PROGRAMME DEMAND PREDICTION FOR A COLLEGE

MINOR PROJECT REPORT

Submitted by

JEGADEESH K

(23MCM022)

Under the Guidance of

Dr. S .VENKATA KRISHNA KUMAR

Associate Professor & Head of the Department

PG and Research Department of Computer Science - Aided

In partial fulfillment of the requirements for the award of the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

of Bharathiar University

PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE - AIDED

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College - Affiliated to Bharathiar University
Accredited with 'A^{++}' grade by NAAC(4th cycle)
College with Potential for Excellence
(Status Awarded by the UGC)
Star College Status Awarded by DBT-MST
An ISO 9001:2015 Certified Institution
Civil Aerodrome Post
Coimbatore -641 014

OCTOBER 2024

PROGRAMME DEMAND PREDICTION FOR A COLLEGE

MINOR PROJECT REPORT

Submitted by

JEGADEESH K

(23MCM022)

Under the Guidance of

Dr. S .VENKATA KRISHNA KUMAR

Associate Professor & Head of the Department

PG and Research Department of Computer Science - Aided

In partial fulfillment of the requirements for the award of the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

of Bharathiar University

PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE - AIDED

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College - Affiliated to Bharathiar University
Accredited with ‘A^{++}’ grade by NAAC(4th cycle)
College with Potential for Excellence
(Status Awarded by the UGC)
Star College Status Awarded by DBT-MST
An ISO 9001:2015 Certified Institution
Civil Aerodrome Post
Coimbatore -641 014

OCTOBER 2024

PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE – AIDED

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College - Affiliated to Bharathiar University

Accredited with A^{++} grade by NAAC(4th cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

Star College Status Awarded by DBT-MST

An ISO 9001:2015 Certified Institution

Civil Aerodrome Post

Coimbatore -641 014

CERTIFICATE

This is to certify that this Minor Project work entitled **"PROGRAMME DEMAND PREDICTION FOR A COLLEGE"** is a bonafide record of work done by **JEGADEESH K (23MCM022)** in partial fulfillment of the requirements for the award of Degree of **Master of Science in Computer Science** of Bharathiar University.

Faculty Guide ______________________ Head of the Department ______________________

Submitted for Viva-Voce Examination held on ____________________

Internal Examiner ______________________ External Examiner ______________________

DECLARATION

I, **JEGADEESH K (23MCM022)**, hereby declare that this Minor Project work entitled **"PROGRAMME DEMAND PREDICTION FOR A COLLEGE"**, is submitted to **PSG College of Arts & Science (Autonomous), Coimbatore** in partial fulfillment for the award of **Master of Science in Computer Science** degree is a record of original work done by me under the supervision and guidance of **Dr .S. VENKATA KRISHNA KUMAR**, Associate Professor & Head, PG and Research Department of Computer Science - Aided, PSG College of Arts & Science, Coimbatore.

This project work has not been submitted by me for the award of any other Degree/ Diploma/ Associate ship/ Fellowship or any other similar degree to any other university.

PLACE	: COIMBATORE	**JEGADEESH K**
DATE	:	**(23MCM022)**

ACKNOWLEDGEMENT

My venture stands imperfect without dedicating my gratitude to few people who have contributed a lot towards the victorious completion for my project work.

I express my sincere thanks to **Dr. D. Brindha,** Principal, PSG for her valuable advice College of Arts & Science Coimbatore for given me an opportunity to do this project.

I sincerely thank **Dr.S. VENKATA KRISHNA KUMAR,** Associate Professor & Head, PG and Research Department of Computer Science (Aided) for her whole hearted help to complete this project successfully by giving valuable suggestions.

I thank my project guide **Dr.S. VENKATA KRISHNA KUMAR,** Associate Professor & Head, PG and Research Department of Computer Science (Aided) for his timely suggestion which had enable me in completing the project successfully.

This note of acknowledgement will be incomplete without paying my heartfelt devotion to my parents, my friends and other people, for their blessings, encouragement, financial support and the patience, without which it would have been impossible for me to complete the job.

PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE - AIDED

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous College - Affiliated to Bharathiar University
Accredited with 'A++' grade by NAAC (4th Cycle)
College with Potential for Excellence
(Status Awarded by the UGC)
Star College Status Awarded by DBT – MST
An ISO 9001:2015 Certified Institution
Civil Aerodrome Post
Coimbatore - 641 014

<u>CERTIFICATE</u>

This is to certify that this Project work entitled as **"PROGRAMME DEMAND PREDICTION FOR A COLLEGE"** subjected to PSG College of Arts & Science (Autonomous), Coimbatore, Affiliated to Bharathiar University in partial fulfilment for the award of Degree of **Master of Science in Computer Science,** a record of original work done by **JEGADEESH K (23MCM022)** during July 2024 to November 2024 of her study in the Department of Computer Science, PSG College of Arts & Science affiliated to Bharathiar University under my supervision and guidance. This minor work has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or any other similar degree to any other university.

Signature of the Guide	**Signature of HoD**
Dr. S. Venkata Krishna Kumar	Dr. S. Venkata Krishna Kumar
Associate Professor & Head	Associate Professor & Head
Department of Computer Science - Aided	Department of Computer Science - Aided
PSG College of Arts & Science	PSG College of Arts & Science
Coimbatore - 641014.	Coimbatore – 641014.

SYNOPSIS

Project Title : Programme Demand Prediction for a College

Overview:

The project "Programme Demand Prediction for a College" focuses on predicting the demand for various college programmes based on multiple influential factors such as job market conditions, educational level, technological advancements, political stability, and more.

Technology Used:

- **Machine Learning Algorithm:** Random Forest Regressor
- **Programming Language:** Python
- **Data Sources:** Educational databases, government reports, trade association publications, and industry feedback.

Goals and Objective:

The primary goal of this project is to provide a predictive model that can forecast the demand for various college programmes based on real-world influencing factors. Analyze the influence of factors such as job market trends, education levels, infrastructure, and demographics on programme demand.

- Use machine learning to predict the demand for specific academic programmes across colleges.
- Provide strategic insights and actionable recommendations to academic institutions for optimizing programme offerings.

Applications:

- **Academic Institutions:** Colleges and universities can utilize this model to predict student enrollment patterns and allocate resources accordingly.
- **Educational Policy Makers:** Government bodies can use the insights generated to adjust policies and educational funding.
- **Career Guidance Platforms:** Such platforms can offer students recommendations on which programmes have a higher likelihood of securing jobs based on market trends.

Advantages:

- **Accurate Prediction:** The use of Random Forest ensures that the model captures complex relationships between the factors and programme demand.
- **Data-Driven Decision Making:** Helps colleges make informed decisions about expanding or downsizing specific programs.
- **Adaptability**: The model can be updated with new data, allowing it to remain relevant as market trends and external conditions change.
- **Strategic Resource Allocation**: Enables efficient use of institutional resources by aligning programme offerings with actual demand.
- **Scenario Simulation:** The model can simulate various scenarios, such as changes in job markets or political shifts, providing valuable foresight into future demand trends.

TABLE OF CONTENTS

CHAPTER - 1
INTRODUCTION

The "Programme Demand Prediction for a College" project is designed to address the need for informed decision-making in higher education institutions. With the increasing diversity of educational programmes and the dynamic nature of factors influencing student demand, predicting which programmes will be in high demand is crucial for colleges. This project utilizes machine learning, specifically the Random Forest Regressor algorithm, to predict programme demand across different regions in India.

The dataset includes detailed information on various districts across India, where each district contains two colleges offering seven streams: IT, Maths, Civil, Mechanical, Life Science, Hospitality, and Commerce. The project analyzes these programmes in the context of ten key factors known to influence demand:

- **Job Market Level:** Current trends in the job market that impact students' programmes choices.
- **Education Level Based on Geographical Location:** Regional disparities in education levels influencing college demand.
- **Technological Advancements:** The impact of emerging technologies on programme popularity.
- **Political Stability:** The effect of stable governance on the region's educational environment.
- **Natural Disasters:** How natural calamities affect migration patterns and, consequently, college admissions.
- **Infrastructure (e.g., E-learning):** Availability and quality of educational infrastructure influencing student enrollment.
- **Demographic Data:** Socio-economic and educational backgrounds of students in each region.
- **Feedback (Initial, Final, Alumni):** Insights from current students and alumni impacting programme reputation and demand.
- **Core Industry Performance:** The influence of industry health on related academic programs.
- **Trade Association Reports:** Projections on industry growth over the next five years, guiding programme demand.

By leveraging the Random Forest Regressor, the project aims to capture complex relationships and interactions among these factors to predict the demand for each stream in each district.

