Department of Computer Science and Engineering

CHITKARA UNIVERSITY
Baddi (Himachal Pradesh) (INDIA)

A PROJECT REPORT

ON

Predict Students' Dropout and Academic Success

Submitted by

Akshit Rai (2011981289)

Amit Aswal (2011980501)

Krish Tapwal (2011981212)

Sagar Mishra (2011980524)

Supervised By

Mr. Shivam Singh

Declaration

We, the undersigned individuals involved in the creation and compilation of this project report, titled "Predict Students' Dropout and Academic Success," solemnly affirm the authenticity, originality, and integrity of the work presented herein.

The content of this report is a result of our independent research, analysis, and scholarly efforts. We assert that all information, findings, and conclusions contained within are genuine and have been meticulously referenced, acknowledging any and all sources of external knowledge or data used in accordance with academic standards.

Furthermore, we declare that this report has not been previously submitted, in part or whole, for any other academic or professional qualification. It stands as a unique and original piece of scholarly work crafted by us, and it abides by the ethical principles and academic integrity expected within our educational institution.

We acknowledge that any contributions from external sources, whether direct quotations, paraphrased information, or borrowed concepts, have been duly cited and referenced following established academic conventions.

By signing this declaration, we affirm our responsibility for the authenticity, accuracy, and originality of the content presented in this project report.

Name of the Student:
Amit Aswal
Akshit Rai
Krish Tapwal
Sagar Mishra

Place: Baddi, Himachal Pradesh

Date: 20/11/2023

Acknowledgement

We express our deepest appreciation to Mr. Shivam Singh, whose dedicated mentorship, expert guidance, and unwavering support steered us through the intricacies of this project. His insightful feedback, encouragement during challenging moments, and commitment to excellence have been pivotal in shaping the course of our research and refining our methodologies.

Our gratitude extends to the esteemed faculty and staff of the Department of Computer Science and Engineering at Chitkara University. Their scholarly wisdom, continuous support, and encouragement fostered an intellectually stimulating environment, propelling our academic growth and providing the necessary resources and infrastructure for this endeavour.

Furthermore, we extend heartfelt thanks to our peers and colleagues whose collaborative spirit, thought-provoking discussions, and constructive criticism were invaluable. Their diverse perspectives and shared enthusiasm significantly contributed to the evolution of our ideas and the refinement of our project.

In addition, we acknowledge the contributions of researchers, scholars, and institutions whose seminal work and comprehensive literature served as guiding beacons, enriching our understanding and providing the foundation upon which this project was built.

Moreover, our profound appreciation goes to our families and friends for their unwavering support, understanding, and encouragement. Their patience, belief in our aspirations, and unwavering support were fundamental in sustaining us through the demanding phases of this academic pursuit.

This project would not have been feasible without the collective support, guidance, and encouragement from these individuals and institutions. Their contributions, whether in mentoring, scholarly resources, or personal support, have been invaluable in shaping this research endeavour and our academic journey as a whole.

Abstract

Within the realm of higher education, this project endeavours to harness the potential of machine learning methodologies to prognosticate not only the trajectories of students' academic success but also anticipate potential dropout occurrences. Through a comprehensive approach encompassing rigorous data analysis, iterative model development, and meticulous evaluation techniques, the primary objective is to deploy predictive models capable of identifying crucial indicators influencing academic outcomes.

This endeavour doesn't solely rest on predictive analytics; its broader mission encompasses the facilitation of proactive interventions and targeted support mechanisms for students at risk of academic underachievement or potential dropout. By leveraging the power of advanced data analytics, the project seeks to unearth patterns, correlations, and predictive markers that can revolutionize educational strategies, ensuring a more inclusive and supportive environment that caters to diverse learning needs.

Furthermore, the overarching goal extends beyond predictive prowess; it aspires to foster an enriched learning ecosystem. The insights gleaned from this research endeavour aim to serve as a compass for educational institutions, guiding the formulation of adaptive pedagogical strategies and personalized interventions that bolster student engagement, resilience, and overall academic success.

In essence, this project is a multidimensional exploration, amalgamating cutting-edge technological applications with a human-centric approach, all in service of optimizing the educational landscape and empowering students to thrive in their academic pursuits.

Table of Contents

Contents

Chapter 1

INTRODUCTION

In the ever-evolving landscape of higher education, the quest for predictive methodologies to anticipate and address students' academic trajectories has gained paramount importance. The ability to foresee potential challenges, identify pivotal markers for success, and mitigate risks of academic underperformance or dropout has become a pressing concern for educational institutions worldwide. In response to this imperative, this project embarks on a journey to harness the potential of machine learning techniques in deciphering the intricate tapestry of students' academic journeys.

The focal point of this endeavour revolves around the predictive analysis of students' academic success and the identification of indicators that might forewarn of potential dropout scenarios. By amalgamating robust data analysis with advanced machine learning algorithms, the aim is to construct predictive models capable of delineating trajectories that lead to academic accomplishment or potential pitfalls.

This project is not merely confined to the realm of predictive analytics. It harbours a broader vision—to serve as a catalyst for proactive interventions and tailored support mechanisms for students deemed at risk of academic disenchantment. Through the identification of predictive markers, this initiative aspires to provide a roadmap for educational institutions to implement targeted interventions, fostering a nurturing environment conducive to holistic student development.

Furthermore, this project endeavours to bridge the gap between technological advancements and human-centric pedagogy. It strives to harness the power of data-driven insights not only to predict academic outcomes but also to inform adaptable and inclusive educational strategies. The aim is to pave the way for an educational landscape that champions individualized learning experiences, amplifies student engagement, and fortifies pathways to academic success.

1.1 Motivation

Amidst concerns of student attrition and academic performance, this project seeks to harness the potential of machine learning to proactively identify patterns and markers indicative of students' academic trajectories. The goal is to empower educational institutions with predictive insights to intervene effectively and support students at risk of underperformance or potential dropout, thereby fostering a more resilient learning ecosystem.

1.2 Problem Statement

The challenge at hand involves developing predictive models capable of discerning critical factors influencing students' academic journeys. By leveraging machine learning techniques, the project aims to identify these influential markers, facilitating early intervention strategies for at-risk students and enhancing overall academic retention rates.

1.3 Objectives

This project endeavours to construct robust predictive models using machine learning algorithms to forecast students' academic success and pre-empt potential dropout. Through data-driven insights, the primary objective is to equip educational institutions with actionable information, enabling them to implement tailored interventions and adaptive strategies that nurture student progress and ensure higher retention rates.

1.4 Summary

Driven by the imperative to enhance student retention and academic success, this project dives into the realm of machine learning to foresee academic trajectories. By unravelling predictive insights, it aims to empower educational institutions with the foresight to deploy proactive interventions, ultimately fostering a more supportive environment for students to thrive in higher education.

Chapter 2

METHODOLOGY

2.1 Problem Formulation and Scope

We address the urgency of predicting students' performance and reducing dropout rates in educational institutions. The project involves classifying students into three categories based on historical data.

2.2 Data Collection and Preparation

Multiple data sources, including academic records, demographic details, socio-economic factors, and past student outcomes, are systematically collected and integrated. This phase involves comprehensive data cleaning, addressing inconsistencies, handling missing values, and performing feature engineering to extract meaningful insights. The dataset is transformed into a structured format suitable for analysis

2.3 Exploratory Data Analysis (EDA)

EDA delves deeply into the dataset, employing advanced visualization techniques, statistical analyses, and correlation studies. Through heatmaps, histograms, and scatter plots, the objective is to identify trends, outliers, and interdependencies among variables. Insights gained during EDA aid in defining influential features for subsequent modelling.

