A

PROJECT REPORT

ON

Student Engagement Prediction in Online Session

Dissertation Submitted in Partial Fulfilment of The

Requirement For the Award Of

MASTER OF TECHNOLOGY

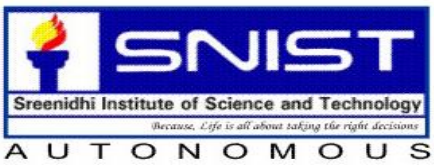
IN

COMPUTER NETWORKS AND INFORMATION SECURITY

By

Y. Trilochan Sashank

(20311D7801)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

SREENIDHI INSTITUTION OF SCIENCE & TECHNOLOGY

(An Autonomous Institution)

(AFFILIATED TO JNTU, HYDERABAD)

Yamnampet, Ghatkesar, R.R. District, Hyd-501301.

2023

Student Engagement Prediction in Online Session

Dissertation submitted in partial fulfilment of the

Requirement for the award of

MASTER OF TECHNOLOGY

IN

COMPUTER NETWORKS AND INFORMATION SECURITY

By

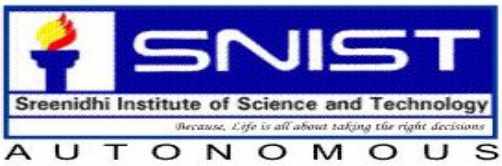
Y. Trilochan Sashank

(20311D7801)

Under the guidance of

Dr. K. Vijaya Lakshmi

Professor, Dept of IT



SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

SREENIDHI INSTITUTION OF SCIENCE AND TECHNOLOGY

(An Autonomous Institution)

(AFFILIATED TO JNTU, HYDERABAD)

Yamnampet, Ghatkesar, R.R. District, Hyd-501301

2023

SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

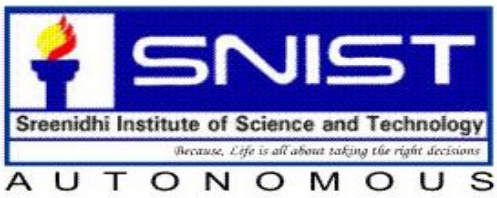
DEPARTMENT OF INFORMATION TECHNOLOGY

SREENIDHI INSTITUTION OF SCIENCE AND TECHNOLOGY

(An Autonomous Institution)

(Affiliated to JNT University; ISO Certified 9001:2000)

Yamnampet, Ghatkesar, Hyderabad-501301



CERTIFICATE

This is to certify that the Dissertation entitled **"Student Engagement Prediction in Online Session"** is bonafide work done and submitted by **Y. Trilochan Sashank (20311D7801)** in partial fulfilment of the requirement for the award of Degree of **Master of Technology in Computer Networks and Information Technology, SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY , Affiliated to Jawaharlal Nehru Technological University, Hyderabad** is a record of bonafide work carried out by her under *our guidance and supervision* from **April 2022** to **March 2023.**

The results presented in this dissertation have been verified and are found to be satisfactory. The results embodied in this dissertation have not been submitted to any other university for the award of any other degree or diploma**.**

Project Internal Guide

Dr. K. VIJAYA LAKSHMI

Professor, Department of IT,

SNIST, Hyderabad.

.

(M. Tech Co-ordinator)

Dr. B. INDIRA REDDY

Professor, Department of IT

SNIST.

Head of the Department

Dr. SUNIL BUTADA

Professor, Department of IT,

SNIST, Hyderabad**.**

External

ACKNOWLEDGEMENTS

I thank **Dr. K. Vijaya Lakshmi, As Professor, Department of IT** for supporting me as my Internal **Guide** and providing seamless knowledge over the past one year, and for providing right suggestion at every phase of the development of our project.

I extend my profound gratitude to thank **Prof. Sunil Butada, (Head of the Department)** & **Dr. B. Indira Reddy**, **(MTech Coordinator)**, Department of IT, SNIST, for his valuable suggestion and support throughout the course.

I express a wholehearted gratitude to **Dr. P. Narsimha Reddy, Executive Director,** and **Dr. Shiva Reddy, Principal, Sreenidhi Institute of Science and Technology** for providing us the conducive environment for carrying through our academic schedules and projects with ease.

I convey my heartful thanks to my Parents, friends, Technical and Non-Technical staff of the college for their constant support in the successful completion of the project.

Y. Trilochan Sashank

(20311D7801)

CONTENTS

ABSTRACT

The individuals who make up the globe constantly advance into the future and improve both their personal lives and the conditions in which they live. One ‘s education is the basis of one ‘s knowledge. Humans' education has a significant impact on their behaviour and IQ. Through the use of diverse pedagogical techniques, instructors always play a part in changing students' ways of thinking and developing their social and cognitive abilities.

However, getting students to participate in an online class is still difficult. In this study, we created an intelligent predictive system that aids instructors in anticipating students' levels of interest based on the information they learn in an online session and in motivating them through regular feedback.

The level of students' engagement is divided into three tiers based on their online session activities (Not engaged, passively engaged, actively engaged). The given data was subjected to the application of Decision Trees (DT), Random Forest Classifiers (RF), Logistic Regression (LR), and Long Short-Term Memory Networks are among the numerous machine learning approaches (LSTM). According to performance measurements, LSTM is the most accurate machine learning algorithm. The instructors can get in touch with the students and inspire them by improving their teaching approaches based on the results the system produces.

CHAPTER-1

INTRODUCTION

The rapid digitization of educational institutions is having a major impact on how teachers and other educational staff can facilitate data collection. In addition to offline-to-online learning styles and changes in institutional governance, the wealth of data about educational institutions is also changing very rapidly. A few examples of the numerous ways that web-based learning is currently used extensively in education (LMS) include Massive Open Online Courses (MOOCs), Virtual Learning Environments (VLEs), and Learning Management Systems. MOOC students can study anytime, anywhere [1]. MOOCs offer a new way of teaching, changing the traditional way of learning and attracting learners from all over the world. The three most popular platforms are Harvard, Edx and Coursera. MOOCs also contribute to the development of higher education [2].

However, it is challenging for teachers to keep track of everyone and identify disinterested students due to the large number of students in the class. You need an intelligent system that can analyse academic data in e-learning system reports or logs to predict student participation. Because student engagement is a measure of academic performance, it is essential to identify students who lack motivation and work to increase their engagement [3, 4]. A lot of people think that student engagement [5] is a proximal outcome that leads to distal outcomes like higher academic performance and lower rates of dropout. Provide a suggestion for a predictive method that is effective for both students. The authors suggest predicting the level of student engagement by using student data activity from the LMS (attendance at meetings, participation in forums and groups, access to course materials, etc.). All of this data can be found in a lot of reports.

The data from the author's Kaggle are used in this study. It provides a variety of reports, including:

(a) Outline of course activities

(b) Summary of session activities

(c) Student profiles for individual courses

(e) All user activity in content sections

(d) a general breakdown of user activity

(f) Participation in forums

(g) Participation in groups

(h) Completion of courses

(i) Completion of course coverage reports

(j) Participation in courses by users.

In this study, we will address the following research question.

Question:

"Can student Engagement be classified as Active (AE), Passive (PE), or Not-Engaged (NE) based on the amount of knowledge learned via online sessions?"

To detect Student Engagement Prediction in an online session in early stages, the Machine Learning [6] algorithms Long-Short Term Memory (LSTM), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF) and Support Vector Machine (SVM) are used in the early stages. Among all the algorithms LSTM is more accurate.

CHAPTER – 2

LITERATURE REVIEW

Numerous studies show a positive correlation between student participation and course outcomes. For instance, Atherton [7] demonstrated that students who regularly take assessments and obtain study materials through web-based systems perform well on exams. According to additional study, students who are very engaged likely to perform well on course quizzes and evaluations. [8] According to Rodgers [9], there was a strong correlation between student use of an e-learning system and the results of the courses. However, the majority of earlier research on engagement has overlooked student participation in web-based systems in favour of conventional instruction in colleges and schools. Furthermore, earlier studies on student engagement have relied on survey, qualitative, and statistical methodologies; however, these statistical tools are unable to detect hidden knowledge in student data.

Additionally, statistically qualitative techniques lack scalability and generalizability. Surveys are a poor choice for gauging student interest because, for instance, smaller kids cannot grasp the questions, and they take a long time to complete. The fact that these studies are focused on the course and the students' emotions and behaviour structure is another drawback. However, student engagement can also be influenced by students' involvement in educational activities.

A current study uses VLE log data to predict students' low participation in web-based learning systems. Students are not interrupted when accessing log data using machine learning techniques [10] and the data is not time sensitive.

CHAPTER-3

SYSTEM ANALYSIS

3.1: Existing System.

The **existing system** of student engagement prediction in online sessions involves the use of various technologies and tools to monitor and analyse student behaviour during an online session. These tools include data analytics, machine learning algorithms, and artificial intelligence.

The system typically tracks various metrics such as student attendance, participation, and interaction with the course material. It also analyses data such as students' response time, keystrokes, and mouse movements, to identify patterns and trends that indicate the level of student engagement.

Based on this analysis, the system generates reports and recommendations for instructors, which help them to adjust their teaching strategies to better engage students. The system can also provide personalized feedback and support to individual students to help them improve their engagement.

Overall, the system of student engagement prediction in online sessions aims to enhance the online learning experience by identifying and addressing factors that may impede student engagement, thereby improving student performance and outcomes.

3.2: Proposed System.

Student engagement is a complex and diverse phenomenon with behavioural, components that are cognitive, social, and emotional [11], [12]. The final element of student engagement is crucial yet difficult to achieve, particularly in an online setting. where the emotional bond between students and professors is difficult to see and assess. Student engagement may be compromised. As a result, teachers need to improve their work and further develop teaching methods (animation of presentations, pop-up quizzes during lectures, etc.). This survey helps predict student engagement based on student comprehension and time spent in online sessions. This benefits both instructors and students. Such a system allows teachers to change pedagogical approaches and course materials when it is clear that students are not working hard enough. Giving students regular comments and encouragement can also increase student engagement. Despite the structural relevance few academics have attempted to establish useful and trustworthy indicators of student involvement that are pertinent to this setting.

3.3 Requirements Specification:

The specification claims to effectively provide highly secure hosting for web servers. Software requirements involve software and hardware basics that must be installed on the server to ensure optimal operation of the application. These software as well as hardware configurations must be bootstrapped before installing the package. The operating system typically defines this set of requirements. During application development, these software and hardware requirements support operating system compatibility.

