****

**LAB MANUAL**

**AML 331 PYTHON AND MACHINE LEARNING LAB**

**SEMESTER V**

***Submitted By***

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**MANGALAM COLLEGE OF ENGINEERING**

**Mangalam Hills Vettimukal P O Kottayam**

**Affiliated to APJ Abdul Kalam Technological University Approved By the All India Council For Technical Education(AICTE)**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**GENERAL LABORATORY INSTRUCTIONS**

**1.** Students are advised to come to the laboratory at least 5 minutes before (to the starting

time),those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab

with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim,

Algorithm, Procedure, Program, Expected Output,etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils

(if any) needed in the lab.

c. Proper Dress code and Identity Card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer

system allotted to you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab

observation note book, and get certified by the concerned faculty.

6.All the students should be polite and cooperative with the laboratory staff, must maintain

the discipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high end branded systems, which

should be utilized properly.

8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the

lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will

attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out ;if

anybody found loitering outside the lab / class without permission during working hours

will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves

the lab after completing the task (experiment) in all aspects. He/she must ensure the

system / seat is kept properly.

**LIST OF EXPERIMENTS**

| **Sl No** | **Experiments** |
| --- | --- |
| **1** | **Introduction to Python Programming.** |
| **2** | **Familiarization of basic Python Libraries such as Sklearn, Numpy, Pandas and Matplotlib.\*** |
| **3** | **Write a Python program to find the union and intersection of two lists.** |
| **4** | **Design a Python program to count the occurrences of each word in a given sentence.** |
| **5** | **Write a Python program to multiply two matrices.\*** |
| **6** | **Write a Python program to find the most frequent words in a text file.\*** |
| **7** | **Implement and demonstrate Single, Multi variable and Polynomial Regression for a given set of training data stored in a .CSV file and evaluate the accuracy.\*** |
| **8** | **Implement a Python program to perform logistic regression on a dataset.** |
| **9** | **Write a Python program to implement Naive Bayes classifier and calculate the accuracy, precision, and recall for your data set.\*** |
| **10** | **Write a Python program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.\*** |
| **11** | **Assuming a set of data that need to be classified, use a Support Vector Machine classifier to perform this task and evaluate the accuracy.\*** |
| **12** | **Implement K-Nearest Neighbor algorithm to classify any dataset.** |
| **13** | **Implement K-Means Clustering using any given dataset.\*** |
| **14** | **Build an Artificial Neural Network using Backpropagation algorithm and test the same with appropriate dataset.\*** |
| **15** | **Implement dimensionality reduction using PCA.** |

**EXPERIMENT NO. 1**

**Introduction to Python Programming**

**Aim:**tto familiarize python programming

## **Python**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.
* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

## 

## **Python Syntax**

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\*Your Name*>python helloworld.py

Where "helloworld.py" is the name of your python file.

Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")

Hello, World!

## 

## **Python Indentation**

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

eg:

if 5 > 2:

print("Five is greater than two!")

Syntax Error:

if 5 > 2:

print("Five is greater than two!")

## **Creating a Commen**t

Comments starts with a #, and Python will ignore them:

### 

### **Example**

#This is a comment

print("Hello, World!")

## 

## **Creating Variables**

Variables do not need to be declared with any particular *type*, and can even change type after they have been set

### 

### **Example**

x = 5

y = "John"

print(x)

print(y)

## 

## **Variable Names**

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

**Legal variable names:**

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

The Python print() function is often used to output variables.

**Example:**

x = "Python is awesome"

print(x)

**Global variables** can be used by everyone, both inside of functions and outside.

Create a variable outside of a function, and use it inside the function

x = "awesome"

def myfunc():

print("Python is " + x)

myfunc()

## 

## **Built-in Data Types**

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

| Text Type: | str |
| --- | --- |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

