A Project Report on

Grade Prediction using Machine Learning

Submitted towards partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Rupjyoti Barman
(D/20/CSE/204)
Jirjar Tokbi
(D/20/CSE/206)

Under the Supervision of

Dr. Moirangthem Marjit Singh

Associate Professor
(Department of Computer Science and Engineering)

Department of Computer Science and Engineering,
North Eastern Regional Institute of Science and Technology
Deemed to-be University Under the Ministry of Education, Govt. of India
Nirjuli-791109, Arunachal Pradesh, India

May 2023

DECLARATION

We hereby declare that the project work entitled "**Grade Prediction using Machine Learning"**, is a bonafide and original record of our work, a prerequisite towards partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering carried out under the guidance of **Dr. Moirangthem Marjit Singh, Associate professor,** Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology, Nirjuli, Arunachal Pradesh-701109. The work has not been submitted for the award of any other degree.

Rupjyoti Barman (D/20/CSE/204) **Jirjar Tokbi (D/20/CSE/206)**

CERTIFICATE OF APPROVAL

Certified that the project report entitled **"Grade Prediction using Machine Learning"** is a bonafide work carried out jointly by **RUPJYOTI BARMAN (D/20/CSE/204)** and **JIRJAR TOKBI (D/20/CSE/206).** The project report embodies the original work done by them towards partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** at **North Eastern Regional Institute of Science and Technology, Arunachal Pradesh.** It is understood by this approval that the undersigned do not endorse or approve any statement made, opinion expressed, or conclusion drawn therein, but approve the project report only for the purpose for which it has been submitted.

Dr. Moirangthem Marjit Singh
(Supervisor)
Associate Professor
Department of Computer Science & Engineering

Dr. Amar Taggu
(Project Coordinator)
Asst. Professor
Department of Computer Science & Engineering

Dr. Satya Jyoti Borah
(Head of the Department)
Department of Computer Science & Engineering

ACKNOWLEDGEMENT

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

We express our extreme gratitude and thanks to our project guide, **Dr. Moirangthem Marjit Singh, Associate Professor**, as well as project coordinator, **Dr. Amar Taggu, Assistant Professor**, **Department of Computer Science and Engineering, NERIST** for taking deep interest during project work and valuable guidance throughout our project work. Without his counsel, we would not have come up to this stage of completion.

We also owe our sincerest gratitude to **Dr. Satya Jyoti Borah, HoD, Department of Computer Science and Engineering, NERIST** for his kind support and cooperation. We are also thankful to our distinguished Faculty members and staff of Computer Science and Engineering Department who played a vital role in bringing us to the level of engineering standards.

Date:

Rupjyoti Barman (D/20/CSE/204) **Jirjar Tokbi (D/20/CSE/206)**

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In recent years, the growth of Massive Open Online Courses (MOOCs) and open education systems such as Udemy, Intern Shala, Coursera, and YouTube has revolutionized the way students learn. The availability of learning content at the click of a button has made it easier for students to access education at their own pace and convenience. However, this convenience may also lead to complacency and laziness among students, making it difficult for educators to predict their performance in advance. This is a crucial challenge in the education system, and several researchers have attempted to find solutions to this problem.

In this research, an attempt is made to help students predict their academic performance in advance using a univariate linear regression model. The model is based on collecting the marks of internal exam components of one subject, including Study hour and Mid-semester. These internal marks are then normalized to 100 (percentage) to ensure accurate results. The model provides a predicted grade of the final examination in a particular subject, thereby helping students plan their studies accordingly. Additionally, the model helps students understand how many marks are required in the internal examination to obtain a particular grade.

The study focuses on the examination pattern of North Eastern Regional Institute of Science and Technology (NERIST). The model developed in this research can be applied to any HEI to predict the performance of students and help them plan their studies accordingly. This research presents an innovative solution to a significant challenge in the education system and has the potential to improve student academic performance by encouraging them to work harder to achieve better grades. This study also highlights the growing importance of predictive analytics in the education system and its potential to solve several educational areas, including student performance, dropout prediction, academic early warning systems, and course selection.

Keywords: Grade, Marks, Linear regression, Logistic regression, NERIST, Massive Open Online Course, open education systems.

CONTENTS

Chapter 1: Introduction

Higher education institutions (HEIs) play a significant role in shaping the future of the new generation of the world by providing education and training to the next generation. To achieve this goal, HEIs collect a vast amount of data on student academic performance. This data is used to evaluate students' course achievements, identify areas of improvement, and provide feedback to students and instructors. However, measuring student academic performance is not an easy challenge in HEIs. Numerous factors can affect academic performance, including socioeconomic background, demographics, learning activities, and individual abilities. Despite this, predicting student grades has emerged as a promising solution to improve academic performance of students.

Predictive analytics has shown immense potential in the HEIs domain. It can be used to find hidden analytics has been applied to several educational areas, including the performance of students, the prediction of dropout students, early warning systems in academics, and select lookoff different courses. Due to its established benefits, the usage of predictive analytics in predicting student instructional success has grown through the years.

This paper focuses on the examination pattern of the North Eastern Regional Institute of Science and Technology (NERIST), an institute in India. As per internal examination pattern of NERIST, at least two quizzes on any subject should be conducted, followed by a sessional examination. The weighing internal examination is 20%, and the external examination is 80%. The paper examines how predictive analytics can be applied to examination data to predict student grades accurately. By doing so, the paper aims to contribute to the ongoing research on predicting student academic performance and improving the quality of education at NERIST.

Chapter 2: Background

By allowing computer systems to analyze and make smart decisions without explicit programming, device studying has revolutionized some sectors, from healthcare to finance. Artificial intelligence (AI) is a fast-developing discipline that enables robots to examine large volumes of records, spot patterns, and derive insightful conclusions. We shall observe the various sorts of machine studying and their sensible uses in this post.

2.1 Types of Machine Learning

- **Supervised Learning:** The most famous type of system mastering is supervised mastering, in which computers research from labelled training data to generate predictions or categorize fresh information. It consists of assigning associated output variables to enter variables. For instance, labelled samples of junk mail and non-junk mail emails can be used to educate a junk mail email classifier. For the motive of categorizing incoming emails as spam or not, the set of rules discovers patterns from the labelled records.
- **Unsupervised Learning:** To find hidden styles or structures, unsupervised mastering includes education algorithms on unlabeled facts. There aren't any distinct labels or outputs like there are in supervised learning. A common unsupervised gaining knowledge of approach used to institution together comparable statistics points is called clustering. By figuring out precise client agencies primarily based on their buy patterns, client segmentation in advertising, for instance, permits focused advertising and marketing techniques.
- **Reinforcement Learning:** Reinforcement studying is concerned with teaching algorithms to have interaction with their environment and advantage understanding via trial and blunders. The algorithm can improve its choice-making method by using receiving comments within the form of incentives or consequences primarily based on its behaviors. One such instance is teaching a self-sustaining automobile to go busy streets. Aiming to maximize benefits (like achieving the goal), and minimize drawbacks (like collisions), the algorithm learns through observing the results of its actions.

- **Semi-supervised Learning:** Algorithms in semi-supervised learning gain know-how from each labelled and unlabeled records. When labelling big datasets calls for several attempts or money, this technique is helpful. In sentiment analysis, for example, a system getting to know model can be educated making use of both a sizeable quantity of unlabeled records and a small series of labelled statistics. The model profits knowledge from labelled statistics and expands its predictions for the unlabeled statistics.
- **Deep Learning:** Artificial neural networks, which can be used in deep mastering, are modelled after the shape and operation of the human brain. These neural networks are made up of layers of related neurons that may research different degrees of abstraction for data illustration. Natural language processing, speech reputation, and image reputation are three regions wherein deep gaining knowledge of has excelled. Convolutional neural networks (CNNs) utilized for photograph category tasks like object identification in pix are an amazing instance.
- **Transfer Learning**: Transfer learning makes use of previously discovered models that have been carried out to new obligations or domains. Transfer gaining knowledge of reduces the amount of time and pc resources had to teach a model while retaining high accuracy. For instance, a version for categorizing pictures that has already been trained may be best tuned for a particular motive, like categorizing diverse kinds of flora. The version is already aware of about not unusual visible characteristics, which enables it to pick up on styles unexpectedly.

