LAB MANUAL OF

**MACHINE LEARNING LAB (CIE-421P)**



**Department of Computer Science and Engineering**

Maharaja Agrasen Institute of Technology, PSP area,

Sector-22, Rohini, New Delhi -110086

## (Guru Gobind Singh Indraprastha University, New Delhi)

**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**



#### VISION OF THE INSTITUTE

To attain global excellence through education, innovation, research, and work ethics with the commitment to serve humanity.

**MISSION OF THE INSTITUTE**

|  |  |
| --- | --- |
| **M1.** | To promote diversification by adopting advancement in science, technology, management, and allied discipline through continuous learning |
| **M2.** | To foster moral values in students and equip them for developing sustainable solutions to serve both national and global needs in society and industry. |
| **M3.** | To digitize educational resources and process for enhanced teaching and effective learning. |
| **M4.** | To cultivate an environment supporting incubation, product development, technology transfer, capacity building and entrepreneurship. |
| **M5.** | To encourage faculty-student networking with alumni, industry, institutions, and other stakeholders for collective engagement. |

## MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY



#### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**VISION OF THE DEPARTMENT**

To attain global excellence through education, innovation, research, and work ethics in the field of Computer Science and engineering with the commitment to serve humanity.

**MISSION OF THE DEPARTMENT**

|  |  |
| --- | --- |
| **M1.** | To lead in the advancement of computer science and engineering through internationally recognized research and education. |
| **M2.** | To prepare students for full and ethical participation in a diverse society and encourage lifelong learning. |
| **M3.** | To foster development of problem solving and communication skills as an integral component of the profession. |
| **M4.**  **M5.** | To impart knowledge, skills and cultivate an environment supporting incubation, product development, technology transfer, capacity building and entrepreneurship in the field of computer science and engineering.  To encourage faculty, student’s networking with alumni, industry, institutions, and other stakeholders for collective engagement. |

## MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY



**PROGRAM OUTCOMES (POs)**

|  |  |
| --- | --- |
| **PO1** | **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering  problems. |
| **PO2** | **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of  mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental  considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex  engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and  need for sustainable development. |
| **PO8** | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO9** | **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give  and receive clear instructions. |
| **PO11** | **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| **PO12** | **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

## MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY



#### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

|  |  |
| --- | --- |
| **PEO1** | Graduates will work with the top institutions and researchers, dedicating themselves to lifelong learning and social responsibility. (M1, M2) |
| **PEO2** | Graduates will exhibit outstanding communication skills and the capacity to collaborate effectively within diverse teams. (M3) |
| **PEO3** | Graduates cultivating skills in computer science and engineering contribute to driving innovation, entrepreneurship, and economic growth. (M4) |
| **PEO4** | Graduates network with stakeholders to contribute to the growth of the department. (M5) |

## MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY



#### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

|  |  |
| --- | --- |
| **PSO1** | Able to explore and apply emerging technologies in computer science and engineering such as Artificial Intelligence, Machine Learning, Data Science, etc. |
| **PSO2** | Able to independently and collaboratively design, develop and evaluate innovative solutions to existing problems, addressing the needs of industry and society. |
| **PSO3** | Able to pursue advanced studies, conduct research and development, and cultivate entrepreneurship skills in the modern computing environment. |

**CO OF THE LAB SUBJECT AND ITS MAPPING WITH PO**

|  |  |
| --- | --- |
| **Course Outcomes** | |
| **CO** | **Statement** |
| CO1 | To Formulate Machine Learning Problems |
| CO2 | To learn about regression and feature selection |
| CO3 | To understand about classification algorithms |
| CO4 | To learn clustering algorithms |

**Course Outcomes (CO) to Programme Outcomes (PO) and Program Specific Outcomes (PSO) mapping (Scale 1: low, 2: Medium, 3: High)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO-PO & CO-PSO Mapping** | | | | | | | | | | | | | | | |
| **CO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| CO1 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | - | - | - | - | 2 | 3 | 3 | 3 |
| CO2 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | - | - | - | - | 2 | 3 | 3 | 3 |
| CO3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | - | - | - | - | 2 | 3 | 3 | 3 |
| CO4 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | - | - | - | - | 2 | 3 | 3 | 3 |
| Average | 3 | 3 | 3 | 3 | 3 | 2 | 2 | - | - | - | - | 2 | 3 | 3 | 3 |

**RUBRICS EVALUATION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rubrics/ Marks** | | **0** | **1** | **2** | **3** |
| **Missing** | **Inadequate** | **Needs Improvement** | **Adequate** |
| **R1** | **Is able to identify the problem to be solved and define the objective of the experiment.** | No mention is made of the problem to be solved. | Problem is vaguely identified and objectives are not relevant or erroneous. | There are minor omissions or vague details. Objectives are conceptually correct and measurable but  incomplete. | The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. |
| **R2** | **Is able to design a reliable experiment that solves the problem.** | The experiment does not solve the problem.  Solution is not implemented at all. | The experiment is designed to solve the problem but contains missing links. Solution is attempted but implementation is not correct. | The experiment attempts to solve the problem but needs improvement to correctly arrive at it. Solution is partially  implemented. | The experiment solves the problem and has generic logic applicable to generic data.  Solution is fully implemented. |
| **R3** | **Is able to communicate the details of an experimental procedure clearly and completely.** | Diagrams are missing and/or experimental procedure is missing or extremely vague. | Diagrams are present but do not properly describe the procedure. | Diagrams and/or experimental procedures are present but with errors that may result in invalid  conclusion. | Diagrams and/or experimental procedures are clear and complete. |
| **R4** | **Is able to record and represent data in a meaningful way.** | Data are either absent or incomprehensible. | Some important data are absent or incomprehensible. | All-important data are present, but recorded in an incomprehensible  way. | All important data are present, organized and recorded clearly. |
| **R5** | **Is able to make a judgment about the results of the experiment.** | No discussion is presented about the results of the experiment. | A judgment is made about the results, but is not reasonable or coherent. | An acceptable judgment is made about the result, but the reasoning is flawed or  incomplete. | An acceptable judgment is made about the result, with valid conclusions. |

**CONTENTS**

|  |  |
| --- | --- |
|  | **Page No.** |
| Introduction of Lab Subject | 1 |
| Hardware & Software Requirement | 3 |
| List of Practicals (As Per syllabus prescribed by G.G.S.I.P.U) | 4 |
| List of Practical (Beyond the syllabus Prescribed by G.G.S.I.P.U) | 5 |
| Format of the lab record to be prepared by the students | 6 |
| Marking scheme for the practical exam | 9 |
| For each Practical  (Steps to be followed & Viva Question). | 11-48 |

# INTRODUCTION TO MACHINE LEARNING LAB

##### Lab Overview:

Machine Learning is used anywhere from automating mundane tasks to offering intelligent insights, industries in every sector try to benefit from it. You may already be using a device that utilizes it. For example, a wearable fitness tracker like Fitbit, or an intelligent home assistant like Google Home. But there are much more examples of ML in use.

* **Prediction:** Machine learning can also be used in the prediction systems. Considering the loan example, to compute the probability of a fault, the system will need to classify the available data ingroups.
* **Image recognition**: Machine learning can be used for face detection in an image as well. There is aseparate category for each person in a database of several people.
* **Speech Recognition:** It is the translation of spoken words into the text. It is used in voice searches and more. Voice user interfaces include voice dialing, call routing, and appliance control. It can also beused a simple data entryand the preparation of structured documents.
* **Medical diagnoses:** ML is trained to recognize cancerous tissues.
* **Financial industry:** Trading companies use ML in fraud investigations and credit checks.

##### Why Python for Machine Learning?

Features of Python that makes it the preferred choice of language for data science:

##### Extensive set of packages

Python has an extensive and powerful set of packages which are ready to be used in various domains. It also has packages like numpy, scipy, pandas, scikit-learn etc. which are required for machine learning and data science.

##### Easy prototyping

Another important feature of Python that makes it the choice of language for data science is the easy and fast prototyping. This feature is useful for developing new algorithm.

##### Collaboration feature

The field of data science basically needs good collaboration and Python provides many useful tools that make this extremely.

##### One language for many domains

A typical data science project includes various domains like data extraction, data manipulation, data analysis, feature extraction, modeling, evaluation, deployment and updating the solution. As Python is a multi-purpose language, it allows the data scientist to address all these domains from a common platform.

**Jupyter Notebook −** Jupyter notebooks basically provides an interactive computational environment for developing Python based Data Science applications

##### Installing Python

For working in Python, we must first have to install it. You can perform the installation of Python in any of the following two ways:

##### Installing Python individually

Using Pre-packaged Python distribution: Anaconda.

If you want to install Python on your computer, then then you need to download only the binary code applicable for your platform. Python distribution is available for Windows, Linux and Mac platforms.

##### On Unix and Linux platform.

With the help of following steps, we can install Python on Unix and Linux platform – First, go to [www.python.org/downloads/.](https://www.python.org/downloads/)

Next, click on the link to download zipped source code available for Unix/Linux. Now, Download and extract files.

Next, we can edit the Modules/Setup file if we want to customize some options.

Next, write the command run. /configure scriptmake make install

**HARDWARE & SOFTWARE REQUIREMENT**

**CPU** : Intel core i5 8GB RAM

1 TB HDD

Keyboard Mouse TFT

**PRINTER** : Laser Jet Printer

**SOFTWARE** : Jupyter Notebook, Python

**LIBRARIES USED**

**IN PYTHON :** Pandas, Numpy, Matplotlib, Seaborn,

**LIST OF PRACTICALS (AS PER SYLLABUS PRESCRIBED BY G.G.S.I.P.U)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Paper Code: CIE -421P** | **L** | **P** | **C** |
| **Paper: - Machine Learning Lab** | **0** | **2** | **1** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Discipline(s) / EAE / OAE** | **Semester** | **Group** | **Sub‐group** | **Paper Code** |
| ECE | 6 | PCE | PCE‐3 | ECE‐350P |
|  |  |  |  |  |
| EAE | 6 | MLDA‐EAE | MLDA‐EAE‐2C | ML‐342P |
| CSE/IT/CST/ITE | 7 | PCE | PCE‐5 | CIE‐421P |
| CSE‐AIML | 7 | PC | PC | ML‐407P |
| EAE | 7 | AIML‐EAE | AIML‐EAE‐3 | ML‐407P |

**Instructions:**

1. The course objectives and course outcomes are identical to that of (Machine Learning) as this is the practical component of the corresponding theory paper.
2. The practical list shall be notified by the teacher in the first week of the class commencement under intimation to the office of the Head of Department / Institution in which the paper is being offered from the list of practicals below. Atleast 10 experiments must be performed by the students, they may be asked to do more. Atleast 5 experiments must be from the given list.

