MATH 118: Statistics and Probability **(Due: 22/06/20)**

Homework #4

Instructor: Dr. Zafeirakis Zafeirakopoulos *Name:* *Furkan ÖZEV* *Student Id:* *161044036*
Assistant: Gizem Süngü

Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- It is not a group homework. Do not share your answers to anyone in any circumstance. Any cheating means at least -100 for both sides.
- Do not take any information from Internet.
- No late homework will be accepted.
- For any questions about the homework, send an email to gizemsungu@gtu.edu.tr.
- Submit your homework (both your latex and pdf files in a zip file) into the course page of Moodle.
- Save your latex, pdf and zip files as "Name_Surname_StudentId".{tex, pdf, zip}.
- The answer which has only calculations without any formula and any explanation will get zero.
- The deadline of the homework is 22/06/20 23:55.
- I strongly suggest you to write your homework on LaTeX. However, hand-written paper is still accepted **IFF** your hand writing is **clear and understandable to read**, and the paper is well-organized. Otherwise, I cannot grade your homework.
- You do not need to write your Student Id on the page above. I am checking your ID from the file name.

Problem 1: (10+10+10+10+10+10+40 = 100 points)

WARNING: Please show your OWN work. Any cheating can be easily detected and will not be graded.

For the question, please follow the file called airplane_crashes.txt while reading the text below.

In each year from 1993 to 2012, the number of airplane crashes in airline companies were counted. The data was collected from 14 different airline companies. The numbers of crashes for the airline companies are indicated in 14 columns following the year column. Assume that the number of crashes per airline company per year is a random variable having a Poisson(λ) and that the number of crashes in different airline company or in different years are independent.
(Note: You should implement a code for your calculations for each following subproblem. You are free to use any programming languages (Python, R, C, C++, Java) and their related library.)

(a) Give a table how many cases occur for all companies between 1993 and 2012 for each number of crashes (# of Crashes).
Hint: When you check the file you will see: # of Crashes = {0, 1, 2, 3, 4}.

\# of Crashes	\# of cases in all company between the years
0	144
1	91
2	32
3	11
4	2

Table 1: Actual cases

(b) Estimate λ from the given data.
♦ The Poisson parameter Lambda is the total number of events (k) divided by the number of units (n) in the data (lambda = k/n)
♦ So, Lambda (λ) = total number of crashes / total number of cases
♦ total number of crashes = (0 * 144) + (1 * 91) + (2 * 32) + (3 * 11) + (4 * 2) = 196
♦ total number of cases = 144 + 91 + 32 + 11 + 2 = 280
So, Lambda (λ) = 196 / 280 = 0.7

(c) Update Table 1 in Table 2 with Poisson predicted cases with the estimated λ.

\# of Crashes	\# of cases in all companies between the years	Predicted \# of cases in all companies between the years
0	144	139
1	91	97
2	32	34
3	11	8
4	2	1

Table 2: Actual vs. Predicted Cases

(d) Draw a barplot for the actual cases (Table 2 in column 2) and the predicted cases (Table 2 column 3) with respect to # of crashes. You should put the figure.

