MACHINE LEARNING ASSIGNEMENT-1

REPORT

SET - 1

Team members:

1. Hareni M – BL.EN. U4AIE22117
2. Surya Krishna – BL. EN. U4AIE22154
3. Swetha Ayyappan – BL.EN. U4AIE22155

QUESTION 1

COUNT PAIRS THAT ADD TO 10

1. Initialize a list of integers as [2,7,4,1,3,6]
2. Define a function: count\_pairs()
3. Initialize variable ‘varCount’ to 0
4. Iterate through list to find the number of pairs that add up to 10:

For each list element at index i from 0 to length(list)-1:

For each element at index j from i+1 to length(list)-1:

If list[i] + list[j] equals to 10:

Increment varCount variable by 1.

1. Return varCount.

1. Define main() function:
2. Call the count\_pairs() function and store in variable ‘result’.
3. Print the total numbers of pairs of elements that add up to 10.

1. Call main() to execute the code.

Short description of code:

* A list of real numbers is initialized.
* A function has been defined that finds number of pairs of numbers that add up to 10.
* Iterate through list with i and j iteration variables and identify numbers that add upto 10.
* Create counter variable named ‘varCount’ and increment its value by 1 when a pair that adds upto 10 is identified.

QUESTION 2

CALCULATE RANGE

1. Initialize an empty list named ‘real\_number\_list’.
2. Define function calculate\_range():
3. If length of real\_number\_list is less than 3:

Return “Range determination is not possible”

1. Find maximum number ‘max\_num’ in real\_number\_list
2. Find minimum number ‘min\_num’ in real\_number\_list
3. Calculate range = max\_num – min\_num
4. Return range
5. Define main():
6. Prompt user to enter number of elements in real\_number\_list and assign it to n.
7. Repeat loop n times and append numbers entered by user to real\_number\_list.
8. Call main()
9. Result = calculate\_range()
10. Diplay result.

SHORT DESCRIPTION

* Initialize an empty list to store real numbers to calculate range.
* Check if number of elements is less than 3. If yes, range cannot be calculated.
* Else, maximum number and minimum number are obtain from list using max and min functions.
* Calculate range and return it.
* In main(), obtain user input for list elements.
* Call calculate\_range() to display range to user.

QUESTION 3

MATRIX POWER

1. Define matrix\_multiply Function.

# Assuming A and B are 2D matrices

rows\_A, cols\_A = len(A), len(A[0])

rows\_B, cols\_B = len(B), len(B[0])

if cols\_A != rows\_B:

raise ValueError("Incompatible matrix dimensions for multiplication.")

Initialize result matrix with zeros .Perform matrix multiplication and return the result.

2. Define the matrix\_power Function def matrix\_power(A, m).

Check if the input matrix is square ,if not is\_square\_matrix(A raise ValueError("Input matrix must be square for matrix power calculation.")

1. .Initialize result as an identity matrix .
2. Multiply A with itself m times.

3:Define is square matrix function :

Check if the number of rows is equal to the number of columns in the input matrix (determining if it is a square matrix).

4.Return True if the matrix is square, and False otherwise.

5. In the Main Function:

6. Define the Matrix A and Call the matrix\_power Function

7. Print the Result, print(f"A raised to the power of {power\_m}:\n{result\_power}").

SHORT DESCRIPTION

* Matrix Multiplication (matrix\_multiply): Initialize result matrix.
* Matrix Power (matrix\_power): Check if square matrix,i.e rows = columns.
* Initialize result as identity matrix.
* Multiply A with itself m times.
* Is Square Matrix (is\_square\_matrix): Check rows equals columns.

Return True if square, else False.

* Main Function: Define matrix A.
* Call matrix\_power.
* Print result: A^m.

QUESTION 4

HIGHEST OCCURRING CHARACTER

1. Define the highest\_occuring\_character function: def highest\_occuring\_character(input\_string)
2. Initialize a dictionary to store character counts char\_count = {}.
3. Iterate through each character in the input string for char in input\_string.
4. Check if the character is an alphabet character if char.isalpha().
5. Increment the count of the character in the dictionary char\_count[char] = char\_count.get(char, 0) + 1.
6. Find the character with the maximum count max\_char = max (char\_count, key=char\_count.get) max\_count = char\_count[max\_char] .
7. Return a formatted string with information about the most frequent character return f"Most frequent character: {max\_char}, Occurrence count: {max\_count}"
8. Iterate through each character in the input string.

For each alphabet character:

Increment its count in char\_count.

1. Finds the character with the maximum count and its occurrence count.
2. Return A formatted string equals most frequent character.

In the main function:

11. Define the input string input\_str\_occurrence.

12. Call the highest\_occuring\_character function with input\_str\_occurrence.

13. Print the result of the most occurring character calculation.

SHORT DESCRIPTION

* Create a function to calculate highest occurring character in a string.
* Define an empty dictionary to store the count of each character.
* Iterate through the string, and increment counter of a specific character each time it occurs.
* Store the count values in dictionary.
* Retrieve maximum occurring alphabet using max() function.
* Under main(), obtain user input (a string), and call highest\_occurring\_character() function to display final result.