PROJECT OVERVIEW

In the rapidly evolving landscape of higher education, colleges and universities face the challenge of aligning their programme offerings with the dynamic demands of the job market and societal needs. The choice of academic programmes by students is influenced by a complex interplay of factors, including economic trends, technological advancements, regional characteristics, and the evolving needs of industries. Consequently, predicting which academic programmes will experience higher demand has become crucial for educational institutions to ensure their relevance and sustainability.

This project, "Programme Demand Prediction for a College," seeks to address this challenge by employing machine learning techniques to forecast the demand for various academic streams across colleges in different districts of India. Using the Random Forest Regressor algorithm, the project analyzes multiple factors—such as job market conditions, educational levels, political stability, infrastructure, demographic data, and more—that influence students' preferences for specific programs.

The dataset encompasses a wide range of variables, including data from two colleges in each district across all states of India, offering seven streams: IT, Maths, Civil, Mechanical, Life Science, Hospitality, and Commerce. By considering diverse and impactful factors such as regional job markets, technological trends, feedback from students and alumni, and industry growth forecasts, the model aims to provide a comprehensive understanding of programme demand patterns.

The insights generated from this project can serve as a valuable resource for educational institutions. By predicting the demand for different programs, colleges can make informed decisions on resource allocation, curriculum development, and strategic planning. This ensures that they can meet the evolving needs of students and the job market, thereby enhancing their competitiveness and ability to provide quality education tailored to future trends.

The resulting predictive model can provide colleges with actionable insights for strategic planning, such as optimizing programme offerings, improving resource allocation, and enhancing their alignment with current and future market trends. Ultimately, this project serves as a decision support tool, helping colleges respond to changing educational and industry landscapes effectively.

CHAPTER - 2
SYSTEM SPECIFICATION

INTRODUCTION

System configuration refers to the arrangement of both hardware and software components in a computer or system that determines how it functions and performs. It encompasses settings, resources, and specifications needed for the system to operate effectively.

2.1 DEVELOPMENT ENVIRONMENT

A setup where software is built, tested, and debugged, including tools like IDEs, compilers, and version control, tailored for developers to write and modify code.

HARDWARE SPECIFICATION

PROCESSOR	:	INTEL i5 PROCESSOR, 12TH GEN.
STORAGE	:	256 GB
MONITOR	:	14“ SAMTRON MONITOR
RAM	:	4 GB
CPU CLOCK	:	1.08 GHz

SOFTWARE SPECIFICATION

OPERATING SYSTEM	:	WINDOWS 10
FRONT END	:	HTML, CSS, JavaScript
BACK END	:	Python 3.x
FRAMEWORK	:	Flask
DEVELOPMENT TOOLS	:	Google Colab (for downloading dataset) PyCharm (for developing the project)

2.2 IMPLEMENTATION ENVIRONMENT

The actual live setup where the final software is deployed and used by end-users, configured with production servers, databases, and networking to ensure smooth operation.

HARDWARE SPECIFICATION

PROCESSOR	:	INTEL i5/i7 or AMD Ryzen 5/7.
STORAGE	:	256 GB SSD or more
MONITOR	:	14“ SAMTRON MONITOR
RAM	:	4 GB / 8 GB
CPU CLOCK	:	1.08GHz

SOFTWARE SPECIFICATION

OPERATING SYSTEM	:	WINDOWS 10
FRONT END	:	HTML, CSS, JavaScript
BACK END	:	Python 3.x
FRAMEWORK	:	Flask
DEVELOPMENT TOOLS	:	Google Colab (for downloading dataset) PyCharm (IDE for developing the project)

2.3 SOFTWARE DESCRIPTION

FRONT END :

HTML

HTML (Hyper Text Markup Language) is the standard language used to create and design web pages. It provides the structure and content of a web page by defining elements like text, images, links, forms, and more. HTML uses a system of tags and attributes to define the layout and appearance of elements on a webpage.

Features of HTML:

- **Simple and Easy to Learn:** HTML is relatively easy to understand and use, making it suitable for beginners who want to create web pages.
- **Platform Independent:** HTML files can be created and viewed on any platform, whether it's Windows, macOS, or Linux.
- **Supports Multimedia:** HTML allows the embedding of images, audio, video, and other multimedia elements, enhancing the user experience.
- **Hyperlink Support:** One of the core features of HTML is the ability to create hyperlinks that connect web pages both internally and externally, creating a network of pages.
- **Structured Documents:** HTML provides a way to structure content using headings, paragraphs, lists, tables, etc., making it easier for browsers to render the content.
- **Forms and Input:** HTML includes a variety of form elements like text boxes, buttons, checkboxes, radio buttons, and more, allowing users to interact with web pages.
- **Semantic Elements:** HTML5 introduced semantic elements like <header>, <footer>, <article>, and <section>, which provide meaning to the content and help search engines and assistive technologies understand the structure of the page.
- **Compatibility with CSS and JavaScript:** HTML can be used in conjunction with CSS for styling and JavaScript for interactivity, allowing for the creation of dynamic and visually appealing websites.
- **Browser Support:** HTML is supported by all modern web browsers, ensuring that HTML-coded web pages are accessible to a wide audience.
- **Responsive Design:** HTML5 supports responsive design techniques through the use of media queries and flexible grid layouts, allowing web pages to adapt to different screen sizes and devices.

CSS

CSS (Cascading Style Sheets) is a stylesheet language used to control the presentation, layout, and visual styling of web pages written in HTML or XML. It allows developers to separate the content structure (HTML) from the design and layout, making it easier to maintain and style websites consistently across multiple pages.

Features of CSS:

- **Separation of Content and Design:** CSS enables a clear separation between the structure (HTML) and presentation (CSS) of a website. This separation makes it easier to maintain and update the design without altering the content.
- **Control Over Layout:** CSS provides advanced layout techniques like Flexbox, Grid, and positioning properties, allowing for precise control over the placement of elements on a page. It helps create complex, responsive, and dynamic layouts.
- **Styling and Theming:** CSS allows you to change the look and feel of a website, including colors, fonts, backgrounds, borders, spacing, and more. You can easily implement themes and modify the site's appearance by changing a few CSS properties.
- **Responsive Design:** CSS includes features like media queries, which allow you to create responsive designs that adapt to different screen sizes and devices, ensuring a consistent user experience across desktops, tablets, and smartphones.
- **Selector Mechanism:** CSS provides a powerful selector mechanism to target HTML elements. Selectors include element selectors, class selectors, ID selectors, attribute selectors, and more, giving precise control over which elements to style.
- **Cascading and Inheritance:** CSS uses a "cascading" mechanism, which means styles can cascade from the top of the stylesheet to the bottom. This allows you to define general styles and override them with more specific styles. CSS also supports inheritance, where some properties of parent elements are inherited by their children.

JavaScript

JavaScript is a versatile, high-level programming language that is widely used to add interactivity, control multimedia, and dynamically update the content of web pages. It's a core technology of the web, along with HTML and CSS, and is essential for creating dynamic and responsive web applications. JavaScript can be executed on the client side (in the user's web browser) and on the server side (using environments like Node.js).

Features of JavaScript:

- **Interactivity:** JavaScript allows you to create interactive elements on web pages, such as form validation, sliders, modals, pop-ups, and interactive maps. It responds to user actions like clicks, key presses, and mouse movements, enhancing user engagement.
- **Client-Side Execution:** JavaScript is primarily executed on the client side, meaning the code runs directly in the user's web browser. This reduces server load and provides a faster, more responsive user experience.
- **Dynamic Content:** JavaScript enables you to modify HTML and CSS on the fly. You can dynamically change the content of a web page, update styles, or even add or remove elements from the DOM (Document Object Model) without reloading the page.
- **Versatility:** JavaScript can be used across various platforms and environments. It's not limited to just web development; with the advent of Node.js, JavaScript can also be used for server-side programming, desktop applications (e.g., Electron), and even mobile app development (e.g., React Native).
- **Event-Driven Programming:** JavaScript supports event-driven programming, allowing developers to define custom behaviors in response to user events such as clicks, form submissions, mouseovers, and more.
- **Asynchronous Programming:** JavaScript supports asynchronous programming using callbacks, promises, and the `async/await` syntax. This feature is essential for handling tasks like network requests, timers, and file operations without blocking the main execution thread.
- **Built-in Functions and Libraries:** JavaScript provides a rich set of built-in functions and libraries for tasks like string manipulation, date and time operations, mathematical calculations, and data handling. Additionally, it has a large ecosystem of third-party libraries and frameworks (e.g., jQuery, React, Angular, Vue.js) that extend its capabilities.

BACK END :

PYTHON

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python's design philosophy emphasizes code readability and the use of significant indentation, which makes it easy to learn and use, even for beginners.