**Marking Scheme:**

1. Teachers Continuous Evaluation: 40 marks
2. Term end Theory Examinations: 60 marks
3. Introduction to JUPYTER IDE and its libraries Pandas and NumPy
4. Program to demonstrate Simple Linear Regression
5. Program to demonstrate Logistic Regression
6. Program to demonstrate Decision Tree – ID3 Algorithm
7. Program to demonstrate k‐Nearest Neighbor flowers classification
8. Program to demonstrate Naïve‐ Bayes Classifier
9. Program to demonstrate PCA and LDA on Iris dataset
10. Program to demonstrate DBSCAN clustering algorithm
11. Program to demonstrate K‐Medoid clustering algorithm
12. Program to demonstrate K‐Means Clustering Algorithm on Handwritten Dataset

**LIST OF PRACTICALS (BEYOND THE LIST PRESCRIBED BY G.G.S.I.P.U)**

1. Program to visualize data classification using SVM with different kernels.
2. Study and Evaluate ML algorithm with balanced and unbalanced datasets.

**FORMAT OF THE LAB RECORDS TO BE PREPARED BY THE STUDENTS**

The front page of the lab record prepared by the students should have a cover page as displayed below.

***MACHINE LEARNING LAB***

***CIE-421P***

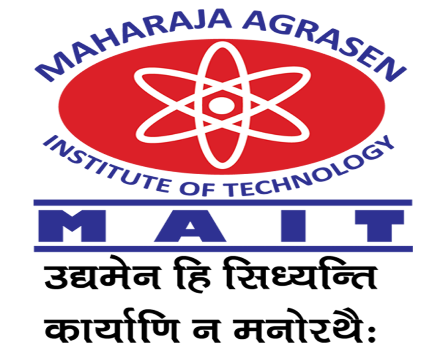
The font should be(Size 20”, italics bold, Times New Roman)

Faculty name Student name

Roll No.:

Semester:

Font should be (12”, Times Roman)



Department of Computer Science and Engineering

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

The font should be (18”, Times Roman)

The format of the Index to be prepared by the students is displayed as below:

Student’s Name:

Roll No.:

1. Index for the Lab File is as follows:

#### MACHINE LEARNING LAB

**PRACTICAL RECORD**

**PAPER CODE : CIE-421P**

Name of the student :

University Roll No. :

Branch :

Section/ Group :

**PRACTICAL DETAILS**

1. Experiments according to the list provided by GGSIPU

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Expt. No.** | **Date** | **Experiment Name** | **Marks (0-3)** | | | | | **Total Marks (15)** | **Signature** |
|  |  |  | **R1** | **R2** | **R3** | **R4** | **R5** |  |  |
| 1. |  |  |  |  |  |  |  |  |  |
| 2. |  |  |  |  |  |  |  |  |  |
| 3. |  |  |  |  |  |  |  |  |  |
| 4. |  |  |  |  |  |  |  |  |  |
| 5. |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |

1. Experiments Beyond the GGSIPU Syllabus:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Expt. No.** | **Date** | **Experiment Name** | **Marks (0-3)** | | | | | **Total**  **Marks (15)** | **Signature** |
|  |  |  | **R1** | **R2** | **R3** | **R4** | **R5** |  |  |
| 1. |  |  |  |  |  |  |  |  |  |
| 2. |  |  |  |  |  |  |  |  |  |

**All the lab records should have the following:**

1. Date
2. Aim
3. Algorithm or The Procedure to be followed
4. Program
5. Output
6. Viva Questions

**MARKING SCHEME FOR THE**

**PRACTICAL EXAMINATION**

There will be two practical exams in each semester.

* 1. Internal Practical Exam
  2. External Practical Exam

**INTERNAL PRACTICAL EXAMINATION**

It is taken by the concerned faculty of the batch.

# MARKING SCHEME FOR INTERNAL EXAM IS:

Total Marks: 40

Division of 40 Marks is as follows:

1. Regularity/ Attendance 05
2. File 15
3. Quiz 10
4. Viva Voce 10

# EXTERNAL PRACTICAL EXAMINATION

It is taken by the concerned faculty of the batch and by an external examiner. In this exam, student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by external examiner.

# MARKING SCHEME FOR THIS EXAM IS:

Total Marks : 60

Division of 60 marks is as follows

|  |  |
| --- | --- |
| 1. Sheet filled by the student : | 15 |
| 2. Viva Voce : | 20 |
| 3. Experiment performance : | 15 |
| 4. File submitted : | 10 |

# NOTE:

* Internal marks + External marks = Total marks given to the students
* (40 marks) (60 marks) (100 marks)

# EXPERIMENT 1

**AIM:** Introduction to JUPYTER IDE and its libraries Pandas and NumPy

## INTRODUCTION:

Jupyter Notebooks have become an integral tool in the fields of data science, machine learning, and scientific research. It is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is part of Project Jupyter, which is a non-profit, open-source project that evolved from the IPython Project in 2014. The name 'Jupyter' is a reference to the core supported programming languages that it was designed to support: Julia, Python, and R, but today it supports over 40 programming languages.

##### Features of Jupyter Notebook

1. Interactive Development Environment
2. Rich Text and Media Support
3. Collaboration and Sharing
4. Extensibility and Customization
5. Reproducibility and Portability
6. Data Science and Visualization
7. Open Source and Community-Driven

##### Steps to follow:

* 1. Launch Jupyter Notebook on your computer.
  2. Create a new notebook by clicking on the "New" button and selecting "Python 3" from the dropdown menu.
  3. Familiarize yourself with the interface: cells, toolbar, menu options, etc.
  4. Rename your notebook to "Intro\_to\_Jupyter.ipynb".

# No code required for this task. Simply launch Jupyter Notebook and create a new notebook

* 1. Understand the two main types of cells: Code and Markdown.
  2. Create a new Markdown cell and write a brief introduction to Jupyter Notebooks.

2.3 Create a new Code cell and write a simple Python expression (e.g., 2 + 2).

* 1. Execute the Code cell by pressing Shift + Enter and observe the output.
  2. Experiment with adding, deleting, and moving cells within the notebook.

# Creating a Markdown cell

# Just type in a Markdown cell: "# Introduction to Jupyter Notebooks" # This will create a heading indicating an introduction to Jupyter Notebooks.

# Creating a Code cell

# This will output 4 when the cell is executed.

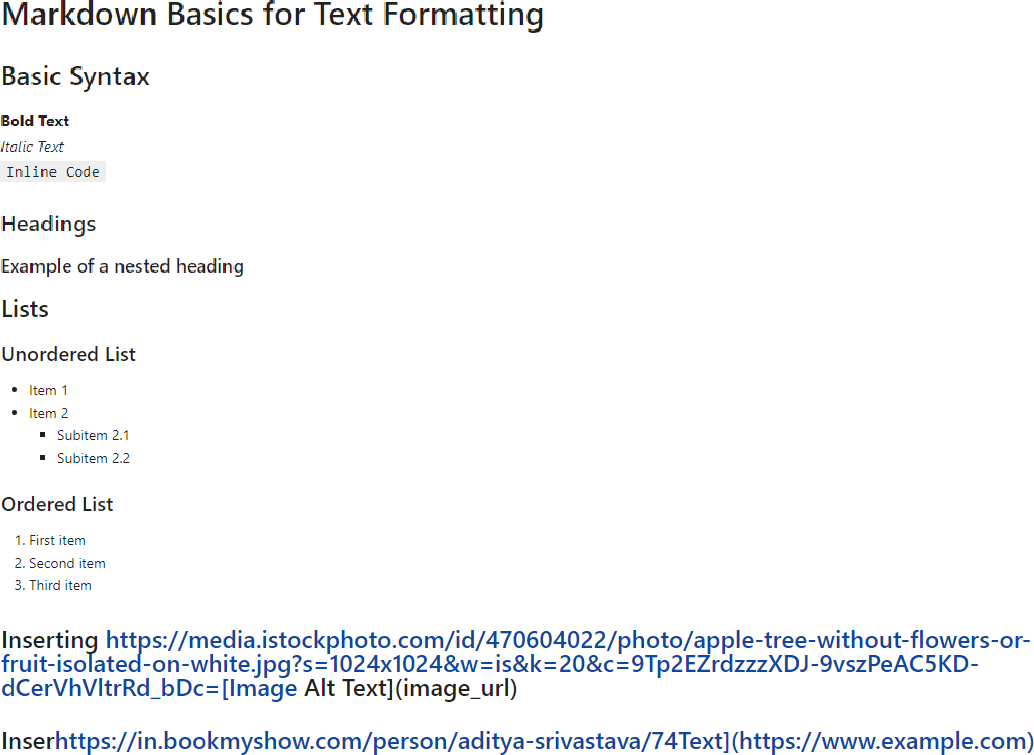
* 1. Learn basic Markdown syntax for text formatting (e.g., headings, bold, italic).
  2. Practice creating Markdown cells with formatted text and headings.
  3. Insert images and hyperlinks using Markdown syntax.
  4. Create a Markdown cell with a list of your favourite programming languages.
  5. Write a Python function that calculates the factorial of a given number.
  6. Test your function with different input values and observe the results.
  7. Import a Python library (e.g., NumPy) and use it to perform a mathematical operation (e.g., calculate mean, median).
  8. Visualize data using Matplotlib or Seaborn libraries within a Jupyter Notebook
  9. Understand what a kernel is in the context of Jupyter Notebooks.
  10. Learn how to switch kernels in a Jupyter Notebook.
  11. If available, switch your notebook's kernel to a different programming language (e.g., R, Julia) and run a simple command.
  12. Reset the kernel and clear the outputs. Observe how this affects your notebook*.*
  13. Learn about nbviewer, an online tool to view Jupyter Notebooks.
  14. Upload your notebook to a public GitHub repository.
  15. Use nbviewer to view your notebook online.

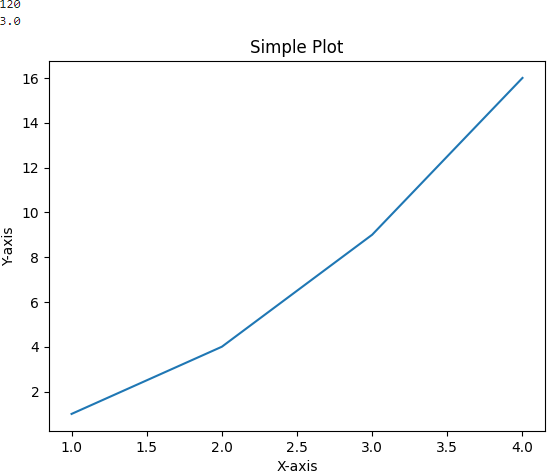
Install any chosen extension and demonstrate its functionality: # One popular extension is jupyter\_contrib\_nbextensions,

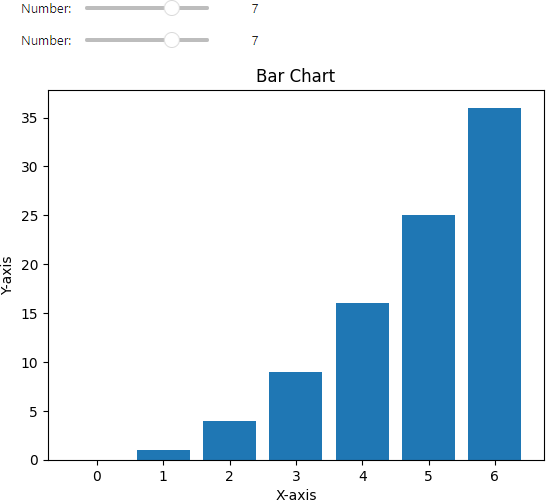
# extension as an example.