3.3.1 HARDWARE REQUIREMENTS:

Hardware requirements define each software and hardware endpoint of the system. These hardware requirements include structural features.

- System: Pentium IV 2.4GHz.
- Hard Disk: 100 GB.
- Monitoring: 15 VGA colours.
- Mouse: Logitech.
- RAM: 4 GB.

The above features described are the basic features that are necessary to run the Project.

3.3.2 SOFTWARE REQUIREMENTS:

The use of all necessary software components is outlined in a software requirement, such as: B. a system for processing data the output and quantity must be specified in the software.

The communication software's intended use for each software product is specified by each interface.

System software: Windows XP, Windows 7, and Windows 10

Development kit for Python: Notebook Jupiter Programming language, Visual Studio Code.

3.3.3 Functional requirements:

Functional necessities talk over with gadget necessities in a especially computationally in depth coding approach. The most important cause of defining "useful necessities" in phrases of especially green manufacturing and transport patterns is to seize the favoured conduct of software program programs in phrases of sensible implementations and technologies. enterprise process.

CHAPTER-4

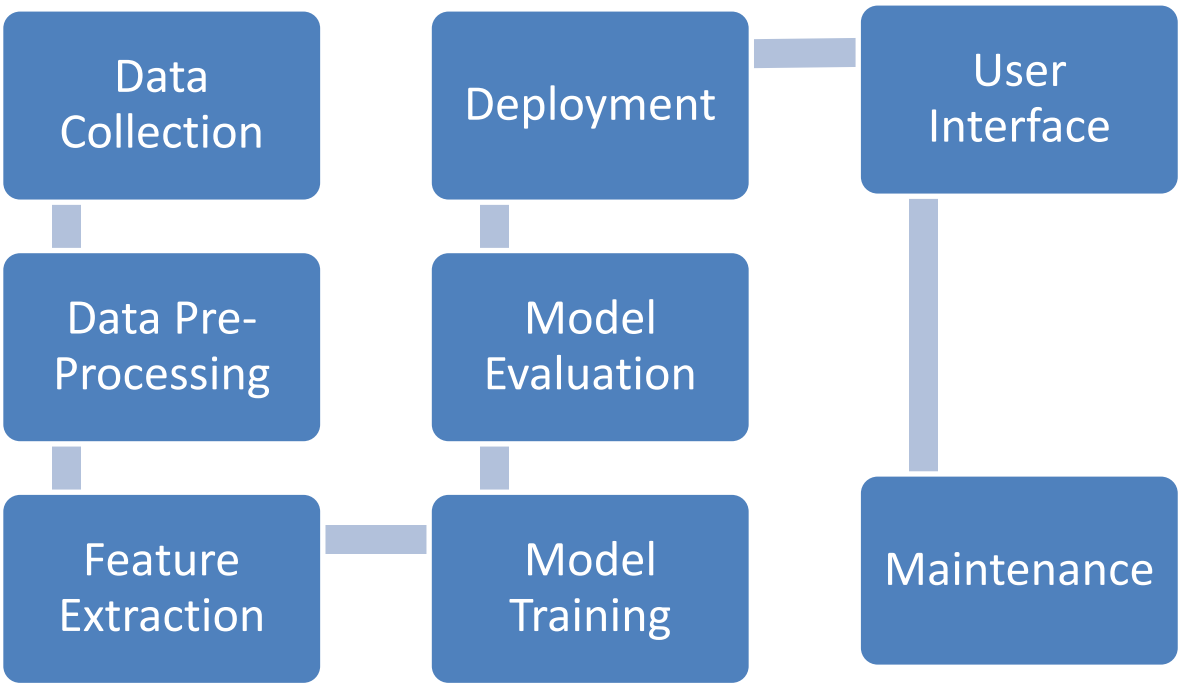
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE:

System architecture is essential for any software project, including machine learning projects. System architecture provides a high-level view of the project's components and how they interact with each other. It helps to ensure that the project is well-organized, efficient, and scalable.

A system architecture is necessary because it helps to:

- Plan the project: System architecture allows us to plan out the different components of our project in a structured manner, making it easier to manage and implement.
- Ensure compatibility: System architecture ensures that all the components of your project work together in a harmonious and compatible manner. This reduces the risk of errors or issues arising due to incompatible components.
- Improve efficiency: By breaking down the project into smaller components, system architecture can help you identify areas where we can optimize and improve efficiency.
- Manage complexity: Machine learning projects can be complex, and system architecture can help you manage this complexity by breaking the project down into smaller, more manageable components.
- Facilitate communication: System architecture provides a common language for discussing the project between team members, stakeholders, and clients. This helps to ensure that everyone is on the same page and working towards the same goals. Overall, system architecture is essential for any machine learning project. It will help us to plan, design, and implement your project efficiently, while ensuring that all components work together seamlessly.

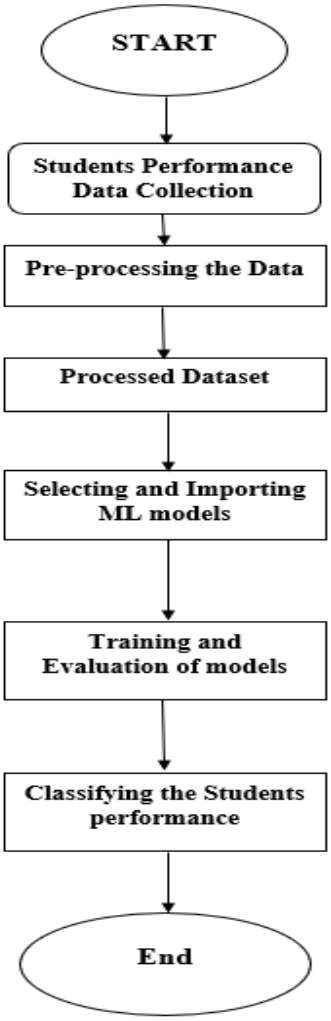


- Data Collection: Gather data from various sources such as online course platforms, learning management systems, and other sources.
- Data Pre-processing: Clean, transform, and normalize data before feeding it to the model.
- Feature Extraction: Extract meaningful features from the pre-processed data that can be used to train the machine learning model. Some examples of features could be time spent on a page, number of clicks, mouse movement, keystrokes, etc.
- Model Training: Train a machine learning model on the extracted features. You can use different algorithms such as logistic regression, decision trees, or neural networks.
- Model Evaluation: Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score.
- Deployment: Deploy the trained model on a web server or cloud platform so that it can be accessed via an API.
- User Interface: Build a user interface that allows users to input data and view the results of the model predictions.
- Maintenance: Continuously monitor the performance of the system and update the model and user interface as needed.

4.2 Data flow diagram:

A data flow diagram (DFD) is a graphical representation of how data flows through a system. It's a modelling technique used to describe the flow of data through a system, and it illustrates how data enters and leaves the system, how it's processed, and how it's stored.

DFDs are used in software engineering and systems analysis to help understand the flow of data in a system and to identify areas where data may be delayed, lost, or misused. They are particularly useful for modelling complex systems as they provide a clear, visual representation of the system's data flow.

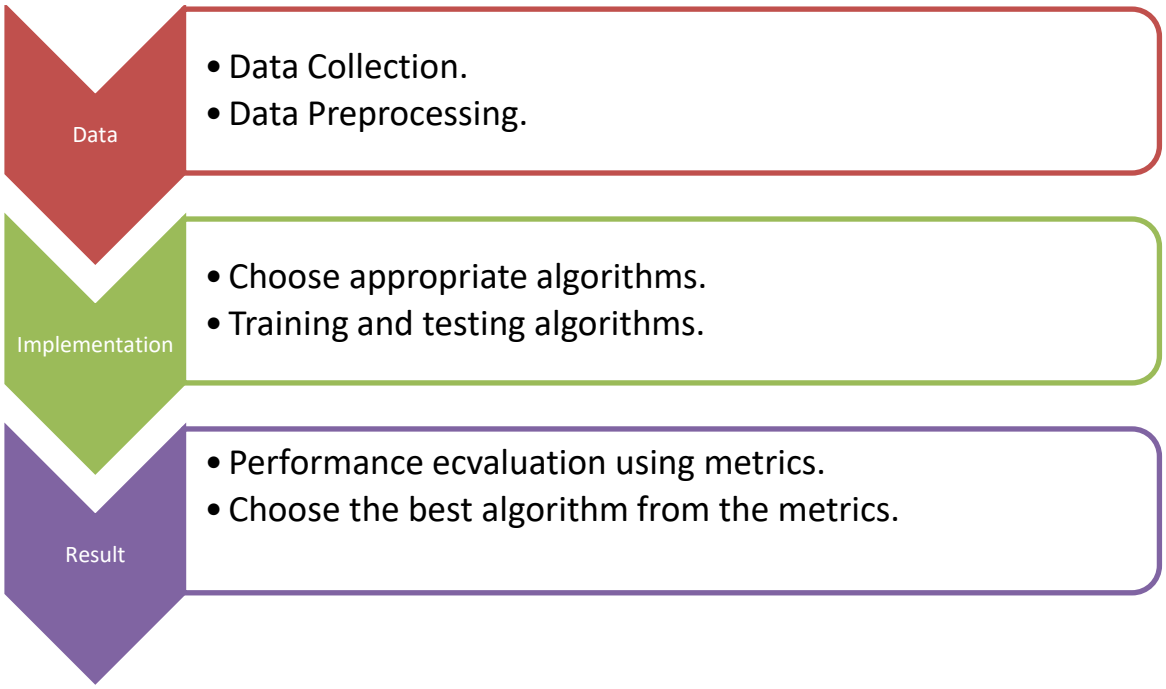


4.3 UML Diagrams:

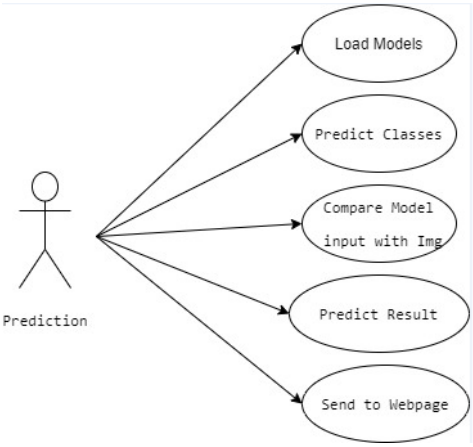
UML stands for Unified Modelling Language, and it's a standardized visual modelling language used in software engineering for designing and documenting software systems. UML provides a set of graphical notations for representing various aspects of a system's architecture, design, and behaviour.