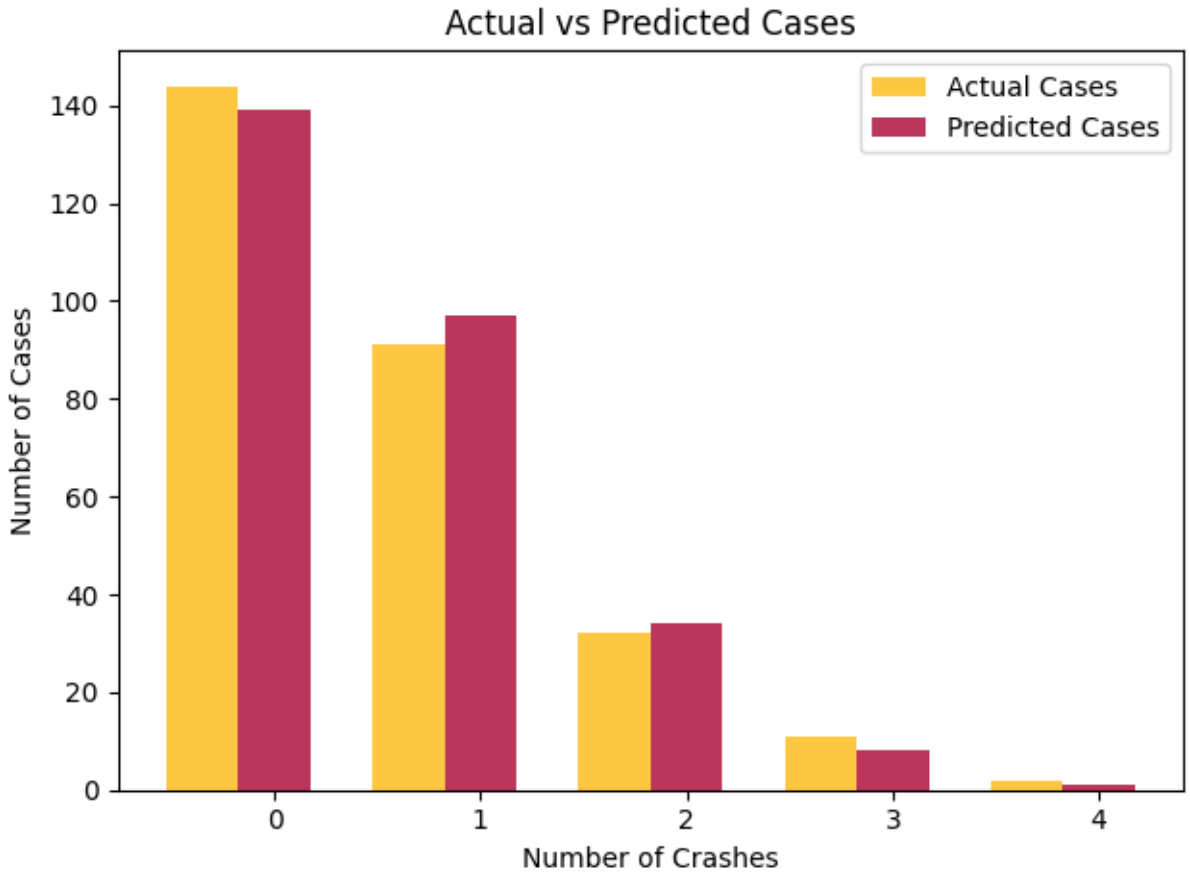

Figure 1: A barplot.

(e) According to the barplot in (c), does the poisson distribution fit the data well? Compare the values of the actual cases and the values of the poisson predicted cases, and write your opinions about performance of the distribution.

♦ To determine if the poisson distribution fits the data well, it is necessary to compare the actual cases with the predicted cases.

Total number of actual cases: 144 + 91 + 32 + 11 + 2 = 280
Total number of predicted cases: 139 + 97 + 34 + 8 + 1 = 279

As we have seen, the distribution data do not fit exactly, but we can say that it fits well enough.
Because there is no serious difference.
This difference is due to rounding to the nearest decimal.

Actual case and predicted case numbers are very close to each other.
When we evaluate in terms of performance, it can make very close prediction using Poisson distribution.
Using the Poisson distribution can be efficient in terms of creating a predicted distribution data.

(f) According to your estimations above, write your opinions considering your barplot and Table 2. Do you think that airplane transportation is dangerous for us? Whether yes or no, explain your reason.

♦ In order to decide that air transport is dangerous for us, we need to compare and interpret the numbers of real cases and predicted cases.

Based on the actual number of cases, 144 cases never had an crashes.
Based on the predicted number of cases, there will never be a crash in 139 cases.

Based on the actual number of cases, 196 crashes occurred. ((91 * 1) + (32 * 2) + (11 * 3) + (4 * 2) = 196)
Based on the predicted number of cases, 191 accidents will occur. ((97 * 1) + (34 * 2) + (8 * 3) + (1 * 2) = 191)

Based on the actual number of cases, the crash-free rate is 0.424 (144 / (144 + 196) = 0.424).
Based on the predicted number of cases, the crash-free rate is 0.421 (139 / (139 + 191) = 0.421).

