COURSE CODE: CSAI2018_3

ELEMENTS OF AI/ML

SUBMITTED BY:

RAJVARDHAN SINGH YADAV (590015570)

COURSE: BCA (B4)

SUBMITTED TO:

Dr. SONALI VYAS

EXPERIMENT – 1

1- WAP for currency conversion where user makes selection

CODE:

p1.py

AIML > Experiments > Exp-1 > p1.py > ...

```python
def currency_converter():
    print("Currency Converter")
    print("Available currencies: INR, USD, EUR, GBP")
    rates = {
        "INR": {"USD": 0.012, "EUR": 0.011, "GBP": 0.0096},
        "USD": {"INR": 83.0, "EUR": 0.92, "GBP": 0.80},
        "EUR": {"INR": 90.0, "USD": 1.09, "GBP": 0.87},
        "GBP": {"INR": 104.0, "USD": 1.25, "EUR": 1.15}
    }

    from_currency = input("Enter from currency (INR/USD/EUR/GBP): ").upper()
    to_currency = input("Enter to currency (INR/USD/EUR/GBP): ").upper()
    amount = float(input("Enter amount: "))

    if from_currency in rates and to_currency in rates[from_currency]:
        converted = amount * rates[from_currency][to_currency]
        print(f"{amount} {from_currency} = {converted:.2f} {to_currency}")
    else:
        print("Conversion not available!")

currency_converter()
```

OUTPUT:

```
Currency Converter
Available currencies: INR, USD, EUR, GBP
Enter from currency (INR/USD/EUR/GBP): INR
Enter to currency (INR/USD/EUR/GBP): USD
Enter amount: 100
100.0 INR = 1.20 USD
```

2- WAP by using numpy perform binary search for searching names of student

CODE:

p2.py

AIML > Experiments > Exp-1 > p2.py > ...

```python
import numpy as np

def binary_search_students():
    students = []
    n = int(input("Enter number of students: "))
    for i in range(n):
        name = input(f"Enter student {i+1} name: ")
        students.append(name)

    # Sort the names before applying binary search
    students = np.array(sorted(students))
    print("Sorted student names:", students)

    target = input("Enter the student name to search: ")

    # Perform binary search
    index = np.searchsorted(students, target)

    if index < len(students) and students[index] == target:
        print(f"{target} found at position {index}")
    else:
        print(f"{target} not found!")

binary_search_students()
```

OUTPUT:

```
Enter number of students: 5
Enter student 1 name: Suyash
Enter student 2 name: RAj
Enter student 3 name: Adheesh
Enter student 4 name: Anurag
Enter student 5 name: Jazz
Sorted student names: ['Adheesh' 'Anurag' 'Jazz' 'RAj' 'Suyash']
Enter the student name to search: Vaibhav
Vaibhav not found!
```

3- WAP and perform following array

a- searching and sorting of array

b- add and remove item from array

CODE:

AIML > Experiments > Exp-1 > p3.py > ...

```python
from array import array

def array_operations():
    arr = array('i', [])
    n = int(input("Enter number of elements in array: "))
    for i in range(n):
        val = int(input(f"Enter element {i+1}: "))
        arr.append(val)

    print("Original Array:", arr.tolist())

    # Searching
    search_val = int(input("Enter value to search: "))
    if search_val in arr:
        print(f"{search_val} found at index {arr.index(search_val)}")
    else:
        print("Value not found!")

    # Sorting
    sorted_arr = sorted(arr)
    print("Sorted Array:", sorted_arr)

    # Adding item
    new_val = int(input("Enter value to add: "))
    arr.append(new_val)
    print("Array after adding:", arr.tolist())

    # Removing item
    remove_val = int(input("Enter value to remove: "))
    if remove_val in arr:
        arr.remove(remove_val)
        print("Array after removing:", arr.tolist())
    else:
        print("Value not found, cannot remove.")

array_operations()
```

OUTPUT:

```
Enter number of elements in array: 5
Enter element 1: 23
Enter element 2: 45
Enter element 3: 67
Enter element 4: 54
Enter element 5: 11
Original Array: [23, 45, 67, 54, 11]
Enter value to search: 45
45 found at index 1
Sorted Array: [11, 23, 45, 54, 67]
Enter value to add:
```

4- WAP and perform following string operation

a- reverse of array

b- concatenation of 2 strings

c- upper and lowercase conversion

d- slicing of a string

CODE:

AIML > Experiments > Exp-1 > p4.py > ...

```python
def string_operations():
    # a- Reverse of string
    s = input("Enter a string: ")
    print("Reversed String:", s[::-1])

    # b- Concatenation of 2 strings
    s1 = input("Enter first string: ")
    s2 = input("Enter second string: ")
    print("Concatenated String:", s1 + s2)

    # c- Upper and lowercase conversion
    print("Uppercase:", s1.upper())
    print("Lowercase:", s1.lower())

    # d- Slicing of a string
    s3 = input("Enter string for slicing: ")
    start = int(input("Enter start index: "))
    end = int(input("Enter end index: "))
    print("Sliced String:", s3[start:end])

string_operations()
```

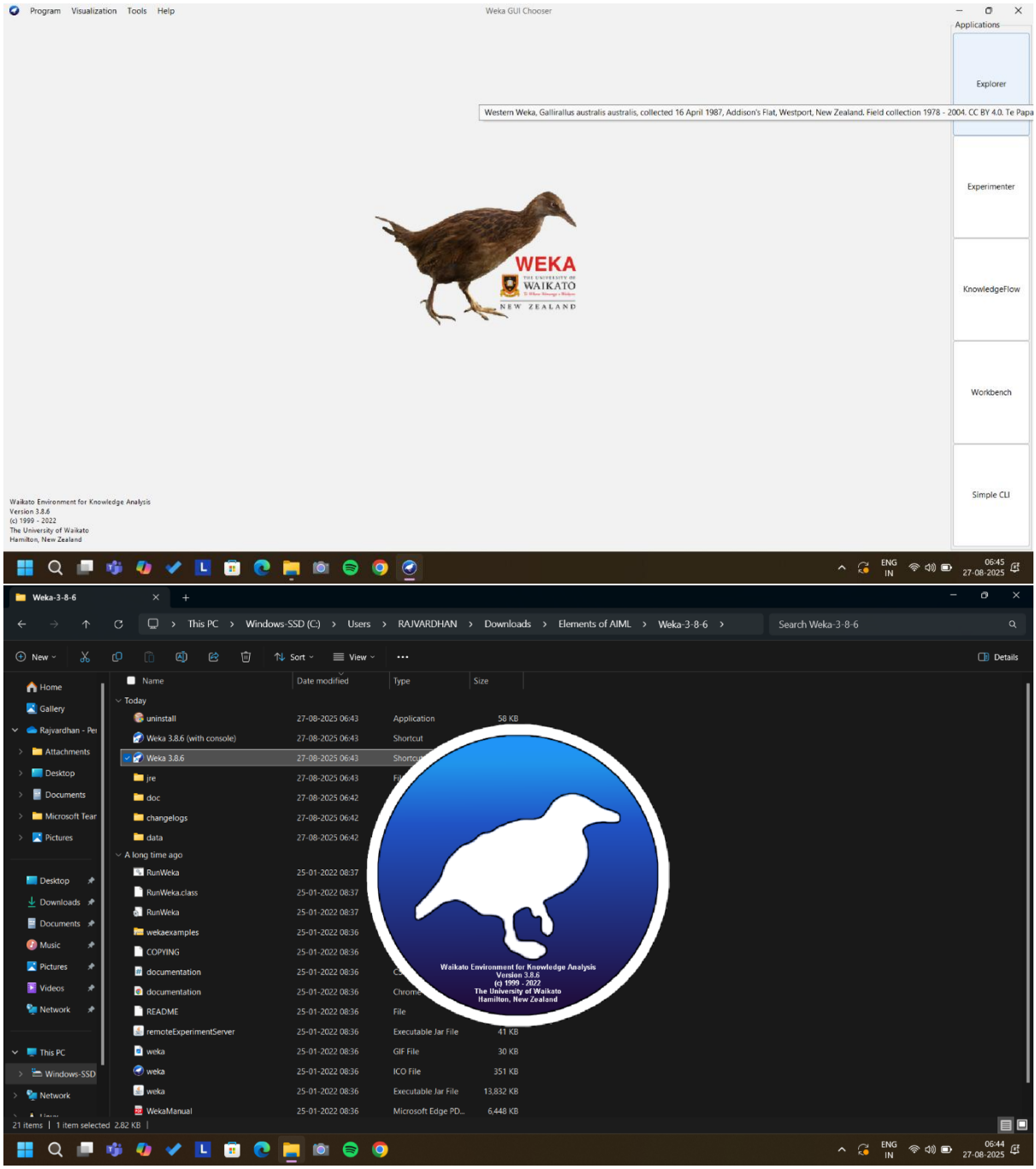
OUTPUT:

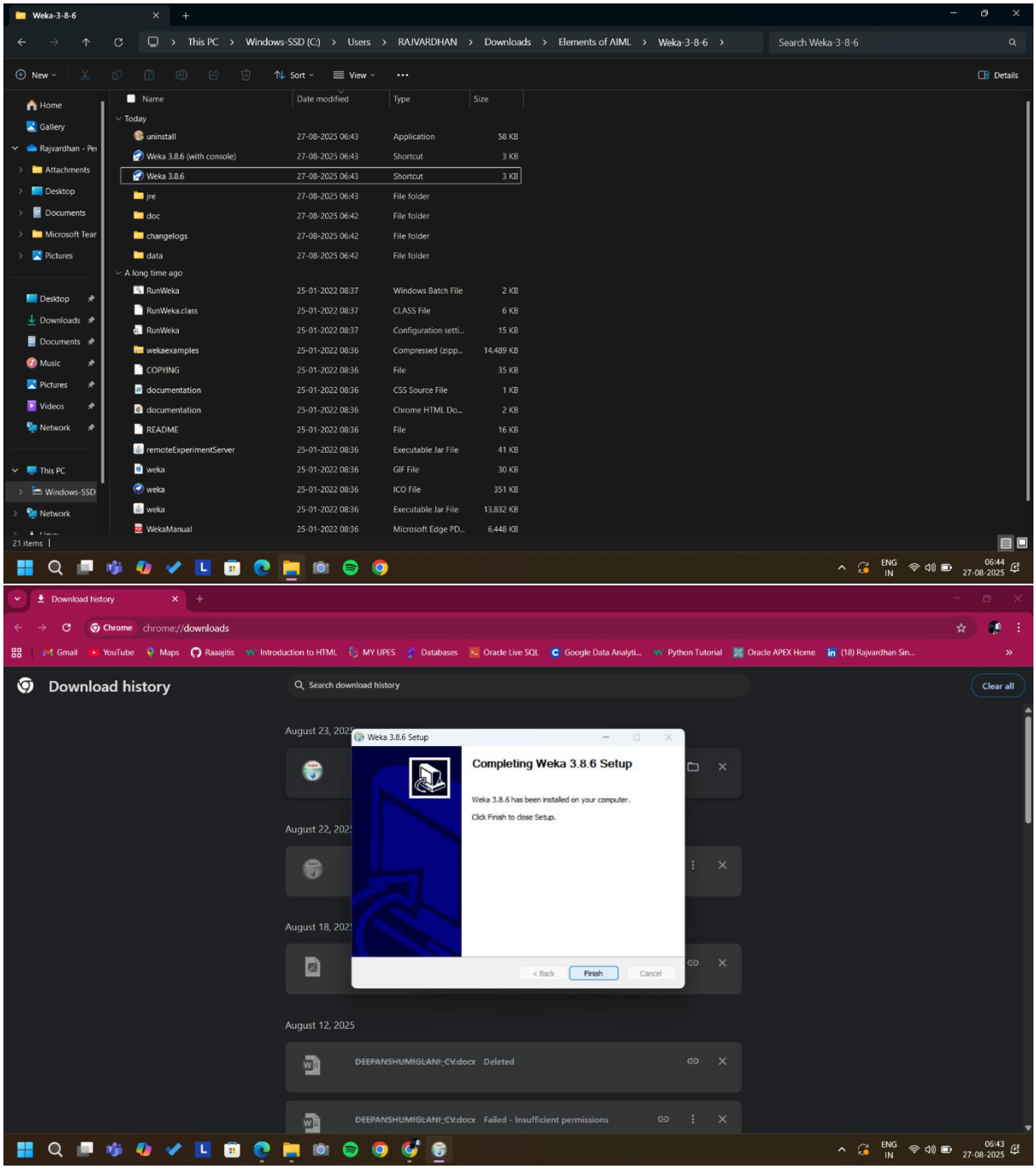
```
Enter a string: Suyash
Reversed String: hsayuS
Enter first string: Suyash
Enter second string: Hatwal
Concatenated String: SuyashHatwal
Uppercase: SUYASH
Lowercase: suyash
Enter string for slicing: Suyash
Enter start index: 2
Enter end index: 4
Sliced String: ya
```

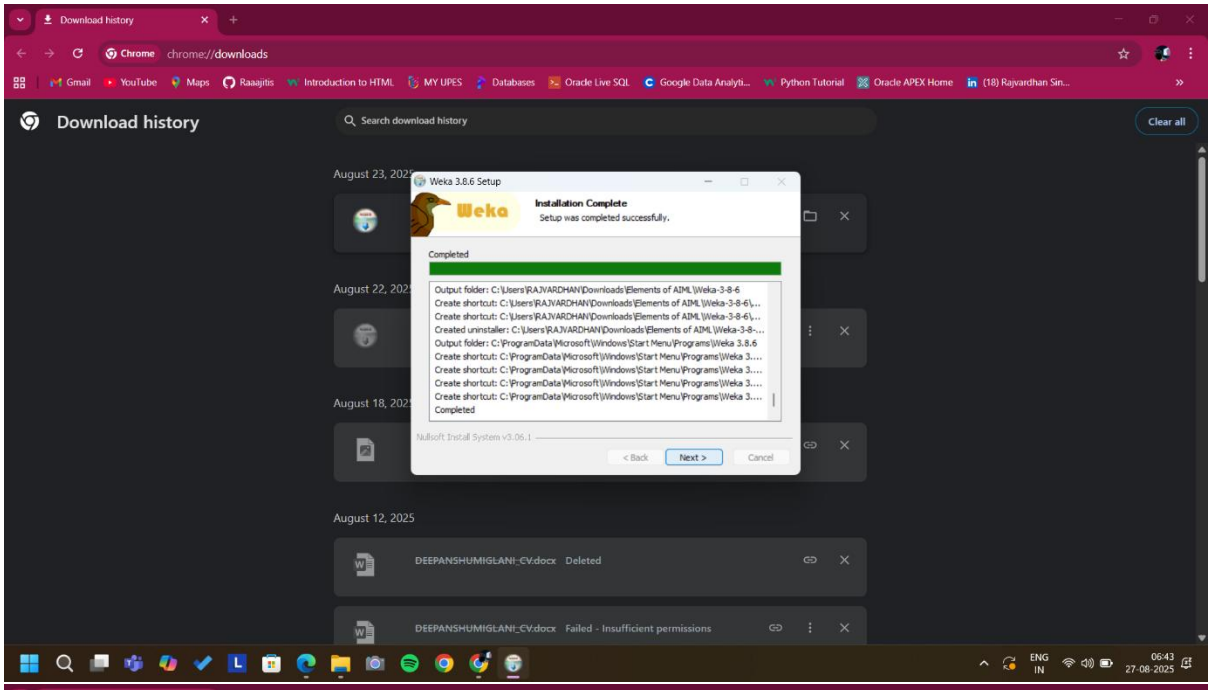
EXPERIMENT – 2

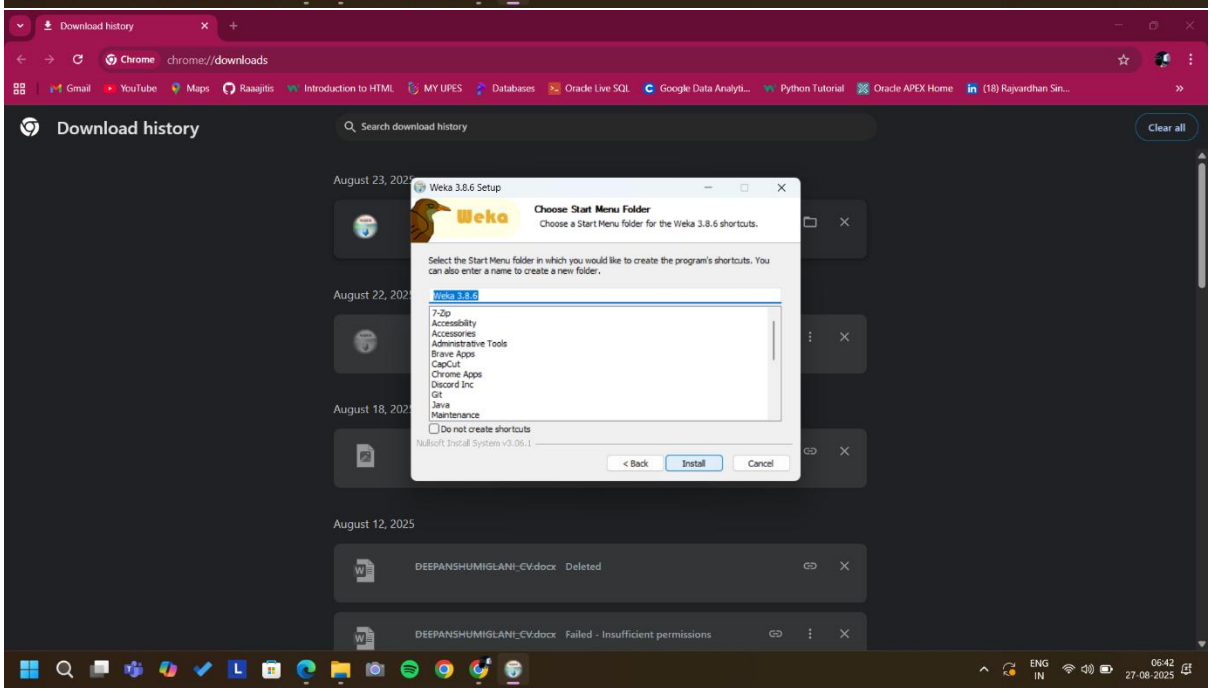
Q.1. Write downloading steps of WEKA Tool with proper screenshots.

