Information Theory for Data Science

Pouria Mohammadalipourahari

Reihaneh Kharazmi

Arezoo Parsian

Armi Okshtuni

Prof. Garello

1 November 2023

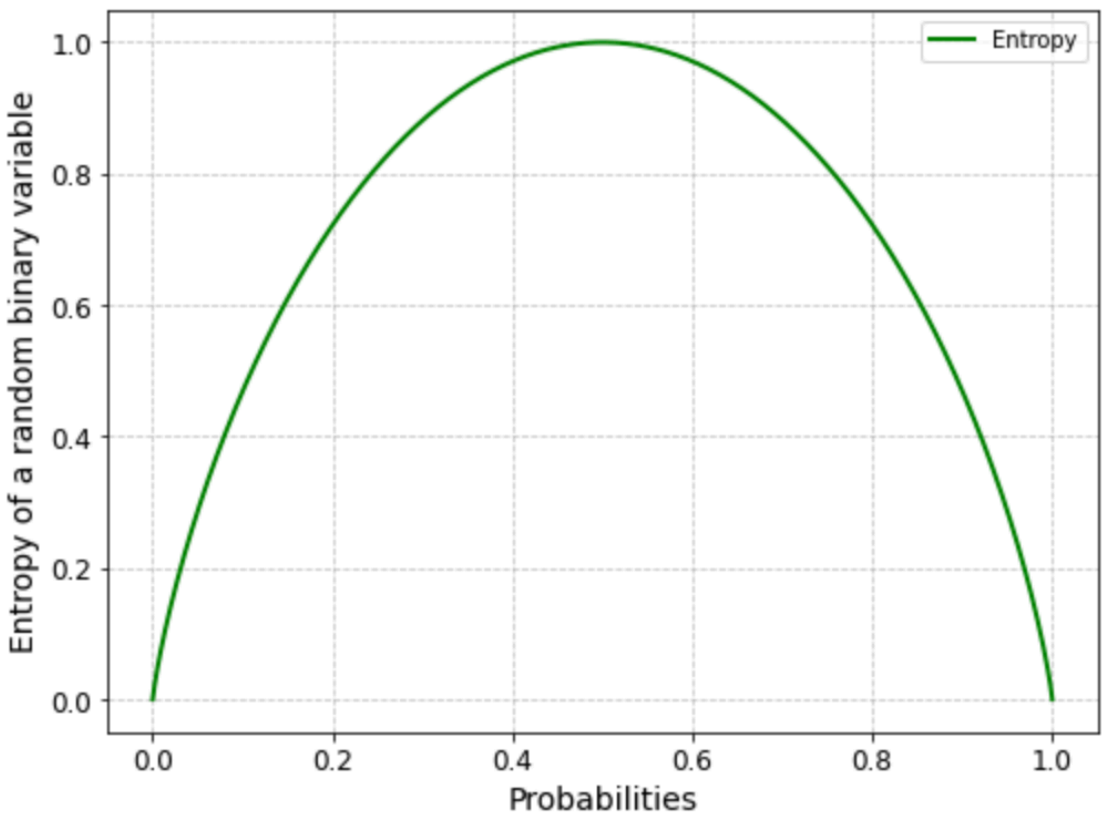
1. Entropy of a binary random variable

1.A

This equation represents the formula of binary random variable and we plotted it.

$$H(x) = -p\log_2(p) - (1-p)\log_2(1-p)$$

1.



2.

We established the entropy function, adhering to the principle that when the probability of an event assumes either of the extreme values, 0 or 1, the entropy, symbolized as 'H,' unequivocally converges to 0. Subsequently, we systematically computed the entropy values for probabilities situated between these extremes, employing the fundamental Shannon entropy formula.

Remarkably, our analysis illuminated a distinctive aspect of the entropy function: the maximum entropy, representing the pinnacle of uncertainty, is attained when the probability 'p' precisely equals 0.5. At this critical juncture, the entropy attains its highest value, registering a numerical value of 1. In stark contrast, we observed that the entropy of the system consistently descends to the minimum value of 0, indicative of complete certainty, when the probability 'p' assumes the values of 0 or 1.

1.B

$$H_\alpha(x) = \frac{1}{1-\alpha}\log_2\left(\sum_i p_i^\alpha\right)$$

3.

Shannon entropy is :

$$H(x) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

When $\alpha \rightarrow 1$:

$\lim_{\alpha\to 1}\frac{1}{1-\alpha}\log_2\left(\sum p^\alpha\right) = \infty$

In order to determine the limit at the point 1, we will employ L'Hôpital's rule, resulting in the following expression:

$$\lim_{\alpha\to 1}\frac{\log_2\left(\sum p^\alpha\right)}{1-\alpha} :$$

I. Derivative of the Numerator

$$\frac{d}{d\alpha}\left(\log_2\left(\sum p^\alpha\right)\right) = \frac{1}{\ln(2)}\cdot\frac{\frac{d}{d\alpha}\left(\sum p^\alpha\right)}{\sum p^\alpha}$$

II. Derivative of the Denominator

$$\frac{d}{d\alpha}(1-\alpha) = -1$$

$$\lim_{\alpha \to 1} \frac{\frac{1}{\ln(2)} \cdot \frac{\frac{d}{d\alpha}\left(\sum p^\alpha\right)}{\sum p^\alpha}}{-1} = -\frac{1}{\ln(2)} \cdot \frac{\frac{d}{d\alpha}\left(\sum p^\alpha\right)}{\sum p}$$

To find the derivative of this sum with respect to alpha, we can differentiate each term individually and then sum the derivatives:

$$\frac{d}{d\alpha}\left(\sum p^\alpha\right) = \sum \frac{d}{d\alpha}\left(p^\alpha\right)$$

$$\frac{d}{d\alpha}\left(p^\alpha\right) = p^\alpha \cdot \ln(p) \cdot \frac{d}{d\alpha}(\alpha)$$

$$\frac{d}{d\alpha}\left(p^\alpha\right) = p^\alpha \cdot \ln(p)$$

Result:

$$\frac{d}{d\alpha}\left(\sum p^\alpha\right) = \sum \left(p^\alpha \cdot \ln(p)\right)$$

Now we should replace the result here:

$$\lim_{\alpha \to 1} \frac{\frac{1}{\ln(2)} \cdot \frac{\frac{d}{d\alpha}\left(\sum p^\alpha\right)}{\sum p^\alpha}}{-1} = -\frac{1}{\ln(2)} \cdot \frac{\frac{d}{d\alpha}\left(\sum p^\alpha\right)}{\sum p^\alpha} = -\frac{1}{\ln(2)} \cdot \frac{\sum \left(p^\alpha \cdot \ln(p)\right)}{\sum p^\alpha}$$