Some of the most commonly used UML diagrams are:

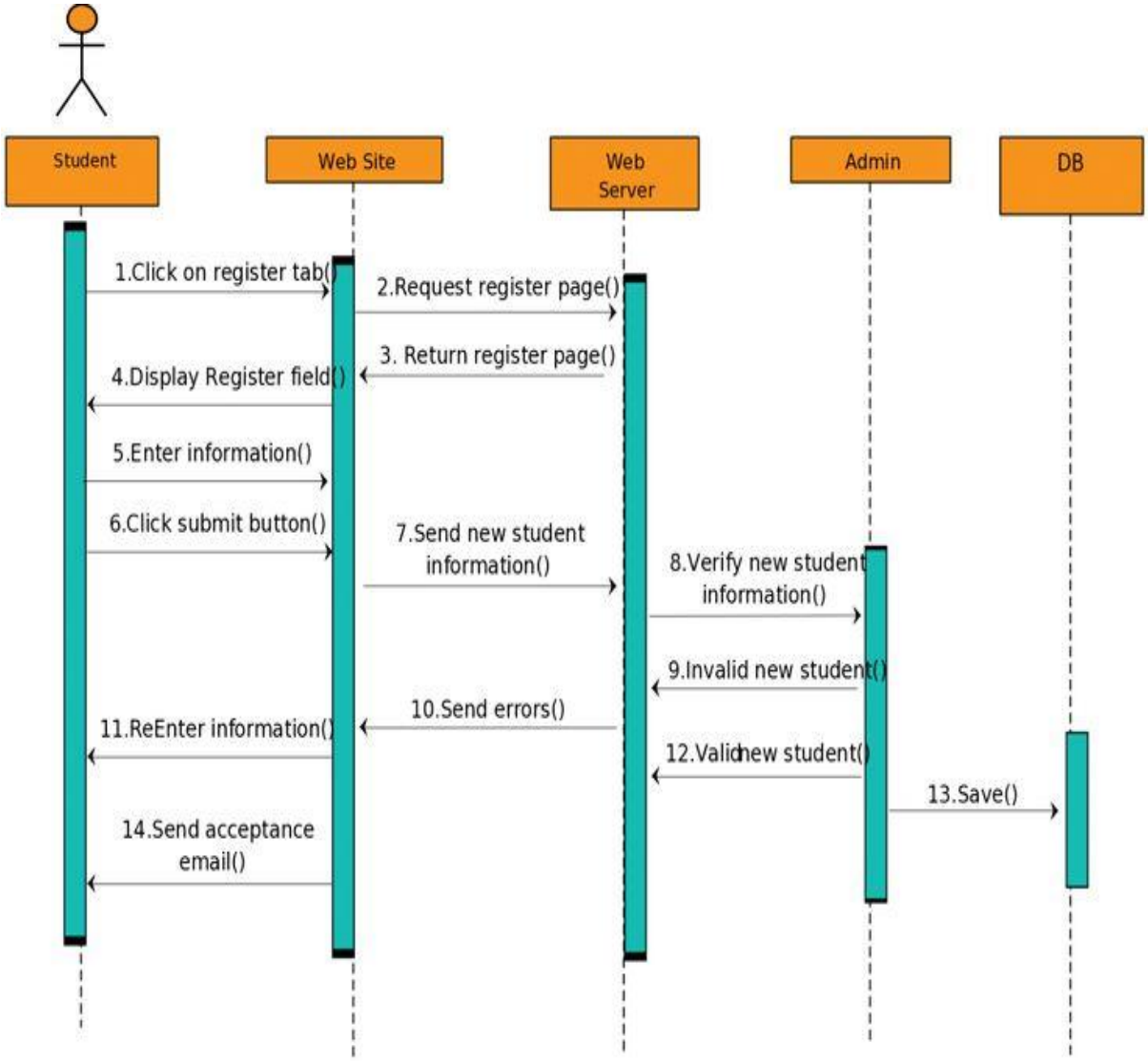
Class diagram: A class diagram is used to represent the static structure of a system, showing the classes and their relationships, attributes, methods, and associations.



Use case diagram: A use case diagram is used to represent the functional requirements of a system, showing the classes, use cases.



Sequence diagram: A sequence diagram is a type of UML (Unified Modelling Language) diagram that depicts the interactions between objects or components in a particular scenario. It shows the sequence of messages exchanged between the objects, along with the order and duration of these interactions.

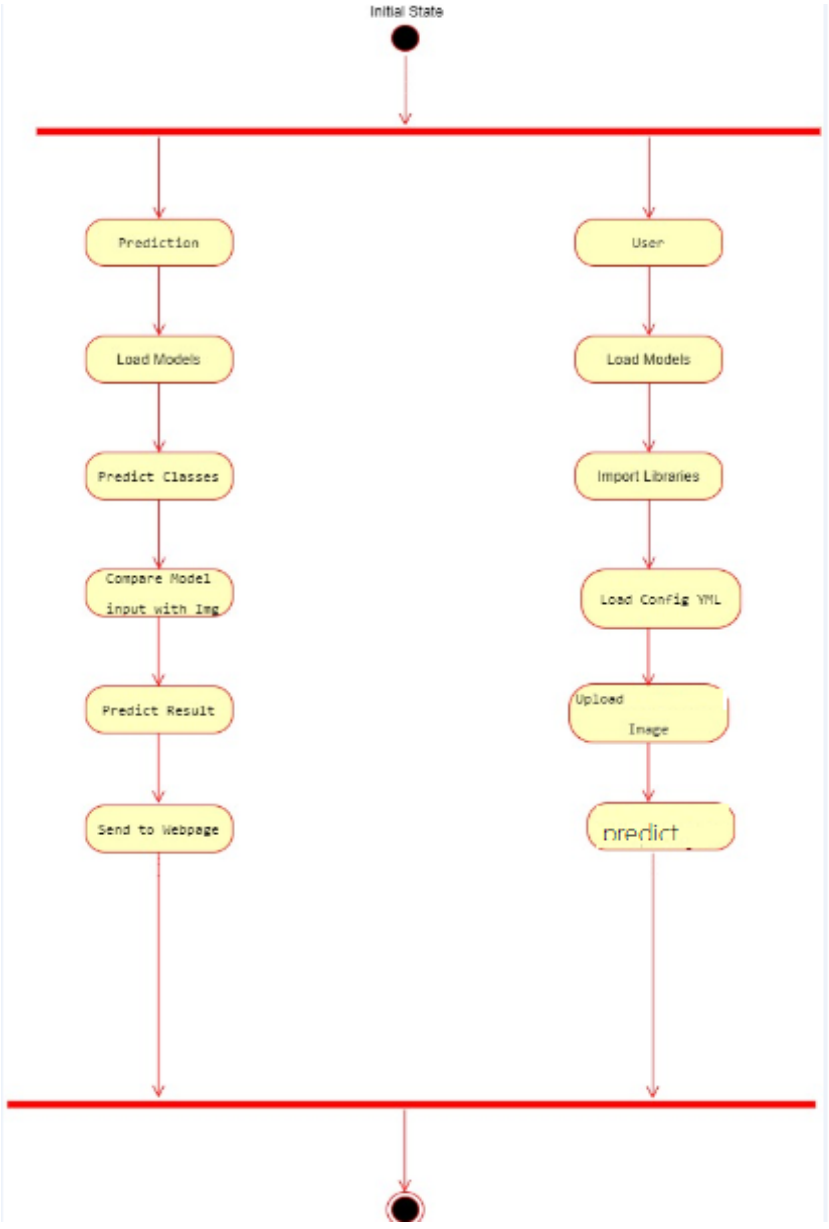


In a sequence diagram, objects or components are represented as lifelines, which are vertical lines that run down the diagram. Messages are represented as arrows that connect the lifelines and can be labelled to indicate the type of message being exchanged (such as a request or response).

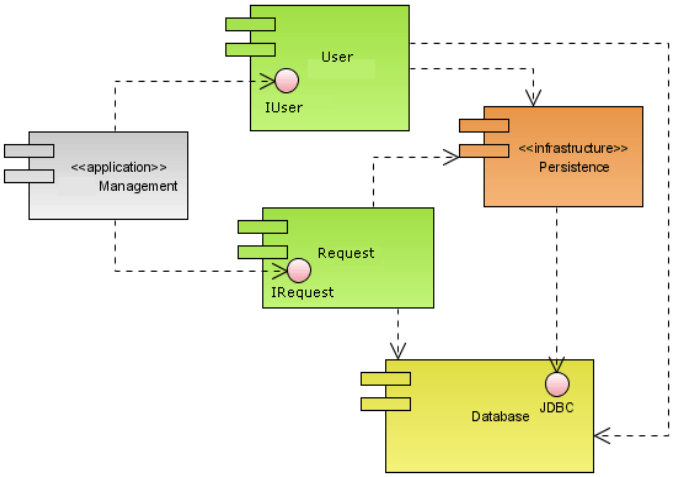
Sequence diagrams can also include other elements such as actors (external entities that interact with the system), gates (entry and exit points for messages), and conditions (such as loops or alternative flows).

Sequence diagrams are useful for visualizing and understanding the dynamic behavior of a system, as they provide a clear and concise representation of the sequence of events that occur during a specific scenario. They are often used in software design and development, as well as in systems analysis and modelling.

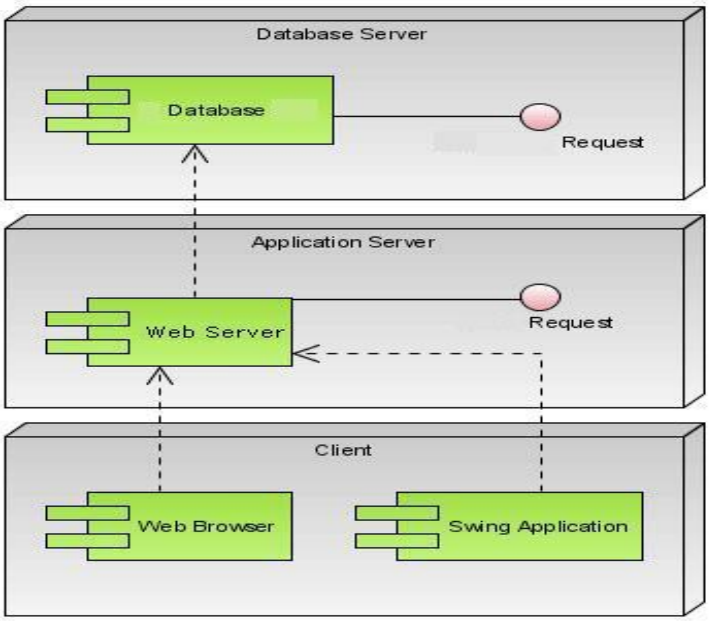
Activity diagram: An activity diagram is used to represent the workflow or business process of a system, showing the activities, transitions, and decisions involved.