# Install jupyter\_contrib\_nbextensions #!pip install jupyter\_contrib\_nbextensions

**Sample output**







**ADDITIONAL PRACTICE PROBLEM:**

* 1. Explore the Google Collab and its libraries.

## VIVA QUESTIONS

* + 1. What techniques do you use to optimize the performance of a Jupyter Notebook?
    2. What is the difference between a Jupyter Notebook and a Python script?
    3. How do you ensure that a Jupyter Notebook is compatible with different versions of Python?
    4. Why do people prefer jupyter notebook over IDE like VS Code?
    5. How would you design a Jupyter Notebook to process large datasets?
    6. How do you integrate a Jupyter Notebook with other applications?

**EXPERIMENT 2**

**AIM:** Program to demonstrate Simple Linear Regression

## INTRODUCTION:

Linear regression predicts the relationship between two variables by assuming a linear connection between the independent and dependent variables. It seeks the optimal line that minimizes the sum of squared differences between predicted and actual values. In a simple linear regression, there is one independent variable and one dependent variable. The model estimates the slope and intercept of the line of best fit, which represents the relationship between the variables. The slope represents the change in the dependent variable for each unit change in the independent variable, while the intercept represents the predicted value of the dependent variable when the independent variable is zero.

To calculate best-fit line linear regression uses a traditional slope-intercept form which is given below, Yi = β0 + β1Xi

##### Steps to follow:

##### Import the necessary libraries for linear regression including

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

1. Generate sample data

X = 2 \* np.random.rand(100, 1)

y = 4 + 3 \* X + np.random.randn(100, 1)

1. Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. Train the linear regression model

model = LinearRegression() model.fit(X\_train, y\_train)

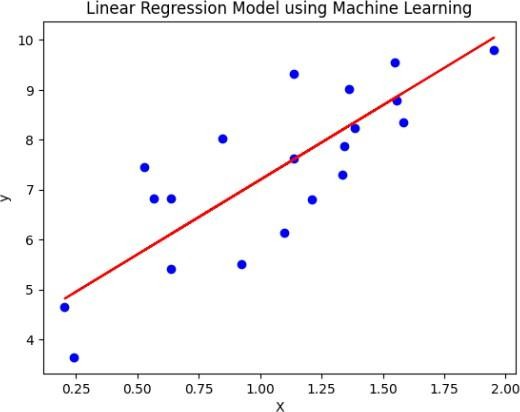
1. Make predictions on the test set

y\_pred = model.predict(X\_test)

1. Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

1. Visualize the model using scatter plot plt.scatter(X\_test, y\_test, color='blue') plt.plot(X\_test, y\_pred, color='red')

**Sample Output**:

## 

**ADDITIONAL PRACTICE PROBLEM:**

## Program for linear regression model using scikit-learn and use the following dataset : <https://www.kaggle.com/datasets/bumba5341/advertisingcsv>

## VIVA QUESTIONS

1. What are outliers? How do you detect and treat them?
2. How do you determine the best fit line for a linear regression model?
3. What are the common types of errors in linear regression analysis?
4. What is the difference between a dependent and independent variable in linear regression?
5. What is the difference between biased and unbiased estimates in linear regression?
6. How do you measure the strength of a linear relationship between two variables?
7. What are the assumptions of the ordinary least squares method for linear regression?

# EXPERIMENT 3

## AIM:

Program to demonstrate Logistic Regression.

## INTRODUCTION:

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation.

The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false. Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used inpredictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.

Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1.

Moreover, if the output of the sigmoid function (estimated probability) is greater than a predefined threshold on the graph, the model predicts that the instance belongs to that class. If the estimated probability is less than the predefined threshold, the model predicts that the instance does not belong to the class.

**Link of data Set to be used:** [**https://www.kaggle.com/datasets/uciml/iris**](https://www.kaggle.com/datasets/uciml/iris)

##### Steps to follow:

Apply simple logistic regression to iris dataset and submit the analysis in the form of a jupyter notebook.

1. Import necessary libraries including

from sklearn.linear\_model import LogisticRegression

1. Load the Iris dataset

# Display the first few rows of the dataset

# Data Exploration sns.pairplot(iris, hue='species')

# Data Preprocessing

# Encode categorical variable 'species' using label encoding

iris['species'] = iris['species'].map({'setosa': 0, 'versicolor': 1, 'virginica': 2})

1. Split the data into features (X) and target variable (y) X = iris.drop('species', axis=1)

y = iris['species']

1. Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. Model Building i.e. create a logistic regression model

model = LogisticRegression()

1. Fit the model to the training data model.fit(X\_train, y\_train)
2. Model Evaluation i.e. predict the target variable for the test set

y\_pred = model.predict(X\_test)

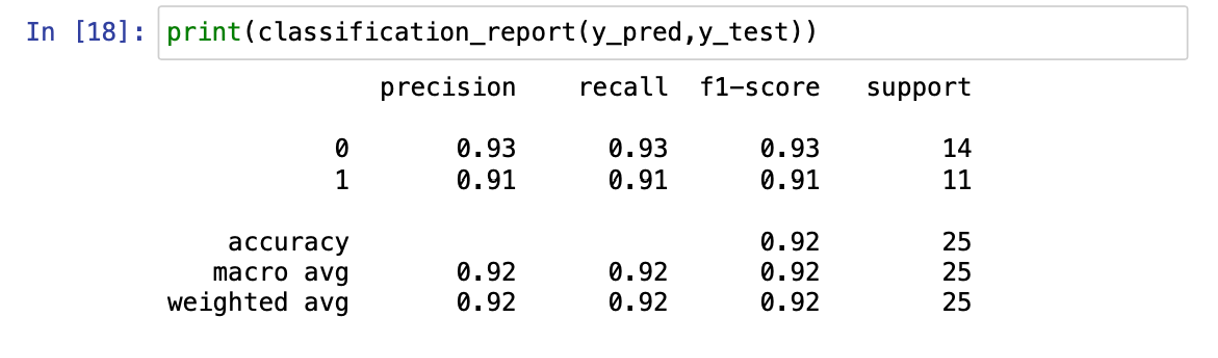
1. Calculate the accuracy of the model

accuracy = accuracy\_score(y\_test, y\_pred)

1. Create a confusion matrix and display the accuracy

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

**Sample Output**:



**ADDITIONAL PRACTICE PROBLEM:**

1. Build a Logistic Regression model specific to college students. Link for the dataset: <https://www.kaggle.com/datasets/kellygakii/student-data-csv>

## VIVA QUESTIONS

1. How do we handle categorical variables in Logistic Regression?
2. Why is Logistic Regression termed as Regression and not classification?
3. Why can’t we use Mean Square Error (MSE) as a cost function for Logistic Regression?
4. Why can’t we use Linear Regression in place of Logistic Regression for Binary classification?
5. How can we express the probability of a Logistic Regression model as conditional probability?

**EXPERIMENT 4**

**AIM:**  Program to demonstrate Decision Tree-ID3 Algorithm.

## INTRODUCTION:

A decision tree is a structure that contains nodes (rectangular boxes) and edges (arrows) and is built from a dataset (table of columns representing features/attributes and rows corresponds to records). Each node is either used to make a decision (known as decision node) or represent an outcome (known as leaf node).

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes (divides) features into two or more groups at each step. Invented by Ross Quinlan, ID3 uses a top-down greedy approach to build a decision tree.

In simple words, the top-down approach means that we start building the tree from the top and the greedy approach means that at each iteration we select the best feature at the present moment to create a node. Most generally ID3 is only used for classification problems with nominal features only.

**Link of data Set to be used:** [**https://www.kaggle.com/datasets/uciml/iris**](https://www.kaggle.com/datasets/uciml/iris)

##### Steps to follow:

##### Import Necessary Libraries including

##### From sklearn.tree import DecisionTreeClassifier

1. Split dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. Use the gini index and entropy as criterion for splitting the tree and fit the model.

DecisionTreeClassifier(criterion='gini')

clf\_entropy = DecisionTreeClassifier(criterion='entropy')

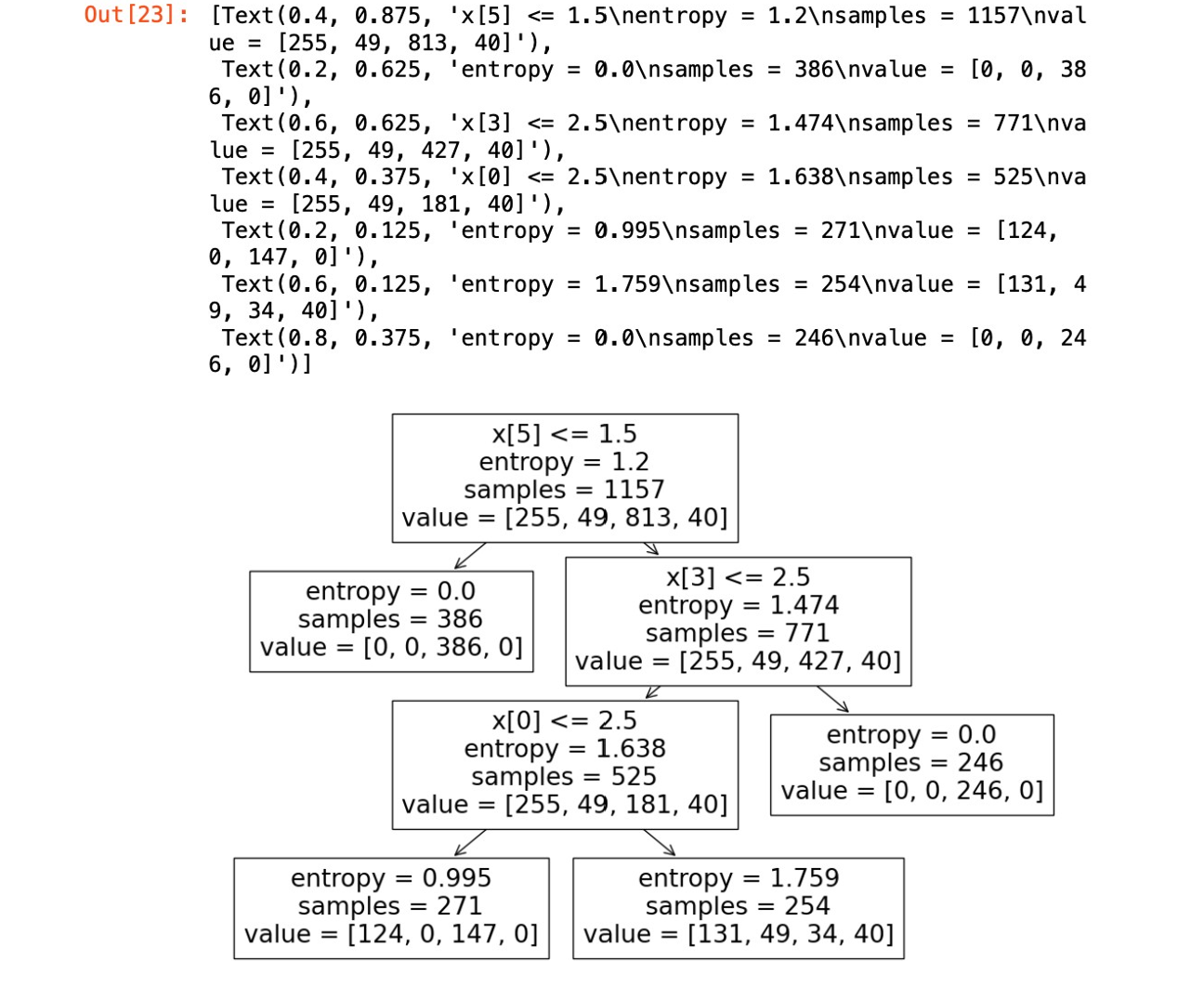
1. Visualize the resulting decision trees using graphviz library and save the visualization in file.

