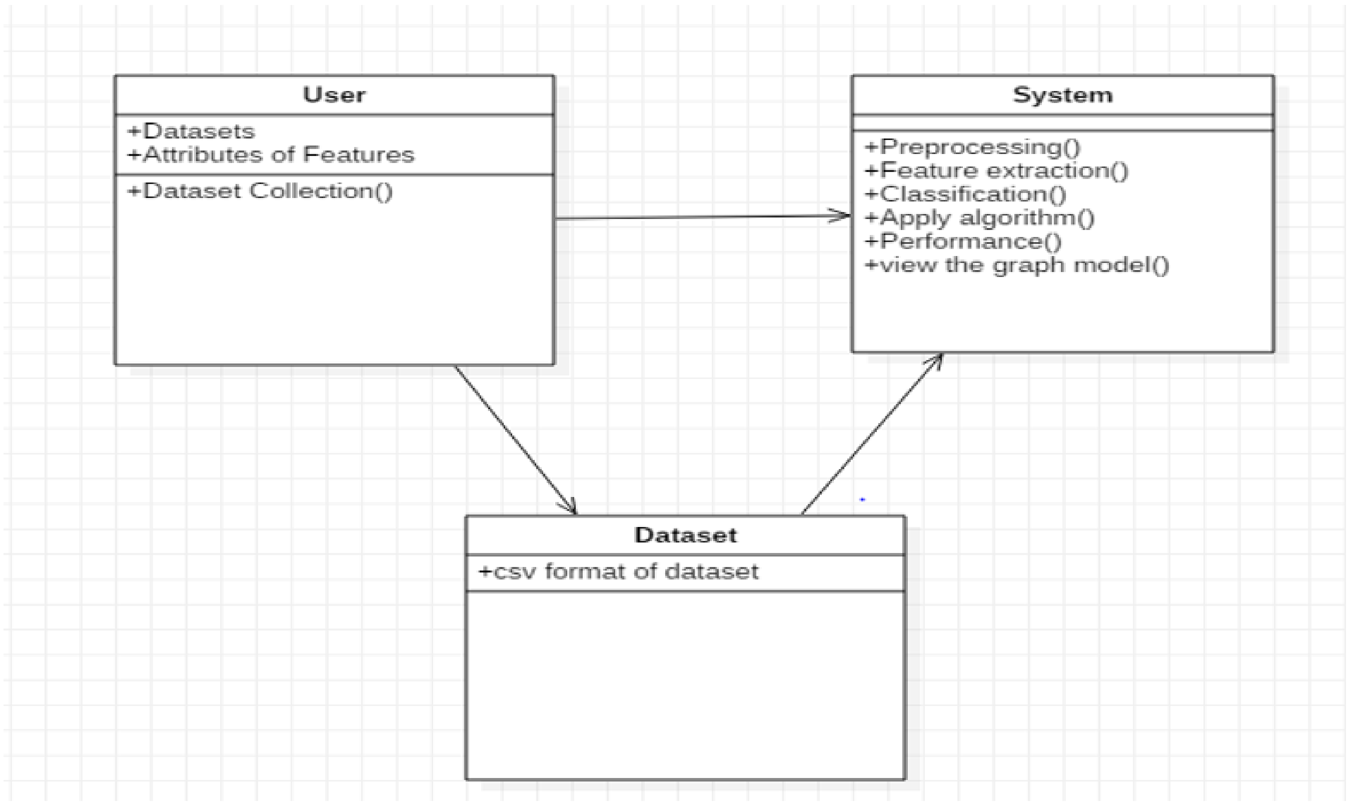


Fig 4.6

Class diagram is basically a graphical representation of the static view of the systemand represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to

describe the aspect of the system. Each element and their relationships should be identifiedin advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever requiredto describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder.

4.4Activity Diagram:

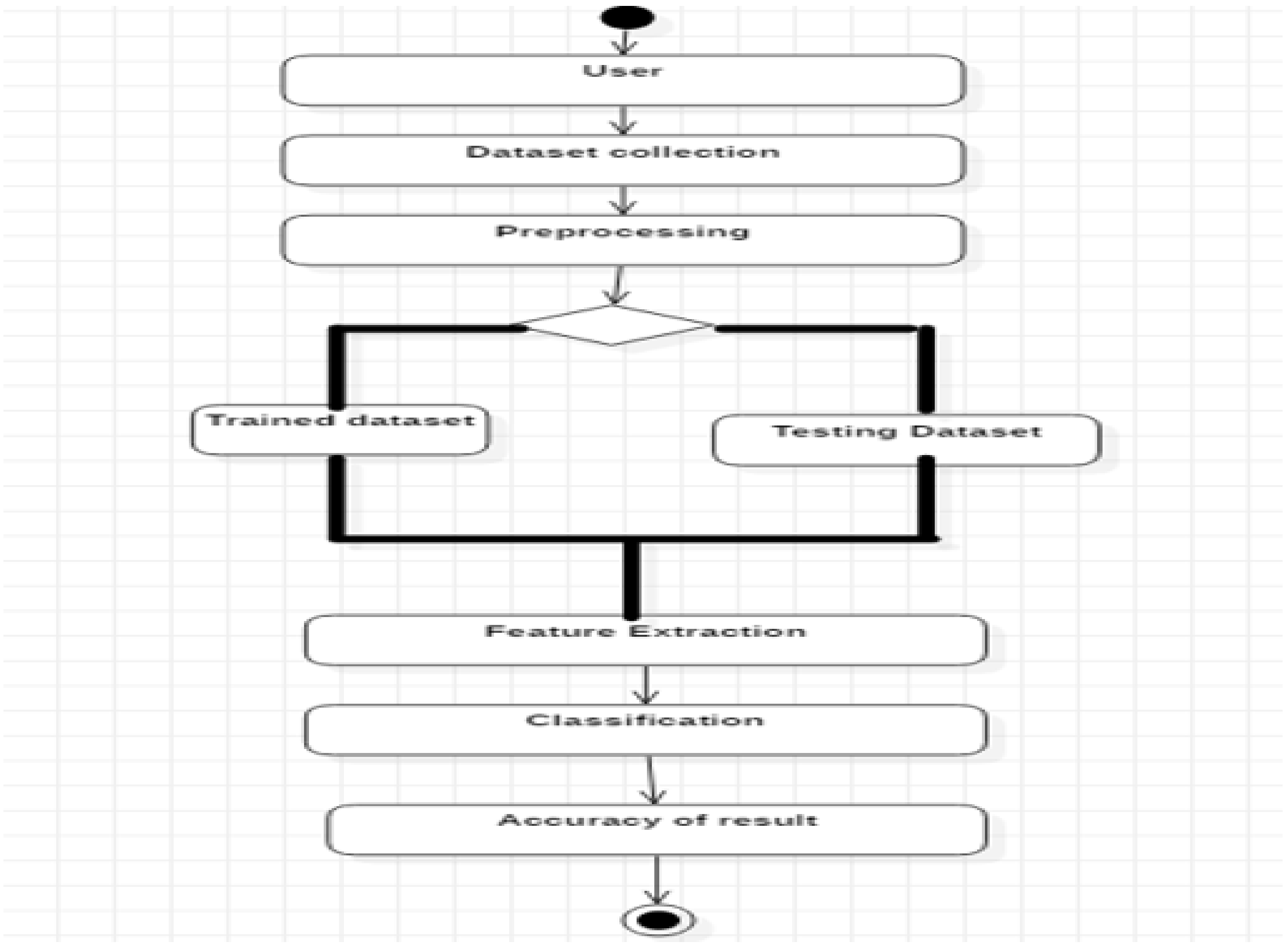


Fig 4.7

Activity is a particular operation of the system. Activity diagrams are not only usedfor visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missingthing in activity diagram is the message part. It does not show any message flow from oneactivity to another. Activity diagram is some time considered as the flow chart. Althoughthe diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

CHAPTER 5
SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES:

DATA GATHERING

This paper's information assortment comprises of various records. The determination of the subset of all open information that you will be working with is the focal point of this stage. Preferably, ML challenges start with a lot of information (models or perceptions) for which you definitely know the ideal arrangement. Marked information will be data for which you are as of now mindful of the ideal result.

PRE-PROCESSING OF DATA

Format, clean, and sample from your chosen data to organise it.

There are three typical steps in data pre-processing:

1.Designing

2.Information cleaning

3.Inspecting

Designing: It's conceivable that the information you've picked isn't in a structure that you can use to work with it. The information might be in an exclusive record configuration and you would like it in a social data set or text document, or the information might be in a social data set and you would like it in a level document.

Information cleaning: is the most common way of eliminating or supplanting missing information. There can be information examples that are inadequate and come up short on data you assume you really want to resolve the issue. These events could should be eliminated..Moreover, a portion of the traits might contain delicate data, and it very well might be important to antonymize or totally eliminate these properties from the information.

Inspecting: You might approach significantly more painstakingly picked information than you want. Calculations might take significantly longer to perform on greater measures of information, and their computational and memory prerequisites may likewise increment. Prior to considering the whole datasets, you can take a more modest delegate test of the picked information that might be fundamentally quicker for investigating and creating thoughts.

FEATURE EXTRACTION

The following stage is to A course of quality decrease is include extraction. Highlight extraction really modifies the traits instead of element choice, which positions the ongoing ascribes as indicated by their prescient pertinence. The first ascribes are straightly joined to create the changed traits, or elements. Finally, the Classifier calculation is utilized to prepare our models. We utilize the Python Normal Language Tool stash's classify module.

