**Machine Learning Report**

**LAB\_ASSIGNMENT-1**

NAME:- Chakka Subrahmanya Sresti

REGNO:- AIE22179

SEMESTER:-4

Question-1)

Psuedo code:-

function sum\_pairs\_with\_10(numbers):

count = 0

for i from 0 to len(numbers) - 1:

begin

for j from i + 1 to len(numbers) - 1:

begin

if numbers[i] + numbers[j] == 10:

begin

count = count + 1

print("Pair of 2 numbers with sum 10:", count)

print(numbers[i], "and", numbers[j])

end

end

print("Number of matched elements is:", count)

end

numbers = [2, 7, 4, 1, 3, 6]

sum\_pairs\_with\_10(numbers)

Explaination:-

In the above code it uses the 2 loops with using functions as pair of elements in input list. The outer loop iterates over all elements. Inside the inner loop it checks whether sum is equal to zero. If the pair is found counter increases.

Question -2)

Pseudo code:-

function min\_max(input\_list):

minimum\_value = input\_list[0]

maximum\_value = input\_list[0]

for i from 1 to len(input\_list) - 1:

if input\_list[i] < minimum\_value:

minimum\_value = input\_list[i]

if input\_list[i] > maximum\_value:

maximum\_value = input\_list[i]

return minimum\_value, maximum\_value

function calculate\_range(minimum\_value, maximum\_value):

return maximum\_value - minimum\_value

function determine\_range(input\_list):

if length of input\_list < 3:

print("Range determination not possible.")

return None

else:

print("Range can be found.")

minimum\_value, maximum\_value = min\_max(input\_list)

range\_value = calculate\_range(minimum\_value, maximum\_value)

print("Minimum is", minimum\_value)

print("Maximum is", maximum\_value)

print("Range is", range\_value)

return range\_value

my\_list = [5, 3, 8, 1, 0, 4]

result = determine\_range(my\_list)

Explaination:-

The min, max functions iterates from the given input list to find minimum and maximum values. The **calculate\_range** calculates the difference between minimum and maximum values.It checks whether the range is determined or not by checking input list is less than 3.

Question-3)

Psuedo code:-

function power\_finding(A, powervalue):

order = length of A

identity\_matrix = identity matrix of size order

repeat powervalue times:

identity\_matrix = matrix\_multiplication(identity\_matrix, A)

return identity\_matrix

function matrix\_multiplication(A, B):

if number of columns in A is not equal to number of rows in B:

begin

print("Cannot multiply two matrices.")

return None

end

result = empty matrix of size (number of rows in A) x (number of columns in B)

for i from 0 to (number of rows in A) - 1:

begin

for j from 0 to (number of columns in B) - 1:

begin

result[i][j] = sum of (A[i][k] \* B[k][j]) for k from 0 to (number of columns in A) - 1

return result

end

A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

m = 4

answer = power\_finding(A, m)

print(answer)

Explaination:-

The power\_finding function finds the power of the matrix by multiplying with itself using matrix\_multiplication function.The matrix\_multiplication function

checks whether it can be multiplied or not.If yes then it will perform multiplication and sum of products in each row of matrix.

Question-4)

Pseudo Code:-

function high\_occurrence(string\_input):

alphabet\_frequency = {}

for character in string\_input:

if character is an alphabet:

increment the frequency count for the character in alphabet\_frequency

maximum\_char = character with the maximum frequency in alphabet\_frequency

maximum\_occurrence = frequency of maximum\_char in alphabet\_frequency

print(f"The highest frequently occurring alphabet is '{maximum\_char}' with a count of: {maximum\_occurrence}")

given\_string = "Subrahmanya Sresti"

high\_occurrence(given\_string)

Explaination:-

The high\_occurrence function iterates through every character in each string.

It checks whether it is alphabet using isalpha() function.By using update\_frequency() function it updates the character in dictionary.After it finds the maximum of alphabest using max() function.It then prints the result, indicating the alphabet with the highest frequency and its count.

NAME : S. Hemanth Sai

REG. NO. : BL.EN.U4AIE22171

SECTION : AIE-E

**Pseudo Codes :**

**Q1)**

*function possible\_pairs(list, target\_sum):*

*count = 0*

*for i from 0 to length(list) - 1:*

*for j from i + 1 to length(list) - 1:*

*if list[i] + list[j] == target\_sum:*

*count += 1*

*return count*

**Explaination** : Code iterates through the list and checks if sum is “10”.

**Q2)**

*function calculate\_range(list):*

*if length(list) < 3:*

*return "Range determination not possible"*

*return max(list) - min(list)*

**Explaination :** Code iterates through the list and gives the *max* and *min* values as a pait (i.e., ***(max – min)*** ).

