**Fall 2023: CS5720**

**Neural Networks and Deep Learning - ICP-3**

1. Create a class Employee and then do the following

• Create a data member to count the number of Employee.

• Create a constructor to initialize name, family, salary, department

• Create a function to average salary

• Create a Fulltime Employee class and it should inherit the properties of Employee class

• Create the instances of Fulltime Employee class and Employee class and call their member functions.

Solution:

class Employee:

num\_employees = 0

def \_\_init\_\_(self, name, family, salary, department):

self.name = name

self.family = family

self.salary = salary

self.department = department

Employee.num\_employees += 1

def employeeInfomation(self):

print(f"Name: {self.name}\nFamily: {self.family}\nSalary: {self.salary}\nDepartment: {self.department}")

def averagesalary(employees):

total\_salary = sum(employee.salary for employee in employees)

return total\_salary / len(employees)

class FulltimeEmployee(Employee):

def \_\_init\_\_(self, name, family, salary, department, experience\_years):

super().\_\_init\_\_(name, family, salary, department)

self.experience\_years = experience\_years

employee1 = Employee("ali", "khan", 50000, "HR")

employee2 = Employee("Janu", "jain", 60000, "IT")

fulltime\_employee1 = FulltimeEmployee("Aravind", "rao", 100000, "software", 3)

fulltime\_employee2 = FulltimeEmployee("nani", "reddy", 75000, "Finance", 5)

# Calling member functions

print("Employee 1 Info:")

employee1.employeeInfomation()

print("Fulltime Employee 2 Info:")

fulltime\_employee2.employeeInfomation()

employees = [employee1, employee2, fulltime\_employee1, fulltime\_employee2]

avg\_salary = Employee.averagesalary(employees)

print(f"Average Salary: {avg\_salary:.2f}")

print(f"Total Number of Employees: {Employee.num\_employees}")

output:

Employee 1 Info:

Name: ali

Family: khan

Salary: 50000

Department: HR

Fulltime Employee 2 Info:

Name: nani

Family: reddy

Salary: 75000

Department: Finance

Average Salary: 71250.00

Total Number of Employees: 4

1. NumPy Using NumPy create random vector of size 20 having only float in the range 1-20. Then reshape the array to 4 by 5 Then replace the max in each row by 0 (axis=1) (you can NOT implement it via for loop)

Solution:

import numpy as np

random\_value = np.random.uniform(1, 20, size=20)

reshaped\_value = random\_value.reshape(4, 5)

max\_indices = np.argmax(reshaped\_value, axis=1)

reshaped\_array[np.arange(4), max\_indices] = 0

print("Random Vector:")

print(random\_value)

print("Reshaped Array (4x5):")

print(reshaped\_array)

output:

Random Vector:

[10.5141923 3.95613767 7.07206715 4.44527515 2.18476843 4.53049059

15.8884029 9.64376066 15.47736979 5.02954002 1.71995219 8.54228073

16.25398827 11.65904842 14.1715416 12.95139663 4.4527202 15.76912325

12.18604061 3.28290015]

Reshaped Array (4x5):

[[ 0. 9.53762285 2.34718155 17.37978603 0. ]

[ 5.78366395 0. 0. 3.87459855 0. ]

[ 1.08817309 0. 0. 7.1268342 2.42019545]

[ 1.78848123 6.87159765 0. 8.20016714 0. ]]