We utilize the gained marked dataset. The models will be surveyed utilizing the excess marked information we have. Pre-handled information was ordered utilizing a couple of AI strategies. Irregular woodland classifiers were chosen. These calculations are generally utilized in positions including text grouping.

ASSESSMENT MODEL

Model The method involved with fostering a model incorporates assessment. Finding the model that best portrays our information and predicts how well the model will act in what's to come is useful. In information science, it isn't adequate to assess model execution utilizing the preparation information since this can rapidly prompt excessively hopeful and overfitted models. Wait and Cross-Approval are two procedures utilized in information science to evaluate models.

The two methodologies utilize a test set (concealed by the model) to survey model execution to forestall over fitting. In light of its normal, every classification model's presentation is assessed. The result will take on the structure that was envisioned. diagram portrayal of information that has been ordered.

Accuracy: The level of precise expectations for the test information is implied by precision. By partitioning the quantity of exact expectations by the complete number of forecasts, it very well might still up in the air.

5.2 Python Libraries &Packages:

Python provides a variety of GUI (Graphical User Interface) creation options. Tkinter is the most frequently used GUI method out of all of them. It is a typical Python connection point to the Python distribution's Tk GUI tool stack. The quickest and easiest way to create GUI apps is with Python and Tkinter. Tkinter makes creating a GUI an easy task.

A tkinter application can be created by:

I. Introducing the tkinter module
II. Create the basic window (holder)
III. Add a lot of devices to the main window.
IV. Apply the event Trigger to the devices.
V. Tkinter can be inserted into the Python code just like any other module.

Note that the module's name changes from "Tkinter" in Python 2.x to "tkinter"

When creating a Python application with a GUI, the client needs to keep in mind two main strategies that are used.

Tk(screenName = 'None', base = 'None', class = 'Tk', useTk = 1):

Tkinter provides the method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)' to create a main window.

You can alter the className to the desired one to change the window's name. The primary code used to create the application's main window is m=tkinter.Tk(), mainloop(), where m is the name of the basic window object.

When your application is ready to run, a technique called as mainloop() is used.

As long as the window is open, the application will run indefinitely throughout the mainloop() loop, which is used to anticipate events and respond to them.

Additionally, tkinter provides access to the mathematical configuration of the devices that can arrange the devices in the parent windows. Fundamentally, there are three main maths classes.

Pack() technique:Prior to inserting the parent device, it organises the devices into blocks.

matrix() technique:Before setting in the parent gadget, it networks the devices together (in a table-like form).

place() technique:It organises the devices by positioning them in clear positions decided upon by the software programmer.

You can include a variety of tools in your tkinter application.

Button: This gadget is used to add a button to your programme.

w=Button(master, option=value) is the general grammar, and ace is the boundary used to refer to the parent window.The arrangement of the Buttons can be changed using a variety of options. Choices can be passed as boundaries that are separated by commas. The list below includes a few of them.

To set the background tone when the button is under the cursor, use activebackground.

When a button is under the cursor, the activeforeground setting sets the forefront tone.

bg: to establish the standard foundational tone.

to call a capability is the order.

textual style: to change the button mark's text style.

image: to place an image on a button.

width: to modify the button's width.

level: to modify the button's level.

NumPy:

The free source Python package NumPy (Mathematical Python) is used in virtually every field of science and design. It serves as the foundation of the logical Python and PyData environments and is the standard practise for working with mathematical data in Python. Clients of NumPy range from novice programmers to seasoned analysts performing top-tier logical and cutting-edge inventive work. Numerous information scientific and logical Python bundles, including Pandas, SciPy, Matplotlib, scikit-learn, and scikit-picture, make extensive use of the NumPy Programming interface. Complex network and exhibit information structures are available in the NumPy library (you'll find out more information about this in later sections). It offers ways to successfully work on ndarray, a homogenous n-layered cluster object.

Numerous numerical operations can be played out on exhibitions using NumPy. It provides a vast library of high level numerical capabilities that are compatible with these displays and frameworks, as well as powerful information designs that are added to Python that assure accurate estimations with clusters and lattices.

The NumPy library's exhibit is its main information design. A matrix of numbers that serves as an exhibit comprises information on the raw data, how to locate a component, and how to interpret a component. It is made up of a lattice of parts that can be filed in various ways.

The components can be expressed as whole numbers, booleans, another exhibit, or analogous types of nonnegative numbers. The number of factors determines where the cluster is located. A tuple of numbers indicating the size of the exhibit along each axis makes up the cluster's state.

Pandas: The cluster dtype is hinted at.

A tuple of pandas, a Python library that provides rapid, adaptive, and expressive information structures designed to enable working with "social" or "marked" information simple and natural, can be filed into a cluster. It intends to serve as the primary major level building piece for conducting practical, accurate information analysis in Python.

Additionally, it aspires to become the most remarkable and flexible open source information examination/control tool obtainable in any language. It has already made significant progress in achieving this goal.

Primary Components

Size variability: portions can be inserted in and erased from DataFrame and higher layered objects Simple treatment of missing information (addressed as NaN, NA, or NaT) in drifting point information as well as non-drifting point information

Programmed and explicit information organisation: Items can be explicitly changed to a group of names, or the client can merely ignore the markings and let Series, DataFrame, and other similar entities subsequently alter the information for calculations.

Make it simple to convert battered, unexpectedly listed information in other Python and NumPy information structures into DataFrame objects. Strong, adaptable collection by usefulness to perform split-apply-join procedure on informational collections, for both aggregating and changing information.

Pip intall pandas

Matplotlib:

A remarkable Python representation library for 2D cluster plots is Matplotlib. Based on NumPy clusters, Matplotlib is a multi-stage information perception library designed to operate with the more comprehensive SciPy stack. In the year 2002, John Tracker delivered the speech. The ability to visually access vast amounts of information in delicious pictures is one of perception's biggest benefits. Several plots, including line, bar, scatter, histogram, and others, are included in Matplotlib. Matplotlib and the great majority of its conditions are available as wheel bundles for Windows, Linux, and macOS.

Run the supplemental command to launch the matplotlib bundle:

```
matplotlib - mpip introduction - python
```

Seaborn:

A matplotlib-based information perception library called Seaborn is tightly integrated with Python's pandas data structures. The main component of Seaborn is perception, which aids in information research and comprehension. To learn about Seaborn, one needs be knowledgeable in Numpy, Matplotlib, and Pandas.

Seaborn provides the following features:

Programming interface with a dataset to determine the relationships between variables.

Straight relapse plots are assessed and plotted using a programme.

For multi-plot frameworks, it supports important level debates.

Imagining the dispersion of univariate and bivariate data.

The procedure used to establish the Seaborn library is as follows:

```
sns import seaborn
```

Here are a handful of well-known plotting libraries to get an outline:

1.Matplotlib: accessible and full of opportunities

2. Pandas Perception is a user-friendly interface built on Matplotlib.

3. Seaborn: outstanding default styles, important level connection point

4.ggplot: makes use of the Syntax of Illustrations in R's ggplot2

5.Plotly: able to create clever plots

In this post, we'll learn how to use Matplotlib, Pandas Perception, and Seaborn to create basic plots and how to make use of a few specific features from each library. Instead of attempting to interpret the schematics, this essay will focus just on the punctuation.

Line Outline

In Matplotlib we can make a line outline by calling the plot strategy. We can likewise plot different sections in a single diagram, by circling through the segments we need, and plotting every section on a similar hub.

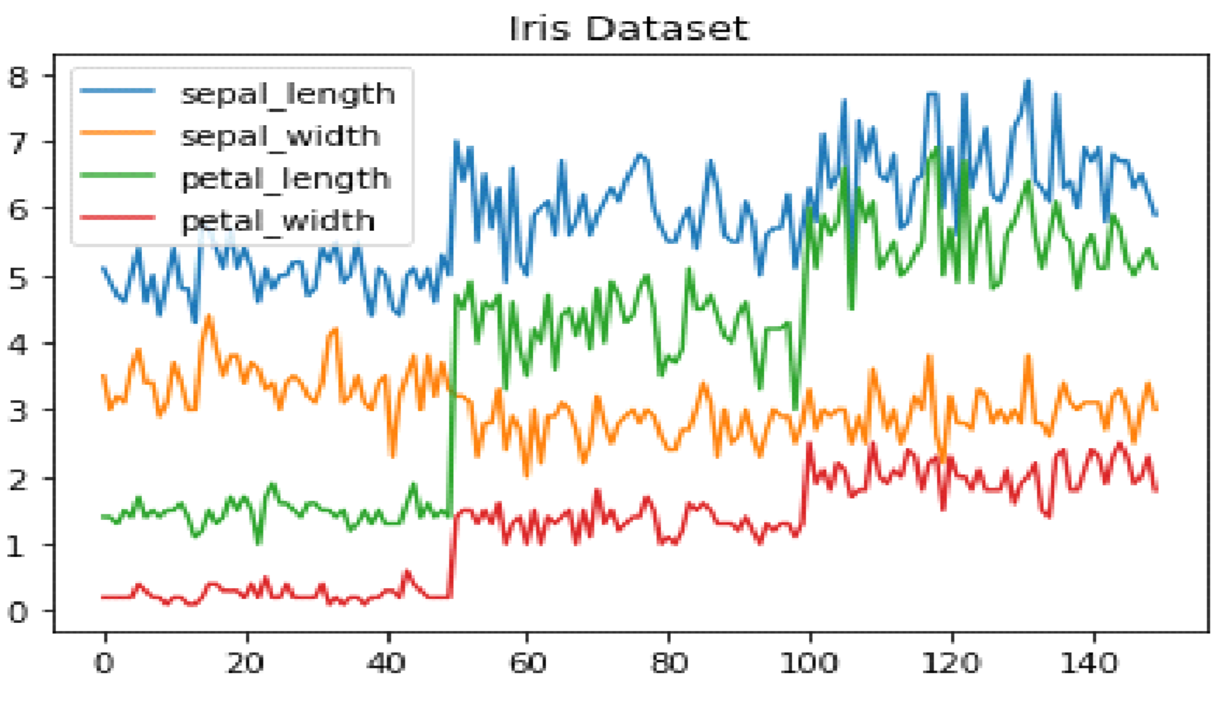


Fig 5.1

Line Chart

Histogram:

In Matplotlib we can make a Histogram utilizing the hist technique. Assuming that we pass it downright information like the focuses segment from the wine-survey dataset it will naturally ascertain how frequently each class happens.