We know that alpha is 1 so:

$$= -\frac{1}{\ln(2)} \cdot \frac{\sum_{i=1}^{n} \left(p_i \cdot \ln(p_i)\right)}{\sum_{i=1}^{n} p_i}$$

And we know that $\sum_{i=1}^{n} p_i$ = 1 and $\frac{\ln(p_i)}{\ln(2)} = \log_2(p_i)$:

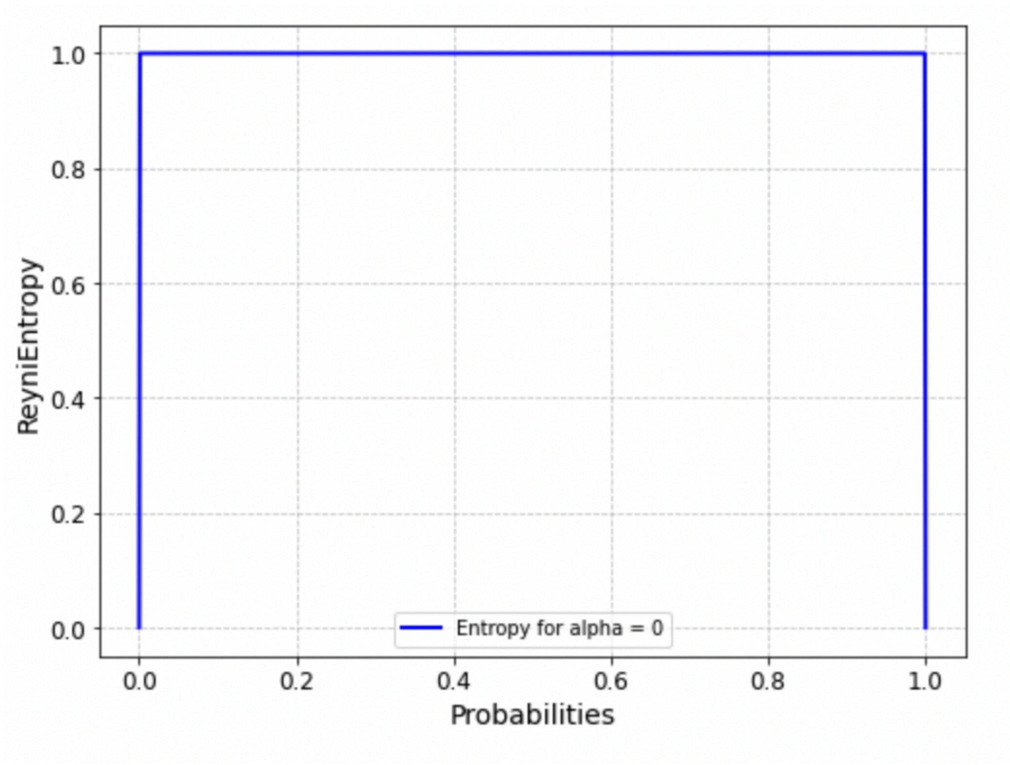
$$= -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

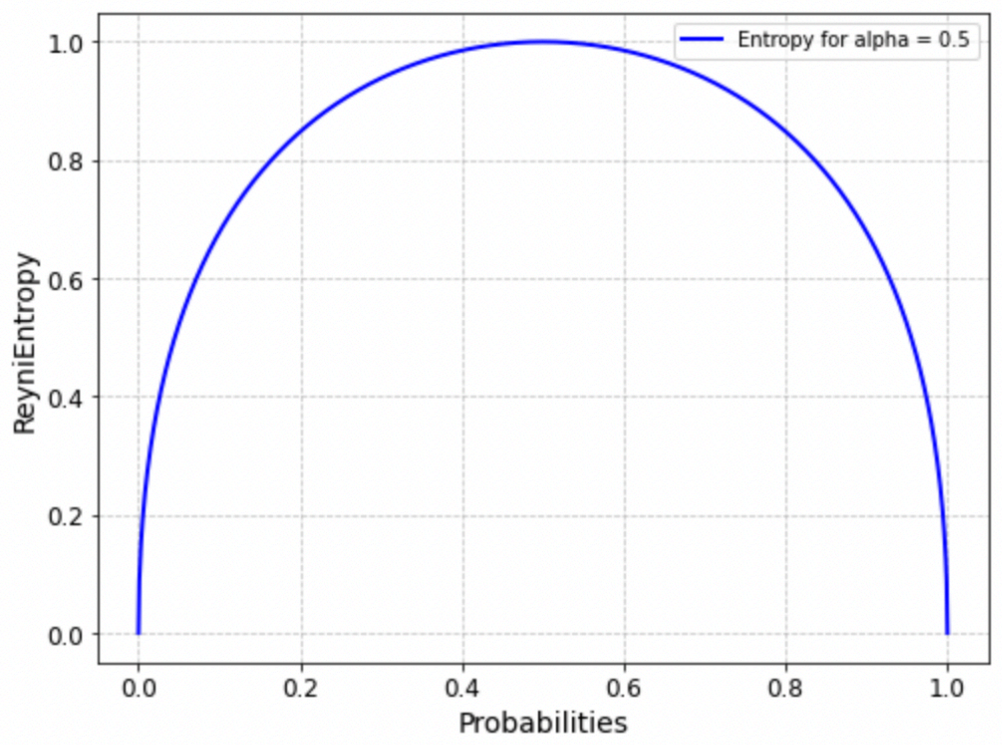
So the final result of $\lim_{\alpha \to 1} \frac{1}{1-\alpha} \log_2\left(\sum p^\alpha\right)$ is:

$$\lim_{\alpha \to 1} \frac{1}{1-\alpha} \log_2\left(\sum p^\alpha\right) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

