Aayush Shah
19BCE245
17 March 2021

Practical 5

Grades with Gaussian distribution

• **Definition :** Write a program which scans marks of n (scan n from the user) students in P & S. Assuming that the students are to be graded on 5 grade-scale (A, B, C, D, and F), display marks and grade of each student. Fit Gaussian distribution to the data. Repeat the exercise if students are to be graded on 7 grade-scale (A, B, C+, C, C-, D, F). Also display bar chart for both the cases.

• **Additional information :**

For more than 30 students in a course, the statistical method shall be used, with marginal adjustment for natural cut-off. The mean(X) and the standard deviation(SD) of marks obtained of all the students in a course shall be calculated and grades shall be awarded to a student depending upon the marks and the standard deviation as per table given below:

- For 5 grade scale :

  A > X+1.5SD

  X+1.0SD < B <= X +1.5 SD

  X+0.5SD < C <= X+1.0SD

  X < D <= X+0.5SD

  Other : F

- For 7 grade scale :

  A+ > X+1.5SD

  X+1.0SD < A <= X +1.5 SD

  X+0.5SD < B+ <= X+1.0SD

  X < B <= X+0.5SD

  X-0.5SD < C <= X

X-1.0SD < D <= X-0.5SD

Other : F

- **Code :**

```
1. import math
2. import random
3. import numpy as np
4. import matplotlib.pyplot as plt
5.
6. def gaussFunction(s, x, m):
7.     return (1 / (s * math.sqrt(2 * math.pi))) *
   math.exp((-1 / 2) * math.pow((x - m) / s, 2))
8.
9. def cal_grade_5scale(marks,mean,standard_deviation):
10.    if (marks > (mean + 1.5 * standard_deviation)):
11.        return 'A'
12.    elif (mean + standard_deviation < marks <= mean +
   1.5 * standard_deviation):
13.        return 'B'
14.    elif (mean + 0.5 * standard_deviation < marks <=
   mean + standard_deviation):
15.        return 'C'
16.    elif (mean < marks <= mean + 0.5 *
   standard_deviation):
17.        return 'D'
18.    else:
19.        return 'F'
20.
21.def cal_grade_7scale(marks,mean,standard_deviation):
22.    if (marks > (mean + 1.5 * standard_deviation)):
23.        return 'A+'
24.    elif (mean + standard_deviation < marks <= mean +
   1.5 * standard_deviation):
25.        return 'A'
26.    elif (mean + 0.5 * standard_deviation < marks <=
   mean + standard_deviation):
27.        return 'B+'
28.    elif (mean < marks <= mean + 0.5 *
   standard_deviation):
29.        return 'B'
```

```
30.   elif (mean - 0.5 * standard_deviation < marks <=
  mean + standard_deviation):
31.         return 'C'
32.   elif (mean - standard_deviation < marks <= mean -
  0.5 * standard_deviation):
33.         return 'D'
34.   else:
35.         return 'F'
36.
37.def cal_grade_custom(marks,z_score_ranges):
38.   grade = 'A'
39.   for grade_index in range(np.shape(z_score_ranges)
  [0]):
40.         if (z_score_ranges[grade_index][0] < marks <=
  z_score_ranges[grade_index][1]):
41.               return grade
42.         else:
43.               grade = chr(ord(grade) + 1)
44.               continue
45.   return 'F'           #by deafaul grade
46.
47.if __name__ == "__main__":
48.   n = int(input("Enter number of students : "))
49.   marks = [random.randint(0, 100) for i in range(n)]
      #generating random marks
50.
51.   mean = np.mean(marks)
52.   standard_deviation = np.std(marks)
53.
54.   grades5 = [cal_grade_5scale(mark, mean,
  standard_deviation) for mark in marks]
55.
56.   print("\n\nGRADES by 5-scale\nMARKS\tGRADES")
57.   for index in range(n):
58.         print(str(marks[index]).ljust(5)," "
  ",grades5[index].ljust(5))
59.
60.   grades7 = [cal_grade_7scale(mark, mean,
  standard_deviation) for mark in marks]
61.
62.   print("\n\nGRADES by 7-scale\nMARKS\tGRADES")
63.   for index in range(n):
```

```
64.         print(str(marks[index]).ljust(5)," 
   ",grades7[index].ljust(5))
65.
66.   total_grades = int(input("\nEnter total number of 
   grades : "))
67.   print("Enter z-score range for each grade one by one 
   : ")
68.   z_score_ranges = np.zeros(shape=(total_grades,2))
69.
70.   for index in range(total_grades):
71.         z_score_ranges[index][0] = float(input("\tEnter 
   LOWER limit of z-score for GRADE {} : 
   ".format(chr(index+65))))
72.         z_score_ranges[index][1] = float(input("\tEnter 
   UPPER limit of z-score for GRADE {} : 
   ".format(chr(index+65))))
73.         print()
74.
75.   z_score = [((mark - np.mean(marks)) / np.std(marks)) 
   for mark in marks]
76.
77.   grades_custom = [cal_grade_custom(z_score_ind, 
   z_score_ranges) for z_score_ind in z_score]
78.   print(grades_custom)
79.
80.   print("\n\nGRADES by CUSTOM\nMARKS\tGRADES")
81.   for index in range(n):
82.         print(str(marks[index]).ljust(5)," 
   ",grades_custom[index].ljust(5))
83.
84.   count5 = []
85.   count7 = []
86.   counter = 0
87.
88.   scale_5_grades = ['A','B','C','D','F']
89.   scale_7_grades = ['A','A+','B+','B','C','D','F']
90.
91.   for grade_main in scale_5_grades:
92.         counter = 0
93.         for grade_cal in grades5:
94.               if(grade_cal == grade_main):
95.                     counter+=1
96.         count5.append(counter)
```

```
97.
98.  for grade_main in scale_7_grades:
99.       counter = 0
100.      for grade_cal in grades7:
101.           if(grade_cal == grade_main):
102.                counter+=1
103.      count7.append(counter)
104.


105. plt.subplot(1,3,1)
106. plt.plot(sorted(marks) ,
     [gaussFunction(standard_deviation, mark, mean) for mark
     in sorted(marks)])
107. plt.title("Gaussian Distribution")
108. plt.xlabel("Marks")
109. plt.ylabel("Probability Density Function")
110.
111. plt.subplot(1,3,2)
112. plt.bar(['A','B','C','D','F'],count5)
113. plt.title("GRADE by scale-5")
114. plt.xlabel("Grades")
115. plt.ylabel("Number of Students")
116.
117. plt.subplot(1,3,3)
118. plt.bar(['A+','A','B+','B','C','D','F'],count7)
119. plt.title("GRADE by scale-7")
120. plt.xlabel("Grades")
121. plt.ylabel("Number of Students")
122.
123. plt.show()
```