Component Diagram: The framework's significant level parts are portrayed in the part graph. This delineation shows the framework's parts & their connections at an undeniable level. The parts that have been wiped out after the framework has gone through the turn of events or development stage are displayed in a component diagram.



Deployment Diagram: The setup of the application's runtime components is depicted in the deployment diagram. This graphic is unquestionably most helpful when a system is completed & prepared considering deployment.



UML diagrams are used by software developers, analysts, and designers to communicate ideas, clarify requirements, and validate designs before implementation. They help to ensure that everyone involved in a software project has a common understanding of the system's architecture and behaviour, which can reduce errors and misunderstandings and improve the overall quality of the software.

CHAPTER-5

IMPLEMENTATION

5.1: Language/Technology used for Implementation.

Python*:* Python is a high-level, interpreted programming language that is widely used in software development, data analysis, scientific computing, artificial intelligence, and web development.

Here are some of the key features of Python:

1. Simple and easy to learn: Python has a simple syntax and easy-to-understand code, making it easy for beginners to learn.
2. Open source: Python is an open-source language, which means that it's free to use and distribute, and developers can access the source code to customize and extend it.
3. Portable: Python code can run on various platforms, including Windows, macOS, Linux, and UNIX.
4. Large standard library: Python has a large standard library that includes modules for many common programming tasks, such as string processing, file I/O, and web services.
5. Dynamic typing: Python is dynamically typed, which means that the type of a variable is determined at runtime, rather than at compile-time.
6. Object-oriented: Python is an object-oriented language, which means that it supports classes and objects, encapsulation, inheritance, and polymorphism.
7. Versatile: Python can be used for a wide range of applications, including web development, data analysis, scientific computing, machine learning, and artificial intelligence.

Overall, Python is a versatile and powerful programming language that is widely used in many industries and applications. Its simplicity, ease of use, and large community of developers make it an attractive choice for many developers and businesses.

Machine Learning*:* Machine learning (ML) is a subfield of artificial intelligence (AI) that involves training computer systems to learn from data and make predictions or decisions without being explicitly programmed. ML algorithms use statistical techniques to identify patterns and relationships in data, and use these patterns to make predictions or decisions.

Some of the most common uses of machine learning include:

1. Image recognition: ML algorithms can be trained to recognize images and objects in photos, videos, and other visual data. This has applications in fields such as healthcare, self-driving cars, and security.
2. Natural language processing: ML algorithms can be used to analyze and understand natural language data, such as text or speech. This has applications in fields such as chatbots, virtual assistants, and sentiment analysis.

3. Predictive modelling: ML algorithms can be used to build predictive models that can forecast future outcomes based on historical data. This has applications in fields such as finance, marketing, and healthcare.
4. Recommendation systems: ML algorithms can be used to build recommendation systems that suggest products, services, or content based on user preferences and behaviour. This has applications in fields such as e-commerce, social media, and entertainment.
5. Fraud detection: ML algorithms can be used to detect fraudulent behaviour and transactions based on patterns in data. This has applications in fields such as finance, insurance, and e-commerce.

Overall, machine learning has a wide range of applications in various fields, and its use is only expected to grow in the coming years as more organizations recognize its potential to improve decision-making, increase efficiency, and drive innovation.

5.1.1: what is difference between Deep Learning and Machine Learning?

Machine learning (ML) and deep learning (DL) are two subfields of artificial intelligence (AI) that are used to create models that can learn from data.

The main difference between the two is that deep learning is a subset of machine learning that utilizes neural networks with multiple layers to learn and represent data. Deep learning is used to build more complex models that can handle larger and more complex datasets, such as image and speech recognition.

On the other hand, machine learning refers to the broader field of creating models that can learn from data without being explicitly programmed. Machine learning algorithms can be either supervised (where the algorithm is trained on labelled data), unsupervised (where the algorithm is trained on unlabelled data), or semi-supervised (where the algorithm is trained on a combination of labelled and unlabelled data).

Another difference between the two is the amount of data required to train the models. Deep learning models typically require a large amount of data to be trained effectively, while machine learning models can be trained on smaller datasets.

In summary, while both machine learning and deep learning are used to create models that can learn from data, deep learning is a subset of machine learning that is used to build more complex models using neural networks with multiple layers. Machine learning algorithms are more diverse and can be used for a variety of tasks, including both supervised and unsupervised learning, with less reliance on large datasets.

5.1.2: Why is Python used in Machine Learning (ML) over Java?

Python is currently the most popular language for machine learning (ML) due to several reasons:

1. Large community and rich ecosystem: Python has a large and active community of developers, scientists, and researchers who have contributed to the development of many powerful libraries and frameworks for ML, such as NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, and PyTorch.
2. Easy to learn and use: Python has a simple and intuitive syntax, which makes it easy for beginners to learn and use. It also has a large number of resources and tutorials available online, which makes it easier to get started with ML.
3. Fast prototyping: Python's interactive shell and dynamic nature make it easy to experiment with ML models and algorithms, which helps in fast prototyping and testing.
4. Flexibility and versatility: Python is a versatile language that can be used for a wide range of applications, including web development, data analysis, scientific computing, and AI. This makes it easier to integrate ML models into other applications.
5. Availability of pre-trained models: Python has a large number of pre-trained models and libraries available, which can be used for various ML tasks, such as image recognition, speech recognition, natural language processing, and more.

While Java can also be used for ML, Python's rich ecosystem, ease of use, and availability of pre-trained models make it a popular choice among developers and data scientists. However, Java may be preferred for certain types of ML projects where performance and scalability are critical.

5.2: Algorithms Implemented.

In this study, we used five different types of algorithms. They are:

1. **Logistic Regression (LR).**
2. **Support Vector Machine (SVM).**
3. **Decision Tree (DT).**
4. **Random Forest Classifier (RF).**
5. **Long-Short Term Memory (LSTM).**

The above five algorithms are used to train our models and also tested our models using the same.

Logistic Regression (LR): Logistic regression is a statistical method used to analyze the relationship between a dependent variable (binary or categorical) and one or more independent variables (continuous or categorical). It is a type of generalized linear model that uses a logistic function to model the probability of the dependent variable being a certain category.

- Logistic Function: g(z) = 1 / (1 + e^(-z)), where z = w^T x + b is the linear function of the model's weights w and bias b, and x is the input vector.
- Cross-Entropy Loss: L = -1/N * sum(y*log(y_hat) + (1-y)*log(1-y_hat)), where y is the true label and y_hat is the predicted probability of the positive class.

Applications: Logistic regression has a wide range of applications in various fields such as healthcare, finance, marketing, and social sciences. Some of the common applications include:

- Predicting the likelihood of a patient having a disease based on their medical history and other factors.
- Predicting the probability of a customer buying a product based on their demographic information and purchase history.
- Analysing the effect of different factors on customer churn in a business.
- Identifying the factors that influence student success in education.

Advantages:

- Logistic regression is a simple and efficient algorithm that can be trained on small datasets.
- It can handle both categorical and continuous independent variables.
- The model output is easily interpretable and can provide insights into the relationship between the dependent and independent variables.
- It can handle nonlinear relationships between the dependent and independent variables using techniques such as polynomial regression.

Disadvantages:

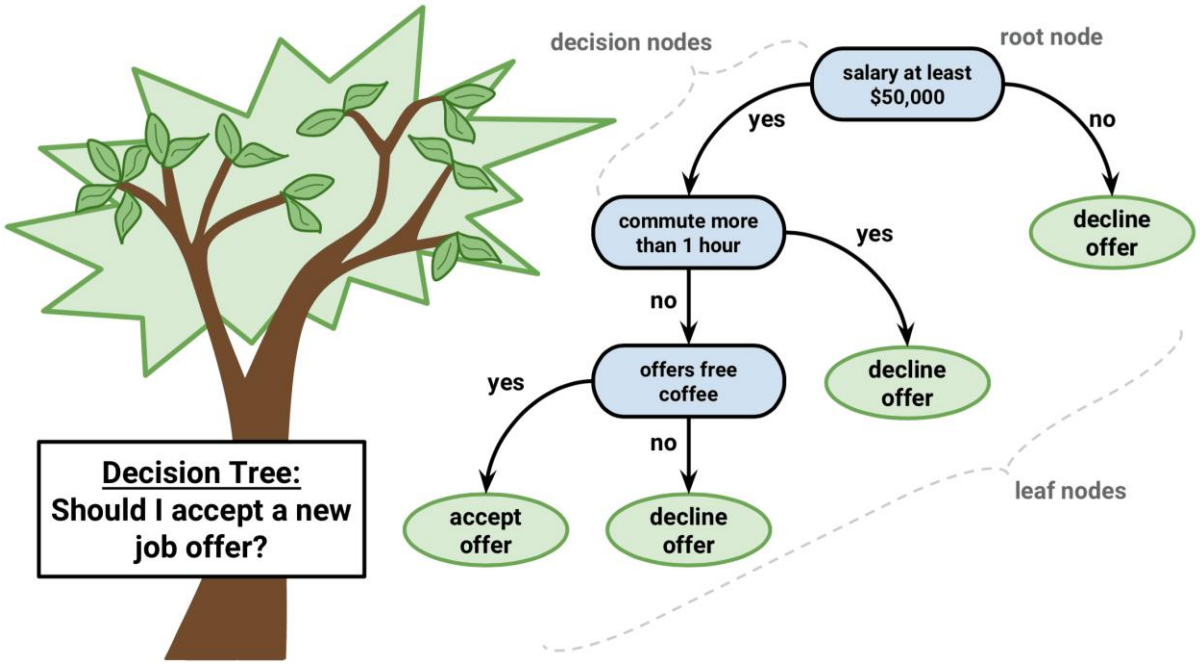
- Logistic regression assumes a linear relationship between the independent and dependent variables, which may not be suitable for some datasets.
- It may suffer from overfitting if the model is too complex, or the dataset is too small.
- It can only be used for binary or categorical dependent variables, not continuous variables.
- It requires that the observations are independent of each other, which may not be true in some datasets.