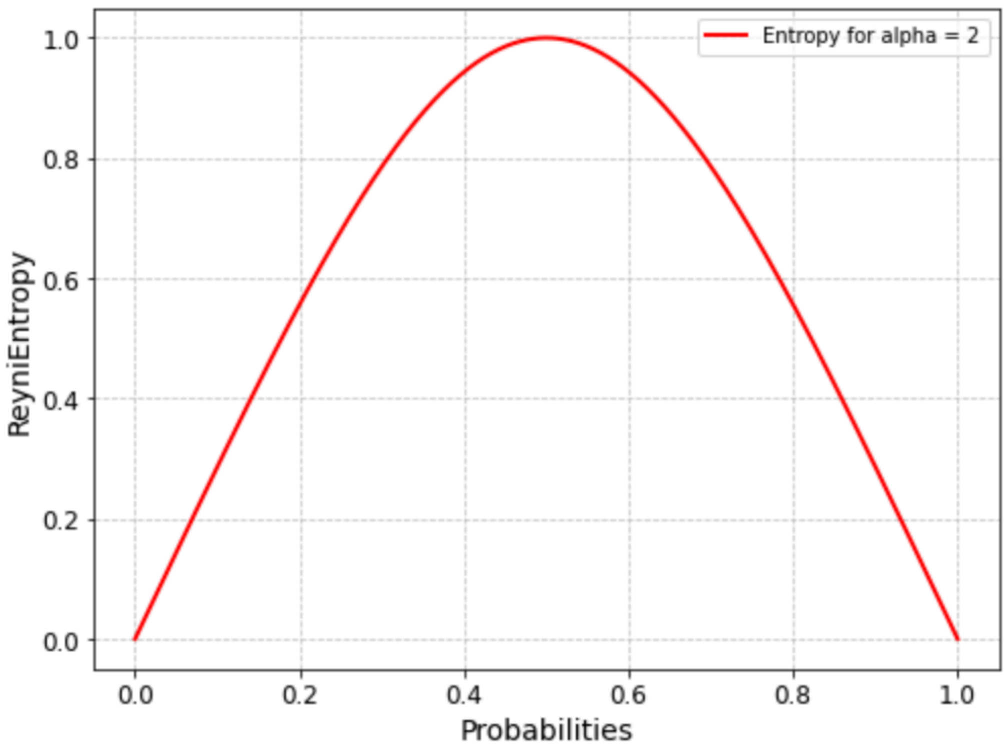
4.

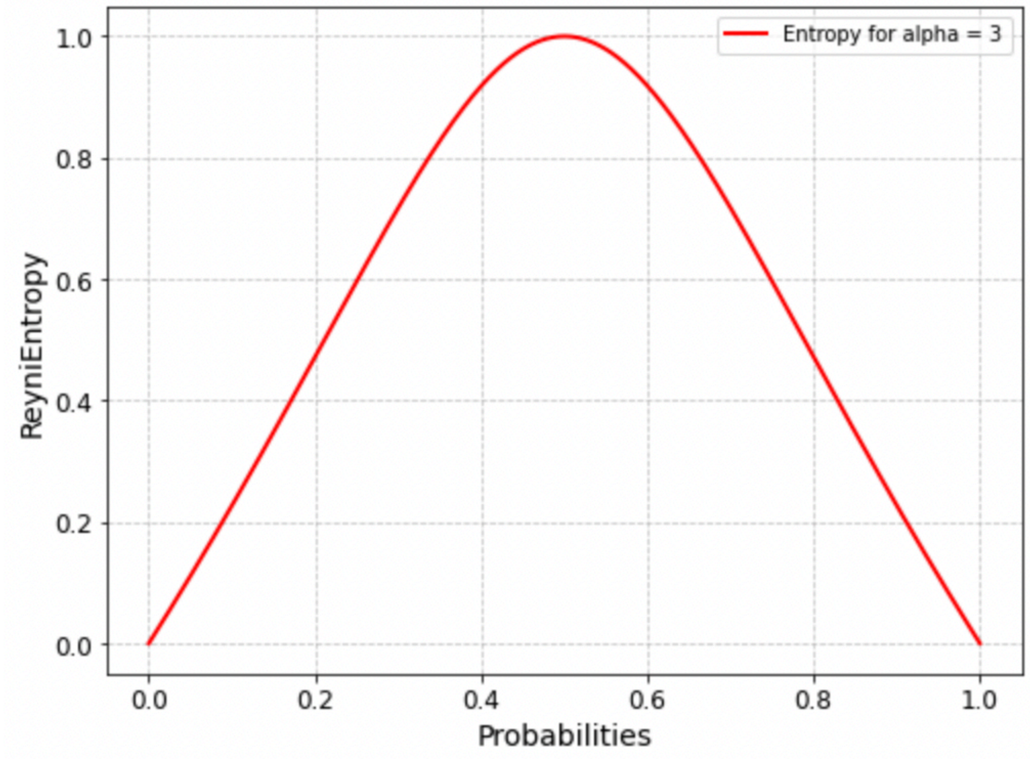
For the smaller than 1 alpha values:





For the bigger alpha values:





5.

It iIn the figures presented, it is evident that as the parameter α converges toward 1, the graph closely resembles the canonical Shannon entropy figure. Conversely, as α approaches values smaller than 1 yet greater than 0, the graph exhibits an expanding pattern until, ultimately, at $\alpha = 0$, it assumes only two distinct values, 0 and 1. In contrast, as α approaches values greater than 1, the graph demonstrates a converging trend, becoming increasingly narrower.

1.C

We know that:

$$H(p_1,p_2,p_3) = -\left[(p_1)\log_2(p_1) + (p_2)\log_2(p_2) + (p_3)\log_2(p_3)\right]$$

$$H(p,p,p_3) = -\left[\frac{p_1+p_2}{2}\log_2\left(\frac{p_1+p_2}{2}\right) + \frac{p_1+p_2}{2}\log_2\left(\frac{p_1+p_2}{2}\right) + p_3\log_2(p_3)\right]$$

We need to prove H(p_1,p_2,p_3) <= H(p,p,p3) :

$$H(p_1,p_2,p_3) - H(p,p,p_3) \leq 0$$

$$(p_1)\left[\log_2\left(\frac{p_1+p_2}{2}\right) - \log_2(p_1)\right] + (p_2)\left[\log_2\left(\frac{p_1+p_2}{2}\right) - \log_2(p_2)\right] \leq 0$$

We know that log2(a) - log2(b) = log2(a/b) :

$$(p_1)\left(\log_2\left(\frac{p_1+p_2}{2p_1}\right)\right) + (p_2)\left(\log_2\left(\frac{p_1+p_2}{2p_2}\right)\right) \leq 0$$

Using log inequality:

Eq1: $$(p_1)\left(\log_e\left(\frac{p_1+p_2}{2p_1}\right)\right) \leq \left(\frac{p_2-p_1}{2p_1}\right)$$

Eq2: $$(p_2)\left(\log_e\left(\frac{p_1+p_2}{2p_2}\right)\right) \leq \left(\frac{p_1-p_2}{2p_2}\right)$$

We know that $\dfrac{\log_a(b)}{\log_a(c)} = \log_c(b)$ and $\log_e(2) \geq 0$:

Eq1/$\log_e(2)$ and Eq2/$\log_e(2)$:

$$(p_1)\left(\log_2\left(\frac{p_1+p_2}{2p_1}\right)\right) \leq \left(\frac{p_2-p_1}{2p_1}\right)\log_2 e$$

$$(p_2)\left(\log_2\left(\frac{p_1+p_2}{2p_2}\right)\right) \leq \left(\frac{p_1-p_2}{2p_2}\right)\log_2 e$$

Now, we can proceed by aggregating these components to establish the conclusion we aimed to demonstrate:

$$p_1\left(\log_2\left(\frac{p_1+p_2}{2p_1}\right)\right) + p_2\left(\log_2\left(\frac{p_1+p_2}{2p_2}\right)\right) \leq \left(\frac{p_2-p_1}{2p_1}\right)\log_2 e + \left(\frac{p_1-p_2}{2p_2}\right)\log_2 e$$

The right part is equal to zero so it is proven that:

$$H(p_1,p_2,p_3) - H(p,p,p_3) \leq 0$$

2. Maximum Entropy

1. We have a lucky wheel with 5 modes which are described below. And if someone wins the mode the cost of each mode for the producer is c1 to c5 respectively.

