

Forecasting Students' Academic Performance Using Support Vector Machine

**A Project Report Submitted in partial fulfillment of the
requirements for the award of the degree of**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

J. SAI MUKESH	19551A0529
G. SRINIVASARAO	19551A0581
M. SAI YOUZEN	19551A0541
M. KARTHIK	19551A05B2
K. NITHIN	19551A05A0

**Under The Supervision of
Mrs. N. GOWRI SREELAKSHMI
Assistant Professor
Department of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GODAVRI INSTITUTE OF ENGINEERING & TECHNOLOGY
CHAI TANYA KNOWLEDGE CITY, NH-16,
RAJAMAHENDRAVARAM, AP
Jawaharlal Nehru Technological University, Kakinada, AP, India
MARCH-2023**

GODAVARI INSTITUTE OF ENGINEERING & TECHNOLOGY

(Autonomous)

CHAITANYA KNOWLEDGE CITY, NH-16, RAJAMAHENDRAVARAM, AP.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



BONAFIDE CERTIFICATE

Certified that this project report “**FORECASTING STUDENTS’ ACADEMIC PERFORMANCE USING SUPPORT VECTOR MACHINE**” is the Bonafide work of “**J. Sai Mukesh (19551A0529), G. Srinivasarao (19555A0581), M. Sai Youzen (19551A0541), M. Karthik (19555A05B2), K. Nithin (19551A05A0)**”, who carried out the project work under my supervision during the year 2022 to 2023, towards partial fulfillment of the requirements of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING of GODAVARI INSTITUTE OF ENGINEERING&TECHNOLOGY(AUTONOMOUS),RAJAHMAHENDRAVARAM, AP.** The results embodied in this report have not been submitted to another university for the award of any degree.

Signature of the Head of the Department

Dr. B. SUJATHA

Professor & Head of The Department

Department of Computer Science & Engineering
Godavari Institute of Engineering
& Technology(A)

Signature of the Supervisor

N. GOWRI SREELAKSHMI

Assistant Professor

Department of Computer Science & Engineering
Godavari Institute of Engineering
& Technology(A)

External Viva voice conducted on _____

Internal Examiner

External Examiner

GODAVARI INSTITUTE OF ENGINEERING & TECHNOLOGY

(Autonomous)

CHAITANYA KNOWLEDGE CITY, NH-16, RAJAMAHENDRAVARAM, AP.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION OF AUTHENTICATION

We solemnly declare that this project report “***FORECASTING STUDENTS’ ACADEMIC PERFORMANCE USING SUPPORT VECTOR MACHINE***” is the bonafide work done purely by us, carried out under the supervision of **Mrs.N.GOWRI SREELAKSHMI** towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Computer Science & Engineering as administered under the Regulations of Godavari Institute of Engineering & Technology, Rajamahendravaram, AP, India and award of the Degree from Jawaharlal Nehru Technological University, Kakinada during the year 2022- 2023.

We also declare that no part of this document has been taken up verbatim from any source without permission from the author(s)/publisher(s). Wherever few sentences, findings, images, diagrams or any other piece of information has been used for the sake of completion of this work, we have adequately referred to the document source. In the event of any issue arising hereafter about this work, we shall be personally responsible.

It is further certified that this work has not been submitted, either in part or full, to any other department of the Jawaharlal Nehru Technological University Kakinada, or any other University, institution or elsewhere, in India or abroad or for publication in any form.

NAME OF THE STUDENTS

SIGNATURE OF THE STUDENTS

J.SAI MUKESH (19551A0529)

G.SRINIVASARAO (19551A0581)

M. SAI YOUZEN (19551A0541)

M. KARTHIK (19551A05B2)

K. NITHIN (19551A05A0)

ACKNOWLEDGEMENT

We feel immense pleasure to express our sincere thanks and profound sense of gratitude to all those people who played a valuable role for the successful completion of our project.

We would like to express my gratitude to **Dr. K.V.V. SATYANARAYANA RAJU**, chairman, Chaitanya group of Institutions, **Sri. K. SASI KIRAN VARMA**, Vice Chairman, GIET group of Institutions for providing necessary infrastructure.

We are thankful to **Dr. P.M.M.S. SARMA** , Principal for permitting and encouraging us in doing this project.

Our special thanks to **Dr. N. LEELAVATHY**, Vice Principal (Academics), for their content guidance and motivation and also for providing an excellent environment. We are truly grateful for their valuable suggestions and advice.

We are proudly grateful to express my deep sense of gratitude and respect towards **Dr. B. SUJATHA**, professor and Head of the Department, Computer Science and Engineering, whose motivation and constant encouragement has led to pursue a project in the field of software development. We are very fortunate, for having her to enlighten us in all the aspects of life and transforming us to be an ideal individual in this field.

We are much obliged and thankful to our internal guide **Mrs.N.GOWRI SREELAKSHMI** ,Assistant professor, Department of CSE, for providing this opportunity and constant encouragement given by her during the course. We are grateful to her valuable guidance and suggestions during my project work.

Our parents have put us ahead of themselves, because of their hard work and dedication, we have had opportunities beyond our wildest dreams. Our heart full thanks to them for giving us all support we needed to be successful student and individual

Finally, we express our thanks to other professors, classmates, friends, neighbours and non-teaching staff who helped us for the completion of my project and without infinite love and patience this would never have been possible.

J. SAI MUKESH

G. SRINIVASARAO

M. SAI YOUZEN

M. KARTHIK

K. NITHIN

ABSTRACT

Student success forecasting is a goal of higher education institutions and a significant area of study. Both students and educational institutions are genuinely concerned with predicting students' academic achievement. Academic advancement and personality traits related to learning activities are only a few variables that might affect pupils' academic performance. Common traits of students like age, gender, how much they study, how often they miss classes, the size of their families, and so on... are the training dataset's primary constituents for the supervised machine learning algorithm to determine whether the student will be successful on their final exam or not. This study investigated the effectiveness of various supervised machine learning methods, like K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Logistic Regression. Previous studies on this dataset relied heavily on the K-Nearest Neighbor and Logistic Regression algorithms, with mixed results. Surprisingly, the Support Vector Machine algorithm is rarely used for predictions, even though it's a popular and reliable technique. To make sure we had a fair comparison, we decided to use Support Vector Machines to predict the students' grades and see how they compared to the other methods.

LIST OF FIGURES

FIG NO		PAGE NO
4.1	SYSTEM ARCHITECTURE	6
4.2.1	USE CASE	8
4.2.2	OBJECT DIAGRAM	9
4.2.3	SEQUENCE	10

TABLE OF CONTENTS

S.NO	CONTENTS	PAGE No.
	ABSTRACT	
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	SYSTEM ANALYSIS	3
	3.1 EXISTING SYSTEM	3
	3.2 PROPOSED SYSTEM	3
	3.3SYSTEM REQUIREMENTS	4
	3.4 SYSTEM STUDY	5
4	SYSTEM DESIGN	6
	4.1 SYSTEM ARCHITECTURE	6
	4.2 UML	7
	4.2.1 USE CASE	8
	4.2.2 OBJECT DIAGRAM	9
	4.2.2 SEQUENCE	10
5	TECHNOLOGIES LEARNT	11
	5.1 ALGORITHMS USED	18
	5.2 CODE	21
6	TESTING	55
	6.1 INTRODUCTION	55
	6.2 TEST RESULTS	61
7	CONCLUSION	62
8	BIBLIOGRAPHY	63

1. INTRODUCTION

Estimating student performance is critical for educators because it allows them to gather feedback early on and take quick action or put in protections if needed to help the student do better. This forecast is manageable if the cause of the problem is identified. If it's from doing after-school activities, having family problems, or health issues, all that stuff can really mess up a student's grades. We can investigate such scenarios with the use of a dataset for student performance. We already said that by looking at the records from previous students, we can create a forecasting model that will help students do better in their tests. We'll try out different classifiers such as KNN or SVM and compare them how they stack up with each other. A variety of elements, such as familial issues or alcohol use, can impact a student's exam performance. By employing our machine learning model we can modify the lectures and curriculum, and lecturers and institutional managers which can support students' learning plans by drawing on the prediction of learning outcomes. The results indicated that the SVM classifier was the most reliable and effective among the three classifiers for predicting students' final grades when the supervised machine learning approaches we have discussed were examined.

Forecasting Students' Academic Performance Using Support Vector Machine (SVM) is a machine learning approach that aims to predict the academic performance of students. It involves the use of historical academic data of students, such as grades, attendance, and other relevant factors, to create a predictive model. The model is then used to forecast the academic performance of students in future semesters or academic years.

SVM is a supervised learning algorithm that can be used for both classification and regression tasks. It is a powerful algorithm that has been used in various fields, including finance, biology, and engineering. SVM is particularly useful in cases where the data is complex, nonlinear, and high-dimensional.

In the context of forecasting students' academic performance, SVM can help educators and institutions identify students who may be at risk of poor performance and provide them with targeted support to improve their academic outcomes. The machine learning algorithms for comparison and predicting whether a student would pass the final exam/ not are:

- Logistic Regression
- K- Nearest Neighbor
- Support Vector Machine (SVM)

2. LITERATURE SURVEY

Earlier models made use of ML algorithms including Random Forest, Naive Bayes Decision Tree, Linear Regression, and Bagging [1][2][3][4]. The previous system uses the machine learning algorithm, which is a decision tree and random forest, to evaluate student performance, and the decision tree is used to take a group of data values from the dataset [1]. The machine learning algorithm is linear regression, which was used to produce the evaluation of student performance but provided less accuracy because of underfitting and overfitting issues [2].

The results of the Empirical study showed that the Support Vector Machine had a slight edge over the K-Nearest Neighbor and Logistic Regression algorithms. [3]. Regression algorithms are slightly better than Neural networks [4]. Boosting student retention rates, streamlining enrolment processes, keeping alumni connected, better-targeted marketing, and all-around improvement of the educational institute's efficacy all benefited from the students' performance projection [5].

“Support Vector Machines for Predicting Academic Performance “ in College by S. Kumar and S. Varma (2008): In this paper, the authors propose an SVM-based model for predicting the academic performance of college students using demographic and academic data. The results show that SVM outperforms traditional regression-based methods.

“Forecasting Student Academic Performance Using Ensemble of Decision Trees and Support Vector Regression" by M. Ayvaz, et al. (2015): This study presents an ensemble model that combines decision trees and support vector regression (SVR) to predict the academic performance of students. The authors report that the proposed model achieved high accuracy in predicting students' grades.

"Predicting Student Academic Performance in a Blended Learning Environment Using Support Vector Regression" by A. Akbulut and T. Kılıçarslan (2019): This study investigates the use of SVM-based regression for predicting the academic performance of students in a blended learning environment. The results suggest that SVM outperforms other regression methods in predicting students' grades.

"An Effective Model for Predicting Student Academic Performance Using Support Vector Machines and Feature Selection" by S. Sen, et al. (2019): This study proposes an SVM

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

K-Nearest Neighbors (KNN) and Logistic Regression are both widely used machine learning algorithms for predicting academic performance. KNN can be used to make predictions based on the past academic performance of similar students, while logistic regression is used to predict whether a student will pass or fail a course based on their past academic performance. However, KNN can be sensitive to the choice of K and computationally expensive, while logistic regression assumes a linear relationship and may suffer from overfitting. Despite these potential drawbacks, both methods can be effective for forecasting student academic performance when applied carefully and with appropriate considerations for the dataset and problem being addressed.

DISADVANTAGES OF EXISTING SYSTEM:

K-Nearest Neighbors (KNN):

- KNN can be sensitive to the choice of K, which determines the number of neighbors used to make the prediction. If K is too small, the model may be too sensitive to noise in the data, while if K is too large, the model may not capture important patterns in the data.
- KNN can also be computationally expensive when dealing with large datasets, since it requires calculating the distance between each test example and every training example.

Logistic Regression:

- Logistic regression assumes that the relationship between the input variables and the output variable is linear, which may not always be the case in practice.
- Logistic regression can also suffer from overfitting if the model is too complex or the number of training examples is too small.
- Logistic regression is limited to binary or multi-class classification tasks, and may not be suitable for predicting continuous outcomes such as GPA.

3.2 PROPOSED SYSTEM

The proposed system includes the algorithms such as K-Nearest Neighbor and Logistic Regression, Support Vector Machine (SVM) that are used to classify the features that helps students for better performance in academics along with the prediction of final exam mark and compares the above 3 algorithms based on quality metrics such as Confusion Matrix, F1 score, Receiver Operating Characteristic(ROC) Curve.

ADVANTAGES OF PROPOSED SYSTEM:

- Came Up With the best classifier that is more accurate and avoid overfitting and underfitting by using simple techniques.
- Know what the most factors affect a student performance

3.3 SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

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

3.4 SYSTEM STUDY

FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:-

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4 SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE:

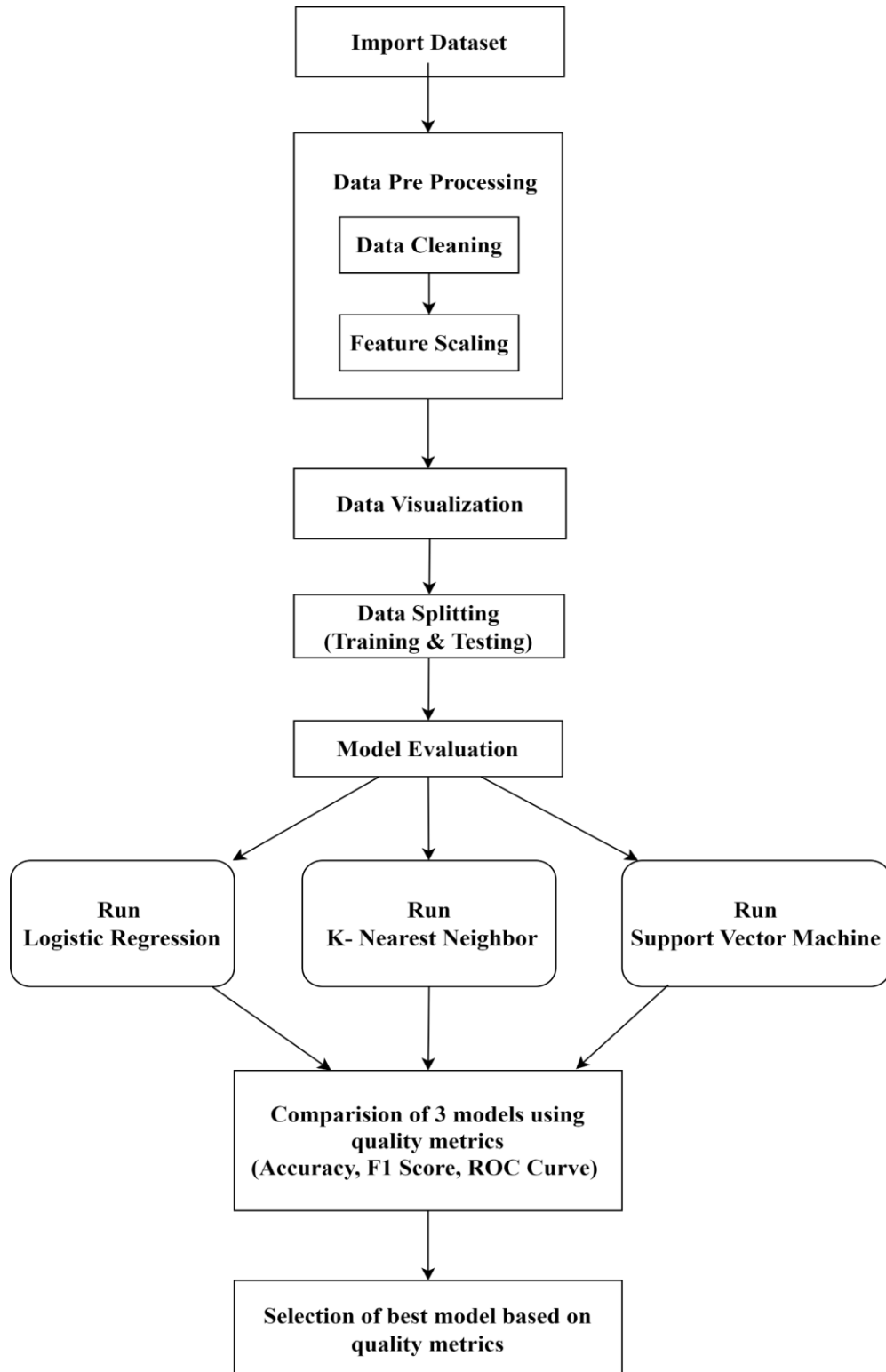


Fig 4.1 System Architecture

4.2 UML

INTRODUCTION TO UML

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

GOALS OF UML

The primary goals in the design of the UML were:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

WHY WE USE UML?

