ADMISSION PREDICTION

AN INTERNSHIP REPORT SUBMITTED IN A PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

By

G. Durga Bhavani(21JG1A1216) G. Haritha Sri(21JG1A1217)
K.B.S. Anusha(21JG1A1226)

Under the esteemed guidance of

DINESH KUMAR HIRAWAT

CEO

HMI ENGINEERING SERVICES

Department of Information Technology

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

Kommadi, Madhurawada, Visakhapatnam 530 048

(Approved by AICTE, New Delhi, Affiliated to Andhra University, Visakhapatnam)

(Accredited by National Board of Acciditation [NBA] for B.Tech CSE, ECE and IT - valid from 2019-22 and 2022-25)

(Accredited by National Assesment and Accreditation Council [NAAC] with A Grade - valid from 2022-2027)

AY:2023– 2024

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN

Kommadi, Madhurawada, Visakhapatnam 530 048

(Approved by AICTE, New Delhi, Affiliated to Andhra University, Visakhapatnam)

(Accredited by National Board of Acciditation [NBA] for B.Tech CSE, ECE and IT - valid from 2019-22 and 2022-25)

(Accredited by National Assesment and Accreditation Council [NAAC] with A Grade - valid from 2022-2027)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the internship project report on **"ADMISSION PREDICTION"** is a bonafide work of following **III B.Tech** student in the **Department of Information Technology**, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2023-2024 in the fulfillment of the requirement for the award of the degree of Bachelor of Technology of this university.

By

G. Durga Bhavani(21JG1A1216) G. Haritha Sri(21JG1A1217)
K.B.S. Anusha(21JG1A1226)

Mr. G. APPAJI	**Dr. M. BHANU SRIDHAR**
Assistant Professor	Professor
(Internal Guide)	**(Head of the Department)**

(External Examiner)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **G.Appaji,** Assistant Professor for encouragement all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We express our deep sense of gratitude and thanks to **Dr. M.Bhanu Sridhar**, Professor and Head of the Department of Computer Science and Engineering for her guidance and for expressing her valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to Vice Principal, **Dr. G. Sudheer** for allowing us to utilize the college resources thereby facilitating the successful completion of our project. We are also thankful to both teaching and non-teaching faculty of the Department of Computer Science and Engineering for giving valuable suggestions for our project.

We would like to take the opportunity to express our profound sense of gratitude to the revered Principal, **Dr. R. K. Goswami** for all the help and support towards the successful completion of our project.

TABLE OF CONTENTS

ABSTRACT

The admission process for higher education institutions is a critical and often opaque decision-making procedure, impacting the future of countless students. This study aims to develop a predictive model for college admission outcomes by examining a diverse array of factors that influence the admissions process.

We collected extensive data from numerous universities, including applicant demographics, standardized test scores, high school transcripts, extracurricular activities, letters of recommendation, and personal essays. This data was then analyzed to identify patterns and relationships between these factors and admission decisions.

Our analysis includes the use of machine learning algorithms, such as logistic regression, decision trees, and neural networks, to create predictive models. Additionally, we employ data visualization techniques to enhance our understanding of feature importance and patterns in the admission process.

Results from our models demonstrate the significance of various factors in predicting admission outcomes, shedding light on the relative importance of academic achievements, extracurricular involvement, and other personal attributes. Furthermore, this study discusses the implications of these findings on the broader context of college admissions, emphasizing the need for transparency and fairness in the process.

Ultimately, this research contributes to a better understanding of the factors that influence college admission decisions and provides valuable insights for both students and institutions. It advocates for a more holistic and equitable approach to the admissions process, potentially reducing biases and enhancing the chances of students to secure a spot in the educational institutions of their choice.

Introduction

In this era of the internet,

The process of college or university admissions is a pivotal moment in the lives of countless students, as it determines their access to higher education and, subsequently, their career prospects. Over the years, this process has become increasingly competitive, making it crucial for both institutions and applicants to gain insights into the factors that influence admission outcomes. Admission prediction, a field that combines data analysis, machine learning, and educational research, plays a vital role in demystifying this complex process.

Admission prediction involves using historical data and statistical methods to forecast the likelihood of an applicant's acceptance into an educational institution. These predictions are based on a wide range of factors, including academic performance, standardized test scores, extracurricular activities, letters of recommendation, personal essays, and demographic information. By analyzing past admission data, a model can be created to estimate the probability of acceptance for future applicants.

This field has gained prominence due to several factors. First, the rising number of college applicants and the limited number of available spots in prestigious institutions have made it essential for institutions to efficiently identify the most suitable candidates. Second, students and their families are investing significant time and resources into the college application process and desire more transparency and predictability.

Admission prediction models are not only beneficial for colleges and universities but also provide students with valuable insights. They help applicants make informed decisions about where to apply, understand the strength of their applications, and strategize for their future.

In this dynamic landscape, the use of machine learning and data analytics has revolutionized admission prediction, offering more accurate and nuanced insights. However, it is essential to address concerns related to fairness, equity, and transparency in this process, as biased models or unethical practices can undermine the integrity of admissions.

This introduction sets the stage for a more in-depth exploration of admission prediction, its methodologies, challenges, and the evolving role it plays in shaping the future of higher education. Predicting admissions not only serves as a tool for decision-making but also advocates for a more equitable and transparent admission process.

PROBLEM STATEMENT:

Admission Prediction for college , university process involves historical data , carrer prospects and etc....

1.2 Objectives:

People can understand the Admission process so that they can choose the best institute to build there own carrier. So, the algorithm makes the easy to predict the admission process.

2. Literature survey

Factors Influencing Admission

Academic performance (GPA, standardized test scores).

- Extracurricular activities.
- Letters of recommendation
- Personal statements or essays
- Demographic information (e.g., ethnicity, socio-economic status)
- Interview performance (if applicable)

Any other relevant factors identified in the literature.

Existing system:

- Determine the primary objectives of the admission prediction system.
- Define the scope of the system, including the specific types of admissions it will predict (e.g., college, graduate school, job applications).

3.System Analysis

- **Gather Requirements:**
  - Conduct interviews and surveys with stakeholders (e.g., admissions officers, students, faculty) to gather detailed requirements.
  - Identify the key features, functionalities, and performance expectations of the system.
- **Identify Data Sources:**
  - Determine the sources of data required for admission prediction. This could include academic records, standardized test scores, application essays, recommendation letters, etc.
- **Feature Selection and Engineering:**
  - Identify relevant features that will be used for prediction (e.g., GPA, test scores, extracurricular activities).
  - Consider creating new features or transforming existing ones to enhance predictive power.
  - Determine if the admission prediction system needs to integrate with other existing systems student information systems, CRM systems

3.1 SYSTEM REQUIREMENTS

Hardware Requirements

• System	: MINIMUM i3.
• Hard Disk	: 40 GB.
• Ram	: 4 GB.