2.4 Feature Selection and Engineering

Utilizing a combination of statistical methods, including correlation matrices and recursive feature elimination essential variables influencing academic outcomes are identified. Feature engineering techniques such as scaling, binning, or creating composite features further refine the dataset to enhance model performance.

2.5 Model Development and Optimization

An array of machine learning algorithms—Logistic Regression, Support Vector Machines (SVM), Gradient Boosting, Neural Networks—is explored and implemented. Model development involves iterative training and validation, leveraging techniques like grid search and cross-validation to optimize hyperparameters for each algorithm.

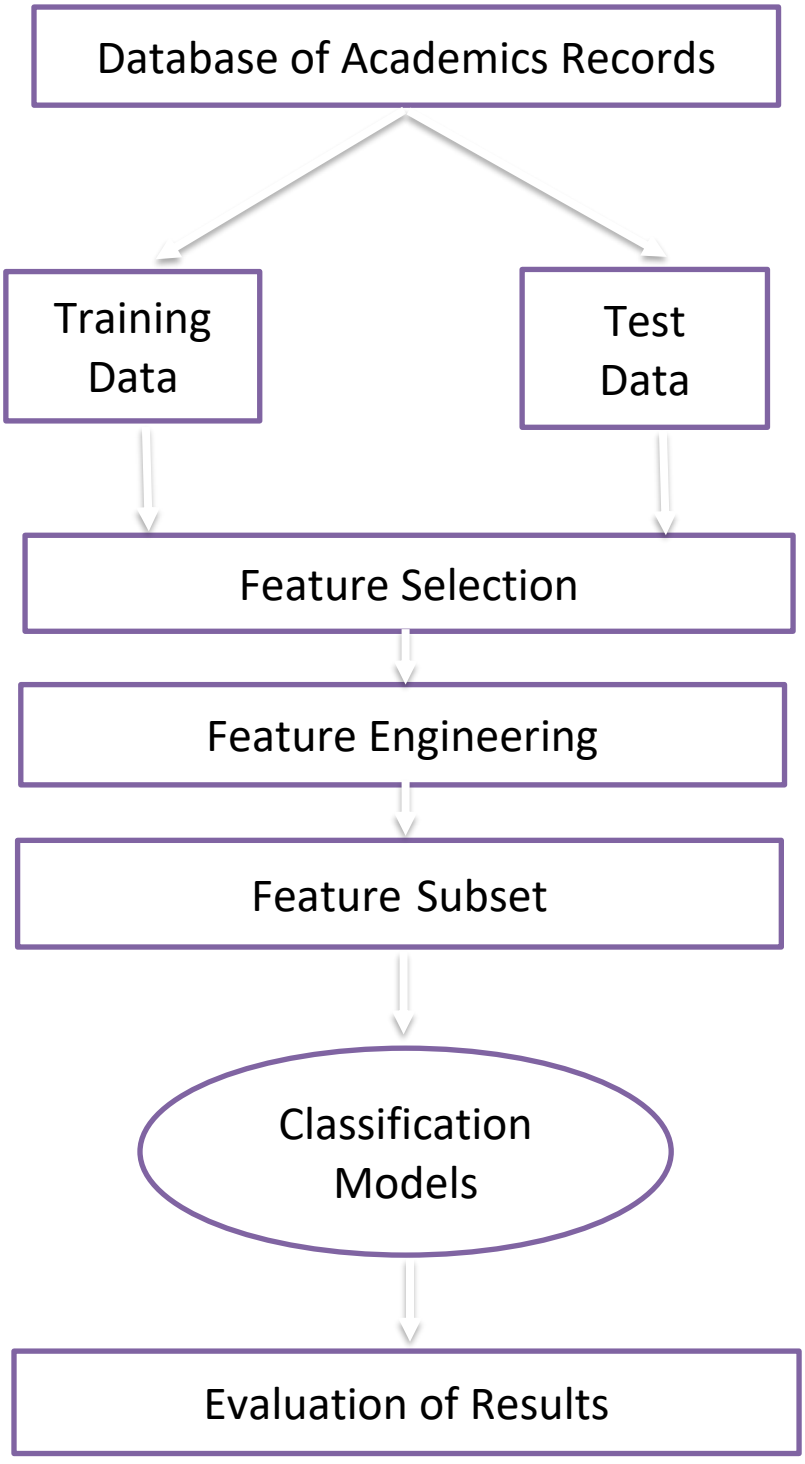
2.6 Insights and Recommendations

Interpreting model outcomes yields actionable insights for educational institutions. These insights guide the formulation of targeted strategies—early intervention programs, personalized mentoring, resource allocation—for supporting at-risk students and bolstering overall student success rates.

2.7 Future Directions and Enhancements

The methodology's adaptable framework lays the groundwork for ongoing enhancements. Future directions may include incorporating real-time data streams, integrating advanced predictive techniques like ensemble learning or deep learning, and developing interactive dashboards for continuous monitoring and intervention optimization.

2.7 System Model



2.8 Summary

Very often, developers repeat the steps of developing a model and evaluating it to compare the performance of different algorithms. Comparison results help to choose the appropriate ML algorithm most relevant to the problem.

The five algorithm we have implemented in this project are XGBoost, Support Vector Machine (SVM), Random Forest, Logistic Regression, and Decision Tree so we have taken the better algorithm which gives us more accuracy.

Chapter 3

SYSTEM REQUIREMENTS

3.1 Introduction

Requirement analysis is an essential part of product development. It plays an important role in determining the feasibility of an application. Requirement analysis mainly consists of software requirements, hardware requirements, and functional requirements.

Software requirements: This mainly refers to the needs that solve the end-user problems using the software. The activities involved in determining the software requirements are:

1. **Elicitation**: This refers to gathering information from the end users and customers.
2. **Analysis**: Analysis refers to logically understanding the customer needs and reaching a more precise understanding of the customer requirements.
3. **Specification**: Specification involves storing the requirements in the form of use cases, user stories, functional requirements, and visual analysis.
4. **Validation**: Validation involves verifying the requirements that have been specified.
5. **Management**: During the course of development, requirements constantly change, and these requirements have to be tested and updated accordingly.

Hardware requirements: Hardware requirements are the physical aspects that are needed for an application. All the software must be integrated with hardware to complete the development process. The hardware requirements that are taken into account are:

1. Processor Cores and Threads
2. GPU Processing Power
3. Memory
4. Secondary Storage
5. Network Connectivity

3.2 Software and Hardware Requirements

SOFTWARE REQUIREMENTS: Listed below are the software requirements for performing time series analysis on the fraud data:

1. **Operating System:** Operating system acts as the interface between the user programs and the kernel. Windows 8 and above (64 bit) operating system is required.
2. **Anaconda**: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda.
3. **Jupyter Notebook**: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.
4. **Data Set**: The dataset contains 4424 record with target variable having value “Dropout” and “Success”.

