Neural Network Deep Learning

Assignment – 3

Name: Kishor Kumar Andekar

Student ID: 700744713

Github Link : https://github.com/kishorreyansh/Neural-Network-Deep-Learning/tree/main/Assignment-3

1. Create a class Employee and then do the following
   • Create a data member to count the number of Employees
   • 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.

In the below code snippet, we are defining Employee class and sub class as FullTimeEmployee to get employee data. Employee Class has one class variable 'noOfEmployees' which initialized to zero. This variable will display total no of employees.Also, Employee class has constructor __init__ and it has 4 parameters i.e., name,family,salary and department. It initializes instance variables to store employee data and increment 'noOfEmployee' by 1 when each time new employee is created. Next, average_salary function invoked to calculate average salary of all the created employees in the list. Then, FullTimeEmployee class is created which inherits the Employee class but it doesn't add any new attributes or methods. Also, I am validating salary input whether it is digit or anything else. If it is other than digit returning 0 as salary to that employee. Next, creating one function 'create_employee' and 'create_fulltime_employee' which accepts all details of employee from console. To create Employee object based on noofemployee input iterating and creating objects and appending to the 'listofallemployees' and similar for FullTimeEmployee object as well. Finally, Printing Total no of

employees and calling average_salary function to display average salary of all the employees available in the list.

```python
#  Create a class Employee and then do the following
# • Create a data member to count the number of Employees
# • 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.

# Creating a Class Employee
class Employee:
    noOfEmployees = 0

    # creating a constructor, in python we use _init_() and this method invokes as soon as an object of a class is instantiated
    # we are passing 4 paramters and self refers to current object and binds the instance to the _init_()
    def __init__(self,name,family,salary,department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.noOfEmployees += 1
# Writing Function to get salary input from console and performing validation on salary input.
# If salary is not entered in digits It will consider as 0
def enter_salary():
    salary = input("Enter Salary: ")
    if salary.isdigit():
        return int(salary)
    else:
        return 0

```

```python

# Function for calculating avaerage salary of all employees and returning it.
def average_salary(totalemployees):
    totalsalary = sum(e.salary for e in totalemployees )
    average_salary = totalsalary / len(totalemployees)
    print("Average Salary: {:.1f}".format(average_salary))

# Creating a Class FullTimeEmployee and inheriting the properties of Employee Class
class FullTimeEmployee(Employee):
    def __init__(self,name,family,salary,department):
        Employee.__init__(self,name,family,salary,department)

# Creating List for all employees
listofallemployees = []

```

First Approach:

employee.py U × randomvector.py U

Assignments > Assignment 3 > employee.py > ...

```python
# First Approach
noofemployee = int(input("Enter the No of Employees: "))
nooffulltime_employees = int(input("Enter the No of Full Time Employees: "))
print(" ")

def create_employee(employee_type):
    return employee_type(
        input("Enter Employee Name: "),
        input("Enter Family Details: "),
        enter_salary(),
        input("Enter Department: ")
)

def create_fulltime_employee(employee_fulltime_type):
    return employee_fulltime_type(
        input("Enter Full Time Employee Name: "),
        input("Enter Family Details: "),
        enter_salary(),
        input("Enter Department: ")
)

for emp in range(noofemployee):
    print("Employee: ",emp+1)
    employee = create_employee(Employee)
    listofallemployees.append(employee)
    print(" ")

for ftemp in range(nooffulltime_employees):
    print("Full Time Employee: ",ftemp+1)
    ft_employee = create_fulltime_employee(FullTimeEmployee)
    listofallemployees.append(ft_employee)
    print(" ")
```

Second Approach:

```
# Second Approach
# Creating Employee Instance and appending it to list
# print("First Employee: ")
# employee1 = Employee(input("Enter First Employee Name: "), input("Enter Family Details: "),
#                      enter_salary(),
#                      input("Enter Department: "))
# listofallemployees.append(employee1)

# print("Second Employee: ")
# employee2 = Employee(input("Enter Second Employee Name: "), input("Enter Family Details: "),
#                      #int(input("Enter Salary: ")) if input("Enter Salary: ").isdigit() else 0,
#                      enter_salary(),
#                      input("Enter Department: "))
# listofallemployees.append(employee2)

# print("First Full Time Employee: ")
# ftemployee1 = FullTimeEmployee(input("Enter First Full Time Employee Name: "), input("Enter Family Details: "),
#                      enter_salary(),
#                      input("Enter Department: "))
# listofallemployees.append(ftemployee1)

# print("Second Full Time Employee: ")
# ftemployee2 = FullTimeEmployee(input("Enter Second Full Time Employee Name: "), input("Enter Family Details: "),
#                      enter_salary(),
#                      input("Enter Department: "))
# listofallemployees.append(ftemployee2)

# Printing Total No of Employees
print("Total Number of Employees: ", Employee.noOfEmployees)
print(" ")
```

```
# Printing Total No of Employees
print("Total Number of Employees: ", Employee.noOfEmployees)
print(" ")

# Calling Average Salary to print Average Salary of all employees
average_salary(listofallemployees)
```