Features of Python:

- **Simple and Easy to Learn:** Python has a clean and straightforward syntax that closely resembles English, making it easy to read and write. This simplicity reduces the learning curve for beginners.
- **Interpreted Language:** Python is an interpreted language, meaning that code is executed line-by-line, making it easier to debug and test. There's no need for compilation, which speeds up the development process.
- **Dynamically Typed:** Python is dynamically typed, so you don't need to declare the type of a variable when you create it. This flexibility allows for faster and more flexible coding but also requires careful handling to avoid runtime errors.
- **Cross-Platform Compatibility:** Python is a cross-platform language, meaning it can run on various operating systems like Windows, macOS, Linux, and more without requiring modification to the codebase.
- **Extensive Standard Library:** Python has a vast standard library that includes modules and packages for various tasks, such as file I/O, system operations, data manipulation, web services, and more. This rich library minimizes the need for external dependencies.
- **Large Ecosystem and Community Support:** Python has a large ecosystem of third-party libraries and frameworks for different domains, including web development (Flask, Django), data science (NumPy, pandas, scikit-learn), artificial intelligence (TensorFlow, PyTorch), automation, and more. Its vibrant community actively contributes to its growth, providing ample resources, documentation, and support.
- **Object-Oriented and Functional Programming:** Python supports both object-oriented and functional programming paradigms, offering flexibility in how you structure your code. You can define classes and create objects, or use higher-order functions, lambda functions, and map/reduce/filter operations.
- **Portability:** Python code can be written once and run anywhere without requiring modifications, making it portable across different platforms. This portability is one of Python's most significant

advantages for cross-platform development.

- **Extensibility and Integration:** Python can easily integrate with other languages like C, C++, and Java using various modules and APIs. This makes it extensible and suitable for performance-critical applications where low-level language integration is necessary.
- **High-Level Language:** Python abstracts away complex details like memory management, pointer arithmetic, and garbage collection, allowing developers to focus on writing logic rather than worrying about low-level system details.
- **Open Source:** Python is open-source software, meaning it is freely available for anyone to use, modify, and distribute. This open-source nature has led to its widespread adoption and continuous improvement.
- **Versatility:** Python is used in a wide range of applications, from web development, data analysis, machine learning, and artificial intelligence to scripting, automation, game development, and more. Its versatility makes it a go-to language for many developers.
- **Asynchronous Programming:** Python supports asynchronous programming using `asyncio`, allowing you to write asynchronous code for handling I/O-bound tasks like network operations and file I/O more efficiently.

FLASK FRAMEWORK :

Flask is a lightweight and micro web framework for Python. It's designed to be simple and easy to use, making it a popular choice for small to medium-sized web applications. Flask provides the essential tools and features needed to build web applications, but it does not enforce any specific project structure or require the use of specific tools, giving developers the freedom to choose their preferred components.

Key Features of Flask:

- **Lightweight and Minimalistic:** Flask is a "micro" framework, meaning it provides the core essentials needed for web development without the bloat of unnecessary features. This makes Flask simple and easy to get started with.
- **Modular and Flexible:** Flask is highly modular, allowing developers to plug in additional libraries or extensions as needed. It does not enforce any strict rules or patterns, providing flexibility in how you structure your application.
- **Built-in Development Server:** Flask includes a built-in development server with debugging and error handling capabilities, allowing for easy testing and debugging during development.

- **Routing:** Flask provides a simple and intuitive way to define routes using decorators, allowing you to map URLs to Python functions. This makes it easy to define the behavior of different endpoints.
- **Template Engine:** Flask uses Jinja2 as its default template engine, allowing you to separate the presentation layer (HTML) from the business logic (Python code). Jinja2 supports template inheritance, filters, and more.
- **RESTful Request Handling:** Flask makes it easy to build RESTful APIs by providing simple request handling mechanisms. You can easily define and handle different HTTP methods (GET, POST, PUT, DELETE) within your application.
- **Extensions:** Flask has a wide range of extensions available for tasks like form validation, database integration (SQLAlchemy), authentication, and more. These extensions can be easily integrated into your Flask application to extend its functionality.

PACKAGES & LIBRARIES

- **pandas:** A powerful Python library for data manipulation and analysis, providing data structures like DataFrames for efficient data handling and operations such as filtering, grouping, and merging.
- **scikit-learn (sklearn):** A machine learning library in Python offering simple and efficient tools for data mining and analysis, including algorithms for classification, regression, clustering, and model evaluation.
- **matplotlib:** A comprehensive library for creating static, animated, and interactive visualizations in Python, allowing for a wide range of plots such as line graphs, histograms, and scatter plots.
- **io (Python I/O module):** Provides a collection of classes and functions for handling various types of I/O operations, including reading and writing files, working with binary and text streams, and memory buffers.
- **base64:** A module that provides functions for encoding binary data into base64-encoded strings and decoding them back, commonly used for encoding data in formats like emails, URLs, and data URLs.

CHAPTER - 3
SYSTEM ANALYSIS

INTRODUCTION

System analysis is conducted for the purpose of studying or its part in order to identify its objectives .It is a problem solving techniques that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

3.1 EXISTING SYSTEM

In the current scenario, many colleges and universities rely on traditional methods for predicting programme demand, such as historical data analysis, expert opinions, surveys, and manual market research. These methods often involve analyzing past enrollment trends, feedback from students and alumni, and qualitative assessments of industry needs. While these approaches have been useful, they have several significant limitations .

Limited Data Scope: Traditional methods often focus on a limited set of factors, such as historical enrollment data and basic market research, ignoring the broader set of dynamic factors like technological advancements, political stability, and core industry performance.

Lack of Real-Time Analysis: Most conventional systems lack the ability to process real-time data, leading to delayed or outdated insights that may not reflect the current market conditions or emerging trends.

Subjectivity: Relying heavily on expert opinions and qualitative assessments can introduce bias, resulting in less objective and potentially inaccurate demand predictions.

Static and Inflexible: Traditional methods do not easily adapt to sudden changes in the environment, such as natural disasters or rapid technological changes, limiting their ability to provide accurate and timely predictions.

High Resource Intensity: Manual data collection, analysis, and market research can be resource-intensive, requiring significant time and effort without guaranteeing high accuracy.

Lack of Predictive Capability: These methods are often descriptive rather than predictive, focusing on explaining past trends rather than forecasting future demand.

3.2 PROPOSED SYSTEM

INTRODUCTION

The proposed system leverages a machine learning-based approach, using the Random Forest Regressor algorithm to predict programme demand in colleges across various districts in India. This system incorporates a comprehensive dataset that includes a wide range of factors influencing programme demand, such as job market conditions, technological advancements, political stability, infrastructure, demographic data, and industry performance.

ADVANTAGES OF THE PROPOSED SYSTEM

Comprehensive Analysis: By including a diverse set of factors, the proposed system offers a holistic view of the elements influencing programme demand, leading to more accurate and nuanced predictions.
Real-Time and Dynamic Predictions: The model can be trained and updated with new data, allowing it to adapt to changing conditions and provide real-time predictions that reflect current trends and future projections.
Objective and Data-Driven: Utilizing machine learning ensures that predictions are based on objective data analysis rather than subjective judgment, reducing bias and increasing the reliability of the predictions.
Handling Complexity: Random Forest Regressor can handle complex relationships and interactions between multiple factors, providing a more sophisticated analysis of how different elements contribute to programme demand.
Resource Efficiency: Automated data analysis reduces the time and resources required for manual research, allowing institutions to focus on strategic planning rather than data gathering.
Adaptive to Change: The system can accommodate sudden changes in the environment, such as natural disasters or shifts in political stability, by incorporating relevant data into the model, ensuring that predictions remain accurate and relevant.
Strategic Decision-Making: With accurate demand predictions, colleges can optimize their programme offerings, improve resource allocation, and enhance curriculum development to better meet the needs of students and the job market.
By using this machine learning approach, the proposed system provides a powerful tool for educational institutions to navigate the complexities of programme demand, ensuring they remain competitive and responsive to the evolving educational and industry landscapes.

CHAPTER - 4
SYSTEM DESIGN

4.1 PROCESS DESCRIPTION

The "Programme Demand Prediction for a College" project involves several key steps, from data collection and preprocessing to model training, evaluation, and deployment. Here's a detailed breakdown of each phase in the project:

1. **Data Collection**

- **Dataset Composition:** The dataset comprises information about districts across all states in India, with each district containing two colleges (College 1 and College 2). Each college offers seven streams: IT, Maths, Civil, Mechanical, Life Science, Hospitality, and Commerce. Additionally, the dataset includes ten influential factors: job market level, education level based on geographical location, technological advancements, political stability, natural disasters, infrastructure, demographic data, feedback, core industry data, and trade association reports.
- **Data Sources:** The data may be sourced from various records, including educational databases, government reports, industry publications, and feedback surveys. This diverse data ensures a comprehensive understanding of the factors affecting programme demand.

