SCC.363 Security and Risk Coursework 2

Academic Honesty and Integrity

Students at Lancaster University are part of an academic community that values trust, fairness and respect and actively encourages students to act with honesty and integrity. It is a University policy that students take responsibility for their work and comply with the university's standards and requirements- found in the Manual of Academic Regulations and Practice. By submitting their answers students will be confirming that the work submitted is completely their own. Academic misconduct regulations are in place for all forms of assessment and students may familiarise themselves with this via the university website:
https://www.lancaster.ac.uk/academic-standards-and-quality/regulations-policies-and-committees/manual-of-academic-regulations-and-procedures/

Plagiarism

Plagiarism involves the unacknowledged use of someone else's work and passing it off as if it were one's own. This covers every form of submitted work, from written essays, video vignettes, and coding exercises. However, deliberately plagiarism with the intent to deceive and gain academic benefit is unacceptable. This is a conscious, pre-meditated form of cheating and is regarded as a serious breach of the core values of the University. More information may be found via the plagiarism framework website. All coursework is to be submitted electronically and will be run through our plagiarism detection mechanisms. Please ensure you are familiar with the University's Plagiarism rules and if you are in any doubt please contact your module tutor.
https://www.lancaster.ac.uk/academic-standards-and-quality/regulations-policies-and-committees/principles-policies-and-guidelines/plagiarism-frame

This is an individual assignment that will count for 30% of your overall marks for this module. The marks will be based on the correctness of the solution.

Learning objectives

- Develop appreciation and understanding of security tools.
- Formulate troubleshooting methods to identify/solve problems.
- Evaluate information to critically argument solution choices.
- Effectively communicate ideas.

Submission requirements

Prepare and submit your coding solutions on Coderunner. For all coding solutions, you must use Python3. You can use modules from standard Python3. Your code should include appropriate comments explaining what you do and why.

Marking guidelines

Task 1. Weight 30% of total marks

- Marks will be allocated based on the correctness of your solution.

Task 2. Weight 30% of total marks

- Marks will be allocated based on the correctness of your solution.

Task 3. Weight 40% of total marks

- Marks will be allocated based on the correctness of your solution.

Deadline for submissions: Friday 15th March 16:00

Task 1 – Calculation of annualized loss expectancy

The annualized loss expectancy (ALE) is an important metric to quantify the exposure of an asset caused by certain risks. It is equal to the product of the annual rate of occurrence (ARO) and the single loss expectancy (SLE). The ARO is an estimate of how often a risk would be successful in exploiting a vulnerability. The SLE is the monetary value expected from the occurrence of a risk on an asset and it is equal to the product of the typical asset value (AV) and the exposure factor (EF), where the EF represents the impact of the risk over the asset or percentage of asset lost. In conclusion of the above, we have the following two relationships:

$$\mathrm{ALE} = \mathrm{ARO} \times \mathrm{SLE} \qquad (\mathrm{Eq.}\ 1)$$

$$\mathrm{SLE} = \mathrm{AV} \times \mathrm{EF} \qquad (\mathrm{Eq.}\ 2)$$

Write a program to implement the following.

(1) Consider a specific cyber risk on an asset. Assume that the AV follows a triangular distribution with lower limit $\boldsymbol{a}$, upper limit $\boldsymbol{b}$, and mode $\boldsymbol{c}$.

(i) Find the probability $\mathbf{prob1}$ that the AV is no greater than $\mathbf{point1}$, where $\boldsymbol{a} \leq \mathbf{point1} \leq \boldsymbol{b}$.

(ii) Find the mean $\mathbf{MEAN\_t}$ and median $\mathbf{MEDIAN\_t}$ of the AV.

(2) We have collected historical data of numbers of annual occurrences and generated the following table

number	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9
probability	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9

Calculate the mean $\mathbf{MEAN\_d}$ and variance $\mathbf{VARIANCE\_d}$ of the numbers of annual occurrences.

(3) The Monte Carlo method has been widely used in risk analysis to generate draws from a probability distribution. It relies on repeated random sampling to obtain numerical results. Suppose that two flaws, A and B, are considered in the above problem. Each flaw will cause certain amount of impact. In particular, the impact caused by flaw A follows the log-normal distribution with $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The impact caused by flaw B follows the Pareto distribution with $\boldsymbol{x_m}$ and $\boldsymbol{\alpha}$. The total impact is the sum of the impacts caused by the two flaws. Use Monte Carlo method to simulate the probability distribution of the total impact. Specifically:

(i) Randomly sample $\mathbf{num}$ points for the total impact.

(ii) Based on your sampling points, derive the probability $\mathbf{prob2}$ that the total impact is greater than $\mathbf{point2}$.

(iii) Based on your sampling points, derive the probability $\mathbf{prob3}$ that the total impact is between $\mathbf{point3}$ and $\mathbf{point4}$, where $\mathbf{point3} < \mathbf{point4}$.

(4) Assume that $\mathbf{MEDIAN\_t}$ derived in (1)-(ii) is used as the AV in (Eq. 2), $\mathbf{MEAN\_d}$ derived in (2) is used as the ARO in (Eq. 1), and $\mathbf{prob2}$ derived in (3)-(ii) is used as the EF in (Eq. 2). Calculate the value of the **ALE**.

Please use the following code

```
from dhCheck_Task1 import dhCheckCorrectness
def Task1(a, b, c, point1, number_set, prob_set, num, point2,
mu, sigma, xm, alpha, point3, point4):
    # TODO
    return (prob1, MEAN_t, MEDIAN_t, MEAN_d, VARIANCE_d, prob2, 
prob3, ALE)
```

Note: do not truncate the results!