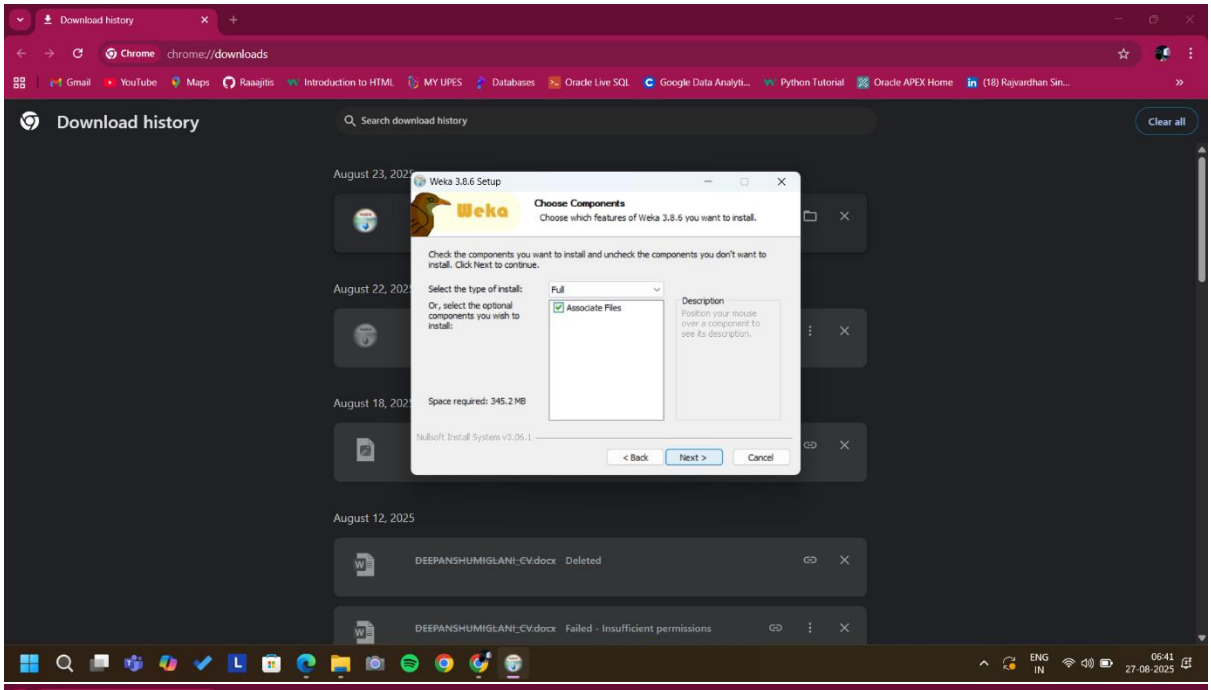
A.1. Following the below steps, I downloaded Weka, and the steps are:

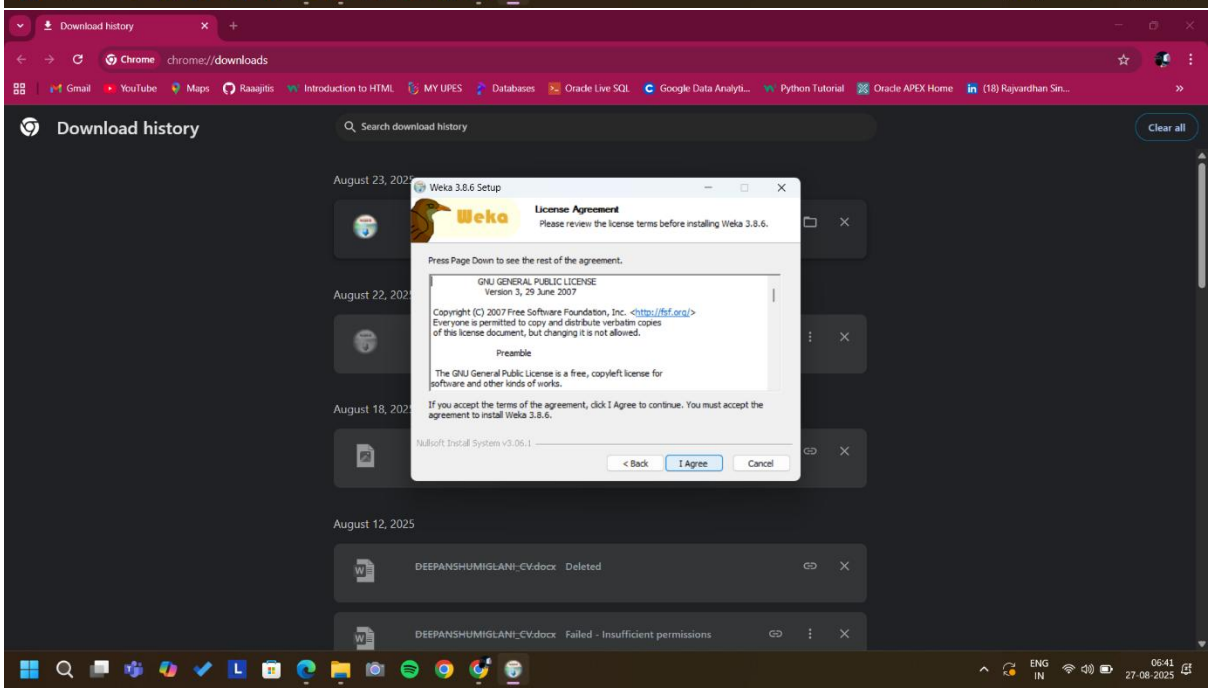


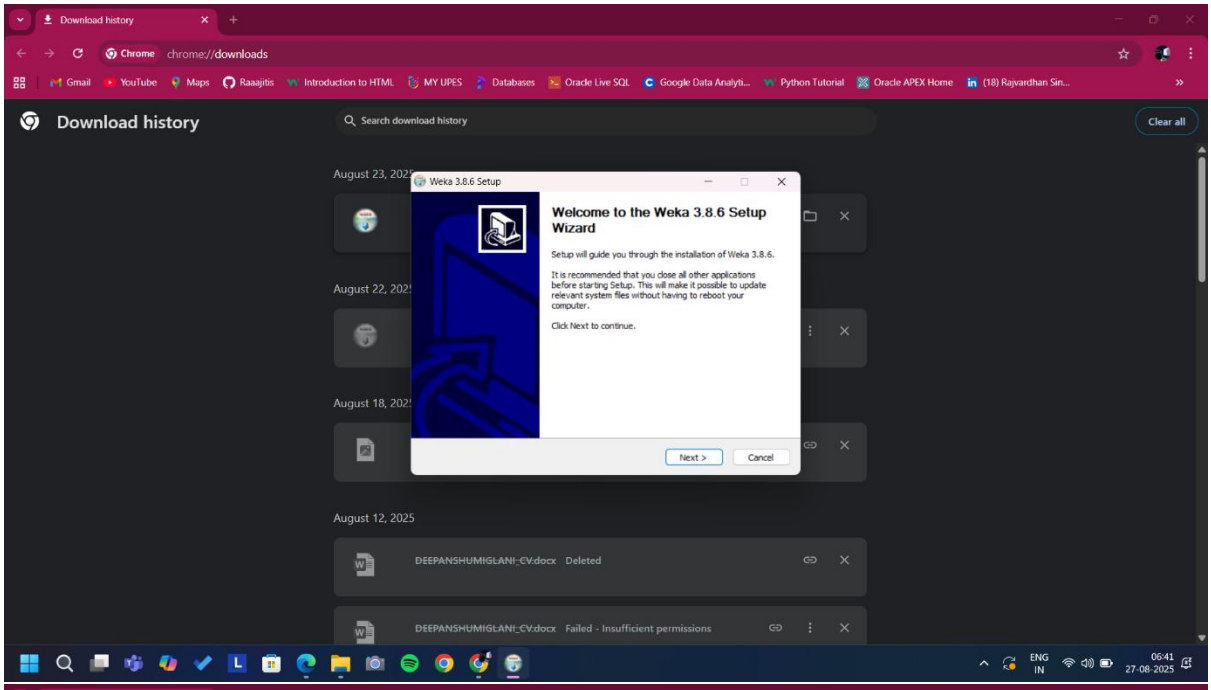

Weka-3-8-6
This PC › Windows-SSD (C:) › Users › RAJVARDHAN › Downloads › Elements of AIML › Weka-3-8-6 ›
Search Weka-3-8-6
New
Sort
View
Details
Home
Gallery
Rajvardhan - Per
Attachments
Desktop
Documents
Microsoft Tear
Pictures
Desktop
Downloads
Documents
Music
Pictures
Videos
Network
This PC
Windows-SSD
Network
Name
Date modified
Type
Size
Today
uninstall 27-08-2025 06:43 Application 58 KB
Weka 3.8.6 (with console) 27-08-2025 06:43 Shortcut 3 KB
Weka 3.8.6 27-08-2025 06:43 Shortcut 3 KB
jre 27-08-2025 06:43 File folder
doc 27-08-2025 06:42 File folder
changelogs 27-08-2025 06:42 File folder
data 27-08-2025 06:42 File folder
A long time ago
RunWeka 25-01-2022 08:37 Windows Batch File 2 KB
RunWeka.class 25-01-2022 08:37 CLASS File 6 KB
RunWeka 25-01-2022 08:37 Configuration setti... 15 KB
wekaexamples 25-01-2022 08:36 Compressed (zipp... 14,489 KB
COPYING 25-01-2022 08:36 File 35 KB
documentation 25-01-2022 08:36 CSS Source File 1 KB
documentation 25-01-2022 08:36 Chrome HTML Do... 2 KB
README 25-01-2022 08:36 File 16 KB
remoteExperimentServer 25-01-2022 08:36 Executable Jar File 41 KB
weka 25-01-2022 08:36 GIF File 30 KB
weka 25-01-2022 08:36 ICO File 351 KB
weka 25-01-2022 08:36 Executable Jar File 13,832 KB
WekaManual 25-01-2022 08:36 Microsoft Edge PD... 6,448 KB
21 items
ENG IN
06:44 27-08-2025
Download history
chrome://downloads
Gmail
YouTube
Maps
Raaajitis
Introduction to HTML
MY UPES
Databases
Oracle Live SQL
Google Data Analyti...
Python Tutorial
Oracle APEX Home
(18) Rajvardhan Sin...
Download history
Search download history
Clear all
August 23, 202
August 22, 202
August 18, 202
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions
Weka 3.8.6 Setup
Completing Weka 3.8.6 Setup
Weka 3.8.6 has been installed on your computer.
Click Finish to close Setup.
< Back
Finish
Cancel
ENG IN
06:43 27-08-2025

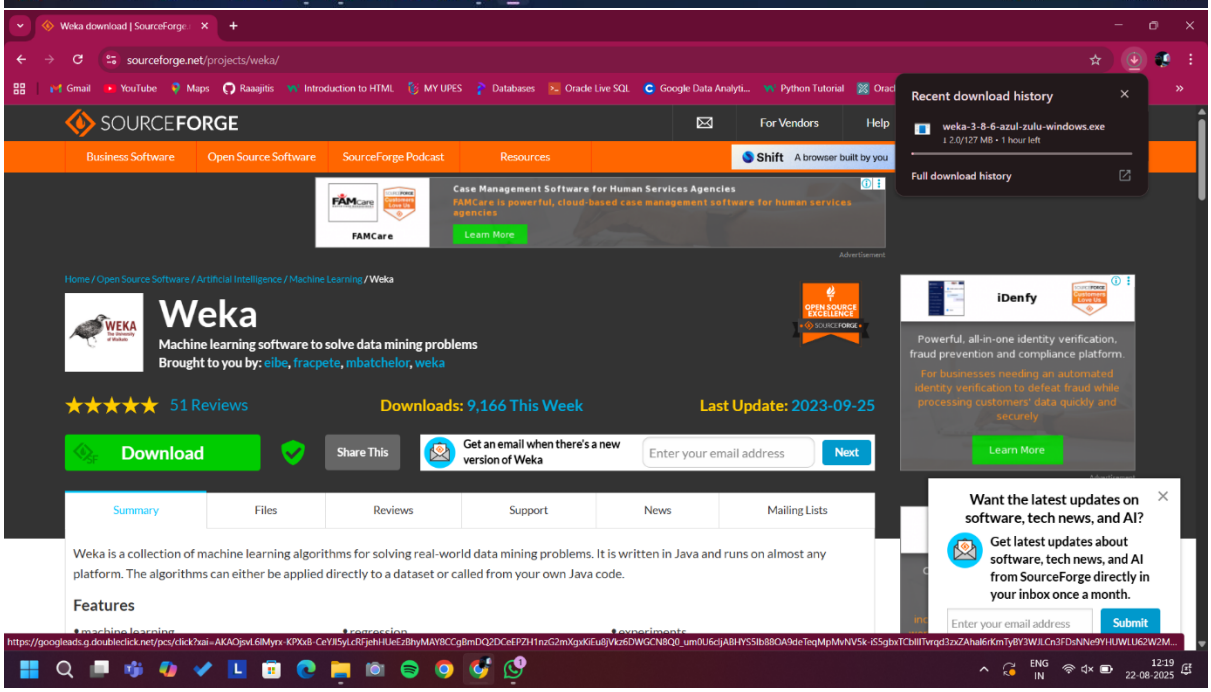

Download history
chrome://downloads
Download history
Search download history
Clear all
Weka 3.8.6 Setup
Installation Complete
Setup was completed successfully.
Completed
Output folder: C:\Users\RAJVARDHAN\Downloads\Elements of AIML\Weka-3-8-6
Create shortcut: C:\Users\RAJVARDHAN\Downloads\Elements of AIML\Weka-3-8-6\...
Create shortcut: C:\Users\RAJVARDHAN\Downloads\Elements of AIML\Weka-3-8-6\...
Created uninstaller: C:\Users\RAJVARDHAN\Downloads\Elements of AIML\Weka-3-8-...
Output folder: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Weka 3.8.6
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Weka 3....
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Weka 3....
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Weka 3....
Completed
Nullsoft Install System v3.06.1
< Back
Next >
Cancel
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions
ENG
IN
06:43
27-08-2025