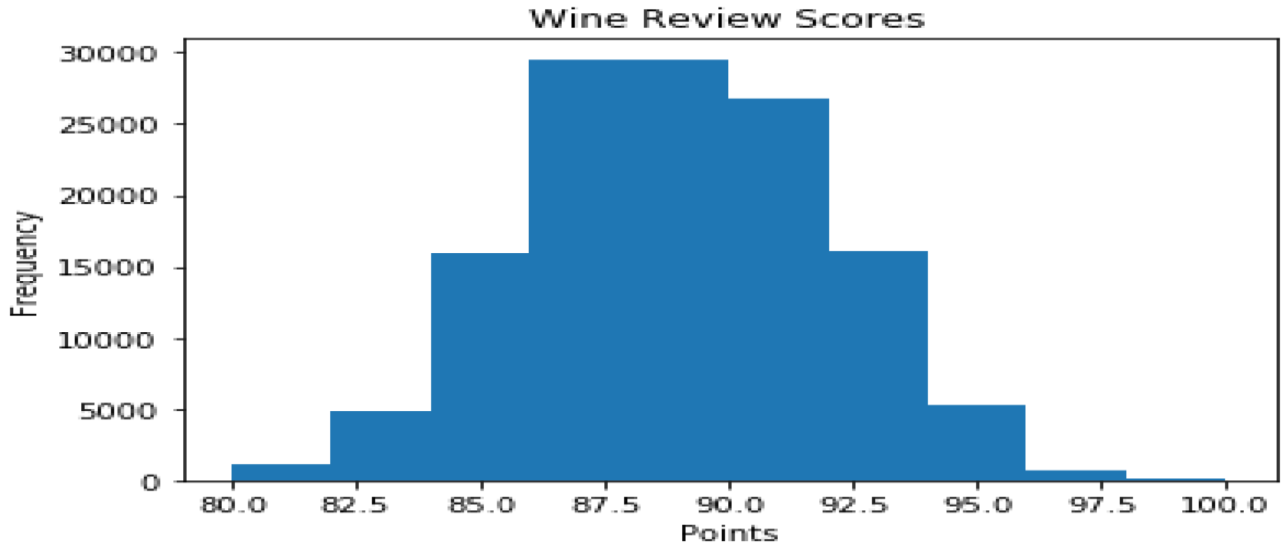


Fig 5.2

Bar Diagram:

A bar-diagram can be made utilizing the bar strategy. The bar-outline isn't consequently computing the recurrence of a classification so we will utilize pandas value_counts capability to do this. The bar-outline is valuable for downright information that has relatively little various classifications (under 30) in light of the fact that else it can get very chaotic.

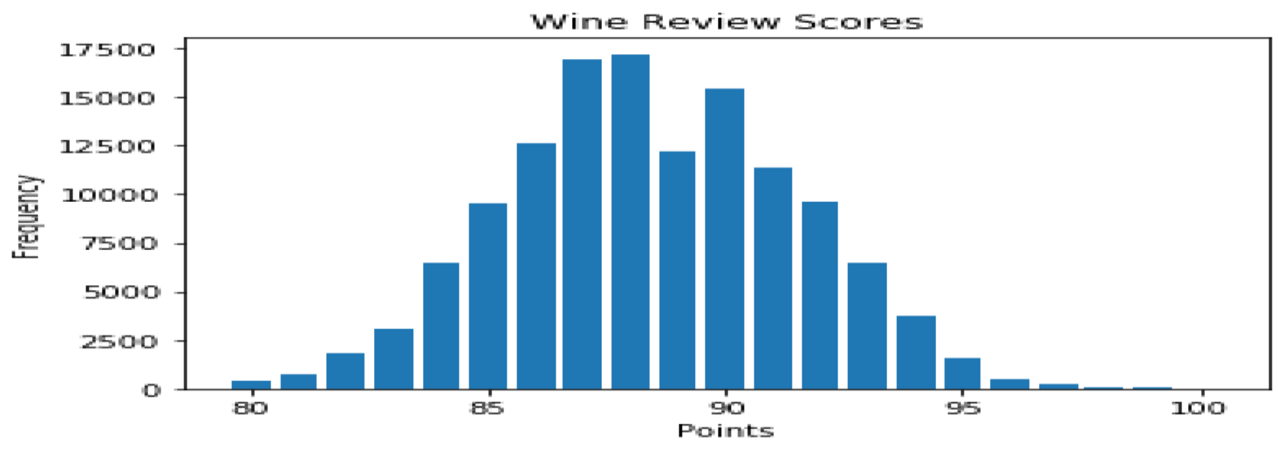


Fig 5.3

Pandas Visualization:

Pandas is an open source superior execution, simple to-utilize library giving information structures, for example, dataframes, and information examination devices like the representation instruments we will use in this article.

Pandas Perception makes it truly simple to make plots out of a pandas dataframe and series. It likewise has a more elevated level Programming interface than Matplotlib and hence we want less code for similar outcomes.

●Pandas can be introduced utilizing either pip or conda.

●pip introduce pandas

●conda introduce pandas

Heatmap:

A Heatmap is a graphical portrayal of information where the singular qualities contained in a lattice are addressed as varieties. Heatmaps are ideally suited for investigating the connection of elements in a dataset.

To get the relationship of the highlights inside a dataset we can call <dataset>.corr() , which is a Pandas dataframe strategy. This will give utilize the connection grid.

We can now utilize either Matplotlib or Seaborn to make the heatmap.

Matplotlib:

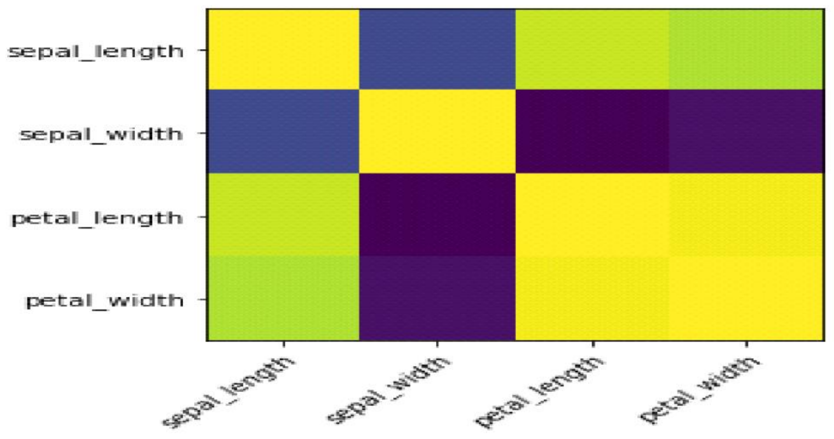


Fig 5.4

Information perception is the discipline of attempting to comprehend information by setting it in a visual setting, so that examples, patterns and relationships that could not in any case be recognized can be uncovered.

Python offers various incredible diagramming libraries that come loaded with bunches of various highlights. In this article we checked out at Matplotlib, Pandas representation and Seaborn.

Scikit-learn:

Scikit-learn is an open source information examination library, and the best quality level for AI (ML) in the Python biological system

Ideas And Highlights:

Characterization: distinguishing and ordering information in light of examples

Relapse: anticipating or projecting information values in light of the typical mean of existing and arranged information.

Bunching: programmed gathering of comparable information into datasets.

Interoperability with NumPy, pandas, and matplotlib libraries.

ML is an innovation that empowers PCs to gain from input information and to fabricate/train a prescient model without unequivocal programming. ML is a subset of Man-made consciousness (simulated intelligence).

Whether you are only searching for a prologue to ML, need to make ready quick, or are searching for the most recent ML research instrument, you will find that scikit-learn is both factual and simple to learn/use. As a significant level library, it allows you to characterize a prescient information model in only a couple of lines of code, and afterward utilize that model to accommodate your information. It's adaptable and coordinates well with other Python libraries, for example, matplotlib for plotting, numpy for cluster vectorization, and pandas for information outlines.