In end, there are numerous styles of machine getting to know, and every has a extraordinary method for looking at statistics and addressing problems. Many sectors, together with healthcare, banking, and transportation, have advanced thanks to supervised studying, unsupervised learning, reinforcement learning, semi-supervised getting to know, deep learning, and transfer gaining knowledge of. Machines getting to know makes use of will grow because it develops, revolutionizing sectors and spurring advancement in the discipline of synthetic intelligence. We can take advantage of new insights, beautify choice-making, and expand shrewd systems which could solve challenging real-global situations through using the electricity of system learning.

2.2 Steps of Machine Learning

The method of system studying, which entails multiple independent strategies, permits computer systems to analyze from statistics and offer insightful records. A machine getting to know algorithms may additionally efficaciously discover styles and generate predictions using a methodical method. To offer a thorough understanding of device studying's complexities, we will spoil down the process grade by grade on this publish.

- **Data Collection:** Gathering pertinent data is the initial degree in each gadget gaining knowledge of effort. This includes finding and getting datasets which might be suitable for the meant software. Several sources, including databases, APIs, and online repositories, are to be had for the statistics. To create reliable and accurate models, it's miles essential to make sure the records is specific, entire, and covers several circumstances.
- **Data preprocessing:** The mastering procedure is probably hampered by way of noise, lacking values, or mistakes which can be regularly present in raw information. Cleaning and altering the data to make it appropriate for analysis is referred to as data practices. In this stage, duplicates are removed, lacking values are treated, numerical values are standardized, and specific variables are encoded. To put the statistics right into a regular variety, it may also be vital to do characteristic scaling and statistics normalization.
- **Engineering and Feature Selection:** Engineering is the method of selecting the variables or functions that are maximum pertinent to the getting to know project. Feature selection aids inside the removal of pointless or redundant data through decreasing the dimensionality of the records, improving model performance, and lowering computing complexity. On the alternative hand, feature engineering entails developing new capabilities from those that exist already as a good way to growth the version's potential for prediction.
- **Model Selection:** Accurate forecasts need the usage of a suitable machine gaining knowledge of version, which must be decided on. The selection is made considering the trouble's nature, the records' characteristics, and the assets to hand. Each model has wonderful advantages and disadvantages, and the choice manner includes

assessing every version's overall performance the use of methods like move-validation.

- **Training the Model:** After the version has been chosen, it must be taught the use of the supplied dataset. The model learns from the enter records at some point of the training phase and modifies its internal parameters to reduce mistakes or maximize a longtime purpose characteristic. Iterative updates using optimization methods like gradient descent are part of the education technique. The model's performance is classified by the usage of the right criteria to gauge its precision and generalizability.
- **Model tuning and evaluation:** After training, the version's effectiveness must be assessed by the use of a specific validation dataset. The version's capability to generalize to new records is shown through this evaluation. To assess the version's performance metrics such as accuracy, precision and F1-rating are generated. Techniques for hyperparameter tuning and optimization can be used to beautify the version's overall performance if it fails to fulfill the required requirements.
- **Model Deployment:** The model is prepared for deployment in a real-international putting once it's been educated, assessed, and stepped forward. The model is covered into the meant utility or device, enabling it to make predictions primarily based on fresh, unforeseen statistics. The model has to be regularly monitored and maintained in order to hold top overall performance and accommodate transferring situations.

In end, device gaining knowledge of includes a fixed of specific degrees, from records gathering and preprocessing to model choice, education, and deployment. Building precise and sincere device studying models requires cautious consideration of every step. Following this methodical method will allow organizations and lecturers to take use of the capability of gadget gaining knowledge of to advantage insightful information, spur innovation, and make smart alternatives in lots of fields.

Chapter 3: Literature Review

There are numerous potentials uses for predicting a student's academic success in a higher education setting[1]. Using the grammar-guided genetic programming technique G3P-Ml, obstacles that would either positively or negatively affect learning from the standpoint of MIL have been discovered in paper one. These exercises also foretell a student's success or failure in a particular subject. To compare their idea to the most widely used Multiple Instance Learning (MIL) strategies, they use computation. The results show that G3P-Ml generates models that are more accurate and perform better.

Due to the truth that neither college students nor teachers are restricted to a particular region and are unbiased of any hardware platform, digital studying environments (V LEs) or e-learning structures have been designed and implemented. These technologies unquestionably can break down boundaries and provide adaptability, continuously up-to-date information, pupil memory attention, and individualized studying, making them crucial tools to complement each traditional study room instruction and online mastering. The use of these apps enables the collection of a lot of data because they may record every aspect of a student's behavior and interactions in a database or server.

Paper One explains how the hassle of predicting a pupil's final grade primarily based on his or her work in the VLE changed into treating the use of G3P-Ml. This kind of problem is solved using the most representative MIL paradigm, and the solutions are compared to gauge effectiveness. According to study findings, G3P-Ml outperforms the competition with an accuracy price of 74°/c and a sensitivity/specificity trade-off at values between zero.702 and 0.775. Additionally, it gathers essential information about the issue, which is extremely helpful in evaluating the effectiveness of extra resources given to students to deepen their grasp of the ideas of subjects addressed in class.

A Decision Tree is used to predict and analyze students' marks in the paper [2]. The writer used a record of students from a university in China to develop and evaluate a model. The Dataset contained Information about Academic Performance and other factors like Study habits and Socioeconomic.

The methodology steps involved, Data Processing, features selection, how the rule is generated in the model, and evaluation of the model. The outcome showed that the model achieved high accuracy in predicting and analyzing students' marks. The writer compared the performance of the decision Tree-based model with other Algorithms such as K-nearest neighbors and support vectors machine, and it is found that the Decision Tree-based model outperformed better in terms of accuracy.

It has been successful to predict student's academic achievement using educational data mining to uncover previously undiscovered connections in educational data[3]. With the grades from their midterm exams as the main source of data, this study offers a new model that employs ML techniques to forecast undergraduate students' test results. To forecast the outcomes of final exams, the predictive ability of several machine learning techniques was measured and compared. These methods included k-nearest neighbor (KNN), closest neighbor, logistic regression, support vector machines (SVM), Random Forests (RF), and closest neighbor. The dataset utilized in the study consisted of the academic achievement grades of 1854 students registered in the Turkish Language-I course at a state institution in Turkey during the autumn semester of 2019-2020.

The study's attention was drawn to two key variables. The original plan was to forecast academic success using past accomplishment ratings. To forecast final test marks, it also examined the performance indicators of several machine learning methods. According to the findings, the suggested model has a classification accuracy of between 70 and 75 percent. This depicts how a student's midterm test grade contributes significantly to their final exam mark. Due to their high rates of accuracy, the algorithms Random Forests (RF), closest neighbor (CN), support vector machines (SVM), Logistic Regression, Naive Bayes (Naiv), and k-nearest neighbor (KNN) were successful in predicting students' final exam results. Notably, the research only included three distinct types of parameters: faculty data, departmental data, and midterm test results.

The researchers of the University of Minnesota's computer science and engineering program developed a novel method for predicting future grades[4]. When they contrasted their procedure with industry standards, they discovered that it was more precise. Their

two techniques, matrix factorization, and linear regression produced more precise predictions. They also found that using a certain set of data made the predictions for each course more accurate. Overall, their research suggests that by utilizing their new strategy, these programs may be able to forecast future grades more accurately.

3.1 Gap in the Study

The G3P-MI algorithm, Decision Trees, Linear Regression, Matrix Factorization (MF), Support Vector Machines (SVM), Naive Bayes (Naiv), K-Nearest Neighbor (KNN), and Logistic Regression are just a few of the techniques we have seen in action.