# 

# **Python Built in Functions**

| **Function** | **Description** |
| --- | --- |
| [abs()](https://www.w3schools.com/python/ref_func_abs.asp) | Returns the absolute value of a number |
| [all()](https://www.w3schools.com/python/ref_func_all.asp) | Returns True if all items in an iterable object are true |
| [any()](https://www.w3schools.com/python/ref_func_any.asp) | Returns True if any item in an iterable object is true |
| [ascii()](https://www.w3schools.com/python/ref_func_ascii.asp) | Returns a readable version of an object. Replaces none-ascii characters with escape character |
| [bin()](https://www.w3schools.com/python/ref_func_bin.asp) | Returns the binary version of a number |
| [bool()](https://www.w3schools.com/python/ref_func_bool.asp) | Returns the boolean value of the specified object |
| [bytearray()](https://www.w3schools.com/python/ref_func_bytearray.asp) | Returns an array of bytes |
| [bytes()](https://www.w3schools.com/python/ref_func_bytes.asp) | Returns a bytes object |
| [callable()](https://www.w3schools.com/python/ref_func_callable.asp) | Returns True if the specified object is callable, otherwise False |
| [chr()](https://www.w3schools.com/python/ref_func_chr.asp) | Returns a character from the specified Unicode code. |
| classmethod() | Converts a method into a class method |
| [compile()](https://www.w3schools.com/python/ref_func_compile.asp) | Returns the specified source as an object, ready to be executed |
| [complex()](https://www.w3schools.com/python/ref_func_complex.asp) | Returns a complex number |
| [delattr()](https://www.w3schools.com/python/ref_func_delattr.asp) | Deletes the specified attribute (property or method) from the specified object |
| [dict()](https://www.w3schools.com/python/ref_func_dict.asp) | Returns a dictionary (Array) |
| [dir()](https://www.w3schools.com/python/ref_func_dir.asp) | Returns a list of the specified object's properties and methods |
| [divmod()](https://www.w3schools.com/python/ref_func_divmod.asp) | Returns the quotient and the remainder when argument1 is divided by argument2 |
| [enumerate()](https://www.w3schools.com/python/ref_func_enumerate.asp) | Takes a collection (e.g. a tuple) and returns it as an enumerate object |
| [eval()](https://www.w3schools.com/python/ref_func_eval.asp) | Evaluates and executes an expression |
| [exec()](https://www.w3schools.com/python/ref_func_exec.asp) | Executes the specified code (or object) |
| [filter()](https://www.w3schools.com/python/ref_func_filter.asp) | Use a filter function to exclude items in an iterable object |
| [float()](https://www.w3schools.com/python/ref_func_float.asp) | Returns a floating point number |
| [format()](https://www.w3schools.com/python/ref_func_format.asp) | Formats a specified value |
| [frozenset()](https://www.w3schools.com/python/ref_func_frozenset.asp) | Returns a frozenset object |
| [getattr()](https://www.w3schools.com/python/ref_func_getattr.asp) | Returns the value of the specified attribute (property or method) |
| [globals()](https://www.w3schools.com/python/ref_func_globals.asp) | Returns the current global symbol table as a dictionary |
| [hasattr()](https://www.w3schools.com/python/ref_func_hasattr.asp) | Returns True if the specified object has the specified attribute (property/method) |
| hash() | Returns the hash value of a specified object |
| help() | Executes the built-in help system |
| [hex()](https://www.w3schools.com/python/ref_func_hex.asp) | Converts a number into a hexadecimal value |
| [id()](https://www.w3schools.com/python/ref_func_id.asp) | Returns the id of an object |
| [input()](https://www.w3schools.com/python/ref_func_input.asp) | Allowing user input |
| [int()](https://www.w3schools.com/python/ref_func_int.asp) | Returns an integer number |
| [isinstance()](https://www.w3schools.com/python/ref_func_isinstance.asp) | Returns True if a specified object is an instance of a specified object |
| [issubclass()](https://www.w3schools.com/python/ref_func_issubclass.asp) | Returns True if a specified class is a subclass of a specified object |
| [iter()](https://www.w3schools.com/python/ref_func_iter.asp) | Returns an iterator object |
| [len()](https://www.w3schools.com/python/ref_func_len.asp) | Returns the length of an object |
| [list()](https://www.w3schools.com/python/ref_func_list.asp) | Returns a list |
| [locals()](https://www.w3schools.com/python/ref_func_locals.asp) | Returns an updated dictionary of the current local symbol table |
| [map()](https://www.w3schools.com/python/ref_func_map.asp) | Returns the specified iterator with the specified function applied to each item |
| [max()](https://www.w3schools.com/python/ref_func_max.asp) | Returns the largest item in an iterable |
| [memoryview()](https://www.w3schools.com/python/ref_func_memoryview.asp) | Returns a memory view object |
| [min()](https://www.w3schools.com/python/ref_func_min.asp) | Returns the smallest item in an iterable |
| [next()](https://www.w3schools.com/python/ref_func_next.asp) | Returns the next item in an iterable |
| [object()](https://www.w3schools.com/python/ref_func_object.asp) | Returns a new object |
| [oct()](https://www.w3schools.com/python/ref_func_oct.asp) | Converts a number into an octal |
| [open()](https://www.w3schools.com/python/ref_func_open.asp) | Opens a file and returns a file object |
| [ord()](https://www.w3schools.com/python/ref_func_ord.asp) | Convert an integer representing the Unicode of the specified character |
| [pow()](https://www.w3schools.com/python/ref_func_pow.asp) | Returns the value of x to the power of y |
| [print()](https://www.w3schools.com/python/ref_func_print.asp) | Prints to the standard output device |
| property() | Gets, sets, deletes a property |
| [range()](https://www.w3schools.com/python/ref_func_range.asp) | Returns a sequence of numbers, starting from 0 and increments by 1 (by default) |
| repr() | Returns a readable version of an object |
| [reversed()](https://www.w3schools.com/python/ref_func_reversed.asp) | Returns a reversed iterator |
| [round()](https://www.w3schools.com/python/ref_func_round.asp) | Rounds a numbers |
| [set()](https://www.w3schools.com/python/ref_func_set.asp) | Returns a new set object |
| [setattr()](https://www.w3schools.com/python/ref_func_setattr.asp) | Sets an attribute (property/method) of an object |
| [slice()](https://www.w3schools.com/python/ref_func_slice.asp) | Returns a slice object |
| [sorted()](https://www.w3schools.com/python/ref_func_sorted.asp) | Returns a sorted list |
| staticmethod() | Converts a method into a static method |
| [str()](https://www.w3schools.com/python/ref_func_str.asp) | Returns a string object |
| [sum()](https://www.w3schools.com/python/ref_func_sum.asp) | Sums the items of an iterator |
| [super()](https://www.w3schools.com/python/ref_func_super.asp) | Returns an object that represents the parent class |
| [tuple()](https://www.w3schools.com/python/ref_func_tuple.asp) | Returns a tuple |
| [type()](https://www.w3schools.com/python/ref_func_type.asp) | Returns the type of an object |
| [vars()](https://www.w3schools.com/python/ref_func_vars.asp) | Returns the \_\_dict\_\_ property of an object |
| [zip()](https://www.w3schools.com/python/ref_func_zip.asp) | Returns an iterator, from two or more iterators |

**EXPERIMENT NO. 2**

**Machine learning implementation tools**

**Aim**: to familiarize machine learning implementation tools such as jupyter notebook,spyder and pycharm

**Jupyter Notebook**

Jupyter Notebook is a notebook authoring application, under the [Project Jupyter](https://docs.jupyter.org/en/latest/) umbrella. Built on the power of the [computational notebook format](https://docs.jupyter.org/en/latest/#what-is-a-notebook), Jupyter Notebook offers fast, interactive new ways to prototype and explain your code, explore and visualize your data, and share your ideas with others.

Notebooks extend the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

A web application: A browser-based editing program for interactive authoring of computational notebooks which provides a fast interactive environment for prototyping and explaining code, exploring and visualizing data, and sharing ideas with others

Computational Notebook documents: A shareable document that combines computer code, plain language descriptions, data, rich visualizations like 3D models, charts, mathematics, graphs and figures, and interactive controls

### **Main features of the web application**

* In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
* The ability to execute code from the browser, with the results of computations attached to the code which generated them.
* Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the [matplotlib] library, can be included inline.
* In-browser editing for rich text using the [Markdown] markup language, which can provide commentary for the code, is not limited to plain text.
* The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by [MathJax](https://www.mathjax.org/).

### **Notebook documents**

Notebook documents contain the inputs and outputs of an interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally [JSON](https://en.wikipedia.org/wiki/JSON) files and are saved with the .ipynb extension.

Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the [nbconvert] command.

**Spyder**

**Spyder** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) cross-platform [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for scientific programming in the [Python language](https://en.wikipedia.org/wiki/Python_(programming_language)). Spyder integrates with a number of prominent packages in the scientific Python stack, including [NumPy](https://en.wikipedia.org/wiki/NumPy), [SciPy](https://en.wikipedia.org/wiki/SciPy), [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)), [IPython](https://en.wikipedia.org/wiki/IPython), [SymPy](https://en.wikipedia.org/wiki/SymPy) and [Cython](https://en.wikipedia.org/wiki/Cython), as well as other open-source software. It is released under the [MIT license](https://en.wikipedia.org/wiki/MIT_license).

Spyder is extensible with first-party and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, [Pylint](https://en.wikipedia.org/wiki/Pylint) and Rope. It is available cross-platform through [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)), on Windows, on macOS through [MacPorts](https://en.wikipedia.org/wiki/MacPorts), and on major Linux distributions such as [Arch Linux](https://en.wikipedia.org/wiki/Arch_Linux), [Debian](https://en.wikipedia.org/wiki/Debian), [Fedora](https://en.wikipedia.org/wiki/Fedora_(operating_system)), [Gentoo Linux](https://en.wikipedia.org/wiki/Gentoo_Linux), [openSUSE](https://en.wikipedia.org/wiki/OpenSUSE) and [Ubuntu](https://en.wikipedia.org/wiki/Ubuntu_(operating_system)).

