Neural Networks and Deep Learning ICP 2

Student Name: Venkata Sai Prasad Nakka

Student ID: 70075524

GitHub Link:

https://github.com/venkat137222/week-2---ICP2

Video Link:

https://drive.google.com/file/d/19-eLuVWaZ0vAuEpkt-KNfCYUMW6WdYq8/view?usp=drive_link

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

   Code:

```
class Employee:
    employees_count = 0

    def __init__(self, name, family, salary, department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.employees_count += 1

    @classmethod
    def avg_salary(cls, employees):
        total_sal = sum(employee.salary for employee in employees)
        if len(employees) > 0:
          return total_sal / len(employees)
        else:
          return 0
```

```python
class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department):
        super().__init__(name, family, salary, department)

employee1 = Employee("venkat", "nakka", 50000, "HR")
employee2 = Employee("raj", "singh", 60000, "Finance")
employee3 = Employee("zakir", "khan", 55000, "IT")
```

```python
fulltime_employee1 = FulltimeEmployee("Rakesh", "prana", 70000, "Marketing")
fulltime_employee2 = FulltimeEmployee("iqbal", "hussain", 75000, "Sales")

employees = [employee1, employee2, employee3, fulltime_employee1, fulltime_employee2]
avg_salary = Employee.avg_salary(employees)
```

```python
print(f"Employees Count: {Employee.employees_count}")
print(f"Average salary: ${avg_salary:.2f}")
```

Output:

```
Employees Count: 5
Average salary: $62000.00
```

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)

Code:

```
[24] import numpy as np

     # Creating a random vector here
     random_vector = np.random.uniform(1, 20, 20)
     print(random_vector)
```

Output:

```
[15.99806589 15.27295064  4.74027832  8.98270639 12.9471136   8.32194484
 13.73472918 11.49950801 10.88292411  5.7117342  11.8335389  17.26917611
  5.06262639 17.40646571 16.02543541 10.75579698  6.1454014  17.43614511
  9.54048354 14.15510697]
```

Code:

```
[25] # Reshaping the generated array to 4 by 5
     reshape_array = random_vector.reshape(4, 5)
     print(reshape_array)
```

Output:

```
[[15.99806589 15.27295064  4.74027832  8.98270639 12.9471136 ]
 [ 8.32194484 13.73472918 11.49950801 10.88292411  5.7117342 ]
 [11.8335389  17.26917611  5.06262639 17.40646571 16.02543541]
 [10.75579698  6.1454014  17.43614511  9.54048354 14.15510697]]
```

Code:

```
[26] # Insteading of looping, we are getting the indices of the maximum values in each row of array

     indices_position = np.argmax(reshape_array, axis=1)
     print(indices_position)
```

Output:

```
[0 1 3 2]
```

Code:

```
# Replace the maximum values with 0
# Using np.arrange function, we are generating an array of values from 0 to 3
# Using advanced indexing, here we are combing the postion indices and our newly generated indices and assigning 0:

reshape_array[np.arange(4), indices_position] = 0

print(reshape_array)
```

Output:

```
[[ 0.         15.27295064  4.74027832  8.98270639 12.9471136 ]
 [ 8.32194484  0.         11.49950801 10.88292411  5.7117342 ]
 [11.8335389  17.26917611  5.06262639  0.         16.02543541]
 [10.75579698  6.1454014   0.          9.54048354 14.15510697]]
```