- Paper [1], The absence of external validation of the G3P-MI algorithm on datasets from other educational institutions may be a weakness in the work. It would be helpful to determine whether the algorithm can generalize effectively to various educational contexts and datasets as the study only evaluated the performance of G3P-MI with other MIL approaches on a particular dataset.
- Paper [2], The absence of external validation of the G3P-MI method on datasets from other academic institutions may represent a research flaw. Only one dataset was used in the study to evaluate the effectiveness of G3P-MI with other MIL approaches, thus it would be helpful to find out how well the algorithm generalizes to different educational contexts and datasets.
- Paper [3], The study's possible weakness could be the constrained range of indicators utilized to forecast final test scores. The study only used three different kinds of parameters: faculty data, department data, and midterm test grades. Other significant factors, such as demographic data or attendance, could exist that might enhance the model's capacity for forecasting.
- Paper [4], The absence of an explanation for why Matrix Factorization and Linear Regression outperformed conventional approaches may be a weakness in the study. The study concentrated on comparing the performance of various algorithms without offering explanations for why certain algorithms performed better than others. Understanding the benefits and drawbacks of various algorithms may aid educators and researchers in selecting the best techniques for forecasting, future course grades.

Chapter 4: Proposed Work

It is seen from the above survey that the linear regression model is suitable for our work, as it predicts the marks and success accuracy is also better than others in comparison.

- Regression model for predicting the grade of students in a particular subject.
- Logistic Regression for predicting Pass/Fail.
- The model will be trained using the data set of existing students' marks in one subject.
- Average (Internal exam components of one subject, including Study hours and Mid-semester).
- To increase the system's accuracy, the marks are converted into percentages. In this system grade point range is converted in Alphanumeric forms, shown below in table 3.1.

Table 4.1: Percentage, Grade Point, and Grade.

Grade	Grade Point Range (%)	Point
S	> 85	10
A	> 75	09
B	> 65	08
C	> 55	07
D	> 45	06
E	> 35	05
F	< 35	00

As shown in the above table, suppose a student got 85 marks in a subject so he/she will get "S" grades, if students get 75 marks, the students get A grade. Similarly, this rule will apply to other grades and mark ranges as well.

4.1 Objectives

This main objective is to develop a linear regression-based model for forecasting a student's final grade in a certain topic. The model will be trained by using the marks of previous/current students, which will be normalized to percentages for better accuracy. The predicted grade will help students to know their performance in advance and guide them to improve their performance.

4.2 Methodology

- Collect data: The first step will be to collect the marks of existing students in the subject, including two quizzes and one sessional exam.
- Data Preprocessing: The data will be cleaned and preprocessed to remove any outliers and inconsistencies.
- Normalization: The marks will be converted into percentages to have accurate results.
- Develop the model: A linear regression-based model will be developed using the normalized marks of existing students. The model will be trained so that it can predict the grade of students.
- Testing and Validation: The developed model will be tested and validated using a separate dataset of student marks. The accuracy of the model will be measured using the R-squared(r^2) score.

4.3 Facilities Required

- **OS Used**
  - Windows 11
  - Windows 10
- **Languages Used**
  - Python
- **Platforms Used**
  - Jupyter notebook
  - Google Chrome

- **Tools Used**
  - Ms. Office 2016

4.4 Library Used

- **4.4.1 Pandas**

  Use the Python Pandas package to alter data sets. It offers tools for examining, categorizing, analyzing, and modifying data. The term "Pandas" is first used in 2008 by Wes McKinney.

  Why Use Pandas?

  - Pandas enable us to examine large amounts of data.
  - Data sets that are disorganized can be reorganized using pandas to make them understandable and helpful.
  - In data science, data is crucial.

  What Can Pandas Do?

  Pandas provide us with data-related responses. Like:

  - Checking if a relationship exists between two or more columns.
  - Average value
  - Max value
  - Min value

- **4.4.2 NumPy**

  Library NumPy is used, for working with arrays requires the usage of NumPy. "Numerical Python" is called NumPy.

  What is NumPy?

  - Python's NumPy library is necessary to work with arrays.–
  - Additionally, it offers matrices, the Fourier transform, the linear algebra domain functions.
  - NumPy was created in 2005 by Travis Oliphant and is source.
  - NumPy is the name of the Python extension for the project.

  Why Use NumPy?

  - Lists are Python's version of arrays, although processing lists takes a while.

- NumPy should make array objects up to 50 times faster than ordinary Python lists.
- The array object in NumPy is called an array, and it comes with several supporting functions that make using an array quite easy.

- **4.4.3 Matplotlib**

  A low-level graph charting toolkit in Python called Matplotlib is used for visualization. John D. Hunter developed Matplotlib. We are free to utilize Matplotlib because it is source. For platform probability, a small portion of Matplotlib is written in C, Objective-C, and JavaScript. Most of the libraries are developed in Python.

  Pyplot

  The majority of the Matplotlib tools are found in the pyplot submodule and are often imported using the pit alias:

  Import matplotlib.pyplot as plt

  You may now use the term plot to refer to the Pyplot package.

- **4.4.4 Seaborn**

  For plotting statistical visualizations, Python's Seaborn visualization package is amazing. let offers exceptional default patterns and shade palettes to make records charts more attractive. The Pandas fact's structure is intently tied to it and it is constructed on top of the Matplotlib toolbox. With Seaborn, information exploration and know-how could be concentrated on visualization. APIs that let us transition among numerous visual representations of equal variables for higher expertise of the dataset.

- **4.4.5 Sklearn**

  Scikit-learn (Sklearn) is the most dependable and practical Python package for machine learning. It gives several efficient tools for statistical modeling and device learning, inclusive of dimensionality discount, clustering, and class, via a Python consistency interface. The foundations of this library, which was broadly speaking

created in Python, are NumPy, SciPy, and Matplotlib.

Splitting the dataset

To check the version's precision, we can also separate the dataset right into an education set and a checking out. After the version has been educated with the training set, use the checking out set to check it. The overall performance of our model may then be evaluated. Using the

scikit-research train test split() function, the dataset is divided.

Syntax

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.34, random_state =1)

This function has the following arguments −

- X, y: Here, the feature matrix (X) and the response vector (Y) must be separated.
- test_size: This chart displays the ratio of test data to total accessible data. In the above example, test_data = 0.34 for 4000 rows of X. It will give test data size 4000*0.34 = 1360 rows.
- random_size: It is used to make sure that the gap doesn't change over time. This is helpful in circumstances when you need repeatable outcomes.

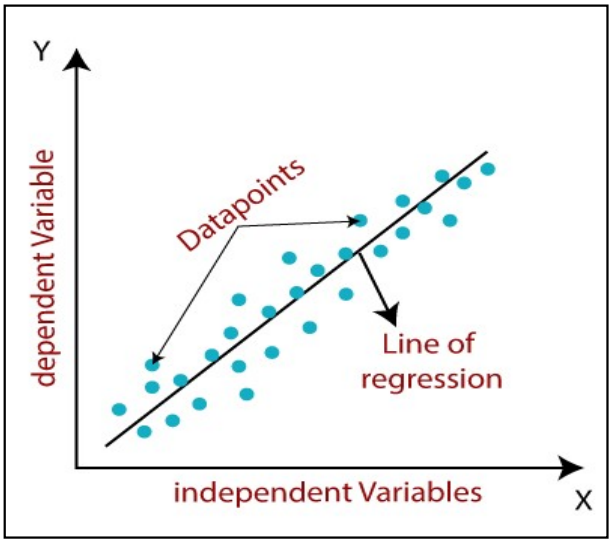
4.5 Regression Algorithms

- **4.5.1 Linear Regression**

  For establishing the relationship between two variables in statistics and machine learning, linear regression is a simple and popular method. It helps in estimating or making predictions about the value of a dependent variable based on the values of an independent variable.

  Consider a set of integer pairs. Using linear regression, the most precise straight line to represent the overall trend in the data is discovered. When we know the value of the independent variable, we may utilize this line to predict the value of the dependent variable.

For instance, linear regression can assist us in estimating a student's test score based on the number of hours they studied if we have data on the number of study hours (independent variable) and matching test scores (dependent variable). It assumes that two variables have a linear connection, such that when study time grows, test results ought to do the same.