Decision Tree

A decision tree is a supervised machine learning algorithm that predicts a target variable based on a series of decision rules. It is constructed by recursively splitting the dataset into subsets based on the value of a feature, so that each subset is as homogeneous as possible with respect to the target variable. The resulting tree can be used to make predictions for new data points by following the path from the root node to a leaf node, where the leaf node corresponds to a specific prediction.

- Entropy: H(S) = -sum(p_i * log2(p_i)), where p_i is the proportion of examples in class i in the set S.

- Information Gain: IG(S, A) = H(S) - sum(|S_v| / |S| * H(S_v)), where A is a feature, S_v is the subset of examples in S with feature value v, and |S| is the total number of examples in S.



Applications:

- Credit risk assessment
- Medical diagnosis
- Customer segmentation
- Fraud detection
- Predictive maintenance

Advantages:

- Easy to interpret and visualize.
- Can handle both categorical and numerical data.
- Can handle missing data.
- Can handle nonlinear relationships.

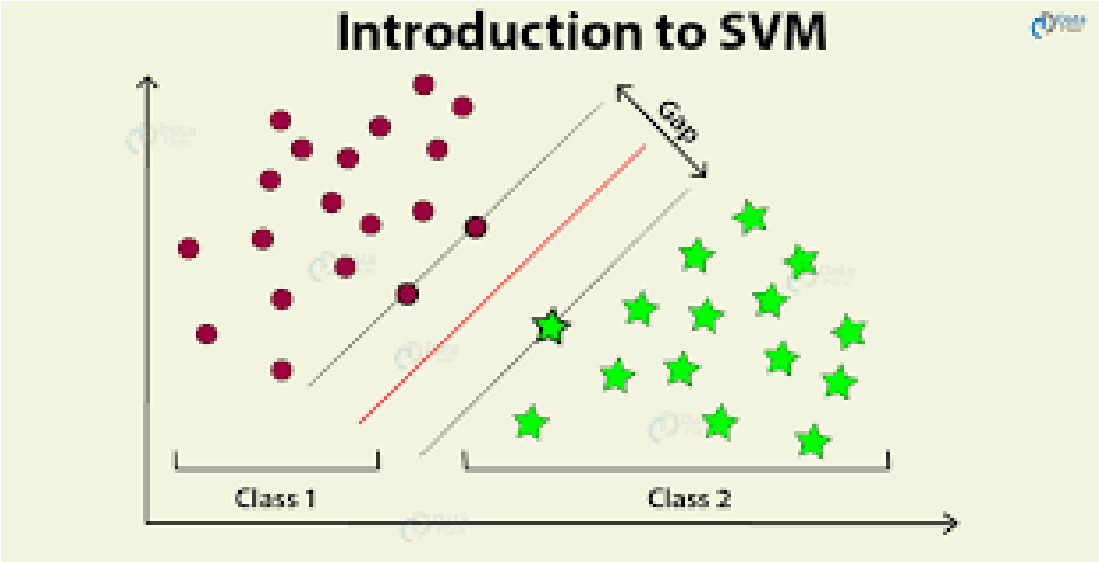
Disadvantages:

- Can be unstable and sensitive to small changes in the data.
- Can overfit the training data if the tree is too complex.
- Can be biased towards features with many levels or high cardinality.
- Cannot capture complex interactions between features.

Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm that finds the hyperplane that best separates the data into different classes. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the closest data points of each class. SVM can also use a kernel function to transform the data into a higher-dimensional space, where a linear hyperplane can separate the classes.

- Hyperplane equation: w^T x + b = 0, where w is the vector of weights and b is the bias term.
- Margin: distance between the hyperplane and the closest data points from both classes.
- Loss function: L = 1/2 * ||w||^2 + C * sum(max(0, 1 - y_i(w^T x_i + b))), where C is the regularization parameter and y_i is the label of the i-th example.



Applications:

- Image classification
- Text classification
- Speech recognition
- Fraud detection
- Bioinformatics

Advantages:

- Effective in high-dimensional spaces
- Can handle nonlinear relationships.
- Can handle small and noisy datasets.
- Provides a sparse solution (i.e., only a subset of the training data is used in the decision function)

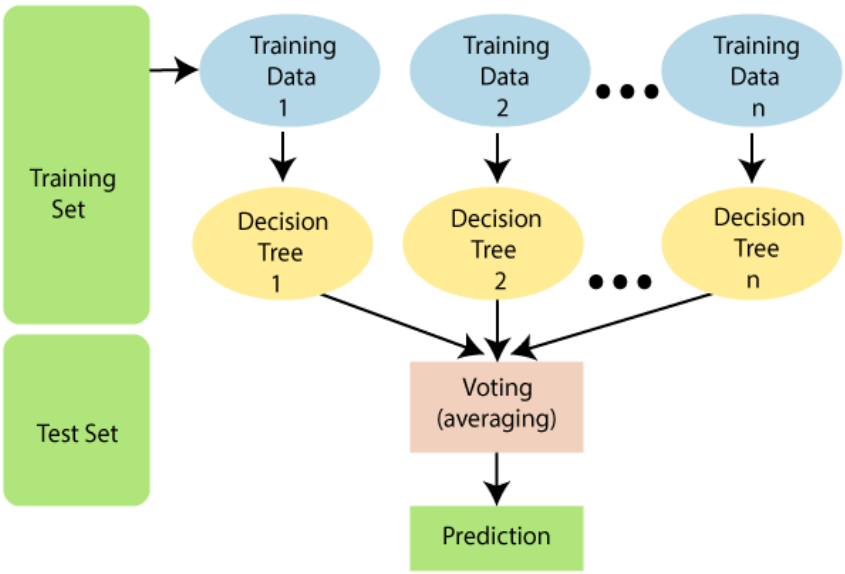
Disadvantages:

- Requires careful selection of the kernel function and its parameters.
- Can be sensitive to the scaling of the features.
- Can be computationally expensive for large datasets.
- Does not provide probabilistic outputs.

Random Forest

A Random Forest is an ensemble learning method that combines multiple decision trees to improve the predictive accuracy and reduce overfitting. Each tree in the forest is built on a bootstrap sample of the training data, and at each node of the tree, a random subset of features is considered for the split. The final prediction is the mode (for classification) or the mean (for regression) of the predictions of the individual trees.

- Bagging: randomly selecting subsets of the training data with replacement to train multiple decision trees.
- Voting: combining the predictions of multiple decision trees using majority voting or weighted voting.



Applications:

- Predictive maintenance
- Fraud detection
- Customer churn prediction
- Image classification
- Drug discovery

Advantages:

- Can handle both categorical and numerical data.
- Can handle missing data.
- Can handle nonlinear relationships.
- Can capture complex interactions between features.
- Provides an estimate of feature importance.

Disadvantages:

- Can be computationally expensive for large datasets and many trees.
- Can be biased towards features with many levels or high cardinality.
- Can overfit the training data if the trees are too complex.
- Can be difficult to interpret and visualize.

LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is designed to overcome the vanishing gradient problem and capture long-term dependencies in sequential data. It was introduced by Hochreiter and Schmidhuber in 1997 and has since been widely used in natural language processing, speech recognition, and other sequence modelling tasks.

LSTM works by using memory cells and gates to selectively forget or remember information over time, and to update the state of the cell based on the current input and the previous hidden state. The memory cell is a vector that stores the information over time, and the gates are vectors that control the flow of information into and out of the cell.

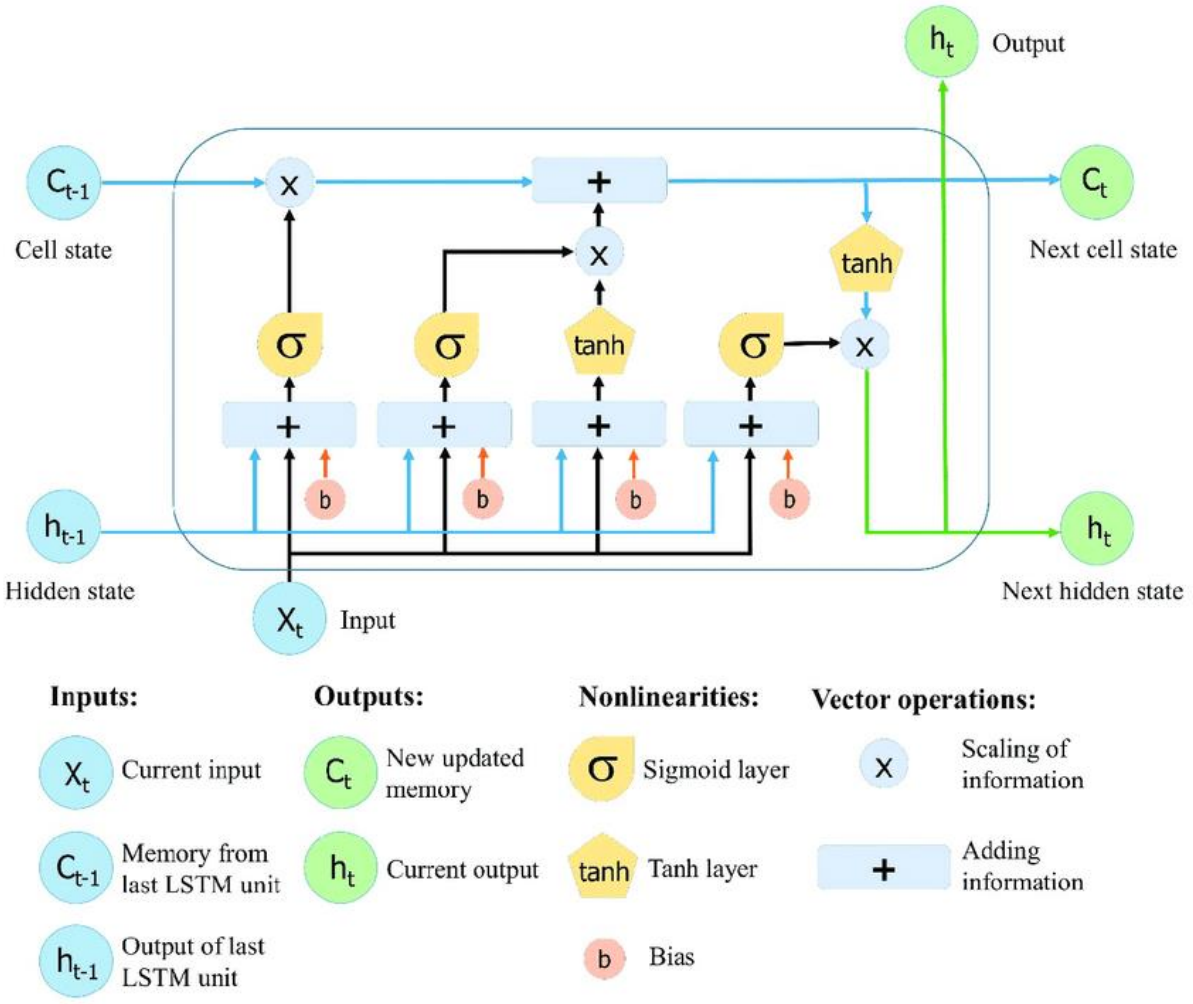
The LSTM unit has three gates:

- Forget Gate: This gate decides which information from the previous hidden state and the current input should be forgotten. It takes the previous hidden state h(t-1) and the current input x(t) as inputs and produces a vector f(t) between 0 and 1 for each element of the memory cell c(t-1). The value of f(t) determines which information from the previous cell state should be kept or discarded.
- Input Gate: This gate decides which new information from the current input and the previous hidden state should be added to the cell state. It takes the previous hidden state h(t-1) and the current input x(t) as inputs and produces a vector i(t) between 0 and 1 for each element of the memory cell c(t-1). The value of i(t) determines how much new information should be added to the cell state.
- Output Gate: This gate decides which information from the current cell state should be output to the next hidden state. It takes the current input x(t) and the previous hidden state h(t-1) as inputs and produces a vector o(t) between 0 and 1 for each element of the memory cell c(t). The value of o(t) determines which information from the current cell state should be output to the next hidden state.