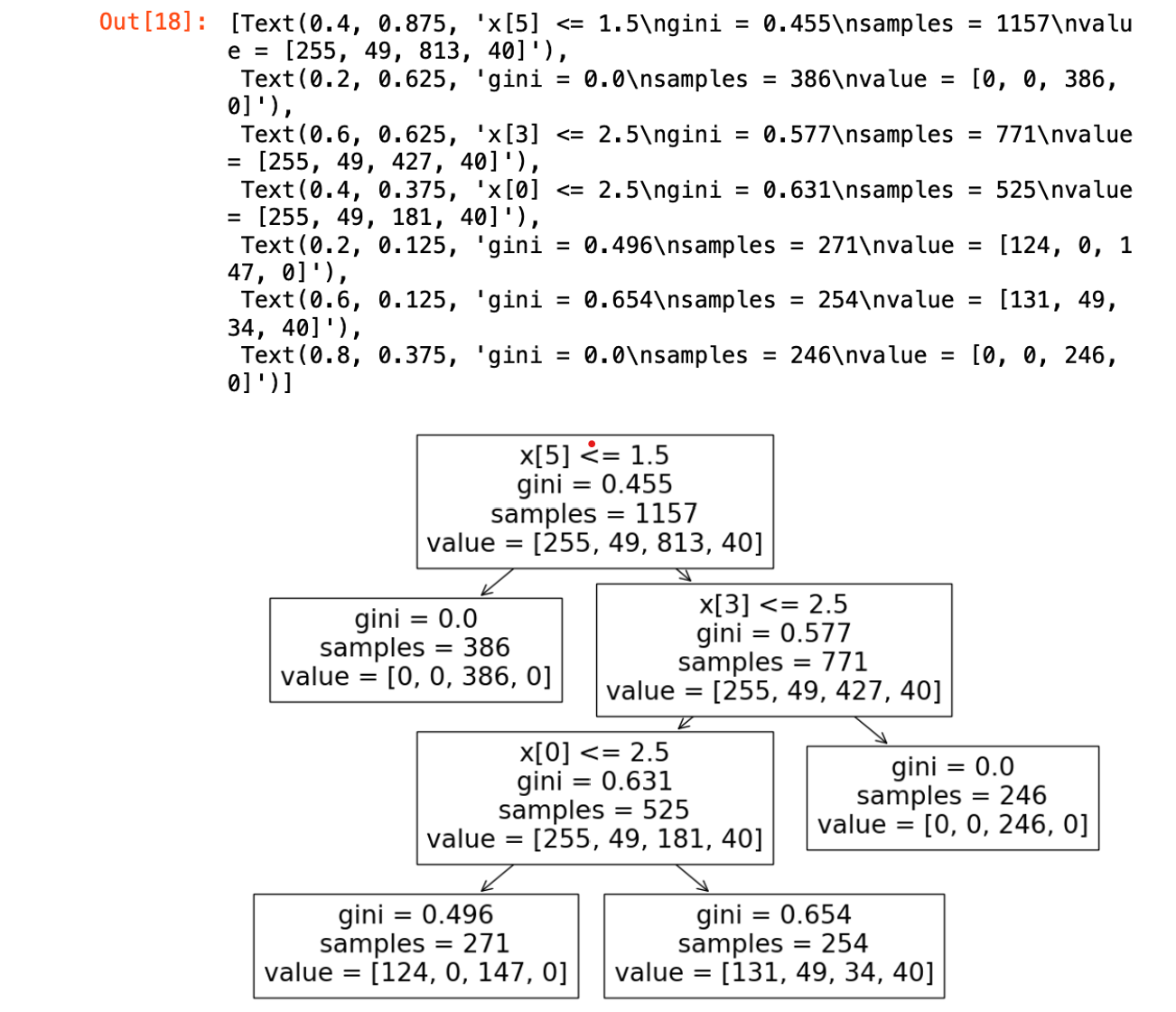
graph\_gini.render('decision\_tree\_gini', format='png', cleanup=True)

graph\_entropy.render('decision\_tree\_entropy', format='png', cleanup=True)

1. Display Decision tree visualizations with Gini Index and show intermediate calculated values in the tree till the pure nodes are not reached.

**Sample Output:**

****



**ADDITIONAL PRACTICE PROBLEM:**

1. Program for Decision Tree without using Sklearn Library.

Link for the dataset: <http://archive.ics.uci.edu/dataset/19/car+evaluation>

1. Program to demonstrate how decision trees can be used for both classification and regression. Link for the dataset: <https://www.kaggle.com/code/pankajy/admission-prediction>

## VIVA QUESTIONS:

1. How does a Decision Tree handle missing attribute values?
2. How does a Decision Tree handle continuous (numerical) features?
3. Are Decision Trees affected by the outliers?
4. How does the decision tree compare with linear regression and logistic regression?
5. If it takes one hour to train a Decision Tree on a training set containing 1 million instances, roughly how much time will it take to train another Decision Tree on a training set containing 10 million instances?
6. What does the function plt.imread ( ) does?
7. What is the use of the function graph\_entropy.render ( )?

# EXPERIMENT 5

## AIM: To demonstrate k-Nearest Neighbor flowers classification

## INTRODUCTION:

The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values. During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the K nearest neighbors to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the K neighbors as the predicted label for the input data point. For regression, it calculates the average or weighted average of the target values of the K neighbors to predict the value for the input data point. KNN Algorithm can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
* 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.
* Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar

features of the new data set to the cats and dog’s images and based on the most similar features it will put it in either cat or dog category.

**Link of data Set to be used:** [**https://www.kaggle.com/datasets/uciml/iris**](https://www.kaggle.com/datasets/uciml/iris)

**Steps to follow:**

The K-NN working can be explained on the basis of the below algorithm:

* Step-1: Select the number K of the neighbors
* Step-2: Calculate the Euclidean distance of K number of neighbors
* Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
* Step-4: Among these k neighbors, count the number of the data points in each category.
* Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
* Step-6: Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the Euclidean distance between the data points.

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.

Use the KNN implementation from scikit learn to classify the Iris dataset. Also show how performance of the model varies with different values of k..

##### ACCURACY vs NUMBER OF NEIGHBORS

# Step 1: Load the Iris dataset

# Step 2: Split the dataset into training and testing sets

# Step 3: Implement KNN using scikit-learn's KNeighborsClassifier

# Step 4 and 5: Evaluate the performance of the model with different values of k # Step 6: Plot a graph showing the accuracy variation with different k values

##### ERROR RATE vs NUMBER OF NEIGHBORS

# Step 1: Load the Iris dataset

# Step 2: Split the dataset into training and testing sets

# Step 3: Implement KNN using scikit-learn's KNeighborsClassifier

# Step 4 and 5: Evaluate the performance of the model with different values of k

# Step 6: Plot a graph showing the error rate variation with different k values plt.plot(k\_values, error\_rates, marker='o')

##### VALIDATION ERROR RATE vs NUMBER OF NEIGHBORS

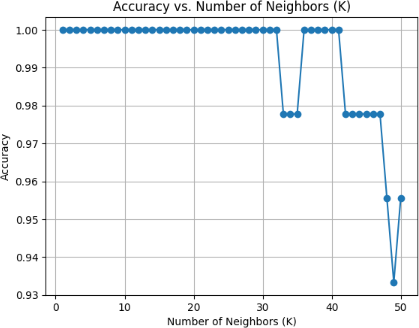
# Step 1: Load the Iris dataset

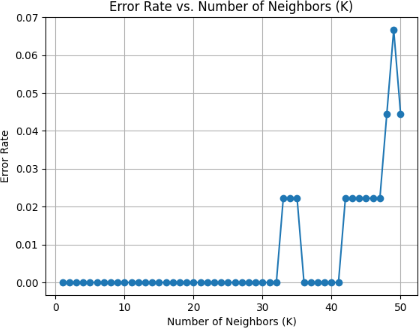
# Step 2: Split the dataset into training, validation, and testing sets

# Step 3: Implement KNN using scikit-learn's KNeighborsClassifier

# Step 4 and 5: Evaluate the performance of the model with different values of k

# Step 6: Plot a graph showing the validation error rate variation with different k values

**Sample Output:**



**ADDITIONAL PRACTICE PROBLEMS**

1. Demonstrate implementation of KNN with different distance metrics. Use the iris dataset only.
2. Demonstrate implementation of Knn to any instance of Pima Indians Diabetes Database Link for the dataset : <https://www.kaggle.com/datasets/saurabh00007/diabetescsv>

## VIVA QUESTIONS

* 1. Why is k-NN a non-parametric algorithm?
  2. Describe the method used for feature scaling in k-NN algorithm?
  3. Why is it recommended not to use the k-NN Algorithm for large datasets?
  4. How to choose the optimal value of k in the k-NN Algorithm?
  5. How can you relate k-NN algorithm to the bias-variance Tradeoff?
  6. Why is the odd value of ‘k’ preferred over an even value in the k-NN algorithm?
  7. What is the role of the k value in the k-NN algorithm?

**EXPERIMENT 6**

## AIM: Program to demonstrate Naive-Bayes Classifier.

**INTRODUCTION:**

Naive Bayes belongs to a family of generative learning algorithms, aiming to model the distribution of inputs within a specific class or category. Unlike discriminative classifiers such as logistic regression, it doesn’t learn which features are most crucial for distinguishing between classes. It’s widely used in text classification, spam filtering, and recommendation systems.

It is a classification technique based on Bayes’ Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes classifiers are among the simplest Bayesian network models, yet they can achieve high accuracy levels when coupled with kernel density estimation. This technique involves using a kernel function to estimate the probability density function of the input data, allowing the classifier to improve its performance in complex scenarios where the data distribution is not well-defined. As a result, the naive Bayes classifier is a powerful tool in machine learning, particularly in text classification, spam filtering, and sentiment analysis, among others.