HARDWARE REQUIREMENTS

1. **Processor:** Intel i5 2.5 Ghz upto 3.5Ghz (or AMD equivalent)
2. **GPU (preferred)**: dedicated GPU from NVIDIA or AMD with 4GB VRAM
3. **Memory**: minimum 8GB RAM
4. **Secondary Storage**: minimum 128GB SSD or HDD
5. **Network Connectivity**: bandwidth ~ 10 Mbps 3 75 Mbps

3.2 Functional and Non-Functional Requirements

FUNCTIONAL REQUIREMENTS: Functional requirements are those which represent the functions of the system. Functional requirements may involve calculations, data manipulation, technical details and data processing. The functional requirements are:

1. **Data Pre-processing**: This process involves cleaning, transforming and reducing data to convert raw data in a useful manner.
2. **Training**: Initially, the system has to train based on the data set given. The training period is when the system learns how to perform the required task based on the inputs given through the dataset.
3. **Forecasting**: Forecasting is the process of making predictions of the future based on past and present data and most commonly by analysis of trends.
4. **Evaluation**: To gauge the efficacy of the system in predicting dropout instances and academic success, the predicted outcomes are subjected to an evaluation process. The model-generated predictions for dropout and academic success are validated against known data or ground truth. This validation allows for an assessment of the model's accuracy in correctly identifying instances of dropout and predicting academic success.

NON-FUNCTIONAL REQUIREMENTS: Non-functional requirements are the specifics that can be used to measure the operation of the system. These include:

1. **Accuracy**: This refers to the number of correct outputs to the total number of outputs.
2. **Openness**: The system must be guaranteed to work efficiently for a certain period of time.
3. **Portability**: The system must be designed in a way which is independent of the platform on which it is run. This makes it possible to run on many systems without making a lot of changes.
4. **Reliability**: The system has to produce fast and accurate results.

4.4 Summary

Requirement analysis is an essential part of product development. It plays an important role in determining the feasibility of an application. By analyzing the different requirements for a system, a concrete plan of action can be designed. The software and hardware requirements can limit the system if not carefully listed down. They have to be compatible with each other to complete integration of the system to deliver the final product. These requirements are a quantitative measure of the system. The user demand is met by breaking down the high-level tasks to requirements, which help in the task of system designing providing clear goals.

The functional and non-functional requirements help measure the system operations. The functional requirements describe the operations that a system has to perform. These can include tasks such as pre-processing, data extraction and evaluation. The non-functional requirements perform the task of measuring of how well the system executes these operations. It assesses the system based on how reliable, accurate and user friendly it is.

Requirement analysis is hence an important task before the start of any project. It allows the developer to find out the feasibility level and complexity of the project. It further helps in building an execution plan for the project to proceed along.

Chapter 4

System Design

4.1 Introduction

System Design is the way towards portraying the components of a framework, for instance, interfaces, algorithms, UML diagrams and information sources or databases utilized for a framework to fulfill determined prerequisites. It is characterized so as to fulfill the necessities and prerequisites of a business or association through the building of a lucid and well-running framework.

Frauds are known to be dynamic and have no patterns, hence they are not easy to identify. Fraudsters use recent technological advancements to their advantage. They somehow bypass security checks, leading to the loss of millions of dollars. Analyzing and detecting unusual activities using data mining techniques is one way of tracing fraudulent transactions. More and more companies are looking to invest in machine learning to improve their services. Machine learning is a combination of various computer algorithms and statistical modeling to allow the computer to perform tasks without hard coding. The acquired model would be learning from the <training data=. Predictions can be made or actions can be performed from stored experiential knowledge.

4.2 Proposed System

The proposed system comprises three primary stages: data acquisition, classification of academic outcomes, and the calculation of predictive indicators using a mobile agent. Initially, the system captures speech signals through the phone's microphone, transmitting them to the academic outcome prediction server. The queried data undergoes classification based on confidence levels and probabilities, with the classification results relayed back to the mobile agent, reflecting the likelihood of dropout or academic success.

Each signal from both the sets is pre-processed to make it suitable for data gathering and analysis.

1. **Sampling**

The initial step involves converting continuous academic performance data into discrete, computationally manageable signals. This transition from continuous to discrete signals facilitates efficient processing in the computational environment.

2**. Feature Extraction**

Academic records often contain diverse data components, including attendance, grades, participation, and other relevant metrics. Feature extraction aims to highlight and emphasize the significant components while disregarding redundant or less informative data segments.

3. **Normalization**

To ensure uniformity and equitable influence of different features in the model, normalization standardizes the extracted features within a consistent range, preventing any individual feature from dominating the model's learning process due to its scale.

4. **Data Cleaning (Silencing)**

Academic datasets might contain sections devoid of meaningful information, such as incomplete records, missing grades, or irregularities. This step involves cleaning the dataset by identifying and eliminating such non-informative or incomplete sections, ensuring a robust dataset for analysis.

5. **Segmentation** (Framing)

To facilitate the analysis and evaluation of academic performance on a more granular level, the dataset is segmented into smaller, discrete segments or frames. This segmentation allows for the examination of academic trends within specific time intervals or academic periods.

6. **Windowing**

Academic data often presents an extensive and complex range of information that cannot be processed comprehensively at once. Windowing partitions the data into manageable sections, aiding in statistical analysis and enabling a focused examination of specific academic patterns or trends.

4.3 Data Flow Diagram

4.3.1 Description

A data flow (DFD) is a graphical representation of the <flow= of data through information through information system, modeling its prospects. DFDs are also used for the visualization of data processing. A DFD shows that what kind of information will be input to the system and output from the system, where the data will come from and go to, and where the data will be stored. It does not show about the timing of the processors, or information about whether processes will operate in sequence or in parallel. Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

4.3.2 Uses of DFD's

- Data flow diagram is useful for establishing the boundary of the business or system domain (the sphere of analysis activity) under investigation.
- It identifies external entities along with their data interfaces that interact with the processes of interest.
- A DFD can be useful tool for helping secure stake holder agreement (sign3off) on the project scope.
- It is also a useful tool for breaking down a process into its sub processes for closer analysis.
- It helps in the logical information flow of the system.
- It determines of physical system construction requirements.
- Simplicity of notation.
- It establishes the manual and automated systems requirements.

4.3.3 Levels of Abstraction

In order to fashion an accurate picture of anything, which need to employ various levels of abstraction. This need arises from the basic relationship between an object and an observer, or a subject and its student. There are qualitative changes, as one move from one level of abstraction to the next, and seemingly a progression of from and/or formative principle.
In Software engineering DFD (data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see mainly 2 levels in data flow diagram, which are: 0-level DFD and 1-level DFD

Level 0:

Level 0 is also called a Context Diagram. It is a basic overview of the whole system or process being analyzed or modeled. It is designed to be an at-a-glance view showing the system as a single high-level process with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

Figure 4.1 Level 0 Dataflow Diagram

Fig 4.1 describes the overall process of the project. We input the important dataset file data and preprocess the data before prediction using different classifier to predict the type of result.

Level 1:

This context-level data flow diagram (DFD) is next explored, to produce a level 1data flow diagram (DFD) that shows some of the detail of the system being modeled. The level 1 data flow diagram (DFD) shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provides all functionality of the system as a whole. It also identifies the internal data stores that must be presenting order for the system to do its job, and flow of data between the various inputs of the system.