The cell state is updated by:

- Forgetting the information that is no longer needed from the previous cell state using the forget gate.
- Adding new information to the cell state using the input gate.
- Updating the cell state by combining the information from the forget gate and the input gate.
- The hidden state is calculated by:
- Applying the output gate to the updated cell state.

LSTM can handle variable-length inputs and outputs and can capture long-term dependencies in sequential data, making it suitable for tasks such as language modelling, machine translation, and speech recognition. However, it can be computationally expensive for long sequences and large models.



LSTM cell: c_t = f_t * c_(t-1) + i_t * tanh(W_c x_t + U_c h_(t-1) + b_c), h_t = o_t * tanh(c_t), where c_t is the cell state, h_t is the hidden state, x_t is the input vector, and f_t, i_t, o_t are the forget, input, and output gates, respectively. W_c, U_c, b_c are the weight matrix, recurrent weight matrix, and bias term for the cell state

Applications:

- Speech recognition
- Language modelling
- Machine translation
- Video activity recognition
- Time-series prediction

Advantages:

- Can handle sequential and temporal data.
- Can capture long-term dependencies.
- Can handle variable-length inputs and outputs.
- Can handle input and output sequences of different lengths.

Disadvantages:

- Can be computationally expensive for long sequences and large models.

5.3: Code

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv(r"C:\Users\Sashank Sharma\Desktop\1060\7801.csv")
df.head()

df.columns = [each.strip() for each in df.columns]
df.columns

df.dtypes
```

```
#Histplots.

plt.hist(df['Total_Items'], bins=20)
plt.title('Distribution of Total Items')
plt.xlabel('Total_Items')
plt.ylabel('Frequency')
plt.show()
plt.hist(df['Time_Spent_ in Course'], bins=20)
plt.title('Distribution of Time Spent in Course')
plt.xlabel('Time_Spent_ in Course')
plt.ylabel('Frequency')
plt.show()
plt.hist(df['Total_Logins'], bins=20)
plt.title('Distribution of Total Logins')
plt.xlabel('Total_Logins')
plt.ylabel('Frequency')
plt.show()
plt.hist(df['Nbre_of_clics'], bins=20)
plt.title('Distribution of Nbre of clics')
plt.xlabel('Nbre_of_clics')
plt.ylabel('Frequency')
plt.show()
plt.hist(df['Nbre_of_ message_ participation'], bins=20)
plt.title('Distribution of Nbre of_ message  participation')
plt.xlabel('Nbre_of_ message_ participation')
plt.ylabel('Frequency')
plt.show()
plt.hist(df['Time_Spent_ in Course'], bins=10)
plt.title('Distribution of Time Spent in Course')
plt.xlabel('Time Spent in Course')
plt.ylabel('Frequency')
plt.show()
plt.scatter(df['Time_Spent_ in Course'], df['Total_Items'])
plt.title('Scatter plot of Time Spent in Course vs Total Items')
plt.xlabel('Time Spent in Course')
plt.ylabel('Total Items')
plt.show()
```

```
import seaborn as sns

sns.barplot(x='Level', y='Total_Logins', data=df)
plt.title('Mean Total Logins for each Level')
plt.show()

grouped_data = df.groupby('Level').sum()[['Time_Spent_on_Session_Attendence', 'Time_Spent_ in Course']]

# create pie chart for 'Time_Spent_on_Session_Attendence'
grouped_data['Time_Spent_on_Session_Attendence'].plot(kind='pie', autopct='%1.1f%%')
plt.title('Time Spent on Session Attendance by Level')
plt.ylabel('')
plt.show()

# create pie chart for 'Time_Spent_in_Course'
grouped_data['Time_Spent_ in Course'].plot(kind='pie', autopct='%1.1f%%')
plt.title('Time Spent in Course by Level')
plt.ylabel('')
plt.show()

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()


from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['Level'] =le.fit_transform(df['Level'])
df

X = df.drop('Level', axis=1)
y = df['Level']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) #test_train_split

# initialize and train the models
lr = LogisticRegression()
lr.fit(X_train, y_train)

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

rf = RandomForestClassifier()
rf.fit(X_train, y_train)

svm = SVC()
svm.fit(X_train, y_train)

# make predictions on test data
y_pred_lr = lr.predict(X_test)
y_pred_dt = dt.predict(X_test)
y_pred_rf = rf.predict(X_test)
y_pred_svm = svm.predict(X_test)

# evaluate model performance
acc_lr = accuracy_score(y_test, y_pred_lr)
acc_dt = accuracy_score(y_test, y_pred_dt)
acc_rf = accuracy_score(y_test, y_pred_rf)
acc_svm = accuracy_score(y_test, y_pred_svm)
```

```
print(f"Logistic Regression Accuracy: {acc_lr}")
print(f"Decision Tree Accuracy: {acc_dt}")
print(f"Random Forest Accuracy: {acc_rf}")
print(f"SVM Accuracy: {acc_svm}")

#Import Necessary libraries to run LSTM.

import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import plot_confusion_matrix
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten, MaxPooling1D
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from keras.layers import Input, Dense, LSTM, MaxPooling1D, Conv1D, TimeDistributed
from keras.layers import Dense, Flatten, Dropout
from keras.models import Model
from sklearn import metrics
import tensorflow as tf
from numpy import unique
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

labels = df['Level'] # target feature
df.drop(['Level'], axis = 1, inplace=True)

#Define Model
y = labels.to_numpy()
x = df.to_numpy()

print(x.shape)
x = x.reshape(x.shape[0], x.shape[1], 1)
print(x.shape)

Xtrain, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

print(Xtrain.shape, y_train.shape, X_test.shape, y_test.shape)

ytrain = tf.keras.utils.to_categorical(y_train)
ytest = tf.keras.utils.to_categorical(y_test)

n_timesteps, n_features, n_outputs = Xtrain.shape[1], Xtrain.shape[2], ytest.shape[1]


# define CNN model
feature_extractor = Sequential()
feature_extractor.add(Conv1D(filters=64, kernel_size=3, activation='tanh', input_shape=(n_timesteps,n_features)))
feature_extractor.add(Conv1D(filters=64, kernel_size=3, activation='tanh'))
feature_extractor.add(MaxPooling1D(pool_size=1))

# define LSTM model
model = Sequential()
model.add(feature_extractor)
model.add(LSTM(100))
model.add(Dense(50, activation='tanh'))
model.add(Dense(n_outputs, activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.build((n_timesteps,n_features))

model.summary()
print("Model Architecture Ready")

# Run the model and get accuracy
print("Model Training Started")
verbose, epochs, batch_size = 1, 200, 64 #32

history = model.fit(X_test, ytest, validation_split=0.3, batch_size=batch_size, epochs=epochs, verbose=verbose)
print("Model Training Completed")

acc = model.evaluate(X_test, ytest)
print("Model Evaluation Completed")

print("Loss:", acc[0], " Accuracy:", acc[1])
#Plot for accuracy.
algorithms = ['LSTM']
accuracy = 'acc'
print(acc)
plt.figure(figsize=[1,5])
plt.bar(algorithms, acc)
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
```

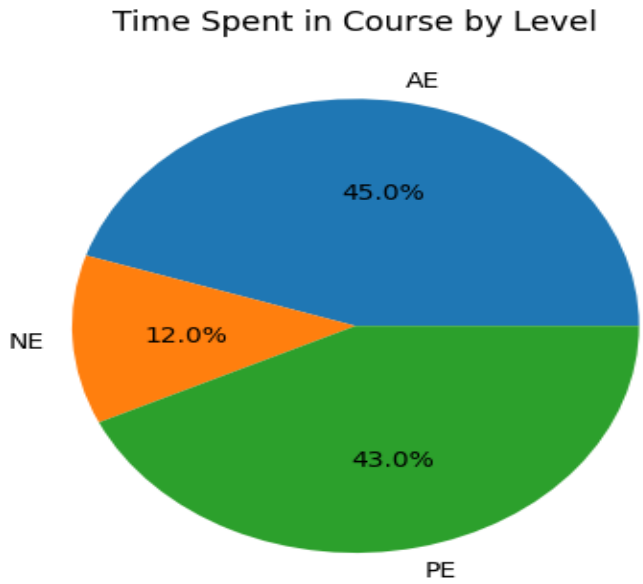
```
#Confusion Matrices for four algorithms.
#1.Logistic Regression.

import seaborn as sns

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred_lr, labels=lr.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=lr.classes_)
disp.plot()

#2.Decision Tree.

cm = confusion_matrix(y_test, y_pred_dt, labels=dt.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=dt.classes_)
disp.plot()

#Visualising the Decision tree.