[Image Source: https://www.javatpoint.com/linear-regression-in-machine-learning]

Fig 4.5.1.1: Linear Regression

Mathematically, we can represent a linear regression as:

y= a0+a1x

Here,

y= Dependent Variable

x= Independent Variable

a0= intercept of the line

a1 = Linear regression coefficient

Types of Linear Regression:

- Simple Linear Regression: The simplest kind of linear regression, known as simple linear regression, predicts one dependent variable from one independent variable. A straight line must be fitted to the data.
- Multiple Linear Regression: A statistical modeling approach called multilinear regression, sometimes referred to as multiple linear regression,

is used to examine the connection between several independent factors and a dependent variable. It broadens the definition of simple linear regression, which uses just one independent variable, to take into account a number of them.

- **4.5.2 Logistic Regression**
  - Logistic regression is a supervised learning method designed specifically to handle categorization problems.
  - This method is quite useful when working with categorical variables, where the outcomes may be classified as "Yes" or "No," "True" or "False," "Spam" or "Not Spam," etc.
  - The concept of probability serves as the foundation for logistic regression. It seeks to forecast the likelihood of an occurrence by simulating the connection between the independent factors and the categorical dependent variable.
  - Despite being a kind of regression, logistic regression is employed differently than linear regression. While linear regression forecasts continuous numerical values, logistic regression focuses on predicting the likelihood of a categorical result.
  - In logistic regression, the data are modelled using a sigmoid function or logistic function. This cost function, which is intricate, modifies the relationship between the variables and allows for exact prediction. The mathematical formula of the sigmoid function is :

$$f(x) = 1 / (1 + e^{(-x)})$$

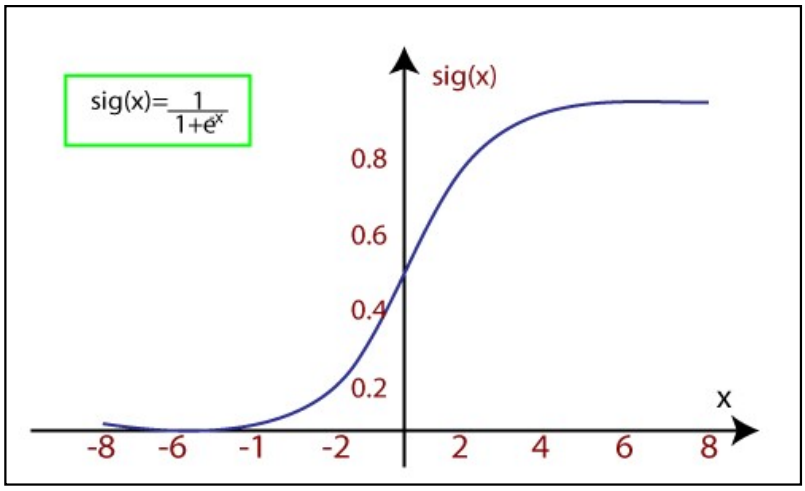
x- Function's input,

e- Natural logarithm's base

f(x)- A number between 0 and 1.

The input is converted into a range between 0 and 1 using the sigmoid function, where 1 denote the more chance that the event will occur. The

function generate the following S-curve when we give the input values to it.



[Image Source: https://www.javatpoint.com/regression-analysis-in-machine-learning]

Fig 4.5.2.1: Logistic Regression

Chapter 5: Results and Discussions

- Screen Shot of the python script to generate Data Set with random values.

```
import pandas as pd
import numpy as np

# set the number of rows
num_rows = 45

# create an empty data frame
df = pd.DataFrame()

# creating a list of study hour--------------------
sh=np.random.randint(low=0, high= 8,size=num_rows)
df['Study_hour']=sh


# creating a list for Assignment marks -----------
assign=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(1,3)
        assign.append(temp)

    elif sh[i]==1:
        temp=np.random.randint(1,3)
        assign.append(temp)

    elif sh[i]==2:
        temp=np.random.randint(2,4)
        assign.append(temp)
```

```
    elif sh[i]==3:
        temp=np.random.randint(3,5)
        assign.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(3,6)
        assign.append(temp)

    elif sh[i]==5:
        temp=np.random.randint(4,6)
        assign.append(temp)

    elif sh[i]==6:
        assign.append(5)

    elif sh[i]==7:
        assign.append(5)

df['Assignment']=assign


# creating a list for Attendance marks ---
atten=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(1,3)
        atten.append(temp)

    elif sh[i]==1:
        temp=np.random.randint(1,4)
        atten.append(temp)
```

```
    elif sh[i]==2:
        temp=np.random.randint(2,4)
        atten.append(temp)

    elif sh[i]==3:
        temp=np.random.randint(2,5)
        atten.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(4,6)
        atten.append(temp)

    elif sh[i]==5:
        temp=np.random.randint(4,6)
        atten.append(temp)

    elif sh[i]==6:
        atten.append(5)

    elif sh[i]==7:
        atten.append(5)

df['Attendance']=atten


# creating a list for Quiz1 marks --------
quiz1=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(1,3)
        quiz1.append(temp)
```

```
    elif sh[i]==1:
        temp=np.random.randint(1,3)
        quiz1.append(temp)

    elif sh[i]==2:
        temp=np.random.randint(1,4)
        quiz1.append(temp)

    elif sh[i]==3:
        temp=np.random.randint(2,5)
        quiz1.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(3,5)
        quiz1.append(temp)

    elif sh[i]==5:
        temp=np.random.randint(4,6)
        quiz1.append(temp)

    elif sh[i]==6:
        quiz1.append(5)

    elif sh[i]==7:
        quiz1.append(5)

df['Quiz_one']=quiz1
```

```
# creating a list for Quiz2 marks -----
quiz2=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(1,3)
        quiz2.append(temp)

    elif sh[i]==1:
        temp=np.random.randint(1,3)
        quiz2.append(temp)

    elif sh[i]==2:
        temp=np.random.randint(1,4)
        quiz2.append(temp)

    elif sh[i]==3:
        temp=np.random.randint(2,5)
        quiz2.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(3,5)
        quiz2.append(temp)

    elif sh[i]==5:
        temp=np.random.randint(4,6)
        quiz2.append(temp)

    elif sh[i]==6:
        quiz2.append(5)

    elif sh[i]==7:
        quiz2.append(5)
```

```
df['Quiz_two']=quiz2


# creating a list for Midsem marks ----
mid=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(1,6)
        mid.append(temp)

    elif sh[i]==1:
        temp=np.random.randint(5,11)
        mid.append(temp)

    elif sh[i]==2:
        temp=np.random.randint(10,16)
        mid.append(temp)

    elif sh[i]==3:
        temp=np.random.randint(15,20)
        mid.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(20,25)
        mid.append(temp)

    elif sh[i]==5:
        temp=np.random.randint(23,28)
        mid.append(temp)

    elif sh[i]==6:
        temp=np.random.randint(25,30)
```

```
        mid.append(temp)

    elif sh[i]==7:
        temp=np.random.randint(27,31)
        mid.append(temp)
df['Midsem']=mid


# creating a list for Endsem marks ------
end=list()
for i in range(len(sh)):
    if sh[i]==0:
        temp=np.random.randint(2,10)
        end.append(temp)

    elif sh[i]==1:
        temp=np.random.randint(6,16)
        end.append(temp)

    elif sh[i]==2:
        temp=np.random.randint(8,19)
        end.append(temp)

    elif sh[i]==3:
        temp=np.random.randint(15,26)
        end.append(temp)

    elif sh[i]==4:
        temp=np.random.randint(24,36)
        end.append(temp)

    elif sh[i]==5:
```

```
        temp=np.random.randint(30,42)
        end.append(temp)

    elif sh[i]==6:
        temp=np.random.randint(38,47)
        end.append(temp)

    elif sh[i]==7:
        temp=np.random.randint(42,51)
        end.append(temp)
df['Endsem']=end


# creating a column to know pass or fail
pf=list()
for i in range(len(sh)):
    total=0
    for j in range(1,7):
        total+=df.iloc[i][j]
    if total>=35:
        pf.append("pass")
    else:
        pf.append("fail")
df['Pass_Fail']=pf
```

- Screen shot of the generated Data Set, which have 4000 rows and 8 columns such as Study_hour, Assignment, Attendance, Quiz_one, Quiz_two, Midsem, Endsem, Pass_Fail.