Spyder uses [Qt](https://en.wikipedia.org/wiki/Qt_(software)) for its GUI and is designed to use either of the [PyQt](https://en.wikipedia.org/wiki/PyQt) or [PySide](https://en.wikipedia.org/wiki/PySide) Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

Features include:

* An editor with [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)), [code completion](https://en.wikipedia.org/wiki/Code_completion)
* Support for multiple [IPython](https://en.wikipedia.org/wiki/IPython) [consoles](https://en.wikipedia.org/wiki/Command-line_interface)
* The ability to explore and edit [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)) from a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface)
* A Help pane able to retrieve and render rich text [documentation](https://en.wikipedia.org/wiki/Application_programming_interface#Documentation) on functions, classes and methods automatically or on-demand
* A [debugger](https://en.wikipedia.org/wiki/Debugger) linked to IPdb, for step-by-step execution
* [Static code analysis](https://en.wikipedia.org/wiki/Static_program_analysis), powered by [Pylint](https://en.wikipedia.org/wiki/Pylint)
* A run-time [Profiler](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), to benchmark code
* Project support, allowing work on multiple development efforts simultaneously
* A built-in [file explorer](https://en.wikipedia.org/wiki/File_manager), for interacting with the filesystem and managing projects
* A "Find in Files" feature, allowing full [regular expression](https://en.wikipedia.org/wiki/Regular_expression) search over a specified scope
* An online help browser, allowing users to search and view Python and package documentation inside the IDE
* A [history log](https://en.wikipedia.org/wiki/Command_history), recording every user command entered in each console
* An internal console, allowing for introspection and control over Spyder's own operation

**PyCharm**

**PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used for programming in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control](https://en.wikipedia.org/wiki/Version_control) systems, and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)).It is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform_software), working on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux). PyCharm has a Professional Edition, released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software) and a Community Edition released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License). Coding assistance and [analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Code_completion), [syntax](https://en.wikipedia.org/wiki/Syntax_highlighting) and error highlighting, [linter](https://en.wikipedia.org/wiki/Linter_(software)) integration, and quick fixes

* Project and code navigation: specialized project views, file structure views and quick jumping between files, [classes](https://en.wikipedia.org/wiki/Class_(computer_programming)), [methods](https://en.wikipedia.org/wiki/Method_(computer_programming)) and usages
* Python [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring): including rename, extract method, introduce variable, introduce constant, pull up, push down and others
* Support for [web frameworks](https://en.wikipedia.org/wiki/Web_framework): Django, [web2py](https://en.wikipedia.org/wiki/Web2py) and [Flask](https://en.wikipedia.org/wiki/Flask_(web_framework))
* Integrated Python [debugger](https://en.wikipedia.org/wiki/Debugger)
* Integrated [unit testing](https://en.wikipedia.org/wiki/Unit_testing), with line-by-line [coverage](https://en.wikipedia.org/wiki/Code_coverage)
* [Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine) Python development
* Version control integration: unified user interface for [Mercurial](https://en.wikipedia.org/wiki/Mercurial), [Git](https://en.wikipedia.org/wiki/Git), [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion), [Perforce](https://en.wikipedia.org/wiki/Perforce) and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) with changelists and merge
* Scientific tools integration: integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.

**EXPERIMENT NO. 3**

**Union and intersection of two list**

**Aim**:The aim of the program is to find the union and intersection of two input lists, where the lists can contain elements of any data type.

**Algorithm:**

Step 1:Function Definition:Define a function union\_intersection that takes two lists (lst1 and lst2) as parameters.

Step 2:Union Calculation:Calculate the union of the two lists using sets:

Step 3:Convert lst1 and lst2 to sets using set(lst1) and set(lst2).

Step 4:Use the union operator | to get the union of the sets.

Step 5:Convert the result back to a list.

Step 6:Intersection Calculation:Calculate the intersection of the two lists using sets:

Step 7:Convert lst1 and lst2 to sets using set(lst1) and set(lst2).

Step 8:Use the intersection operator & to get the intersection of the sets.

Step 9:Convert the result back to a list.

Step 10:Return Result:Return a tuple containing the union and intersection lists.

**Python Code:**

def union\_intersection(lst1, lst2):

union = list(set(lst1) | set(lst2))

intersection = list(set(lst1) & set(lst2))

return union, intersection

nums1 = [1,2,3,4,5]

nums2 = [3,4,5,6,7,8]

print("Original lists:")

print(nums1)

print(nums2)

result = union\_intersection(nums1, nums2)

print("\nUnion of said two lists:")

print(result[0])

print("\nIntersection of said two lists:")

print(result[1])

colors1 = ["Red", "Green", "Blue"]

colors2 = ["Red", "White", "Pink", "Black"]

print("Original lists:")

print(colors1)

print(colors2)

result = union\_intersection(colors1, colors2)

print("\nUnion of said two lists:")

print(result[0])

print("\nIntersection of said two lists:")

print(result[1])

**EXPERIMENT NO. 4**

**Occurance of words in a given sentence**

**Aim** :design a python program to count the occurrence of each word in a sentence

**Algorithm:**

Step 1:Function Definition:Define a function word\_count that takes a string (str) as a parameter.

Step 2:Initialize Dictionary:Create an empty dictionary called counts to store the word counts.

Step 3:Split the String:Split the input string into a list of words using the split() method. This creates a list named words.

Step 4:Count Word Occurrences:Iterate through each word in the words list.

Step 5:Check if the word is already a key in the counts dictionary.

Step 6:If yes, increment the count for that word.

Step 7:If no, add the word as a key to the dictionary with a count of 1.

Step 8:Return Result:Return the counts dictionary containing word counts.

**Python Code:**

def word\_count(str):

counts = dict()

words = str.split()

for word in words:

if word in counts:

counts[word] += 1

else:

counts[word] = 1

return counts

print( word\_count('the quick brown fox jumps over the lazy dog.'))

**EXPERIMENT NO. 5**

**Matrix multiplication**

AIM:Write a Python program to multiply two matrices.

Algorithm:

Step 1:Function Definition:Define a function multiply\_matrices that takes two matrices (matrix1 and matrix2) as parameters.

Step 2:Check Compatibility:Retrieve the number of rows and columns for both matrices.

Step 3:Check if the number of columns in matrix1 is equal to the number of rows in matrix2 for valid matrix multiplication.

Step 4:If not, return an error message indicating that matrix multiplication is not possible.

Step 5:Initialize Result Matrix:If matrix multiplication is possible, initialize a result matrix (result) with dimensions rows1 x cols2 filled with zeros.

Step 6:Matrix Multiplication:Use nested loops to iterate through each element of the result matrix.

Step 7:Perform the dot product of the corresponding row from matrix1 and column from matrix2.

Step 8Accumulate the products to get the result for each element in the result matrix.

Step 9Return Result:Return the resulting matrix.