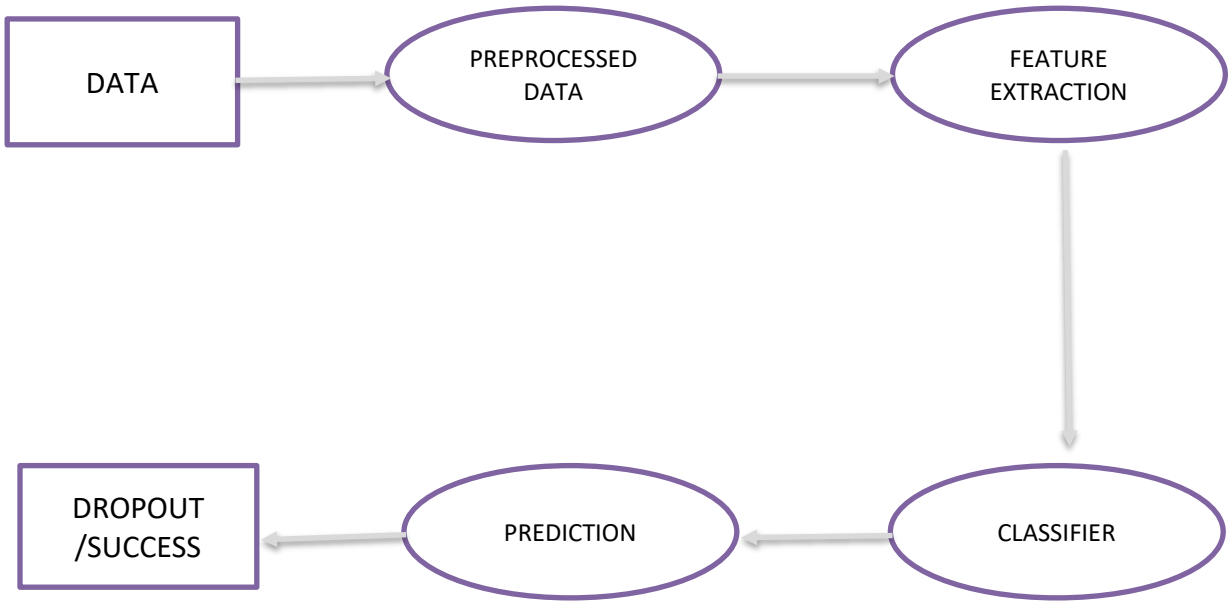


Figure 4.2 Level 1 Dataflow Diagram

Figure 4.2 illustrates the extraction of preprocessed academic data, incorporating crucial features such as attendance records, course grades, participation metrics, and other pertinent indicators. These extracted features, including academic performance indicators and behavioral metrics, play a pivotal role in determining student outcomes, aiding in the classification of academic success or dropout using advanced classifiers such as Random Forest and Gradient Boosting. By leveraging these key features and employing sophisticated machine learning models, the system can predict the likelihood of academic success or potential drop.

4.4 Summary

This chapter gives a brief introduction to the system design process, its methodologies requires for developing a system. It deals with different types of design processes used in real world and system architectures. Proposed model mainly describes the how exactly the system works. Data flow diagram for the proposed model is designed in three different levels of abstraction. Dataflow diagram (DFD9s) offers a graphical representation for summarizing the movement of data flows in the different levels of processes. It is mainly discussed about what is the proposed system that is implementing with that of the existing system. It describes the how the complexity is reduced, reduced in the cost and about the performance of the system.

Chapter 5

IMPLEMENTATION

5.1 Introduction

Implementation is akin to translating a blueprint of ideas and concepts into a language that computers can comprehend and execute. It's the tangible realization of various plans, models, designs, or policies. At its core lies computer programming, a comprehensive process involving designing, coding, testing, debugging, and maintaining software source code written in programming languages. The ultimate goal is to craft programs that demonstrate specific behaviors or functionalities. This process demands a wide array of expertise, encompassing not just coding skills but also a deep understanding of diverse domains, specialized algorithms, and formal logic.

In the realm of systems design, which applies systems theory to the realm of product development, there's a convergence with fields such as systems analysis, architecture, and engineering. Architectural design, for instance, zooms in on defining the structure, behavior, and various perspectives of a system. Meanwhile, the logical design abstractly represents data flows, inputs, and outputs through models that offer a high-level conceptual view of the system.

This is where physical design comes into play, focusing on the practical implementation of these abstract models by detailing the actual input and output mechanisms of the system. It dives into the specifics of how data is received, verified, processed, and finally displayed or utilized. This holistic approach ensures a thorough understanding and execution of both theoretical frameworks and practical applications within the development process.

5.2 Methodology

Language used: Python

Python is an interpreted, object-oriented, and high-level programming language renowned for its dynamic semantics. Its appeal lies in its built-in high-level data structures, coupled with dynamic typing and binding, making it a preferred choice for rapid application development and scripting purposes. Python's simplicity and readability, evident in its easy-to-learn syntax, significantly reduce the overheads associated with program maintenance.

The language's support for modules and packages fosters a modular approach to programming, promoting code reuse and facilitating program organization. Python's interpreter, along with its extensive standard library, is freely accessible in source or binary form across major platforms, allowing seamless distribution.

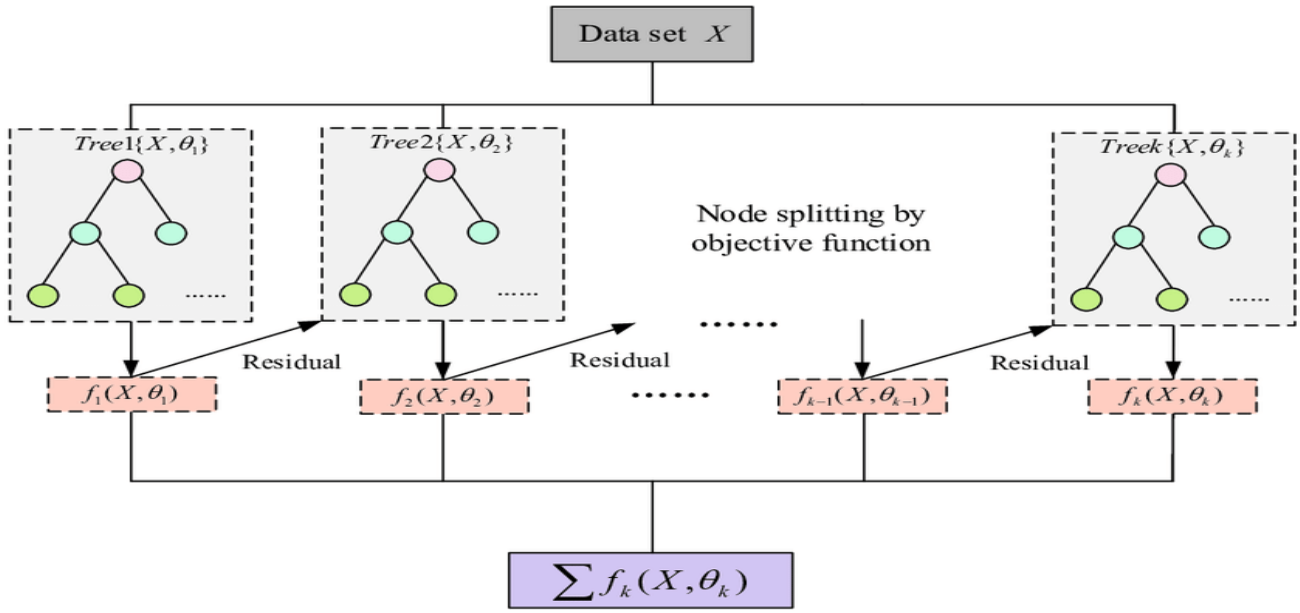
One of Python's standout features is its propensity to enhance programmer productivity. Owing to its interpretive nature, the edit-test-debug cycle is notably swift as there's no compilation step. Additionally, debugging Python programs is relatively straightforward; errors or flawed inputs won't lead to system crashes, mitigating the risk of segmentation faults. These attributes contribute to Python's allure among developers seeking a balance between efficiency, ease of use, and reliability in their programming endeavours.