Study_ho	Assignme	Attendanc	Quiz_one	Quiz_two	Midsem	Endsem	Pass_Fail
0	1	1	1	1	5	8	fail
1	2	3	1	2	9	12	fail
4	3	4	4	3	23	25	pass
5	5	5	5	4	23	36	pass
0	1	2	1	2	1	8	fail
4	5	4	4	4	23	30	pass
0	1	2	2	1	4	5	fail
6	5	5	5	5	26	42	pass
2	2	3	2	2	14	15	pass
4	5	4	3	3	22	32	pass
6	5	5	5	5	27	43	pass
1	2	3	1	2	10	9	fail
1	1	2	2	2	10	12	fail
1	1	3	2	2	8	9	fail
6	5	5	5	5	29	39	pass
5	4	5	5	5	25	40	pass
7	5	5	5	5	29	44	pass
3	4	4	3	3	19	18	pass
1	2	2	2	2	10	7	fail
0	1	1	2	2	5	7	fail
3	4	4	4	4	15	20	pass
7	5	5	5	5	28	47	pass

3979	7	5	5	5	5	30	48	pass
3980	6	5	5	5	5	25	42	pass
3981	7	5	5	5	5	29	42	pass
3982	6	5	5	5	5	26	39	pass
3983	0	2	1	1	2	5	7	fail
3984	4	3	4	4	3	23	32	pass
3985	6	5	5	5	5	27	40	pass
3986	6	5	5	5	5	25	44	pass
3987	0	2	1	2	2	2	4	fail
3988	1	2	1	2	2	10	13	fail
3989	7	5	5	5	5	27	43	pass
3990	6	5	5	5	5	28	41	pass
3991	0	2	1	1	1	4	3	fail
3992	6	5	5	5	5	28	41	pass
3993	3	3	3	3	4	19	21	pass
3994	6	5	5	5	5	29	45	pass
3995	0	1	2	2	2	3	7	fail
3996	1	2	3	2	2	10	13	fail
3997	6	5	5	5	5	25	43	pass
3998	7	5	5	5	5	27	48	pass
3999	1	2	1	1	2	10	15	fail
4000	6	5	5	5	5	29	46	pass
4001	3	4	4	2	2	19	15	pass

- Screen shot of the python code to save the generated Data Set in Disk as a (.csv) file.

```
# converting the Data Frame to CSV file

df.to_csv("lastDS_4000.csv",index=False)
```

- **Attributes Information**
  - Study hour – Daily study hour.
  - Assignment - Marks obtained out of 5
  - Attendance - Marks obtained out of 5
  - Quiz one – Marks obtained out of 5
  - Quiz two – Marks obtained out of 5
  - Mid Sem – Marks obtained out of 30
  - End Sem – Marks obtained out of 50
  - Pass Fail – Showing pass or fail result

- Importing all the required libraries for the project.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

- Importin Data Set into project from the disk.

```
# importing csv file
df=pd.read_csv("C:\\Users\\rupjyoti\\Desktop\\MachineLearProject\\B.Tech Final Year\\End Sem\\updated\lastDS_4000.csv")
```

- Showing the correlation of each column with each other.

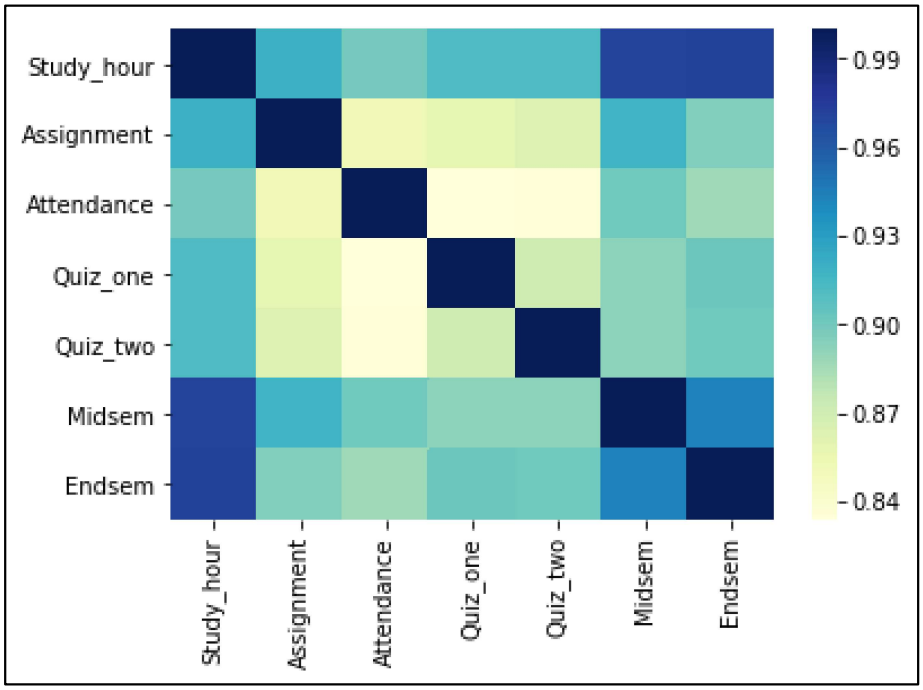


Fig 5.1 : A Correlation matrix for representing the correlation between different variables.

5.1 Linear Regression Model

- Set independent and dependent variables into variables X and y.

```
# Let's separate Independent variable X and Dependent variable y

X= df[['Study_hour', 'Assignment', 'Attendance', 'Quiz_one', 'Quiz_two', 'Midsem']]
y = df['Endsem']
```

- Splitting data for Training and Testing into variables such as X_train, X_test, y_train, y_test.

```
# split data for training and testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.34)
```

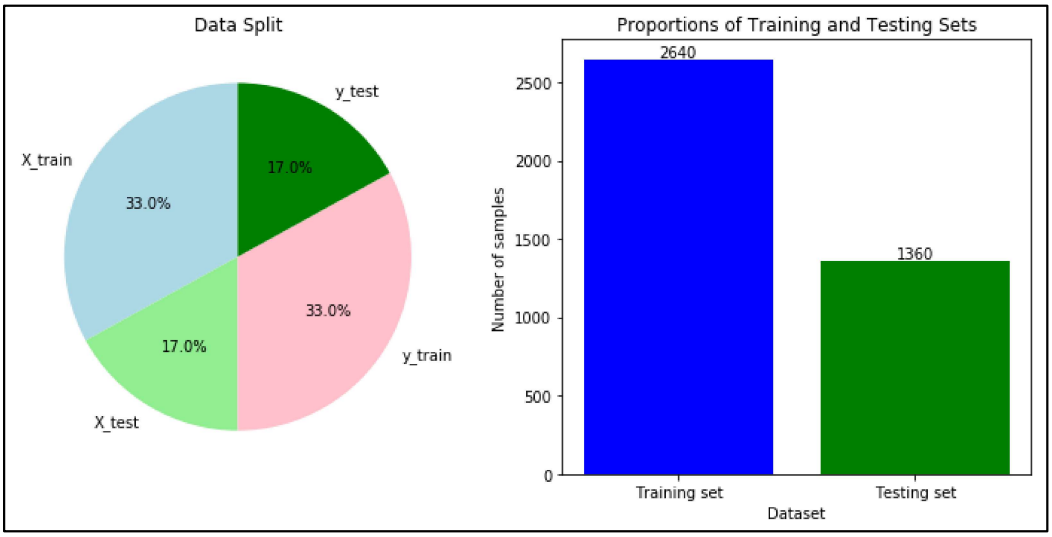


Fig 5.1.1 : A Pie chart and Bar graph for Training and Testing Dataset.

