In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import math
```

In [2]:

```
# define a suitable object controller
class ContinuousDatasets:
    def __init__(self, ages, fnlwgt, education_num, capital_gain, capital_loss, hours_per_w
        #constructor for ContinuousDatasets
        self.ages = ages
        self.fnlwgt = fnlwgt
        self.education_num = education_num
        self.capital_gain = capital_gain
        self.capital_loss = capital_loss
        self.hours_per_week = hours_per_week
    def getAges(self):
        return self.ages
    def getFnlwgt(self):
        return self.fnlwgt
    def getEducation_num(self):
        return self.education_num
    def getCapital_gain(self):
        return self.capital_gain
    def getCapital_loss(self):
        return self.capital_loss
    def getHours_per_week(self):
        return self.hours_per_week
```

In [3]:

```
datafile = np.loadtxt('adult.data', dtype='str', delimiter=',')
# print a record to check file loading is sucessful or not
print(datafile[0])
```

```
['39' ' State-gov' ' 77516' ' Bachelors' ' 13' ' Never-married'
 ' Adm-clerical' ' Not-in-family' ' White' ' Male' ' 2174' ' 0' ' 40'
 ' United-States' ' <=50K']
```

In [4]:

```
# load data into object to make calculation easier
all_objs = ContinuousDatasets(datafile[:,0], datafile[:,2], datafile[:,4], datafile[:,10],
data_len = len(all_objs.getAges())
print("Number of Record: "+str(data_len))
```

```
Number of Record: 32561
```

In [5]:

```
# calculating Average
def avg_calculation(data):
    #ensure that the list of data is always numeric
    data = list(map(int, data))
    age_sum = 0;
    for val in data:
        age_sum+=val
    data=False
    return age_sum/(data_len)


# for all feature call the avg_calculation function
avg_age = avg_calculation(all_objs.getAges())
avg_fnlwgt = avg_calculation(all_objs.getFnlwgt())
avg_education_num = avg_calculation(all_objs.getEducation_num())
avg_capital_gain = avg_calculation(all_objs.getCapital_gain())
avg_capital_loss = avg_calculation(all_objs.getCapital_loss())
avg_hours_per_week = avg_calculation(all_objs.getHours_per_week())
# print all values
print("Average:- ")
print("\tAge: "+str(avg_age))
print("\tFnlwgt: "+str(avg_fnlwgt))
print("\tEducation Num: "+str(avg_education_num))
print("\tCapital Gain: "+str(avg_capital_gain))
print("\tCapital Loss: "+str(avg_capital_loss))
print("\tHours-per-week: "+str(avg_hours_per_week))
```

```
Average:-
        Age: 38.58164675532078
        Fnlwgt: 189778.36651208502
        Education Num: 10.0806793403151
        Capital Gain: 1077.6488437087312
        Capital Loss: 87.303829734959
        Hours-per-week: 40.437455852092995
```

In [6]:

```
# calculating Max, Min, Median
def calculating_medianMaxMin(feature, data):
    #ensure that the list of data is always numeric
    data = list(map(int, data))
    data.sort()
    med_indx = int(data_len/2)+1
    print(feature+":-")
    print("\tMedian " + ": "+ str(data[med_indx]))
    print("\tMaximum " + ": "+ str(max(data)))
    print("\tMinimum " + ": "+ str(min(data))+"\n")

# Call functions- max, min, median
calculating_medianMaxMin("Age", all_objs.getAges())
calculating_medianMaxMin("Fnlwgt", all_objs.getFnlwgt())
calculating_medianMaxMin("Education Num", all_objs.getEducation_num())
calculating_medianMaxMin("Capital Gain", all_objs.getCapital_gain())
calculating_medianMaxMin("Capital Loss", all_objs.getCapital_loss())
calculating_medianMaxMin("Hours-per-week", all_objs.getHours_per_week())
```

```
Age:-
        Median : 37
        Maximum : 90
        Minimum : 17

Fnlwgt:-
        Median : 178370
        Maximum : 1484705
        Minimum : 12285

Education Num:-
        Median : 10
        Maximum : 16
        Minimum : 1

Capital Gain:-
        Median : 0
        Maximum : 99999
        Minimum : 0

Capital Loss:-
        Median : 0
        Maximum : 4356
        Minimum : 0

Hours-per-week:-
        Median : 40
        Maximum : 99
        Minimum : 1
```

In [7]:

```
# calculating standard deviation sqrt(sum(v[i]*v[i])/n) where i=0 to n-1
def std_dev(data, avg):
    #ensure that the list of data is always numeric
    data = list(map(int, data))
    total_val = 0
    for val in data:
        temp = val-avg
        total_val += (temp*temp)
    return math.sqrt(total_val/data_len)


# calling the standard deviation function for each continuous features
print("standard deviation:-")
print("\tAge: "+str(std_dev(all_objs.getAges(), avg_age)))
print("\tFnlwgt: "+str(std_dev(all_objs.getFnlwgt(), avg_fnlwgt)))
print("\tEducation Num: "+str(std_dev(all_objs.getEducation_num(), avg_education_num)))
print("\tCapital Gain: "+str(std_dev(all_objs.getCapital_gain(), avg_capital_gain)))
print("\tCapital Loss: "+str(std_dev(all_objs.getCapital_loss(), avg_capital_loss)))
print("\tHours-per-week: "+str(std_dev(all_objs.getHours_per_week(), avg_hours_per_week)))
```

```
standard deviation:-
        Age: 13.640223092304081
        Fnlwgt: 105548.3568808909
        Education Num: 2.5726808256012395
        Capital Gain: 7385.178676946586
        Capital Loss: 402.9540308275458
        Hours-per-week: 12.347239075706955
```