2. **Data Preprocessing**

- **Data Cleaning:** The dataset is cleaned to handle missing values, outliers, and inconsistencies. For instance, missing values in feedback or economic data may be imputed using statistical methods or domain knowledge.
- **Feature Engineering:** Relevant features are extracted and engineered to enhance the predictive power of the model. This could include aggregating feedback scores into a single metric, normalizing the job market level, or encoding categorical variables like political stability into numerical form.
- **Data Normalization:** Continuous variables such as infrastructure quality, demographic data, and job market levels are normalized to ensure they contribute proportionately to the model.
- **Feature Selection:** Redundant or irrelevant features are identified and removed to improve model performance and reduce computational complexity. Feature importance can be assessed using statistical tests or correlation matrices to select the most influential factors.

3. Model Development

- **Algorithm Selection:** The Random Forest Regressor is chosen for this project due to its robustness and ability to handle complex relationships between features. It is an ensemble learning method that builds multiple decision trees and merges their outputs to provide accurate predictions.
- **Model Training:** The dataset is split into training and testing sets. The training set is used to train the Random Forest Regressor, where the model learns patterns and relationships between the input factors (job market level, infrastructure, etc.) and the target variable (demand for each stream).
- **Hyperparameter Tuning:** The model's performance is optimized by tuning hyperparameters such as the number of decision trees, maximum tree depth, and minimum samples required to split a node. Cross-validation techniques, like k-fold cross-validation, are used to find the optimal hyperparameters that minimize prediction errors.

4. Model Evaluation

- **Performance Metrics:** The model is evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2) to assess its predictive accuracy. These metrics help determine how well the model can predict programme demand in new, unseen data.
- **Validation:** The testing set is used to validate the model's performance, ensuring that it generalizes well to new data. If the model's performance is not satisfactory, further fine-tuning and adjustments to the preprocessing steps or hyperparameters are made.

5. Model Interpretation

- **Feature Importance:** The Random Forest Regressor provides insights into feature importance, showing which factors have the most significant influence on programme demand. This information helps in understanding the key drivers behind student preferences and assists colleges in strategic planning.
- **Visualization:** Data visualization techniques are employed to represent the model's predictions and feature importances. For example, bar charts can illustrate the importance of each factor, while heatmaps can show correlations between features.

6. Deployment :

- **Model Integration:** The trained model is integrated into a user-friendly application or platform, where college administrators can input relevant data to receive demand predictions for various programs.

- **Real-Time Predictions:** The system is designed to accommodate new data, allowing for real-time updates and predictions. This feature is crucial for responding to dynamic changes in factors like job market trends or sudden natural disasters.
- **User Interface:** An intuitive user interface is developed to allow college stakeholders to interact with the model. The interface displays predictions, insights into key factors influencing demand, and allows for scenario analysis by adjusting input parameters.

7. Post-Deployment Monitoring and Maintenance

- **Continuous Monitoring:** The model's performance is continuously monitored to ensure it remains accurate and relevant. Regular updates to the dataset (e.g., new feedback, changes in job market trends) are used to retrain and refine the model.
- **Model Retraining:** Periodic retraining of the model is conducted to incorporate new data and maintain its predictive accuracy. This involves repeating the training and evaluation steps with the latest data.
- **Feedback Loop:** A feedback loop is established to gather input from users (e.g., college administrators) on the model's predictions. This feedback helps in further refining the model and improving its accuracy and usability.

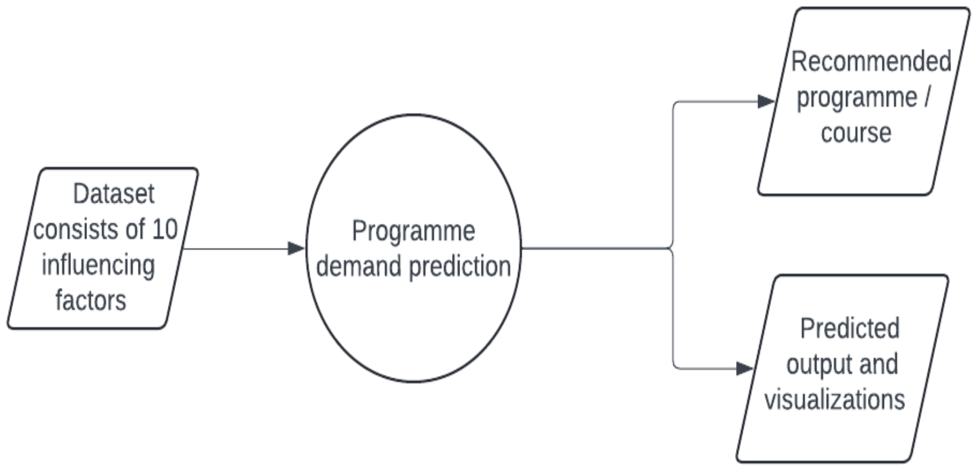
8. Outcome and Insights

- **Strategic Planning:** The final model provides colleges with actionable insights into future programme demand, aiding in resource allocation, curriculum development, and strategic decision-making.
- **Scenario Analysis:** Colleges can use the model to simulate different scenarios, such as changes in job market trends or infrastructure improvements, to understand their potential impact on programme demand.
- **Policy Implications:** The insights from the model can also guide policymakers in shaping educational strategies that align with market and societal needs.

By following this detailed process, the project aims to develop a robust and reliable predictive model that assists colleges in understanding and responding to the complex factors influencing programme demand. This data-driven approach enables more strategic, informed decision-making, ensuring that educational offerings remain relevant and aligned with future trends.

4.2 SYSTEM FLOW DIAGRAM

The system flow diagram is one of the graphical representations of the flow of data in a system. The diagram consists of several steps that identify where the input is coming to the system and output going out of the system. With the help of the diagram, it is possible to control the event decisions of the system and how data is flowing to the system. Therefore, the system flow diagram is basically a visual representation of data flow, excluding the minor parts and including the major parts of the system in a sequential manner.

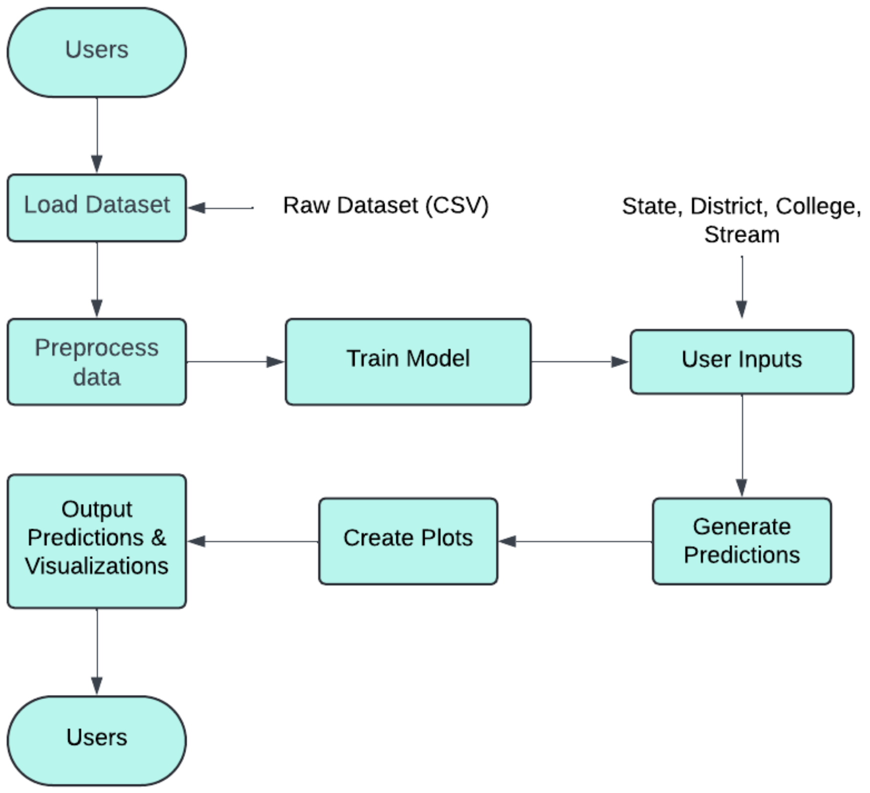

Dataset consists of 10 influencing factors
Programme demand prediction
Recommended programme / course
Predicted output and visualizations

The 10 influencing factors are : job market level, education level based on geographical location, technological advancements, political stability, natural disasters, infrastructure (like e-learning facilities), demographic data (educational, social, and economic status), feedback from different stakeholders, core industry performance, and trade association reports.

By incorporating these diverse and influential factors, the model aims to provide a robust and dynamic prediction of programme demand. This will assist educational institutions in strategic planning, programme offerings, and resource allocation, thereby aligning academic programmes with market and societal needs.

Finally we can able to view the predicted output which is a visualization graph consists of how many percentage that all the 10 factors influencing a programme demand. Another output also shows the all over percentage of particular programme demand.