from sklearn import tree
feature_names = ['Total_Items', 'Time_Spent_ in Course', 'Total_Logins',
      'Activity_inside_content_ area', 'User_Activity_in_Goup',
      'Nbre_of_clics', 'Nbre_of_ message_ participation', 'join_session',
      'Time_Spent_on_Session_Attendence', 'Level']
class_names = ['AE','PE','NE']
fig,axes = plt.subplots(nrows = 1, ncols=1,figsize = (4,4), dpi = 300)
tree.plot_tree(dt,feature_names = feature_names,class_names= class_names,filled= True);
```

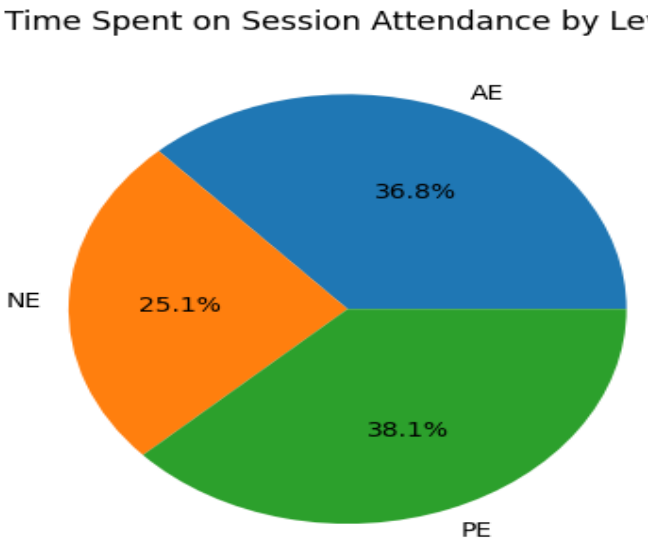
The above code is used to train and test our models. While we train all models simultaneously, we also save individual model separately in order to save them in the form of ".h5" for LSTM and ".pkl" for the rest of algorithms. So that we can load the individually trained models using joblib library during web deployment. In this Study, we used streamlit to do web deployment.

CHAPTER 6

RESULTS AND ANALYSIS

6.1 Experimental Results: The dataset that was downloaded the machine learning repository at UCI is used. to train and test our proposed system. The fractal dimension, texture, concavity, concave points, radius, perimeter, area, smoothness, compactness, and ten real valued attributes in the dataset. These characteristics are calculated from the student engagement parameters. The below Pie charts show the relation between the target and Two main student engagement parameters.





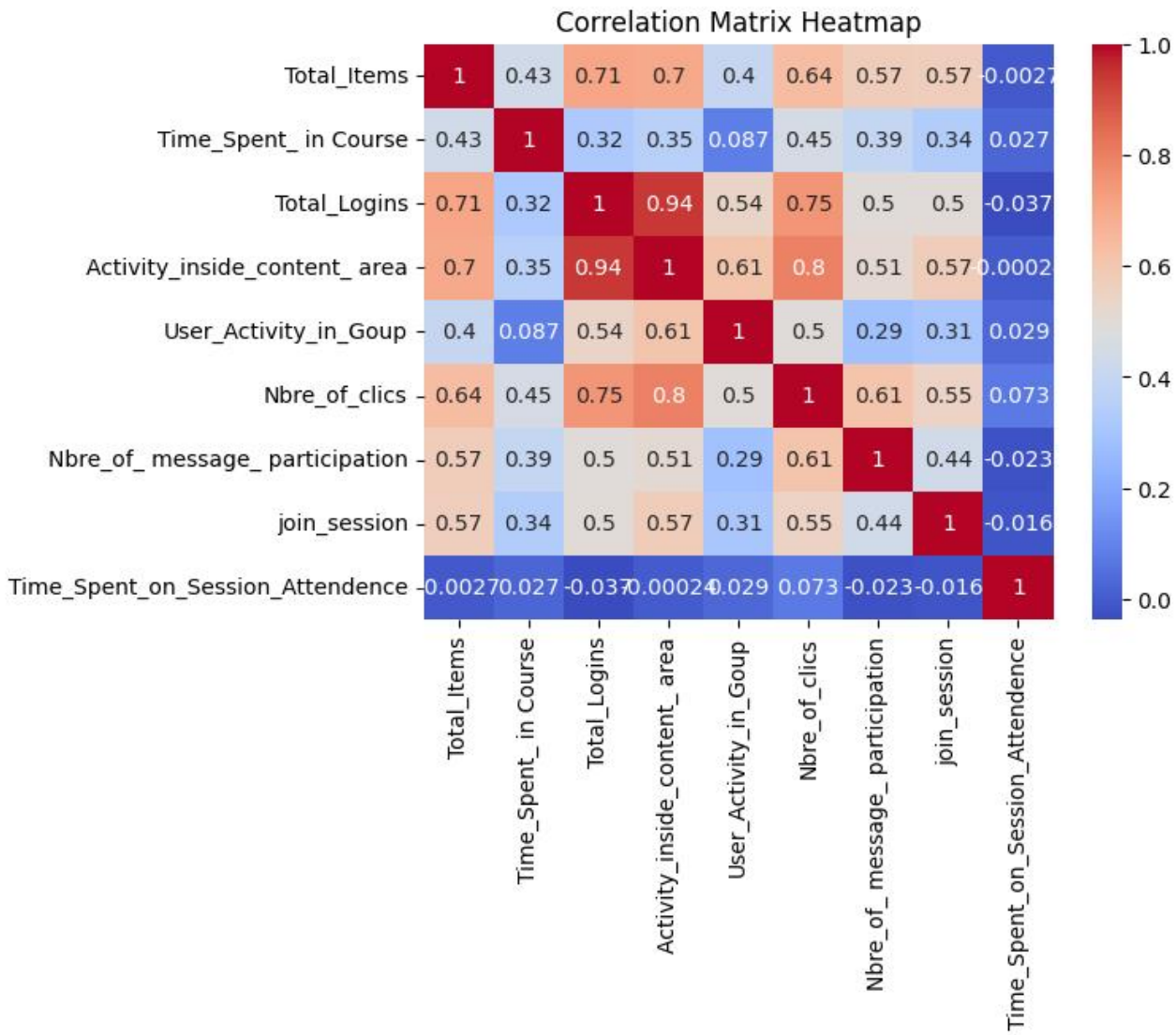
The Pie charts display the engagement level of students who spent time on courses and time in session for attendance.

Correlational matrix: A correlational matrix is a table that shows the correlations between multiple variables. In statistics, correlation refers to the degree to which two variables are related to each other. The correlational matrix displays the correlation coefficients, which are numerical values that indicate the strength and direction of the relationship between each pair of variables.

In a correlational matrix, the variables are listed in both the rows and columns of the table. The diagonal of the matrix shows the correlation of each variable with itself, which is always 1. The upper triangle of the matrix shows the correlations between pairs of variables, while the lower triangle shows the same values but mirrored.

A correlational matrix is useful in data analysis because it allows researchers to quickly see the relationships between multiple variables. It can help to identify patterns or trends in the data, as well as potential areas of further study. By examining the correlations between variables, researchers can gain insights into how different factors are related and how they might influence each other.

The correlation between each parameter is displayed by the following correlational matrix.



Confusion Matrix: A confusion matrix is a table that is used to evaluate the performance of a classification model. It shows the number of correct and incorrect predictions made by the model, compared to the actual outcomes.

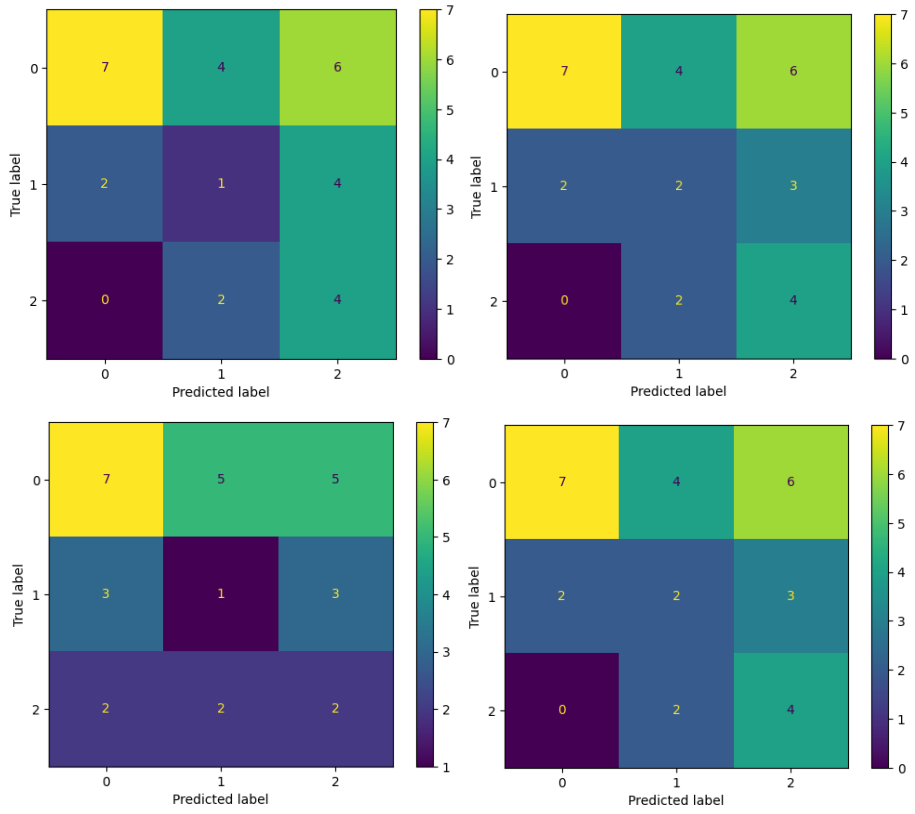
A confusion matrix typically consists of four values:

- True Positives (TP): The number of correct predictions that the model made for the positive class.
- False Positives (FP): The number of incorrect predictions that the model made for the positive class.
- True Negatives (TN): The number of correct predictions that the model made for the negative class.
- False Negatives (FN): The number of incorrect predictions that the model made for the negative class.

The confusion matrix is arranged in a table with the actual values (positive or negative) listed in the rows, and the predicted values (positive or negative) listed in the columns.

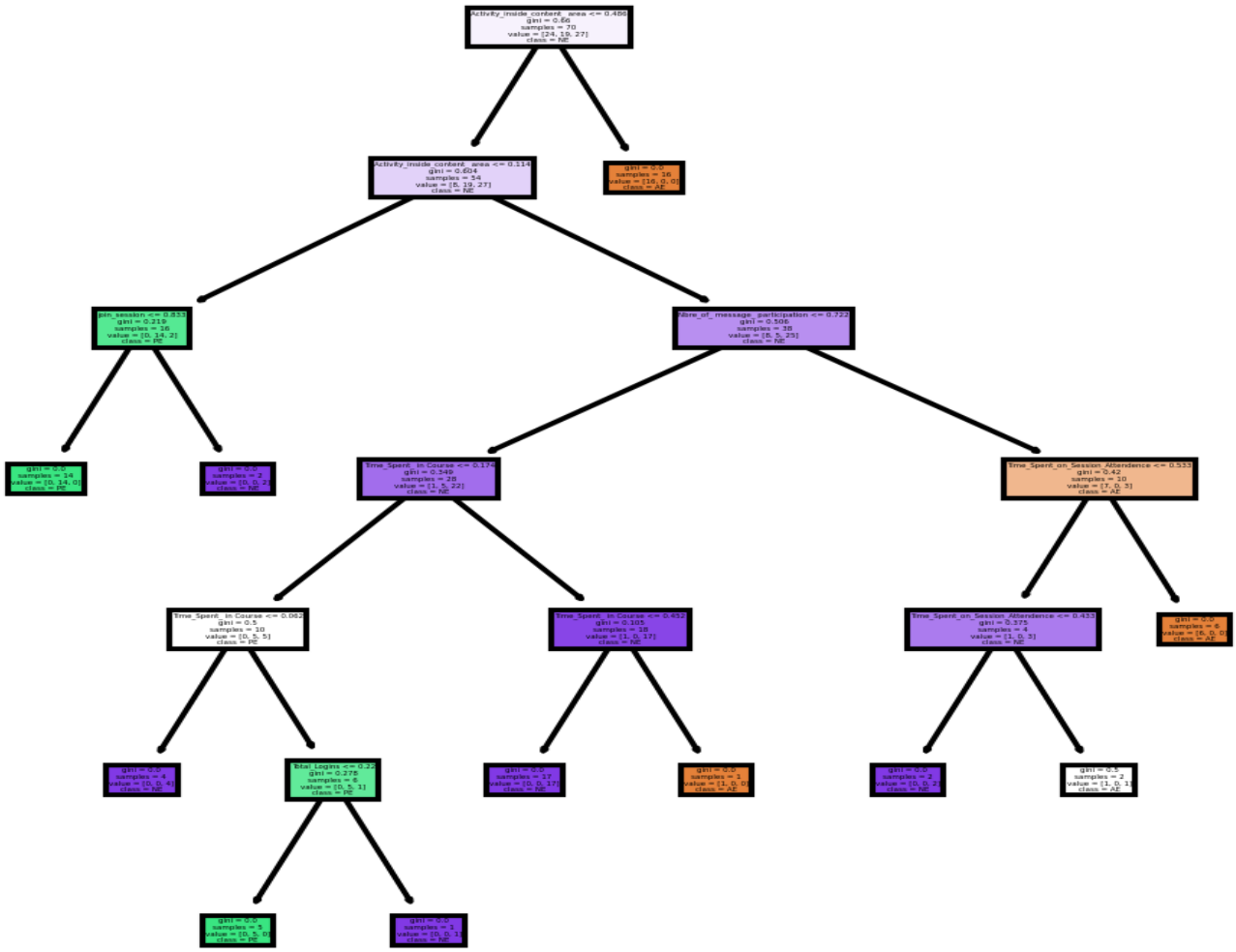
The information in a confusion matrix can be used to calculate several performance metrics for a classification model, such as accuracy, precision, recall, and F1 score. These metrics can help to evaluate the effectiveness of the model and identify areas for improvement.

