Homework 1: Combinatorics & Empirical Distributions

Question 1 (20 points)

A system is called “k out of n” if it functions reliably when at least k of its n components are working; in other words, the system uses redundancy to ensure robustness to failure. As an example, consider a redundant array of inexpensive disks (RAID) in which one uses n disks to store a collection of data, and as long as at least k are functioning the data can be correctly read. Suppose that disks fail independently, and that the probability of an individual disk failing in a one-year period is p.

a) *Suppose we have a n = 3 disk array which can survive one failure (k = 2). What is the expected number of disk failures in one year? As a function of p, what is the probability that the whole array will continue to function without any data loss after one year?*

Number of disk failures = X.

$$E(X) = np = 3p$$

b) *Suppose we have a n = 5 disk array which can survive two failures (k = 3). What is the expected number of disk failures in one year? As a function of p, what is the probability that the whole array will continue to function without any data loss after one year?*

Number of disk failures = X.

$$E(X) = np = 5p$$

c) *Suppose p = 0.15. Which is more reliable (has greater probability of not losing any data in one year), the RAID from part (a) or part (b)?*

Probability that Part A RAID not losing any data, where X = number of failures:

$$P(X=0) = 0.61412$$
$$P(X=1) = 0.32512$$
$$P(X \leq 1) = P(X=0) + P(X=1) = P(no\ data\ loss) = 0.93924$$

Probability that Part B RAID not losing any data, where X = number of failures:

$$P(X=0) = 0.44371$$
$$P(X=1) = 0.3915$$
$$P(X=2) = 0.13818$$
$$P(X \leq 2) = P(X=0) + P(X=1) + P(X=2) = P(no\ data\ loss) = 0.97339$$

The RAID from part (b) has a greater probability of not losing any data. Therefore the RAID from part(b) is more reliable.

d) *Suppose p = 0.55. Which is more reliable, the RAID from part (a) or part (b)?*

Probability that Part A RAID not losing any data, where X = number of failures:

$$P(X=0) = 0.09113$$
$$P(X=1) = 0.33413$$
$$P(X \leq 1) = P(X=0) + P(X=1) = P(no\ data\ loss) = 0.42525$$

Probability that Part B RAID not losing any data, where X = number of failures:

$$P(X=0) = 0.01845$$
$$P(X=1) = 0.11277$$
$$P(X=2) = 0.27565$$
$$P(X \leq 2) = P(X=0) + P(X=1) + P(X=2) = P(no\ data\ loss) = 0.40687$$

The RAID from part (a) has a greater probability of not losing any data. Therefore the RAID from part(a) is more reliable.

Question 2 (20 points)

Consider a social network that allows accounts to be secured with a 7-digit passcode (any sequence of exactly seven digits between 0-9 is valid). Assume the network has m users including you, and that all users choose one of the valid 7-digit passcodes uniformly at random. A user's passcode is considered safe if no other user has the same passcode.

a) *As a function of m, what is the probability that your own passcode is safe?*

10^7 is the number of password combinations

$$\left(1 - \frac{1}{10^7}\right)^m$$
$$= (0.9999999)^m$$

Approximately 0.9999999 chance that my own passcode is the same as someone else's passcode. Therefore the chances that my own passcode is not safe is $1 - (0.9999999)^m$

b) *How many users must there be for there to be a 50% or greater chance that your own passcode is not safe? Your answer should be a positive integer.*

$$0.5 = \left(\frac{10^7 - 1}{10^7}\right)^m$$
$$log(0.5) = log\left(\frac{10^7 - 1}{10^7}\right)^m$$
$$-0.301029996 = -4.34294504 * 10^{-8} * m$$
$$m \approx 6931471$$

Therefore, there must be 6931471 users for there to be a 50% or greater chance that my own passcode is not safe.

c) *As a function of m, what is the probability that all users have a safe passcode?*

Probability that all users have a safe passcode is the same probability that no users share a passcode, AKA every pair of passcodes is distinct.

Therefore,

$$P(D_m) = \prod_{i=0}^{m-1}\left(1 - \frac{i}{10^7}\right)$$

is the probability that all users have a safe passcode.

d) *How many users must there be for there to be a 50% or greater chance that at least one user's passcode is not safe? Your answer should be a positive integer.*

$$1 - P(D_m) = 1 - \prod_{i=0}^{m-1} \left(1 - \frac{i}{10^7}\right) \geq 0.5$$
$$= 3724$$

Therefore, there must be 3724 users for there to be a 50% or greater chance that at least one user's passcode is not safe.

Question 3 (20 points)

Consider a set of n people who are members of an online social network. Suppose that each pair of people are linked as "friends" independently with probability 1/2. We can think of their relationships as a graph with n nodes (one for each person), and an undirected edge between each pair that are friends. A clique is a fully connected subset of the graph, or equivalently a subset of people for which all pairs are friends.

a) *A clique of size 2 is simply a pair of nodes that are linked by an edge. Find the expected number of edges as a function of the number of nodes, n. What is the expected number of friend relationships among n = 25 people?*

Total number of possible friend relationships: $\binom{25}{2} = 300$
Since there is a probability of 0.5 that every pair of people are friends, we can multiply the total number of possible friend relationships by this probability: $300 * 0.5 = 150$.