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

UML DIAGRAM

The underlying premise of UML is that no one diagram can capture the different elements of a system in its entirety. Hence, UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system.

The nine UML diagrams are:

4.2.1 USE CASE DIAGRAM:

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

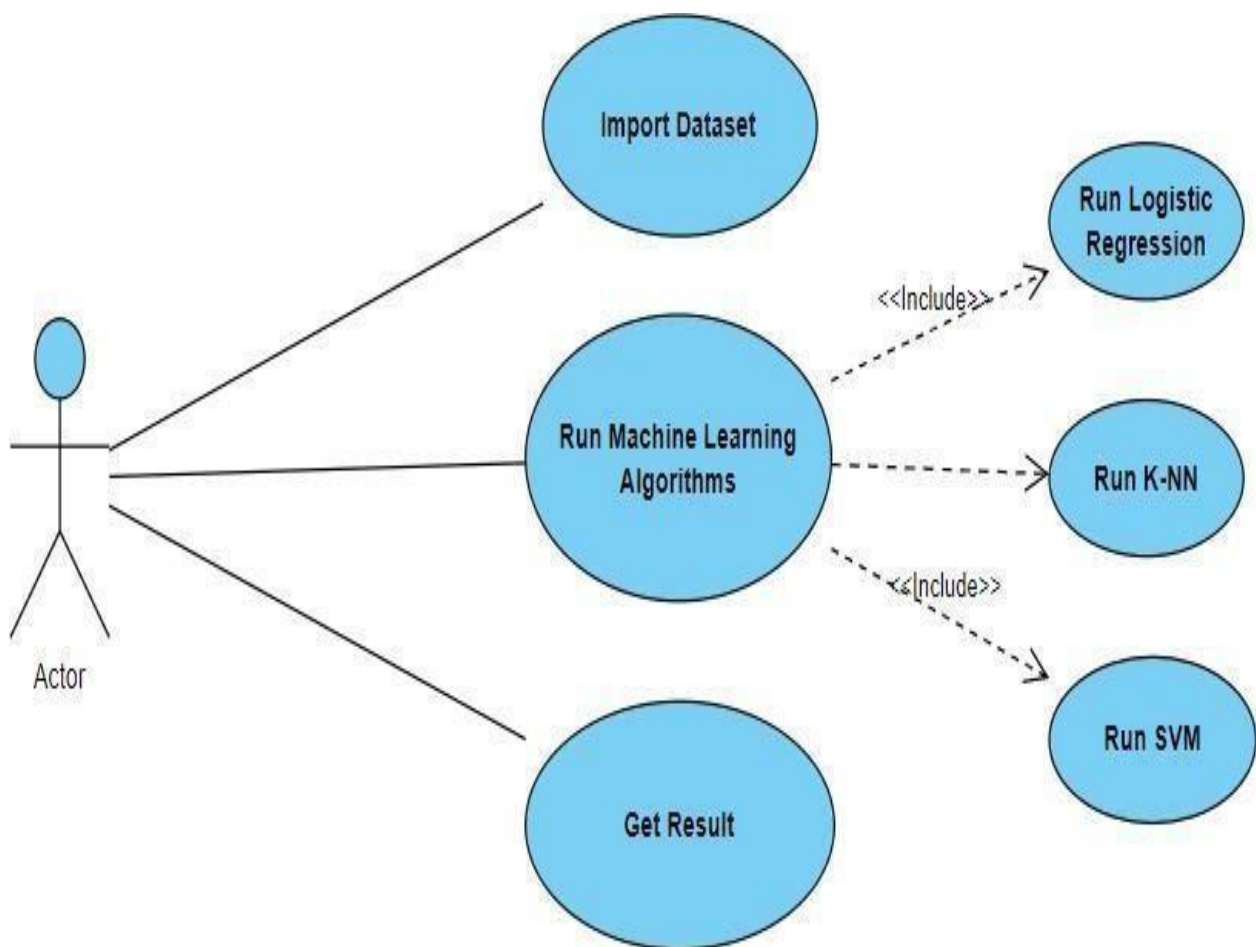


Fig 4.2.1 Use case

CLASS DIAGRAM:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

4.2.2 OBJECT DIAGRAM:

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.



Fig 4.2.2 Object Diagram

SEQUENCE DIAGRAM:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

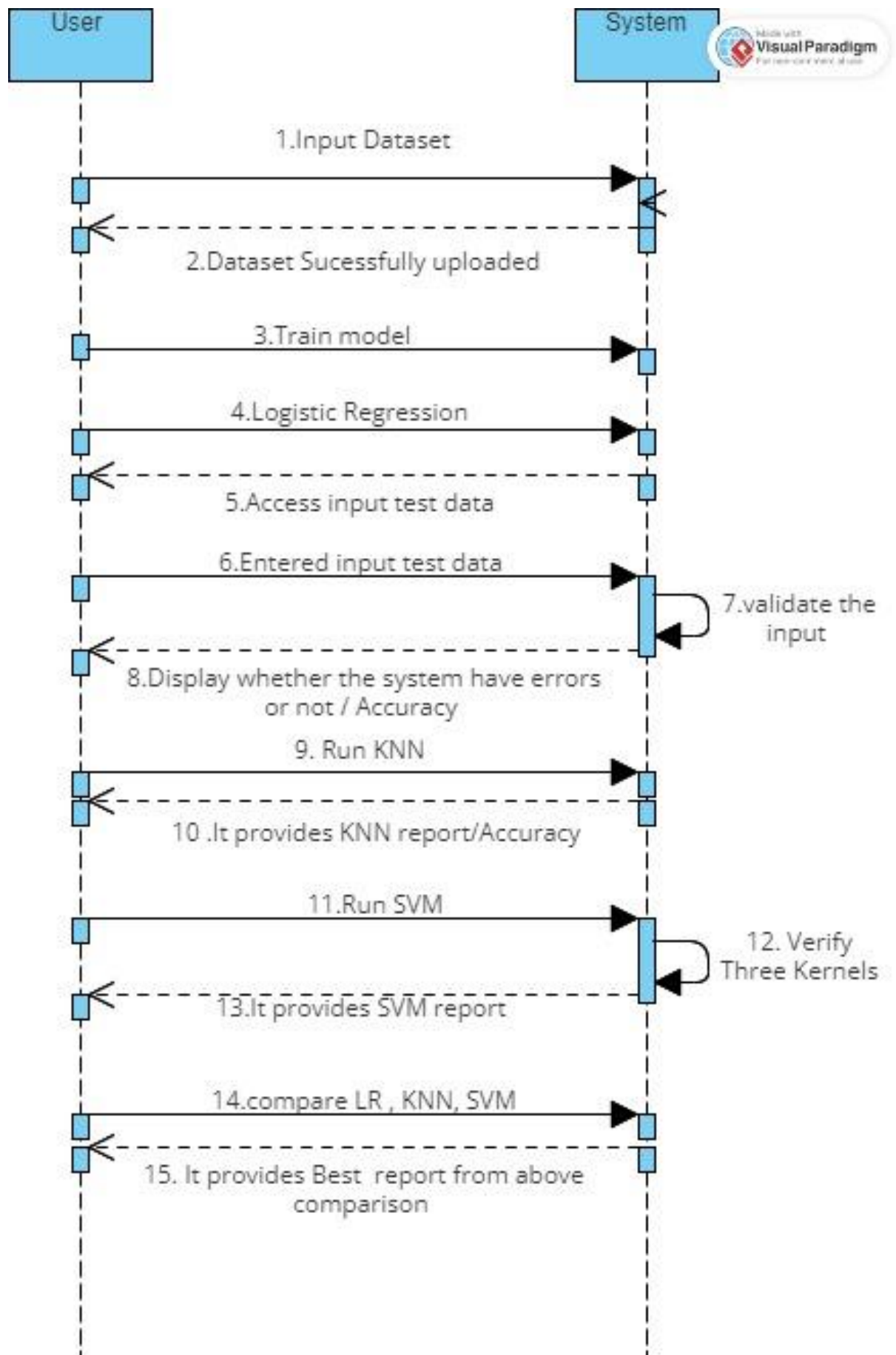


Fig 4.2.3 Sequence Diagram

5. TECHNOLOGIES LEARNT

WHAT IS PYTHON:

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc..)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

EXTENSIVE LIBRARIES

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

EMBEDDABLE:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

IMPROVED PRODUCTIVITY

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do.

SIMPLE AND EASY

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite **easy to learn, understand, and code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

READABLE

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

OBJECT-ORIENTED

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

FREE AND OPEN-SOURCE

Like we said earlier, Python is **freely available**. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it.

PORTABLE

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere** (WORA).

INTERPRETED

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

WHAT IS MACHINE LEARNING

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here

CATEGORIES OF MACHINE LEARNING:

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

APPLICATIONS OF MACHINES LEARNING:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

VARIOUS ML CONCEPTS

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

a. Terminologies of Machine Learning

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

b. TYPES OF MACHINE LEARNING

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

MODULES USED IN PROJECT:

NUMPY

Numpy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

PANDAS

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

JUPYTER NOTEBOOK

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

ANACONDA:

What is Anaconda Python?

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages.

This work environment, Anaconda is used for scientific computing, data science, statistical analysis, and machine learning. The latest version of Anaconda 5.0.1 is released in October 2017. The released version 5.0.1 addresses some minor bugs and adds useful features, such as updated R language support. All of these features weren't available in the original 5.0.0 release. This package manager is also an environment manager, a Python distribution, and a collection of open source packages and contains more than 1000 R and Python Data Science Packages.

Why Anaconda for Python?

There's no big reason to switch to Anaconda if you are completely happy with your regular python. But some people like data scientists who are not full-time developers, find anaconda much useful as it simplifies a lot of common problems a beginner runs into.

Anaconda can help with –

- Installing Python on multiple platforms
- Separating out different environments
- Dealing with not having correct privileges and

Getting up and running with specific packages and libraries

5.1 ALGORITHMS USED IN PROJECT

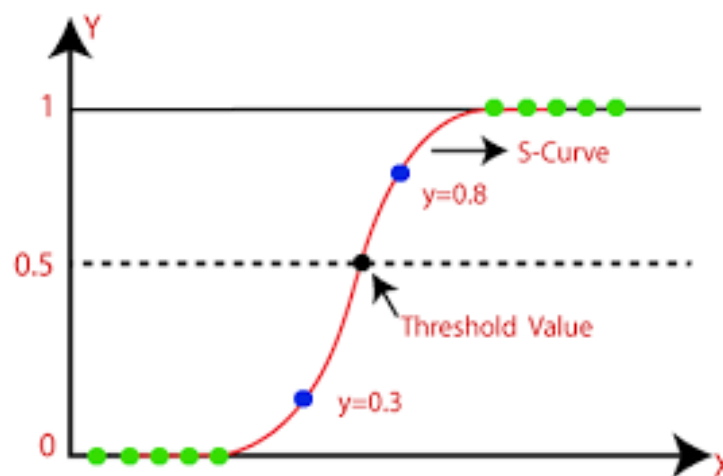
LOGISTIC REGRESSION:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic regression is used for solving the classification problems.

Types of logistic Regression

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".



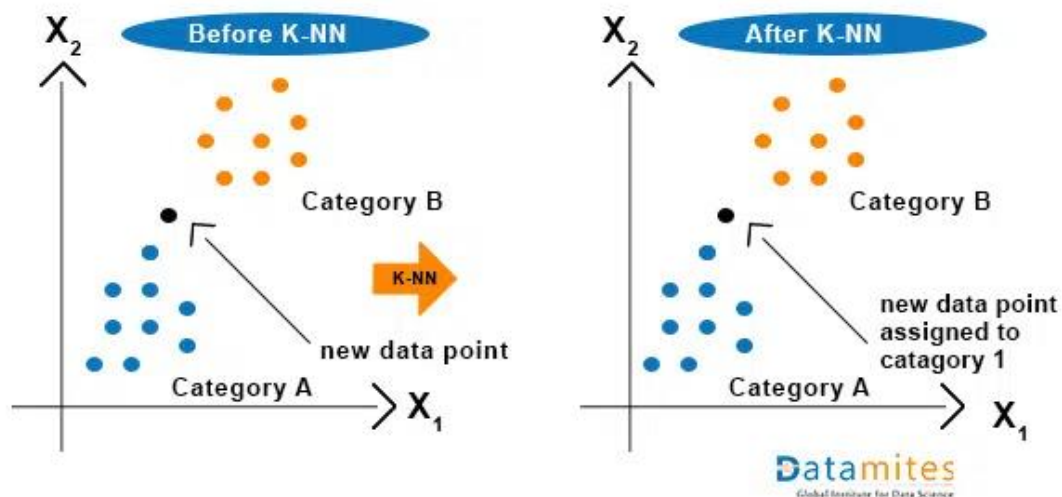
K-NEAREST NEIGHBOR (KNN)

KNN is a simple, supervised machine learning (ML) algorithm that can be used for classification or regression tasks - and is also frequently used in missing value imputation. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points. By choosing K , the user can select the number of nearby observations to use in the algorithm.

K-NN is a non-parametric algorithm. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Steps to execute

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.



SUPPORT VECTOR MACHINE (SVM)

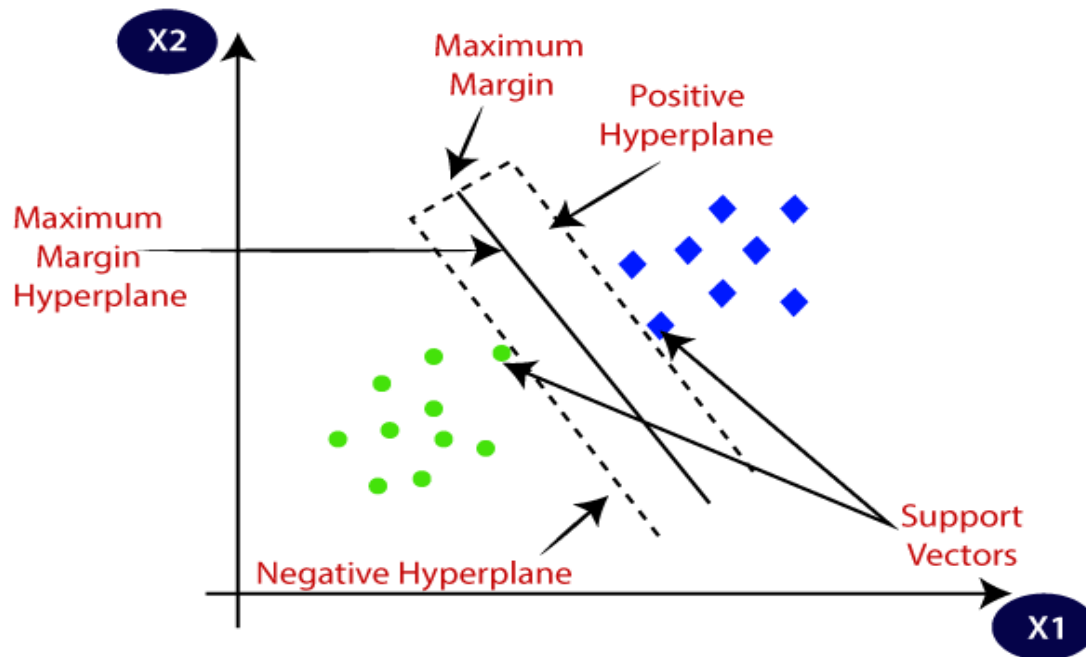
Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Support Vectors – Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

Hyperplane – It is a decision plane or space which is divided between a set of objects having different classes.