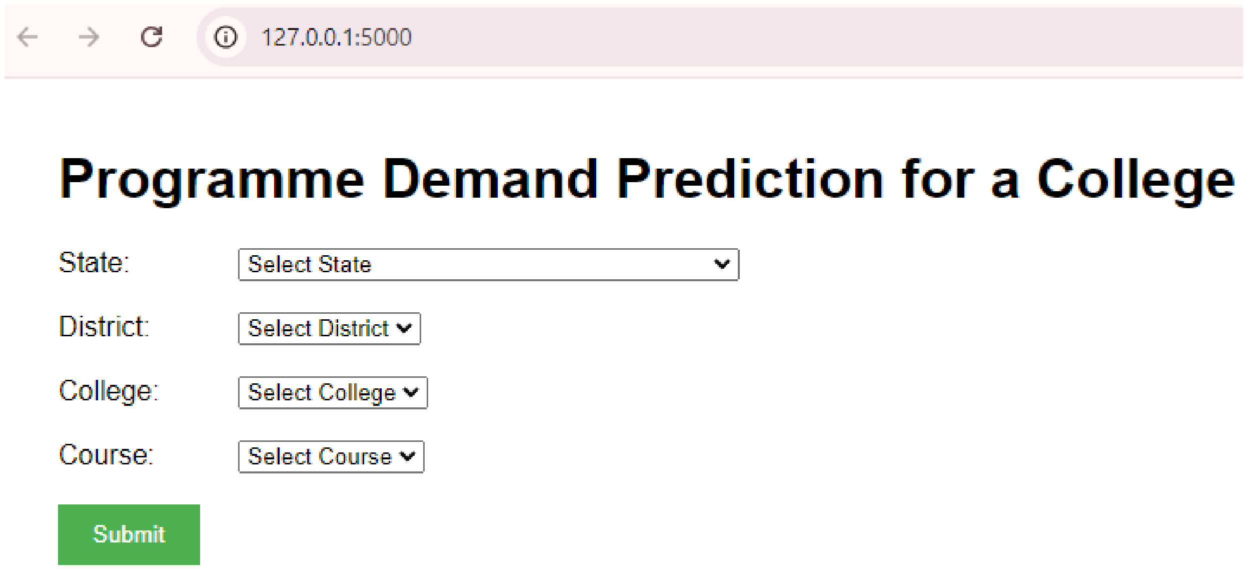
Steps to get a predicted output :



- **Data Collection and Integration:** Data from multiple sources are collected and integrated into a unified dataset.
- **Data Preprocessing:** The integrated dataset undergoes cleaning, normalization, feature engineering, and feature selection.
- **Model Development:** The Random Forest Regressor algorithm is used to train the model on the preprocessed data, with hyperparameter tuning to optimize performance.
- **Model Evaluation:** The model's performance is evaluated using various metrics to ensure its accuracy and reliability.
- **Model Interpretation:** The importance of features is analyzed, and results are visualized for better understanding.
- **Deployment:** The model is integrated into an application or platform, allowing users to input data and receive predictions in real-time.
- **Outcome and Insights:** The final model provides actionable insights and recommendations for colleges to make informed decisions regarding programme demand.

4.3 INPUT DESIGN

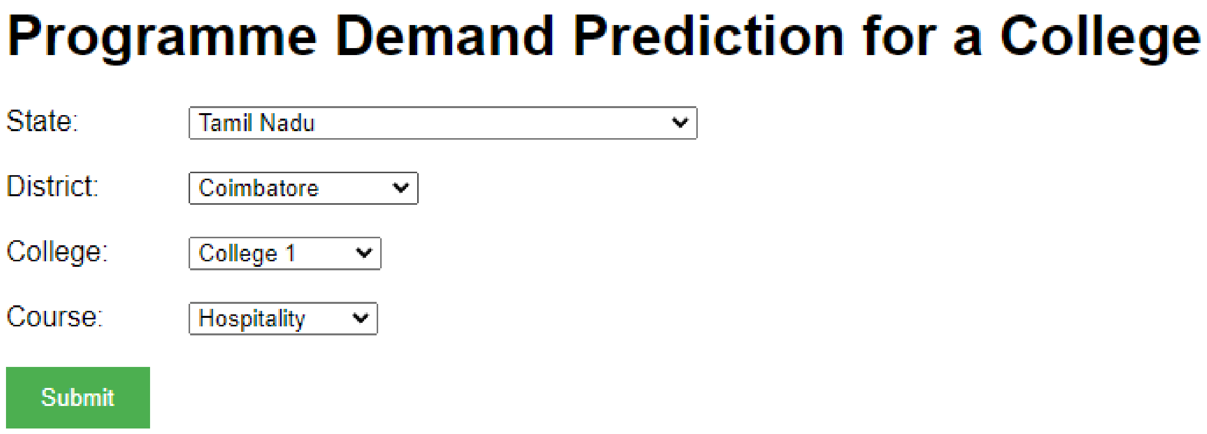
Input design is the process of converting user-originated inputs to a computer-based format . Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system.



127.0.0.1:5000

Programme Demand Prediction for a College

State:	Select State
District:	Select District
College:	Select College
Course:	Select Course



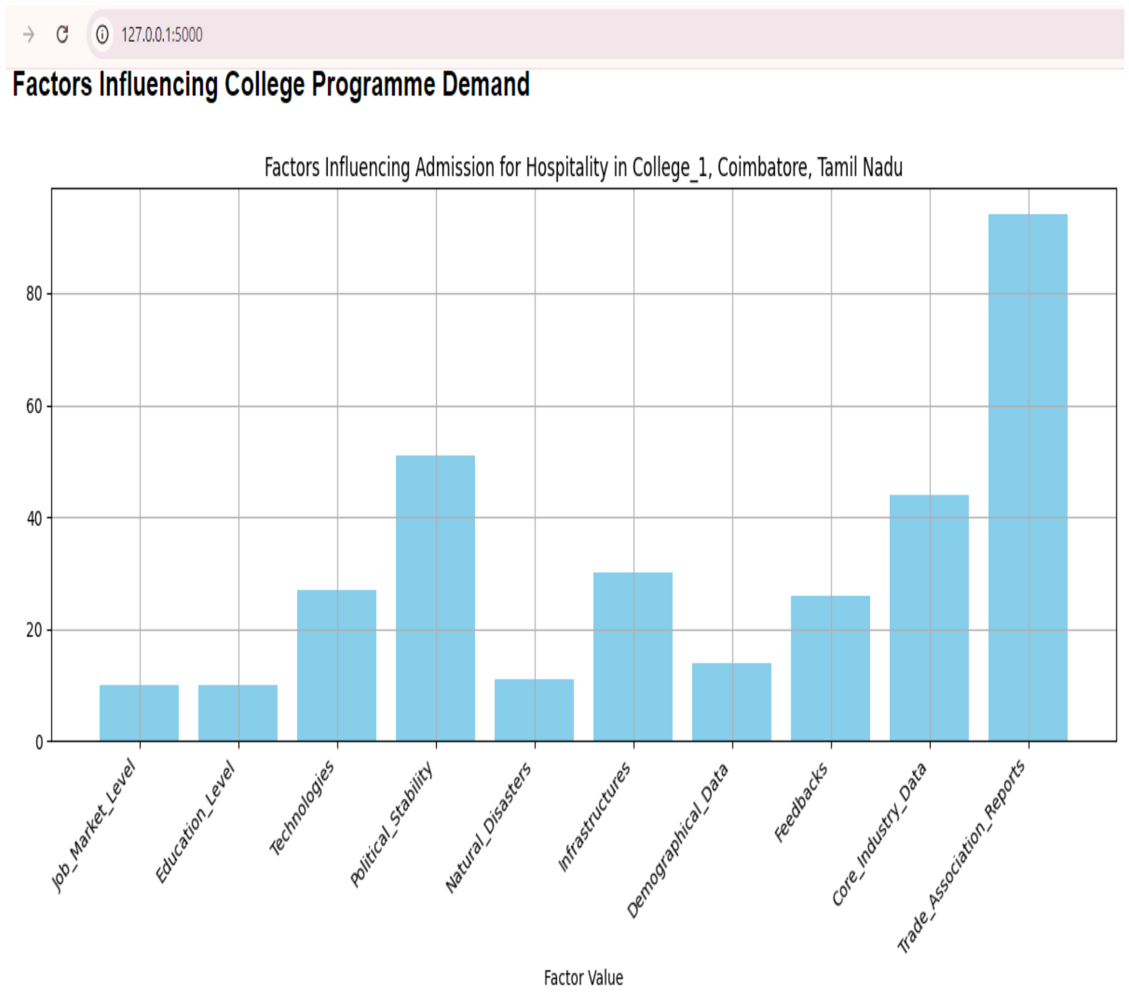
Programme Demand Prediction for a College

State:	Tamil Nadu
District:	Coimbatore
College:	College 1
Course:	Hospitality

4.4 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. The output is designed in such a way that it is attractive, convenient and informative.