Software Requirements:

- **Operating System:** Windows 11
- **Coding Language**: Python 3.7

PYTHON:

Python is a high level general purpose open source programming language. It is both object oriented and procedural. Python is an extremely powerful language. This language is very easy to learn and is a good choice for most of the professional programmers.

Python is invented by **Guido Van Rossum** at CWI in Netherland in 1989. It is binding of **C, C++, and Java**. It also provides a library for GUI.

Python Features and Characteristics:

- Python is a high level, open source, general purpose programming language.
- It is object oriented, procedural and functional.
- It has library to support **GUI**.
- It is extremely powerful and easy to learn.
- It is open source, so free to available for everyone.
- It supports on **Windows, Linux** and **Mac OS**.
- As code is directly compiled with byte code, python is suitable for use in scripting languages.
- Python enables us to write clear, logical applications for small and large tasks.
- It has high level built-in data types: **string, lists, dictionaries** etc.
- Python has a set of useful Libraries and Packages that minimize the use of code in our day to day life, like- **Django, Tkinter, OpenCV, NextworkX, lxml.**
- It plays well with java because of jython. Jython is a version of Python.
- It encourages us to write clear and well structured code.
- It is easy to interface with C, Objective C, java, etc.
- We can easily use Python with other languages. It has different varieties of languages like-
    - **CPython –** Python implemented in C.
    - **Jython –** Python implemented in java Environment.
    - **PYPY –** Python with JIT compiler and stackless mode.
    - **IronPython –** Python for Net and CLR.

WHY CHOOSE PYTHON:

If you're going to write programs, there are literally dozens of commonly used languages to choose from.

Why choose Python? Here are some of the features that make Python an appealing choice.

Python is popular:

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of Programming Language Index.

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you'd like to dig deeper into Python salary statistics and job opportunities, you can do so here.

Python is interpreted:

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly. This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

PYTHON is Free :

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, MACOS, smart phones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple:

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

For all its syntactical simplicity, Python supports most constructs that would be expected in a very high- level language, including complex dynamic data types, structured and functional programming, and object- oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Conclusion:

This section gave an overview of the Python programming language, including:

- A brief history of the development of Python
- Some reasons why you might select Python as your language of choice .

Python is a great option, whether you are a beginning programmer looking to learn the basics, an experienced programmer designing a large application, or anywhere in between. The basics of Python are easily grasped, and yet its capabilities are vast. Proceed to the next section to learn how to acquire and install Python on your computer.

Python is an open source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Python use [change / change source]:-

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development
- Scientific programming
- Desktop GUIs
- Network programming
- Game programming

Jupyter Notebook:

Jupyter Notebook can be a valuable tool in various aspects of admission prediction in educational institutions. Here are some ways it can be used:

1. Data Preprocessing and Cleaning:

Jupyter Notebook allows for interactive data cleaning and preprocessing. This is crucial as raw admission data might be messy and contain missing values or outliers.

2. Exploratory Data Analysis (EDA):

Jupyter Notebooks are excellent for performing EDA. You can create visualizations and conduct statistical analyses to understand the relationships between different admission features and the target variable (e.g., admission status).

3. Feature Engineering:

It allows you to create new features based on existing ones, which can improve the performance of your admission prediction model.

4. Model Building and Training:

Jupyter Notebooks provide an interactive environment for building and training machine learning models. You can use libraries like scikit-learn, TensorFlow, or PyTorch to develop predictive models.

5. Hyperparameter Tuning:

Grid search or random search for hyperparameters can be performed iteratively in a Jupyter Notebook. This helps in optimizing the performance of the predictive model.

6. Model Evaluation:

You can use Jupyter Notebooks to evaluate the performance of different models using metrics like accuracy, precision, recall, F1-score, etc. You can also visualize the results for better interpretation.

7. Model Interpretability:

Techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can be implemented in a Jupyter Notebook to gain insights into how the model is making predictions.

8. Deployment Prototyping:

While the actual deployment might happen in a different environment, Jupyter Notebooks can be used for prototyping and testing the model before it's deployed in a production setting.

9. Interactive Reports:

Jupyter Notebooks allow you to create interactive reports with widgets. This can be useful for creating dashboards or dynamic reports for stakeholders.

10. Documentation and Collaboration:

Jupyter Notebooks serve as an excellent platform for documenting the entire process of admission prediction. You can include code, visualizations, and explanations in a single document, making it easy to share with others.

11. Experiment Tracking:

Jupyter Notebooks can be integrated with tools like MLflow or TensorBoard for experiment tracking. This helps in keeping a record of different model iterations, their parameters, and performance metrics.

12. Automated Pipelines:

Using tools like Papermill, you can parameterize your Jupyter Notebooks and run them as part of an automated pipeline. This is particularly useful for retraining models with new data.

Key Features of Jupyter Notebook:

1. Interactive Environment: Jupyter Notebook provides an interactive computing environment where you can execute code cells one at a time. This is extremely useful for data exploration and iterative development.

2. Data Visualization: Jupyter supports a wide range of data visualization libraries such as Matplotlib, Seaborn, Plotly, etc. These libraries help you create meaningful plots and graphs to visualize the data and insights.

3. Data Exploration and Analysis: You can easily load, manipulate, and analyze data using Python libraries like Pandas, NumPy, and SciPy. These libraries allow you to perform tasks such as data cleaning, aggregation, statistical analysis, etc.

4. Machine Learning Libraries: Jupyter Notebook is compatible with popular machine learning libraries like scikit-learn, TensorFlow, PyTorch, and Keras. This enables you to build, train, and evaluate machine learning models for admission prediction.

5. Data Preprocessing: Techniques like one-hot encoding, feature scaling, handling missing values, and data normalization can be performed within the notebook using libraries like Scikit-learn or custom Python code.

6. Version Control and Collaboration: Jupyter Notebooks can be easily integrated with version control systems like Git. This facilitates collaboration with other team members and helps in tracking changes to the project.

7. Interactive Widgets: Jupyter allows you to create interactive widgets that can be used to control parameters or display dynamic information, which can be especially useful for visualizing model outputs or exploring data.

4.System Design

4.1 Introduction:

Admission prediction involves using historical data and statistical methods to forecast the likelihood of an applicant's acceptance into an educational institution. These predictions are based on a wide range of factors, including academic performance, standardized test scores, extracurricular activities, letters of recommendation, personal essays, and demographic information. By analyzing past admission data, a model can be created to estimate the probability of acceptance for future applicants.