Margin – It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin. The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps –

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly



KERNEL

SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space.

Types of Kernels

- **Linear Kernel**
- **Polynomial Kernel**
- **Radical Basis Function (RBF) Kernel**

LINER KERNEL

It can be used as a dot product between any two observations. The formula of linear kernel is as below

$$K(x, x_i) = \sum (x * x_i)$$

Here x, x_i are input variables.

POLYNOMIAL KERNEL

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel

$$K(x, x_i) = 1 + \sum (x * x_i)^d$$

Here d is a degree of polynomial.

RADICAL BASIS FUNCTION (RBF) KERNEL

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically

$$K(x, x_i) = \exp(-\gamma \sum (x * x_i^2))$$

Here, gamma ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of gamma is 0.1.

5.2 CODE

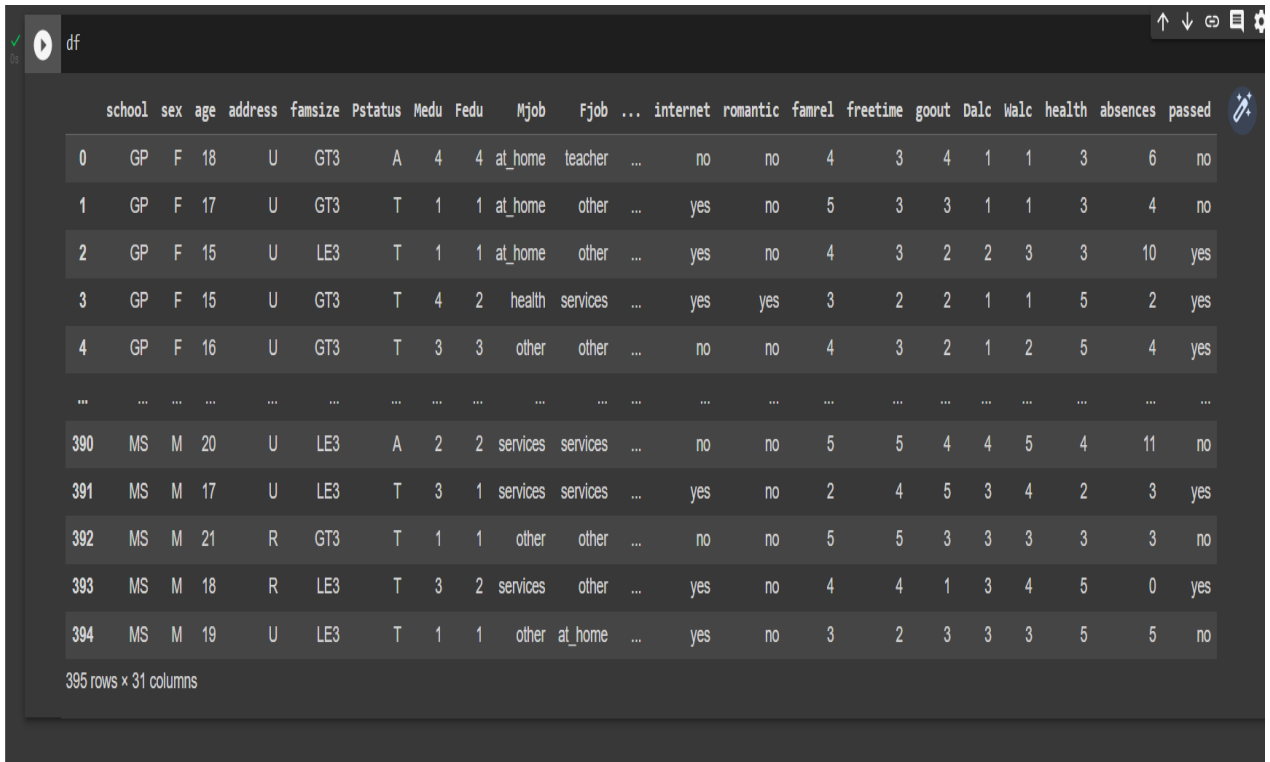
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from time import time
from astropy.table import Table
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, roc_curve, accuracy_score, f1_score,
roc_auc_score, classification_report, mean_squared_error
```

##Uploading Dataset

```
from google.colab import files
uploaded=files.upload()
df=pd.read_csv('student-data.csv')
dfv=pd.read_csv('student-data.csv')
```

```
df
```

Output :



	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	internet	romantic	famrel	freetime	goout	Dalc	Walc	health	absences	passed
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	no	no	4	3	4	1	1	3	6	no
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	yes	no	5	3	3	1	1	3	4	no
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	yes	no	4	3	2	2	3	3	10	yes
3	GP	F	15	U	GT3	T	4	2	health	services	...	yes	yes	3	2	2	1	1	5	2	yes
4	GP	F	16	U	GT3	T	3	3	other	other	...	no	no	4	3	2	1	2	5	4	yes
...
390	MS	M	20	U	LE3	A	2	2	services	services	...	no	no	5	5	4	4	5	4	11	no
391	MS	M	17	U	LE3	T	3	1	services	services	...	yes	no	2	4	5	3	4	2	3	yes
392	MS	M	21	R	GT3	T	1	1	other	other	...	no	no	5	5	3	3	3	3	3	no
393	MS	M	18	R	LE3	T	3	2	services	other	...	yes	no	4	4	1	3	4	5	0	yes
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	yes	no	3	2	3	3	3	5	5	no

395 rows x 31 columns

##Data Preprocessing & Visualization

```
def numerical_data():
```

```
    df['school'] = df['school'].map({'GP': 0, 'MS': 1})
    df['sex'] = df['sex'].map({'M': 0, 'F': 1})
    df['address'] = df['address'].map({'U': 0, 'R': 1})
    df['famsize'] = df['famsize'].map({'LE3': 0, 'GT3': 1})
    df['Pstatus'] = df['Pstatus'].map({'T': 0, 'A': 1})
    df['Mjob'] = df['Mjob'].map({'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4})
    df['Fjob'] = df['Fjob'].map({'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4})
    df['reason'] = df['reason'].map({'home': 0, 'reputation': 1, 'course': 2, 'other': 3})
    df['guardian'] = df['guardian'].map({'mother': 0, 'father': 1, 'other': 2})
    df['schoolsup'] = df['schoolsup'].map({'no': 0, 'yes': 1})
    df['famsup'] = df['famsup'].map({'no': 0, 'yes': 1})
    df['paid'] = df['paid'].map({'no': 0, 'yes': 1})
    df['activities'] = df['activities'].map({'no': 0, 'yes': 1})
    df['nursery'] = df['nursery'].map({'no': 0, 'yes': 1})
    df['higher'] = df['higher'].map({'no': 0, 'yes': 1})
    df['internet'] = df['internet'].map({'no': 0, 'yes': 1})
    df['romantic'] = df['romantic'].map({'no': 0, 'yes': 1})
    df['passed'] = df['passed'].map({'no': 0, 'yes': 1})
```

```
col = df['passed']
del df['passed']
df['passed'] = col
```

```
def feature_scaling(df):
    for i in df:
        col = df[i]
        if(np.max(col)>6):
            Max = max(col)
            Min = min(col)
            mean = np.mean(col)
            col = (col-mean)/(Max)
            df[i] = col
        elif(np.max(col)<6):
            col = (col-np.min(col))
            col /= np.max(col)
            df[i] = col
```

```
numerical_data()
df
```

Output:

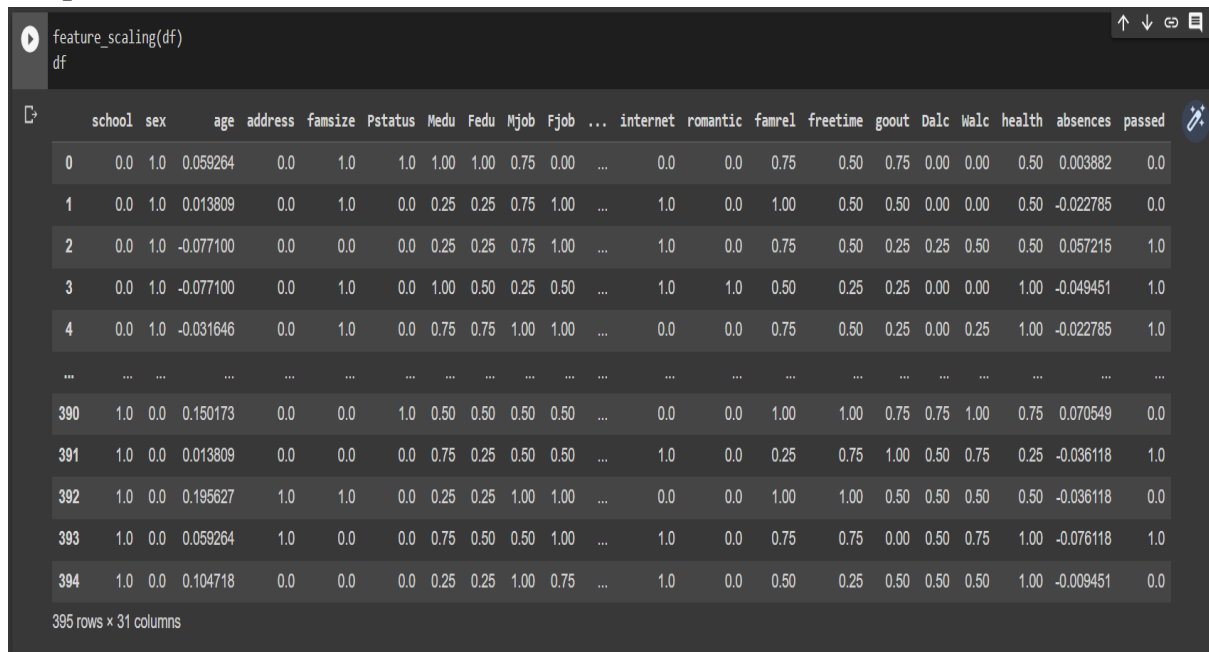
numerical_data()
df

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	internet	romantic	famrel	freetime	goout	Dalc	Walc	health	absences	passed
0	0	1	18	0	1	1	4	4	3	0	...	0	0	4	3	4	1	1	3	6	0
1	0	1	17	0	1	0	1	1	3	4	...	1	0	5	3	3	1	1	3	4	0
2	0	1	15	0	0	0	1	1	3	4	...	1	0	4	3	2	2	3	3	10	1
3	0	1	15	0	1	0	4	2	1	2	...	1	1	3	2	2	1	1	5	2	1
4	0	1	16	0	1	0	3	3	4	4	...	0	0	4	3	2	1	2	5	4	1
...
390	1	0	20	0	0	1	2	2	2	2	...	0	0	5	5	4	4	5	4	11	0
391	1	0	17	0	0	0	3	1	2	2	...	1	0	2	4	5	3	4	2	3	1
392	1	0	21	1	1	0	1	1	4	4	...	0	0	5	5	3	3	3	3	3	0
393	1	0	18	1	0	0	3	2	2	4	...	1	0	4	4	1	3	4	5	0	1
394	1	0	19	0	0	0	1	1	4	3	...	1	0	3	2	3	3	3	5	5	0

395 rows x 31 columns

```
feature_scaling(df)
df
```

Output:



```
feature_scaling(df)
df
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	internet	romantic	famrel	freetime	goout	Dalc	Walc	health	absences	passed
0	0.0	1.0	0.059264	0.0	1.0	1.0	1.00	1.00	0.75	0.00	...	0.0	0.0	0.75	0.50	0.75	0.00	0.00	0.50	0.003882	0.0
1	0.0	1.0	0.013809	0.0	1.0	0.0	0.25	0.25	0.75	1.00	...	1.0	0.0	1.00	0.50	0.50	0.00	0.00	0.50	-0.022785	0.0
2	0.0	1.0	-0.077100	0.0	0.0	0.0	0.25	0.25	0.75	1.00	...	1.0	0.0	0.75	0.50	0.25	0.25	0.50	0.50	0.057215	1.0
3	0.0	1.0	-0.077100	0.0	1.0	0.0	1.00	0.50	0.25	0.50	...	1.0	1.0	0.50	0.25	0.25	0.00	0.00	1.00	-0.049451	1.0
4	0.0	1.0	-0.031646	0.0	1.0	0.0	0.75	0.75	1.00	1.00	...	0.0	0.0	0.75	0.50	0.25	0.00	0.25	1.00	-0.022785	1.0
...
390	1.0	0.0	0.150173	0.0	0.0	1.0	0.50	0.50	0.50	0.50	...	0.0	0.0	1.00	1.00	0.75	0.75	1.00	0.75	0.070549	0.0
391	1.0	0.0	0.013809	0.0	0.0	0.0	0.75	0.25	0.50	0.50	...	1.0	0.0	0.25	0.75	1.00	0.50	0.75	0.25	-0.036118	1.0
392	1.0	0.0	0.195627	1.0	1.0	0.0	0.25	0.25	1.00	1.00	...	0.0	0.0	1.00	1.00	0.50	0.50	0.50	0.50	-0.036118	0.0
393	1.0	0.0	0.059264	1.0	0.0	0.0	0.75	0.50	0.50	1.00	...	1.0	0.0	0.75	0.75	0.00	0.50	0.75	1.00	-0.076118	1.0
394	1.0	0.0	0.104718	0.0	0.0	0.0	0.25	0.25	1.00	0.75	...	1.0	0.0	0.50	0.25	0.50	0.50	0.50	1.00	-0.009451	0.0

395 rows x 31 columns

Data Shape

```
df.shape
```

Output : (395, 31)

##Dropping null records

```
df.dropna().shape
```

Output : (395, 31)

##Feature Naming

```
features=['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
```

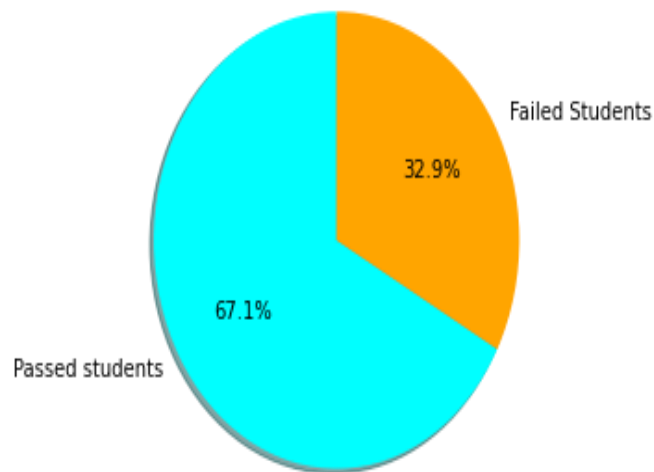
###Plotting students' status based on dataset

```
dfv['passed'].value_counts()
```

Output :