5.3 Algorithms:

Boosting Algorithm:

Boosting algorithms are a class of machine learning techniques that aim to enhance the predictive performance of models by combining the strengths of multiple weak learners. Weak learners, often simple decision trees, are sequentially trained on the data, with each subsequent learner focusing on the mistakes of its predecessors. The key idea is to assign higher weights to misclassified instances, thereby emphasizing their importance in subsequent iterations. Gradient Boosting, AdaBoost, and XG Boost are popular boosting algorithms. Gradient Boosting optimizes model errors by minimizing the gradient of the loss function, AdaBoost assigns varying weights to instances based on their classification success, and XG Boost extends these concepts with regularization and parallel processing capabilities. Boosting algorithms excel in improving predictive accuracy and are widely utilized in diverse machine learning applications**.**

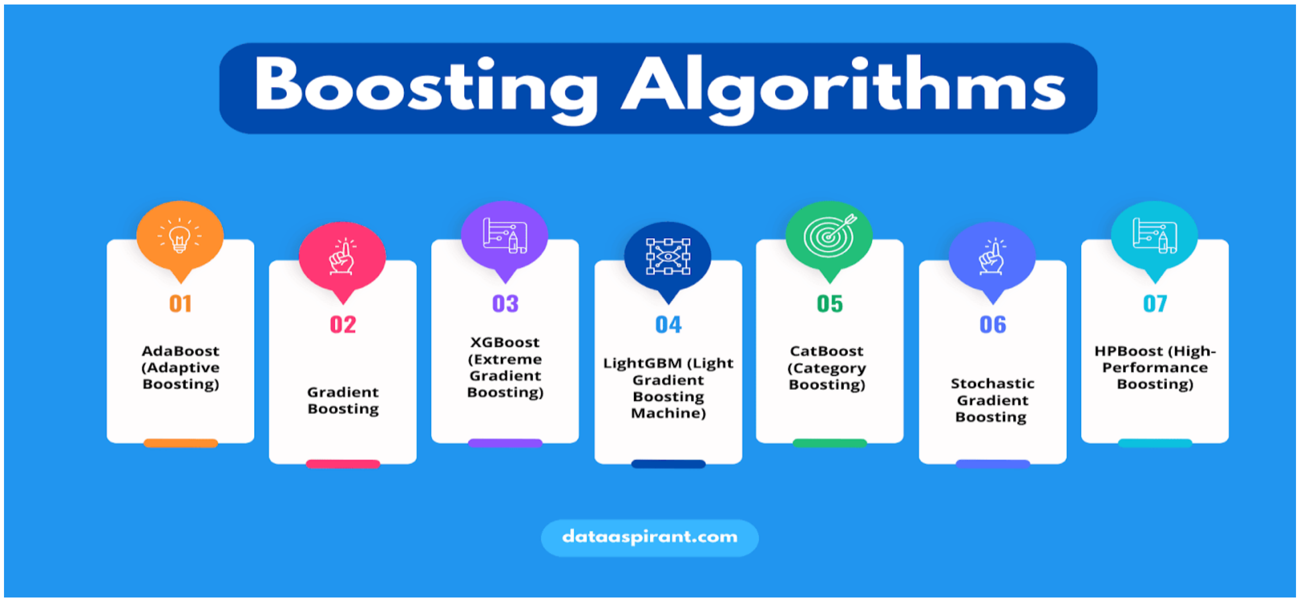

Boosting Algorithms
01
AdaBoost (Adaptive Boosting)
02
Gradient Boosting
03
XGBoost (Extreme Gradient Boosting)
04
LightGBM (Light Gradient Boosting Machine)
05
CatBoost (Category Boosting)
06
Stochastic Gradient Boosting
07
HPBoost (High-Performance Boosting)
dataaspirant.com

Fig 5.5

Boosting algorithms, like Gradient Boosting or AdaBoost, can be used to predict stock price movements. These algorithms work by combining multiple weak learners (usually decision trees) into a strong learner. Each weak learner corrects the errors of its predecessor, gradually improving prediction accuracy. In the context of stock prices, boosting algorithms can learn complex patterns and relationships in the data, helping to forecast price movements more accurately. Trained on historical price data and relevant features, boosting models can provide insights into potential stock price deductions, indicating when prices might decrease based on the learned patterns. These algorithms are popular in financial modeling due to their ability to handle non-linear relationships and adapt to changing market conditions. However, like all predictive models, their effectiveness depends on the quality of data and features used for training.

CHAPTER 6
RESULTS AND TEST CASES

6.1 TESTING APPROACH

Test Integration Approaches

There are fundamentally 2 approaches for doing test integration:

1.Bottom-up approach

2.Top-down approach.

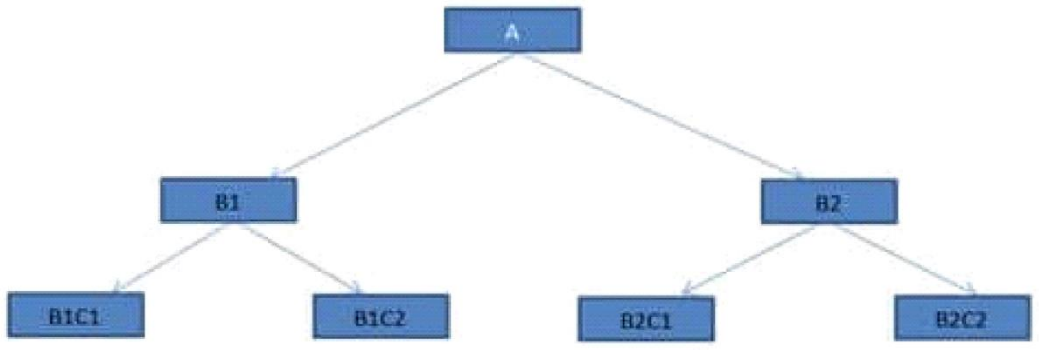


Fig 6.1

Bottom-up approach

As the name implies, base up testing starts with the smallest or deepest unit of the application and gradually works its way up. Reconciliation testing starts with the smallest module and works its way up to the largest components in the application. This cooperation continues until all of the modules are integrated, at which point the entire application is tested as a single entity.

Hierarchical approach

This approach starts with the topmost module and works its way down to the lower modules gradually. Only the top module is being tested in isolation. The lower modules are then independently integrated after this. The cycle is repeated until each module has been used and tested.

Programming testing is an examination directed to furnish partners with data about the nature of the item or administration under test. Programming Testing additionally gives a goal, free perspective on the product to permit the business to appreciate and figure out the dangers at execution of the product. Test procedures incorporate, however are not restricted to, the most common way of executing a program or application with the plan of finding programming bugs.

Programming Testing can likewise be expressed as the method involved with approving and confirming that a product program/application/item:

●Meets the business and specialized prerequisites that directed its plan and Improvement.

●Fills in true to form and can be executed with similar qualities.

6.2 TESTING Techniques

Useful Testing

Utilitarian tests give deliberate exhibitions that capabilities tried are accessible as determined by the business and specialized necessities, framework documentation, and client manuals.

Useful testing is focused on the accompanying things:

●Capabilities: Recognized capabilities should be worked out.

●Yield: Distinguished classes of programming yields should be worked out.

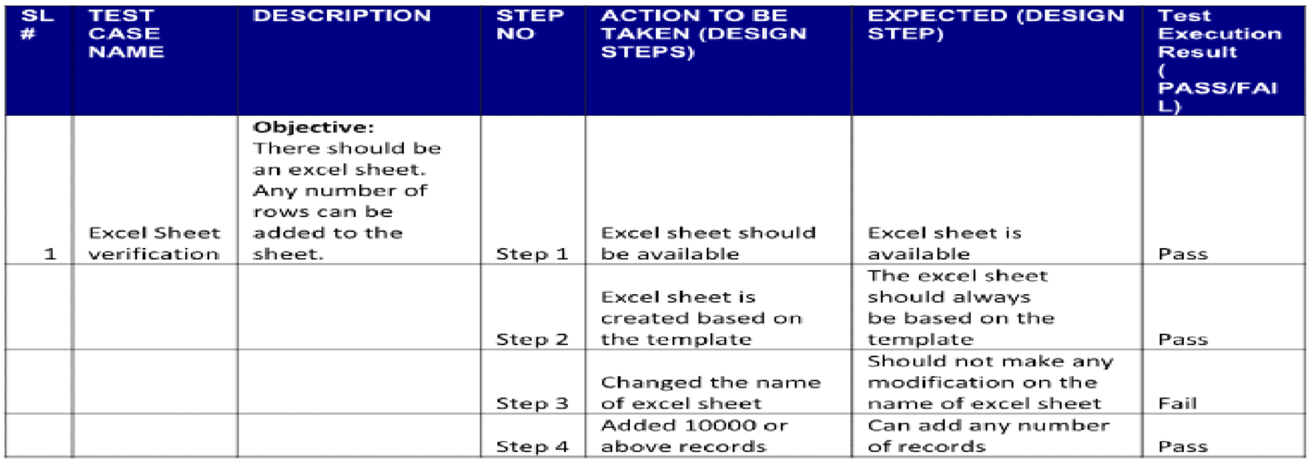
●Frameworks/Strategies: framework ought to work appropriately

Combination Testing

Programming combination testing is the steady coordination testing of at least two incorporated programming parts on a solitary stage to deliver disappointments brought about by interface surrenders.

Experiment for Succeed Sheet Check:

Here in AI we are managing dataset which is in succeed sheet design so assuming that any experiment we really want implies we really want to check succeed document.