**Link of data Set to be used:** [**https://archive.ics.uci.edu/dataset/2/adult**](https://archive.ics.uci.edu/dataset/2/adult)

**Steps to follow:**

1. Import necessary libraries

from sklearn.naive\_bayes import GaussianNB

1. Load the dataset and preprocess the dataset and drop the null values
2. Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,random\_state=42)

4. Stop Wireshark packet capture and Initialize the Naive Bayes classifier clf = GaussianNB()

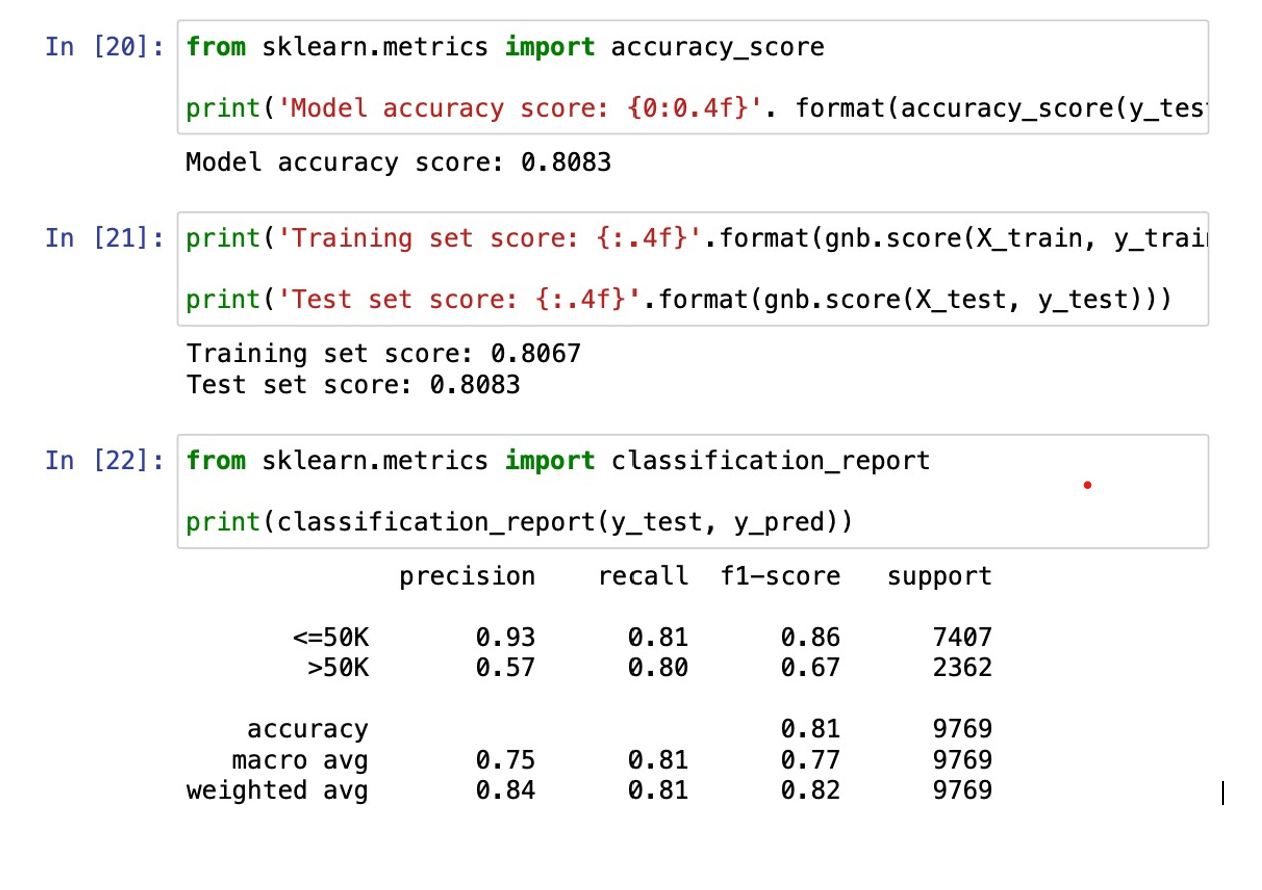
5. Train the classifier and make predictions on the test data

clf.fit(X\_train, y\_train) and set y\_pred = clf.predict(X\_test)

1. Calculate accuracy of the model

accuracy = accuracy\_score(y\_test, y\_pred)

**Sample Output:**

****

## VIVA QUESTIONS

1. What is the basic assumption in the case of the Naive Bayes classifier?
2. What are the possible advantages of choosing the Naive Bayes classifier?
3. Is feature scaling required in Naive Bayes?
4. What are different problem statements you can solve using Naive Bayes?
5. Is feature scaling required in Naive Bayes?
6. Does Naive Bayes fall under the category of the discriminative or generative classifier?
7. How does Naive Bayes treats categorical and numerical values?

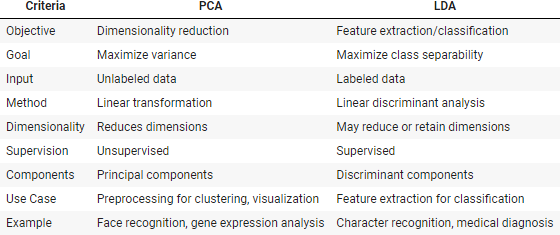
**EXPERIMENT 7**

24

**AIM:** Program to demonstrate PCA and LDA on Iris dataset.

**INTRODUCTION:**

**PCA** is an unsupervised method of dimensionality reduction that aims to find the directions of maximum variance in a dataset. The idea is to find a smaller set of variables or features that capture the most important patterns in the data. PCA works by first centering the data on its mean and then finding the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the directions of maximum variance, while the eigenvalues represent the amount of variance explained by each eigenvector. The eigenvectors are then used to project the data onto a lower-dimensional space. The number of principal components to keep is determined by the amount of variance we want to retain.

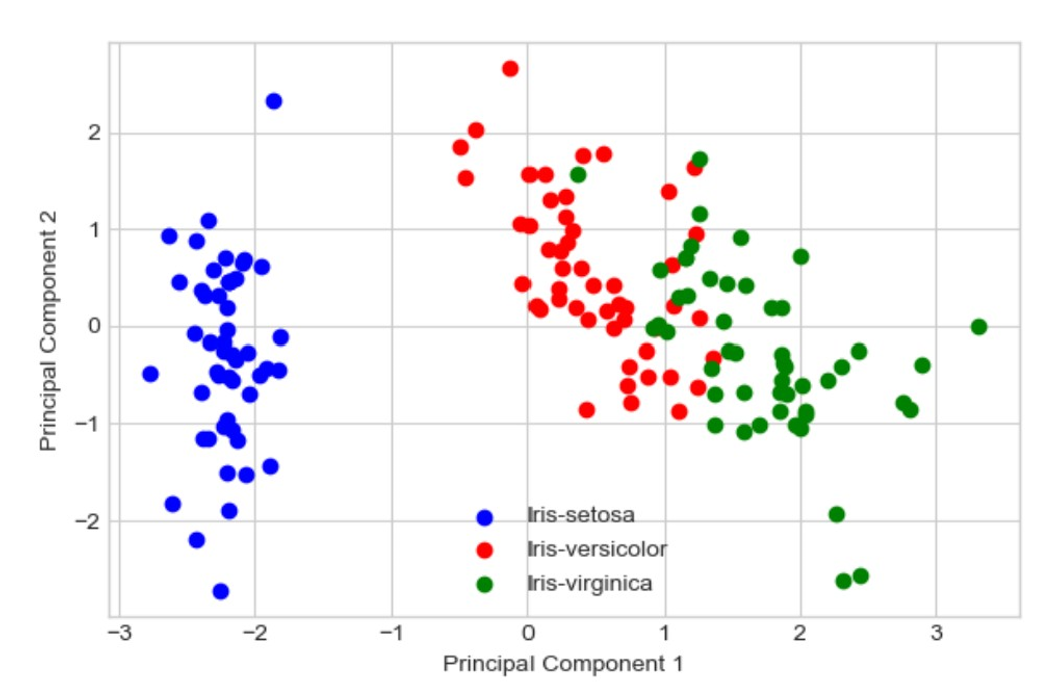
**LDA** is a supervised method of dimensionality reduction that aims to find the linear combination of features that best separates the classes in a dataset. The idea is to reduce the dimensionality of the data while preserving the information that is most relevant for class discrimination. LDA works by first calculating the mean and covariance matrix for each class in the data. It then calculates the between-class scatter matrix and the within-class scatter matrix. The goal is to find a projection that maximizes the ratio of the between-class scatter matrix to the within-class scatter matrix. This projection is the linear discriminant function. Once the linear discriminant function is found, we can project the data onto this function to obtain the reduced representation of the data. The resulting transformed data can be used for classification.

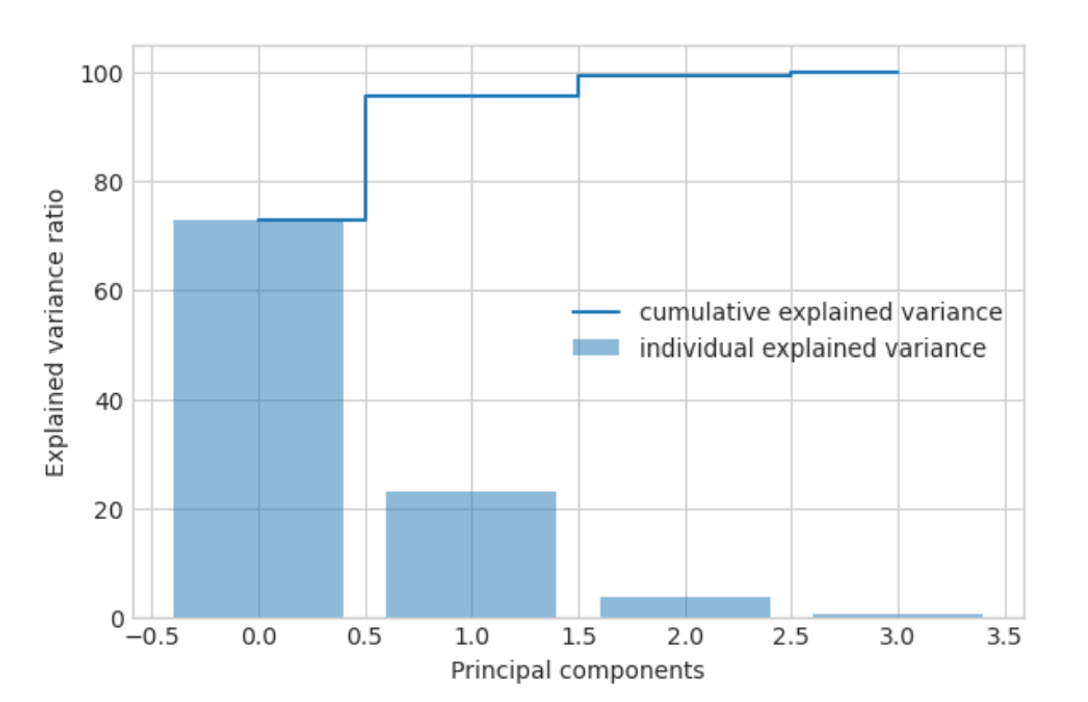
**Link of data Set to be used:** [**https://www.kaggle.com/datasets/uciml/iris**](https://www.kaggle.com/datasets/uciml/iris)