GUI used: Jupyter Notebook

Jupyter is a dynamic project and collaborative community aimed at fostering the development of open-source software, open standards, and interactive computing tools spanning various programming languages. At the forefront of this initiative is Jupyter Notebook, a web-based, open-source application empowering data scientists to craft and distribute documents amalgamating live code, mathematical equations, computational results, visual representations, and diverse multimedia elements within a unified interface. This versatile platform allows for the creation of cohesive narratives that seamlessly weave together explanatory text with interactive code and rich visualizations, enhancing the communication and exploration of data-driven insights.

5.2 Algorithm

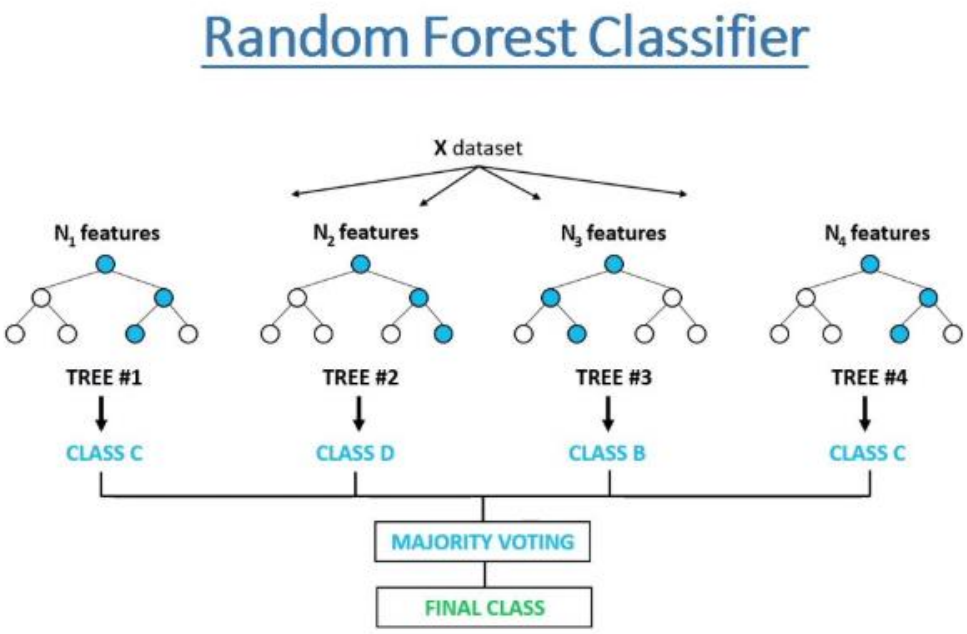
5.2.1 XG BOOST



XGBoost, or eXtreme Gradient Boosting, is an ensemble learning algorithm renowned for its exceptional speed, scalability, and performance in diverse machine learning tasks. It operates within a gradient boosting framework, utilizing an ensemble of decision trees as its base learners. What sets XGBoost apart is its innovative algorithmic enhancements and optimization techniques, making it highly efficient even with large-scale datasets.

This algorithm stands out due to its ability to handle both linear and tree-based models, blending them to reduce predictive errors effectively. Its parallel computing capabilities leverage hardware resources efficiently, accelerating training times significantly. XGBoost is widely favored in machine learning competitions and real-world applications for its exceptional predictive accuracy and robustness across various domains, including classification, regression, and ranking tasks.

5.2.2 Random Forest



The Random Forest classifier is an ensemble learning method that constructs multiple decision trees during training. It combines predictions from these trees to determine the final output - for classification tasks, it selects the most commonly predicted class among the trees, and for regression, it averages the individual tree predictions. Widely used for its robustness, scalability, and ability to handle various data types, the Random Forest classifier is favored in numerous domains for classification and regression tasks.

Key features:

Ensemble Technique: Utilizes multiple decision trees for prediction.

Tree Diversity: Trees are built using random subsets of data and features, reducing overfitting.

Aggregation: Combines tree predictions for robust final output.

Robustness: Resilient against overfitting and noisy data due to diverse tree training.

Feature Importance: Provides insights into the importance of different features.

Scalability: Efficiently handles large datasets and is parallelizable for faster computation.

5.3 Feature Extraction

Certainly! Here's the revised feature extraction process without bold formatting:

Feature Extraction Process

Academic data represents multifaceted information showcasing student performance and behavioral patterns. Extracting meaningful features from this data involves recognizing key indicators pivotal in predicting academic outcomes.

Understanding Academic Signals: Academic records encompass a diverse range of factors influenced by student behavior, attendance, course grades, and participation. These signals fluctuate based on a student's engagement and performance within an educational environment.

Significance of Features: Selecting features crucial in predicting academic success or dropout is imperative. These features should exhibit high informativeness and relevance to the context of academic performance. Identifying features that strongly correlate with potential dropout indicators or academic achievements enhances the predictive power of the model.

Consistency and Reliability: It's essential to ensure the selected features demonstrate consistency across various student samples. Features portraying consistent trends across different academic scenarios provide more reliable insights into potential outcomes.

Preprocessing and Engineering: The initial feature selection might yield a raw feature vector that demands refinement. Feature engineering involves handling outliers, addressing missing or null values, and standardizing the feature values, preparing them for effective model training and analysis.

In essence, the feature extraction process for predicting academic outcomes involves identifying, selecting, and processing key indicators from academic datasets, aiming to unveil informative patterns that significantly contribute to predicting student success or potential dropout scenarios.

5.4 Packages/Libraries Used

Pandas: A powerful data manipulation and analysis library in Python that offers data structures and operations for manipulating numerical tables and time series data efficiently.

Numpy: Fundamental package for scientific computing in Python, providing support for multidimensional arrays, mathematical functions, linear algebra, and random number capabilities.

Matplotlib: A versatile plotting library for creating static, interactive, and publication-quality visualizations in Python, offering a wide range of plotting functionalities.

Seaborn: Built on top of Matplotlib, Seaborn provides a high-level interface for creating attractive and informative statistical graphics, simplifying the creation of complex visualizations.

Pickle: Python module used for serializing and deserializing Python objects, essential for saving and loading machine learning models.

Dython: A library offering tools for assessing associations and correlation in categorical data, providing additional statistical functions for categorical data analysis.

Scikit-learn (sklearn): A comprehensive machine learning library offering various algorithms for classification, regression, clustering, dimensionality reduction, and model evaluation.

Warnings: Python module used to handle warning messages, allowing control over how warnings are displayed and handled during program execution.

5.5 Summary

Implementation is the stage of the project when the theoretical design is turned out into a working system. The implementation stage involves careful planning, investigation of the existing system and the constraints on implementation, designing of methods to achieve change over and evaluation of changeover methods.

It also seems that the XG Boost Algorithm performs better than Other algorithm during the implementation and so we go ahead with XG Boost testing where a real time data will be inputted result will be predicted and will be 88% accurate

Chapter 6

System Testing

6.1 Introduction

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions, but only that it does not function properly under specific conditions. The scope of software testing often includes the examination of code as well as the execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

6.1.1 Testing Approaches

Static, dynamic and passive testing

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, like proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules.

Exploratory approach

Exploratory testing is an approach to software testing that is concisely described simultaneous learning, test design and test execution. CemKaner, who coined the term in 1984, defines exploratory testing as "a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.