This field has gained prominence due to several factors. First, the rising number of college applicants and the limited number of available spots in prestigious institutions have made it essential for institutions to efficiently identify the most suitable candidates. Second, students and their families are investing significant time and resources into the college application process and desire more transparency and predictability.

This field has gained prominence due to several factors. First, the rising number of college applicants and the limited number of available spots in prestigious institutions have made it essential for institutions to efficiently identify the most suitable candidates. Second, students and their families are investing significant time and resources into the college application process and desire more transparency and predictability.

Admission prediction models are not only beneficial for colleges and universities but also provide students with valuable insights. They help applicants make informed decisions about where to apply, understand the strength of their applications, and strategize for their future.

In this dynamic landscape, the use of machine learning and data analytics has revolutionized admission prediction, offering more accurate and nuanced insights. However, it is essential to address concerns related to fairness, equity, and transparency in this process, as biased models or unethical practices can undermine the integrity of admissions.

This introduction sets the stage for a more in-depth exploration of admission prediction, its methodologies, challenges, and the evolving role it plays in shaping the future of higher education. Predicting admissions not only serves as a tool for decision-making but also advocates for a more equitable and transparent admission process.

4.2 UML Diagrams

4.2.1 USE CASE DIAGRAM:

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

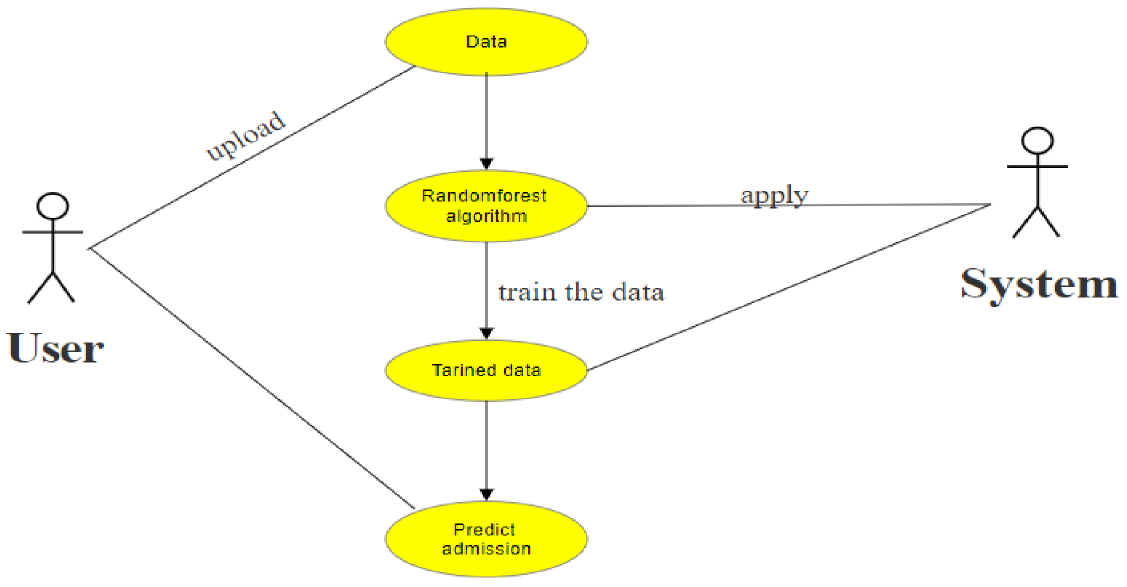


Fig: Usecase Diagram

4.2.2 CLASS DIAGRAM:

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

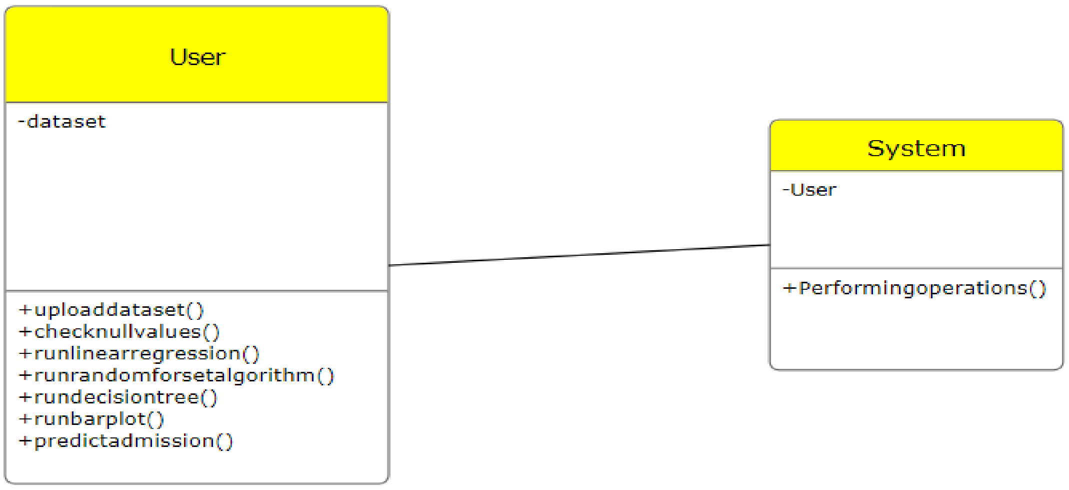


Fig:Class Diagram

4.2.3 SEQUENCE DIAGRAM:

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

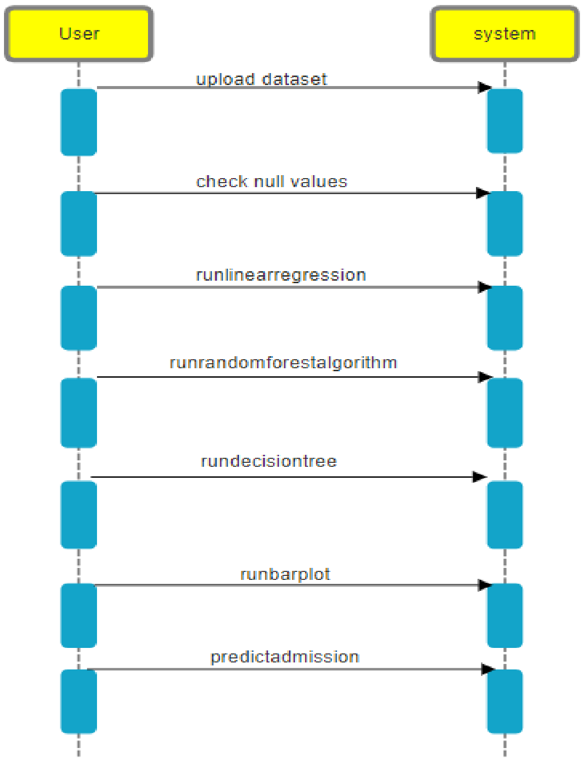


Fig: Sequence Diagram

4.2.4 ACTIVITY DIAGRAM:

An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very **similar to a flowchart**.