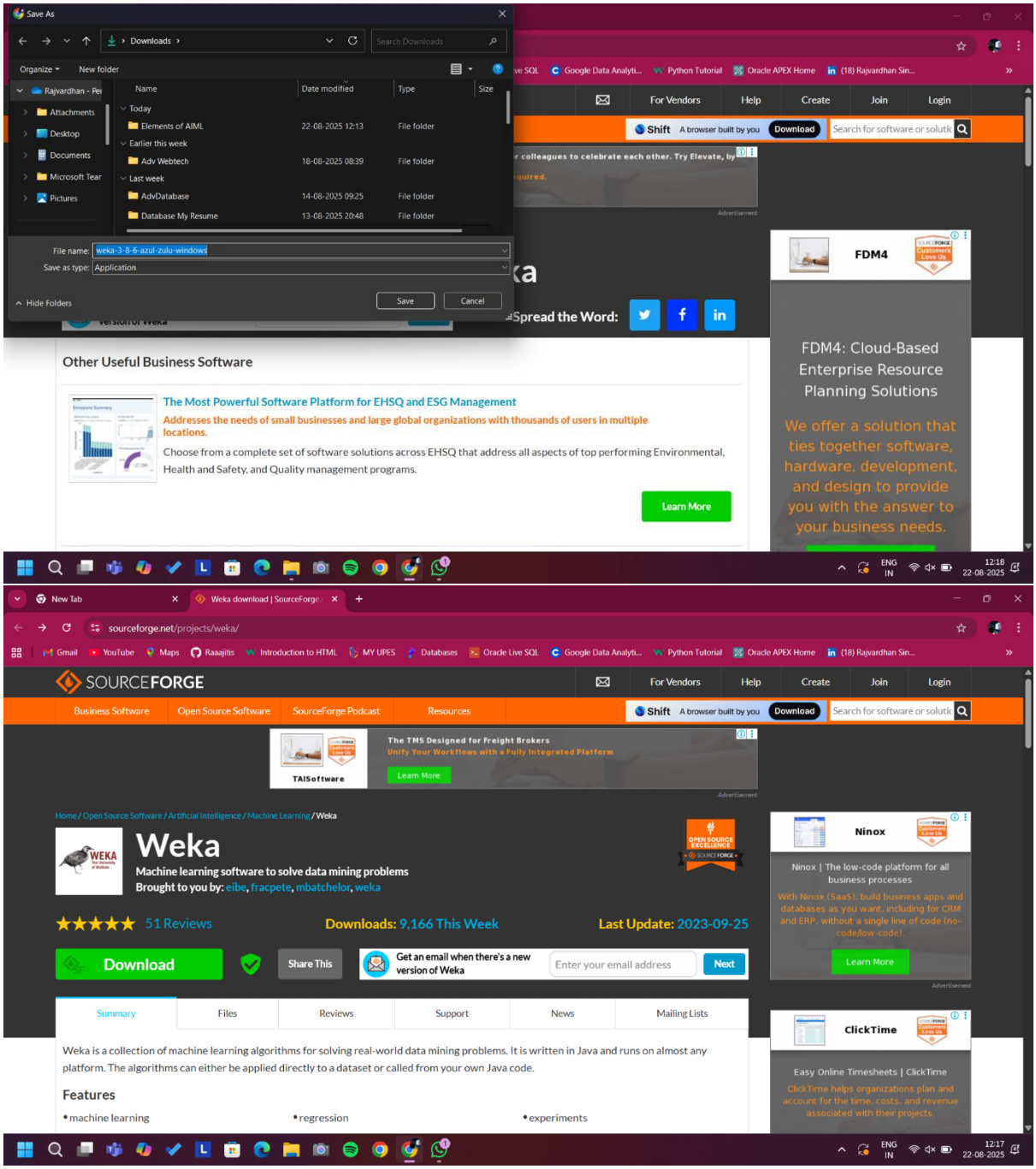

Download history
chrome://downloads
Download history
Search download history
Clear all
Weka 3.8.6 Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Weka 3.8.6 shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Weka 3.8.6
7-Zip
Accessibility
Accessories
Administrative Tools
Brave Apps
CapCut
Chrome Apps
Discord Inc
Git
Java
Maintenance
Do not create shortcuts
Nullsoft Install System v3.06.1
< Back
Install
Cancel
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions
ENG
IN
06:42
27-08-2025


Download history
chrome://downloads
Download history
Search download history
Clear all
August 23, 2025
August 22, 2025
August 18, 2025
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions
Weka 3.8.6 Setup
Weka
Choose Components
Choose which features of Weka 3.8.6 you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select the type of install:
Full
Or, select the optional components you wish to install:
Associate Files
Description
Position your mouse over a component to see its description.
Space required: 345.2 MB
Nullsoft Install System v3.06.1
< Back
Next >
Cancel


Download history
chrome://downloads
Download history
Search download history
Clear all
August 23, 2025
August 22, 2025
August 18, 2025
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions
Weka 3.8.6 Setup
Weka
License Agreement
Please review the license terms before installing Weka 3.8.6.
Press Page Down to see the rest of the agreement.
GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The GNU General Public License is a free, copyleft license for
software and other kinds of works.
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Weka 3.8.6.
Nullsoft Install System v3.06.1
< Back
I Agree
Cancel


Download history
chrome://downloads
Search download history
Clear all
Weka 3.8.6 Setup
Welcome to the Weka 3.8.6 Setup Wizard
Setup will guide you through the installation of Weka 3.8.6.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel
August 12, 2025
DEEPANSHUMIGLANI_CV.docx Deleted
DEEPANSHUMIGLANI_CV.docx Failed - Insufficient permissions


Recent download history
weka-3-8-6-azul-zulu-windows.exe
Full download history
SOURCEFORGE
For Vendors
Help
Business Software
Open Source Software
SourceForge Podcast
Resources
Case Management Software for Human Services Agencies
FAMCare is powerful, cloud-based case management software for human services agencies
Learn More
Home / Open Source Software / Artificial Intelligence / Machine Learning / Weka
Weka
Machine learning software to solve data mining problems
Brought to you by: eibe, fracpete, mbatchelor, weka
51 Reviews
Downloads: 9,166 This Week
Last Update: 2023-09-25
Download
Share This
Get an email when there's a new version of Weka
Enter your email address
Next
Summary
Files
Reviews
Support
News
Mailing Lists
Weka is a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost any platform. The algorithms can either be applied directly to a dataset or called from your own Java code.
Features
iDenfy
Powerful, all-in-one identity verification, fraud prevention and compliance platform.
For businesses needing an automated identity verification to defeat fraud while processing customers' data quickly and securely
Learn More
Want the latest updates on software, tech news, and AI?
Get latest updates about software, tech news, and AI from SourceForge directly in your inbox once a month.
Enter your email address
Submit


Save As
Downloads
Search Downloads
Organize
New folder
Rajvardhan - Pe
Attachments
Desktop
Documents
Microsoft Tear
Pictures
Name
Date modified
Type
Size
Today
Elements of AIML
22-08-2025 12:13
File folder
Earlier this week
Adv Webtech
18-08-2025 08:39
File folder
Last week
AdvDatabase
14-08-2025 09:25
File folder
Database My Resume
13-08-2025 20:48
File folder
File name: weka-3-8-6-azul-zulu-windows
Save as type: Application
Hide Folders
Save
Cancel
For Vendors
Help
Create
Join
Login
Shift
A browser built by you
Download
Search for software or solutic
Spread the Word:
Other Useful Business Software
The Most Powerful Software Platform for EHSQ and ESG Management
Addresses the needs of small businesses and large global organizations with thousands of users in multiple locations.
Choose from a complete set of software solutions across EHSQ that address all aspects of top performing Environmental, Health and Safety, and Quality management programs.
Learn More
FDM4
FDM4: Cloud-Based Enterprise Resource Planning Solutions
We offer a solution that ties together software, hardware, development, and design to provide you with the answer to your business needs.
ENG
IN
12:18
22-08-2025
New Tab
Weka download | SourceForge
sourceforge.net/projects/weka/
Gmail
YouTube
Maps
Raaajitis
Introduction to HTML
MY UPES
Databases
Oracle Live SQL
Google Data Analyti...
Python Tutorial
Oracle APEX Home
(18) Rajvardhan Sin...
SOURCEFORGE
Business Software
Open Source Software
SourceForge Podcast
Resources
The TMS Designed for Freight Brokers
Unify Your Workflows with a Fully Integrated Platform
TAISoftware
Learn More
Home / Open Source Software / Artificial Intelligence / Machine Learning / Weka
Weka
Machine learning software to solve data mining problems
Brought to you by: eibe, fracpete, mbatchelor, weka
51 Reviews
Downloads: 9,166 This Week
Last Update: 2023-09-25
Download
Share This
Get an email when there's a new version of Weka
Enter your email address
Next
Summary
Files
Reviews
Support
News
Mailing Lists
Weka is a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost any platform. The algorithms can either be applied directly to a dataset or called from your own Java code.
Features
machine learning
regression
experiments
Ninox
Ninox | The low-code platform for all business processes
With Ninox (SaaS), build business apps and databases as you want, including for CRM and ERP, without a single line of code (no-code/low-code).
Learn More
ClickTime
Easy Online Timesheets | ClickTime
ClickTime helps organizations plan and account for the time, costs, and revenue associated with their projects.
ENG
IN
12:17
22-08-2025

Q.2. Download student dataset from kaggle, check for the fields- Student name, course, marks/percentage. Classify this dataset course wise and marks wise (Set your own range for marks.) Also display visualization charts for classification.