As the outputs are the most important sources of information to the users, better design should improve the system's relationships with us and also will help in decision-making. Form design elaborates the way output is presented and the layout available for capturing information.



CHAPTER - 5
SYSTEM IMPLEMENTATION AND TESTING

5.1 SYSTEM IMPLEMENTATION :

System implementation is the process of :

1. Defining how the information system should be built (i.e., physical system design),

2. Ensuring that the information system is operational and used,

3. Ensuring that the information system meets quality standard (i.e., quality assurance).

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective. Implementation of a modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

The system for "Programme Demand Prediction for a College" is implemented using the Random Forest Regressor algorithm. Data is first collected from various sources, including educational databases, industry reports, and feedback surveys, and then preprocessed to handle missing values, outliers, and feature selection. The Random Forest model is trained on this cleaned data to predict programme demand for different colleges and streams. Post model development, the system is deployed as an application, allowing real-time predictions based on new inputs, and continuously monitored for performance improvements with updates based on new data and feedback.

5.2 SYSTEM TESTING

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the testdata, specially designed to show that the system will operate successfully in all its aspects conditions. The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated.

Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. The final step involves Validation testing, which determines whether the software function as the user expected.

The end-user rather than the system developer conduct this test most software developers as a process called “Alpha and Beta test ” to uncover that only the end users seems able to find. This is the final step in system life cycle.

Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

5.2.1 UNIT TESTING:

Unit testing verification efforts on the smallest unit of software design, module. This is known as “Module Testing”. The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

Test Case : Data Preprocessing

Test Data :

```
raw_data = {
    'job_market_level': [1, 2, None, 4],
    'education_level': ['low', 'medium', 'high', 'low'],
    'technologies_developed': [None, 2, 3, 4]
}
```

Test result :

```
processed_data = preprocess_data(raw_data)
# Expected output
{
    'job_market_level': [1, 2, 0, 4],  # Missing values replaced with 0 or mean
    'education_level': [0, 1, 2, 0],  # Categorical encoding
    'technologies_developed': [0, 2, 3, 4]  # Missing values replaced
}
```

5.2.2 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

5.2.3 VALIDATION TESTING:

To uncover functional errors, that is, to check whether functional characteristics confirm to specification or not specified.

Test Case : Accuracy of Predictions (Validation Against Actual Data)

Test Data :

```
X_test = [[4, 5, 6], [2, 3, 4], [7, 8, 9]]
y_test_actual = [400, 150, 600]  # Real-world demand
```

Test Result :

```
predictions = model.predict(X_test)
r2_score_value = evaluate_model(model, X_test, y_test_actual)
# Expected output: The R² score should be at least 0.7 (or equivalent measure)
assert r2_score_value > 0.
```

5.3 SYSTEM MAINTENANCE

The objectives of this maintenance work are to make sure that the system gets into work all time without any bug. Provision must be for environmental changes which may affect the computer or software system. This is called the maintenance of the system. Nowadays there is the rapid change in the software world. Due to this rapid change, the system should be capable of adapting these changes.

Maintenance plays a vital role. The system liable to accept any modification after its implementation. This system has been designed to favor all new changes. Doing this will not affect the system's performance or its accuracy.

In the project system testing is made as follows :

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. Then the web form level testing is made.

This is the final step in system life cycle. Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization

CHAPTER – 6
SCOPE FOR FUTURE ENHANCEMENTS

- **Incorporation of Additional Data:** Integrating more data sources, such as real-time economic indicators, labor market trends, and regional policy changes, can improve the model's accuracy and provide a more comprehensive view of demand drivers.
- **Advanced Machine Learning Techniques:** Exploring advanced algorithms like Gradient Boosting, XGBoost, or Neural Networks could enhance prediction performance and capture more complex patterns in the data.
- **Feature Engineering:** Developing additional features or transforming existing ones, such as creating interaction terms or using domain-specific knowledge, may reveal deeper insights and improve model performance.
- **Temporal Analysis:** Including time-series data to analyze trends over time could help in understanding how demand patterns evolve and predicting future changes more accurately.
- **Geographical Granularity:** Increasing the geographical resolution of the data, such as analyzing demand at a more granular level within districts, could provide more localized insights and aid in targeted decision-making.
- **User Feedback Integration:** Incorporating real-time feedback from students and industry stakeholders into the model can ensure that the predictions remain relevant and responsive to changing preferences.
- **Scenario Analysis:** Implementing scenario-based forecasting to assess the impact of potential future events or policy changes on programme demand can help institutions prepare for various contingencies.
- **Visualization Tools:** Developing advanced visualization tools and dashboards to present the predictions and insights in a user-friendly manner can facilitate better decision-making for educational planners.
- **Automated Updating:** Creating an automated system for regularly updating the model with new data can ensure that predictions remain current and reflect the latest trends and conditions.

These enhancements could significantly improve the model's accuracy, usability, and adaptability, providing more valuable insights for educational institutions and stakeholders.

CHAPTER - 7
CONCLUSION

The project utilizes the Random Forest Regressor to predict college programme demand, leveraging a comprehensive dataset that includes geographical, educational, and socio-economic factors. The model effectively forecasts demand trends, demonstrating the algorithm's capacity to handle complex, multi-factorial data. Key factors such as job market conditions, education levels, technological advancements, and infrastructure quality are crucial in shaping programme demand.

The analysis reveals that programme preferences are influenced by a combination of these factors rather than any single variable. The predictive insights offered by the model enable educational institutions to make informed decisions on programme offerings and resource allocation.

Additionally, the project highlights the importance of adaptability to external changes, such as natural disasters and industry shifts. By anticipating these factors, colleges can better manage enrollment and enhance their strategic planning.

The project's results provide a robust foundation for future enhancements and refinements, potentially incorporating additional data or advanced techniques to further improve predictive accuracy. Overall, this project underscores the value of machine learning in educational planning and offers actionable insights to optimize programme demand strategies

CHAPTER - 8
BIBILIOGRAPHY

REFERENCES :

- **Books and Texts**

  - Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.
  - Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

- **Research Articles**

  - Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. DOI: 10.1023/A:1010933404324.
  - Zhang, H., & Singer, B. (2010). Recursive partitioning and applications. *Springer Series in Statistics*.

- **Reports and Guidelines**

  - All India Survey on Higher Education (AISHE). (2020). *Ministry of Education, Government of India*. Retrieved from https://aishe.nic.in
  - National Skill Development Corporation (NSDC). (2021). *Annual Report*. Retrieved from https://nsdcindia.org

- **Websites and Online Resources**

  - World Bank. (2022). Education Overview. Retrieved from https://www.worldbank.org/en/topic/education/overview
  - National Association of Colleges and Employers (NACE). (2023). Job Market Trends. Retrieved from https://www.naceweb.org

- **Conference Papers**

  - Gupta, A., & Sharma, R. (2022). Predicting Higher Education Programme Demand Using Machine Learning Techniques. In *Proceedings of the International Conference on Education and Data Science* (pp. 112-120). IEEE.

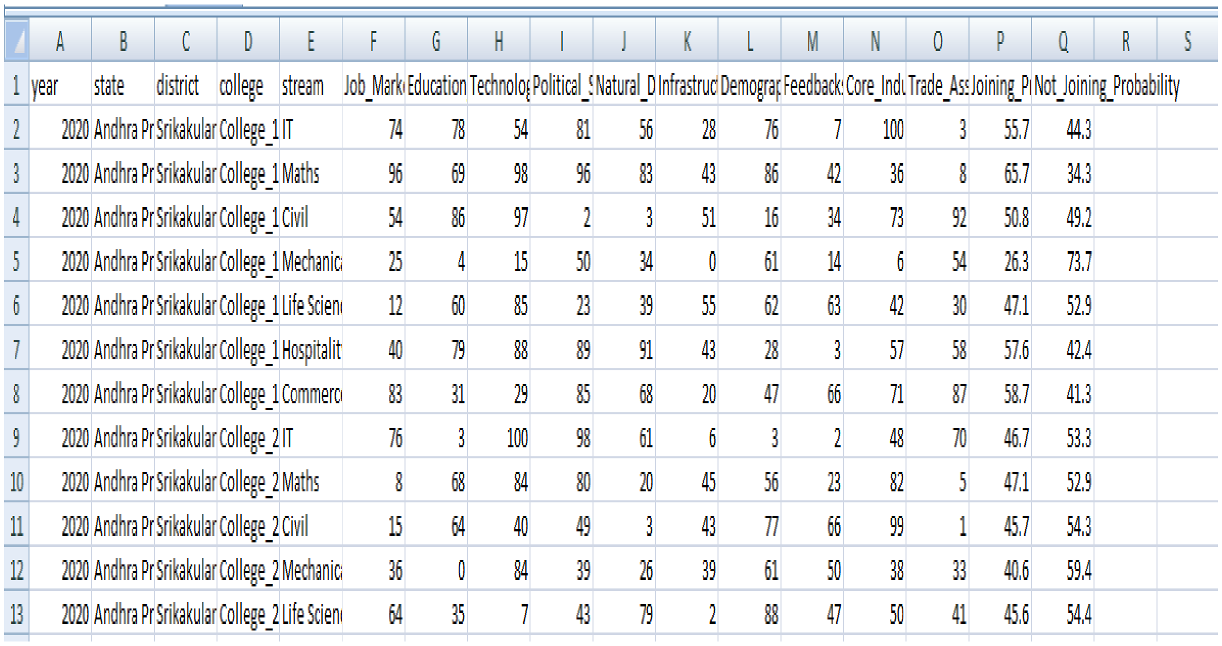
- **Theses and Dissertations**

  - Kumar, S. (2021). *Analysis of Factors Influencing College Programme Demand in India* (Master's thesis). University of Delhi.