The "box" approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that the tester takes when designing test cases. A hybrid approach called grey-box testing may also be applied to software testing methodology. With the concept of grey-box testing4which develops tests from specific design elements4gaining prominence, this "arbitrary distinction" between black- and white-box testing has faded somewhat

White-box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs

Black-box testing

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements.

Component interface testing

Component interface testing is a variation of black-box testing, with the focus on the data values beyond just the related actions of a subsystem component. The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit.

Visual testing

The aim of visual testing is to provide developers with the ability to examine what was happening at the point of software failure by presenting the data in such a way that the developer can easily find the information she or he requires, and the information is expressed clearly. At the core of visual testing is the idea that showing someone a problem (or a test failure), rather than just describing it, greatly increases clarity and understanding. Visual testing, therefore, requires the recording of the entire test process 3 capturing everything that occurs on the test system in video format. Output videos are supplemented by real-time tester input via picture-in-a- picture webcam and audio commentary from microphones.

6.1.2 Testing Levels

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

System testing

System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a login interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

6.2 Test Cases

A test case is a documentation which specifies input values, expected output and the preconditions for executing the test. Test case document is also a part of test deliverables, by reading test case document stakeholders get an idea about the quality of test cases written and the effectiveness of those test cases. Stakeholders can also provide inputs about the current set of test cases as well as suggest some more missing test cases.

A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test cases for software helps to guide the tester through a sequence of steps to validate whether a software application is free of bugs and working as required by the end user. The expected result tells the tester what they should experience as a result of the test performed. In this way we can determine if the test case is a <pass= or <fail=.

Test case 1:

Figure 6.1 Unit Testing Table

TEST CASE	UNIT TESTING
TEST NAME	PREDICTION
ITEMS BEING TESTED	PREDICT FUNCTIONALITY
SAMPLE INPUT	ACADEMIC PERFORMANCE INDICATORS
EXPECTED OUTPUT	ACCURATE PREDICTION OF ACADEMIC OUTCOME
ACTUAL OUTPUT	SUCCESSFUL ACADEMIC PERFORMANCE PREDICTION
REMARKS	PASS

Test case 2:

Figure 6.2 Integration Testing Table

TEST CASE	INTEGRATION TESTING
TEST NAME	ACADEMIC OUTCOME PREDICTION
ITEMS BEING TESTED	PREDICTION FUNCTIONALITY
SAMPLE INPUT	INCOMPLETE OR INACCURATE ACADEMIC DATA
EXPECTED OUTPUT	DETECTION OF INCONSISTENCIES AND INACCURACIES IN ACADEMIC OUTCOME PREDICTION
ACTUAL OUTPUT	UNSUCCESSFUL ACADEMIC PREDICTION DUE TO INCOMPLETE/INVALID DATA
REMARKS	PASS

Test case 3:

Figure 6.3 System Testing Table

TEST CASE	SYSTEM TESTING
TEST NAME	SYSTEM TESTING IN DIFFERENT OS
SAMPLE INPUT	EXECUTE THE PROGRAM IN WINDOWS AND macOS
EXPECTED OUTPUT	EQUALL PERFORMANCE IN BOTH WINDOWS AND macOS
ACTUAL OUTPUT	SAME AS EXPECTED OUTPUT
REMARKS	PASS

6.3 Result

A result is the final consequence of actions or events expressed qualitatively or quantitatively. Performance analysis is an operational analysis, is a set of basic quantitative relationship between the performance quantities.

XGBoost

```
In [76]: xgb = XGBClassifier(objective='binary:logistic', random_state=0)
         xgb.fit(X_train, y_train)
         y_preds = xgb.predict(X_test)

In [77]: print_results('XGBoost', y_test, y_preds)

         XGBoost
         Accuracy: 0.888
         Precision: 0.885
         Recall: 0.931
         F1 Score: 0.908
```

Figure 6.4 XGBOOST Score

The XGBoost Classifier algorithm achieved an accuracy score of 89% in predicting academic outcomes on the test dataset, utilizing the train dataset for model training. (Refer to Figure 6.4)