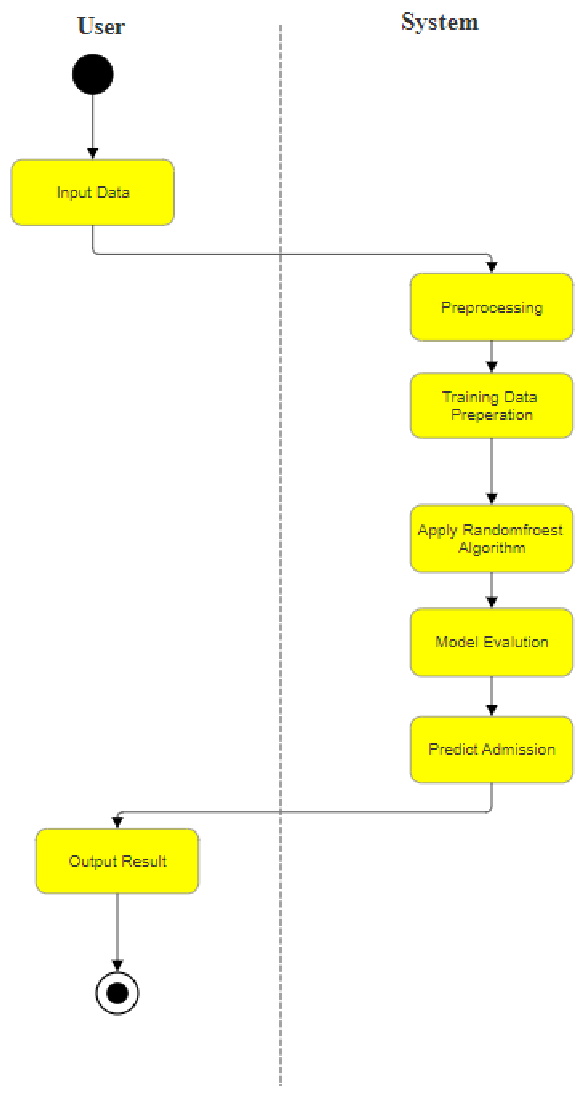


Fig:Activity diagram

4.3 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input ?

  How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

4. . Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input

process and show the correct direction to the management for getting correct information from the computerized system.

5. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

6. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

4.4 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following

objectives.

- Convey information about past activities, current status or projections .
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action .

5.Methodology

5.1 SYSTEM ARCHITECTURE:

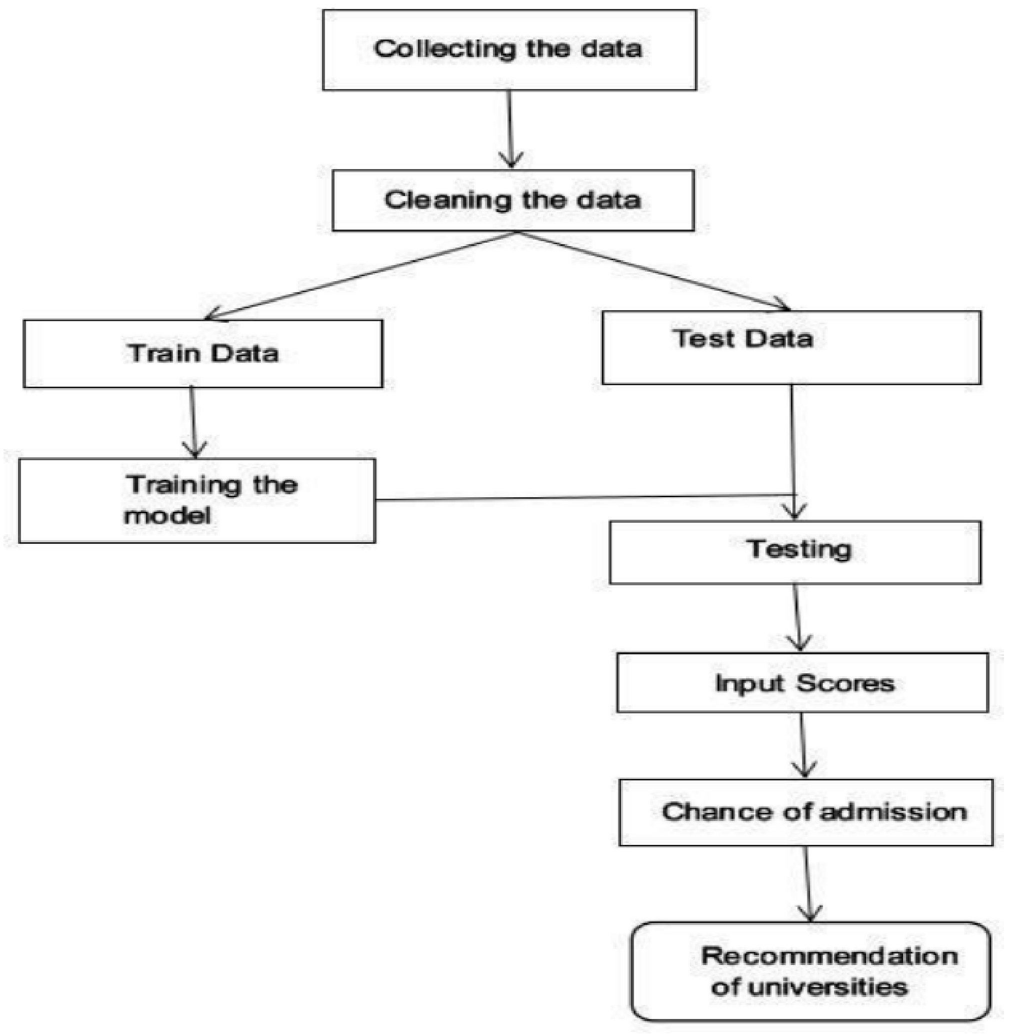


FIG 5.1: System Architecture

FLOW DIAGRAM:

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

to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

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

5.2 MODULES

- **Numpy:**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

- **Pandas:**
  Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library of Python. Pandas is fast and it has high performance & productivity for users.
- **Mathplotlib:**
  Python is the most used language for Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits like Tkinter, awxPython, etc.
- **Warnings:**
  A warning in a program is distinct from an error. Python program terminates immediately if an error occurs. Conversely, a warning is not critical. It shows some message, but the program runs. The warn() function defined in the 'warning' module is used to show warning messages. The warning module is actually a subclass of Exception which is a built-in class in Python.
- **Random:**
  Python Random module is an in-built module of Python that is used to generate random numbers in Python. These are pseudo-random numbers means they are not truly random. This module can be used to perform random actions such as generating random numbers, printing random a value for a list or string, etc.