A.2. After following numerous steps, the final data visualization is as given below,

	A	B	C	D	E	F	G	H
1	gender	race/ethni	parental le	lunch	test prepa	math score	reading sc	writing score
2	female	group B	bachelor's	standard	none	72	72	74
3	female	group C	some colle	standard	completed	69	90	88
4	female	group B	master's d	standard	none	90	95	93
5	male	group A	associate's	free/reduc	none	47	57	44
6	male	group C	some colle	standard	none	76	78	75
7	female	group B	associate's	standard	none	71	83	78
8	female	group B	some colle	standard	completed	88	95	92
9	male	group B	some colle	free/reduc	none	40	43	39
10	male	group D	high schoo	free/reduc	completed	64	64	67
11	female	group B	high schoo	free/reduc	none	38	60	50
12	male	group C	associate's	standard	none	58	54	52
13	male	group D	associate's	standard	none	40	52	43
14	female	group B	high schoo	standard	none	65	81	73
15	male	group A	some colle	standard	completed	78	72	70
16	female	group A	master's d	standard	none	50	53	58
17	female	group C	some high	standard	none	69	75	78
18	male	group C	high schoo	standard	none	88	89	86
19	female	group B	some high	free/reduc	none	18	32	28
20	male	group C	master's d	free/reduc	completed	46	42	46
21	female	group C	associate's	free/reduc	none	54	58	61
22	male	group D	high schoo	standard	none	66	69	63
23	female	group B	some colle	free/reduc	completed	65	75	70
24	male	group D	some colle	standard	none	44	54	53
25	female	group C	some high	standard	none	69	73	73
26	male	group D	bachelor's	free/reduc	completed	74	71	80
27	male	group A	master's d	free/reduc	none	73	74	72

StudentsPerformance

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter

Choose **None** | Apply | Stop

Current relation

Relation: StudentsPerformance-weka.filters.unsupervised.attribu... Attributes: 4

Instances: 1000 Sum of weights: 1000

Attributes

All | None | Invert | Pattern

No.	Name
1	gender
2	math score
3	reading score
4	writing score

Remove

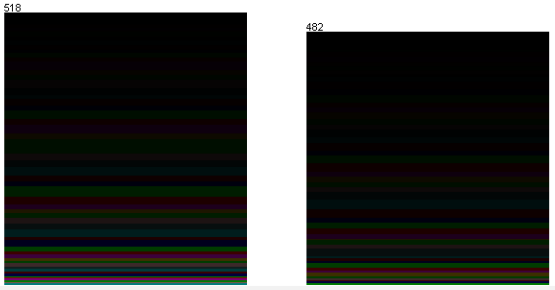
Selected attribute

Name: gender Type: Nominal

Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	female	518	518
2	male	482	482

Class: math score (Nom) | Visualize All



Status

OK | Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier

Choose **RandomForest** -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Test options

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) writing score

Start | Stop

Result list (right-click for options)

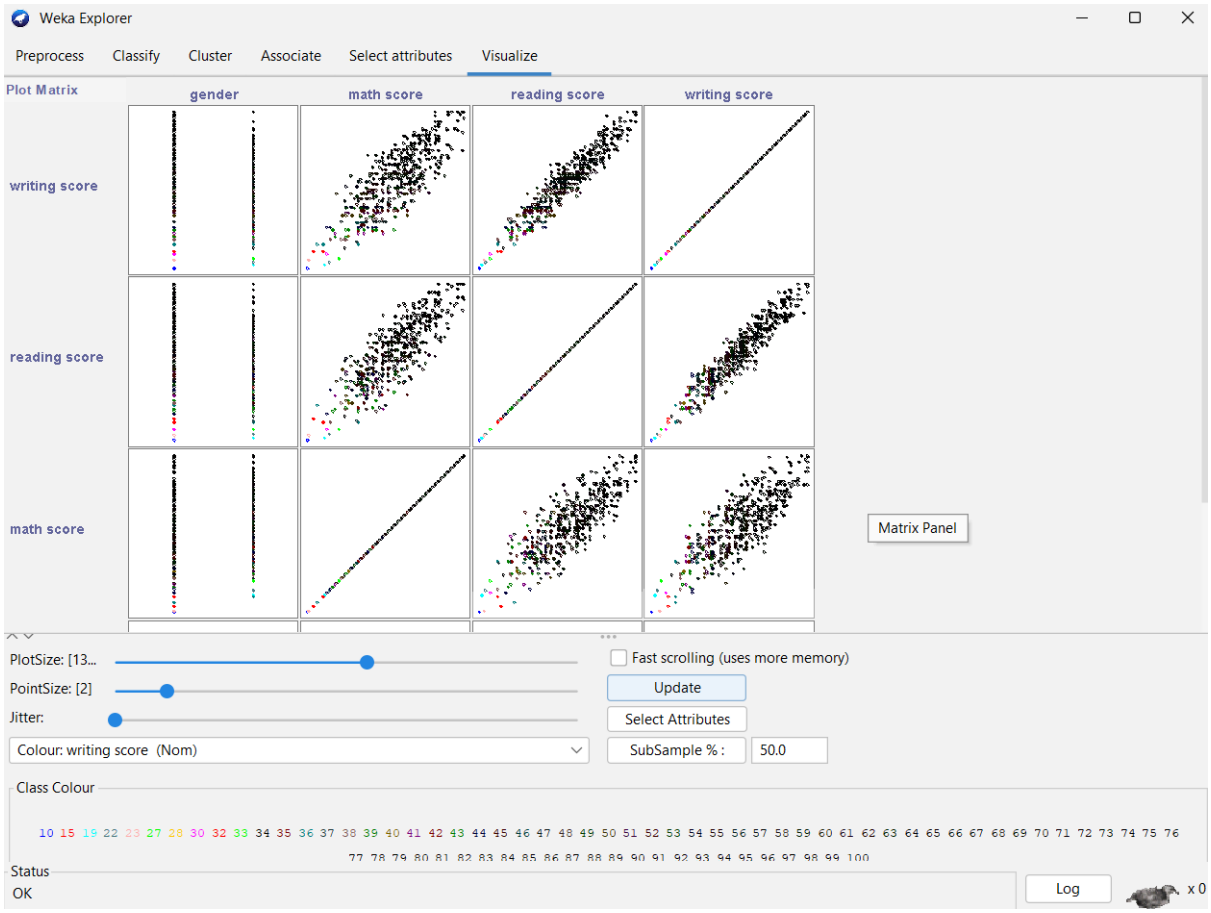
12:54:35 - trees.RandomForest

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation:     StudentsPerformance-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka
Instances:    1000
Attributes:   4
              gender
              math score
              reading score
              writing score
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 1.71 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          88               8.8    %
Incorrectly Classified Instances       912              91.2    %
Kappa statistic                          0.0693
Mean absolute error                      0.0243
Root mean squared error                  0.1191
Relative absolute error                 95.4252 %
Root relative squared error            105.4935 %
Total Number of Instances             1000
```

Status

OK | Log | x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Plot Matrix
gender
math score
reading score
writing score
writing score
reading score
math score
Matrix Panel
PlotSize: [13...
PointSize: [2]
Jitter:
Fast scrolling (uses more memory)
Update
Select Attributes
Colour: writing score (Nom)
SubSample % :
50.0
Class Colour
10 15 19 22 23 27 28 30 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Status
OK
Log
x 0

EXPERIMENT – 3

TRAFFIC SIGNAL CASE

1. Defining the propositions

- **R: The traffic light is Red**
- **G: The traffic light is Green**
- **Y: The traffic light is Yellow**

Since only one traffic light can be active at a time, the following mutual exclusivity condition must hold:

R OR G OR Y

AND

NOT(R AND G) AND NOT(G AND Y) AND NOT(R AND Y)

Next, we define the possible car behaviors:

- **S: Cars Stop**
- **Go: Cars Go**
- **Sl: Cars Slow down**

The rules governing the system are expressed as implications:

R -> S (If the light is Red, cars must Stop)

G -> Go (If the light is Green, cars must Go)

Y -> Sl (If the light is Yellow, cars must Slow down)

2. Case: "Cars did not stop" is true

Suppose the statement “Cars did not stop” is true.

- **Formally: NOT S**
- **From the rule R -> S, we know that if R were true, then S would also be true.**
- **But since NOT S holds, this implies that R cannot be true.**

Conclusion:

- **The traffic light is not Red.**
- **Therefore, the light must be either Green or Yellow.**

3. If the light is not Yellow

Now suppose it is also given that the light is not Yellow.

- **Formally: NOT Y**
- **This leaves two possibilities: R or G.**
- **But we have already concluded that R is false (since the cars did not stop).**

Conclusion:

- **The only remaining option is G.**
- **Therefore, the light must be Green, and the cars must Go.**

4. Why and Why Not?

- **If we only know that the light is not Yellow, the possible states are Red or Green.**
- **If it were Red, cars would have stopped — but we know they did not.**
- **Therefore, Red is ruled out, and the only possibility is Green.**

- **Hence, cars must be going.**

Python code for the following logic:

```
def traffic_rule(light_color):
    if light_color == "red":
        return "Cars must stop"
    elif light_color == "green":
        return "Cars must go"
    elif light_color == "yellow":
        return "Cars must slow down"
    else:
        return "Invalid light color"

cars_stopped = False

if not cars_stopped:
    print("Cars did not stop → Light is not red.")
    possible_lights = ["green", "yellow"]
    print("So possible lights are:", possible_lights)

    light_is_yellow = False
    if not light_is_yellow:
        print("Since it's not yellow → It must be green.")
```

print(traffic_rule("green"))

OUTPUT

aimlexp3.py > ...