As can be seen, the rate of no crash-free has decreased according to the predicted data. (0.424 > 0.421)
Therefore, we can say that the risk of crashes increased.
When we interpret according to the predicted data, we can say that airplane transportation is not safe.
So, answer is yes.

(g) Paste your code that you implemented for the subproblems above. Do not forget to write comments on your code.
Example:

- The common code block for all subproblems

Listing 1: The common code - Import modules and File operations

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

if __name__ == '__main__':

    # Open file
    file = open("airplane_crashes.txt")

    # Read all context from file
    line = file.read()

    # Close file
    file.close()
```

- The code block for (a)

Listing 2: The code block a - Compute the values in Table 1 and Print table

```

    # Create a dictionary structure to keep crashes
    cases = dict()

    # Split lines
    line = line.split('\n')

    for cline in line:

        # Split line with tab, because there is tab among the numbers.
        item = cline.split('\t')

        # For each column except first and second column
        for i in item[2:]:
            # convert string to integer number
            x = int(i)
            # If number exists in dictionary, increment by 1.
            if x in cases:
                cases[x] += 1
            # Else, add new key with value 1.
            else:
                cases[x] = 1

    # For print table 1
    table1=[["\\# of Crashes","\\# of cases in all company between the years"]]

    for i in cases:
        table1.append([i, cases[i]])

    print("\n(a)\t\t\tTABLE 1: Actual cases")
    print(tabulate(table1, headers="firstrow", tablefmt='orgtbl'))
```

- The code block for (b)

Listing 3: The code block b - Compute Lambda

```
    sum = 0
    count = 0

    for i in cases:
        # Calculate total number of cases
        count += cases[i]
        # Calculate total number of crashes
        sum += cases[i] * i

    # The Poisson parameter Lambda is the total number of events (k) divided by
    # the number of units (n) in the data ( lambda = k/n)
    # So, Lambda = total number of crashes / total number of cases
    lambdaa = sum / count

    print("\n(b) Lambda: {}\n". format(lambdaa))
```

- The code block for (c)

Listing 4: The code block c - Compute the values in Table 2 and Print table

```
    predict_case = dict()
    table2 = [["\\# of Crashes","\\# of cases in all company between the years"
            ,"Predicted \\# of cases in all companies between the years"]]

    for i in cases:
        # Poisson formula applied.
        # Then, the result was multiplied by the total number of cases.
        # round () function was used to round the result to the nearest decimal
            .
        predict_case[i] = pow(math.e, -1*lambdaa) * pow(lambdaa, i)
        predict_case[i] /= math.factorial(i)
        predict_case[i] *= count
        predict_case[i] = round(predict_case[i])

        table2.append([i, cases[i], predict_case[i]])

    # For print table 2
    print("(c)\t\t\t\t\tTABLE 2: Actual vs. Predicted Cases")
    print(tabulate(table2, headers="firstrow", tablefmt='orgtbl'))
```

- The code block for (d)

Listing 5: The code block d - Draw the barplot

```
    caselist = list()
    predictlist = list()

    # Create actual and predict list.
    for i in cases:
        # Take value from dictionary structure, and add these lists.
        caselist.append(cases[i])
        predictlist.append(predict_case[i])
```

```
10
11      # Number of crashes, like 0,1,2,3,4
12      n_groups = len(cases)
13      nlist = list()
14      for i in range(n_groups):
15          nlist.append(i)
16
17      # Create barplot for the actual cases
18      # and the predicted cases with respect to # of crashes.
19      fig, ax = plt.subplots()
20      index = np.arange(n_groups)
21      bar_width = 0.35
22      opacity = 0.8
23
24      # Give actual and predicted case lists.
25      rects1 = plt.bar(index, caselist, bar_width, alpha=opacity,
26               color='#fdb912', label='Actual Cases')
27
28      rects2 = plt.bar(index + bar_width, predictlist, bar_width, alpha=opacity,
29              color='#a90432', label='Predicted Cases')
30
31      # Determine labels.
32      plt.xlabel('Number of Crashes')
33      plt.ylabel('Number of Cases')
34      plt.title('Actual vs Predicted Cases')
35      plt.xticks(index + bar_width, nlist)
36      plt.legend()
37
38      plt.tight_layout()
39      # Show barplot
40      plt.show()
```