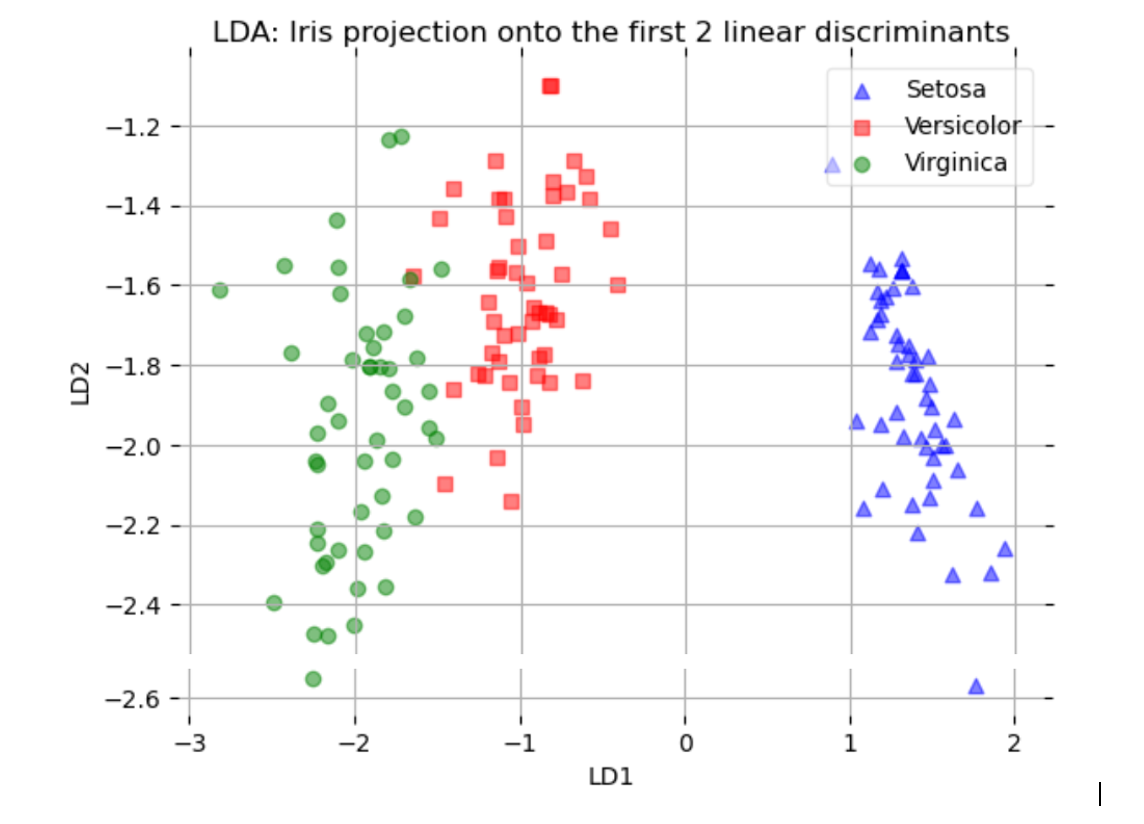
##### Steps to follow:

1. Import the necessary libraries
2. Load the IRIS dataset iris and preprocess the dataset
3. For Principal component analysis:
4. Standardize the data
5. Do eigen decomposition
6. Sort the eigen value and eigen vector tuples from high to low and determine the top k eigenvectors
7. Make the projection matrix and and project data X onto new feature space.
8. For Linear Discriminant Analysis:
9. First encode the label so its convenient to work with
10. Set the bin sizes and plot the histogram and plot the annotation
11. Hide axis ticks and remove axis spines

**Sample output:**

****

****

****

## ADDITIONAL PRACTICE PROBLEM

## Program to demonstrate PCA and LDA from different datasets. Use the datasets from Sklearn.

## VIVA QUESTIONS

1. What is the most suitable application of LDA?
2. How do you estimate how much each variable contributes to the separation?
3. Is LDA a supervised or unsupervised method?
4. Can LDA be used as a multi-class classifier? If so how would it work?
5. Can Linear Discriminant Analysis be used for clustering?
6. What do the terms sensitivity and specificity mean?

**EXPERIMENT 8**

## AIM: Program to demonstrate DBSCAN clustering algorithm

## INTRODUCTION:

Clusters are dense regions in the data space, separated by regions of the lower density of points. Clustering analysis is an unsupervised learning method that separates the data points into several specific bunches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises of many different methods based on different distance measures. Centrally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches.

The DBSCAN algorithm is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.

**Link of data Set to be used:** **<https://www.kaggle.com/datasets/uciml/iris>**

**Steps to follow:**

1. Import the necessary libraries including

from sklearn.cluster import DBSCAN

from sklearn.preprocessing import StandardScaler

1. Load Iris dataset data and standardize the data .
2. Implement the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm.

##### Args:

D: An array representing the dataset, where each row is a data point.

eps: The maximum distance between two points to be considered neighbors.

MinPts: The minimum number of points required to form a dense region.

##### Returns:

An array of labels for each data point, where:

* 0: Unclassified
* -1: Noise
* Positive value: Cluster ID

1. Initialize labels for all points and keep track of the cluster id
2. Find the neighbors of a point P within a radius of eps.

##### Args:

P: The index of the point in the dataset.

eps: The radius within which to search for neighbors.

##### Returns: A list of indices of the neighboring points.

1. Expands a cluster from a seed point P by recursively adding its neighbors.

##### Args:

P: The index of the seed point.

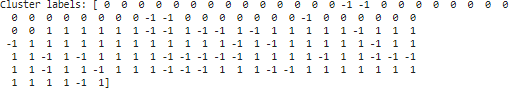
neighbors: A list of indices of the initial neighbors of P.

cluster\_id: The ID of the cluster being expanded.

UNCLASSIFIED neighbor: Add to cluster and explore its neighbors

Mark as NOISE if not in a dense region else assign a new cluster id and return label

**Sample output**



## ADDITIONAL PRACTICE PROBLEM

1. Program to compare K-means clustering and DBSCAN for IRIS dataset based on performance metrics.

Link for dataset: <https://www.kaggle.com/datasets/uciml/iris>

## VIVA QUESTIONS

1. List out the Input parameters given to the DBSCAN Algorithm.
2. How can we interpret the parameters “eps” and “min\_pts” in high dimensions for the DBSCAN Algorithm?
3. How does the epsilon value affect the DBSCAN Clustering Algorithm?
4. How is the parameter “Distance-function” estimated in the DBSCAN Algorithm?
5. How many types of points do we get after applying a DBSCAN Algorithm to a particular dataset?
6. How is the parameter “eps” estimated in the DBSCAN Algorithm?
7. What is the time complexity of the DBSCAN Clustering Algorithm?

# EXPERIMENT 9

# AIM: Program to demonstrate K-Medoid clustering algorithm

# Introduction:

K-Medoids clustering is an unsupervised machine learning algorithm used to group data into different clusters. It is an iterative algorithm that starts by selecting k data points as medoids in a dataset. After this, the distance between each data point and the medoids is calculated. Then, the data points are assigned to clusters associated with the medoid at the minimum distance from each data point. Here, the medoid is the most centrally located point in the cluster. Once we assign all the data points to the clusters, we calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. We term the sum of distances as the cost. After calculating the cost, we select a temporary non-medoid point randomly from the dataset and swap a medoid with the selected point. Then we recalculate the cost of all the non-medoid data points to the medoid of each cluster considering the temporarily selected point as the medoid. If the newly calculated cost is less than the previous cost, we make the temporary point the permanent centroid. If the new cost is greater than the previous cost, we undo the changes. Then, we again select a non-medoid point and repeat the process until the cost is minimized.

**Link of Dataset to be used:** <https://www.kaggle.com/datasets/dongeorge/seed-from-uci>

**Steps to follow:**

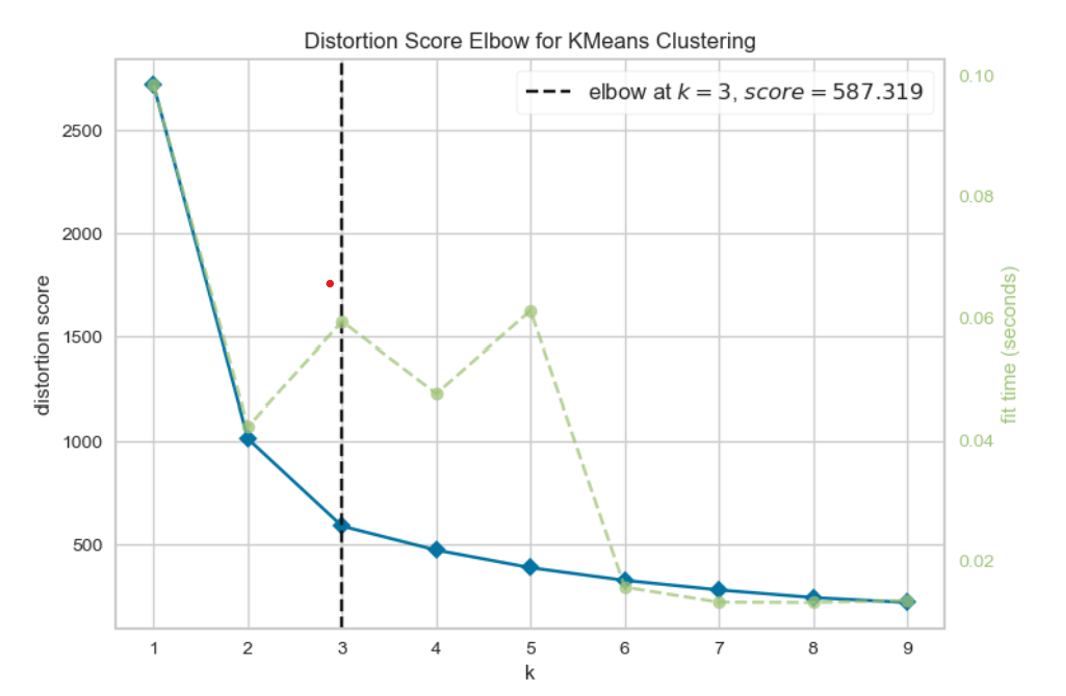
1. Import the necessary libraries including

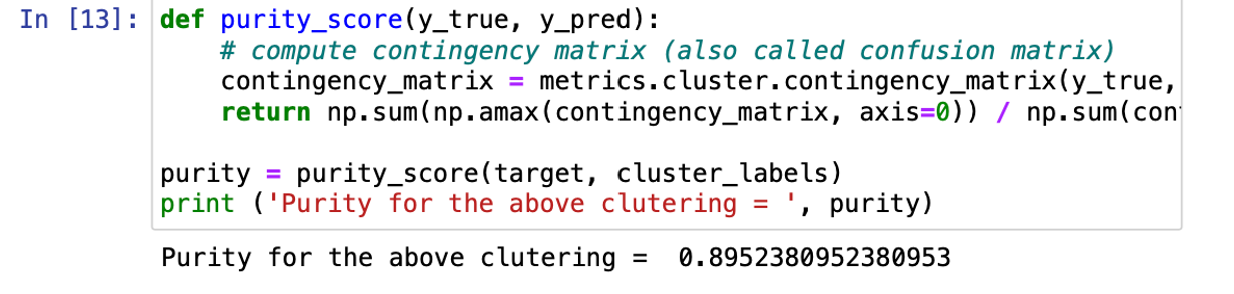
From sklearn.cluster import KMeans

From yellowbrick.cluster import KElbowVisualizer

1. Load the dataset and use KElbowVisualizer to fit the model and display the distortion score Elbow for KMeans Clustering.
2. From pyclustering.cluster.kmediods import kmediods and randomly pick 3 indexes from the original sample as the mediods.
3. Create instance of K-Mediods algorithm with prepared centers.
4. Run cluster analysis and prepare cluster labels.
5. Find the silhouette coefficient for the clustering and compute contingency matrix to find the purity score.