**Python code**

def multiply\_matrices(matrix1, matrix2):

rows1 = len(matrix1)

cols1 = len(matrix1[0])

rows2 = len(matrix2)

cols2 = len(matrix2[0])

if cols1 != rows2:

return "Matrix multiplication not possible. The number of columns in the first matrix must be equal to the number of rows in the second matrix."

result = [[0 for \_ in range(cols2)] for \_ in range(rows1)]

for i in range(rows1):

for j in range(cols2):

for k in range(cols1):

result[i][j] += matrix1[i][k] \* matrix2[k][j]

return result

matrix1 = [

[1, 2, 3],

[4, 5, 6]

]

matrix2 = [

[7, 8],

[9, 10],

[11, 12]

]

result\_matrix = multiply\_matrices(matrix1, matrix2)

if isinstance(result\_matrix, str):

print(result\_matrix)

else:

for row in result\_matrix:

print(row)

**Python code using numpy**

import numpy as np

matrix1 = np.array([

[1, 2, 3],

[4, 5, 6],

])

matrix2 = np.array([

[7, 8],

[9, 10],

[11, 12],

])

result\_matrix = np.dot(matrix1, matrix2)

print("Result Matrix:")

print(result\_matrix)

**EXPERIMENT NO. 6**

**frequent words in a text file**

Aim Write a Python program to find the most frequent words in a text file

**Steps**

1. Read the text from the file.
2. Tokenize the text into words.
3. Count the frequency of each word.
4. Sort the words by frequency.
5. Display the most frequent words.

Algorithm:

Step 1: Open File:Open the file "gfg.txt" in read mode using open("gfg.txt", "r") and assign it to the variable file.

Step 2:Initialize Variables:Initialize variables frequent\_word and frequency to store the most repeated word and its frequency.

Step 3:Initialize an empty list words to store all the words from the file.

Step 4:Read Lines and Extract Words:Iterate over each line in the file using a loop.

Step 5:Convert each line to lowercase, remove punctuation (comma and period), and split it into words.

Step 6:Append each word to the words list.

Step 7:Count Word Frequencies:Iterate over the words list.

Step 8:For each word, count its occurrence in the list and update the variables frequent\_word and frequency if a higher count is found.

Step 9:Print Result:Print the most repeated word (frequent\_word) and its frequency (frequency).

Step 10:Close File:Close the file using file.close().

**Python code**

file = open("gfg.txt","r")

frequent\_word = ""

frequency = 0

words = []

for line in file:

line\_word = line.lower().replace(',','').replace('.','').split(" ");

for w in line\_word:

words.append(w);

for i in range(0, len(words)):

count = 1;

for j in range(i+1, len(words)):

if(words[i] == words[j]):

count = count + 1;

if(count > frequency):

frequency = count;

frequent\_word = words[i];

print("Most repeated word: " + frequent\_word)

print("Frequency: " + str(frequency))

file.close();

**EXPERIMENT NO. 7**

**Single, Multi variable and Polynomial Regression**

**Aim** Implement and demonstrate Single, Multi variable and Polynomial Regression for a given set of training data stored in a .CSV file and evaluate the accuracy

**Algorithm**

**Simple Linear Regression:**

Step1:Load Data:Use pandas to read the training data from the 'training\_data.csv' file.

Step 2:Prepare Features and Target:Extract the feature 'X' and the target 'Y' from the data.

Step 3:Build and Train Model:Create a Linear Regression model.

Step 4:Fit the model using the feature 'X' and target 'Y'.

Step 5:Predict and Evaluate:Predict the target 'Y' using the trained model.

Step 6:Calculate Mean Squared Error (MSE) and R-squared for evaluation.

Step 7:Visualize:Scatter plot the original data points.

sTEP 7:Plot the regression line.

Step 8:Display Results:Show the scatter plot and print MSE and R-squared.

**Multivariable Linear Regression:**

Step 1:Load Data:Use pandas to read the training data from the 'training\_data.csv' file.

Step 2:Prepare Features and Target:Extract all features (excluding 'Y') and the target 'Y' from the data.

Step 3:Build and Train Model:Create a Linear Regression model.

Step 4:Fit the model using all features and the target 'Y'.

Step 5:Predict and Evaluate:Predict the target 'Y' using the trained model.

Step 6:Calculate Mean Squared Error (MSE) and R-squared for evaluation.

Step 7Display Results:Print MSE and R-squared.

**Polynomial Regression:**

Step 1:Load Data:Use pandas to read the training data from the 'training\_data.csv' file.

Step 2:Prepare Features and Target:Extract the feature 'X' and the target 'Y' from the data.

Step 3:Polynomial Features:Use PolynomialFeatures to transform the feature 'X' into polynomial features.

Step 4:Build and Train Model:Create a Linear Regression model.

Step 5:Fit the model using the polynomial features and the target 'Y'.

Step 6:Predict and Evaluate:Predict the target 'Y' using the trained model.

Step 7:Calculate Mean Squared Error (MSE) and R-squared for evaluation.

Step 8:Visualize:Scatter plot the original data points.

Step 9:Plot the polynomial regression curve.

Step 10:Display Results:Show the scatter plot and print MSE and R-squared.

**Python code**

**SINGLE LINEAR REGRESSION**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

data = pd.read\_csv('training\_data.csv')

X = data['X'].values.reshape(-1, 1)

Y = data['Y'].values

model = LinearRegression()

model.fit(X, Y)

Y\_pred = model.predict(X)

mse = mean\_squared\_error(Y, Y\_pred)

r2 = r2\_score(Y, Y\_pred)

plt.scatter(X, Y, label='Data')

plt.plot(X, Y\_pred, color='red', label='Regression Line')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Single Linear Regression')

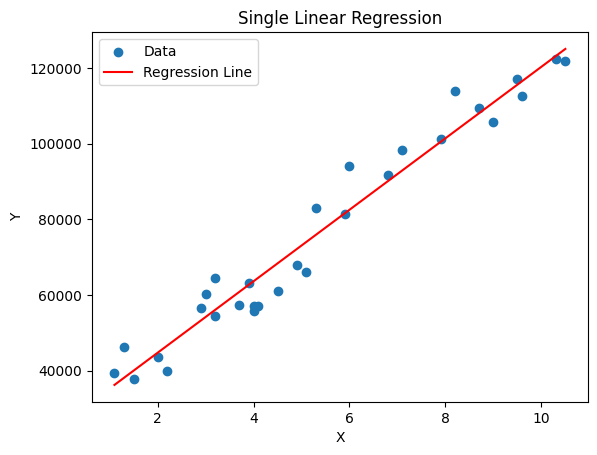
plt.legend()

plt.show()

print(f"Mean Squared Error: {mse}")

print(f"R-squared (Accuracy): {r2}")

**OUTPUT FOR LINEAR REGRESSION**

****

**Mean Squared Error: 31270951.722280968**

**R-squared (Accuracy): 0.9569566641435086**

**MULTIVARIABLE LINEAR REGRESSION**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

data = pd.read\_csv('training\_data.csv')

X = data.drop('Y', axis=1)