- **Seaborn:**
  Seaborn is a library mostly used for statistical plotting in Python. It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.
- **Randomforestregressor:**

Random forest is an ensemble learning algorithm based on decision tree learners. The estimator fits multiple decision trees on randomly extracted subsets from the dataset and averages their prediction.Scikit-learn API provides the RandomForestRegressor class included in ensemble module to implement the random forest for regression problem.

A random forest regressor. A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

- **Stats:**
  The Scipy has a package or module scipy.stats that contains a huge number of statistical functions. Although statistics is a very broad area, here module contains the functions related to some of the major statistics.

5.3 ALGORITHMS USED

A.Linear Regression

Regression models are used to describe a relation between different variables by using the observed data into a line. Straight lines are used in linear regression models, whereas curved line is used in logistic and non-linear regression models. Linear regression model is a method used as response for only a single feature, it is based on supervised learning. Regression models always target a prediction value which is based on independent variables. It is used to calculate the relationship between two quantitative variables. Regression models differ upon – the type of relation between independent and dependent variables, the number of variables being used and the ones they are considering. It is taken under assumption that these variables are linearly related. Henceforth, we try to define a linear function that predicts the response value(y) as accurately as possible as a function ofthe feature or independent variable(x).cut-offs of the colleges, admission intake and preferences of students. Also, it helps students avoid spending time and money on counsellor and stressful research related to finding a suitable college.

B.Random Forests

The random forest is a machine learning algorithm which is widely used in regression and classification problems. Decision trees are built upon multiple different samples and then take their majority vote for average and bifurcation in case of regression. Random forest has the ability to handle a data set which contains continuous variables in case of regression and categorical variables in case of classification. Hence, it provides good results for classificationproblems.In industry lingo, reason behind forest works algorithm works so well is: Any huge quantity of moderately uncorrelated trees working as a body will outperform any of the individual constituent models

Implementation and Code with Comments

Sample Code:

```
import numpy as np              # linear algebra
import pandas as pd             # data processing, CSV file I/O (e.g. pd.read_csv)
import pandas as pd
Admission_Predict = pd.read_csv("/content/drive/MyDrive/admission_prediction.csv")
df=pd.read_csv("/content/drive/MyDrive/admission_prediction.csv")
df.head()
df.tail()
df['Chance of admit class']=df['Chance of Admit '].apply(lambda x:1 if x>0.80 else 0)
df.head()
print(' Shape of Data \n Rows :',df.shape[0],', Columns : ',df.shape[1])    ## Shape of data

        ## Checking for null values

missing_values = df.isnull().sum() * 100/len(df)
missing_values_df =
pd.DataFrame({'Column_name':df.columns,'Missing_percent':missing_values})
missing_values_df

            #iNFORMATION ABOUT THE DATA we can observe data doesnot have
any Null value

df.info()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
import seaborn as sns
import warnings
from scipy import stats
warnings.filterwarnings('ignore')

         # lets see the distribution for the target variable

print('Skewness of chance of admit : ',df['Chance of Admit '].skew())
plt.figure(figsize = (10,5))
sns.distplot(df['Chance of Admit '],kde = True,color = 'g',fit = stats.norm)
fig, ax = plt.subplots(figsize=(15,6))
rs = stats.probplot(df['Chance of Admit '],plot = ax)
plt.show()
```

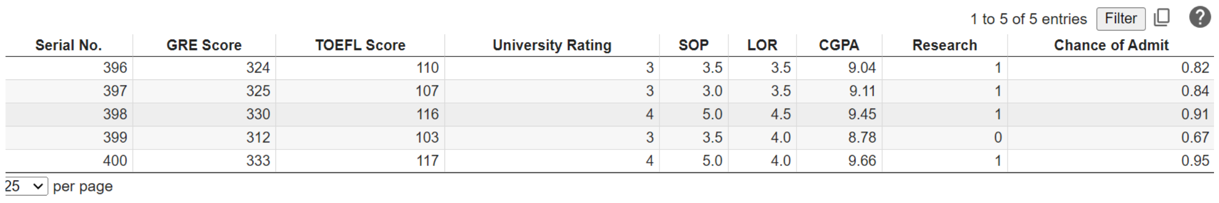
```
from sklearn.metrics import r2_score, mean_squared_error

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
features = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
      'LOR ', 'CGPA', 'Research']
X = df[features]
y = df['Chance of Admit '] X = df[features]
y = df['Chance of Admit ']
trainX ,testX , trainY, testY = train_test_split(X, y, train_size = 0.7,random_state = 5)
lin_reg = LinearRegression()
predY = lin_reg.fit(trainX,trainY).predict(testX)
m1 = r2_score(testY,predY)
print('Accuracy/RSquared : ',r2_score(testY, predY))
print('Root Mean Squared Error : ',np.sqrt(mean_squared_error(testY,predY)))
dec_tree = DecisionTreeRegressor()
predY = dec_tree.fit(trainX,trainY).predict(testX)


m2 = dec_tree.score(testX,testY)
print('Accuracy : ',dec_tree.score(testX, testY))
print('Root Mean Squared Error : ',np.sqrt(mean_squared_error(testY,predY)))
corr = df.corr()
sns.heatmap(corr,
          xticklabels=corr.columns.values,
          yticklabels=corr.columns.values)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV,train_test_split
from sklearn.metrics import mean_absolute_error
rf_model = RandomForestRegressor(n_estimators = 100,random_state = 42)
rf_model.fit(trainX,trainY)
print('Mean absolute error for RF model: %0.4f'
%mean_absolute_error(testY,rf_model.predict(testX)))
feature_importance = pd.DataFrame(sorted(zip(rf_model.feature_importances_,
X.columns)), columns=['value','Feature'])
plt.figure(figsize=(10, 6))
sns.barplot(x="value", y="Feature", data=feature_importance.sort_values(by="value",
ascending=False))
```

RESULTS

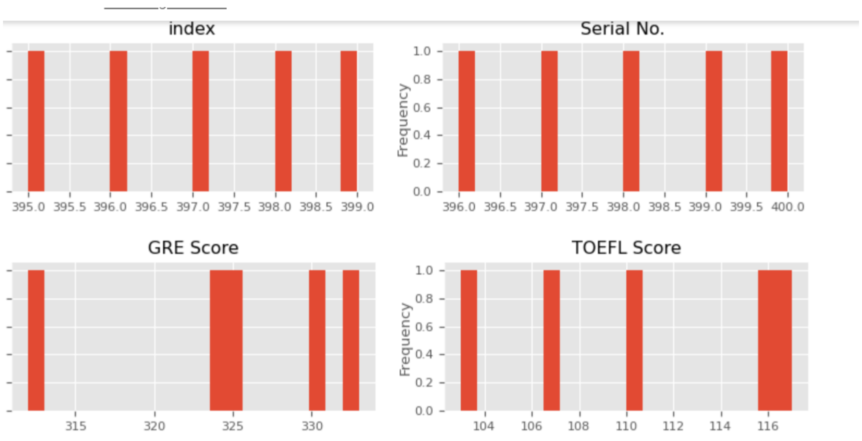
SCREENSHOTS:

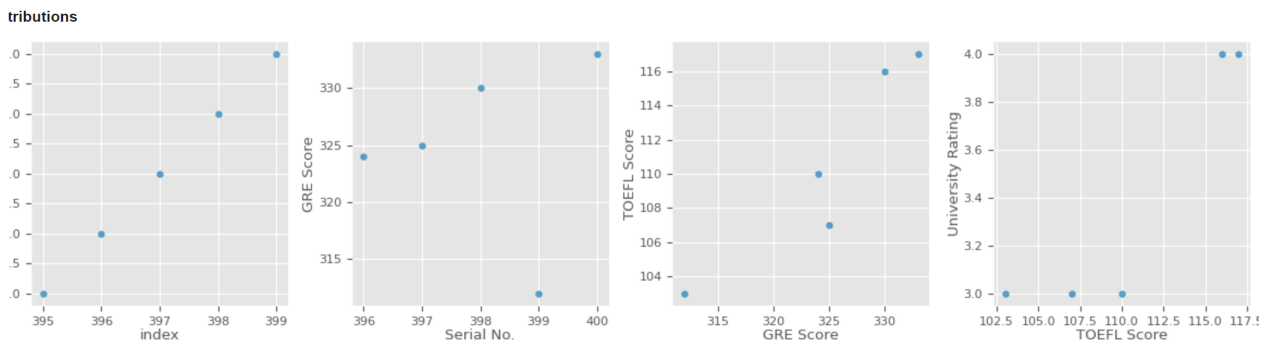


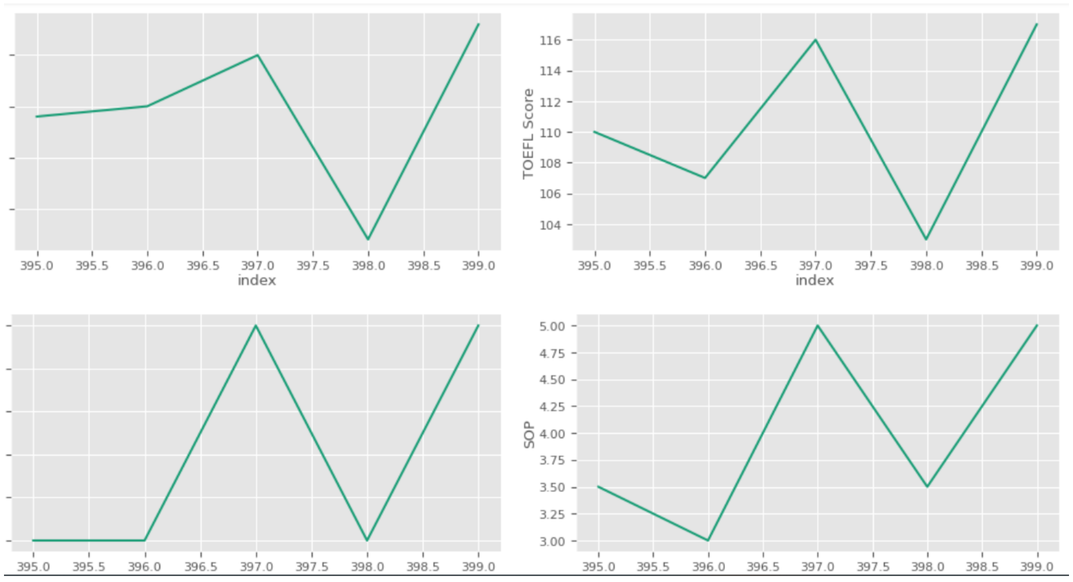
1 to 5 of 5 entries Filter

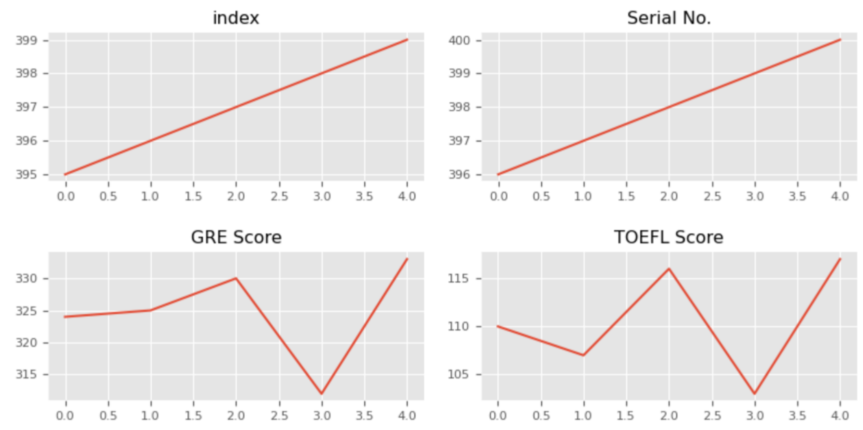
Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
396	324	110	3	3.5	3.5	9.04	1	0.82
397	325	107	3	3.0	3.5	9.11	1	0.84
398	330	116	4	5.0	4.5	9.45	1	0.91
399	312	103	3	3.5	4.0	8.78	0	0.67
400	333	117	4	5.0	4.0	9.66	1	0.95

