CENG 222

Probability and Statistics

Term Project for 2022-2023 Spring

DUE DATE: 26.06.2023 - 23:59

In this project, you will analyze the relationship between five random variables in a healthcare context: exercise frequency (**A**), caloric intake (**B**), patient category (**C**), cholesterol level (**D**), blood pressure (**E**), and Hemoglobin A1c (HbA1c) level (**H**). For this analysis, you will first implement algorithms to generate those variables and create a population. Then by sampling from this population, you will analyze their statistics, estimate different distribution parameters, compute confidence intervals, test a hypothesis, and automatically classify patients into patient categories by looking at their D, E, and H values.

You will also produce relevant plots for each step to help you understand and interpret the exercises and check your results. In the meantime, you will calculate or derive estimators for the unknown distribution parameters required for implementations.

In conclusion, your project submission will be in two folds: the implementation part for which a single Python script is expected and the reporting part for which a pdf file is expected. The project is broken down into many small steps, and if it is in the implementation part, a step is marked with **red** and otherwise with **blue**. Only 3 packages are required for implementation: **numpy**, **sympy**, and **pyplot**.

The random variables' details, distributions, and relations are as follows:

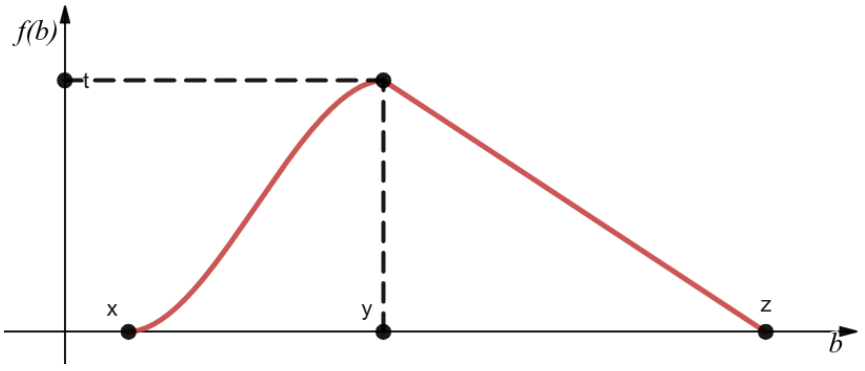
- **Exercise frequency (A):** This represents how often a patient engages in physical exercise per week. Patients who exercise more frequently are generally associated with better health outcomes.

  A has infinitely many possible values {0, 1, 2,…} and $P(a) = 0.5^{a+1}$.

- **Caloric intake (B):** This refers to the kilocalories the patient consumes daily. Higher caloric intake may lead to weight gain and potential health risks if not balanced with exercise and a healthy diet.

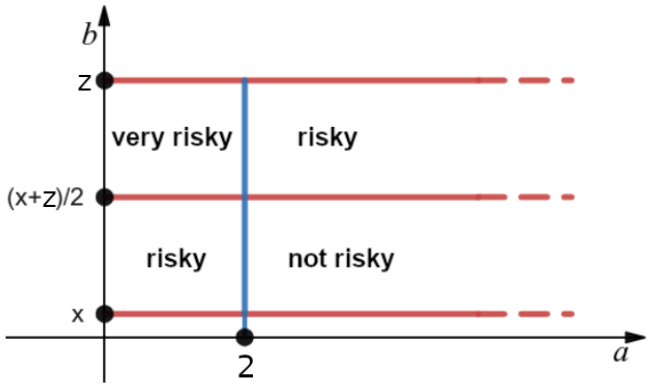
  B is a continuous random variable with the given probability density function:

$$f(b) = -0.096b^3 + 0.432b^2 - 0.352b + 0.08,\ x \le b \le y$$
$$f(b) = (-2b + 11)/15,\ y \le b \le z$$
$$f(b) = 0,\ elsewhere$$



- **Patient category (C):** The patient category is determined based on exercise frequency (a) and caloric intake (b), and this categorization helps assess the patient's overall health risk based on their fitness and dietary habits.

  The categories include "**not risky**," "**risky**," and "**very risky**," and how they are determined is shown in the graph below:



- **Cholesterol level (D):** Cholesterol level (mg/dL) measures the amount of cholesterol in the patient's body. Higher-risk patients tend to have higher cholesterol levels.

  D is a continuous random variable with a Normal distribution. It has different parameters for different patient categories:

category	very risky	risky	not risky
μ	240	200	160
σ	50	70	30

- **Systolic blood pressure (E):** Blood pressure indicates the pressure exerted on the arterial walls (100mmHg). Higher-risk patients tend to have elevated blood pressure.