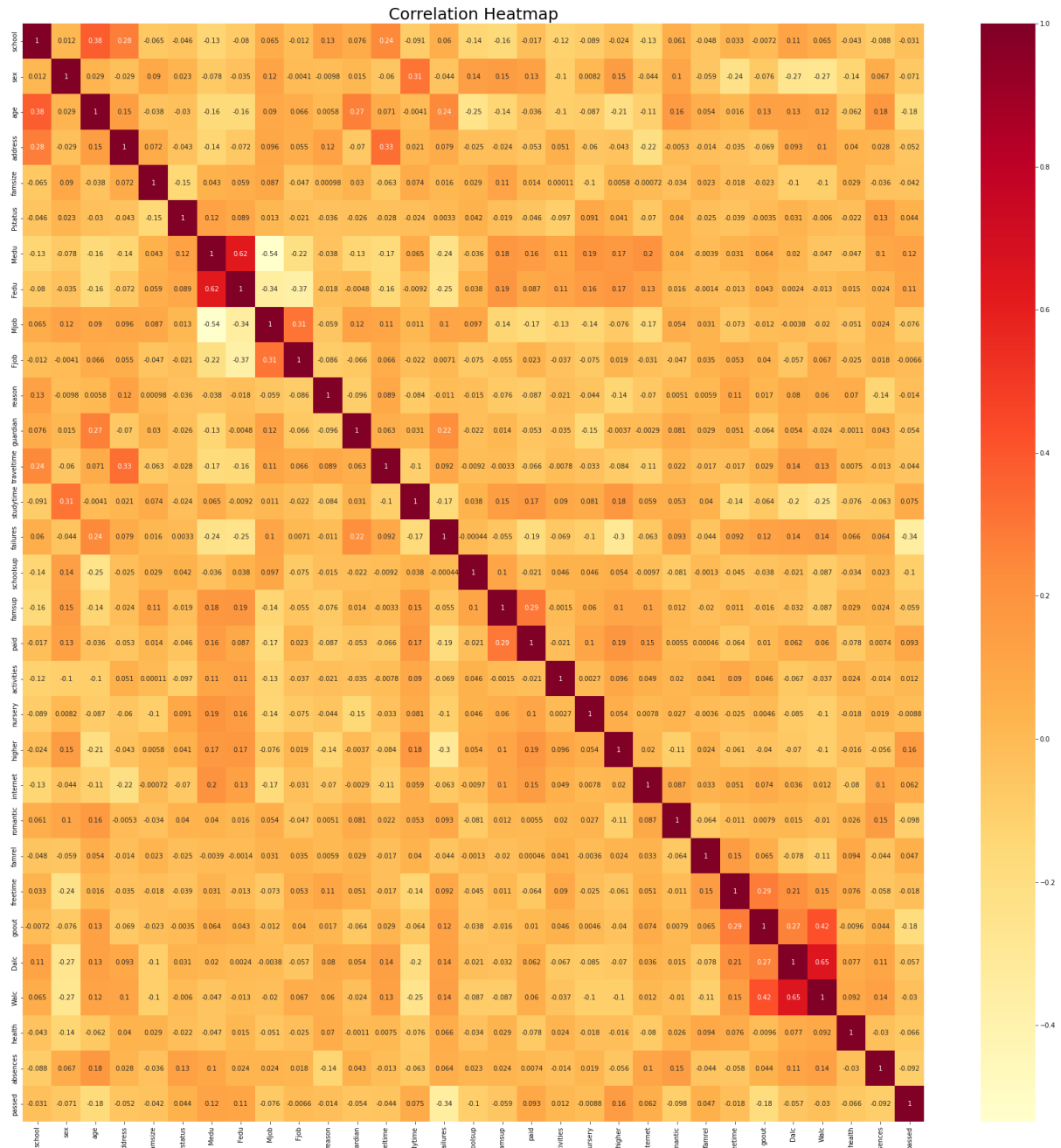
Yes 265
No 130
Name : Passed, dtype: int64

```
labels= 'Passed students', 'Failed Students'  
sizes=[265,130]  
colors=['cyan', 'orange']  
fig1,ax1= plt.subplots()  
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, shadow=True, startangle=90)  
ax1.axis('equal') #to draw circle  
plt.show()
```

Output:**###Correlation heatmap : To find the potential relationship between 2 numeric variables**

```
corr=df.corr()  
plt.figure(figsize=(31,31))  
sns.heatmap(corr,annot=True,cmap='YlOrRd')  
plt.title('Correlation Heatmap',fontsize=25)
```


Output:



##1.Logistic Regression

#Splitting the data for training(70%) & testing(30%)

data= df.to_numpy() ##to convert into array

n= data.shape[1] ##no.of rows

x= data[:,0:n-1] ##(factors i.e school,sex.....absences)

y= data[:,n-1] ##(result i.e passed)

x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.3,random_state=0)

##now we will work on x_train,y_train. These are the samples for training our model

logiRegr = LogisticRegression(C=1)

##Training data

logiRegr.fit(x_train,y_train)

Output : LogisticRegression(C=1)

##Making predictions with trained data

```
y_pred=logiRegr.predict(x_test)
```

```
y_pred
```

Output :

```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
        1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0.,
        1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
        1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1.]])
```

####Score: Here after the model is trained with the data, it will predict with data x_test and compare it with actual predictions(y_test).

```
Sctest=logiRegr.score(x_test,y_test)
```

```
Sctrain=logiRegr.score(x_train,y_train)
```

```
print('#Accuracy test is: ',Sctest)
```

```
print('#Accuracy train is: ',Sctrain)
```

```
f1 = f1_score(y_test, y_pred, average='macro')
```

```
##use macro when all classes hav equal importance, otherwise use mini
```

```
print("\n#f1 score is: ',f1)
```

Output:

```
#Accuracy test is: 0.6386554621848739
```

```
#Accuracy train is: 0.7463768115942029
```

```
#f1 score is: 0.5533734834598935
```

```
confusion_matrix(y_test, y_pred)
```

Output:

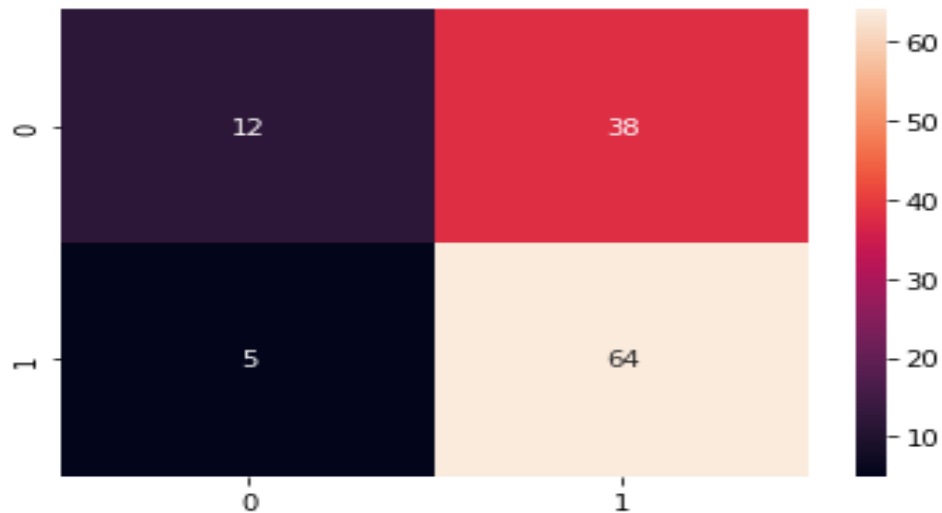
```
array([[12, 38],
```

```
       [ 5, 64]])
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
sns.heatmap(cm,annot=True)
```

Output:



```
#import classification_report  
print(classification_report(y_test, y_pred))
```

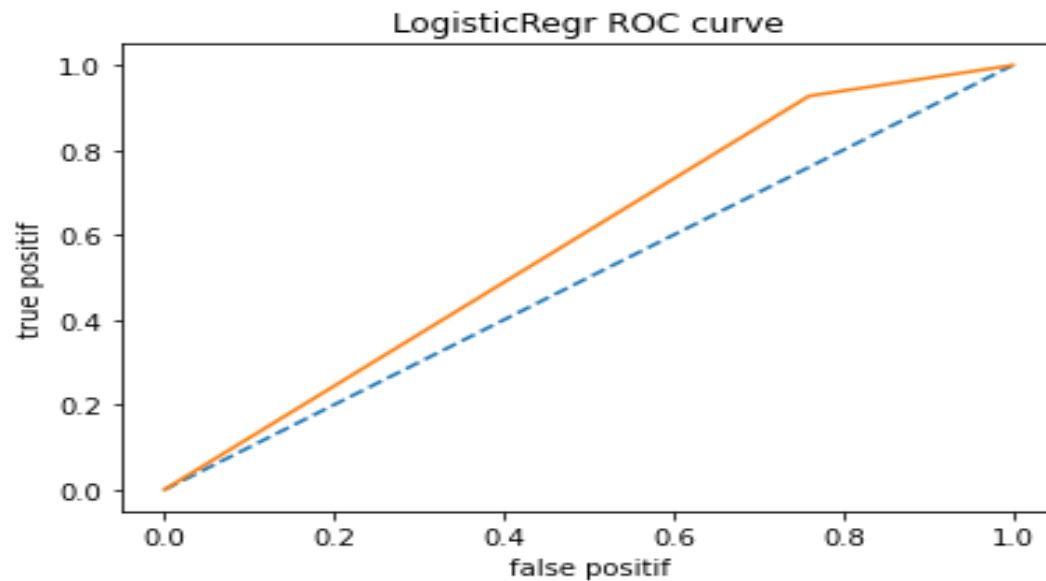
Output:

	precision	recall	f1-score	support
0.0	0.71	0.24	0.36	50
1.0	0.63	0.93	0.75	69
accuracy			0.64	119
macro avg	0.67	0.58	0.55	119
weighted avg	0.66	0.64	0.58	119

#support: number of actual occurrences of the class in the specified dataset.

```
#ploting the roc_curve  
fpositif, tpositif, thresholds = roc_curve(y_test, y_pred)  
plt.plot([0,1],[0,1], '--')  
plt.plot(fpositif, tpositif, label='LogisticRegr')  
plt.xlabel('false positif')  
plt.ylabel('true positif')  
plt.title('LogisticRegr ROC curve')  
p=plt.show()
```

Output:



```
auc = np.round(roc_auc_score(y_test, y_pred), 3)
print("\nArea Under Curve(AUC) is ", auc)
mse_log = np.round(math.sqrt(mean_squared_error(y_test, y_pred)),3)
print('Root mean square error', mse_log)
```

Output:

Area Under Curve(AUC) is 0.584
Root mean square error 0.601

**#####Here Thresholds= (max value of y_pred+1, y_pred range(decreasing))
i.e Thresholds=[1.0+1.0, 1.0, 0.0] => [2.0, 1.0, 0.0]**

```
print("True Positive Rate ",list(tpositif))
print("False Positive Rate ",list(fpositif))
print("Thresholds ", list(thresholds))
```

Output:

True Positive Rate [0.0, 0.927536231884058, 1.0]
False Positive Rate [0.0, 0.76, 1.0]
Thresholds [2.0, 1.0, 0.0]

#This code is to find out the best optimal state by checking different random states. Since we are taking large no.of cases it will take large time to
to check evrytime.

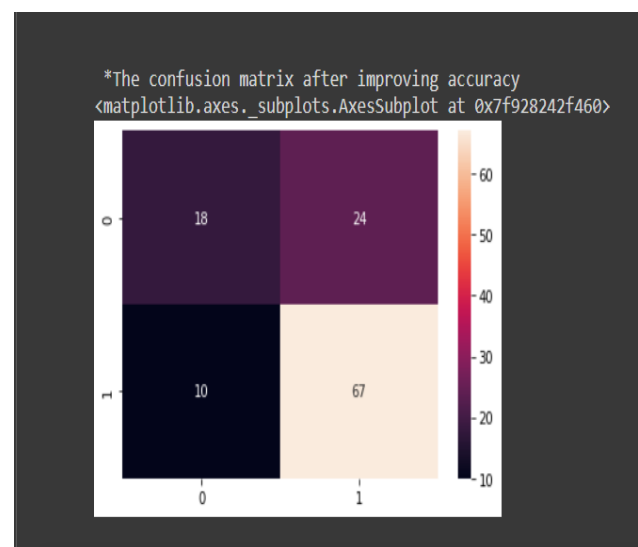
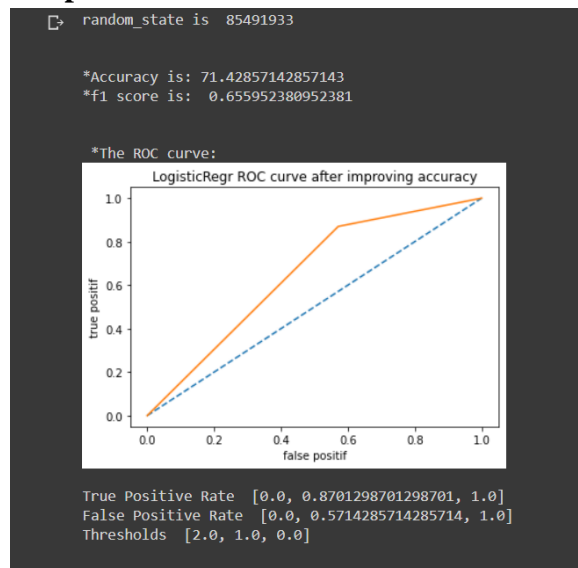
```
max_iteration = 0
maxF1 = 0
```

```

maxAccuracy = 0
optimal_state = 0
import random
for k in range(max_iteration):
    print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+ ' , Current f1 : '+str(maxF1), end="\r")
    split_state = np.random.randint(1,1000000)-1
    x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.3, random_state = split_state)
    logisticRegr = LogisticRegression(C=1)
    logisticRegr.fit(x_train,y_train)
    y_pred=logisticRegr.predict(x_test)
    f1 = f1_score(y_test, y_pred, average='macro')
    accuracy = accuracy_score(y_test, y_pred)*100
    if (accuracy>maxAccuracy and f1>maxF1):
        maxF1 = f1
        maxAccuracy = accuracy
        optimal_state = split_state
#After checking, the random state "85491933", is giving best accuracy
optimal_state = 85491933
print ('random_state is ',optimal_state)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=optimal_state)
logisticRegr = LogisticRegression(C=1)
logisticRegr.fit(x_train,y_train)
y_pred=logisticRegr.predict(x_test)
f1 = f1_score(y_test, y_pred, average='macro')
accuracy = accuracy_score(y_test, y_pred)*100
print("\n\n*Accuracy is: "+str(accuracy)+"\n*f1 score is: ",f1)
yt_lg,yp_lg = y_test,y_pred
print ( '\n\n *The ROC curve: ')
fpositif, tpositif, thresholds = roc_curve(y_test, y_pred)
plt.plot([0,1],[0,1], '--')
plt.plot(fpositif,tpositif, label='LogisticRegr')
plt.xlabel('false positif')
plt.ylabel('true positif')
plt.title('LogisticRegr ROC curve after improving accuracy')
p=plt.show()
print("\nTrue Positive Rate ",list(tpositif))
print("False Positive Rate ",list(fpositif))
print("Thresholds ", list(thresholds))
#visualizig the confusion matrix:
print ('\n\n\n *The confusion matrix after improving accuracy')
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm,annot=True)

```

Output:



```
auc = np.round(roc_auc_score(y_test, y_pred), 3)
print("\nArea Under Curve(AUC) is after improving accuracy", auc)
```

Output:

Area Under Curve(AUC) is after improving accuracy 0.649

#Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data

```
mse_log = np.round(math.sqrt(mean_squared_error(y_test, y_pred)),3)
print('Root mean square error', mse_log)
```

Output:

Root mean square error 0.535

##2. K-Nearest Neighbor Algorithm

#defining the data

```
y=df.passed
```

```
target=["passed"]
```

```
x = df.drop(target,axis = 1 )
```