Cost = [5,3,7,1,4]

A. If the mean value was 4.5 what would be the amounts of probabilities for each mode?

B. Solve this question with a mean value which is equal to the arithmetic average.

C. Solve it for the mean value equal to 3.2.

Solution:

We defined and use a function to calculate betha and then we calculated the probabilities.

We use two equations to calculate betha:

b = betha

Eq1 : $\sum_{i=1}^{n} cost[i] \cdot (b^{cost[i]})$

Eq2: $\sum_{i=1}^{n} (b^{cost[i]})$

$$p_i = \frac{b^{cost[i]}}{b^{cost[0]} + b^{cost[1]} + \ldots + b^{cost[n]}}$$

2. Where arithmetic average is 4 and mean value is 4.5:

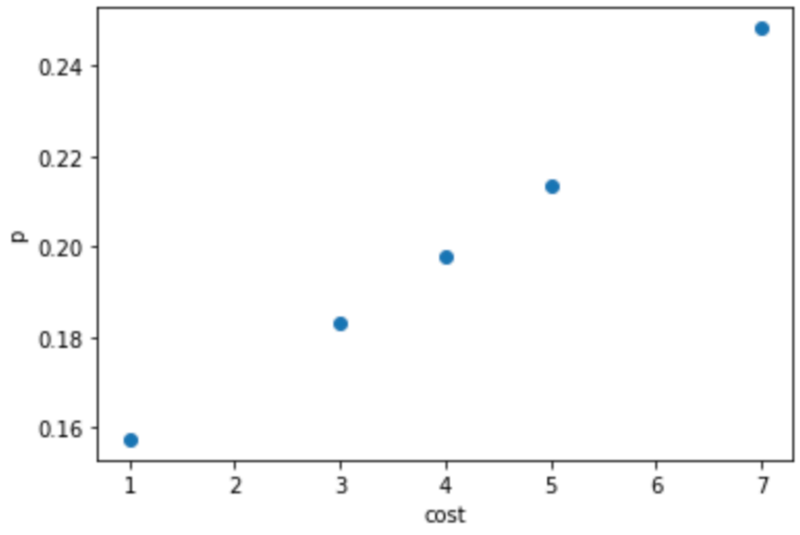
Probabilities : {0.213295605990214,

0.183274868129919,

0.248233792221214,

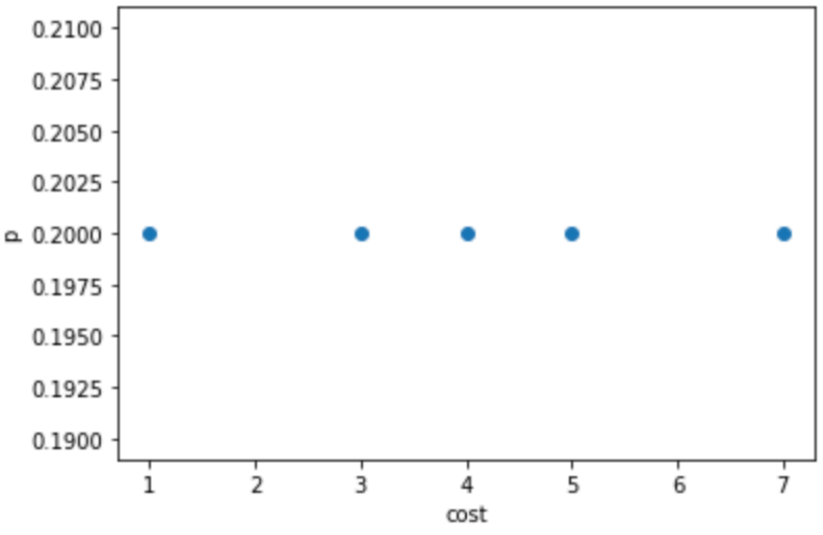
0.157479462045647,

0.197716271613005}



3. Where arithmetic average is 4 and mean value is 4 :

Probabilities : {0.2, 0.2, 0.2, 0.2, 0.2}



4. Where arithmetic average is 4 and mean value is 3.2 :

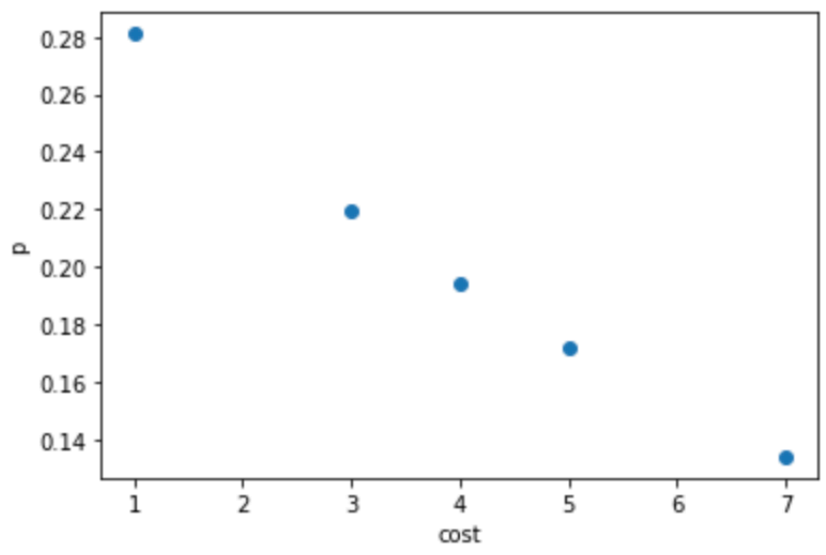
Probabilities : {0.171459341438099,

0.219531930565170,

0.133913575536382,