25 per page



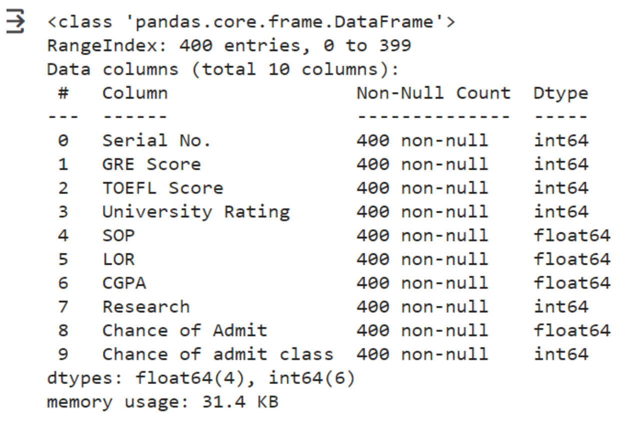






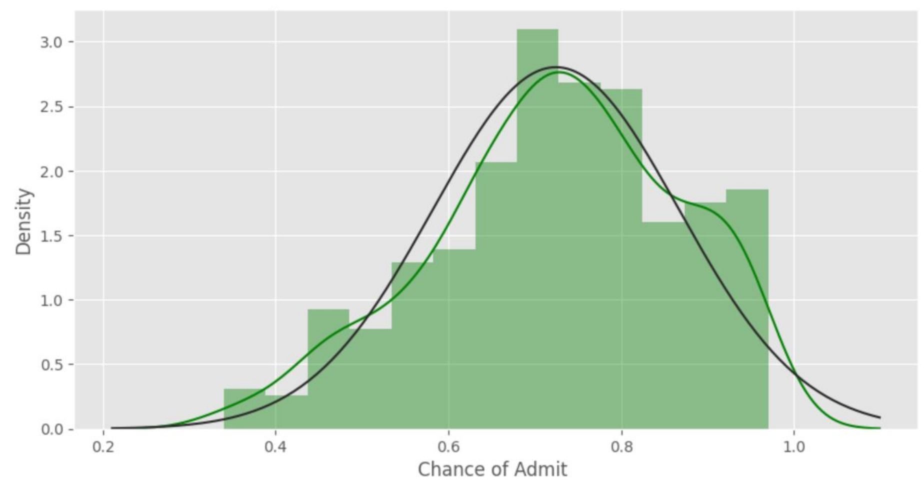
```
Shape of Data
Rows : 400 , Columns :  10
```

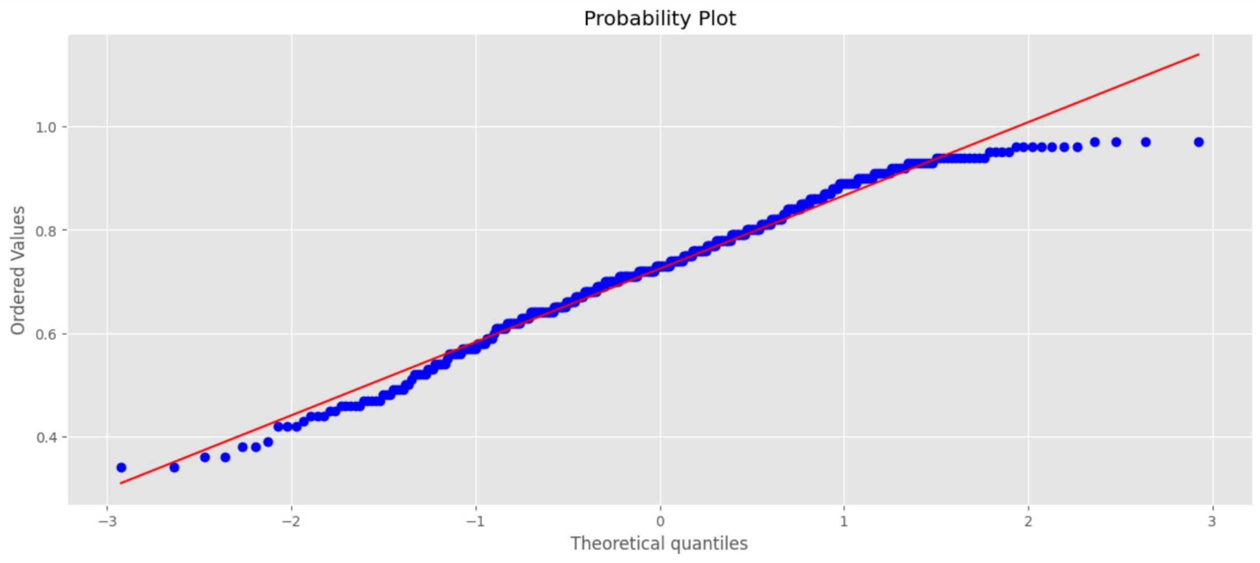
	Column_name	Missing_percent
Serial No.	Serial No.	0.0
GRE Score	GRE Score	0.0
TOEFL Score	TOEFL Score	0.0
University Rating	University Rating	0.0
SOP	SOP	0.0
LOR	LOR	0.0
CGPA	CGPA	0.0
Research	Research	0.0
Chance of Admit	Chance of Admit	0.0
Chance of admit class	Chance of admit class	0.0

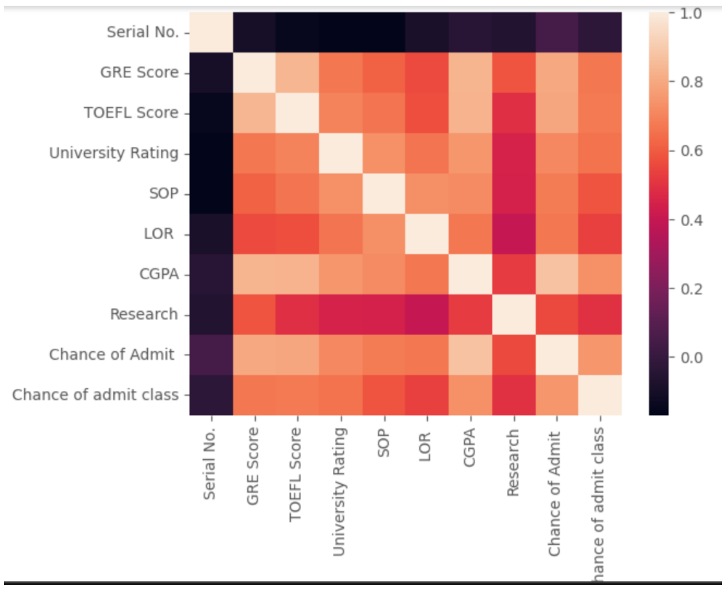


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Serial No.             400 non-null    int64  
 1   GRE Score              400 non-null    int64  
 2   TOEFL Score            400 non-null    int64  
 3   University Rating      400 non-null    int64  
 4   SOP                    400 non-null    float64
 5   LOR                    400 non-null    float64
 6   CGPA                   400 non-null    float64
 7   Research               400 non-null    int64  
 8   Chance of Admit        400 non-null    float64
 9   Chance of admit class  400 non-null    int64  
dtypes: float64(4), int64(6)
memory usage: 31.4 KB
```