Y = data['Y']

model = LinearRegression()

model.fit(X, Y)

Y\_pred = model.predict(X)

mse = mean\_squared\_error(Y, Y\_pred)

r2 = r2\_score(Y, Y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared (Accuracy): {r2}")

**OUTPUT FOR MULTIVARIABLE LINEAR REGRESSION**

**Mean Squared Error: 31270951.722280968**

**R-squared (Accuracy): 0.956956664143508**

**POLYNOMIAL REGRESSION**

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

data = pd.read\_csv('training\_data.csv')

X = data['X'].values.reshape(-1, 1)

Y = data['Y'].values

poly = PolynomialFeatures(degree=2) # You can adjust the degree

X\_poly = poly.fit\_transform(X)

model = LinearRegression()

model.fit(X\_poly, Y)

Y\_pred = model.predict(X\_poly)

X\_sort, Y\_sort = zip(\*sorted(zip(X, Y\_pred))

mse = mean\_squared\_error(Y, Y\_pred)

r2 = r2\_score(Y, Y\_pred)

plt.scatter(X, Y, label='Data')

plt.plot(X\_sort, Y\_sort, color='red', label='Polynomial Regression')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Polynomial Regression')

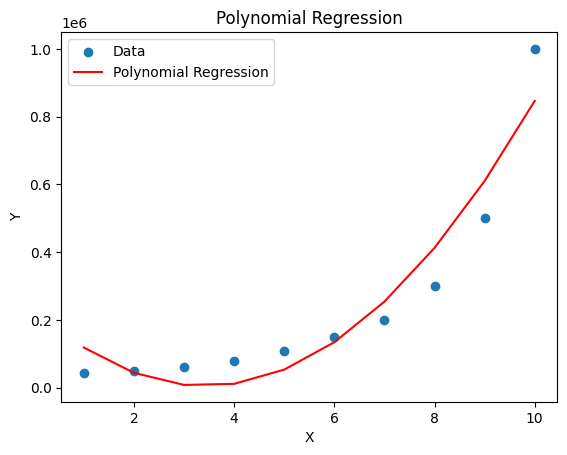
plt.legend()

plt.show()

print(f"Mean Squared Error: {mse}")

print(f"R-squared (Accuracy): {r2}")

**OUTPUT FOR POLYNOMIAL REGRESSION**

****

**Mean Squared Error: 6758833333.333338**

**R-squared (Accuracy): 0.9162082221443942**

**EXPERIMENT NO. 8**

**Logistic Regression**

**Aim** Implement a python program to perform logistic regression on a dataset

**Algorithm:**

Step 1:Load Dataset:Use pandas to read the dataset from the "User\_Data.csv" file.

Step 2:Extract Features and Target:Extract the features (columns 2 and 3) into x and the target variable (column 4) into y.

Step 3:Split Dataset:Split the dataset into training and testing sets using train\_test\_split from sklearn.model\_selection.

Step 4:Feature Scaling:Standardize the features using StandardScaler from sklearn.preprocessing.

Step 5:Print Scaled Training Data:Print the first 10 rows of the scaled training data.

Step 6:Build Logistic Regression Model:Create a logistic regression classifier using LogisticRegression from sklearn.linear\_model.

Step 7;Fit the classifier on the scaled training data.

Step 8 : Make Predictions:Use the trained classifier to make predictions on the test set.

Step 9:Evaluate Model:Compute the confusion matrix and print it.

Step 10:Compute and print the accuracy score of the model using accuracy\_score from sklearn.metrics.

Step 11:Visualize Results:Create a meshgrid for visualizing the decision boundary.

Step 12:Plot the decision regions using contourf.

Step 13:Scatter plot the test set points with different colors for different classes.

Step 14:Display Plot:Display the plot with appropriate labels and legend.

**Python code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset = pd.read\_csv("User\_Data.csv")

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

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

xtrain, xtest, ytrain, ytest = train\_test\_split( x, y, test\_size=0.25, random\_state=0)

from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(xtrain)

xtest = sc\_x.transform(xtest)

print (xtrain[0:10, :])

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(xtrain, ytrain)

y\_pred = classifier.predict(xtest)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(ytest, y\_pred)

print ("Confusion Matrix : \n", cm)

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(ytest, y\_pred))

from matplotlib.colors import ListedColormap

X\_set, y\_set = xtest, ytest

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),np.arange(start = X\_set[:, 1].min() - 1,stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red',

'green'))(i), label = j)

plt.title('Classifier (Test set)')

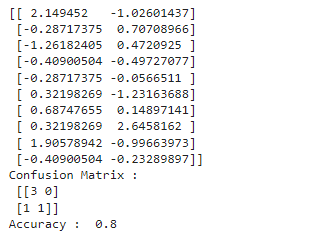
plt.xlabel('Age')

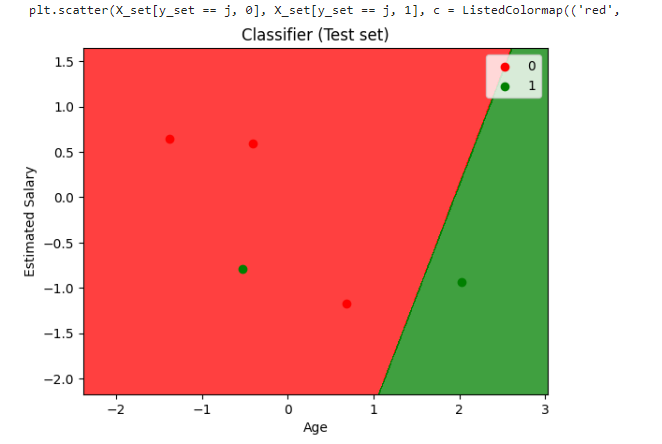
plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**OUTPUT**





**Dataset**

| user ID | Gender | Age | EstimatedSalary | Purchased |
| --- | --- | --- | --- | --- |
| 15624510 | Male | 19 | 19000 | 0 |
| 15810944 | Male | 35 | 20000 | 0 |
| 15668575 | Female | 26 | 43000 | 0 |
| 15603246 | Female | 27 | 57000 | 0 |
| 15804002 | Male | 19 | 76000 | 0 |
| 15728773 | Male | 27 | 58000 | 0 |
| 15598044 | Female | 27 | 84000 | 0 |
| 15694829 | Female | 32 | 150000 | 0 |
| 15600575 | Male | 25 | 33000 | 1 |
| 15727311 | Female | 35 | 65000 | 0 |
| 15570769 | Female | 26 | 80000 | 0 |
| 15606274 | female | 26 | 52000 | 0 |
| 15746139 | Male | 20 | 86000 | 0 |
| 15704987 | Male | 32 | 18000 | 0 |
| 15628972 | Male | 18 | 82000 | 0 |
| 15697686 | Male | 29 | 80000 | 0 |
| 15733883 | Male | 47 | 25000 | 1 |
| 15617482 | Male | 45 | 26000 | 1 |
| 15704583 | Male | 46 | 28000 | 1 |

**EXPERIMENT NO. 9**

**Naive Bayes**

**Aim :Write a Python program to implement Naive Bayes classifier and calculate the accuracy, precision, and recall for your data set.**

**Algorithm**

Step 1:Import necessary libraries:train\_test\_split for splitting the dataset.