SL #	TEST CASE NAME	DESCRIPTION	STEP NO	ACTION TO BE TAKEN (DESIGN STEPS)	EXPECTED (DESIGN STEP)	Test Execution Result (PASS/FAIL)
1	Excel Sheet verification	**Objective:** There should be an excel sheet. Any number of rows can be added to the sheet.	Step 1	Excel sheet should be available	Excel sheet is available	Pass
			Step 2	Excel sheet is created based on the template	The excel sheet should always be based on the template	Pass
			Step 3	Changed the name of excel sheet	Should not make any modification on the name of excel sheet	Fail
			Step 4	Added 10000 or above records	Can add any number of records	Pass

Fig 6.2

White Box

TEST CASE1:

INPUT: Anaconda Navigator

OUTPUT: Jupyter Notebook and Browser

TEST CASE2:

INPUT: PYTHON PACKAGES IMPORT (Pandas, Numpy, Scikit, Matplot, Seaborn)

OUTPUT: CHECKING OF MODULE

TEST CASE3:

INPUT: EXECUTION OF CODE

OUTPUT: GRAPH AND ACCURACY

CHAPTER 7

CONCLUSION AND FUTURE WORK

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion:

In summary, the proposed dyslexia detection system, driven by a boosting algorithm in a Flask web application, offers efficient and accurate identification. With real-time predictions, continuous monitoring, and user-friendly features, it stands as a promising solution for early dyslexia detection. The system's scalability, adaptability, and privacy measures contribute to its potential to positively impact education and support for individuals with dyslexia.

7.2 Future Work:

Potential future enhance ements for the Dyslexia Detection Web App could include integrating additional machine learning models for improved accuracy, incorporating user feedback mechanisms to continuously refine and adapt the algorithm, enhancing the user interface for a more intuitive experience, and exploring the integration of emerging technologies such as natural language processing or deep learning to further advance dyslexia detection capabilities. Additionally, ongoing research and collaboration with experts in dyslexia and related fields could contribute to refining the algorithm and expanding the app's effectiveness in real-world scenarios

7.3 Source Code:

dyslexia boost code

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd

from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.metrics import precision_recall_fscore_support
%matplotlib inline

data=pd.read_csv('labelled_data_new.csv')
y=data.Label
X=data.drop(['Label'],axis=1)
data

data.head()

data.columns

prec=[0,0,0,0,0]
rec=[0,0,0,0,0]
f1=[0,0,0,0,0]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.7,random_state=1
0)

X_test2=X_test.copy()

sc=StandardScaler(copy=False)
sc.fit_transform(X_train)
sc.transform(X_test)

X_train

from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
pre = clf.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre, average='macro')
print("SVM with linear kernel - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))

from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf2 = SVC(kernel='rbf')
clf2.fit(X_train, y_train)
pre2 = clf2.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre2, average='macro')
print("SVM with rbf kernel - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))

tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
clf3 = GridSearchCV(SVC(), tuned_parameters,scoring='f1_macro')
clf3.fit(X_train, y_train)
print("Best parameters set found on development set:")
print(clf3.best_params_)

means = clf3.cv_results_['mean_test_score']
for mean, params in zip(means, clf3.cv_results_['params']):
        print("%0.3f for %r"% (mean, params))
```

```
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf3 = SVC(kernel='rbf')
clf3.fit(X_train, y_train)  # Fit the model with training data

pre3 = clf3.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre3, average='macro')
print("Gridsearch with SVM - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))


from sklearn.metrics import precision_recall_fscore_support
from sklearn.ensemble import RandomForestClassifier

clf4 = RandomForestClassifier(random_state=0)
clf4.fit(X_train, y_train)
pre4 = clf4.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre4, average='macro')
print("Random forest classifier - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))


param2={'n_estimators' : [10,100,500,1000]}
clf5=GridSearchCV(RandomForestClassifier(random_state=0),param2,scoring='
f1_macro')
clf5.fit(X_train,y_train)
print("Best parameters set found on development set:")
print(clf5.best_params_)


means1 = clf5.cv_results_['mean_test_score']
for mean1, params1 in zip(means1, clf5.cv_results_['params']):
        print("%0.3f for %r"% (mean1, params1))
```

```
from sklearn.metrics import precision_recall_fscore_support

# Assuming clf5 is already defined and fitted

pre5 = clf5.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre5, average='macro')
print("Gridsearch with Random forest classifier - Precision = %0.3f, Recall =
%0.3f, F1-score = %0.3f"
      % (prec, rec, f1))


plt.figure(figsize=(15,5))
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=prec,label='precision')
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=rec,label='recall')
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=f1,label='f1-score')

pd.concat([X_test2,y_test],axis=1)

pd.Series(pre3,index=X_test.index)

!pip install xgboost
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report
# Initialize the XGBoost classifier
xgb_classifier = xgb.XGBClassifier()

# Train the classifier
xgb_classifier.fit(X_train, y_train)

# Make predictions
y_pred = xgb_classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

!pip install catboost
from catboost import CatBoostClassifier

catboost_classifier = CatBoostClassifier()

# Train the classifier
catboost_classifier.fit(X_train, y_train, verbose=False)

# Make predictions
y_pred = catboost_classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# ... (your existing code)

# 1. Get input from the user
user_input = {
    'Language_vocab': float(input("Enter Language_vocab value: ")),
    'Memory': float(input("Enter Memory value: ")),
    'Speed': float(input("Enter Speed value: ")),
    'Visual_discrimination': float(input("Enter Visual_discrimination value: ")),
    'Audio_Discrimination': float(input("Enter Audio_Discrimination value: ")),
    'Survey_Score': float(input("Enter Survey_Score value: "))
}

# 2. Create a new DataFrame with user input
new_data = pd.DataFrame([user_input])
```

```
# 3. Standardize the input data
sc.transform(new_data)

# 4. Use the trained models to predict the label for the new data
predicted_label_svm_linear = clf.predict(new_data)
predicted_label_svm_rbf = clf2.predict(new_data)
predicted_label_svm_grid_search = clf3.predict(new_data)
predicted_label_rf = clf4.predict(new_data)
predicted_label_rf_grid_search = clf5.predict(new_data)

# 5. Display the predicted label
print("Predicted Label (SVM Linear):", predicted_label_svm_linear[0])
print("Predicted Label (SVM RBF):", predicted_label_svm_rbf[0])
print("Predicted Label (SVM Grid Search):",
predicted_label_svm_grid_search[0])
print("Predicted Label (Random Forest):", predicted_label_rf[0])
print("Predicted Label (Random Forest Grid Search):",
predicted_label_rf_grid_search[0])


import pickle
filename = 'model.pkl'
pickle.dump(clf4, open(filename, 'wb'))

loaded_model = pickle.load(open('model.pkl', 'rb'))


test_vector = loaded_model.predict([[0.5,0.6,0.5,0.8,0.6,0.7]])
test_vector
if test_vector == 1:
   print("1st stage")
elif test_vector == 2:
   print("2nd stage")
else:
   print("No Dyslexia")
```

2.dylexia.py

```
import numpy as np
import pandas as pd
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
loaded_model = pickle.load(open('model.pkl', 'rb'))



@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    float_features = [float(x) for x in request.form.values()]
    final_features = [np.array(float_features)]
    prediction = loaded_model.predict( final_features )
    if prediction == 1:
        pred = "1st stage"
    elif prediction == 2:
        pred = "2nd stage"
    else:
        pred = "No Dyslexia"
    output = pred

    return render_template('index.html', prediction_text=output)

if __name__ == "__main__":
    app.run(debug=True)
```

3.dyslexia.ipynb

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd

from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.metrics import precision_recall_fscore_support
%matplotlib inline

data=pd.read_csv('labelled_data_new.csv')
y=data.Label
X=data.drop(['Label'],axis=1)
data

data.head()

data.columns

prec=[0,0,0,0,0]
rec=[0,0,0,0,0]
f1=[0,0,0,0,0]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.7,random_state=1
0)

X_test2=X_test.copy()
```

```
sc=StandardScaler(copy=False)
sc.fit_transform(X_train)
sc.transform(X_test)

X_train

from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
pre = clf.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre, average='macro')
print("SVM with linear kernel - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))


from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf2 = SVC(kernel='rbf')
clf2.fit(X_train, y_train)
pre2 = clf2.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre2, average='macro')
print("SVM with rbf kernel - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))


tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
clf3 = GridSearchCV(SVC(), tuned_parameters,scoring='f1_macro')
clf3.fit(X_train, y_train)
print("Best parameters set found on development set:")
print(clf3.best_params_)
```

```
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC

clf3 = SVC(kernel='rbf')
clf3.fit(X_train, y_train)  # Fit the model with training data

pre3 = clf3.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre3, average='macro')
print("Gridsearch with SVM - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))


from sklearn.metrics import precision_recall_fscore_support
from sklearn.ensemble import RandomForestClassifier

clf4 = RandomForestClassifier(random_state=0)
clf4.fit(X_train, y_train)
pre4 = clf4.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre4, average='macro')
print("Random forest classifier - Precision = %0.3f, Recall = %0.3f, F1-score =
%0.3f"
      % (prec, rec, f1))



param2={'n_estimators' : [10,100,500,1000]}
clf5=GridSearchCV(RandomForestClassifier(random_state=0),param2,scoring='
f1_macro')
clf5.fit(X_train,y_train)
print("Best parameters set found on development set:")
print(clf5.best_params_)

from sklearn.metrics import precision_recall_fscore_support

# Assuming clf5 is already defined and fitted
```

```
pre5 = clf5.predict(X_test)
prec, rec, f1, _ = precision_recall_fscore_support(y_test, pre5, average='macro')
print("Gridsearch with Random forest classifier - Precision = %0.3f, Recall =
%0.3f, F1-score = %0.3f"
      % (prec, rec, f1))


plt.figure(figsize=(15,5))
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=prec,label='precision')
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=rec,label='recall')
sns.lineplot(x=['SVM linear ','SVM rbf','SVM grid
search','RandomForest','RandomForest gridsearch'],y=f1,label='f1-score')

# ... (your existing code)

# 1. Get input from the user
user_input = {
    'Language_vocab': float(input("Enter Language_vocab value: ")),
    'Memory': float(input("Enter Memory value: ")),
    'Speed': float(input("Enter Speed value: ")),
    'Visual_discrimination': float(input("Enter Visual_discrimination value: ")),
    'Audio_Discrimination': float(input("Enter Audio_Discrimination value: ")),
    'Survey_Score': float(input("Enter Survey_Score value: "))
}

# 2. Create a new DataFrame with user input
new_data = pd.DataFrame([user_input])

# 3. Standardize the input data
sc.transform(new_data)

# 4. Use the trained models to predict the label for the new data
predicted_label_svm_linear = clf.predict(new_data)
predicted_label_svm_rbf = clf2.predict(new_data)
predicted_label_svm_grid_search = clf3.predict(new_data)
predicted_label_rf = clf4.predict(new_data)
```

```
predicted_label_rf_grid_search = clf5.predict(new_data)

# 5. Display the predicted label
print("Predicted Label (SVM Linear):", predicted_label_svm_linear[0])
print("Predicted Label (SVM RBF):", predicted_label_svm_rbf[0])
print("Predicted Label (SVM Grid Search):",
predicted_label_svm_grid_search[0])
print("Predicted Label (Random Forest):", predicted_label_rf[0])
print("Predicted Label (Random Forest Grid Search):",
predicted_label_rf_grid_search[0])



test_vector = clf.predict([[0.3,0.3,0.3,0.2,0.4,0.2]])
#test_vector
if test_vector[0] == 1:
    print("1st stage")
elif test_vector[0] == 2:
    print("2nd stage")
else:
    print("No Dyslexia");
```