#So here, the x is input i.e features(school,sex....absences) and y is output i.e passed/not

```
max_iteration = 0
```

```
maxF1 = 0
```

```
maxAccuracy = 0
```

```
optimal_state = 0
```

```
for k in range(max_iteration):
```

```

        print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+ ' , Current f1 : '+str(maxF1), end="\r")
        split_state = np.random.randint(1,1000000000)-1
        x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=split_state)
        KNN = KNeighborsClassifier()
        KNN.fit(x_train,y_train)
        y_pred=KNN.predict(x_test)
        f1 = f1_score(y_test, y_pred, average='macro')
        accuracy = accuracy_score(y_test, y_pred)*100

        if (accuracy>maxAccuracy and f1>maxF1):
            maxF1 = f1
            maxAccuracy = accuracy
            optimal_state = split_state
    optimal_state = 71027429
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=optimal_state)
    KNN= KNeighborsClassifier()
    KNN.fit(x_train,y_train)
    y_pred=KNN.predict(x_test)
    f1 = f1_score(y_test, y_pred, average='macro')
    accuracy = accuracy_score(y_test, y_pred)*100

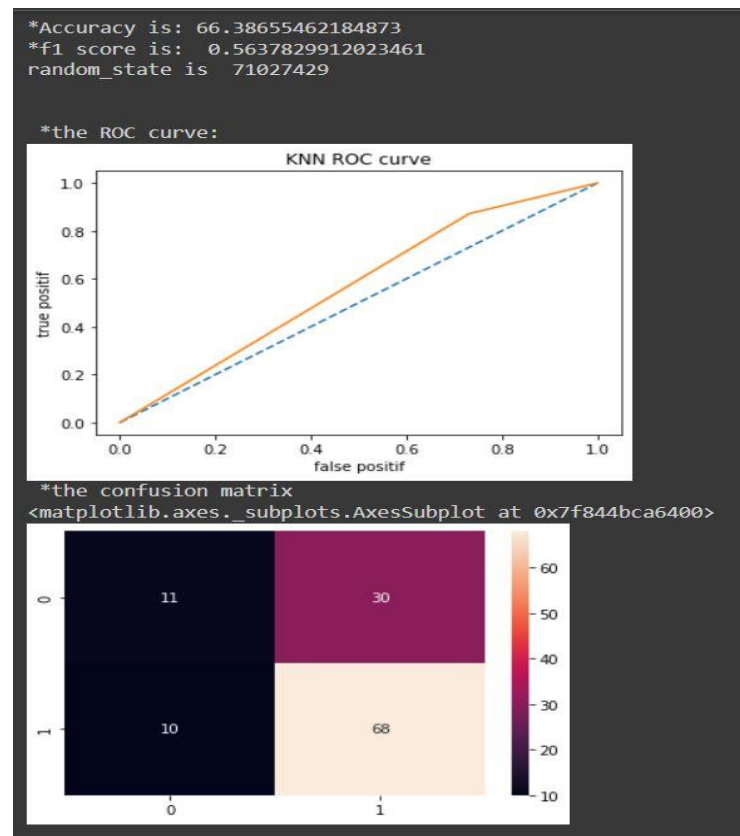
    print("\n\n\n*Accuracy is: "+str(accuracy)+"\n*f1 score is: ',f1)
    print ('random_state is ',optimal_state)

#ploting the roc_curve
print ( '\n\n *the ROC curve: ')
fpositif, tpositif, thresholds = roc_curve(y_test, y_pred)
plt.plot([0,1],[0,1], '--')
plt.plot(fpositif, tpositif, label='knn')
plt.xlabel('false positif')
plt.ylabel('true positif')
plt.title('KNN ROC curve')
p=plt.show()
yt_knn, yp_knn= y_test, y_pred

#visualizig the confusion matrix
print ( ' *the confusion matrix ')
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm,annot=True)

```

Output:



```
auc = np.round(roc_auc_score(y_test, y_pred), 3)
print("\nArea Under Curve(AUC) is ", auc)
```

Output:

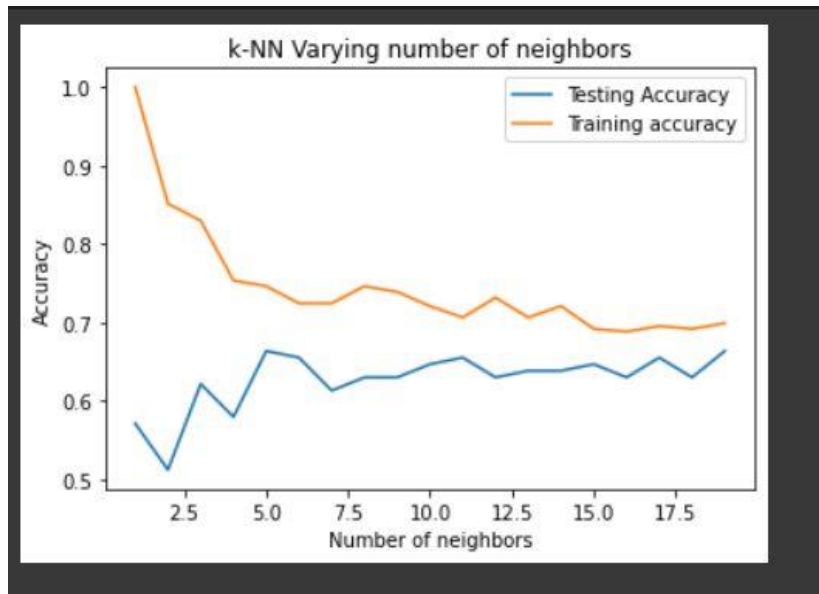
Area Under Curve(AUC) is 0.57

```
#Setup arrays to store training and test accuracies
neighbors= np.arange(1,20)
train_accuracy =np.empty(19)
test_accuracy = np.empty(19)
for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)
    #Fit the model
    knn.fit(x_train, y_train)
    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(x_train, y_train)
    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(x_test, y_test)

# Plotting the curve
plt.title('k-NN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
```



```
plt.ylabel('Accuracy')
plt.show()
```



Output:

```
#In case of classifier like knn the parameter to be tuned is n_neighbors
param_grid = {'n_neighbors':np.arange(1,20)}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(x_train,y_train)
#best score\n",
knn_cv.best_score_
```

Output:

0.7065584415584415

```
knn_cv.best_params_
```

Output:

{'n_neighbors': 12}

```
param_grid = {'n_neighbors':np.arange(1,20)}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(x_test,y_test)
#best score\n",
knn_cv.best_score_
```

Output:

0.6717391304347827

```
knn_cv.best_params_
```

Output:

{'n_neighbors': 18}

```

param_grid = {'n_neighbors':np.arange(1,20)}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(x,y)
#best score\n",
knn_cv.best_score_

```

Output:

0.6734177215189873

knn_cv.best_params_

Output:

```

{'n_neighbors': 7}
params = {"n_neighbors":[7,18] , "metric":["euclidean", "manhattan", "chebyshev"]}
acc = {}
for m in params["metric"]:
    acc[m] = []
    for k in params["n_neighbors"]:
        print("Model_{ } metric: { }, n_neighbors: { }".format(i, m, k))
        i += 1
        t = time()
        knn = KNeighborsClassifier(n_neighbors=k, metric=m)
        knn.fit(x_train,y_train)
        pred = knn.predict(x_test)
        print("Time: ", time() - t)
        acc[m].append(accuracy_score(y_test, y_pred))
        print("Acc: ", acc[m][-1])

```

Output:

```

➤ Model_18 metric: euclidean, n_neighbors: 7
Time: 0.01066446304321289
Acc: 0.6638655462184874
Model_19 metric: euclidean, n_neighbors: 18
Time: 0.010305643081665039
Acc: 0.6638655462184874
Model_20 metric: manhattan, n_neighbors: 7
Time: 0.011976003646850586
Acc: 0.6638655462184874
Model_21 metric: manhattan, n_neighbors: 18
Time: 0.011404991149902344
Acc: 0.6638655462184874
Model_22 metric: chebyshev, n_neighbors: 7
Time: 0.01051473617553711
Acc: 0.6638655462184874
Model_23 metric: chebyshev, n_neighbors: 18
Time: 0.011378288269042969
Acc: 0.6638655462184874

```

```

max_iteration = 0
axF1 = 0
maxAccuracy = 0
optimal_state = 0
f1 = 0
accuracy = 0
True60 = False
for k in range(max_iteration):
    print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+ ' , Current f1 : '+str(maxF1), end="\r")
    split_state = np.random.randint(1,1000000000)-1
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3, random_state = split_state)
    KNN = KNeighborsClassifier(n_neighbors=7,metric='chebyshev')
    KNN.fit(x_train,y_train)
    y_pred=KNN.predict(x_test)
    f1 = f1_score(y_test, y_pred, average='macro')
    accuracy = accuracy_score(y_test, y_pred)*100
    if accuracy>maxAccuracy and f1>=0.5:
        maxF1 = f1
        maxAccuracy = accuracy
        optimal_state = split_state
        if maxAccuracy>79:
            break
optimal_state = 54160100
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=optimal_state)
KNN_f= KNeighborsClassifier(n_neighbors=7,metric='chebyshev')
KNN_f.fit(x_train,y_train)
y_pred=KNN_f.predict(x_test)
f1 = f1_score(y_test, y_pred, average='macro')
accuracy = accuracy_score(y_test, y_pred)*100
print("\n\n*Accuracy is: "+str(accuracy)+"\n*f1 score is: ',f1)
print ('random_state is ',optimal_state)
yt_knn,yp_knn= y_test,y_pred

```

Output:

```

*Accuracy is: 68.90756302521008
*f1 score is: 0.40796019900497515
random_state is 54160100

```

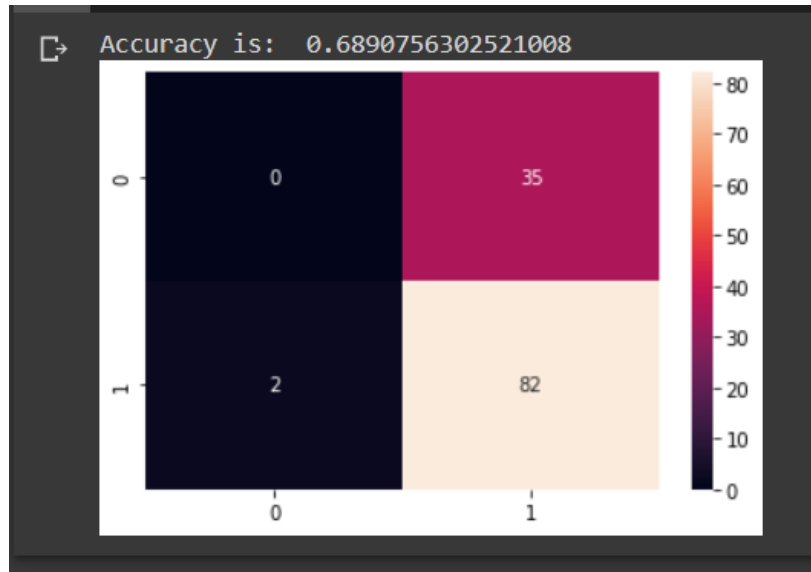
1. Confusion Matrix

```

ac = accuracy_score(yt_knn,yp_knn)
print('Accuracy is: ',ac)

```

```
cm= confusion_matrix(yt_knn,yp_knn)
sns.heatmap(cm,annot=True)
yt_knn,yp_knn = y_test,y_pred
Output:
```



```
mse_knn = np.round(math.sqrt(mean_squared_error(yt_knn, yp_knn)),3)
print('Root mean square error', mse_knn)
```

Output:

Root mean square error 0.558

2.Classification Report

```
print(classification_report(y_test,y_pred,zero_division=0))
```

Output:

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	35
1.0	0.70	0.98	0.82	84
accuracy			0.69	119
macro avg	0.35	0.49	0.41	119
weighted avg	0.49	0.69	0.58	119

##3.Support Vector Machine(SVM)

Show results of every model

```
def showResults(accuracy, trainingTime, y_pred,model):
```

```

print('-----Results :',model,'-----')
print('-----')
confusionMatrix = confusion_matrix(y_test, y_pred)
print("\n The ROC curve is :\n")
fig, _ = plt.subplots()
fpr,tpr,thresholds=roc_curve(y_test,y_pred)
plt.plot([0, 1],[0, 1], '--')
plt.plot(fpr,tpr,label=model)
plt.xlabel('false positive')
plt.ylabel('false negative')
plt.legend()
fig.suptitle('ROC curve: '+str(model))
plt.show()
print('-----')
print('The model accuracy:', round(accuracy), '%')
print('-----')
print('The training time is: ',trainingTime)
print('-----')
print('The f1 score is :',round(100*f1_score(y_test, y_pred, average='macro'))/100)
print('-----')
print('The roc_auc_score is :',round(100*roc_auc_score(y_test, y_pred))/100)
print('-----')
print('The confusion matrix is :\n')
ax = plt.axes()
sns.heatmap(confusionMatrix,annot=True)
# -----
# -----
# Hyperparameter Tuning :
# C, degree and gamma are the parameters that are used in SVM classfier
'svc(C=...,...),svc(C,degree=..)',svc(C,gamma=..)
# The following functions will return those values that minimize the error on (X_val,y_val) set
# So this (X_val,y_val) set will be used to get the optimal SVM parameters before evaluating
the model on the test set
# Optimal C
def optimal_C_value():
    Ci = np.array(( 0.0001,0.001,0.01,0.05,0.1,4,10,40,100))
    minError = float('Inf')
    optimal_C = float('Inf')

    for c in Ci:
        clf = SVC(C=c,kernel='linear')
        clf.fit(X_train, y_train)
        predictions = clf.predict(X_val)
        error = np.mean(np.double(predictions != y_val))
        if error < minError:

```

```

        minError = error
        optimal_C = c
    return optimal_C

# Optimal C and the degree of the polynomial
def optimal_C_d_values():
    Ci = np.array(( 0.0001,0.001,0.01,0.05,0.1,4,10,40,100))
    Di = np.array(( 2, 5, 10, 15, 20, 25, 30))
    minError = float('Inf')
    optimal_C = float('Inf')
    optimal_d = float('Inf')
    for d in Di:
        for c in Ci:
            clf = SVC(C=c,kernel='poly', degree=d)
            clf.fit(X_train, y_train)
            predictions = clf.predict(X_val)
            error = np.mean(np.double(predictions != y_val))
            if error < minError:
                minError = error
                optimal_C = c
                optimal_d = d
    return optimal_C,optimal_d

# Optimal C and gamma
def optimal_C_gamma_values():
    Ci = np.array(( 0.0001,0.001,0.01,0.05,0.1,4,10,40,100))
    Gi = np.array(( 0.000001,0.00001,0.01,1,2,3,5,20,70,100,500,1000))
    minError = float('Inf')
    optimal_C = float('Inf')
    optimal_g = float('Inf')

    for g in Gi:
        for c in Ci:
            clf = SVC(C=c,kernel='rbf', gamma=g)
            clf.fit(X_train, y_train)
            predictions = clf.predict(X_val)
            error = np.mean(np.double(predictions != y_val))
            if error < minError:
                minError = error
                optimal_C = c
                optimal_g = g
    return optimal_C,optimal_g

# -----
# -----
# Compare the three kernels

```

```

def compare_kernels():
    X_train1,X_val1,X_test1,y_train1,y_val1,y_test1 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state1)
    X_train2,X_val2,X_test2,y_train2,y_val2,y_test2 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state2)
    X_train3,X_val3,X_test3,y_train3,y_val3,y_test3 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state3)
    print('----- Comparison -----')
    print('\n')
    f11 = "{:.2f}".format(f1_score(y_test1, y_linear, average='macro'))
    f22 = "{:.2f}".format(f1_score(y_test2, y_poly, average='macro'))
    f33 = "{:.2f}".format(f1_score(y_test3, y_gauss, average='macro'))
    roc1 = "{:.2f}".format(roc_auc_score(y_test1, y_linear))
    roc2 = "{:.2f}".format(roc_auc_score(y_test2, y_poly))
    roc3 = "{:.2f}".format(roc_auc_score(y_test3, y_gauss))
    a1,a2 = confusion_matrix(y_test1, y_linear)[0],confusion_matrix(y_test1, y_linear)[1]
    b1,b2 = confusion_matrix(y_test2, y_poly)[0],confusion_matrix(y_test2, y_poly)[1]
    c1,c2 = confusion_matrix(y_test3, y_gauss)[0],confusion_matrix(y_test3, y_gauss)[1]
    data_rows = [('training time',time1, time2, time3),
        (",",",","),
        ('accuracy %',linear_accuracy, poly_accuracy, gauss_accuracy),
        (",",",","),
        ('confusion matrix',a1, b1, c1),
        ("",a2,b2,c2),
        (",",",","),
        ('f1 score',f11,f22,f33),
        (",",",","),
        ('roc_auc_score',roc1,roc2,roc3)]
    t = Table(rows=data_rows, names=('metric','Linear kernel', 'polynomial kernel', 'gaussian
kernel'))
    print(t)
    print("\n\n")
    print("The Roc curves :\n")
    y_pred1 = y_linear
    y_pred2 = y_poly
    y_pred3 = y_gauss
    fig, _ = plt.subplots()
    fig.suptitle('Comparison of three ROC curves')
    fpr,tpr,thresholds=roc_curve(y_test1,y_pred1)
    plt.plot([0, 1],[0, 1], '--')
    plt.plot(fpr,tpr,label='Linear kernel :'+str(roc1))
    plt.xlabel('false positive')
    plt.ylabel('false negative')