0.281082781103938,

0.194012371356412}

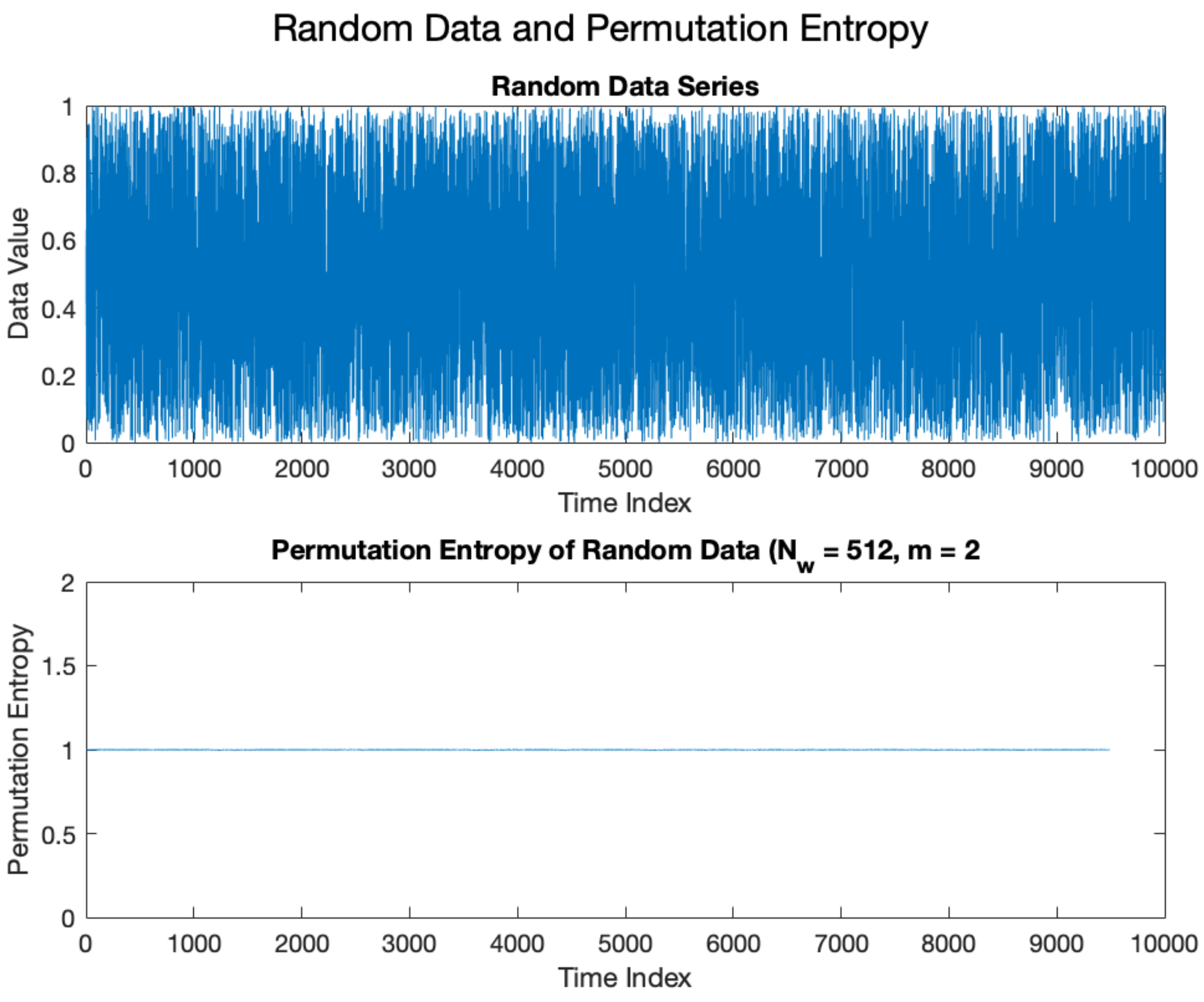


5.

It is noticeable that in the context of maximum entropy, the relationship between the mean value and the arithmetic average is critical. When the mean value matches the arithmetic average, all possible outcomes have the same probability. However, when the mean value exceeds the arithmetic average, the probabilities tend to rise with higher costs. Conversely, when the mean value is below the arithmetic average, the probabilities decrease as the costs increase.

3. Permutation Entropy for Time Series Anomaly Detection

Part A



Instead of m = 3 we assumed = 2.

Correlated Data Generation:

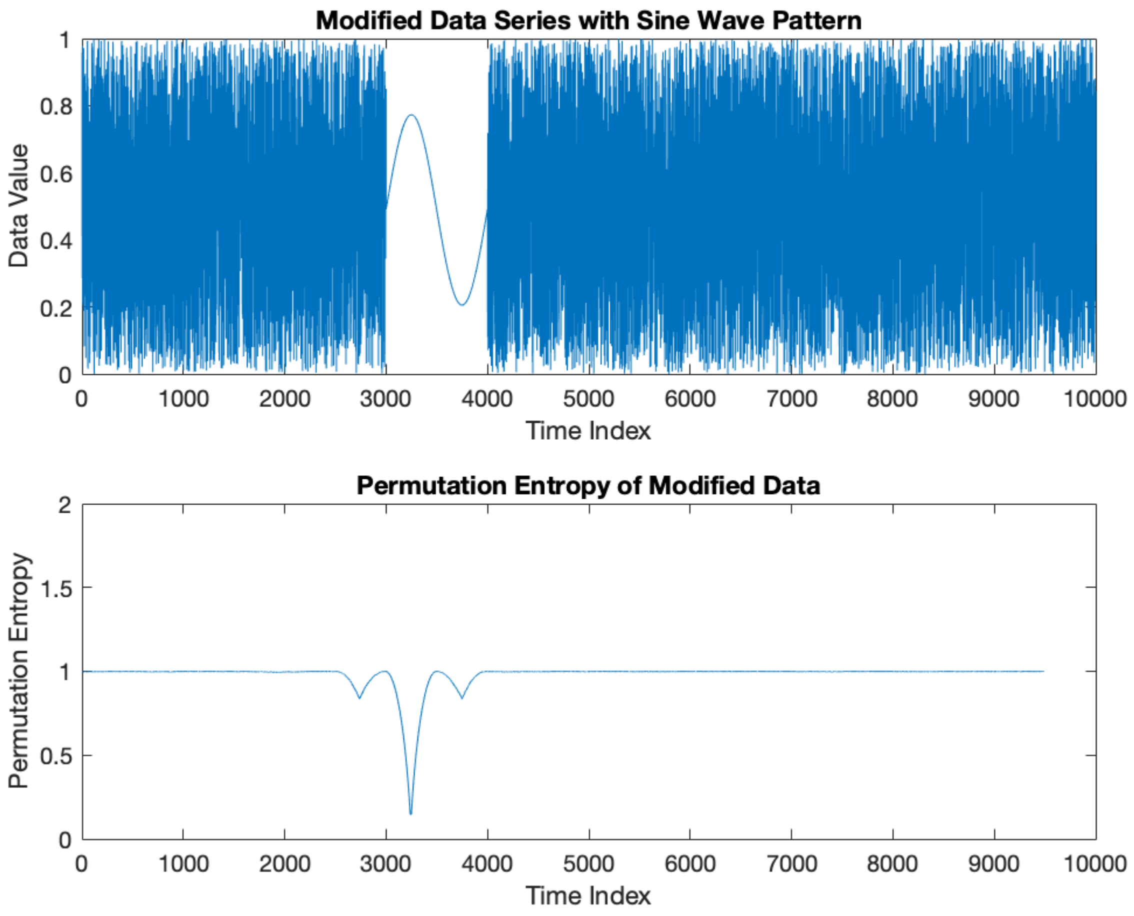
Within the specified segment, a sine wave pattern is generated with a frequency of 1/1000, which means that the pattern completes one full cycle over a range of 1000 data points. The pattern's mean and variance are adjusted to match the characteristics of the original segment, so we can have a similar central value and a similar spread of values as the original data.