**Sample Output:**

****

****

**VIVA QUESTIONS:**

1. Give difference between K-means and K-medoid clustering?
2. What are the advantages of K-medoid clustering?
3. What is the silhouette score, and how is it used in evaluating clustering results?
4. How do you handle missing or noisy data in clustering algorithms?
5. What is the role of feature scaling in clustering algorithms?
6. What is the curse of dimensionality, and how does it impact clustering?

# EXPERIMENT 10

**AIM:** Program to demonstrate K-Means Clustering Algorithm on Handwritten Dataset

**Introduction**: K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training. It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters. The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters.

**Steps to follow:**

1. Import the necessary libraries including

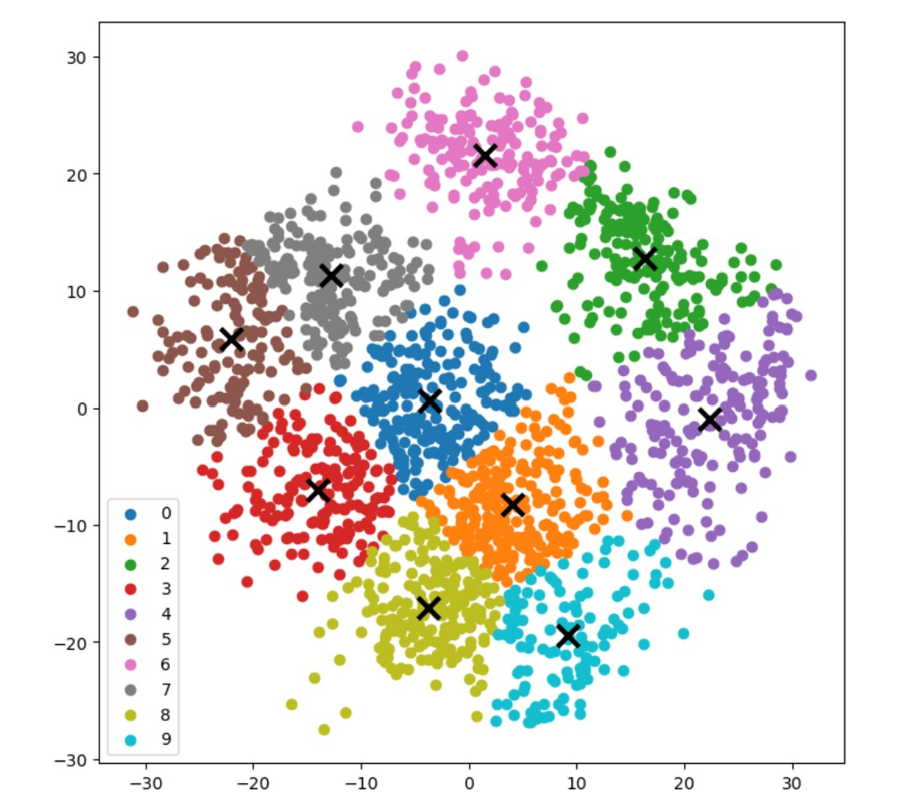
From sklearn.cluster import KMeans

From sklearn.datasets import load\_digits

# digits dataset from scikit learn consists of 8x8 pixel images. This is the copy of the testset of the UCI ML hand-written digits database.

1. Load the dataset in digits and run Kmeans clustering.
2. Reshape 10 rows of clusters and find the accuracy score.
3. Perform PCA by reducing the dataset, fit the kmeans model, calculate the centroids and plot the clusters.

**Sample Output:**



**ADDITIONAL PRACTICE PROBLEM:**

1. Perform the K-means on the mall-customers dataset. Link for the dataset: <https://www.kaggle.com/datasets/shwetabh123/mall-customers>

**VIVA QUESTIONS:**

1. What are centroids in the context of K-Means?
2. What is the role of distance metrics in K-Means, and which distances can be used?
3. How do you decide on the number of clusters (k) in a K-Means algorithm?
4. Can K-Means clustering be used for categorical data? If so, how?
5. Compare K-Means clustering with hierarchical clustering.
6. How does K-Means Clustering react to non-spherical cluster shapes?

**BEYOND CURRICULUM**

# EXPERIMENT 11

## AIM: Program to visualize data classification using SVM with different kernels

## INTRODUCTION:

Support Vector Machines aim to find a hyperplane that effectively separates the classes in their training data by maximizing the margin between the outermost data points of each class. This is achieved by finding the best weight vector that defines the decision boundary hyperplane and minimizes the sum of hinge losses for misclassified samples, as measured by the hinge\_loss function. By default, regularization is applied with the parameter C=1, which allows for a certain degree of misclassification tolerance.

If the data is not linearly separable in the original feature space, a non-linear kernel parameter can be set. Depending on the kernel, the process involves adding new features or transforming existing features to enrich and potentially add meaning to the data. When a kernel other than "linear" is set, the SVC applies the kernel trick, which computes the similarity between pairs of data points using the kernel function without explicitly transforming the entire dataset. The kernel trick surpasses the otherwise necessary matrix transformation of the whole dataset by only considering the relations between all pairs of data points. The kernel function maps two vectors (each pair of observations) to their similarity using their dot product. The hyperplane can then be calculated using the kernel function as if the dataset were represented in a higher- dimensional space. Using a kernel function instead of an explicit matrix transformation improves performance, as the kernel function has a time complexity of O(n2) whereas matrix transformation scales according to the specific transformation being applied. In this example, we compare the most common kernel types of Support Vector Machines: the linear kernel ("linear"), the polynomial kernel ("poly"), the radial basis function kernel ("rbf") and the sigmoid kernel ("sigmoid").

##### Training SVC model and plotting decision boundaries

We define a function that fits a SVC classifier, allowing the kernel parameter as an input, and then plots the decision boundaries learned by the model using DecisionBoundaryDisplay.

Notice that for the sake of simplicity, the C parameter is set to its default value (C=1) in this example and the gamma parameter is set to gamma=2 across all kernels, although it is automatically ignored for the linear kernel. In a real classification task, where performance

matters, parameter tuning (by using GridSearchCV for instance) is highly recommended to capture different structures within the data.

Setting response\_method="predict" in DecisionBoundaryDisplay colors the areas based on their predicted class. Using response\_method="decision\_function" allows us to also plot the decision boundary and the margins to both sides of it. Finally the support vectors used during training (which always lay on the margins) are identified by means of the support\_vectors\_ attribute of the trained SVCs, and plotted as well.

##### Linear kernel

Linear kernel is the dot product of the input samples. It is then applied to any combination of two data points (samples) in the dataset. The dot product of the two points determines the cosine\_similarity between both points. The higher the value, the more similar the points are. plot\_training\_data\_with\_decision\_boundary("linear")

Training a SVC on a linear kernel results in an untransformed feature space, where the hyperplane and the margins are straight lines. Due to the lack of expressivity of the linear kernel, the trained classes do not perfectly capture the training data.

##### Polynomial kernel

The polynomial kernel changes the notion of similarity. Training a SVC on a linear kernel results in an untransformed feature space, where the hyperplane and the margins are straight lines. Due to the lack of expressivity of the linear kernel, the trained classes do not perfectly capture the training data.

plot\_training\_data\_with\_decision\_boundary("poly")

##### RBF kernel

The radial basis function (RBF) kernel, also known as the Gaussian kernel, is the default kernel for Support Vector Machines in scikit-learn. It measures similarity between two data points in infinite dimensions and then approaches classification by majority vote. plot\_training\_data\_with\_decision\_boundary("rbf")