  E is a continuous random variable with the following cumulative distribution function:

$$F(e; i, j) = (e - i)^2 - j \; for \; (i + \sqrt{j}) \leq e \leq (i + \sqrt{j + 1})$$

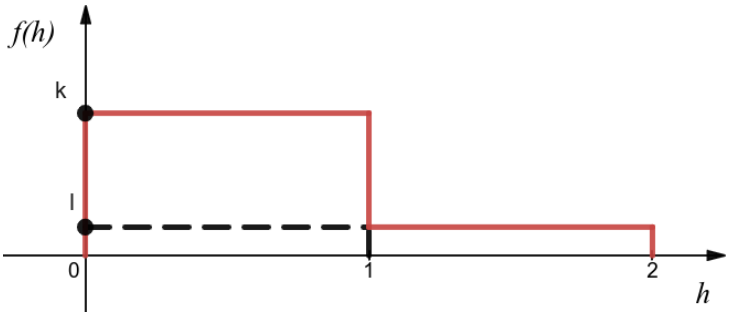
  It has different parameters for different patient categories:

category	very risky	risky	not risky
i	0.9	0.5	0.1
j	0.2	0.5	0.8

- **Hemoglobin A1c (HbA1c) level (H):** Hemoglobin A1c (HbA1c) level is a specific medical measurement used to assess long-term blood glucose control in patients with diabetes. Patients in higher-risk categories tend to have higher HbA1c values, indicating suboptimal blood glucose control and a greater risk for diabetes-related complications.

  H is a continuous random variable with the following probability density function:

$$f(h) = k, \; 0 \leq h \leq 1$$
$$f(h) = l, \; 1 < h \leq 2$$
$$f(h) = 0, \; elsewhere$$



It has a different k parameter for different patient categories:

category	very risky	risky	not risky
k	0.1	0.4	0.7

TASK 1: Generating the population

Your first task is to generate synthetic patient data.

Firstly, you must calculate the unknown distribution parameters and implement the generator functions. You are **advised** to use online equation solvers or integral calculators such as Wolfram Alpha or Symbolab in your calculations. Just **do not forget** to include each step in detail in your report. In your implementations, you are **not allowed** to use any number generator function other than `numpy.random.random` unless stated otherwise.

Next, you must generate the population data for each random variable using its generator function. To have a better insight into the population and check if you have implemented everything correctly, you should plot the population histograms and probability mass/density functions (PMF/PDF).

Here are the detailed steps:

1. Implement a function (**pmf_a**) to compute the PMF of A.
2. Looking at the PMF of A, comment on which distribution family it looks like. Let's say a random variable X is from that distribution family; how can you define A in terms of X?
3. Using your findings in step 2, implement a function (**generate_a**) to generate A.
4. For B, find the values of x, y, z, and t.
5. Implement a function (**pdf_b**) to compute the PDF of B.
6. Implement a function (**generate_b**) to generate B using the rejection method.
7. Implement a function (**calculate_c**) to calculate C using generated A and B values. You can enumerate C if you like.
8. Implement a function (**pdf_d**) to compute the PDF of D.
9. Implement a function (**generate_d**) to generate D for the calculated C values using NumPy's normal distribution function: **numpy.random.normal**.
10. Calculate the PDF of E.
11. Implement a function (**pdf_e**) to compute the PDF of E.
12. Implement a function (**generate_e**) to generate E using the inverse transformation method and the calculated C values.
13. For PDF of H, define the parameter l in terms of k. Rewrite the PDF with a single parameter.
14. Based on step 13, implement a function (**pdf_h**) to compute the PDF of H.
15. Implement a function (**generate_h**) to generate H using the rejection method.

16. Implement a function (**generate_population**) that takes the number of patients (W) as input and generates and returns a population of size (Wx6) using the functions implemented above. In other words, for W many patients, it should first generate A and B values and then calculate the C values for them. Then based on the patients' computed C values, it should generate D, E, and H values. Finally, it should also plot the following:
    a. In the same figure, plot the population histogram for A and the pmf of A.
    b. In the same figure, plot the population histogram for B and the pdf of B.
    c. Plot the population histogram for C and print the pmf of C estimated using the generated population.
    d. In the same figure, plot the population histogram for D and the joint pdf of D with each possible value of C.
    e. In the same figure, plot the population histogram for E and the joint pdf of E with each possible value of C.
17. Generate a population of 1000 patients. It should be a 2D array of shape 1000x6 where the random variables appear in the order of A, B, C, D, E, and H.
18. Similar to step 17, generate a population of 1000000 patients. **This will be your hypothetical population to be analyzed in the next tasks.** Compute and print the random variable means and variances for this population using **numpy.mean** and **numpy.var**.
19. Copy your figures from steps 17 and 18 into your report so that the two plots for each variable are side by side. You should observe that when the number of generated patients increases, the population statistics converge to the actual PMF/PDF.