```

```

fpr,tpr,thresholds=roc_curve(y_test2,y_pred2)
plt.plot(fpr,tpr,label='Polynomial kernel :'+str(roc2))
fpr,tpr,thresholds=roc_curve(y_test3,y_pred3)
plt.plot(fpr,tpr,label='Gaussian kernel :'+str(roc3))
plt.legend()
plt.show()

# -----
# Print results of the choosen kernel

def best_kernel(kernel):
    X_train1,X_val1,X_test1,y_train1,y_val1,y_test1 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state1)
    X_train2,X_val2,X_test2,y_train2,y_val2,y_test2 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state2)
    X_train3,X_val3,X_test3,y_train3,y_val3,y_test3 =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state3)

    time = 0
    f1 = 0
    accuracy = 0
    rc = 0
    y = 0
    if kernel == 'linear kernel':
        time = time1
        f1 = "{:.2f}".format(f1_score(y_test1, y_linear, average='macro'))
        accuracy = round(100*linear_accuracy)/100
        rc = round(100*roc_auc_score(y_test1, y_linear))/100
        y_test = y_test1
        y = y_linear
    elif kernel == 'polynomial kernel':
        time = time2
        f1 = "{:.2f}".format(f1_score(y_test2, y_poly, average='macro'))
        accuracy = round(100*poly_accuracy)/100
        rc = round(100*roc_auc_score(y_test2, y_poly))/100
        y_test = y_test2
        y = y_poly
    else :
        time = time3
        f1 = "{:.2f}".format(f1_score(y_test3, y_gauss, average='macro'))
        accuracy = round(100*gauss_accuracy)/100
        rc = round(100*roc_auc_score(y_test3, y_gauss))/100
        y_test = y_test3

```



```

y = y_gauss

# used for comparing three classifiers(knn, logistic regression and svm)
yt_svm,yp_svm = y_test, y

print('The choosen kernel :',kernel)
print('the training :',time)
print('the accuracy :',round(accuracy),'%')
print('the f1 score :',f1)
print('The roc_auc_score is :',rc)
print('-----\nThe ROC curve :')
fig, _ = plt.subplots()
fpr, tpr, thresholds = roc_curve(y_test, y)
plt.plot([0, 1], [0, 1], '--')
plt.plot(fpr, tpr, label=kernel+' : '+str(rc))
plt.xlabel('false positive')
plt.ylabel('false negative')
plt.legend()
plt.show()
confusionMatrix = confusion_matrix(y_test, y)
print('-----\nThe confusion matrix is :')
ax = plt.axes()
sns.heatmap(confusionMatrix, annot=True)
ax.set_title('Confusion matrix of SVM '+str(kernel))
return yt_svm, yp_svm

# -----
# svm factor : factor affecting students performance, later on on this Ipython notebook we will
explain how we will do this

# 1) factor as svm coefficients
def factors(array, K, max_or_min, df):
    n = array.shape[1]
    array = array.reshape(n,1)
    my_list = array.tolist()

    if max_or_min == 'max':
        temp = sorted(my_list)[-K:]
        res = []
        for ele in temp:
            res.append(my_list.index(ele))
        return(get_factors(res, df))
    elif max_or_min == 'min':

```

```

temp = sorted(my_list, reverse=True)[-K:]
temp = temp = np.array(temp).reshape(K,1)
res = []
for ele in temp:
    if ele<0:
        res.append(my_list.index(ele))
    return(get_factors(res, df))
else:
    return
# 2) converts those factors to dataset columns name
def get_factors(index, df):
    f = []
    for i in index:
        f.append(df.columns[i])
    return f
# 3) Convert column names to understandable string

columns_name = {'famsize': 'family size', 'Pstatus': "parent's cohabitation status ", 'Medu':
"mother's education",
                'Fedu': "father's education", 'Mjob': "mother's job", 'Fjob': "father's job",
                'reason': 'reason to choose this school ', 'schoolsup': 'extra educational support',
'famsup': 'family educational support',
                'paid': 'extra paid classes within the course subject', 'higher': 'wants to take higher
education',
                'romantic': 'with a romantic relationship ', 'famrel': 'quality of family relationships',
'goout': 'going out with friends',
                'Dalc': 'workday alcohol consumption', 'Walc': 'weekend alcohol consumption'}
def column_to_string(fcts,max_or_min):

    if max_or_min == 'max':
        print('-----')
        print('Factors helping students succeed :')
    else:
        print('-----')
        print('-----')
        print('Factors leading students to failure')
    for fct in fcts:
        if fct in columns_name:
            print(columns_name[fct])
        else:
            print(fct)

# -----
# Splitting the data for SVM

```

```

# Here We will split data into test set, cross validation (X_val, y_val) set and training set
# The cross validation (X_val, y_val) is used for choosing the optimal value for svm parameters
C, degree and gamma
def split(df,rest_size,test_size,randomState):
    data = df.to_numpy()
    n = data.shape[1]
    x = data[:,0:n-1]
    y = data[:,n-1]
    if(randomState):
        X_train,X_rest,y_train,y_rest=
train_test_split(x,y,test_size=rest_size,random_state=randomState)
        X_val,X_test,y_val,y_test=
train_test_split(X_rest,y_rest,test_size=test_size,random_state=randomState)
    else:
        X_train,X_rest,y_train,y_rest = train_test_split(x,y,test_size=rest_size,random_state=0)
        X_val,X_test,y_val,y_test=
train_test_split(X_rest,y_rest,test_size=test_size,random_state=0)

    return X_train,X_val,X_test,y_train,y_val,y_test
# We will use the three different svm classifier kernels
# Linear kernel, polynomial kernel and gaussian kernel and we will choose the most accurate

```

#1.Linear Kernel

```

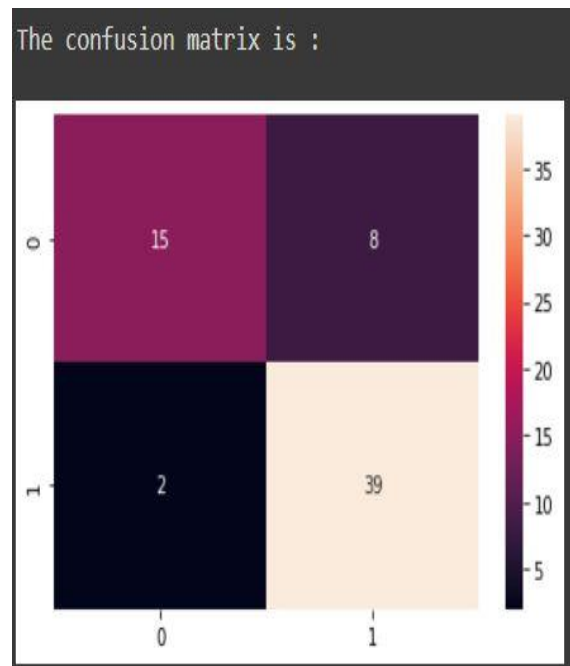
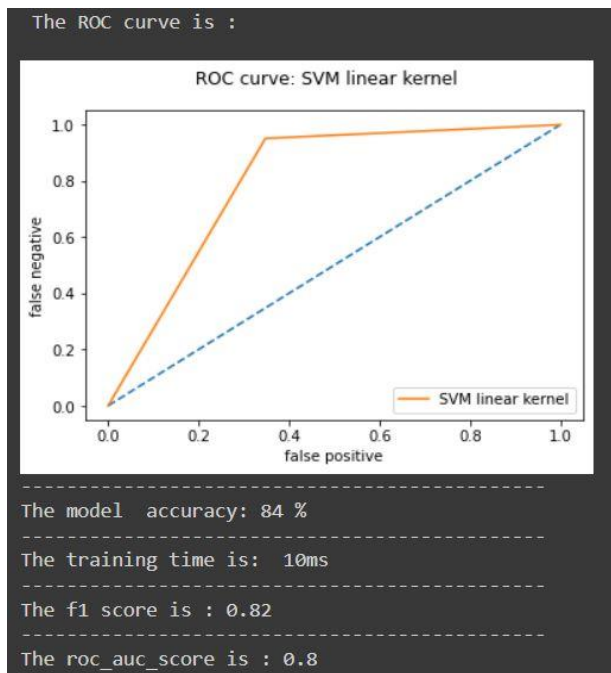
#####Linear Kernel#####
maxAccuracy = 0
maxF1 = 0
# We already tune parameters, we do not need to loop over all the hyperparamters again,
# if you want to do so just set max_iteration to 2000 for example
# and remove the line 'optimal_split_state = 388628375' at the bottom of this cell.
max_iteration = 0
if max_iteration != 0:
    print ('-----Hyperparameters tuning starts-----
-----\n\n')
    for k in range(max_iteration):
        print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+' Current f1 '+str(maxF1),
end="\r")
        # Let's get the optimal C value for the linear kernel
        split_state = np.random.randint(1,10000000000)-1
        X_train,X_val,X_test,y_train,y_val,y_test =
split(df,rest_size=0.4,test_size=0.4,randomState=split_state)
        optimal_C = optimal_C_value()
        # Now let's use the optimal C value
        linear_clf = SVC(C=optimal_C,kernel='linear')
        # Let's train the model with the optimal C value and calculate the training time
        tic = time()

```

```
linear_clf.fit(X_train, y_train)
toc = time()
time1 = str(round(1000*(toc-tic))) + "ms"
y_linear = linear_clf.predict(X_test)
linear_f1 = f1_score(y_test, y_linear, average='macro')
linear_accuracy = accuracy_score(y_test, y_linear)*100
if linear_accuracy>maxAccuracy and linear_f1>maxF1:
    maxAccuracy = linear_accuracy
    maxF1 = linear_f1
    optimal_split_state1 = split_state
if maxAccuracy>86 and maxF1>80:
    break;

# We've already tuned our hyperparameters, we will not repeat that again as it takes soo long.
# The optimal split state for linear kernel is 388628375
# Let's try that split state
optimal_split_state1 = 388628375
X_train,X_val,X_test,y_train,y_val,y_test = \
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state1)
optimal_C = optimal_C_value(
# Now let's use the optimal C value
linear_clf = SVC(C=optimal_C,kernel='linear')
# Let's train the model with the optimal C value and calculate the training time
tic = time()
linear_clf.fit(X_train, y_train)
toc = time()
time1 = str(round(1000*(toc-tic))) + "ms"
y_linear = linear_clf.predict(X_test)
linear_accuracy = accuracy_score(y_test, y_linear)*100
if max_iteration != 0:
    print("\n\n\n-----process ended\\
'-----\n\n\n")

# Let's show the results
showResults(linear_accuracy, time1, y_linear,'SVM linear kernel')
```



#2.Polynomial Kernel

#####Polynomial kernel #####

optimal_split_state2 = 0

maxAccuracy = 0

maxF1 = 0

We already tune parameters, we do not need to loop over all the hyperparamters again,

if you want to do so just set max_iteration to 500 for example

and remove the line 'optimal_split_state2 = 7070621' at the bottom of this cell.

max_iteration = 0

if max_iteration != 0:

print ('-----Hyperparameters tuning starts-----')

-----\n\n')

for k in range(max_iteration):

print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+' , Current f1 '+str(maxF1),
end="\r")

split_state = np.random.randint(1,100000000)-1

X_train,X_val,X_test,y_train,y_val,y_test

split(df,rest_size=0.4,test_size=0.4,randomState=split_state)

Let's get the optimal C and the degree value for the polynomial kernel

optimal_C, optimal_d = optimal_C_d_values()

Now let's use the optimal c value and the optimal degree value

poly_clf = SVC(C=optimal_C,kernel='poly', degree=optimal_d)

Let's train the model with the optimal C value

poly_clf.fit(X_train, y_train)

y_poly = poly_clf.predict(X_test)

poly_f1 = f1_score(y_test, y_poly, average='macro')

```

poly_accuracy = accuracy_score(y_test, y_poly)*100
if poly_accuracy>maxAccuracy and poly_f1>maxF1:
    maxAccuracy = poly_accuracy
    maxF1 = poly_f1
    optimal_split_state2 = split_state
# We've already tuned our hyperparameters, we will not repeat that again as it takes soo long.
# The optimal split state for polynomial kernel is 7070621
# Let's try that split state

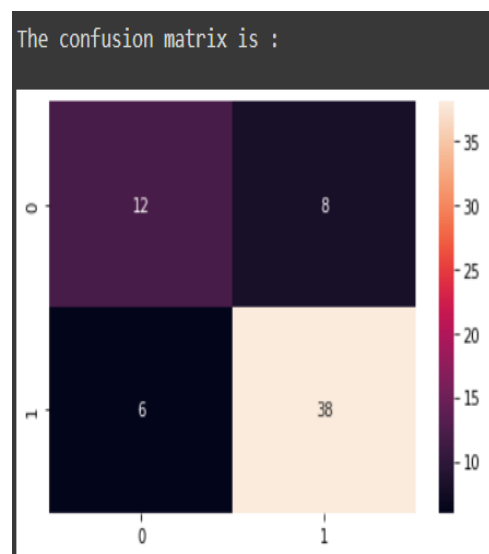
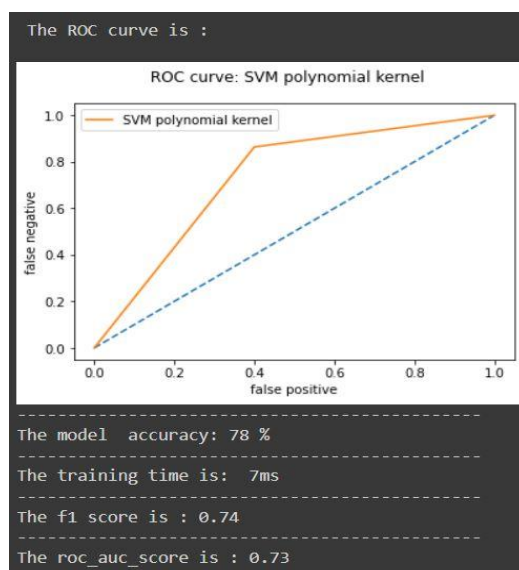
optimal_split_state2 = 7070621
X_train,X_val,X_test,y_train,y_val,y_test=
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state2)
optimal_C, optimal_d = optimal_C_d_values()
# Now let's use the optimal C value
poly_clf = SVC(C=optimal_C,kernel='poly', degree=optimal_d)
# Let's train the model and calculate the training time

tic = time()
poly_clf.fit(X_train, y_train)
toc = time()
time2 = str(round(1000*(toc-tic))) + "ms"
y_poly = poly_clf.predict(X_test)
poly_accuracy = accuracy_score(y_test, y_poly)*100
if max_iteration != 0:

    print("\n\n\n          -----process ended"\n\n\n')
    # Let's show the results
    showResults(poly_accuracy, time2, y_poly,'SVM polynomial kernel')

```

Output:



#3.RBF/Gaussian Kernel

```
#####Gaussiankernel #####
optimal_split_state3 = 0
maxAccuracy = 0
maxF1 = 0
# We already tune parameters, we do not need to loop over all the hyperparamters again,
# if you want to do so just set max_iteration to 500 for example
# and remove the line 'optimal_split_state3 = 93895097' at the bottom of this cell.
max_iteration = 0
if max_iteration != 0:
    print ('-----Hyperparameters tuning starts\'
          '-----\n\n')
for k in range(max_iteration):
    print ('Iteration :'+str(k)+' , Current accuracy: '+str(maxAccuracy)+' , Current f1 '+str(maxF1),
end="\r")
    split_state = np.random.randint(1,100000000)-1
    X_train,X_val,X_test,y_train,y_val,y_test =
split(df,rest_size=0.4,test_size=0.4,randomState=split_state)

    # Let's get the optimal C and the degree value for the polynomial kernel
    optimal_C, optimal_gamma = optimal_C_gamma_values()

    # Now let's use the optimal c value and the optimal degree value
    gauss_clf = SVC(C=optimal_C,kernel='rbf',gamma=optimal_gamma)

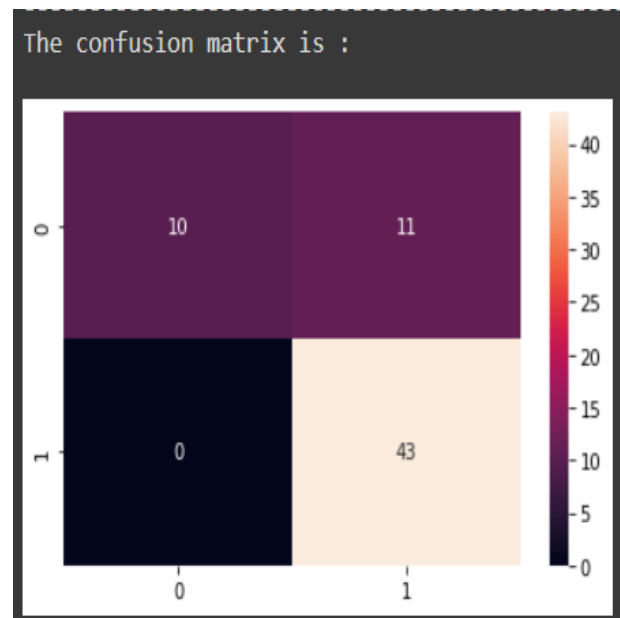
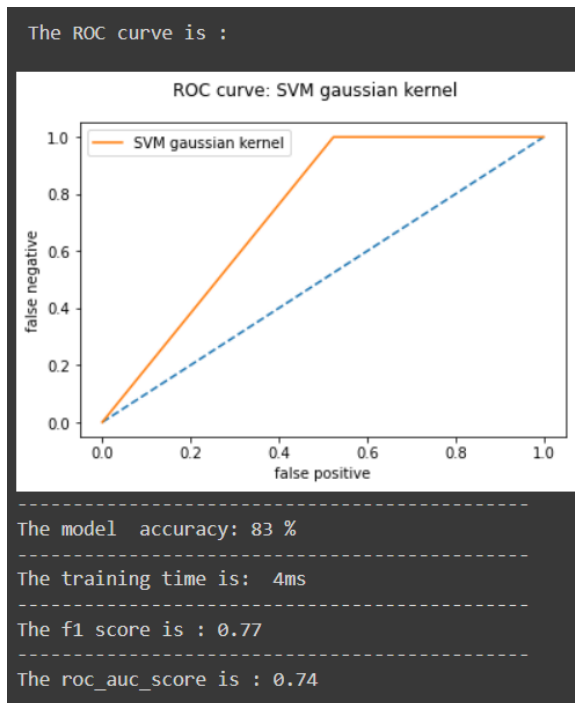
    # Let's train the model with the optimal C value
    gauss_clf.fit(X_train, y_train)
    y_gauss = gauss_clf.predict(X_test)
    gauss_f1 = f1_score(y_test, y_gauss, average='macro')
    gauss_accuracy = accuracy_score(y_test, y_gauss)*100

    if gauss_accuracy>maxAccuracy and gauss_f1>maxF1:
        maxAccuracy = gauss_accuracy
        maxF1 = gauss_f1
        optimal_split_state3 = split_state
# We've already tuned our hyperparameters, we will not repeat that again as it takes soo long.
# The optimal split state for polynomial kernel is 93895097
# Let's try that split state
optimal_split_state3 = 93895097

X_train,X_val,X_test,y_train,y_val,y_test =
split(df,rest_size=0.4,test_size=0.4,randomState=optimal_split_state3)
optimal_C, optimal_gamma = optimal_C_gamma_values()
# Now let's use the optimal C value
gauss_clf = SVC(C=optimal_C,kernel='rbf',gamma=optimal_gamma)
```

```
# Let's train the model and calculate the training time
tic = time()
gauss_clf.fit(X_train, y_train)
toc = time()
time3 = str(round(1000*(toc-tic))) + "ms"
y_gauss = gauss_clf.predict(X_test)
gauss_accuracy = (accuracy_score(y_test, y_gauss)*100)
if max_iteration != 0:
    print("\n\n\n-----process ended\
    '-----\
    \n\n\n')
# Let's show the results
showResults(gauss_accuracy, time3, y_gauss, 'SVM gaussian kernel')
```

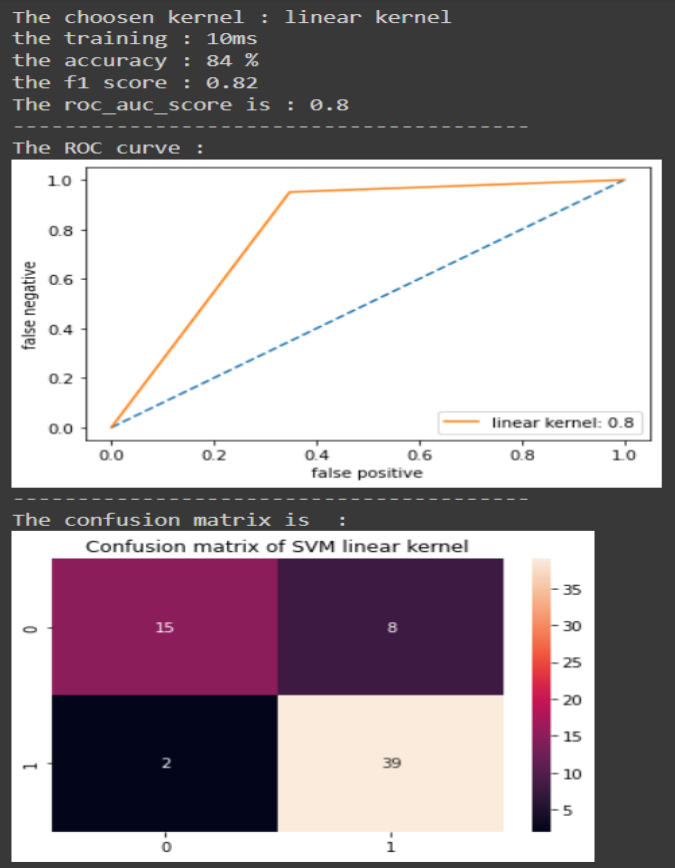
Output:



###Comparison of 3 kernels

```
compare_kernels()
```

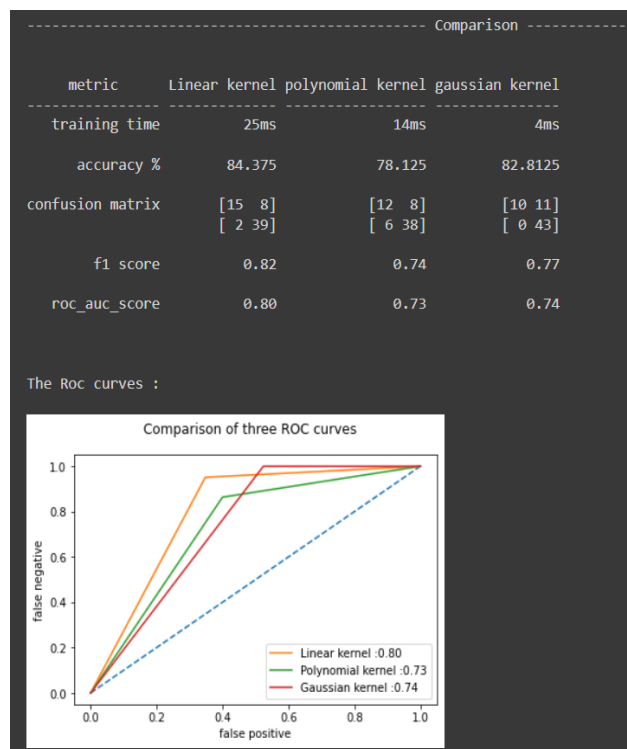
Output:



###The most accurate svm kernel is the linear kernel

yt_svm,yp_svm = best_kernel('linear kernel')

Output:



```
mse_svm = np.round(math.sqrt(mean_squared_error(yt_svm, yp_svm)),3)
print('Root mean square error', mse_svm)
```

Output:

Root mean square error 0.395

#Comparison of 3 classifiers

Function to compare the three classifiers (Logistic regression, KNN and SVM) performances:

```
def compare_lg_knn_svm(yt_knn,yp_knn,yt_lg,yp_lg,yt_svm,yp_svm):
    #F1 score
    f1_lg = round(f1_score(yt_lg, yp_lg, average='macro')*100)
    f1_knn = round(f1_score(yt_knn, yp_knn, average='macro')*100)
    f1_svm = round(f1_score(yt_svm, yp_svm, average='macro')*100)
    print(f1_lg)
    #Accuracy score
    acc_lg = round(accuracy_score(yt_lg, yp_lg)*100)
    acc_knn = round(accuracy_score(yt_knn, yp_knn)*100)
    acc_svm = round(accuracy_score(yt_svm, yp_svm)*100)
    #Confusion matrix
    conf_lg = confusion_matrix(yt_lg, yp_lg)
    conf_knn = confusion_matrix(yt_knn, yp_knn)
    conf_svm = confusion_matrix(yt_svm, yp_svm)

    #ROC score
    roc_c_lg = round(roc_auc_score(yt_lg, yp_lg)*100)
    roc_c_knn = round(roc_auc_score(yt_knn, yp_knn)*100)
    roc_c_svm = round(roc_auc_score(yt_svm, yp_svm)*100)

    #ROC curve thresholds
    roc_knn = roc_curve(yt_knn,yp_knn)
    roc_lg = roc_curve(yt_lg,yp_lg)
    roc_svm = roc_curve(yt_svm,yp_svm)

    # Table of metrics

    print('-----Table of metrics-----\n\n')
    data_rows = [('f1 score',f1_lg,f1_knn,f1_svm),
                  ("", "", ""),
                  ('accuracy %', acc_lg, acc_knn, acc_svm) ,
                  ("", "", ""),
                  ('confusion matrix',conf_lg[0], conf_knn[0], conf_svm[0]),
```

```

        ("conf_lg[1], conf_knn[1], conf_svm[1]),
        ("", "", ""),
        ('ROC score', roc_c_lg, roc_c_knn, roc_c_svm),
        ("", "", ""),
        ('Mean Square Error', mse_log, mse_knn, mse_svm),
        ("", "", "")]
t = Table(rows=data_rows, names=('metric', 'Logistic regression', 'KNN', 'SVM'))
print(t)
#Plot ROC curve

print("\n\n-----ROC curves-----\n\n")
fig, _ = plt.subplots()
fig.suptitle('Comparison of three ROC curves')
fpr, tpr, thresholds = roc_lg
plt.plot([0, 1], [0, 1], '--')
plt.plot(fpr, tpr, label='Logistic regression :'+str(roc_c_lg))
plt.xlabel('false positive')
plt.ylabel('false negative')
fpr, tpr, thresholds = roc_knn
plt.plot(fpr, tpr, label='KNN :'+str(roc_c_knn))
fpr, tpr, thresholds = roc_svm
plt.plot(fpr, tpr, label='SVM :'+str(roc_c_svm))
plt.legend()
plt.show()
print("\n\n-----Comparision-----\n\n")
N = 3
ind = np.arange(N)
width = 0.2
Accuracy = [acc_lg, acc_knn, acc_svm]
bar1 = plt.bar(ind, Accuracy, width, color = 'brown')
f1_scores = [f1_lg, f1_knn, f1_svm]
bar2 = plt.bar(ind+width, f1_scores, width, color='gold')
roc_scores = [roc_c_lg, roc_c_knn, roc_c_svm]
bar3 = plt.bar(ind+width*2, roc_scores, width, color = 'royalblue')
plt.ylim(0,100)
plt.xlabel("Classifiers")
plt.ylabel('Quality Metrics')
plt.title("Comparison of all Classifiers")
plt.xticks(ind+width, ['Logistic Regression', 'K-Nearest Neighbor', 'Support Vector Machine'])
plt.legend( (bar1, bar2, bar3), ('Accuracy', 'F1 Score', 'ROC Score') )
plt.show()
# Maximum metrics

print("\n\n-----Max of metrics-----\n\n")

```

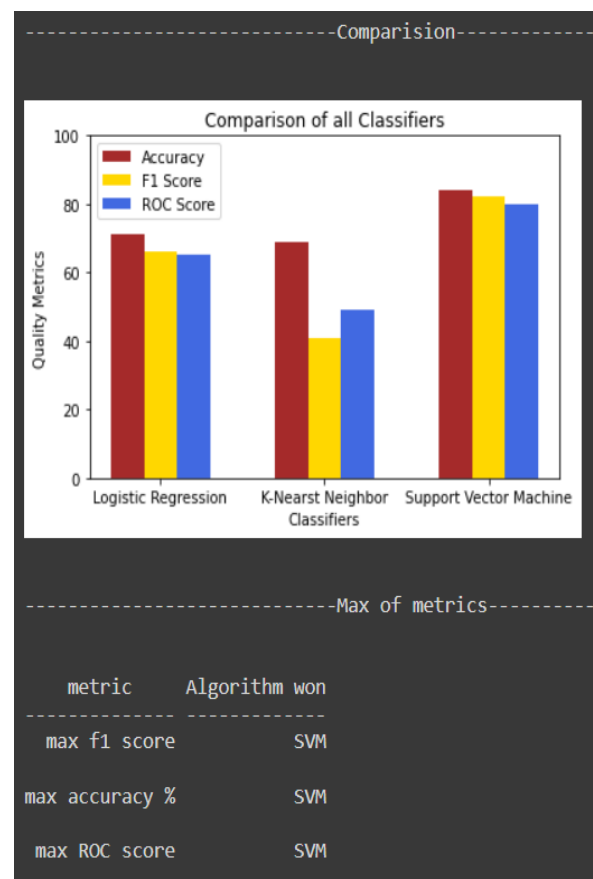
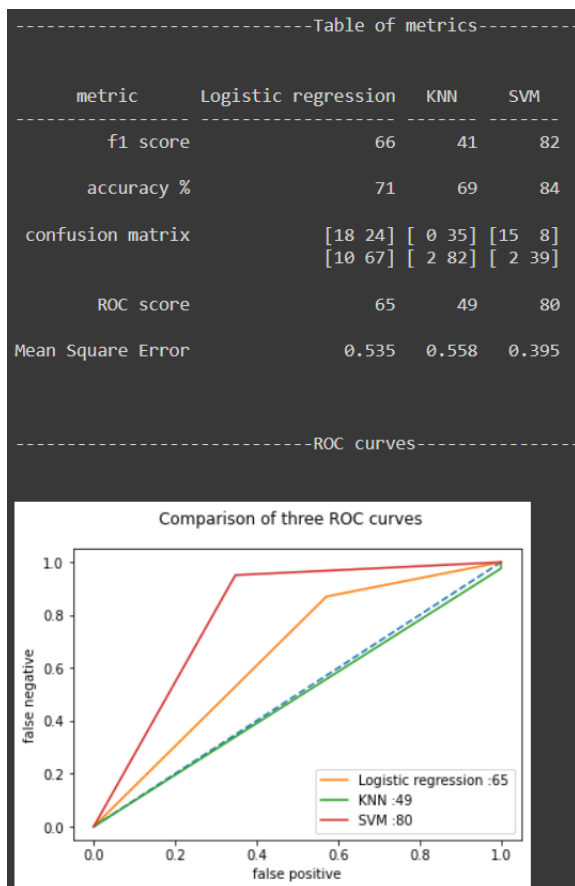
```

data_rows = [('max f1 score',algo_with_max_metric(f1_lg,f1_knn,f1_svm)),
             (",",",","),
             ('max accuracy %',algo_with_max_metric(acc_lg,acc_knn,acc_svm)),
             (",",",","),
             ('max ROC score',algo_with_max_metric(roc_c_lg,roc_c_knn,roc_c_svm))]
t = Table(rows=data_rows, names=('metric','Algorithm won'))
print(t)
# Function returning name of winning algorithm based on a single metric

def algo_with_max_metric(a,b,c):
    max_metric = max(a,b,c)
    if max_metric == a:
        return 'Logistic regression'
    elif max_metric == b:
        return 'KNN'
    else:
        return 'SVM'

compare_lg_knn_svm(yt_knn,yp_knn,yt_lg,yp_lg,yt_svm,yp_svm)
Output:

```



```
# Get svm parameters
coefs = linear_clf.coef_

# factors helping students to succeed
column_to_string(factors(coefs, 5, 'max', df), 'max')

# factors leading students to failure
column_to_string(factors(coefs, 5, 'min', df), 'min')
```

Output:

```
Factors helping students succeed :
father's education
guardian
wants to take higher education
studytime
father's job
-----
Factors leading students to failure
age
health
going out with friends
absences
failures
```

6.TESTING

6.1 INTRODUCTION

In a software development project, errors can be introduced at any stage during development. Though errors are detected after each phase by techniques like inspections, some errors remain undetected.