Inputs format:

number_set = [N0, N1, N2, N3, N4, N5, N6, N7, N8, N9]
prob_set = [P0, P1, P2, P3, P4, P5, P6, P7, P8, P9]

Reading:
Annualized loss expectancy: https://en.wikipedia.org/wiki/Annualized_loss_expectancy
Single-loss expectancy: https://en.wikipedia.org/wiki/Single-loss_expectancy
Triangular distribution: https://en.wikipedia.org/wiki/Triangular_distribution
Log-normal distribution: https://en.wikipedia.org/wiki/Log-normal_distribution
Pareto distribution: https://en.wikipedia.org/wiki/Pareto_distribution
Monte Carlo method: https://en.wikipedia.org/wiki/Monte_Carlo_method

Task 2 – Probability theory

A security scanning process has two phases, finding (denoted by random variable X) and fixing (denoted by random variable Y). The process is implemented on **num** cases and the joint distribution of the times required to complete the phases are provided by the following table. In particular, the second row lists the possible days for X, and the second column lists the possible days for Y. For each combination of the values of X and Y, the number at the intersection block is the number of occurrences of this combination. For example, among the **num** cases, there are ***a*** cases that take X 2 days and Y 6 days to complete.

		X			
		2	3	4	5
Y	6	a	b	c	d
	7	e	f	g	h
	8	i	j	k	l

(1) Calculate the probability **prob1** of $3 \leq X \leq 4$ and the probability **prob2** of $X+Y \leq 10$.

(2) Another screening procedure was also tested on the **num** cases. Denote by T the event of being tested positive. The following probabilities were obtained: P(T|X=2) = PX2, P(T|X=3) = PX3, P(T|X=4) = PX4, P(T|X=5) = PX5, P(T|Y=6) = PY6, and P(T|Y=7) = PY7. Find the probability **prob3** of Y=8 given that a case is tested positive.

Please use the following code

```
from dhCheck_Task2 import dhCheckCorrectness
def Task2(num, table, probs):
    # TODO

    return (prob1, prob2, prob3)
```

Note: do not truncate the results!

Inputs format:

probs = [PX2, PX3, PX4, PX5, PY6, PY7]
table = [[a, b, c, d], [e, f, g, h], [i, j, k, l]]

Reading:
Bayes' theorem: https://en.wikipedia.org/wiki/Bayes%27_theorem

Task 3 – Linear regression and linear programming

A company has been using four types of security controls to protect its computer network. To determine a better deployment, the company would like to first understand the safeguard effects and maintenance loads of the security controls. Assume that both the total safeguard effect y and the total maintenance load z are linear functions of the numbers of the security controls and can be expressed as $y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$ and $z = d_0 + d_1x_1 + d_2x_2 + d_3x_3 + d_4x_4$, respectively, where x_i is the number of the i-th type of security controls applied, b_i is the associated weight which represents the unit effect, and d_i is the associated weight which represents the unit load. The company has collected 9 historical pairs of joint input $\boldsymbol{x} = [x_1, x_2, x_3, x_4]$ and outputs $\boldsymbol{y}$ and $\boldsymbol{z}$, as listed in the following table.

y	z	x1	x2	x3	x4
y1	z1	x11	x21	x31	x41
y2	z2	x12	x22	x32	x42
y3	z3	x13	x23	x33	x43
y4	z4	x14	x24	x34	x44
y5	z5	x15	x25	x35	x45
y6	z6	x16	x26	x36	x46
y7	z7	x17	x27	x37	x47
Y8	z8	x18	x28	x38	x48
y9	z9	x19	x29	x39	x49

(1) Based on the table, use linear regression to derive the underlying **weights_b** $(b_0, b_1, b_2, b_3, b_4)$ and **weights_d** $(d_0, d_1, d_2, d_3, d_4)$.

(2) The company now aims to enhance its safeguard. The current deployment of the security controls is x_{01}, x_{02}, x_{03}, and x_{04} (**x_initial**). The company plans to enhance the total safeguard effect to be no less than **se_bound**. Meanwhile, the total maintenance load needs to be no greater than **ml_bound**. In addition, there is an upper bound for the number of each type of security controls. Under these constraints, the company aims to minimize the total cost. The cost (**c**) and number bound (**x_bound**) for each type of security controls are given in the table below. The safeguard effect and the maintenance load are the ones derived in (1). Use linear programming to find how many *additional* security controls (**x_add**) for each type should be deployed. (Notice that the company already has a base deployment and wants to add more security controls to strengthen the safeguard.)

Index	1	2	3	4
Cost	c1	c2	c3	c4
Number bound	nb1	nb2	nb3	nb4

Please use the following code

```
from dhCheck_Task3 import dhCheckCorrectness
def Task3(x, y, z, x_initial, c, x_bound, se_bound, ml_bound):
    # TODO
    return (weights_b, weights_d, x_add)
```

Note:

(i) **x_add must be the additional security controls, not the total security controls.**

(ii) **You may have to use the dot operation to extract x_add from the solution returned by linprog.**

(iii) **Do not truncate or round x_add to integers.**

Inputs format:

x =
[[x11,x12,x13,x14,x15],[x21,x22,x23,x24,x25],[x31,x32,x33,x34,x35],[x41,x42,x43,x44,x45]]
y = [y1,y2,y3,y4,y5]
z = [z1,z2,z3,z4,z5]
x_initial = [x01,x02,x03,x04]
c = [c1,c2,c3,c4]
x_bound = [nb1,nb2,nb3,nb4]

Output format:

weights_b, weights_d and x_add must all be of type array

Reading:
Linear regression: https://en.wikipedia.org/wiki/Linear_regression
Linear programming: https://en.wikipedia.org/wiki/Linear_programming