- Training the Linear Regression model by giving variables such as X_train and y_train to the fit() function.

```
# Training the model

regression = LinearRegression()
regression.fit(X_train,y_train)
```

- Predicting marks by giving the X_test data to the model and getting prediction marks into y_pred variable.

```
# prediction by giving the X_test data to the model and
# geting prediction data in y_pred variable

y_pred=regression.predict(X_test)
```

- Putting, Actual End Sem marks and Predicted marks in a new Data Frame to show the difference.

```
df2 = pd.DataFrame({'Actual_Endsem': y_test, 'Predicted_Endsem': y_pred})
df2.head(20)
```

	Actual_Endsem	Predicted_Endsem
1585	34	34.634527
441	47	46.350629
2254	30	34.730009
730	25	22.053280
1444	16	15.794351
3279	45	46.350629
2160	8	16.419427
1136	30	34.932136
2161	9	3.849639
254	50	46.546058
2850	13	10.358159
3670	41	41.249988
2614	37	34.050770
2862	45	41.152274

- Graph showing comparison between actual total and predicted total.

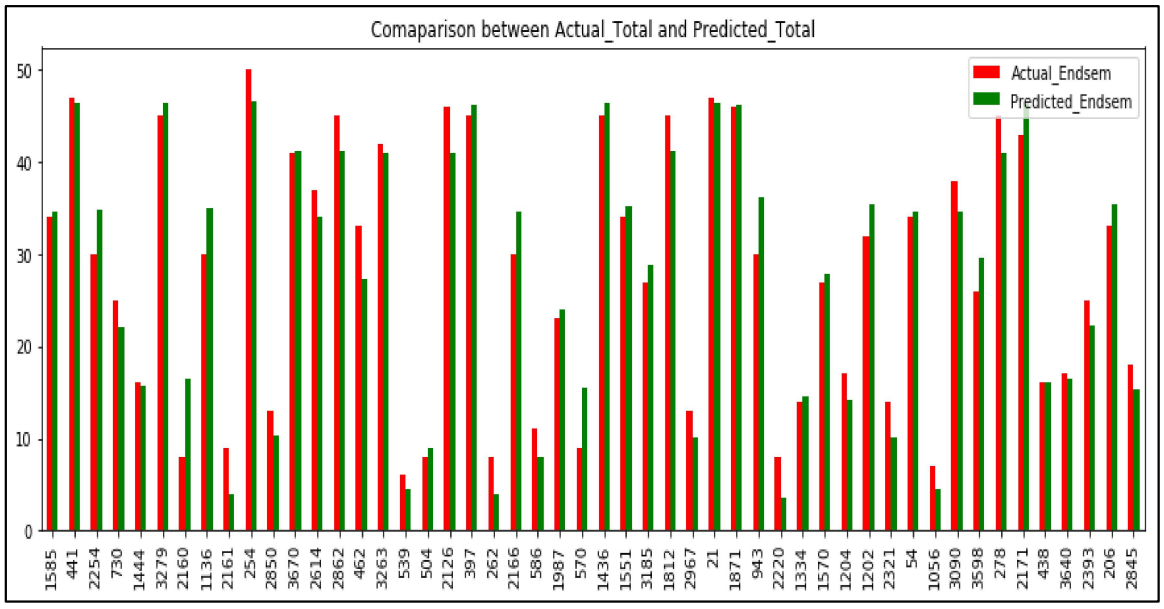


Fig 5.1.2 : Comaparison between Actual_Total and Predicted_Total Endsem marks

- Let's calculate Total Actual and Predicted Marks.

```
# calculate total of actual marks (X_test+y_test)

temp_sum=0
total=0
total_actual_marks=[]

len_test_data=len(y_test)

for i in range(0,len_test_data):
        for j in range(1,6):
                temp_sum+=X_test.iat[i,j]

        total=temp_sum +y_test.iat[i]
        total_actual_marks.append(total)
        temp_sum=0
        total=0
```

```
# calculate total of predicted marks (X_test+y_pred)

temp_sum=0
total=0
total_predicted_marks=[]

len_pred_data=len(y_pred)

for i in range(0,len_pred_data):
        for j in range(1,6):
                temp_sum+=X_test.iat[i,j]

        total=temp_sum +y_pred[i]
        total_predicted_marks.append(total)
        temp_sum=0
        total=0
```

- Putting, total Actual marks and Total Predicted Marks in a new data frame to show the difference between them.

```
# Comparing total Actual marks and Total Predicted Marks

df3 = pd.DataFrame({'Actual_Total':total_actual_marks, 'Predicted_Total':total_predicted_marks})
df3.head(20)
```

	Actual_Total	Predicted_Total
0	77	77.634527
1	96	95.350629
2	72	76.730009
3	55	52.053280
4	39	38.794351
5	94	95.350629
6	30	38.419427
7	75	79.932136
8	16	10.849639
9	97	93.546058
10	26	23.358159
11	87	87.249988
12	79	76.050770
13	92	88.152274
14	70	64.256103

- Comparing Total Actual marks and Total Predicted Marks

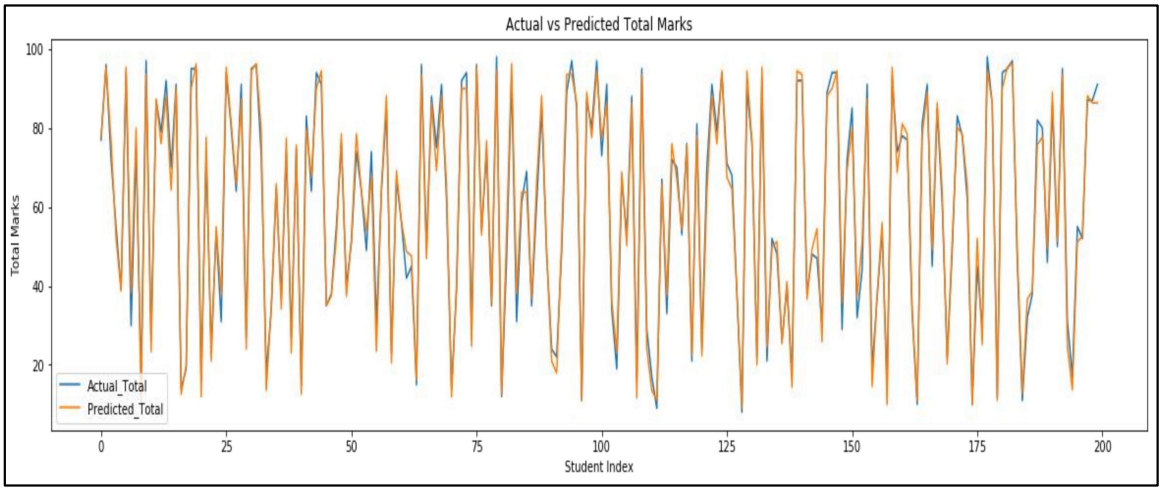


Fig 5.1.3 : Comparison of Total Actual marks and Total Predicted Marks

- Calculating Actual Grade and Predicted Grade

```
# Calculating Actual Grade

len_actual_marks=len(total_actual_marks)
actual_grade=[]

for i in range(0,len_actual_marks):
    if total_actual_marks[i]>85:
        actual_grade.append("S")
    elif total_actual_marks[i]>75:
        actual_grade.append("A")
    elif total_actual_marks[i]>65:
        actual_grade.append("B")
    elif total_actual_marks[i]>55:
        actual_grade.append("C")
    elif total_actual_marks[i]>45:
        actual_grade.append("D")
    elif total_actual_marks[i]>34:
        actual_grade.append("E")
    else:
        actual_grade.append("F")
```

```
# Calculating Predicted Grade

len_predicted_marks=len(total_predicted_marks)
predicted_grade=[]

for i in range(0,len_predicted_marks):
    if total_predicted_marks[i]>85:
        predicted_grade.append("S")
    elif total_predicted_marks[i]>75:
        predicted_grade.append("A")
    elif total_predicted_marks[i]>65:
        predicted_grade.append("B")
    elif total_predicted_marks[i]>55:
        predicted_grade.append("C")
    elif total_predicted_marks[i]>45:
        predicted_grade.append("D")
    elif total_predicted_marks[i]>34:
        predicted_grade.append("E")
    else:
        predicted_grade.append("F")
```