Ultimately, these remaining errors are reflected in the code. There are two types of approaches for identifying defects in the software: static and dynamic. In static analysis, the code is not executed but is evaluated through some process or some tools for locating defects. Code inspections, which we discussed in the previous chapter, are one static approach. Another is static analysis of code through the use of tools. In dynamic analysis, code is executed, and the execution is used for determining defects.

Testing is the most common dynamic technique that is employed. Indeed, testing is the most commonly used technique for detecting defects, and performs a very critical role for ensuring quality.

During testing, the software under test (SUT) is executed with a finite set of test cases, and the behavior of the system for these test cases is evaluated to determine if the system is performing as expected. The basic purpose of testing is to increase the confidence in the functioning of SUT.

Clearly, the effectiveness and efficiency of testing depends critically on the test cases selected.

Here we focus on:

Basic concepts and definitions relating to testing, like error, fault, failure, test case, test suite, test harness, etc.

- The testing process—how testing is planned and how testing of a unit is done.
- Test case selection using black-box testing approaches.
- Test case selection using white-box approaches.
- Some metrics like coverage and reliability that can be employed during testing

The basic goal of the software development process is to produce software that has no errors or very few errors. We have seen that different levels of testing are needed to detect the defects injected during the various tasks in the project. The testing process for a project consists of three high-level tasks test planning, test case design, and test execution. We will discuss these in the rest of this section.

TEST PLAN

In a project, testing commences with a test plan and terminates with successful execution of acceptance testing. A test plan is a general document for the entire project that defines the scope, approach to be taken, and the schedule of testing, as well as identifies the test items for testing and the personnel responsible for the different activities of testing.

The test planning can be done well before the actual testing commences and can be done in parallel with the coding and design activities.

The inputs for forming the test plan are:

- Project plan
- Requirements document
- Architecture or design document.

Test Case Design:

The test plan focuses on how the testing for the project will proceed, which units will be tested, and what approaches (and tools) are to be used during the various stages of testing. However, it does not deal with the details of testing a unit, nor does it specify which test cases are to be used.

TEST CASE EXECUTION:

The test case specifications only specify the set of test cases for the unit to be tested. However, executing the test cases may require construction of driver modules or stubs. It may also require modules to set up the environment as stated in the test plan and test case specifications.

If test frameworks are being used, then the setting of the environment as well as inputs for a test case is already done in the test scripts, and execution is straightforward.

During test case execution, defects are found

Testing Methods

The following are the Testing Methodologies:

- White box testing
- Black box testing
- Unit testing
- Integration testing
- Validation testing

WHITE BOX TESTING

“White box testing” (also known as clear, glass box or structural testing) is a testing technique which evaluates the code and the internal structure of a program.

White box testing involves looking at the structure of the code. When you know the internal structure of a product, tests can be conducted to ensure that the internal operations performed according to the specification. And all internal components have been adequately exercised. we perform white box testing for following reasons :

To ensure:

- That all independent paths within a module have been exercised at least once.
- All logical decisions verified on their true and false values.
- All loops executed at their boundaries and within their operational bounds internal data structures validity.

To discover the following types of bugs:

- Logical error tend to creep into our work when we design and implement functions, conditions or controls that are out of the program
- The design errors due to difference between logical flow of the program and the actual implementation.

Typographical errors and syntax checking.

Steps to Perform White Box Testing

Step 1 – Understand the functionality of an application through its source code. Which means that a tester must be well versed with the programming language and the other tools as well techniques used to develop the software.

Step 2– Create the tests and execute them.

MAIN WHITE BOX TESTING TECHNIQUES

- Statement Coverage
- Branch Coverage
- Path Coverage

Note that the statement, branch or path coverage does not identify any bug or defect that needs to be fixed. It only identifies those lines of code which are either never executed or remains untouched. Based on this further testing can be focused on.

BLACK BOX TESTING

Black Box Testing is also known as behavioural, opaque-box, closed-box, specification based or eye-to-eye testing. It is a Software Testing method that analyses the functionality of a software/application without knowing much about the internal structure/design of the item that is being tested and compares the input value with the output value.

The main focus in Black Box Testing is on the functionality of the system as a whole. The term 'Behavioural Testing' is also used for Black Box Testing. Behavioural test design is slightly different from the black-box test design because the use of internal knowledge isn't strictly forbidden, but it's still discouraged. Each testing method has its own advantages and disadvantages. There are some bugs that cannot be found using the only black box or only white box technique.

Majority of the applications are tested by Black Box method. We need to cover the majority of test cases so that most of the bugs will get discovered by a Black-Box method.

EQUIVALENCE PARTITIONING

This technique is also known as Equivalence Class Partitioning (ECP). In this technique, input values to the system or application are divided into different classes or groups based on its similarity in the outcome.

UNIT TESTING

In software engineering, unit testing is a level of software testing in which individual units/components are tested. A unit is a smallest testable part/module of any software application. In procedural programming, a unit is an individual program, function, procedure. In object-oriented programming, a unit may be a method.

Unit testing of the software product is done during the software development stage saves time and money in the end. If unit testing is skipped by the software developers then there is a higher chance of defects during the integration, system, acceptance and also beta testing.

The goal of unit testing is to isolate each part of source code and verifies that each part works properly as designed and to check that source code meets the requirements and gives the expected output.

The main advantage of unit testing is that it identify the problem earlier in the application and if any issue occurs then it can be fix before integrate the units.

In SDLC, STLC, V Model, unit testing is the first level of software testing done before integration testing.

Unit testing is a white box testing which is performed by the software developers and in rare cases, this testing may be done by independent software testers and quality assurance engineers also do unit testing.

INTEGRATION TESTING

Integration testing is the process of testing the interface between two software units or module. It's focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

UNIT TESTING

In software engineering, unit testing is a level of software testing in which individual units/components are tested. A unit is a smallest testable part/module of any software application. In procedural programming, a unit is an individual program, function, procedure. In object-oriented programming, a unit may be a method.

Unit testing of the software product is done during the software development stage saves time and money in the end. If unit testing is skipped by the software developers then there is a higher chance of defects during the integration, system, acceptance and also beta testing.

The goal of unit testing is to isolate each part of source code and verifies that each part works properly as designed and to check that source code meets the requirements and gives the expected output.

The main advantage of unit testing is that it identify the problem earlier in the application and if any issue occurs then it can be fix before integrate the units.

In SDLC, STLC, V Model, unit testing is the first level of software testing done before integration testing.

Unit testing is a white box testing which is performed by the software developers and in rare cases, this testing may be done by independent software testers and quality assurance engineers also do unit testing.

INTEGRATION TESTING

Integration testing is the process of testing the interface between two software units or module. It's focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

Integration test approaches –

There are four types of integration testing approaches. Those approaches are the following:

BIG-BANG INTEGRATION TESTING:

It is the simplest integration testing approach, where all the modules are combining and verifying the functionality after the completion of individual module testing. In simple words, all the modules of the system are simply put together and tested.

This approach is practicable only for very small systems. If once an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during big bang integration testing are very expensive to fix.

BOTTOM-UP INTEGRATION TESTING:

In bottom-up testing, each module at lower levels is tested with higher modules until all modules are tested. The primary purpose of this integration testing is, each subsystem is to test the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower level modules.

TOP-DOWN INTEGRATION TESTING:

Top-down integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

MIXED INTEGRATION TESTING:

A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches.

A mixed integration testing is also called sandwiched integration testing.

VALIDATION TESTING:

Validation in Software Testing is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not.

The process helps to ensure that the software fulfils the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing

TEST CASE:

A TEST CASE is a set of actions executed to verify a particular feature or functionality of your software application. A test case contains test steps, test data, precondition, post condition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

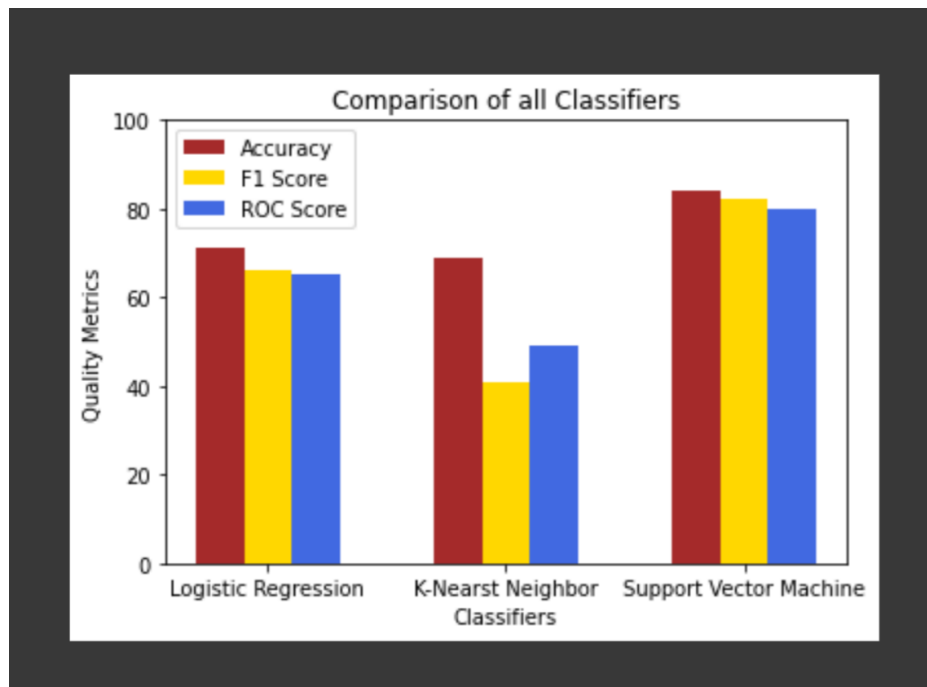
2.2 TEST RESULTS:

Test Case Id	Input	Actual Output	Predicted Output
1	(GP,F,16,U,GT3,T,4,4,services,services,reputation,mother,1,3,0,no,yes,yes,yes,yes,yes,yes,no,3,2,3,1,2,2,6)	Pass	Pass
2	(GP,F,15,U,LE3,A,4,3,other,other,course,mother,1,2,0,yes,yes,yes,yes,yes,yes,yes,yes,5,2,2,1,1,5,8)	Fail	Fail

7.CONCLUSION

Figuring out which classification algorithm works best and determining which factors have the greatest impact on students' academic standing so they can get a clear picture of what it takes to succeed in school and stay away from trouble. In conclusion, the results of the comparison between the SVM, logistic regression, and KNN models for forecasting students' academic performance showed that the SVM model outperformed the other two models in terms of multiple quality metrics, including F1 score, accuracy, and ROC. The SVM model had the highest F1 score, accuracy, and ROC AUC value (**Fig 9**), indicating that it was the most effective model in correctly classifying students as pass or fail. The results of this comparison highlight the importance of considering multiple quality metrics when evaluating model's performance. The SVM model's superior performance in all four metrics suggests that it is the best choice for this particular task of forecasting students' academic performance.

In summary, the use of SVM models in this area has the potential to improve the accuracy of predictions and provide valuable insights into the aspects that effects students' exam score. By making the use of machine learning techniques, it is possible to make informed decisions and take targeted actions to improve students' outcomes



8.BIBLIOGRAPHY

1. "Comparative study of supervised learning algorithms for student performance prediction," by W. Tomohisa, M. Dawodi, N. Ahmadi and M. Mohammadi, 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), 2019, pp. 124- 127, Doi:10.1109/ICAIIIC.2019.8669085.
2. Ayan, M., & García, M. (2008) "Prediction of University Students' Academic Achievement by Linear and Logistic Models. The Spanish Journal of Psychology, 11(1), 275- 288." doi:10.1017/S1138741600004315.
3. "Early Student Grade Prediction: An Empirical Study, 2019 2nd International Conference on Advancements in Computational Sciences (ICACS)" by S. Latif, Z. Iqbal, J. Qadir and A. Qayyum 2019, pp. 1-7, doi:10.23919/ICACS.2019.8689136.
4. .N. M. Rusli, Z. Ibrahim and R. M. Janor, "Predicting students' academic achievement: Comparison between logistic regression, artificial neural network, and Neuro-fuzzy," 2008 International Symposium on Information Technology, 2008, pp. 1- 6, doi:10.1109/ITSIM.2008.4631535.
5. "A Systematic Literature Review of Students' Performance Prediction Using Machine Learning Techniques." Education Sciences. 2021; 11(9):552 by Zaki N, Albreiki B, and Alashwal H. <https://doi.org/10.3390/educsci11090552>
6. Leena H. Alamri, Ranim S. AL Muslim, Mona S. Alotaibi, Dana K. Alkadi, Irfan Ullah Khan, and Nida Aslam. 2021. Predicting Student Academic Performance using Support Vector Machine and Random Forest. In 2020 3rd International Conference on Education Technology Management (ICETM 2020). Association for Computing Machinery, New York, NY, USA, 100–107. <https://doi.org/10.1145/3446590.3446607>
7. Hamsa, Hashmi, Simi Indurative, and Jubilant J. Kizhakkethottam. "Student academic performance prediction model using decision tree and fuzzy genetic algorithm." Procedia Technology 25 (2016): 326-332.
8. Vijayalakshmi, V., and K. Venkatachalapathy. "Comparison of predicting student's performance using machine learning algorithms." International Journal of Intelligent Systems and Applications 11.12 (2019): 34.
9. Borkar, Suchita, and K. Rajeswari. "Predicting students' academic performance using education data mining." International journal of computer science and mobile computing 2.7 (2013): 273-279.

10. Suthaharan, Shan, and Shan Suthaharan. "Support vector machine." *Machine learning models and algorithms for big data classification: thinking with examples for effective learning* (2016): 207-235.
11. Elbadrawy, Asmaa, et al. "Predicting student performance using personalized analytics." *Computer* 49.4 (2016): 61-69.
12. Yildiz, Osman, et al. "Rules Optimization Based Fuzzy Model for Predicting Distance Education Students' Grades." *International Journal of Information and Education Technology* 4.1 (2014): 59-62.
13. Xu, Jie, Kyeong Ho Moon, and Mihaela Van Der Schaar. "A machine learning approach for tracking and predicting student performance in degree programs." *IEEE Journal of Selected Topics in Signal Processing* 11.5 (2017): 742-753.
14. S. Wiyono and T. Abidin, "COMPARATIVE STUDY OF MACHINE LEARNING KNN, SVM, AND DECISION TREE ALGORITHM TO PREDICT STUDENT'S PERFORMANCE", *Int. J. Res. Granthaalayah*, vol. 7, no. 1, pp. 190–196, Jan. 2019.
15. Tarek Abd El-Hafeez, Ahmed Omar. *Student Performance Prediction Using Machine Learning Techniques*, 23 March 2022, PREPRINT (Version 1) available at Research Square [<https://doi.org/10.21203/rs.3.rs-1455610/v1>]