7.4 SCREEN SHOT:

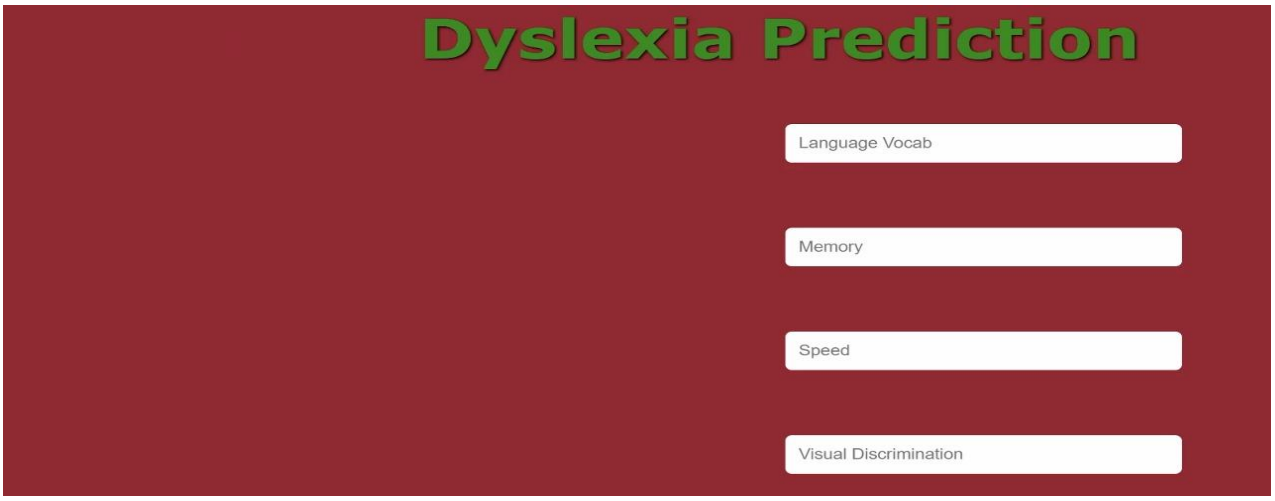


Fig 7.1

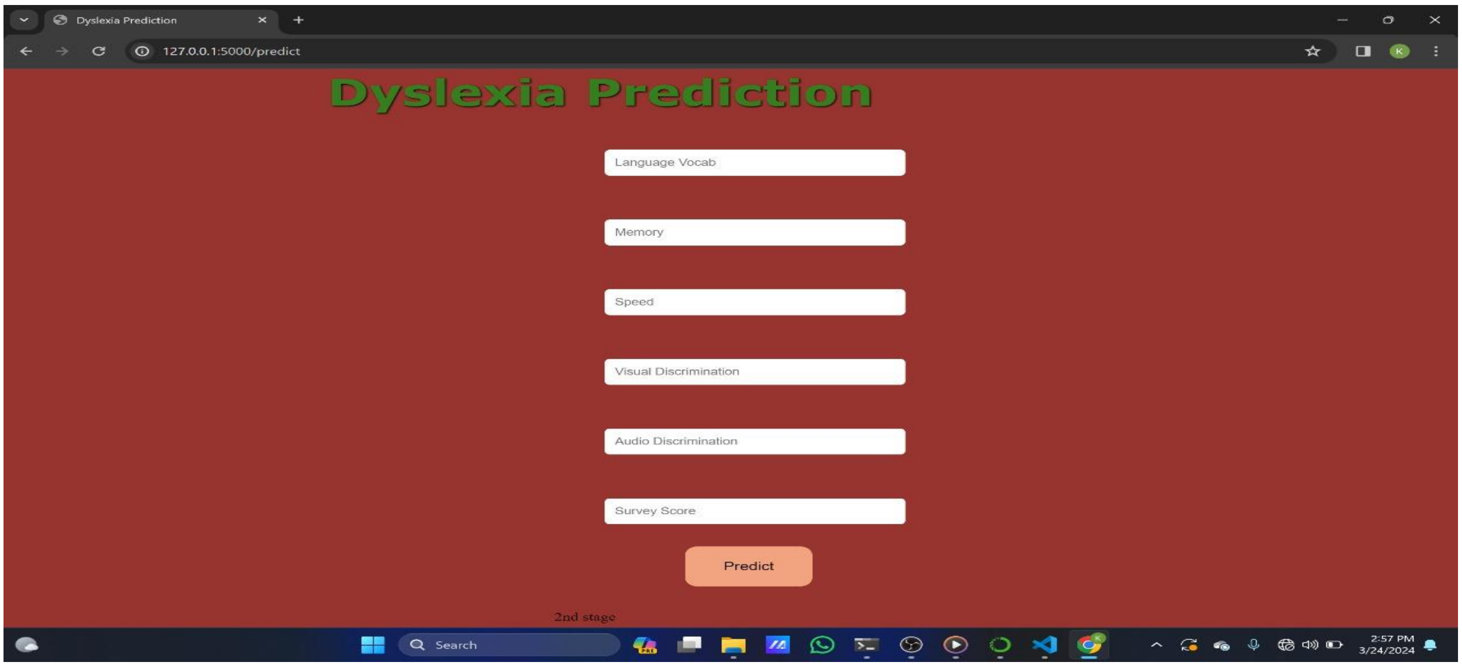
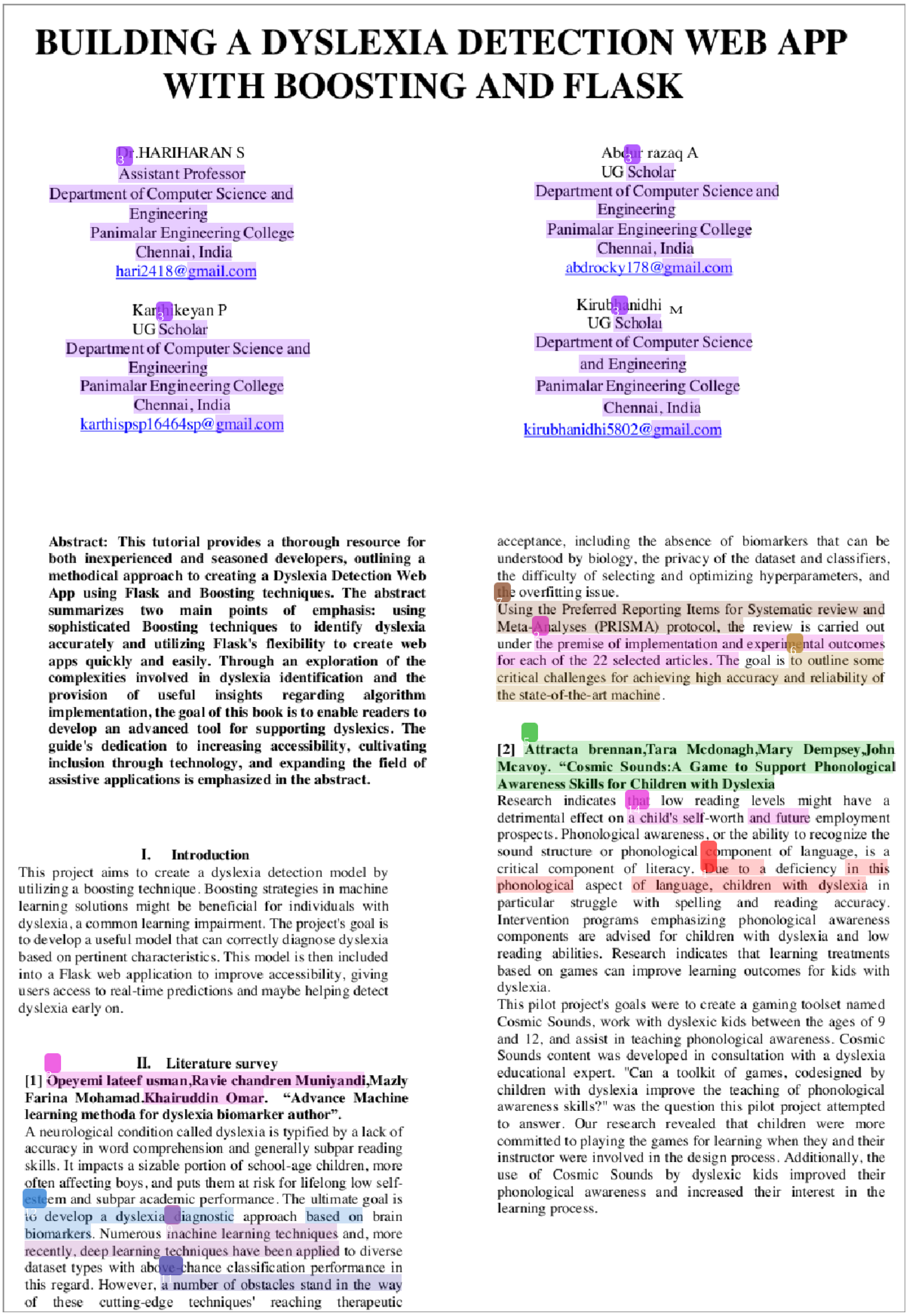