Output:

employee.py U × randomvector.py U

Assignments > Assignment 3 > employee.py > ...

```python
# listofallemployees.append(ftemployee2)

# Printing Total No of Employees
print("Total Number of Employees: ", Employee.noOfEmployees)
print(" ")

# Calling Average Salary to print Average Salary of all employees
average_salary(listofallemployees)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\UCM\Kishor\Neural Network Deep Learning\Assignments\Assignment 3> python .\employee.py
Enter the No of Employees: 1
Enter the No of Full Time Employees: 2

Employee:  1
Enter Employee Name: Reyansh
Enter Family Details: R
Enter Salary: 100000
Enter Department: Doctor

Full Time Employee:  1
Enter Full Time Employee Name: Kishor
Enter Family Details: A
Enter Salary: 200000
Enter Department: IT

Full Time Employee:  2
Enter Full Time Employee Name: Ridhansh
Enter Family Details: R
Enter Salary: 300000
Enter Department: Air Force

Total Number of Employees:  3

Average Salary: 200000.0
PS D:\UCM\Kishor\Neural Network Deep Learning\Assignments\Assignment 3>
```

2. 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)

In the below Code snippet, Python program uses the NumPy library to perform operations on a randomly generated vector. First I import the NumPy library, then I Generate a random vector of 20 elements using NumPy's 'random.uniform' function, with values ranging between 1 and 20. This vector is stored in the variable 'randomVector',then I print the 'randomVector'. Next, I Reshape the 'randomVector' into a 4 by 5 matrix (a 2-dimensional array) using the 'reshape' method. This reshaped matrix is stored in the variable 'reshapeArray' and print it. Then Using NumPy functions I identify the index of the maximum value in each row of 'reshapeArray'. I do this by first using 'numpy.argmax' along 'axis=1' to find the column index with the maximum value for each row and store it in maxValue, and then uses 'numpy.arange' to generate row indices, then i replace this max value to zero and print the modified array.

employee.py U randomvector.py U ×

Assignments > Assignment 3 > randomvector.py > ...

```python
# 2. 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)

import numpy as np

# random.uniform will generate float numbers in the range of 1 to 20
# random.uniform() will generate float numbers from 0 to 1
randomVector = np.random.uniform(1,20, size=20)
print("Random Vector: ")
print(randomVector)

# we are reshaping randomvector to 4 by 5, we are passing as parameter to reshape()
reshapeArray = randomVector.reshape(4,5)
print(" ")
print("Reshape Array: ")
print(reshapeArray)

# np.argmax means we are finding max value in each row and axis=1 means it will operate on row wise (horizontal)
maxValue = np.argmax(reshapeArray,axis=1)
reshapeArray[np.arange(reshapeArray.shape[0]),maxValue] = 0
print(" ")
print("Replace Max in each row by 0: ")
print(reshapeArray)
```

Output:

employee.py U randomvector.py U

Assignments > Assignment 3 > randomvector.py > ...

```python
# 2. 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)

import numpy as np

# random.uniform will generate float numbers in the range of 1 to 20
# random.uniform() will generate float numbers from 0 to 1
randomVector = np.random.uniform(1,20, size=20)
print("Random Vector: ")
print(randomVector)

# we are reshaping randomvector to 4 by 5, we are passing as parameter to reshape()
reshapeArray = randomVector.reshape(4,5)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\UCM\Kishor\Neural Network Deep Learning\Assignments\Assignment 3> python .\randomvector.py
Random Vector:
[ 7.39809376  7.43477985 14.00874772 11.39889207 15.92633669 11.49964075
  1.31798873  5.23590087 14.56188764 13.50241482  8.89384033 19.56195323
  5.43454032 17.27079121 17.78033572 13.62375693 11.23164072  4.37848001
  2.71418075 13.85814384]

Reshape Array:
[[ 7.39809376  7.43477985 14.00874772 11.39889207 15.92633669]
 [11.49964075  1.31798873  5.23590087 14.56188764 13.50241482]
 [ 8.89384033 19.56195323  5.43454032 17.27079121 17.78033572]
 [13.62375693 11.23164072  4.37848001  2.71418075 13.85814384]]

Replace Max in each row by 0:
[[ 7.39809376  7.43477985 14.00874772 11.39889207  0.        ]
 [11.49964075  1.31798873  5.23590087  0.         13.50241482]
 [ 8.89384033  0.          5.43454032 17.27079121 17.78033572]
 [13.62375693 11.23164072  4.37848001  2.71418075  0.        ]]
PS D:\UCM\Kishor\Neural Network Deep Learning\Assignments\Assignment 3>
```