- Putting, Actual and Predicted Grade in a new data frame and showing the difference between them.

```
df4 = pd.DataFrame({'Actual_Grade':actual_grade, 'Predicted_Grade':predicted_grade})
df4.head(20)
```

	Actual_Grade	Predicted_Grade
0	A	A
1	S	S
2	B	A
3	D	D
4	E	E
5	S	S
6	F	E
7	B	A
8	F	F
9	S	S
10	F	F
11	S	S
12	A	A
13	S	S
14	B	C
15	S	S

- Comparing Total Actual marks, Predicted Marks, Actual Grade and Predicted Grade Together and showing the result or difference between them.

```
df5 = pd.DataFrame({'Actual_Total_Marks':total_actual_marks, 'Predicted_Total_Marks':total_predicted_marks,
                    'Actual_Grade':actual_grade, 'Predicted_Grade':predicted_grade})
df5.head(20)
```

	Actual_Total_Marks	Predicted_Total_Marks	Actual_Grade	Predicted_Grade
0	77	77.634527	A	A
1	96	95.350629	S	S
2	72	76.730009	B	A
3	55	52.053280	D	D
4	39	38.794351	E	E
5	94	95.350629	S	S
6	30	38.419427	F	E
7	75	79.932136	B	A
8	16	10.849639	F	F
9	97	93.546058	S	S
10	26	23.358159	F	F
11	87	87.249988	S	S
12	79	76.050770	A	A
13	92	88.152274	S	S
14	70	64.256103	B	C
15	91	89.956845	S	S

- Checking Accuracy of the Model by using r2_score () method.

```
# Accuracy check
# usign R-squared (R²) score

from sklearn.metrics import r2_score

acc = 100*(r2_score(y_test, y_pred))
print("Accuracy of the model is : %.2f" %acc)
```

```
Accuracy of the model is : 94.70
```

4.2 Logistic Regression Model

- Assigning Independent variable X and Dependent variable y.

```
# Let's separate Independent variable X and Dependent variable y

X= df[['Study_hour', 'Assignment', 'Attendance', 'Quiz_one', 'Quiz_two', 'Midsem', 'Endsem']]
y = df['Pass_Fail']
```

- Splitting data for Training and Testing into variables such as X_train, X_test, y_train, y_test.

```
# split data for training and testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40)
```

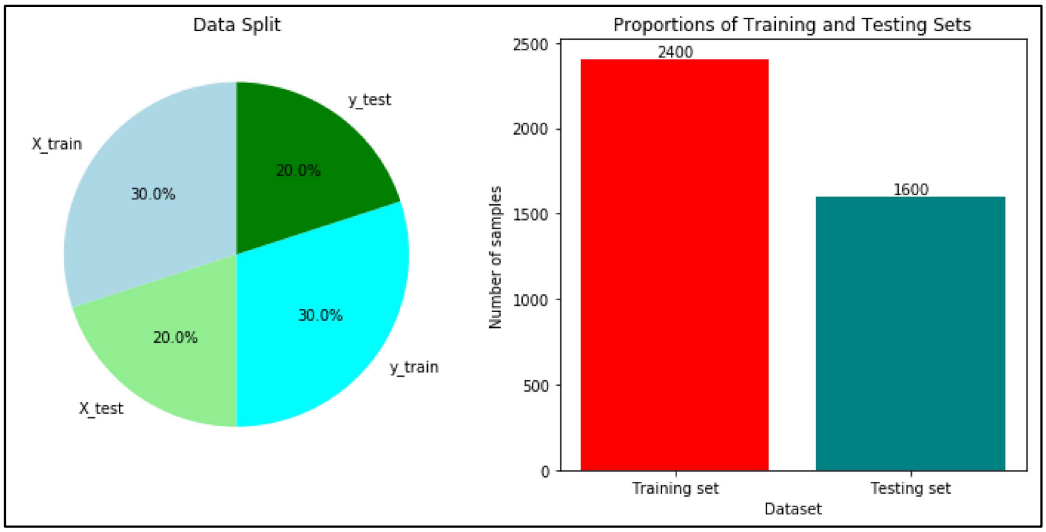


Fig 5.2.1 : A Pie chart and Bar graph for Training and Testing Dataset.

- Training the model

```
# Training the model

model = LogisticRegression()
model.fit(X_train, y_train)
```

- Prediction by giving the X_test data to the model and getting prediction data in y_pred variable.

```
# prediction by giving the X_test data to the model
# and geting prediction data in y_pred variable

y_pred=model.predict(X_test)
```

- Comparing Actual Pass/Fail result with the Prediction result.

```
df6 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df6.head(20)
```

	Actual	Predicted
919	pass	pass
1295	pass	pass
2854	pass	pass
2964	pass	pass
1778	pass	pass
396	pass	pass
61	pass	pass
526	pass	pass
1377	pass	pass
3002	pass	pass
896	pass	pass
1283	pass	pass
3323	fail	fail
2661	pass	pass
2282	fail	fail
165	pass	pass
2738	pass	pass

- Checking Accuracy of the Logistic regression Model

```
# checking Accuracy
acc=100*(accuracy_score(y_test, y_pred))
print("Accuracy of the model is : %.2f" %acc)
```

```
Accuracy of the model is : 98.81
```

Chapter 6: Conclusion and Future Scope

In conclusion, higher education institutions (HEIs) are educating and developing the future. By obtaining academic data on students, HEIs may assess performance, identify areas for improvement, and provide feedback. However, predicting student grades is a challenging process because of a variety of influencing factors. Predictive analytics, which utilizes data to identify patterns and project outcomes, offers one potential solution. Course selection, early warning systems, dropout prediction, and student performance are some of their applications. By concentrating on the NERIST test pattern, this study investigates how predictive analytics may accurately predict student grades. By contributing to the field of academic performance prediction, this effort aims to enhance instruction at NERIST and advance the study of student analytics in higher education.

The following features can be added to this application: -

- Enhanced Accuracy
- Early warning system for students.
- Students' Guidance and Support: These models can suggest courses, recommend resources, and offer carrier guidance based on individual performance.
- Personalized Learning: The grade prediction model can be used to develop personalized learning plans for students based on their learning styles, strengths, and weaknesses. This may assist pupils in getting better scores and performing better in class overall.

References

[1]. A. Zafra, C. Romero, and S. Ventura. Predicting academic achievement using multiple instance genetic programming. In 2009 Ninth International Conference on Intelligent Systems Design and Applications, page 11201125, Nov 2009

[2]. Z. Liu and X. Zhang. Prediction and analysis for students' marks based on decision tree algorithm. In 2010 Third International Conference on Intelligent Networks and Intelligent Systems, page 338341, Nov 2010.

[3]. Mustafa Yagci, "Educational data mining: prediction of students' academic performance using machine learning algorithms" published at Open Access on September, 2022

[4]. A. Polyzou and G. Karypis, "Grade prediction with models specific to students and courses", *Int. J. Data Sci. Anal.*, vol. 2, no. 3, pp. 159-171, Dec. 2016