TASK 2: Sampling and Descriptive Statistics

Your second task is to take samples from your population and compute and visualize some descriptive statistics for each sample. If you fail to accomplish Task 1, you can load the **population.txt** file provided along with this assignment as a NumPy array using `numpy.loadtxt`.

Firstly, you have to implement the sampler function to take samples from the population. You are **not allowed** to use any number generator function other than `numpy.random.random`. Next, you must implement estimators for the population mean and variance. You are **not allowed** to use **numpy.mean**, **numpy.std**, or **numpy.var** functions. To have a better insight into the performances of these estimators, you should take multiple samples to estimate from and plot the histograms of the estimations with indicators for the actual population mean and variance.

Here are the detailed steps:

1. Implement a function (**random_sample**) that takes a population and a sample size (n) as inputs and returns a random sample of that population with dimensions of nx6 where the random variables appear in the same order as the population.

   For the next steps, a column of that population sample (nx1) will be referred to as a **sample**. The first column is a sample of A; the second column is a sample of B, and so on.
2. Implement a function (**estimate_mean**) that estimates the population mean of a variable with its sample mean. It should take a sample and return an estimation for the mean.
3. Implement a function (**estimate_variance_1**) that estimates the population variance of a variable using its sample with the formula below. It should take a sample and return an estimation for the variance.

$$s^2 = \sum_{i=1}^{N} (X_i - \overline{X})^2 / (n - 1)$$

4. Implement a function (**estimate_variance_2**) that estimates the population variance for a variable using its sample with the formula below. It should take a sample and return an estimation for the variance.

$$s^2 = \sum_{i=1}^{N} (X_i - \overline{X})^2 / n$$

5. Implement a function (**descriptive_stats**) that takes a population, sample size (n), number of sampling (m), and variable index as input. It returns lists of estimated means and variances of type 1 and type 2 using the previously implemented functions. In other words, it should take a sample of size n from the population m times and create descriptive statistics lists of length m for the variable at the given index.
6. Calculate the theoretical mean and variance of variable A.
7. Call the descriptive_statistics function for variable A with the sample size as 100 and the number of samples as 10000.
    a. In the same figure, plot the histogram of the estimated means, a vertical line at the average of the estimated means, and a vertical line at the theoretical mean.
    b. In the same figure, plot the histogram of the estimated variances of type 1, a vertical line at the average of the estimated variances of type 1, and a vertical line at the theoretical variance.
    c. In the same figure, plot the histogram of the estimated variances of type 2, a vertical line at the average of the estimated variances of type 2, and a vertical line at the theoretical variance.
8. Copy your figures from step 7 into your report. What do the figures analyze? What is your conclusion?
9. Call the descriptive_statistics function for variable A with the sample size as (100, 200, 300, …, 5000) and the number of samples as 500. Compute the variances of the estimated means and the estimated variances of type 1 for each sample size and create the sample size v.s. variance plot for both of them in the same figure.
10. Copy your figure from step 9 into your report. What does this figure analyze? What is your conclusion?

TASK 3: Parameter Estimation

Your third task is to estimate some of the distribution parameters using the Method of Moments and the Method of Maximum Likelihood.

Here are the detailed steps:

1. For variable D, search online and find out about the Maximum Likelihood estimation of the normal distribution parameters. Add these findings to your report.
2. Implement a function (**estimate_mml_d**) that takes a sample of D and returns the Method of Maximum Likelihood estimations for μ and σ. You can use previously implemented functions.
3. For variable E, calculate the estimators for its parameters i and j using the Method of Moments. As a result of your calculations, you should reach a point where you have $f(j) = 0$

equation such that f is a function with fractional exponents and g(j)=i where g is another function to calculate i using j.

4. Implement a function (**estimate_mom_e**) that takes a sample of E and returns the Method of Moments estimations for i and j. You should use **sympy**'s solver to find j using the following code where you have to replace f(j) with your function expression:

```
j = sympy.S('j')
sol = sympy.solve(f(j), j)
  ind = 0
  while True:
    j = float(sol[ind].as_real_imag()[0])
    ind += 1
    if j >= 0:
      break
```

After this point, you should be able to compute i using g(j).