Computation:

We used formulas in the slide and computed the permutation entropy with the same N_W and N_R:

$$0 \leq H' = \frac{\sum p_i \log_2(1/p_i)}{\log_2(m!)} \leq 1$$

Part B

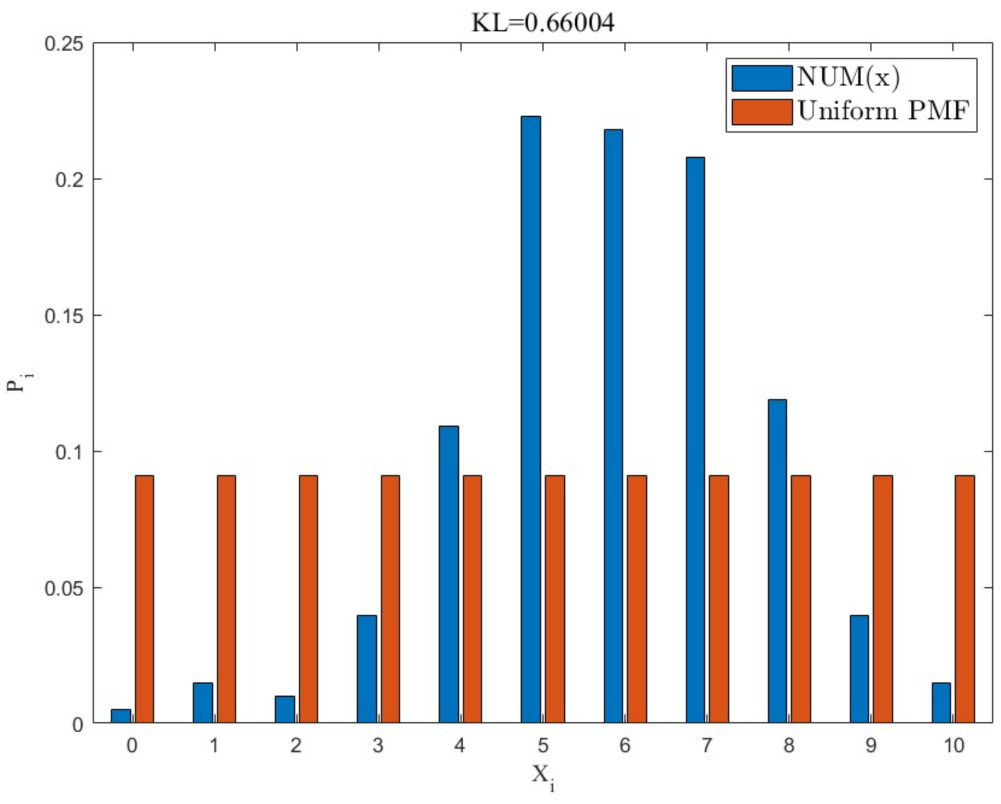


Periodic spikes occur because the sine wave pattern is repeating within the sliding window. At certain points in the pattern's cycle, the permutation order does not change significantly, resulting in higher permutation entropy values (i.e., lower complexity). These spikes represent the regular, predictable portions of the data.
The valleys represent the less predictable regions of the data. This corresponds to the moments when the pattern is transitioning from one phase to another. These transitions introduce randomness and complexity into the data, leading to lower entropy values.

The reason the permutation entropy starts to drop slightly before the 3000th data point is that the pattern introduced is continuous and extends over a range of data points, affecting the entropy values in the surrounding region. Additionally, the choice of **N_w** may cause a shift in the perceived timing of the drop in entropy.

4. Kullback-Leibler distance from empirical distribution

1.



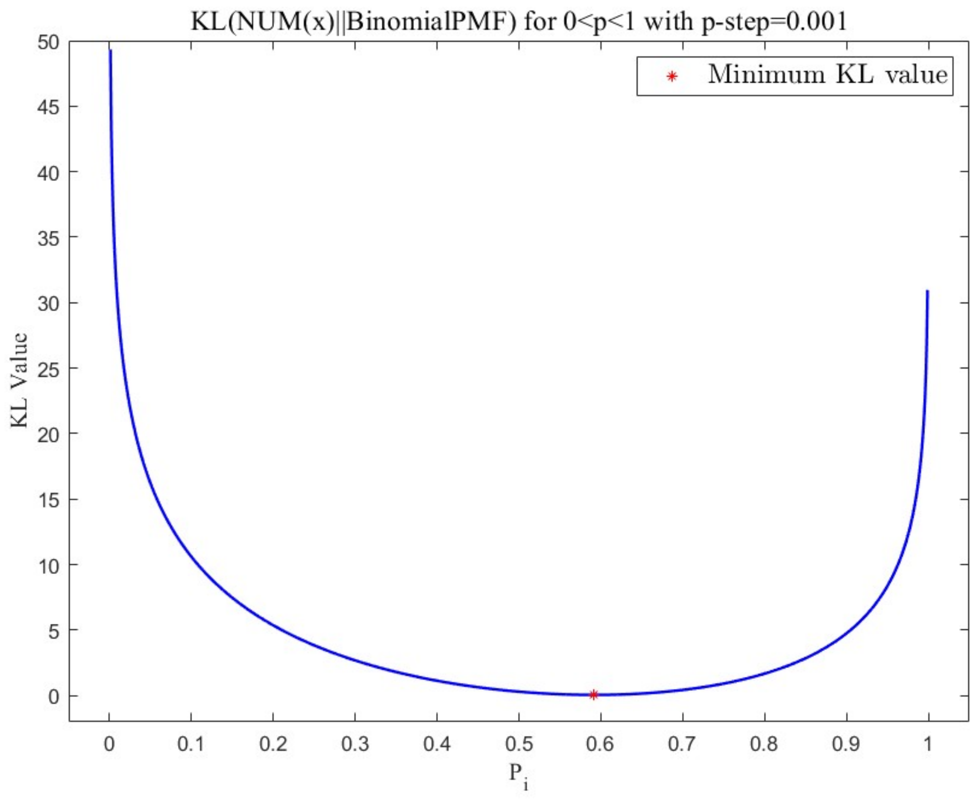
2.