Q3)

*function multiply\_matrices(matrix1, matrix2):*

*result = []*

*for i from 0 to length(matrix1) - 1:*

*row = []*

*for j from 0 to length(matrix2[0]) - 1:*

*value = 0*

*for k from 0 to length(matrix2) - 1:*

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

*append value to row*

*append row to result*

*return result*

*function recursive\_multiplication(matrix, power):*

*if power == 0:*

*return identity matrix of the same size as matrix*

*result = matrix*

*for \_ from 0 to power - 2:*

*result = multiply\_matrices(result, matrix)*

*return result*

**Explaination** : Code consists of 2 functions : multiply\_matrices() and recursive\_multiplication() .

* multiply\_matrices() performs multiplication of 2 matrices similar to matrix multiplication in mathematics by taking row from a matrix and colomn from other matrix.
* recursive\_multiplication() performs the multiplication till the requirement is reached (Till what power is required ) .

Q4)

*function character\_frequency(input\_string):*

*char\_count = empty dictionary*

*for each char in input\_string:*

*if char is an alphabet:*

*char\_count[char] = char\_count.get(char, 0) + 1*

*frequent\_char = character with the maximum value in char\_count*

*frequency = value corresponding to frequent\_char in char\_count*

*return frequent\_char, frequency*

**Explaination** : Code performs the function by taking the string as input and iterates through the all characters in the string . It calculates the frequency of its occurrence in the character on the word and stores the data in a dictionary.

NAME : V.Bhavaharnath Sharma

REG NO.: BL.EN.U4AIE22162

SEM: 4 (AIE-E)

**QUESTION 1)**

**Pseudocode:**

function finding\_no\_of\_vowels\_and\_consonants(input\_string):

vowels = "aeiouAEIOU"

count\_of\_vowels = 0

count\_of\_consonants = 0

for char in input\_string:

if char in vowels:

count\_of\_vowels += 1

else:

count\_of\_consonants += 1

return count\_of\_vowels, count\_of\_consonants

input\_string = input("Enter the string :")

count\_of\_vowels, count\_of\_consonants = finding\_no\_of\_vowels\_and\_consonants(input\_string)

print("The no of vowels in the given string are : ", count\_of\_vowels)

print("The no of consonants in the given string are : ", count\_of\_consonants)

In the above code, Firstly I defined a function finding\_no\_of\_vowels\_and-consonants which takes a string as input from the user. It iterates through each character in the input string and checks whether the character is a vowel either it is a uppercase or lowercase and adds to the count which was initially initialised as 0.Same for the consonants. then it returns the no of vowels and consonants. Then It will ask to prompt the string ,processes using the defined function and outputs the count of vowels and consonants.

**QUESTION 2)**

function matrix\_multiplication(X, Y, Row\_X, Col\_X, Row\_Y, Col\_Y):

if Col\_X != Row\_Y:

print("Multiplication of the 2 matrices is not possible.")

return None

else:

Z = initialize\_matrix(Row\_X, Col\_Y)

for i in range(Row\_X):

for j in range(Col\_Y):

for k in range(Col\_X):

Z[i][j] += X[i][k] \* Y[k][j]

return Z

Row\_X = input("Enter the no of rows of X are :")

Col\_X = input("Enter the no of cols of X are :")

X = initialize\_matrix(Row\_X, Col\_X)

Row\_Y = input("Enter the no of rows of Y are :")

Col\_Y = input("Enter the no of cols of Y are :")

Y = initialize\_matrix(Row\_Y, Col\_Y)

resultant\_matrix = matrix\_multiplication(X, Y, Row\_X, Col\_X, Row\_Y, Col\_Y)

if resultant\_matrix is not None:

print("\n Matrix product of XY is :")

for row in resultant\_matrix:

print(row)

The above code defines a matrix multiplication function ,matrix\_multiplication which will take input matrices X and Y along with their sizes.It also checks whether those 2 matrices can be multiplied or not and if not then it will show error.Firstly It prompts user to input sizes and elements for X and y matrices from the function it will print the resulting matrix .

**QUESTION 3)**

**Pseudocode:**

function find\_common\_elements(list1, list2):

common\_elements = set(list1) intersect set(list2)

return common\_elements

list1 = [2,3,4,5,10]

list2 = [3,2,12,3,23]

common\_elements = find\_common\_elements(list1, list2)

print("Common elements between the two lists:", common\_elements)

In the above python code firstly I defined a function find\_common\_elements which takes 2 lists as input convert them to sets and find the intersection of the sets to identify common elements. Then I demonstrated it taking 2 lists list1 and list2 .Common elements are stored in the variable common\_elements and the elements are printed .

**QUESTION 4)**

**Pseudocode:**

function calculate\_transpose\_of\_matrix(matrix):

result\_matrix = initialize\_matrix(len(matrix[0]), len(matrix))

for i from 0 to len(matrix) exclusive:

for j from 0 to len(matrix[0]) exclusive:

result\_matrix[j][i] = matrix[i][j]

return result\_matrix

Matrix = [[7,8],[9,10]]

transposed\_matrix = calculate\_transpose\_of\_matrix(Matrix)

print("Original matrix is:")

for row in Matrix:

print(row)

print("\nTransposed matrix is:")

for row in transposed\_matrix:

print(row)

The above code defines a function calculate\_transpose\_of\_matrix then I took t the matrix as input which should be transposed and calculates its transpose. I initialized an empty matrix result\_matrix with sizes swapped from the input matrix. Then It iterates through each element of the matrix and assigning its transpose to the corresponding position in the resultant matrix. And Finally It will print the original and transposed matrice.