```python
def traffic_rule(light_color):
    if light_color == "red":
        return "Cars must stop"
    elif light_color == "green":
        return "Cars must go"
    elif light_color == "yellow":
        return "Cars must slow down"
    else:
        return "Invalid light color"

cars_stopped = False

if not cars_stopped:
    print("Cars did not stop → Light is not red.")
    possible_lights = ["green", "yellow"]
    print("So possible lights are:", possible_lights)


    light_is_yellow = False
    if not light_is_yellow:
        print("Since it's not yellow → It must be green.")
        print(traffic_rule("green"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR

```
er and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\RAJVARDHAN\Downloads\Hello world> python -u "c:\Users\RAJVARDHAN\Downloads\Hello world\aimlexp3,py"
Possible light colors: ['red', 'green', 'yellow']
PS C:\Users\RAJVARDHAN\Downloads\Hello world> python -u "c:\Users\RAJVARDHAN\Downloads\Hello world\aimlexp3,py"
Cars did not stop → Light is not red.
So possible lights are: ['green', 'yellow']
Since it's not yellow → It must be green.
Cars must go
PS C:\Users\RAJVARDHAN\Downloads\Hello world>
```

EXPERIMENT – 4

PREDICATE LOGIC AND RESOLUTION

1. Translate into predicate logic:

a. Every teacher teaches some subject.

$$\forall x(\boldsymbol{Teacher}(x) \rightarrow \exists y\ \boldsymbol{Teaches}(x, y))$$

(If you prefer to name the predicate Subject(y) too:
$\forall x(\boldsymbol{Teacher}(x) \rightarrow \exists y(\boldsymbol{Subject}(y) \wedge \boldsymbol{Teaches}(x, y)))$**.)**

b. No student likes exams.

$$\forall x(\boldsymbol{Student}(x) \rightarrow \neg \boldsymbol{Likes}(x, \boldsymbol{Exams}))$$

(or equivalently $\neg \exists x(\boldsymbol{Student}(x) \wedge \boldsymbol{Likes}(x, \boldsymbol{Exams}))$**.)**

c. Some animals are not carnivores.

$$\exists x(\boldsymbol{Animal}(x) \wedge \neg \boldsymbol{Carnivore}(x))$$

d. Everyone who studies hard succeeds.

$$\forall x(\boldsymbol{StudiesHard}(x) \rightarrow \boldsymbol{Succeeds}(x))$$

2. Use resolution to prove (I'll show reasoning and implemented

Q2 a.)

a. If it is cold, then people wear jackets. It is cold. → People wear jackets.

- **Formalize propositional atoms: Cold, WearJacket.**
- **Clause form: ¬Cold ∨ WearJacket (the implication) and Cold.**
- **Negate the goal WearJacket → add ¬WearJacket.**

- **Resolve ¬Cold ∨ WearJacket with Cold to get WearJacket. Alternatively, resolution with ¬WearJacket yields contradiction**
- **b. If a person is a parent, then they care for their child. John is a parent. → John cares for his child.**

- **Predicates: Parent(x), CaresForChild(x), constant John.**
- **Clause form: ¬Parent(x) ∨ CaresForChild(x); add Parent(John). Negate goal ¬CaresForChild(John) and resolve:**
  1. **¬Parent(x) ∨ CaresForChild(x) with Parent(John) yields CaresForChild(John).**
  2. **Resolving CaresForChild(John) with ¬CaresForChild(John) gives contradiction, so CaresForChild(John) follows.**

3. Resolution satisfiability test

Given clauses:

- **(a)** $P \vee Q$
- **(b)** $\neg P \vee R$
- **(c)** $\neg R$

What can be derived?

Resolution steps:

1. **Resolve (b)** $\neg P \vee R$ **with (c)** $\neg R$ **— since R and ¬R are complementary, we get ¬P.**
2. **Resolve ¬P with (a)** $P \vee Q$ **to get Q.**

4. Given KB: All doctors are educated; Smith is a doctor. Prove Educated(Smith).

KB in clause form:

- **∀x (Doctor(x) → Educated(x)) ≡ clause: ¬Doctor(x) ∨ Educated(x)**
- **Doctor(Smith)**

Negate goal: ¬Educated(Smith) and add it. Resolve:

- **Resolve ¬Doctor(x) ∨ Educated(x) with ¬Educated(Smith) (not directly complementary),**
- **Instead resolve ¬Doctor(x) ∨ Educated(x) with Doctor(Smith) to get Educated(Smith).**
- **Resolving Educated(Smith) with ¬Educated(Smith) yields contradiction. So Educated(Smith) is proved.**

5. Knowledge base (to be proved with resolution; I implemented this in Python)

KB:

- **All students who read books are intelligent:**
  $\forall x(\boldsymbol{Student}(x) \wedge \boldsymbol{Reads}(x) \rightarrow \boldsymbol{Intelligent}(x))$
  → clause form: ¬Student(x) ∨ ¬Reads(x) ∨ Intelligent(x)
- **Student(Ravi)**
- **Reads(Ravi)**

Goal: Intelligent(Ravi). (Negate goal ¬Intelligent(Ravi) and resolve; a derivation leads to contradiction so the goal is proved.)

Python codes for:

Q2. A) def resolve(c1, c2):

```
resolvents = []
for lit in c1:
if lit.startswith("¬"):
complementary = lit[1:]
else:
complementary = "¬" + lit
if complementary in c2:
new_clause = (c1 - {lit}) | (c2 - {complementary})
resolvents.append(new_clause)
return resolvents

kb = [ {"¬Cold", "WearJacket"}, {"Cold"} ]
goal = {"¬WearJacket"}
kb.append(goal)

print("Initial Clauses:")
for i, clause in enumerate(kb, 1):
print(f"C{i}: {clause}")

new = set()
found_contradiction = False
```

```
while True:
pairs = [(kb[i], kb[j]) for i in range(len(kb)) for j in range(i+1, len(kb))]
for (c1, c2) in pairs:
resolvents = resolve(c1, c2)
for res in resolvents:
if not res:
print(f"\nResolved {c1} and {c2} -> Empty Clause")
print("Contradiction found — WearJacket is proved.")
found_contradiction = True
break
if res not in kb:
kb.append(res)
print(f"Resolved {c1} and {c2} -> {res}")
if found_contradiction:
break
if found_contradiction:
break
```

Q5)

```
import re
from copy import deepcopy

def parse_literal(s):
```

```
s = s.strip()
neg = False
if s.startswith("¬") or s.startswith("~"):
neg = True
s = s[1:].strip()
match = re.match(r'(\w+)\((.*?)\)', s)
if match:
pred = match.group(1)
args = [a.strip() for a in match.group(2).split(",")]
else:
pred = s
args = []
return {"pred": pred, "args": args, "neg": neg}


def literal_to_str(lit):
return ("¬" if lit["neg"] else "") + lit["pred"] + "(" +
",".join(lit["args"]) + ")"


def clause_to_str(clause):
return " ∨ ".join([literal_to_str(l) for l in clause]) if clause else "∅"


def is_variable(x):
return x.islower()
```

```
def unify_terms(x, y, subs):
if subs is None:
return None
if x == y:
return subs
if is_variable(x):
subs = subs.copy()
subs[x] = y
return subs
if is_variable(y):
subs = subs.copy()
subs[y] = x
return subs
return None

def unify_literals(l1, l2):
if l1["pred"] != l2["pred"] or l1["neg"] == l2["neg"]:
return None
subs = {}
for a, b in zip(l1["args"], l2["args"]):
subs = unify_terms(a, b, subs)
if subs is None:
return None
return subs
```

```
def substitute(lit, subs):
return {
"pred": lit["pred"],
"neg": lit["neg"],
"args": [subs.get(a, a) for a in lit["args"]]
}

def resolve(c1, c2):
resolvents = []
for l1 in c1:
for l2 in c2:
subs = unify_literals(l1, l2)
if subs is not None:
new_clause = [substitute(x, subs) for x in c1 if x != l1] +
[substitute(x, subs) for x in c2 if x != l2]
uniq = []
for l in new_clause:
if l not in uniq:
uniq.append(l)
resolvents.append(uniq)
return resolvents

KB = [
```

```
[parse_literal("¬Student(x)"), parse_literal("¬Reads(x)"),
parse_literal("Intelligent(x)")],

[parse_literal("Student(Ravi)")],

[parse_literal("Reads(Ravi)")]

]


goal = [parse_literal("¬Intelligent(Ravi)")]

KB.append(goal)


print("Initial Clauses:")

for i, c in enumerate(KB, 1):

print(f"C{i}: {clause_to_str(c)}")


new = []

found = False


while True:

pairs = [(KB[i], KB[j]) for i in range(len(KB)) for j in range(i+1,
len(KB))]

for (c1, c2) in pairs:

resolvents = resolve(c1, c2)

for r in resolvents:

if not r:

print(f"\nResolved {clause_to_str(c1)} and {clause_to_str(c2)} ->
Ø")
```

print("Ravi is intelligent (proved).")