GaussianNB for the Gaussian Naive Bayes classifier.

accuracy\_score, precision\_score, and recall\_score for model evaluation.

load\_iris for loading the Iris dataset.

Step 2:Load and Split Dataset:Load the Iris dataset using load\_iris.

Step 3:Extract features X and target y.

Step 4:Split the dataset into training and testing sets using train\_test\_split.

Step 5:Create and Train Naive Bayes Classifier:

Step 6:Create a Gaussian Naive Bayes classifier using GaussianNB().

Step 7:Fit the classifier using the training data (X\_train, y\_train).

Make Predictions:Predict the target values for the test set using the trained classifier.

Step 8:Evaluate Model:

Calculate accuracy, precision, and recall using appropriate metrics:

accuracy\_score for overall accuracy.

precision\_score for precision.

recall\_score for recall.

Step 9:Print Results:Print the calculated accuracy, precision, and recall.

**Python code**

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

nb\_classifier = GaussianNB()

nb\_classifier.fit(X\_train, y\_train)

y\_pred = nb\_classifier.predict(X\_test)

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

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

print(f'Accuracy: {accuracy:.2f}')

print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

**OUTPUT**



**EXPERIMENT NO. 10**

**Decision Trees**

Aim :write a python program to demonstrate the working of the decision tree based id3 algorithm. use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample

Algorithm

Algorithm:

Step 1:Import Libraries:

Step 2:Load and Explore Dataset:

step 3:Read the dataset from 'car\_evaluation.csv' using pd.read\_csv.

Step 4:Set column names using colnames.

Step 5:Display a count plot for the 'decision' variable.

Step 6:Preprocess Data

Step 7:Use LabelEncoder to encode categorical variables in the dataset.

Step 8:Display the value counts for the 'decision' column.

Step 9:Split Dataset

Step 10:Build and Train Decision Tree Model.

Step 11:Fit the model using the training data.

Step 12:Predict the 'decision' variable for the test set.

Step 13:Evaluate Model:Compute the confusion matrix using confusion\_matrix.

Step 14:Display the confusion matrix using ConfusionMatrixDisplay.

PYTHON CODE

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sklearn

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

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report,confusion\_matrix

colnames=['Buying\_price', 'maint\_cost', 'doors', 'persons','lug\_boot','safety','decision']

data = pd.read\_csv('car\_evaluation.csv', names=colnames, header=None)

plt.figure(figsize=(5,5))

sns.countplot(x='decision',data=data)

plt.title('Count plot for decision')

data.decision.replace('vgood','acc',inplace=True)

data.decision.replace('good','acc',inplace=True)

data['decision'].value\_counts()

new\_data=data.apply(LabelEncoder().fit\_transform)

new\_data

x=new\_data.drop(['decision'], axis=1)

y =new\_data['decision']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 42)

dt =DecisionTreeClassifier(criterion="entropy")

dt.fit(x\_train,y\_train)

dt\_pred=dt.predict(x\_test)

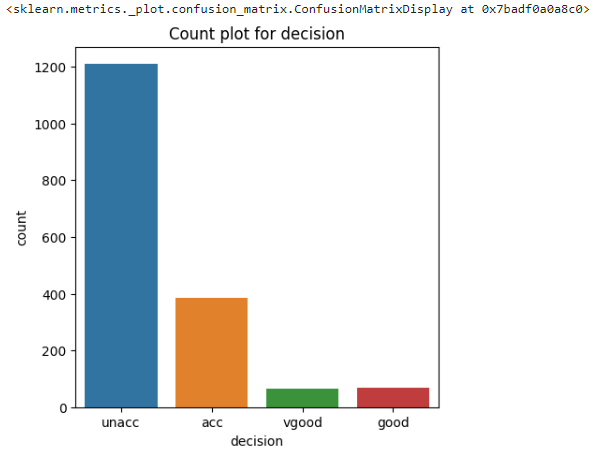
cm=confusion\_matrix(y\_test,dt\_pred)

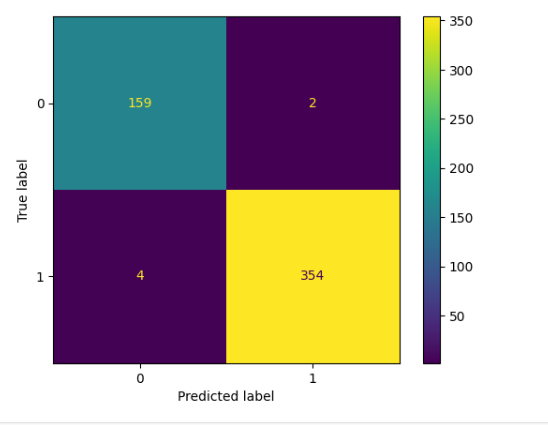
cm\_display = sklearn.metrics.ConfusionMatrixDisplay(confusion\_matrix = cm)

cm\_display.plot()

plt.show()

**OUTPUT**





**EXPERIMENT NO. 11**

**SUPPORT VECTOR MACHINE**

**Aim:**

Assuming Use a Support Vector Machine classifier to classify a set of data and evaluate the accuracy

**Algorithm:**

Step 1:Import necessary libraries:

Step 2:Load and Split Dataset:

Step 3:Load the dataset.Split the dataset into training and testing sets using train\_test\_split.

Step 4:Build and Train SVM Model:

Step 5:Create a Support Vector Machine classifier (SVC) with a linear kernel.

Step 6:Fit the classifier using the training data.

Step 7:Make Predictions:Predict the target values for the test set using the trained classifier.

Step 8:Evaluate Model:Calculate accuracy using accuracy\_score.

Step 9:Generate a confusion matrix using confusion\_matrix.

Step 10:Display Results:Print the accuracy of the SVM model.

Step 11:Print the confusion matrix.

**PYTHON CODE**import numpy as np

from sklearn.datasets import load\_digits

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

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, accuracy\_score

dataset = load\_digits()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(dataset.data, dataset.target, test\_size=0.30,

random\_state=4)

Classifier = SVC(kernel="linear")

Classifier.fit(x\_train, y\_train)

y\_pred = Classifier.predict(x\_test)

accuracy = accuracy\_score(y\_test,y\_pred)\*100

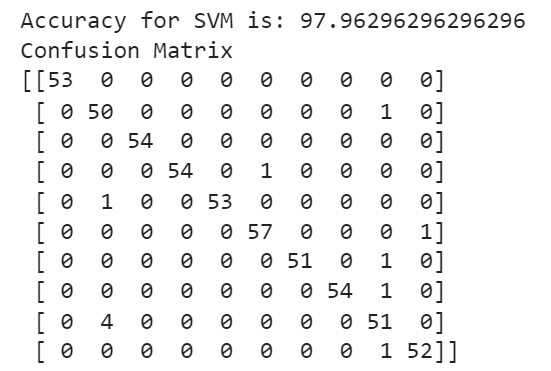
confusion\_mat = confusion\_matrix(y\_test,y\_pred)

print("Accuracy for SVM is:",accuracy)

print("Confusion Matrix")

print(confusion\_mat)

**OUTPUT**



**EXPERIMENT NO. 12**

**K-Nearest Neighbor algorithm**

Aim:Implement K-Nearest Neighbor algorithm to classify any dataset.