Therefore, the expected number of friend relationships among n=25 people is 150.

b) *A clique of size 3 is a triplet of nodes within which all three pairs are linked by an edge. Find the expected number of 3-cliques as a function of the number of nodes, n. What is the expected number of 3-cliques among n = 25 people?*

Total number of possible cliques: $\binom{25}{3} = 2300$

Since a 3-clique has a $0.5 * 0.5 * 0.5 = 0.125$ chance of occuring (three nodes connected by an edge), we can multiply this with the total number of possible cliques: $2300 * 0.125 = 287.5$

Therefore, the expected number of 3-cliques among n=25 people is 287.5.

c) *Larger cliques may occur involving groups of nodes of any size k. Derive a general formula for the expected number of cliques of any size 2 ≤ k ≤ n as a function of the number of nodes, n. What is the expected number of cliques of size k = 4 among n = 25 people?*

General formula: $0.5^{\binom{k}{2}} * \binom{n}{k}$

Plugging this in to our general formula:

$$0.5^{\binom{4}{2}} * \binom{25}{4}$$

$$0.5^6 * 12650 = 197.65625$$

Therefore, the expected number of 4-cliques among n=25 people is 197.65625.

Question 4: (35 points)

We will now analyze some data collected by observing the famous “Old Faithful” geyser in Yellowstone National Park. We define random variable S to be the time an eruption lasts, and random variable T to be the “waiting time” until the next eruption. These are clearly continuous random variables, but we do not precisely know their true distribution. Instead we have a dataset with $n = 272$ independent observations $(s_i, t_i), i = 1, \ldots, 272$, of the eruption time s_i and subsequent waiting time t_i. See Figure 1 for a plot of this data.

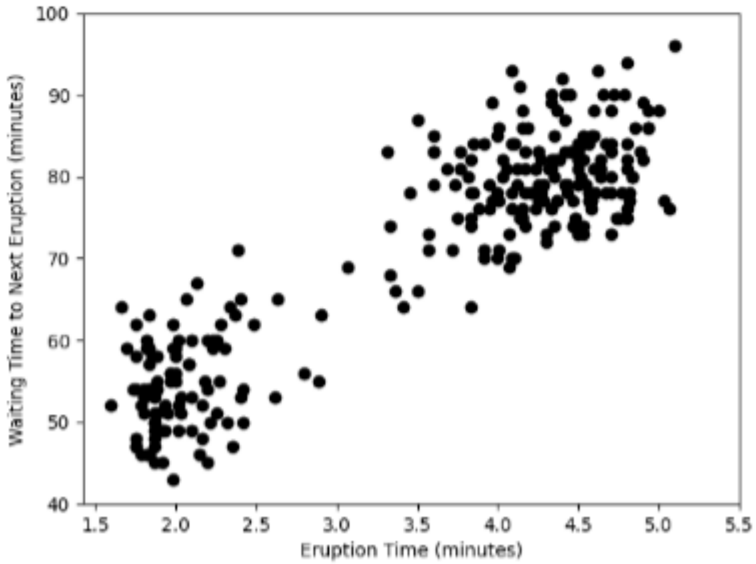

Figure 1: A “scatter plot” of the observations of Old Faithful’s eruption time (horizontal axis) and waiting time to the next eruption (vertical axis). Each point is one of the $n = 272$ observations.

In the following questions, we compute various quantities using the empirical distribution of the data. The empirical distribution of eruption time and waiting time can be represented by a probability mass function $p_S T(s,t)$ which places probability $1/n$ on each of the n data points, and probability 0 on the continuous range of other (s,t) values. Under this distribution, the expected values of S and T then take the following simple form:

In [108…

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

# Load data
S = np.load('eruptions.npy')  # vector of observed eruption times
T = np.load('waiting.npy')    # vector of observed waiting times
n = S.shape[0]                # number of observations
```

a) *The variance of random variable S equals $Var[S] = E[S^2] - E[S]^2$. Give formulas for computing $Var[S]$ and $Var[T]$ under the empirical distribution. Use Python's `numpy.sum` function to write your own code that computes these variances, and report their values.*
Hint: Various definitions of the "sample variance" can be found in statistics references, and they are not all equivalent to the variance of the empirical distribution.

In [109…

```
# Var S
var_S = (np.sum(np.square(S)) / n) - np.square((np.sum(S) / n))
print(var_S)

# Var T
var_T = (np.sum(np.square(T)) / n) - np.square((np.sum(T) / n))
print(var_T)
```

```
1.2979510049735055
184.15571388000717
```

Therefore,

$$Var[S] = E[S^2] - E[S]^2 = 1.298$$
$$Var[T] = E[T^2] - E[T]^2 = 184.156$$

b) *The cumulative distribution of S equals $F_S(s) = P(S \leq s)$, where the probability is under the empirical distribution. Find eruption times $\bar{s}_1, \bar{s}_2, \bar{s}_3$ such that $F_S(\bar{s}_1) = 0.25$, $F_S(\bar{s}_2) = 0.50$, $F_S(\bar{s}_3) = 0.75$. Using the cumulative distribution of T, also find waiting times $\bar{t}_1$, $\bar{t}_2$, $\bar{t}_3$ such that $F_T(\bar{t}_1) = 0.25$, $F_T(\bar{t}_2) = 0.50$, $F_T(\bar{t}_3) = 0.75$.* Hint: One solution would be to use Python's `numpy.sort` function.

In [110…