found = True

break

if r not in KB:

KB.append(r)

print(f"Resolved {clause_to_str(c1)} and {clause_to_str(c2)} -> {clause_to_str(r)}")

if found:

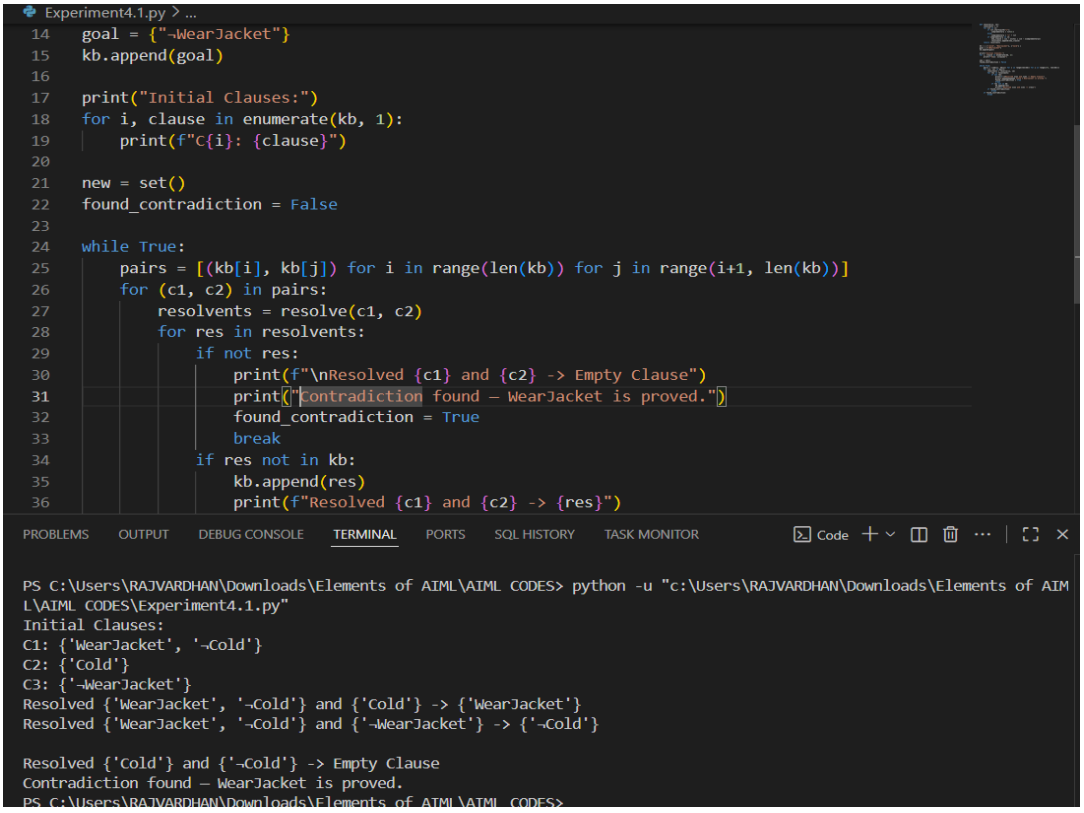
break

if found:

break

OUTPUT

Q2. A)



Q5)

Experiment4.2.py > ...

```
def resolve(c1, c2):
                complementary = lit[1:]
            else:
                complementary = "¬" + lit
            if complementary in c2:
                new_clause = (c1 - {lit}) | (c2 - {complementary})
                resolvents.append(new_clause)
    return resolvents

kb = [ {"¬Cold", "WearJacket"}, {"Cold"} ]
goal = {"¬WearJacket"}
kb.append(goal)

print("Initial Clauses:")
for i, clause in enumerate(kb, 1):
    print(f"C{i}: {clause}")

new = set()
found_contradiction = False

while True:
    pairs = [(kb[i], kb[j]) for i in range(len(kb)) for j in range(i+1, len(kb))]
    for (c1, c2) in pairs:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR Code

```
PS C:\Users\RAJVARDHAN\Downloads\Elements of AIML\AIML CODES> python -u "c:\Users\RAJVARDHAN\Downloads\Elements of AIML\AIML CODES\Experiment4.2.py"
Initial Clauses:
C1: {'WearJacket', '¬Cold'}
C2: {'Cold'}
C3: {'¬WearJacket'}
Resolved {'WearJacket', '¬Cold'} and {'Cold'} -> {'WearJacket'}
Resolved {'WearJacket', '¬Cold'} and {'¬WearJacket'} -> {'¬Cold'}

Resolved {'Cold'} and {'¬Cold'} -> Empty Clause
Contradiction found — WearJacket is proved.
PS C:\Users\RAJVARDHAN\Downloads\Elements of AIML\AIML CODES>
```

EXPERIMENT – 5

UNIFICATION IN AI

Unification Algorithm:

Unification is a crucial technique in first-order logic (FOL) that allows two logical expressions to be

made equal by finding an appropriate substitution. The UNIFY algorithm determines whether two

expressions can be unified and provides the necessary substitutions to achieve this.

Algorithm: UNIFY(Expression1, Expression2)

Step 1: Check if either expression is a variable or constant.

• If the expressions are identical, return an empty substitute (no changes are needed).

• If one of them is a variable, check if the variable occurs in the other expression (to avoid circular

dependency). If so, return FAILURE. Otherwise, substitute the variable with the corresponding

term.

• If neither is a variable expression, then proceed to the next step.

Step 2: Compare the names of the predicates

• If the predicate names in the two expressions are different, unification cannot occur as they

represent different relationships.

Step 3: Verify the number of arguments.

• If the number of parameters in the two expressions is not the same, unification is not possible.

The algorithm will return a FAILURE response.

Step 4: Create an empty substitution set.

• Establish an empty set to hold any variable substitutions required for unification.

Step 5: Attempt to unify each argument recursively.

Examine the arguments of the two expressions −

• Recursively use the UNIFY function on each pair of parameters.

• If a failure occurs at any point, return a FAILURE.

• If a substitution is identified, apply it to the remaining expressions.

Step 6: Return to the Most General Unifier (MGU)

• Once all the words have been successfully unified, present the final substitution set, known as

the Most General Unifier (MGU).

• This ensures that the expressions are simplified to be equal using the simplest substitutions

available.

Python code for the following logic:

```
def unify(term1, term2, substitutions=None):
        if substitutions is None:
```

```
    substitutions = {}
  if term1 == term2:
    return substitutions  # Terms are already identical
  if isinstance(term1, str) and term1.islower():  # If term1 is a variable
    return unify_var(term1, term2, substitutions)
  if isinstance(term2, str) and term2.islower():  # If term2 is a variable
    return unify_var(term2, term1, substitutions)
  if isinstance(term1, tuple) and isinstance(term2, tuple) and len(term1) == len(term2):
    for t1, t2 in zip(term1, term2):
      substitutions = unify(t1, t2, substitutions)
      if substitutions is None:
        return None
    return substitutions
  return None  # Unification fails

def unify_var(var, value, substitutions):
  if var in substitutions:
    return unify(substitutions[var], value, substitutions)
  elif value in substitutions:
    return unify(var, substitutions[value], substitutions)
  else:
    substitutions[var] = value
```

return substitutions

expr1 = ("loves", "x", "y")

expr2 = ("loves", "Alice", "Bob")

result = unify(expr1, expr2)

print("Result:", result)

OUTPUT

unification.py > ...

```
def unify(term1, term2, substitutions=None):
            if substitutions is None:
                return None
        return substitutions
    return None  # Unification fails

def unify_var(var, value, substitutions):
    if var in substitutions:
        return unify(substitutions[var], value, substitutions)
    elif value in substitutions:
        return unify(var, substitutions[value], substitutions)
    else:
        substitutions[var] = value
        return substitutions


expr1 = ("loves", "x", "y")
expr2 = ("loves", "Alice", "Bob")

result = unify(expr1, expr2)
print("Result:", result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR

```
er and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\RAJVARDHAN\Downloads\Hello world> python -u "c:\Users\RAJVARDHAN\Downloads\Hello world\unification.py"
Result: {'x': 'Alice', 'y': 'Bob'}
PS C:\Users\RAJVARDHAN\Downloads\Hello world>
```