Algorithm:

Step 1: Import Libraries:

Step 2:Define Data and Classes:

Step 3: Define the input features (x and y) and corresponding classes (classes).

Step 4: Combine Data:Combine the x and y coordinates into a list of tuples data.

Step 5: Create K-Nearest Neighbors (KNN) Model:

Step 6: Create a KNeighborsClassifier with n\_neighbors=1 (1 nearest neighbor).

Step 7: Fit the model using the combined data and classes.

Step 8: Predict New Point.

Step 9: Predict the class of the new point using the trained KNN model.

Step 10:Visualization:Plot the existing points and the new point using plt.scatter.

Step 11:Display the class of the new point as text near the point.

Step 12:Show the plot.

Python code

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

x=[4,5,10,4,3,11,14,6,10,12]

y=[21,19,24,17,16,25,24,22,21,21]

classes=[0,0,1,0,0,1,1,0,1,1]

data=list(zip(x,y))

knn=KNeighborsClassifier(n\_neighbors=1)

knn.fit(data,classes)

new\_x=8

new\_y=21

new\_point=[(new\_x,new\_y)]

prediction=knn.predict(new\_point)

print(prediction)

#print(prediction=knn.predict(new\_point))

plt.scatter(x+[new\_x],y+[new\_y],c=classes+[prediction[0]])

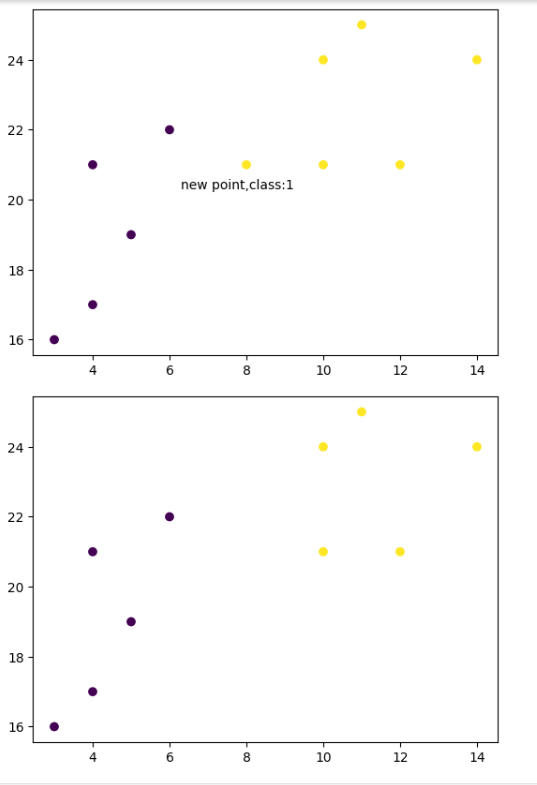
plt.text(x=new\_x-1.7,y=new\_y-0.7,s=f"new point,class:{prediction[0]}")

plt.show()

plt.scatter(x,y,c=classes)

plt.show()

**OUTPUT**



**EXPERIMENT NO. 13**

**K-Means Clustering**

Aim:Implement K-Means Clustering using any given dataset

Algorithm:

Step 1: Import Libraries:

Step 2: Load Iris Dataset:

Step 3: Load the Iris dataset using load\_iris and return features X and labels y.

Step 4: Apply K-Means Clustering:

Step 5: Create a KMeans model with n\_clusters=3 and fit it to the dataset.

Step 6: Obtain cluster assignments for each data point.

Step 7: Visualize Clusters (Petal Features):

Step 8: Plot a scatter plot for petal length against petal width (X[:,0] vs X[:,1]).

Step 9: Color points based on cluster assignments.

Step 10: Visualize Clusters (Sepal Features):

Step 11: Plot a scatter plot for sepal length against sepal width (X[:,2] vs X[:,3]).

Step 12: Color points based on cluster assignments.

Step 13: Plot cluster centers as red triangles.

Step 14: Show Plots:Display the plots.

Python code

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.cm as cm

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

X, y = load\_iris(return\_X\_y=True)

kmeans = KMeans(n\_clusters = 3, random\_state = 2)

kmeans.fit(X)

pred = kmeans.fit\_predict(X)

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.scatter(X[:,0],X[:,1],c = pred, cmap=cm.Accent)

plt.grid(True)

for center in kmeans.cluster\_centers\_:

center = center[:2]

plt.scatter(center[0],center[1],marker = '^',c = 'red')

plt.xlabel("petal length (cm)")

plt.ylabel("petal width (cm)")

plt.subplot(1,2,2)

plt.scatter(X[:,2],X[:,3],c = pred, cmap=cm.Accent)

plt.grid(True)

for center in kmeans.cluster\_centers\_:

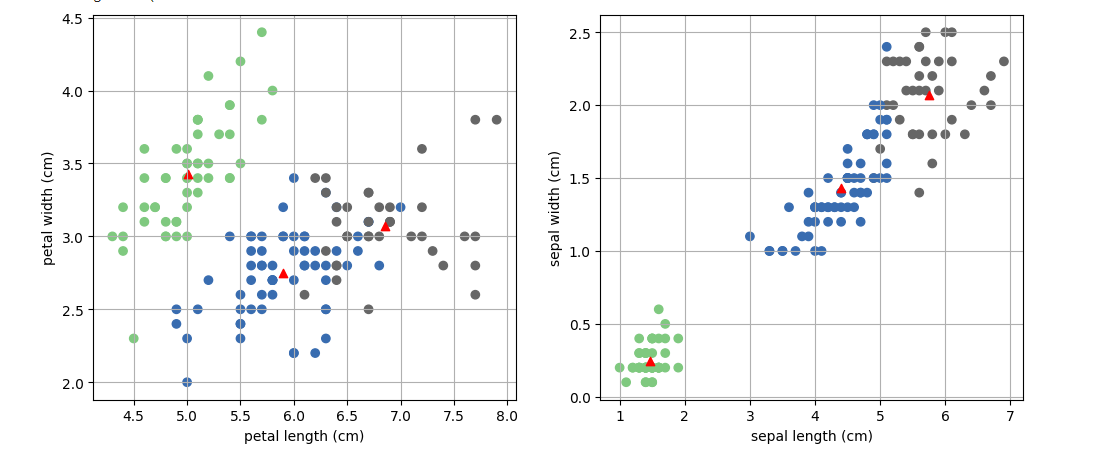
center = center[2:4]

plt.scatter(center[0],center[1],marker = '^',c = 'red')

plt.xlabel("sepal length (cm)")

plt.ylabel("sepal width (cm)")

plt.show()

**OUTPUT**  


**EXPERIMENT NO. 14**

**BUILD ANN USING BACKPROPAGATION ALGORITHM**

Aim:Build an Artificial Neural Network using Backpropagation algorithm and test the

same with appropriate dataset.