Overall, a confusion matrix provides a useful way to visually represent the performance of a classification model and gain insights into its strengths and weaknesses.



The above plots for confusion matrix can be plotted using seaborn library. This will help us to understand Metrics and can calculate them from the pictures.

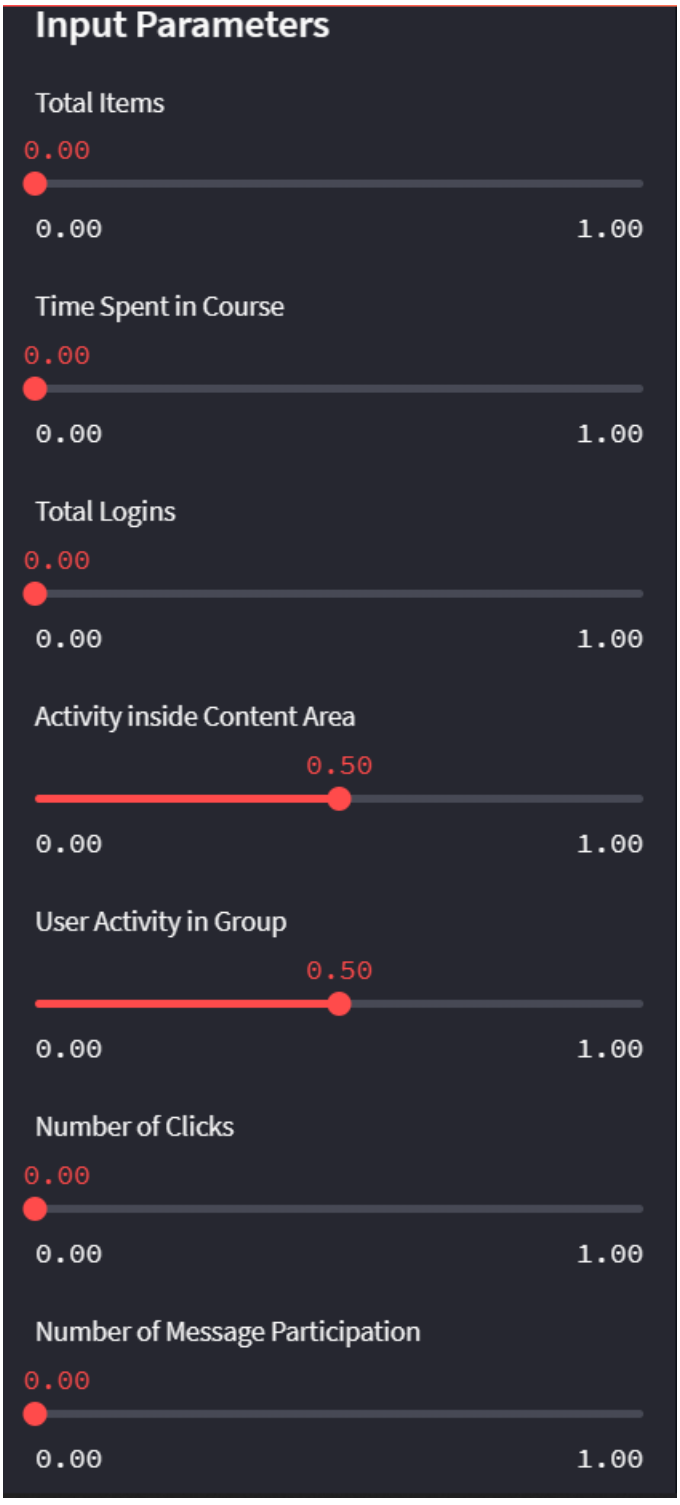
The decision Tree algorithm help us to get a clear overview of the project.



The above figures of correlational matrix and the confusion matrix help us to understand the salient features of the data.

S. No	Salient Features	Importance
1	Total Logins	0310349
2	Activity inside content area	0.249501
3	Nbr. clicks	0.137109
4	Join session	0.100937
5	User Activity group	0.074397
6	Total items	0.034525
7	Time spent	0.031743
8	Time Spent session attendance	0.016438

Input.


Input Parameters
Total Items
0.00
0.00
1.00
Time Spent in Course
0.00
0.00
1.00
Total Logins
0.00
0.00
1.00
Activity inside Content Area
0.50
0.00
1.00
User Activity in Group
0.50
0.00
1.00
Number of Clicks
0.00
0.00
1.00
Number of Message Participation
0.00
0.00
1.00

Output:

CHAPTER-7

CONCLUSION

This study's main focus has been on predicting student involvement using the information that students learned during an online session. Among the four models in this investigation, the LSTM algorithm has the highest accuracy. However, the DT and Random Forest algorithms are employed to identify the key data characteristics associated with the engagement elements. The overall percentile of test achievement, the amount of time spent on each session, and the knowledge levels of individual students are used to determine student involvement. The suggested system can help educational institutions when their ability to provide instruction is disrupted, whether it's because of a COVID-19-like pandemic or another form of natural disaster that forces them to close for a while.

Future studies will examine aspects of time associated with student assessment, mentor presence, and effectiveness of online sessions.

CHAPTER-8

REFERENCES

[1] M. H. Baturay, "An Overview of the World of MOOCs," *Procedia Soc Behav Sci*, vol. 174, pp. 427–433, Feb. 2015,

https://doi.org/10.1016/j.sbspro.2015.01.685

[2] I. Maiz, "Research on MOOCs: Trends and Methodologies."

[3] Robert M. Carini and George D. Kuh, "Student engagement and student learning: Testing the linkages," *Res High Educ*, vol. 47, no. 1, Feb. 2006.

[4] M. D. Dixson, "Measuring Student Engagement in the Online Course: The Online Student Engagement Scale (OSE)."

[5] J. S. Lee, "The relationship between student engagement and academic performance: Is it a myth or reality?" *Journal of Educational Research*, vol. 107, no. 3, pp. 177–185, May 2014, https://doi.org/10.1080/00220671.2013.807491.

[6] J. VanderPlas, "Python Data Science Handbook: Essential Tools for Working with Data."

[7] M. Atherton, M. Shah, J. Vazquez, Z. Griffiths, B. Jackson, and C. Burgess, "Using learning analytics to assess student engagement and academic outcomes in open access enabling programmes," *Open Learning*, vol. 32, no. 2, pp. 119–136, May 2017,

https://doi.org/10.1080/02680513.2017.1309646

[8] Mutahi, "Studying Engagement and Performance with Learning Technology in an African Classroom," *Association for Computing Machinery*, pp. 148–152, Mar. 2017.

[9] Timothy Rodgers, "Student engagement in the e-learning process and the impact on their grades," *International Journal of Cyber Society and Education*, vol. 1, no. 2, Dec. 2008, Accessed: Jan. 18, 2023. [Online].

https://www.learntechlib.org/p/209167

[10] Zhuang, "Understanding Engagement through Search Behaviour," *Association for Computing Machinery*, pp. 1957–1966, Nov. 2017,

https://doi.org/10.1145/3132847.3132978

CHAPTER-9

BIBLIOGRAPHY

- Python Technologies
- Python Complete Reference
- Python Script Programming by Yehuda Shirin
- Mastering python Security python Professional by Shadab Siddiqui
- Python server pages by Larne Pekowsley
- Python Server pages by Nick Todd
- HTML HTML Black Book by Holzer
- SQLite