6.3.1 Result (Predicting Success)

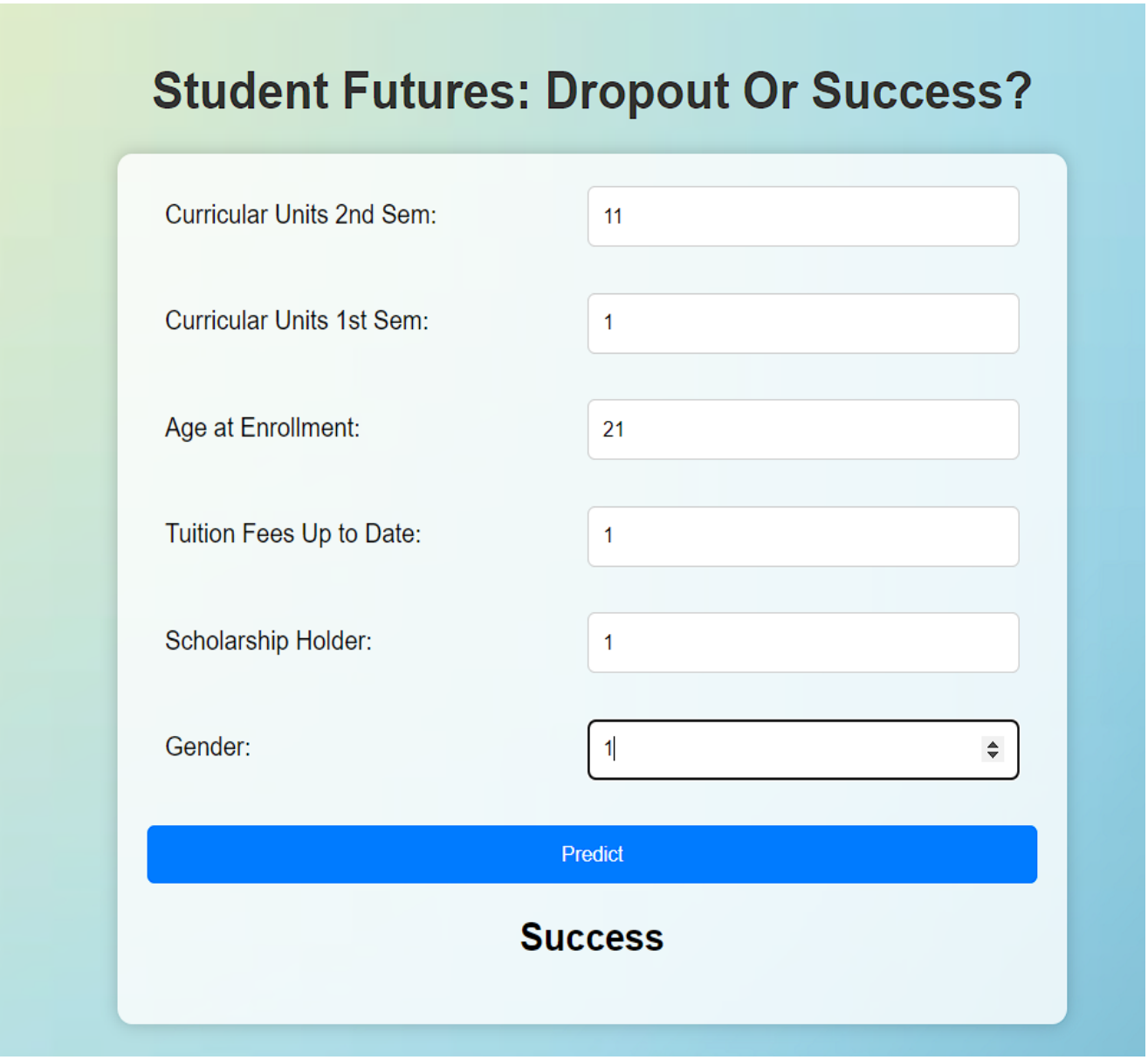


Figure 6.1 Success Prediction

6.3.2 Result (Predicting Dropout)

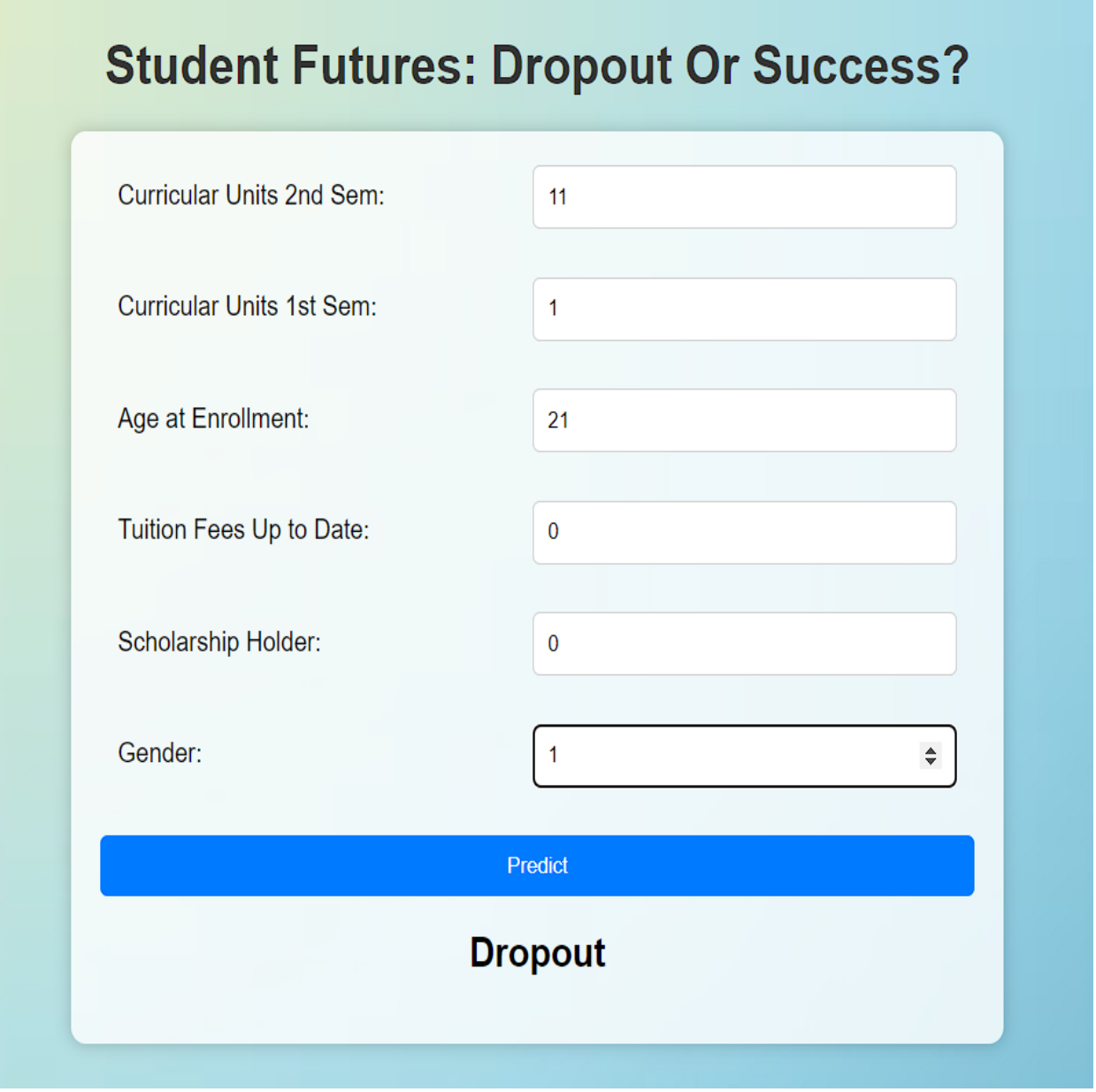


Figure 6.1 Dropout Prediction

6.4 Performance Evaluation

After training a set of five models, including XGBoost, Support Vector Machine (SVM), Random Forest, Logistic Regression, and Decision Tree, their performance evaluation was conducted on test data. Several evaluation metrics, such as accuracy, were utilized to gauge their predictive capability. Notably, the XGBoost model exhibited the highest accuracy among the tested models, achieving an impressive accuracy score of 88.76%. This outcome signifies the superior predictive performance of the XGBoost model compared to the other models in this evaluation.