```
Skewness of chance of admit :  -0.3534480999327828
```

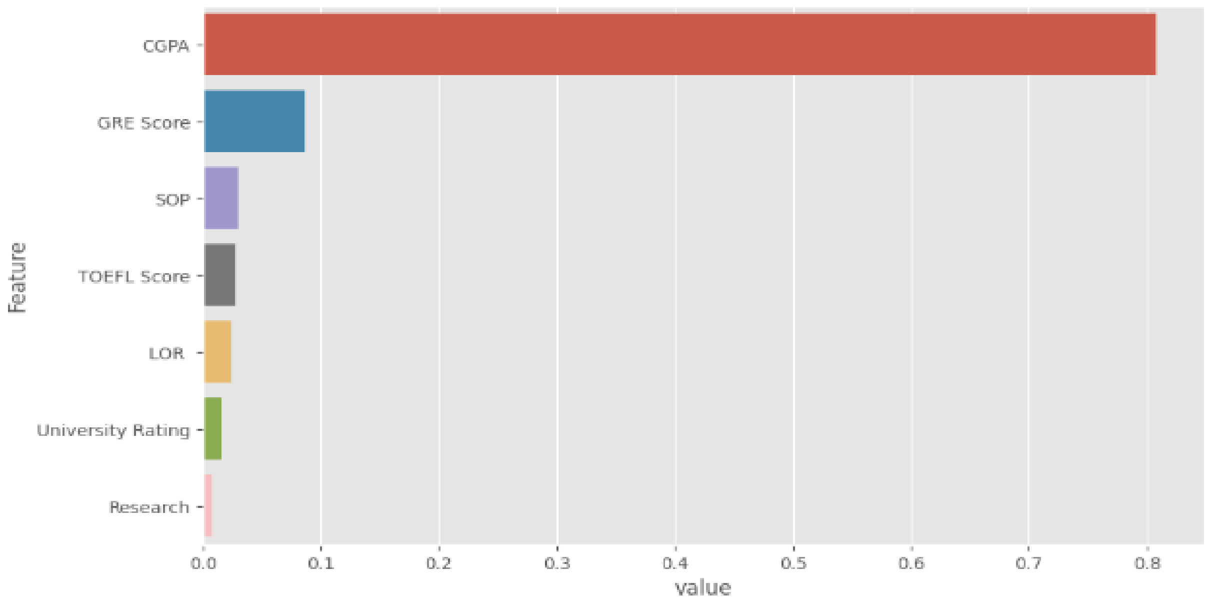






Serial No.
GRE Score
TOEFL Score
University Rating
SOP
LOR
CGPA
Research
Chance of Admit
Chance of admit class
1.0
0.8
0.6
0.4
0.2
0.0

```
<Axes: xlabel='value', ylabel='Feature'>
```



CGPA
GRE Score
SOP
TOEFL Score
LOR
University Rating
Research
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
value

Testing

8.1 SYSTEM TESTING

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

Testing is the process of checking the quality, functionality, and performance of a software product before launching. To do software testing, testers either interact with the software manually or execute test scripts to find bugs and errors, ensuring that the software works as expected. Software testing is also done to see if business logic is fulfilled, or if there are any missing gaps in requirements that need immediate tackles.

Testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.

8.2 TYPES OF TESTS

8.2.1 Unit testing

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

8.2.2 Integration testing

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

8.2.3 Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be
accepted. Invalid Input : identified classes of invalid input
must be rejected. Functions : identified functions must be
exercised.
Output : identified classes of application outputs must be
exercised. Systems/Procedures : interfacing systems
or procedures must be invoked.

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

8.2.4 System Test

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

8.2.5 White Box Testing

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

8.2.6 Black Box Testing

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

8.2.7 Testing

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

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

EXAMPLE OF TEST CASES:

TC ID	Condition Being Tested	Expected Result	Result
UITC_CkOt 1	path of the dataset file	Display path	pass
UITC_CkOt 2	if null values found replace with new values during preprocessing	Display sum of null values for each attribute and remove them or replace them	pass
UITC_CkOt 3	Display the extracted features from the dataset	Display features and their count	Passed
UITC_CkOt 4	Display the accuracy score	Accuracy score and evolution metrics in results	passed

Executing Steps:

1)Collect data: The first step is to collect data on students who have applied to and been admitted to the program you are interested in predicting. This data can include academic performance, extracurricular activities, demographics, and other factors that may be relevant to the admissions process.

2) Prepare the data: Once you have collected your data, you need to prepare it for modeling. This may involve cleaning the data, removing outliers, and converting the data into a format that is compatible with your chosen machine learning algorithm.

3) Choose a machine learning algorithm: There are many different machine learning algorithms that can be used for admission prediction. Some popular choices include logistic regression, decision trees, random forests, and gradient boosting machines.

4) Train the model: Once you have chosen a machine learning algorithm, you need to train the model on your prepared data. This involves feeding the data to the algorithm and allowing it to learn the relationships between the input features and the output target (i.e., whether or not the student was admitted).

5) Evaluate the model: Once the model is trained, you need to evaluate its performance on a held-out test set. This will give you an idea of how well the model will generalize to new data.

6) Deploy the model: Once you are satisfied with the performance of the model, you can deploy it to production. This may involve integrating the model into a web application or other software system.

CONCLUSION AND FUTURE WORK

The admission prediction problem statement is a complex one, involving a variety of factors such as academic performance, extracurricular activities, letters of recommendation, and personal statements. Machine learning models can be used to predict the likelihood of a student being admitted to a particular university or program, based on historical data.

In conclusion, machine learning models can be a valuable tool for both students and universities in the admissions process. Students can use these models to assess their chances of admission to different schools and programs, and to identify areas where they may need to improve their applications. Universities can use these models to streamline the admissions process and to make more informed decisions about admissions.

Students can make more informed decisions about their applications. By knowing their chances of admission to different schools and programs, students can focus their efforts on applying to schools where they are most likely to be accepted. This can save them time and money, and it can also help them to avoid the disappointment of being rejected from their dream schools.

Universities can make more informed decisions about admissions. Machine learning models can help universities to identify students who are a good fit for their institution, based on their academic performance, extracurricular activities, and other factors. This can help universities to build a more diverse and qualified student body.

The admissions process can be streamlined.Machine learning models can automate some of the tasks involved in the admissions process, such as reviewing applications and predicting admission probabilities. This can free up admissions staff to focus on more complex tasks, such as interviewing applicants and making final admissions decisions.

Overall, machine learning models have the potential to make the admissions process more efficient and effective for both students and universities.

REFERENCES

* Dekker, T., van den Heuvel, O. A., Bosma, H., Hermans, E. J., Hulshoff Pol, H. E., & Kahn, R. S. (2014). Brain imaging predictors of intelligence in a life-span perspective. Nature Biotechnology, 32(9), 885-889.
* EEG-Based Prediction of Student Academic Performance in Mathematics and Science, PLOS One, 9(12), e114328.
* Prediction of Graduate Admission Using Machine Learning, Nirbhay Rajgor, Academia.edu
* University Admissions Predictor, ResearchGate
* Graduate Admission Prediction using Ensemble Machine Learning Models, IRJET
* College Admission Prediction using Machine Learning, arXiv.org