- **Sample I/O :**

```
Enter number of students : 500


GRADES by 5-scale
MARKS   GRADES
0        F
85       B
82       B
42       F
30       F
78       C
50       F
80       B
31       F
37       F
```

Running... | CPU 0% | Memory 789.2M | cal_grade_7scale | Tabs: 4 | Line 59, Column 20

- Like this, it will show the randomly generated marks of all 500 students and corresponding grades assigned by calculation.
- The same will be applied for 7-grade scale as well.

```


GRADES by 7-scale
MARKS   GRADES
0        F
85       A
82       A
42       C
30       D
78       B+
50       C
80       A
31       D
37       C
12       F
57       B
```

Running... | CPU 0% | Memory 789.3M | cal_grade_7scale | Tabs: 4 | Line 59, Column 20

Z-score values which are entered by user.

```
Enter total number of grades : 4
Enter z-score range for each grade one by one :
    Enter LOWER limit of z-score for GRADE A : -2

    Enter UPPER limit of z-score for GRADE A : -1

    Enter LOWER limit of z-score for GRADE B : -1
    Enter UPPER limit of z-score for GRADE B : 0

    Enter LOWER limit of z-score for GRADE C : 0
    Enter UPPER limit of z-score for GRADE C : 1

    Enter LOWER limit of z-score for GRADE D : 1
    Enter UPPER limit of z-score for GRADE D : 2
```

Running... | CPU 0% | Memory 789.3M | cal_grade_7scale | Tabs: 4 | Line 59, Column 20

Graph generated :