KL divergence is 0.66004 and it is written in the title

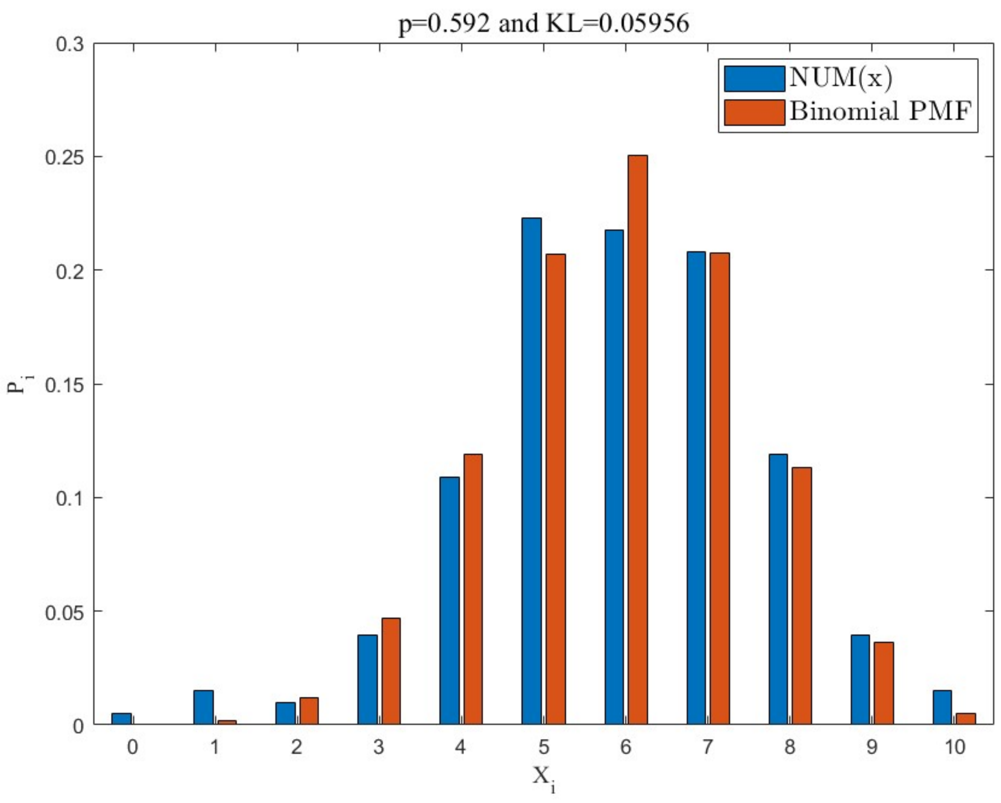
In the first section, our objective is to evaluate the dissimilarity between the observed data and a uniform Probability Mass Function (PMF) using Kullback-Leibler (KL) divergence. KL divergence is a measure that quantifies how one probability distribution, denoted as P, differs from a second reference distribution, denoted as Q. The resulting KL divergence value falls within the range [0, ∞). Specifically, when KL(P, Q) equals 0, it signifies that the two distributions, P and Q, are identical.

In the context of the first section, we begin by normalizing NUM(x) to have a total probability of 1, creating Px. Subsequently, we compute the KL divergence between Px and a uniform PDF (Uni_PDF). This analysis serves as an essential step in assessing the level of divergence between our observed data and a uniform distribution.

3.



4,5.



In the second section, our objective is to establish various Binomial Probability Mass Functions (PMFs) and subsequently identify the PMF that exhibits the minimum Kullback-Leibler (KL) divergence, along with the associated probability values. The core concept here involves the exploration of numerous binomial PMFs, each with distinct parameters, in an attempt to ascertain the PMF that best converges with the observed data.

It is important to highlight that the binomial PMF showing the highest level of convergence with the observed data will possess the lowest KL divergence when compared to other distributions. This process enables us to determine the most suitable binomial PMF that aligns closely with the characteristics of the observed data, aiding in statistical modeling and analysis.

5. Information Gain and Tree Classifiers

In the first step we defined many functions for using them during calculations.

1.**Calculate_entropy**, which get a list and calculate entropy for list

2.**Conditional_entropy**, which gets two lists like X,Y and find H(X|Y)

3.**Calculate_informaiton_gain**:which gets the name of feature(among x1,x2,x3) and whole of data and then by using calculate_entropy and calculate_conditional_entropy find the information gain of the specific feature

4.**Calculate_information_gain_ratio**:which gets the name of feature(among x1,x2,x3) and whole of data and then by using calculate_entropy and calculate_information_gain find the information gain ratio of the specific feature

5.**Calculate_information_gain_ratio_for_features**:this function works like calculate_information_gain_ratio but it calculates IGR for all of features, which is used for picking a feature with highest IGR for splitting data in each node

First Step:

Finding first feature for splitting:In this phase, we calculate IGR for all of the features

X1:0.0487,X2:0.0.487,X3:0.3470

So we choose **X3**, As X3 is the numerical feature we should set a threshold for this

And It's **55.0**

And Now we just need to separate data by **x3**’s value of each item and the output of this step is

'True': [[30, 0, 70, 0], [30, 1, 80, 1], [60, 0, 60, 1], [60, 1, 60, 1]],

'False': [[30, 0, 10, 0], [30, 1, 20, 0], [60, 0, 40, 0], [60, 1, 50, 0]]

As it’s obvious, All False Outputs (which didn’t fulfil the condition) have the labels equal to **0**, so it’s finished and also doesn’t go in depth.However for True items(which fulfil the condition)

We should calculate IGR and comparing values with each other

Second Step:

Finding second feature for splitting:In this phase, we calculate IGR for all of the features

X1:0.3112,X2:0.3112,X3:0.5408

So again we choose **X3**, As X3 is the numerical feature we should set a threshold for this