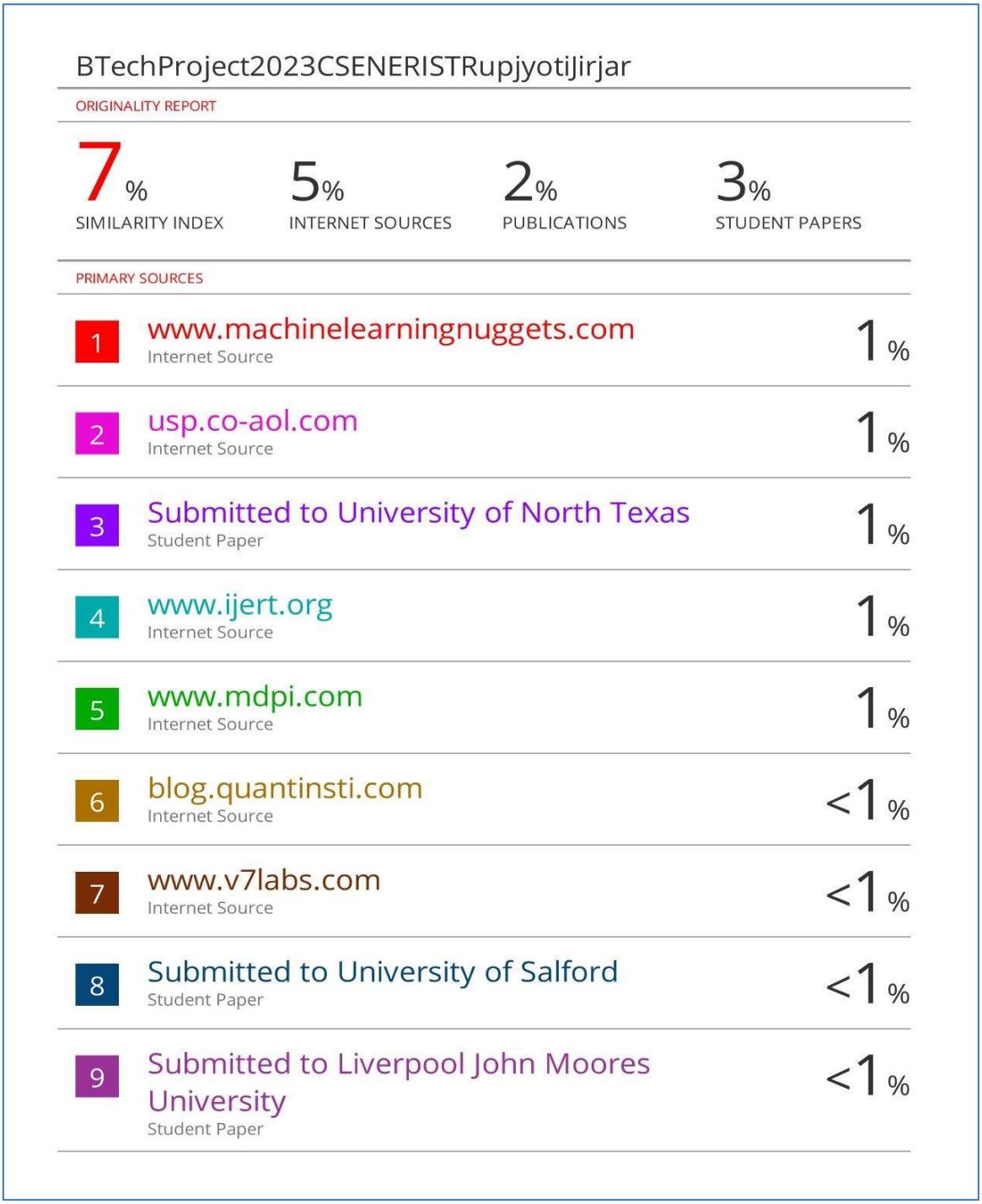
APPENDIX

Originality Report

BTechProject2023CSENERISTRupjyotiJirjar

by Dr Moirangthem Marjit Singh

Submission date: 21-May-2023 04:01PM (UTC+0530)
Submission ID: 2098229565
File name: Updated_B.tech_Project_Report_for_plagiarism.docx (686.29K)
Word count: 4746
Character count: 26594



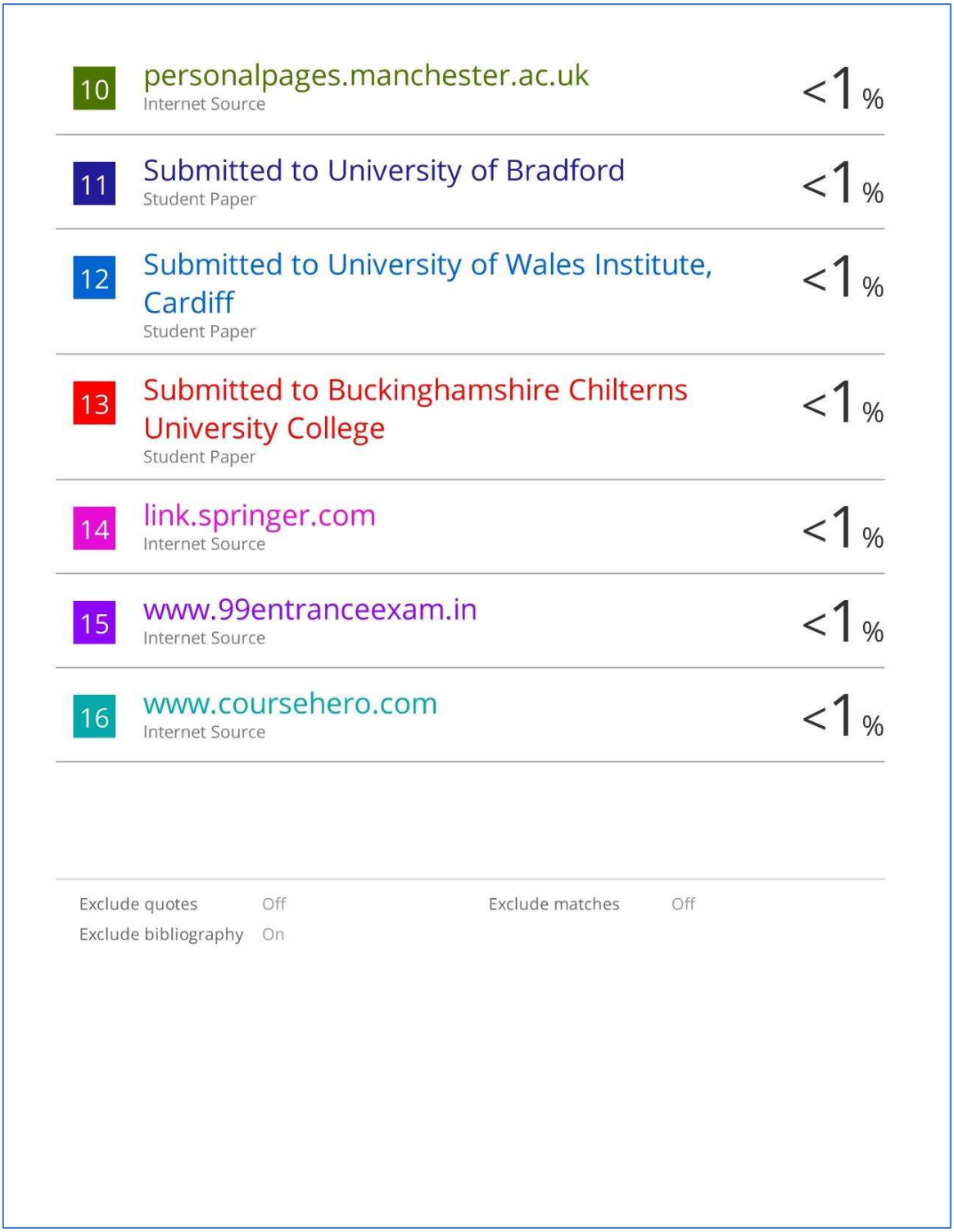
BTechProject2023CSENERISTRupjyotiJirjar

ORIGINALITY REPORT

7%	5%	2%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.machinelearningnuggets.com Internet Source	1%
2	usp.co-aol.com Internet Source	1%
3	Submitted to University of North Texas Student Paper	1%
4	www.ijert.org Internet Source	1%
5	www.mdpi.com Internet Source	1%
6	blog.quantinsti.com Internet Source	<1%
7	www.v7labs.com Internet Source	<1%
8	Submitted to University of Salford Student Paper	<1%
9	Submitted to Liverpool John Moores University Student Paper	<1%



10	personalpages.manchester.ac.uk Internet Source	<1 %
11	Submitted to University of Bradford Student Paper	<1 %
12	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
13	Submitted to Buckinghamshire Chilterns University College Student Paper	<1 %
14	link.springer.com Internet Source	<1 %
15	www.99entranceexam.in Internet Source	<1 %
16	www.coursehero.com Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		