Algorithm

Algorithm:

Import Libraries:

Step 1:Load and Explore Dataset:

Step 2:Read the dataset from 'Churn\_Modelling.csv' using pd.read\_csv.

Step 3:Preprocess Data:

Step 4:Perform one-hot encoding for categorical variables 'Geography' and 'Gender'.

Step 5:Drop original categorical columns and concatenate the one-hot encoded columns.

step 6:Split the dataset into training and testing sets using train\_test\_split.

Step 7:Standardize the features using StandardScaler.

Step 8:Build Neural Network:

Step 9:Compile the model using binary crossentropy loss, Adam optimizer, and accuracy metric.

Step 10:Train Model:Fit the model on the training data with validation split, batch size, and epochs specified.

Step 11:Make Predictions:Predict the target variable for the test set.

Step 12:Evaluate Model:Compute confusion matrix, accuracy, and classification report using scikit-learn metrics.

Step 13:Visualize Training History:Plot the training accuracy and validation accuracy over epochs.

**python code**

import numpy as np

import pandas as pd

import tensorflow as tf

import keras

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LeakyReLU,PReLU,ELU

from keras. layers import Dropout

dataset= pd.read\_csv('Churn\_Modelling.csv')

dataset.head()

X= dataset.iloc[:,3:-1]

y=dataset.iloc[:,-1]

X.head()

geography=pd.get\_dummies(X["Geography"],drop\_first=True)

gender= pd.get\_dummies(X['Gender'],drop\_first=True)

X=pd.concat([X,geography,gender],axis=1)

X.drop(['Geography','Gender'],axis=1,inplace=True)

X.head()

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=0)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

X\_train=sc.fit\_transform(X\_train)

X\_test=sc.transform(X\_test)

classifier= Sequential()

classifier.add(Dense(units=6,kernel\_initializer='he\_uniform',activation='relu',input\_dim=11))

classifier.add(Dense(units=6,kernel\_initializer='he\_uniform',activation='relu'))

classifier.add(Dense(units=1,kernel\_initializer='glorot\_uniform',activation='sigmoid'))

classifier.summary()

classifier.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

model\_history=classifier.fit(X\_train,y\_train,validation\_split=0.33,batch\_size=10,epochs=100)

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

y\_pred= (y\_pred>0.5)

y\_pred

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

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

print(cm)

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

print('The accuracy of the model is',accuracy)

cl\_report = classification\_report(y\_test,y\_pred)

print(cl\_report)

print(model\_history.history.keys())

plt.plot(model\_history.history['accuracy'])

plt.plot(model\_history.history['val\_accuracy'])

plt.title('model accuracy')

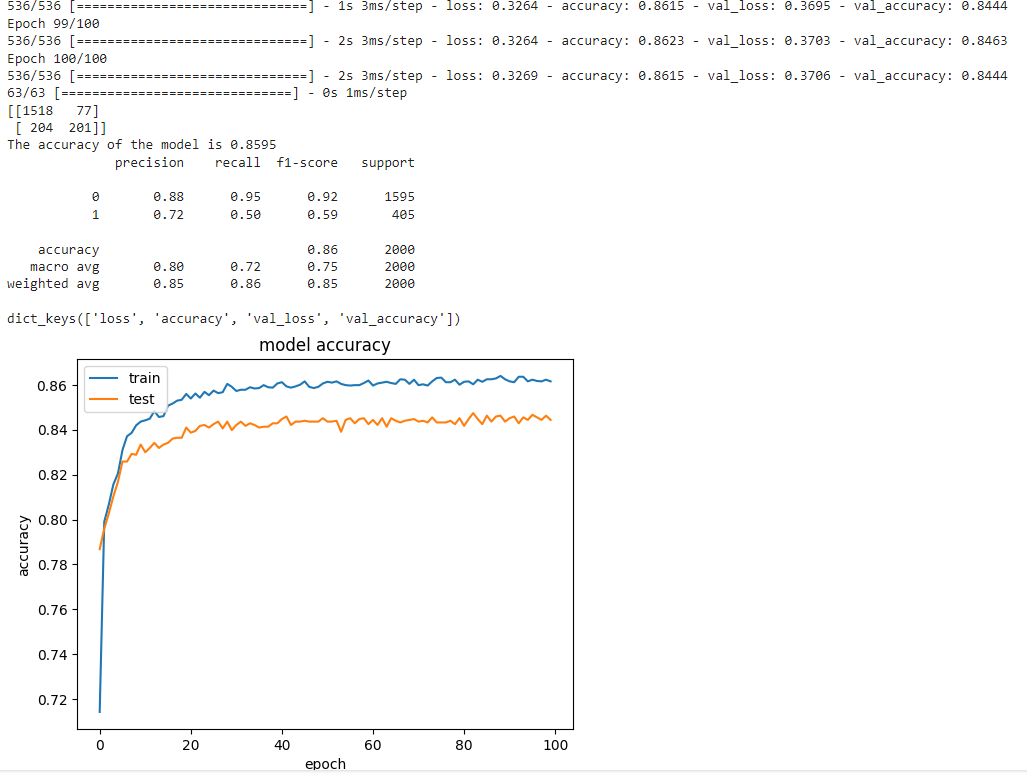
plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

**OUTPUT**

****

**EXPERIMENT NO. 14**

**PRINCIPAL COMPONENT ANALYSIS**

**Aim**:. Implement dimensionality reduction using PCA

Algorithm

Algorithm:

Step 1:Import Libraries:

Step 2:Generate Random Data:Create a random dataset X with shape (100, 5) for demonstration purposes.

step 3:Apply PCA:Fit a PCA model on the dataset using PCA().fit(X).

Step 4:Calculate Cumulative Variance Explained:Calculate the cumulative sum of the explained variance ratio to determine the cumulative variance explained.

Step 5:Plot Scree Plot:Plot the scree plot to visualize the variance explained by each principal component.

Step 6:X-axis: Principal Component Index.

Step 7:Y-axis: Explained Variance Ratio.

Step 8:Display the scree plot.

PYTHON CODE

from sklearn.decomposition import PCA

import numpy as np

import matplotlib.pyplot as plt

X = np.random.rand(100,5)

pca = PCA().fit(X)

var\_exp = pca.explained\_variance\_ratio\_

cum\_var\_exp = np.cumsum(var\_exp)

plt.plot(range(1,len(var\_exp)+1), pca.explained\_variance\_ratio\_, 'ro-',

linewidth=2)

plt.title('Scree Plot')

plt.xlabel('Principal Component')

plt.ylabel('Variance Explained')

plt.show()

**OUTPUT**