CHAPTER - 9
APPENDIX

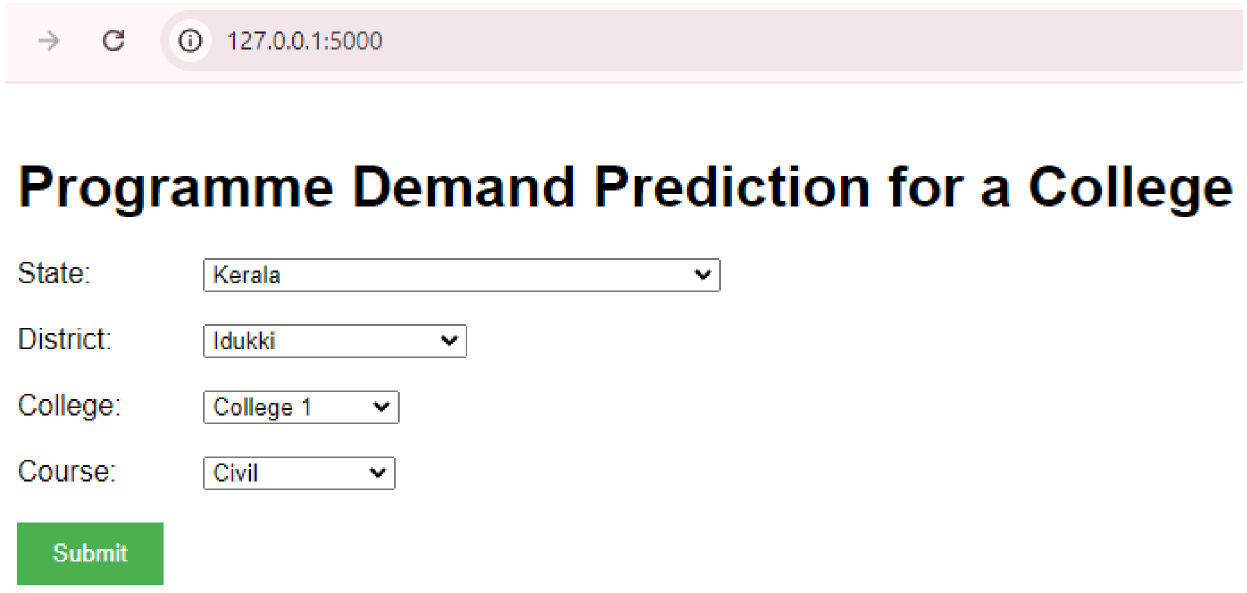
A. SCREENSHOTS

Dataset :



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	year	state	district	college	stream	Job_Mark	Education	Technolog	Political_S	Natural_D	Infrastruc	Demograp	Feedback	Core_Indu	Trade_Ass	Joining_P	Not_Joining_Probability		
2	2020	Andhra Pr	Srikakular	College_1	IT	74	78	54	81	56	28	76	7	100	3	55.7	44.3		
3	2020	Andhra Pr	Srikakular	College_1	Maths	96	69	98	96	83	43	86	42	36	8	65.7	34.3		
4	2020	Andhra Pr	Srikakular	College_1	Civil	54	86	97	2	3	51	16	34	73	92	50.8	49.2		
5	2020	Andhra Pr	Srikakular	College_1	Mechanica	25	4	15	50	34	0	61	14	6	54	26.3	73.7		
6	2020	Andhra Pr	Srikakular	College_1	Life Scien	12	60	85	23	39	55	62	63	42	30	47.1	52.9		
7	2020	Andhra Pr	Srikakular	College_1	Hospitalit	40	79	88	89	91	43	28	3	57	58	57.6	42.4		
8	2020	Andhra Pr	Srikakular	College_1	Commerc	83	31	29	85	68	20	47	66	71	87	58.7	41.3		
9	2020	Andhra Pr	Srikakular	College_2	IT	76	3	100	98	61	6	3	2	48	70	46.7	53.3		
10	2020	Andhra Pr	Srikakular	College_2	Maths	8	68	84	80	20	45	56	23	82	5	47.1	52.9		
11	2020	Andhra Pr	Srikakular	College_2	Civil	15	64	40	49	3	43	77	66	99	1	45.7	54.3		
12	2020	Andhra Pr	Srikakular	College_2	Mechanica	36	0	84	39	26	39	61	50	38	33	40.6	59.4		
13	2020	Andhra Pr	Srikakular	College_2	Life Scien	64	35	7	43	79	2	88	47	50	41	45.6	54.4		

Input :