**Link of Dataset to be used:** <https://www.kaggle.com/datasets/rakeshrau/social-network-ads>

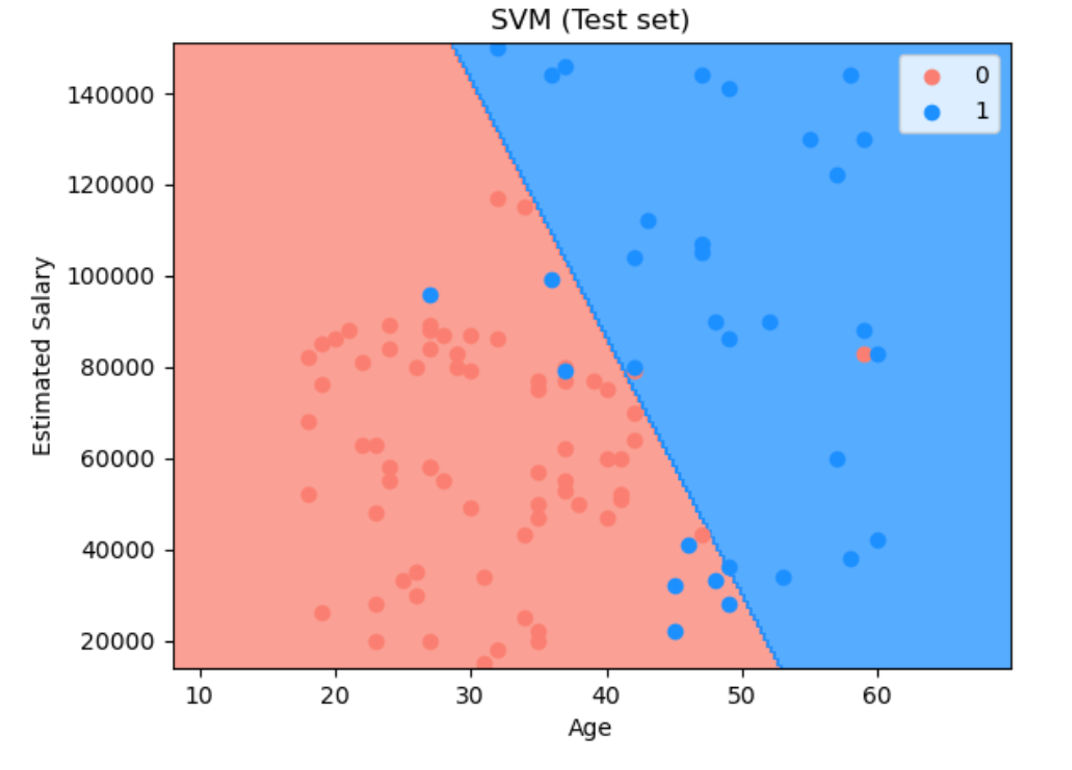
**Steps to follow:**

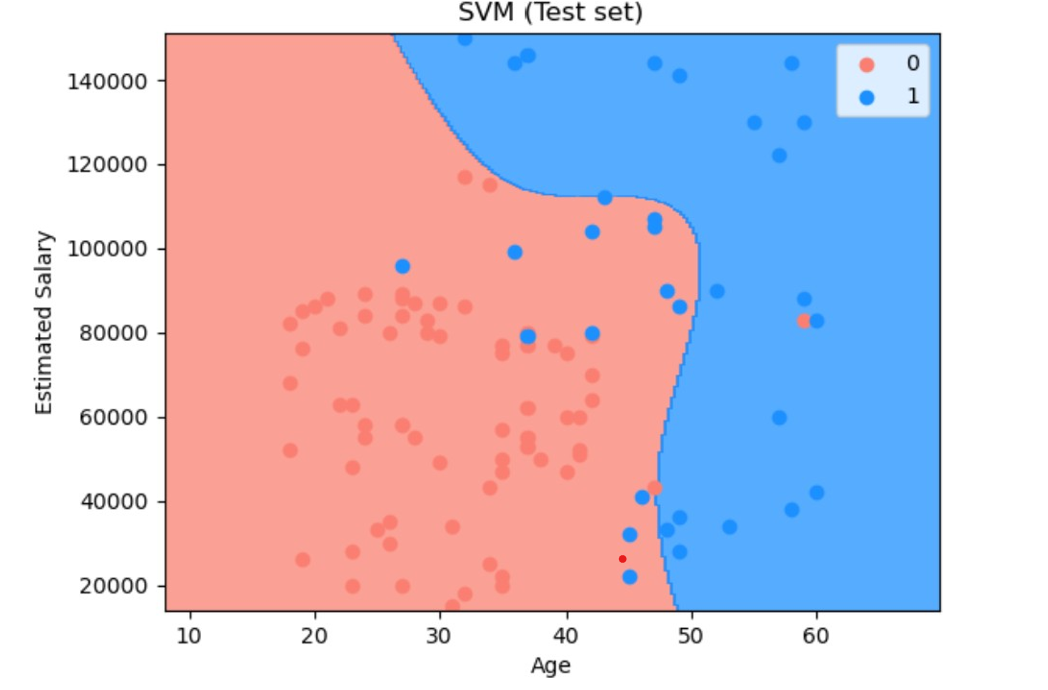
1. Import necessary libraries including

From sklearn.svm import SVC

1. Load the dataset and perform preprocessing of data
2. Build a classifier using SVC for Kernel = linear and find the confusion matrix and accuracy score. Also draw the scatter plot between Attributes Age and Estimated Salary by SVM(test set)
3. Build a classifier using SVC for Kernel = Poly and find the confusion matrix and accuracy score. Also draw the scatter plot between Attributes Age and Estimated Salary by SVM(test set)

**Sample Output:**

****

****

## VIVA QUESTIONS

1 In what conditions will you choose SVM over any other algorithm?

1. How SVM is impacted when the dataset has missing values?
2. What are the possible outcomes if outliers are present in the dataset?
3. How is SVM different from K-Nearest Neighbors (KNN)?
4. Which among SVM and Logistic regression is the better algorithm for handling outliers?

# EXPERIMENT 12

**AIM:** Study and Evaluate ML algorithm with balanced and unbalanced datasets.

## INTRODUCTION:

Evaluating machine learning algorithms on balanced versus unbalanced datasets is crucial because the performance metrics can vary significantly depending on the class distribution. Here's a detailed approach to evaluate and compare these scenarios:

### Understanding Dataset Balance

* + **Balanced Dataset**: Each class has approximately the same number of samples. For example, in a binary classification problem, both classes might have 500 examples each.
  + **Unbalanced Dataset**: One class significantly outnumbers the other. For example, in a binary classification problem, you may have 950 examples of Class A and only 50 examples of Class B.

### Choosing Metrics

The choice of evaluation metrics is essential for both balanced and unbalanced datasets. Common metrics include:

* + **Accuracy**: The ratio of correctly predicted instances to the total instances. However, it can be misleading in unbalanced datasets.
  + **Precision**: The ratio of true positive predictions to the sum of true positives and false positives. Useful when the cost of false positives is high.
  + **Recall (Sensitivity)**: The ratio of true positive predictions to the sum of true positives and false negatives. Useful when the cost of false negatives is high.
  + **F1 Score**: The harmonic mean of precision and recall. It balances the two metrics and is especially useful in unbalanced datasets.
  + **ROC-AUC**: The Area under the Receiver Operating Characteristic Curve. It evaluates the performance across all classification thresholds and is useful for comparing classifiers in unbalanced datasets.
  + **Precision-Recall AUC**: The Area under the Precision-Recall Curve. It is particularly informative for unbalanced datasets where one class is rare.

### Evaluation Strategy

##### For Balanced Datasets:

1. **Train-Test Split**: Use a standard train-test split or cross-validation to evaluate the model.
2. **Model Selection**: Since the dataset is balanced, traditional accuracy might be a good indicator of model performance, but also consider precision, recall, and F1 score to get a fuller picture.
3. **Cross-Validation**: K-fold cross-validation can help assess how the model generalizes to different subsets of the data.

##### For Unbalanced Datasets:

1. **Resampling Techniques**:
   * **Oversampling**: Increase the number of samples in the minority class (e.g., SMOTE).
   * **Undersampling**: Decrease the number of samples in the majority class.

##### Evaluation Metrics:

* + Focus on metrics like Precision, Recall, F1 Score, and ROC-AUC.
  + Precision-Recall AUC can be particularly useful.

1. **Confusion Matrix**: Examine the confusion matrix to understand the types of errors your model is making (false positives vs. false negatives).
2. **Class Weights**: Some algorithms allow you to assign weights to different classes, which can help the model focus more on the minority class.

### Example Comparison

* + **Balanced Dataset**: If you have a balanced dataset and your model achieves 90% accuracy, it’s likely performing well across all classes. However, you should still check precision, recall, and F1 score to confirm this.
  + **Unbalanced Dataset**: In an unbalanced dataset where the model achieves 90% accuracy, it might simply be predicting the majority class most of the time. In such cases, a high F1 score and ROC-AUC are more reliable indicators of true performance.

### Visualizing Performance

* + **ROC Curve**: Plot ROC curves to visualize performance across different thresholds.
  + **Precision-Recall Curve**: Plot Precision-Recall curves to assess the trade-offs between precision and recall for different thresholds.
  + **Confusion Matrix**: Visualize the confusion matrix to get detailed insights into false positives and false negatives.

## Steps to follow:

## Import the necessary libraries inclusing

from sklearn.datasets import make\_classification from sklearn.linear\_model import LogisticRegression

1. Generate balanced dataset

X\_balanced, y\_balanced = make\_classification(n\_samples=1000, n\_features=20, n\_classes=2, n\_clusters\_per\_class=1, weights=[0.5, 0.5], flip\_y=0.1, random\_state=42)

1. Generate unbalanced dataset

X\_unbalanced, y\_unbalanced = make\_classification(n\_samples=1000, n\_features=20, n\_classes=2, n\_clusters\_per\_class=1,weights=[0.9, 0.1], flip\_y=0.1, random\_state=42)

1. Perform the Train-test split for balanced and unbalanced dataset.
2. Initialize and train classifier using Logistic Regression
3. Generate the Predictions and probabilities
4. Find the Evaluation metrics like accuracy scores, precision, score, F1 Score
5. Plot the ROC Curve and Precision -Recall curves for both balanced and unbalanced datasets and display the confusion matrix.

## VIVA QUESTIONS

1. Explain the significance of precision, recall, and F1 score in the context of an unbalanced dataset.
2. Why might accuracy be misleading in an unbalanced dataset?
3. Why do we use predict\_proba instead of predict for evaluating ROC and Precision-Recall curves?
4. How does the train\_test\_split function contribute to model evaluation?
5. How does a confusion matrix help in understanding model performance?
6. How do you handle class imbalance when training a model?
7. How would you interpret an F1 score of 0.5 in the context of an unbalanced dataset?

**SAMPLE VIVA QUESTIONS**

1. What's the trade-off between bias and variance?
2. What does a high true positive rate but low false positive rate indicate about a model’s performance?
3. Explain how you might handle a situation where the model performs well on training data but poorly on test data.
4. Explain over- and under-fitting and how to combat them
5. What is regularization, why do we use it, and give some examples of common methods?
6. Given stride S and kernel sizes for each layer of a (1-dimensional) CNN, create a function to compute the receptive field of a particular node in the network. This is just finding how many input nodes actually connect through to a neuron in a CNN
7. How would you remove outliers when trying to estimate a flat plane from noisy samples?
8. Describe how convolution works. What about if your inputs are grayscale vs RGB imagery? What determines the shape of the next layer?
9. Why do we have max-pooling in classification CNNs?
10. Why do we need a validation set and test set? What is the difference between them? [
11. Why do ensembles typically have higher scores than individual models?
12. Explain the differences between supervised, unsupervised, and reinforcement learning?
13. What is the difference between Batch Gradient Descent and Stochastic Gradient Descent?
14. What's the difference between boosting and bagging?
15. Explain how a ROC curve works.
16. What’s the difference between Type I and Type II error?
17. When to use a Label Encoding vs. One Hot Encoding?
18. What is the difference between LDA and PCA for dimensionality reduction?
19. Differentiate between inductive learning and deductive learning?
20. How do classification and regression differ?
21. What are the common ways to handle missing data in a dataset?
22. What are the classification methods that SVM can handle?
23. Explain True Positive, True Negative, False Positive, and False Negative in Confusion Matrix with an example.
24. Why instance-based learning algorithm sometimes referred to as Lazy learning algorithm?
25. How is a decision tree pruned?
26. Why do we need to convert categorical variables into factor? Which functions are used to perform the conversion?
27. How is machine learning used in day-to-day life?
28. How is a logistic Regression model evaluated?
29. How to handle multicollinearity in regression model?
30. What is the vanishing gradient problem, and how is it resolved?
31. What are Generative Adversarial Networks (GANs), and how do they work?
32. How do you handle imbalanced datasets?
33. What is transfer learning, and how is it applied in machine learning?
34. How do you deal with outliers in data?
35. What are common pitfalls in feature selection?
36. How do you identify and mitigate bias in machine learning models?
37. What are the ethical concerns associated with AI and machine learning?
38. How can machine learning models unintentionally perpetuate bias?
39. How do you deploy machine learning models into production?
40. What is gradient descent, and how does it optimize a model?
41. What are learning rate and momentum in optimization?
42. What is early stopping, and why is it used?
43. What is data augmentation, and how can it improve model performance?
44. What are batch size and epoch, and how do they affect training?
45. What are the differences between Adam, SGD, and RMSprop optimizers?
46. What are autoencoders, and what types of problems are they used for?
47. How do XGBoost and LightGBM improve upon traditional Gradient Boosting?
48. Suppose you’re given a high-dimensional dataset. How would you reduce its dimensionality?
49. Your model's accuracy is low. How would you improve it?
50. What is the "black-box" nature of machine learning models, and how can it be addressed?