5. For variable H, compute the estimator for its parameters k and l using the Method of Maximum Likelihood.
6. Implement a function (**estimate_mml_h**) that takes a sample of H and returns the Method of Maximum Likelihood estimations for k.
7. Implement a function (**parameter_estimation**) that takes a population, sample size (n), the number of sampling (m) and patient category as input. Using the previously implemented functions, it samples from a subset of the population where the patient is in the given category and returns lists of estimated parameters of variables D, E, and H.
8. Call the parameter_estimation function for the "very risky" patient category with the sample size as 100 and the number of samples as 1000.
    a. In the same figure, plot the histogram of the estimated i parameter of E, a vertical line at the average of the estimations, and a vertical line at the actual i value.
    b. In the same figure, plot the histogram of the estimated j parameter of E, a vertical line at the average of the estimations, and a vertical line at the actual i value.
    c. In the same figure, plot the histogram of the estimated k parameter of H, a vertical line at the average of the estimations, and a vertical line at the actual i value.
    d. In the same figure, plot the histogram of the estimated i parameter of H, a vertical line at the average of the estimations, and a vertical line at the actual i value.
9. Copy your figures from step 8 into your report.

TASK 4: Confidence intervals

Your fourth task is to calculate confidence intervals for the population mean of variable A.

Here are the detailed steps:

1. Implement a function (**calc_conf_int_mean**) that takes a population, sample size (n), variable index, population standard deviation for that variable, and the confidence level as input. Using a look-up dictionary for quantiles $z_{0.1}$, $z_{0.05}$, $z_{0.025}$, $z_{0.01}$, and $z_{0.05}$, it computes and returns a confidence interval for the population mean of the requested variable.
2. For random variable A and sample size of 1000, compute the confidence intervals for 20 different samples for both confidence levels as 0.98 and 0.8, using the calc_conf_int_mean function.
    a. In the same figure, plot the computed confidence intervals for level 0.98, confidence intervals for level 0.8, and a vertical line at the actual population mean for A.

3. Copy your figure from step 2 into your report. How do the confidence intervals change with respect to the confidence levels? Comment on your findings.
4. For random variable A and a confidence level of 0.9, compute the confidence intervals for the sample size of [100, 400, …, 4900] using the calc_conf_int_mean function.
    a. In the same figure, plot the computed confidence intervals with varying sample sizes and a vertical line at the actual population mean for A.
5. Copy your figure from step 4 into your report. How do the confidence intervals change with respect to the sample sizes? Comment on your findings.

TASK 5: Hypothesis testing

Your fifth task is to do hypothesis testing for the case given below. The answer should only be included in your report.

After a year-long national publicity campaign on physical fitness, a hospital embarks on a mission to determine if the campaign has been effective. For this purpose, they survey 500 randomly selected patients and find out that their average exercise frequency is now 1.2. Assuming that the standard deviation of exercise frequency has not changed, does this mean that, at a 3% significance level, the mean number of times a patient engages in physical exercise each week has increased? Explain your steps clearly.

TASK 6: Naive Bayes classifier

Your last task is to apply and evaluate the Naive Bayes classifier for our population.

Here are the detailed steps:

1. Implement a function (**estimate_parameters**) that takes a population sample, and using the previously implemented functions, it estimates and returns the category-conditioned probability distribution parameters of variables D, E, and H and prior category probabilities (3x2 for D and E, and 3 for C and H, in total 18 parameters).
2. Derive the formula for the posterior probability of each patient category (C) given D, E, and H.
3. Implement a function (**calc_posterior**) that takes the cholesterol level (d), systolic blood pressure (e), and Hemoglobin A1c (HbA1c) level (h) of a patient and the distribution parameters estimated using the estimate_parameters function. Using the Bayes Rule, it computes and returns patient category (C) probabilities.
4. Create a sample of size 10000 for training. Training means that the patient categories are known (labeled) and will be used for estimating the category-conditioned probability distribution parameters.
5. Create a sample size of 1000 for testing. Testing means that the patient categories are to be predicted. They will be determined as the category with the maximum posterior probability.
6. Using the training sample, compute the category-conditioned probability distribution parameters.
7. For each patient in the test sample, compute the posterior probability for each category using the corresponding d, e, and h values and the previously estimated distribution parameters. Determine the category of the patient as the one with the highest posterior probability. Compare the estimated and the actual categories to check if the classification was correct. In 1000 tests, compute and print the ratio of the correctly classified patients.

IMPORTANT NOTES:

1. A **lecture** on Python, NumPy, and Pyplot focusing on this term project will be given next week. **Notes** from that lecture will also be available for you.
2. This is **NOT** a group activity. You can discuss some points with your friends, but each student will submit and be graded **individually**. If you want to be safe rather than sorry, you should consult with the **course assistants** as your primary and with the **instructor** as your secondary source of knowledge.
3. The Python scripts and the reports will be processed manually and automatically to detect plagiarism and cheating. Again: You should be careful about the level of cooperation with your friends. In case of cheating, all parties will be punished equally.
4. Your Python script should be named **studentid.py**, and your report **studentid.pdf**.
5. If your report is hand-written, the responsibility is yours to make it as legible as possible. Illegible parts will not be graded.
6. Once the projects are graded, you will receive detailed feedback about your submission. There will be a grace period for your objections.