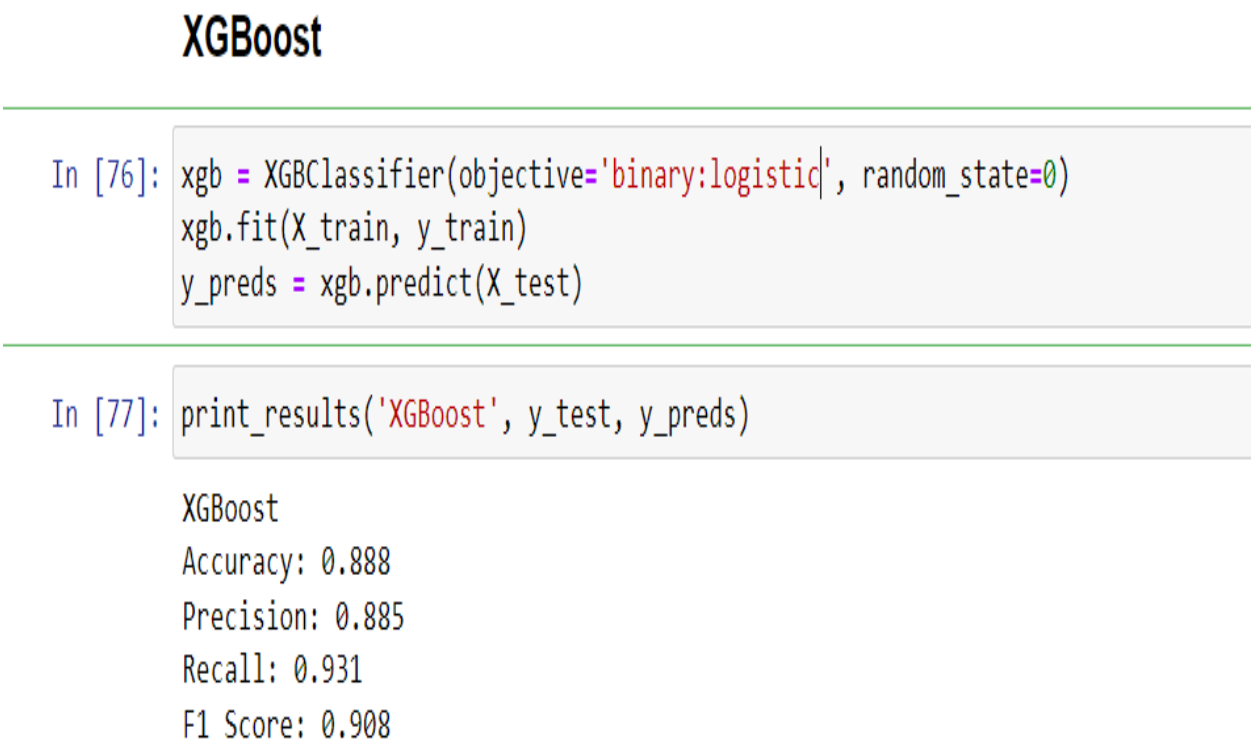


Figure 6.5 XGBOOST Accuracy

Following the assessment of these models, besides accuracy, various other evaluation metrics were employed to comprehensively gauge their performance. Metrics such as precision, recall, F1-score, and area under the curve (AUC) were computed. Each metric provides distinct insights into the model's performance: precision measures the accuracy of positive predictions, recall assesses the model's ability to identify all relevant instances, the F1-score balances precision and recall, and the AUC calculates the model's ability to distinguish between classes. These metrics collectively offer a more nuanced understanding of each model's strengths and weaknesses, contributing valuable insights into their overall performance characteristics beyond accuracy alone.

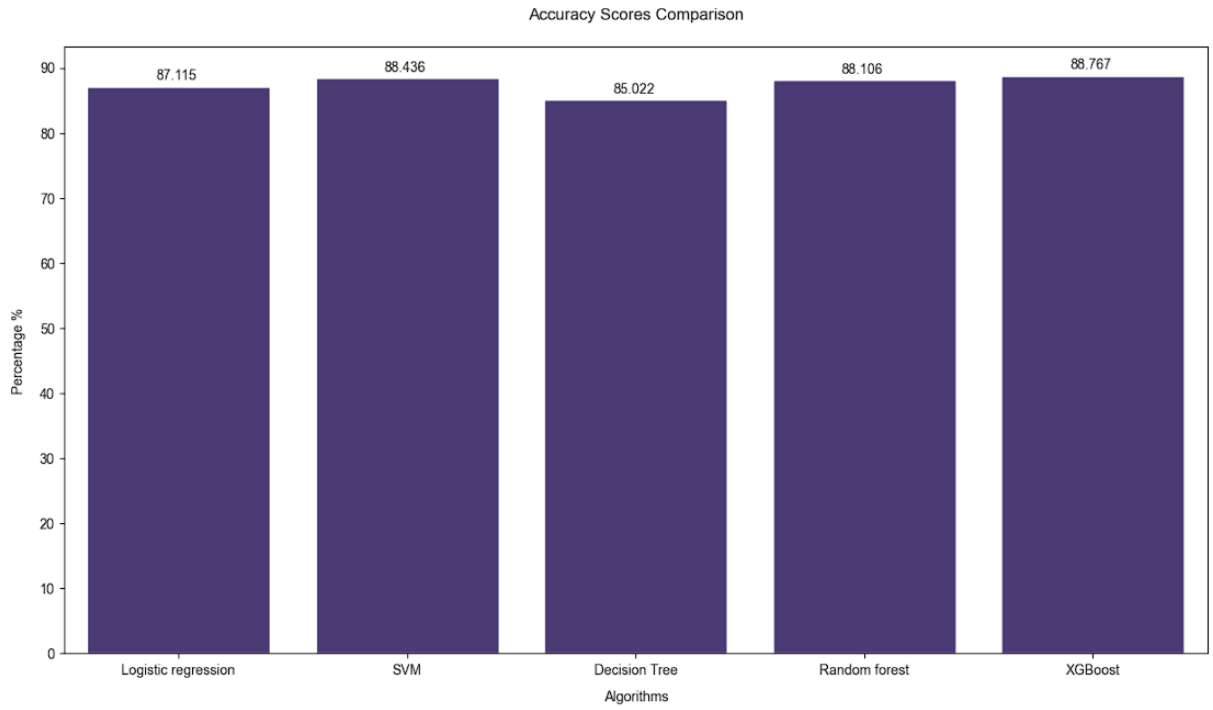


Figure 6.6 Performance Evaluation

6.5 Summary

The primary objective of this system revolves around predicting dropout occurrences and academic success using a supervised learning framework. Following the training of 5 models, namely the XGBoost, Support Vector Machine (SVM), Random Forest, Logistic Regression, and Decision Tree, performance evaluation was conducted, employing accuracy as the key evaluation metric. The XG Classifier demonstrated an accuracy of 88.76%, which is higher than all models. As a result, considering the superior accuracy achieved by the XG Classifier, it has been selected to facilitate the real-time prediction of dropout instances and academic success, leveraging live data input obtained through user-initiated interactions.

References

1. Ameen, A. O., Alarape, M. A., & Adewole, K. S. (2019). STUDENTS' ACADEMIC PERFORMANCE AND DROPOUT PREDICTION. MALAYSIAN JOURNAL OF COMPUTING, 4(2), 278.

2. Alyahyan, E., & Düştegör, D. (2020). Predicting academic success in higher education: literature review and best practices. International Journal of Educational Technology in Higher Education, 17(1).

3. Respondek, L., Seufert, T., Stupnisky, R., & Nett, U. E. (2017). Perceived Academic Control and Academic Emotions Predict Undergraduate University Student Success: Examining Effects on Dropout Intention and Achievement.

4. Niyogisubizo, J., Liao, L., Nziyumva, E., Murwanashyaka, E., & Nshimyumukiza, P. C. (2022). Predicting student's dropout in university classes using a two-layer ensemble machine learning approach: A novel stacked generalization.

5. Chen, T., & Guestrin, C. (2016). XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

6. Patel, H. H., & Prajapati, P. (2018). Study and Analysis of Decision Tree-Based Classification Algorithms.

Appendices

Project Report Overview

This comprehensive project report encapsulates our endeavour to predict students' dropout occurrences and academic success, a critical pursuit in the realm of education. The report serves as an extensive narrative, detailing our methodologies, findings, and implications for future research and practical applications within the educational domain.

Research Scope and Objectives

Our primary goal was to develop predictive models capable of anticipating instances of dropout and forecasting academic success among students. The scope encompassed the utilization of advanced machine learning algorithms, data analysis, and predictive modelling techniques to create robust frameworks for this crucial task.

Methodologies and Approaches

The project delved into comprehensive data analysis, exploring diverse attributes and patterns intrinsic to student academic performance. Leveraging sophisticated machine learning algorithms like XGBoost, Random Forest we constructed predictive models capable of discerning potential dropout indicators and forecasting academic accomplishments.

Key Findings and Insights

Through meticulous evaluation and validation processes, our models exhibited significant predictive prowess. Notably, the XGBoost model emerged as a prominent performer, boasting an accuracy of 88.76% in anticipating dropout incidents and academic achievements.

Implications and Future Directions

Our findings hold substantial implications for educational institutions and policymakers. The predictive models developed in this project offer invaluable insights into identifying students at risk of dropout and forecasting academic success, enabling proactive interventions and tailored support mechanisms.

Conclusion

In conclusion, this project report stands as a testament to our commitment to leveraging machine learning methodologies to address pertinent challenges in education. Our predictive models exhibit considerable potential to revolutionize educational practices by aiding in early intervention strategies for dropout prevention and fostering academic success among students.

This report sets the stage for continued advancements in predictive modelling within education and underscores the significance of data-driven approaches in shaping the future of student success and educational outcomes.