127.0.0.1:5000

Programme Demand Prediction for a College

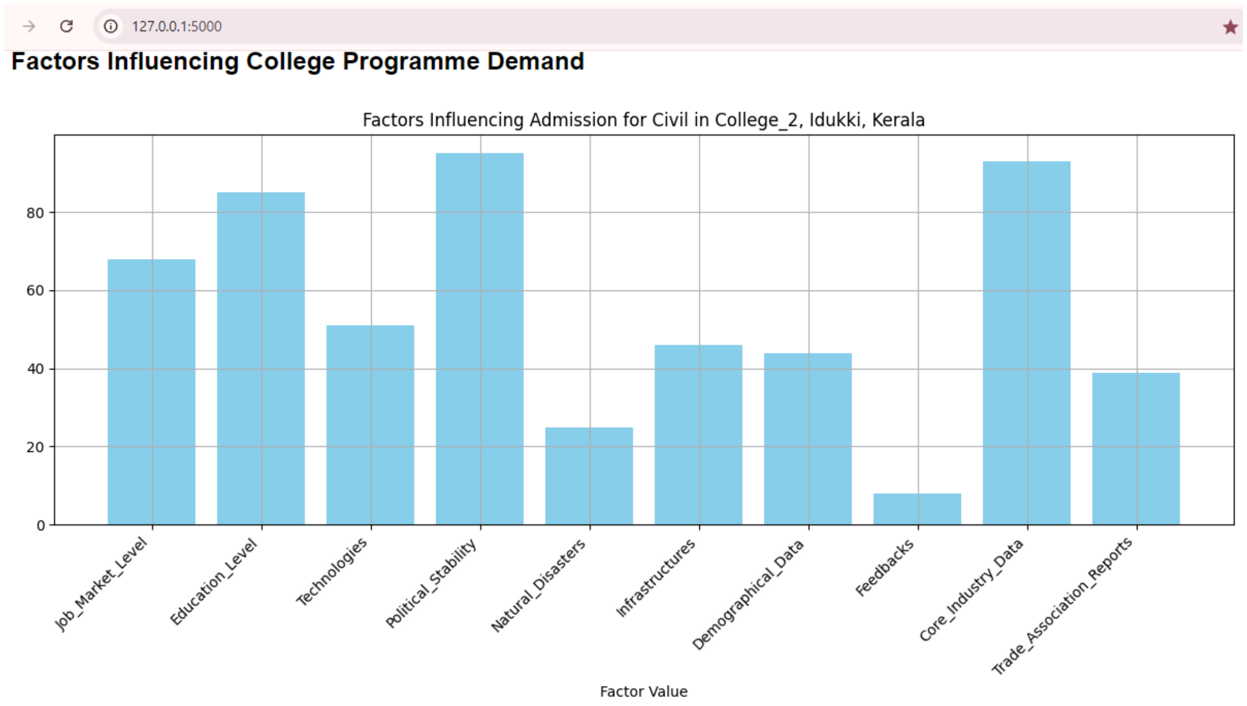
State: Kerala

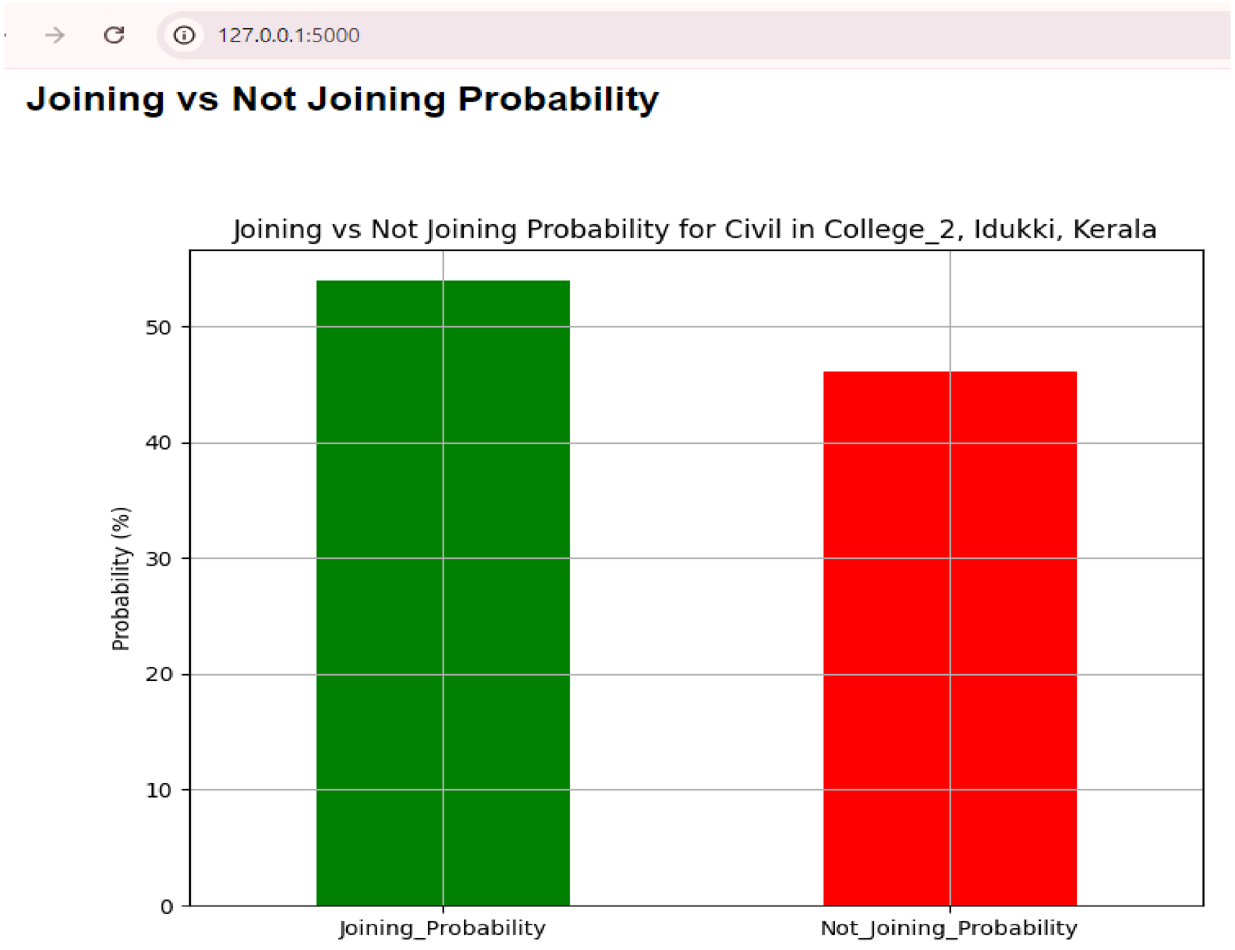
District: Idukki

College: College 1

Course: Civil

Output :


127.0.0.1:5000
Factors Influencing College Programme Demand
Factors Influencing Admission for Civil in College_2, Idukki, Kerala
0
20
40
60
80
Job_Market_Level
Education_Level
Technologies
Political_Stability
Natural_Disasters
Infrastructures
Demographical_Data
Feedbacks
Core_Industry_Data
Trade_Association_Reports
Factor Value


127.0.0.1:5000
Joining vs Not Joining Probability
Joining vs Not Joining Probability for Civil in College_2, Idukki, Kerala
Probability (%)
0
10
20
30
40
50
Joining_Probability
Not_Joining_Probability

B. SAMPLE CODING

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admission Probability</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Programme Demand Prediction for a College</h1>
    <form method="POST">
        <label for="state">State:</label>
        <select id="state" name="state" required>
            <option value="">Select State</option>
        </select><br><br>

        <label for="district">District:</label>
        <select id="district" name="district" required>
            <option value="">Select District</option>
        </select><br><br>

        <label for="college">College:</label>
        <select id="college" name="college" required>
        <option value="">Select College</option>

        <option value="College_1">College 1</option>

        <option value="College_2">College 2</option>

        <!-- Add more college options here if needed -->

    </select><br><br>


    <label for="course">Course:</label>

    <select id="course" name="course" required>

        <option value="">Select Course</option>

        <option value="IT">IT</option>

        <option value="Maths">Maths</option>

        <option value="Civil">Civil</option>

        <option value="Mechanical">Mechanical</option>

        <option value="Life Science">Life Science</option>

        <option value="Hospitality">Hospitality</option>

        <option value="Commerce">Commerce</option>

        <!-- Add more course options here if needed -->

    </select><br><br>


    <button type="submit">Submit</button>

</form>


{% if error %}

    <p>{{ error }}</p>

{% endif %}


{% if factor_graph %}
```

```
        });
    </script>
  </body>
  </html>
```

style.css

```
body {
      font-family: Arial, sans-serif;
      margin: 40px;
  }

  form {
      margin-bottom: 20px;
  }

  label {
      display: inline-block;
      width: 100px;
  }

  input {
      margin-bottom: 10px;
  }

  button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}
```

app.py

```
from flask import Flask, request, render_template
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)

# Load the data
file_path = 'college_joining_probabilities_5_years.csv'
data = pd.read_csv(file_path)
```

```
# Define the feature columns and target columns
feature_columns = ['Job_Market_Level', 'Education_Level', 'Technologies', 'Political_Stability',
                  'Natural_Disasters', 'Infrastructures', 'Demographical_Data', 'Feedbacks',
                  'Core_Industry_Data', 'Trade_Association_Reports']
target_columns = ['Joining_Probability', 'Not_Joining_Probability']

# Separate the features and targets
X = data[feature_columns]
y = data[target_columns]
# Standardize the feature columns
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train the RandomForestRegressor on all data
model = RandomForestRegressor(random_state=42)
model.fit(X_scaled, y)



def get_admission_probability(state, district, college, course):
    # Filter data for the given state, district, college, and course
    filtered_data = data[(data['state'] == state) & (data['district'] == district) &
                         (data['college'] == college) & (data['stream'] == course)]
    if filtered_data.empty:
        return None, None, "No data available for the specified input."

    # Get the feature values
```

```
features = filtered_data[feature_columns]

# Standardize the features

features_scaled = scaler.transform(features)


# Predict probabilities

probabilities = model.predict(features_scaled)

# Add predictions to the filtered data

filtered_data['Joining_Probability'] = probabilities[:, 0]

filtered_data['Not_Joining_Probability'] = probabilities[:, 1]


# Plotting the factors

factor_values = filtered_data[feature_columns].iloc[0].values

factor_names = feature_columns

factor_values = [float(value) for value in factor_values]  # Convert to float

plt.figure(figsize=(12, 6))

plt.bar(factor_names, factor_values, color='skyblue')

plt.xlabel('Factor Value')

plt.title(f'Factors Influencing Admission for {course} in {college}, {district}, {state}')

plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels to avoid overlap

plt.grid(True)

plt.tight_layout()  # Adjust layout to make sure everything fits

img = io.BytesIO()

plt.savefig(img, format='png')

img.seek(0)

factor_graph = base64.b64encode(img.getvalue()).decode()

# Plotting the joining and not joining probabilities

probabilities_df = filtered_data[['Joining_Probability', 'Not_Joining_Probability']].iloc[0].astype(float)
```

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

    probabilities_df.plot(kind='bar', color=['green', 'red'])

    plt.ylabel('Probability (%)')

    plt.title(f'Joining vs Not Joining Probability for {course} in {college}, {district}, {state}')

    plt.xticks(rotation=0)

    plt.grid(True)

    img = io.BytesIO()

    plt.savefig(img, format='png')

    img.seek(0)

    probability_graph = base64.b64encode(img.getvalue()).decode()

    return factor_graph, probability_graph, None

@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        state = request.form['state']

        district = request.form['district']

        college = request.form['college']

        course = request.form['course']

        factor_graph, probability_graph, error = get_admission_probability(state, district, college, course)

        return render_template('index.html', factor_graph=factor_graph, probability_graph=probability_graph,

                               error=error)

    return render_template('index.html', factor_graph=None, probability_graph=None, error=None)




if __name__ == '__main__':

    app.run(debug=True)
```