Fig 7.2

RE-2022-221783 - Turnitin Plagiarism Report

by Abdur Razaq A

Submission date: 24-Mar-2024 02:55PM (UTC+0300)
Submission ID: 271711301284
File name: RE-2022-221783.docx (77.86K)
Word count: 2152
Character count: 13042



BUILDING A DYSLEXIA DETECTION WEB APP WITH BOOSTING AND FLASK

Dr.HARIHARAN S
Assistant Professor
Department of Computer Science and Engineering
Panimalar Engineering College
Chennai, India
hari2418@gmail.com

Abdur razaq A
UG Scholar
Department of Computer Science and Engineering
Panimalar Engineering College
Chennai, India
abdrocky178@gmail.com

Karthikeyan P
UG Scholar
Department of Computer Science and Engineering
Panimalar Engineering College
Chennai, India
karthispsp16464sp@gmail.com

Kirubhanidhi M
UG Scholar
Department of Computer Science and Engineering
Panimalar Engineering College
Chennai, India
kirubhanidhi5802@gmail.com

Abstract: This tutorial provides a thorough resource for both inexperienced and seasoned developers, outlining a methodical approach to creating a Dyslexia Detection Web App using Flask and Boosting techniques. The abstract summarizes two main points of emphasis: using sophisticated Boosting techniques to identify dyslexia accurately and utilizing Flask's flexibility to create web apps quickly and easily. Through an exploration of the complexities involved in dyslexia identification and the provision of useful insights regarding algorithm implementation, the goal of this book is to enable readers to develop an advanced tool for supporting dyslexics. The guide's dedication to increasing accessibility, cultivating inclusion through technology, and expanding the field of assistive applications is emphasized in the abstract.

I. Introduction

This project aims to create a dyslexia detection model by utilizing a boosting technique. Boosting strategies in machine learning solutions might be beneficial for individuals with dyslexia, a common learning impairment. The project's goal is to develop a useful model that can correctly diagnose dyslexia based on pertinent characteristics. This model is then included into a Flask web application to improve accessibility, giving users access to real-time predictions and maybe helping detect dyslexia early on.

II. Literature survey

[1] Opeyemi lateef usman,Ravie chandren Muniyandi,Mazly Farina Mohamad,Khairuddin Omar. "Advance Machine learning methoda for dyslexia biomarker author".
A neurological condition called dyslexia is typified by a lack of accuracy in word comprehension and generally subpar reading skills. It impacts a sizable portion of school-age children, more often affecting boys, and puts them at risk for lifelong low self-esteem and subpar academic performance. The ultimate goal is to develop a dyslexia diagnostic approach based on brain biomarkers. Numerous machine learning techniques and, more recently, deep learning techniques have been applied to diverse dataset types with above-chance classification performance in this regard. However, a number of obstacles stand in the way of these cutting-edge techniques' reaching therapeutic acceptance, including the absence of biomarkers that can be understood by biology, the privacy of the dataset and classifiers, the difficulty of selecting and optimizing hyperparameters, and the overfitting issue.
Using the Preferred Reporting Items for Systematic review and Meta-Analyses (PRISMA) protocol, the review is carried out under the premise of implementation and experimental outcomes for each of the 22 selected articles. The goal is to outline some critical challenges for achieving high accuracy and reliability of the state-of-the-art machine.

[2] Attracta brennan,Tara Mcdonagh,Mary Dempsey,John Mcavoy. "Cosmic Sounds:A Game to Support Phonological Awareness Skills for Children with Dyslexia
Research indicates that low reading levels might have a detrimental effect on a child's self-worth and future employment prospects. Phonological awareness, or the ability to recognize the sound structure or phonological component of language, is a critical component of literacy. Due to a deficiency in this phonological aspect of language, children with dyslexia in particular struggle with spelling and reading accuracy. Intervention programs emphasizing phonological awareness components are advised for children with dyslexia and low reading abilities. Research indicates that learning treatments based on games can improve learning outcomes for kids with dyslexia.
This pilot project's goals were to create a gaming toolset named Cosmic Sounds, work with dyslexic kids between the ages of 9 and 12, and assist in teaching phonological awareness. Cosmic Sounds content was developed in consultation with a dyslexia educational expert. "Can a toolkit of games, codesigned by children with dyslexia improve the teaching of phonological awareness skills?" was the question this pilot project attempted to answer. Our research revealed that children were more committed to playing the games for learning when they and their instructor were involved in the design process. Additionally, the use of Cosmic Sounds by dyslexic kids improved their phonological awareness and increased their interest in the learning process.

[3] Ivan A.Vajs,Goran S.Kvascev,Tamara M.Papic,Milica M.Jankovic, " Eye-Tracking Image Encoding: Autoencoders for the Crossing of Language Boundaries in Developmental Dyslexia Detection".

Academic performance is negatively impacted by developmental dyslexia, an issue that has been extensively studied and documented. Even though there is a wealth of research on the identification and assessment of developmental dyslexia, the study methods differ greatly, which frequently makes it challenging to share the knowledge that is discovered. In order to bridge the gap between various study designs, this paper developed a machine learning based pipeline that was tested on two distinct eye-tracking datasets (using one for testing and using the other for training). Thirty volunteers were tracked using a remote eye tracker as they read text printed in Serbian on various color configurations. In the second dataset, 185 participants used a goggle-based technology to record eye-tracking data while they read material written in Swedish.
Using different time window configurations, the data from both datasets were converted to grayscale images, parsed the signals, and plotted in a two-dimensional plane. An Autoencoder neural network was trained using the train images, and features describing each instance of the training and test sets were created using the reconstruction error of the images. Several machine learning methods were trained on the train feature set and subsequently assessed on the testing feature dataset. Testing on data from Serbian readers yielded a classification accuracy of 85.6%, whereas testing on Swedish readers produced an accuracy of 82.9%. Despite significant variations in the trial design, the suggested pipeline was demonstrated to be portable between the datasets, indicating promise for integrating different eye-tracking dyslexia Studies.

[4] SANDRA SANCHEZ-GORDON, CARMEN AGUILAR-MAYANQUER, TANIA CALLE-JIMENEZ, " model for profiling users with disabilities on e-Learning Platforms"
Worldwide, e-learning platforms are used by millions of individuals. However, because these platforms are not accessible, those with impairments continue to face difficulties when utilizing them. A model for creating user profiles of people with disabilities is put forth in this study. With the help of this paradigm, e-learning systems can automatically alter their interfaces based on user accessibility demands. This would allow all students, disabled or not, to have equal access. Several studies about user profiling on e-learning platforms were found and examined in this context. The model taken into consideration takes into account the Web Content Accessibility Guidelines, which may be incorporated into the e-Learning platform interfaces, as well as the metadata, which is based on Schema.org and represents the accessibility requirements of users. The researchers designed the model and defined a description of the interaction between users, user interfaces, WCAG, Schema.org, and the extensible Markup Language (XML) profile developed using Unified Modelling Language diagrams. WAVE and ARC Toolkit were used for the prototype's testing. Using the help of forty-four users, the System Usability Scale (SUS) was used for validation. The inclusion of two special categories for combined accessibility needs related to linguistics and the elderly, the automatic generation of the profile as an XML file containing the metadata required to enable the adaptation of e-Learning platform interfaces, and the identification of automatically implementable WCAG success criteria are the most notable outcomes of this study.

[5] Douglas Teoh, " TestGraphia, A Software System for the Early Diagnosis of Dysgraphia"

One particular writing issue that affects the replication of alphabetical and numeric symbols is called dysgraphia. Dyspraxia, a condition attributable to incomplete lateralization marked by difficulty reproducing alphabetical and numeric signs, may be connected to dysgraphia. Since dysgraphia's causes are unknown, it's critical to identify symptoms as soon as possible. The most popular method for diagnosing dysgraphia in academic and medical settings is a writing quality assessment on paper sheets. Writing quality score guidelines provide the basis of a writing analysis. In this study, we describe TestGraphia, a software solution designed to help physicians objectively monitor patients with dysgraphia and assist with diagnosis.
[illegible]
and custom algorithms for document analysis. With the help of this software, a forms analysis can be completed much faster than with conventional methods, which allows for extensive screening operations at a lower cost and time. Potential dynamic changes in dysgraphia screening can be evaluated by keeping a suitable eye on writing quality at the right frequency and in a non-invasive manner, both at the patient's home and in the laboratory. The mean time to do a diagnosis with the method we shall present is over ten times faster with reliable outcomes.