```
sorted_S = np.sort(S)

s_bar_1 = sorted_S[len(S) // 4 - 1]
print(s_bar_1)

s_bar_2 = sorted_S[len(S) // 2 - 1]
print(s_bar_2)

s_bar_3 = sorted_S[len(S) // 4 * 3 - 1]
print(s_bar_3)

sorted_T = np.sort(T)

t_bar_1 = sorted_T[len(T) // 4 - 1]
print(t_bar_1)

t_bar_2 = sorted_T[len(T) // 2 - 1]
print(t_bar_2)

t_bar_3 = sorted_T[len(T) // 4 * 3 - 1]
print(t_bar_3)
```

```
2.1507
4.0005
4.4503
58.0067
76.0035
82.0003
```

Therefore,

$$\bar{s}_1 = 2.1507,\ \bar{s}_2 = 4.0005,\ \bar{s}_3 = 4.4503$$
$$\bar{t}_1 = 58.0067,\ \bar{t}_2 = 76.0035,\ \bar{t}_3 = 82.003$$

Consider two new random variables. Let X indicate whether the eruption time S is “short” or “long”: $X = 0$ if $S \leq 3.5$, and $X = 1$ if $S > 3.5$. Let Y indicate whether the waiting time T is “short” or “long”: $Y = 0$ if $T \leq 70$, and $Y = 1$ if $T > 70$.

c) *Using the empirical distribution of* S *and* T*, determine and report the joint probability mass function* $p_{XY}(x, y)$*. Also determine and report the marginal probability mass functions* $p_X(x)$ *and* $p_Y(y)$*.*

In [111…

```
X = np.copy(S)
X[X <= 3.5] = 0
X[X > 3.5] = 1
# print(X)

Y = np.copy(T)
Y[Y <= 70] = 0
Y[Y > 70] = 1
# print(Y)

mar_X = np.empty(2)
mar_X[0] = np.count_nonzero(X==0) / len(X)
mar_X[1] = np.count_nonzero(X==1) / len(X)
print(f'Marginal probability mass function pX: {mar_X}')

mar_Y = np.empty(2)
mar_Y[0] = np.count_nonzero(Y==0) / len(Y)
mar_Y[1] = np.count_nonzero(Y==1) / len(Y)
print(f'Marginal probability mass function pY: {mar_Y}')
print()
print('The 0 index in the marginal probability mass function arrays are where the variable = 0.')
print('The 1 index in the marginal probability mass function arrays are where the variable = 1.')
print()

p_XY = np.empty(shape=(2,2))
p_XY[0, 0] = (np.count_nonzero(X==0) / len(X)) * (np.count_nonzero(Y==0)) / len(Y)
p_XY[0, 1] = (np.count_nonzero(X==0) / len(X)) * (np.count_nonzero(Y==1)) / len(Y)
p_XY[1, 0] = (np.count_nonzero(X==1) / len(X)) * (np.count_nonzero(Y==0)) / len(Y)
p_XY[1, 1] = (np.count_nonzero(X==1) / len(X)) * (np.count_nonzero(Y==1)) / len(Y)
print(f'Joint probability mass function pXY: \n{p_XY}')
print()
print('The 0,0 index is where x = 0 and y = 0')
print('The 0,1 index is where x = 0 and y = 1')
print('The 1,0 index is where x = 1 and y = 0')
print('The 1,1 index is where x = 1 and y = 1')
print()
```

```
Marginal probability mass function pX: [0.38235294 0.61764706]
Marginal probability mass function pY: [0.37867647 0.62132353]

The 0 index in the marginal probability mass function arrays are where the variable = 0.
The 1 index in the marginal probability mass function arrays are where the variable = 1.

Joint probability mass function pXY:
[[0.14478806 0.23756488]
 [0.23388841 0.38375865]]

The 0,0 index is where x = 0 and y = 0
The 0,1 index is where x = 0 and y = 1
The 1,0 index is where x = 1 and y = 0
The 1,1 index is where x = 1 and y = 1
```

d) *Are the random variables X and Y independent? If not, is the amount of dependence weak or strong? Clearly justify your answer using the probability mass functions from (c).*

To check independence, we can check if $P_{XY}(y=1|x=0) = P_Y(y=1)$: $.23756488 \neq .62132353$. Therefore, the variables X and Y are not independent.

To check for weak dependence, we can use the formula $\frac{p_{XY}(x,y)}{(p_X(x)*p_Y(y))}$, and check if it approximately equals 1. Using the formula, we get 4.234 (see work below in the code cell), meaning that the relationship between X and Y is **strongly dependent**.

In [112…

```
print(np.sum(p_XY / (mar_X * mar_Y)))
```

```
4.2344322344322345
```