III. Existing System

Traditionally, psychological and educational tests are used to diagnose dyslexia. Though some researchers are investigating machine learning for this reason, no standardized method has been developed as of yet. It is advisable to look through recent studies and open-source initiatives in related domains to find the most recent information. Stakeholder involvement and ethical considerations are crucial while creating treatments for learning difficulties.

Drawback:
- Evaluation subjectivity
- Reliance on human expertise
- Restricted accessibility
- Expensive interventions
- Absence of personalization

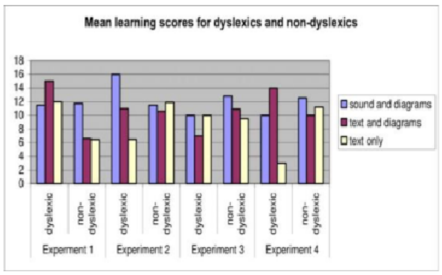
IV. Proposed System

The solution that is being suggested takes a fresh approach to dyslexia identification by integrating a boosting algorithm that is intended to improve precision and effectiveness when compared to traditional techniques. This method makes use of machine learning, automating the identification procedure by leveraging the boosting algorithm's capabilities. A Flask web application facilitates real-time predictions, overcoming subjectivity issues and providing ongoing monitoring. The system's adaptability to new technologies guarantees a data-driven, user-friendly, and efficient way to identify dyslexia in children at an early age.

Merits:

• Accuracy and Customization.
• Time-saving and effectiveness.
• Ongoing Observation
• Forecasts in Real Time
• Effective Pattern Identification

V. Results



VI. Conclusion

In conclusion, the suggested dyslexia detection method provides quick and precise identification. It is powered by a boosting algorithm within a Flask web application. With its user-friendly features, continuous monitoring, and real-time forecasts, it appears to be a potential approach for early dyslexia diagnosis. The system has the opportunity to have a significant impact on dyslexics' education and support because of its scalability, customization, and privacy features.

VII. Future Work

.Potential future enhance ements for the Dyslexia Detection Web App could include integrating additional machine learning models for improved accuracy, incorporating user feedback mechanisms to continuously refine and adapt the algorithm, enhancing the user interface for a more intuitive experience, and exploring the integration of emerging technologies such as natural language processing or deep learning to further advance dyslexia detection capabilities. Additionally, ongoing research and collaboration with experts in dyslexia and related fields could contribute to refining the algorithm and expanding the app's effectiveness in real-world scenarios

VIII. Reference

[1] .A. Elnakib, A. Soliman, M. Nitzken, M. F. Casanova, G. Gimel'farb and A. El-Baz, "Magnetic resonance imaging findings for dyslexia: A review", J. Biomed. Nanotechnol., vol. 10, no. 10, pp. 2778-2805, Oct. 2014.

[2] J. M. Fletcher et al., "Classification of learning disabilities: An evidence-based evaluation" in Identification of Learning Disabilities: Research to Practice, Washington, DC, USA:Erlbaum Associates Publishers, pp. 185-250, Jan. 2002.

[3].P. Tamboer, H. C. M. Vorst, S. Ghebreab and H. S. Scholte, "Machine learning and dyslexia: Classification of individual structural neuro-imaging scans of students with and without dyslexia", NeuroImage: Clin., vol. 11, pp. 508-514, 2016.

[4].S. O. Wajuihian, "Neurobiology of developmental dyslexia Part 1: A review of evidence from autopsy and structural neuro-imaging studies", Optometry Vis. Develop., vol. 43, no. 3, pp. 121-131, 2012.

[5].S. O. Wajuihian and K. S. Naidoo, "Dyslexia: An overview", Afr. Vis. Eye Health, vol. 70, no. 2, pp. 89-98, Dec. 2011.

[6].B. Sarah, C. Nicole, H. Ardag, M. Madelyn, S. K. Holland and H. Tzipi, "An fMRI Study of a Dyslexia biomarker", J. Young Investigators, vol. 26, no. 1, pp. 1-4, 2014.

[7].Dyslexia in the Classroom—What Every Teacher Needs to Know, 2017, [online] Available: http://www.DyslexiaIDA.org.

[8].N. A. M. Yuzaidey, N. C. Din, M. Ahmad, N. Ibrahim, R. A. Razak and D. Harun, "Interventions for children with dyslexia: A review on current intervention methods", Med. J. Malaysia, vol. 73, no. 5, pp. 311-320, Oct. 2018.

[9].S. Kaisar, "Developmental dyslexia detection using machine learning techniques: A survey", ICT Exp., vol. 6, no. 3, pp. 181-184, 2020.

[10].F. Ramus, I. Altarelli, K. Jednoróg, J. Zhao and L. Scotto di Covella, "Neuroanatomy of developmental dyslexia: Pitfalls and promise", Neurosci. Biobehav. Rev., vol. 84, pp. 434-452, Jan. 2018

RE-2022-221783-plag-report

ORIGINALITY REPORT

17 %	14%	15%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	5%
2	www.journaltocs.ac.uk Internet Source	2%
3	www.ijritcc.org Internet Source	1%
4	Ivan A. Vajs, Goran S. Kvascev, Tamara M. Papic, Milica M. Jankovic. "Eye-Tracking Image Encoding: Autoencoders for the Crossing of Language Boundaries in Developmental Dyslexia Detection", IEEE Access, 2023 Publication	1%
5	dblp.dagstuhl.de Internet Source	
6	Submitted to Universiti Kebangsaan Malaysia Student Paper	1%
7	discovery.researcher.life Internet Source	1%
		1%

8 Sandra Sanchez-Gordon, Carmen Aguilar- Mayanquer, Tania Calle-Jimenez. "Model for profiling users with disabilities on e-Learningplatforms", IEEE Access, 2021 1%

Publication

paperswithcode.com

Internet Source

9 <1%

N.P. Guhan Seshadri, Sneha Agrawal, BikeshKumar Singh, B. Geethanjali, V. Mahesh, RamBilas Pachori. "EEG based classification of children with learning disabilities using shallow and deep neural network", Biomedical Signal Processing and Control, 2023

10 <1%

Publication

scholarship.law.upenn.edu

Internet Source

www.ijercse.com

Internet Source

Giovanni Dimauro, Vitoantonio Bevilacqua, Lucio Colizzi, Davide Di Pierro. "TestGraphia, aSoftware System for the Early Diagnosis of Dysgraphia", IEEE Access, 2020

11 <1%

Publication

12 <1%

Attracta Brennan, Tara McDonagh, Mary Dempsey, John McAvoy. "Cosmic Sounds: A Game to Support Phonological Awareness

13 <1%

14 <1%

CHAPTER 8
REFERENCE

8.REFERENCE:

1.A. Elnakib, A. Soliman, M. Nitzken, M. F. Casanova, G. Gimel’farb and A. El-Baz, "Magnetic resonance imaging findings for dyslexia: A review", J. Biomed. Nanotechnol., vol. 10, no. 10, pp. 2778-2805, Oct. 2014.

2.J. M. Fletcher et al., "Classification of learning disabilities: An evidence-based evaluation" in Identification of Learning Disabilities: Research to Practice, Washington, DC, USA:Erlbaum Associates Publishers, pp. 185-250, Jan. 2002.

3.P. Tamboer, H. C. M. Vorst, S. Ghebreab and H. S. Scholte, "Machine learning and dyslexia: Classification of individual structural neuro-imaging scans of students with and without dyslexia", NeuroImage: Clin., vol. 11, pp. 508-514, 2016.

4.S. O. Wajuihian, "Neurobiology of developmental dyslexia Part 1: A review of evidence from autopsy and structural neuro-imaging studies", Optometry Vis. Develop., vol. 43, no. 3, pp. 121-131, 2012.

5.S. O. Wajuihian and K. S. Naidoo, "Dyslexia: An overview", Afr. Vis. Eye Health, vol. 70, no. 2, pp. 89-98, Dec. 2011.

6.B. Sarah, C. Nicole, H. Ardag, M. Madelyn, S. K. Holland and H. Tzipi, "An fMRI Study of a Dyslexia biomarker", J. Young Investigators, vol. 26, no. 1, pp. 1-4, 2014.

7.Dyslexia in the Classroom—What Every Teacher Needs to Know, 2017, [online] Available: http://www.DyslexiaIDA.org.

8.N. A. M. Yuzaidey, N. C. Din, M. Ahmad, N. Ibrahim, R. A. Razak and D. Harun, "Interventions for children with dyslexia: A review on current intervention methods", Med. J. Malaysia, vol. 73, no. 5, pp. 311-320, Oct. 2018.

9.S. Kaisar, "Developmental dyslexia detection using machine learning techniques: A survey", ICT Exp., vol. 6, no. 3, pp. 181-184, 2020.

10.F. Ramus, I. Altarelli, K. Jednoróg, J. Zhao and L. Scotto di Covella, "Neuroanatomy of developmental dyslexia: Pitfalls and promise", Neurosci. Biobehav. Rev., vol. 84, pp. 434-452, Jan. 2018.