And It’s **65.0**

And Now we just need to separate data(Not all of samples, only the True samples of the first step output) by **x3**’s value of each item and the output of this step is

'True': [[30, 0, 70, 0], [30, 1, 80, 1]],

'False': [[60, 0, 60, 1], [60, 1, 60, 1]]

As it’s obvious, All False Outputs (which didn’t fulfil the condition) have the labels equal to **1,**so it’s finished and also doesn’t go in depth.However for True items(which fulfil the condition)

We should calculate IGR and comparing values with each other

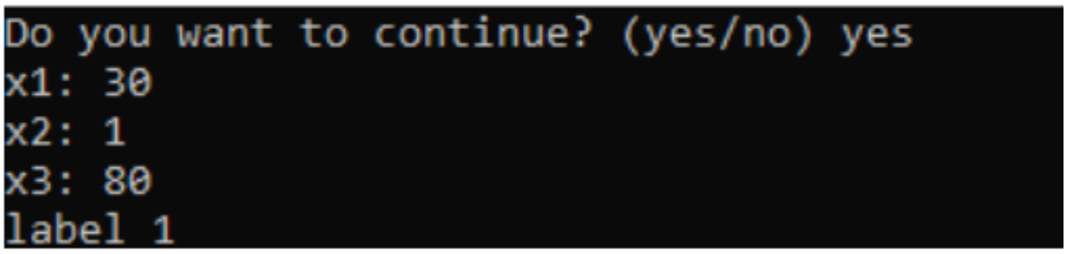
Third Step:

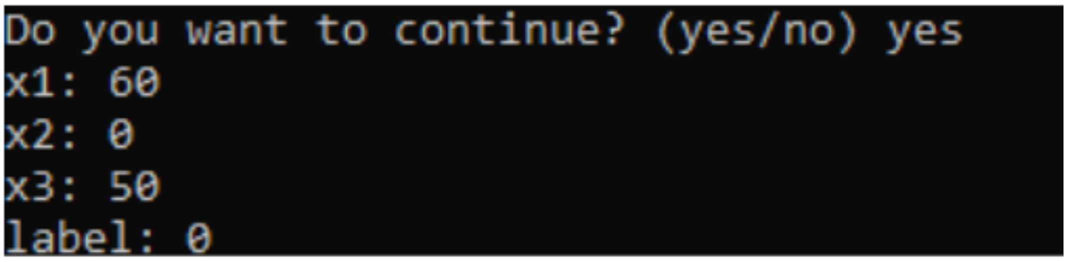
Finding second feature for splitting:In this phase, we calculate IGR for 2 features which are x2 and x3, as for x1 because x1 values of both items are the same and in this case IGR
Equals to 0

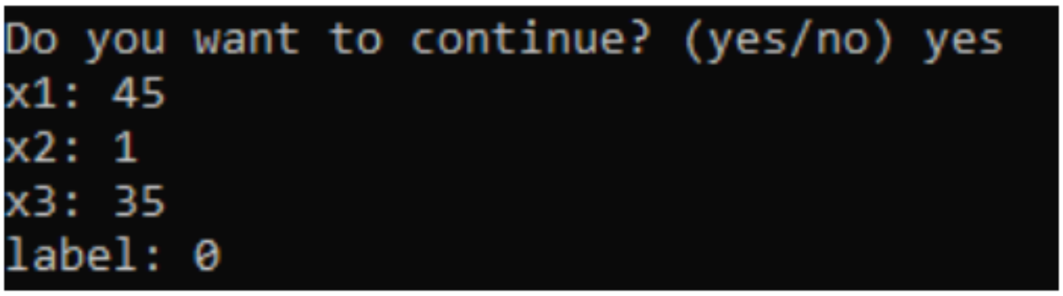
X1:0,X2:1,X3:1

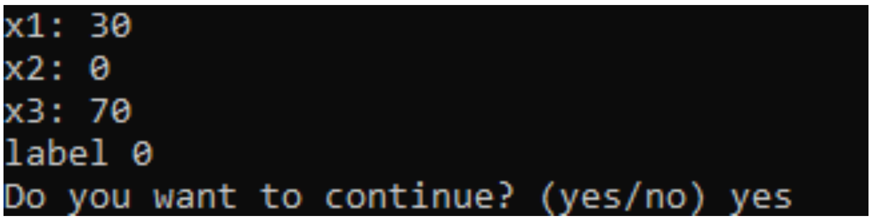
Here, because both IGRs are 1, we choose **X2** because of testing binary feature in practice.and because all the values of this feature are either 0 or 1, so it's obvious that threshold for this column(feature) is **0.5**, And Now we just need to separate data(Not all of samples, only the True samples of the first step output) by **x2**'s value of each item and the output of this step is
{'True': [[30, 1, 80, 1]],
'False': [[30, 0, 70, 0]]}
And here is the final step because both True and False nodes have just one item.
For testing this program, we stored all of the values of True and False and Also the conditions which are the name of the features and their threshold of each step in a dictionary, and once the user gives an input the program will compare the input values with the thresholds and give the label of it.
If you run the program, first of all you can see the diagram of the Training Set which is separated by thresholds, you can give items for testing the program. In the pictures below you can see many test items that program returned their label as an output.

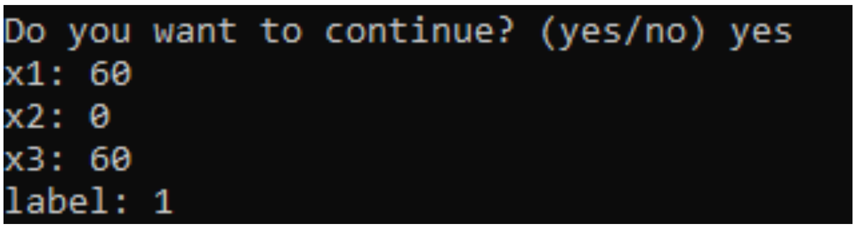
Examples:

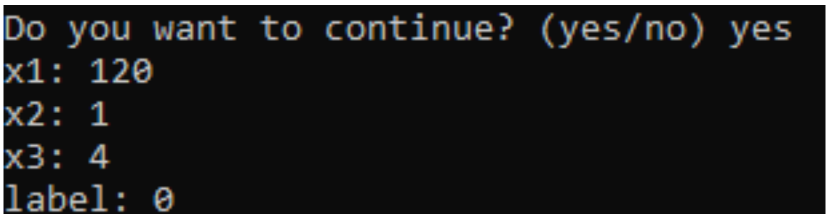

```
Do you want to continue? (yes/no) yes
x1: 30
x2: 1
x3: 80
label 1
```

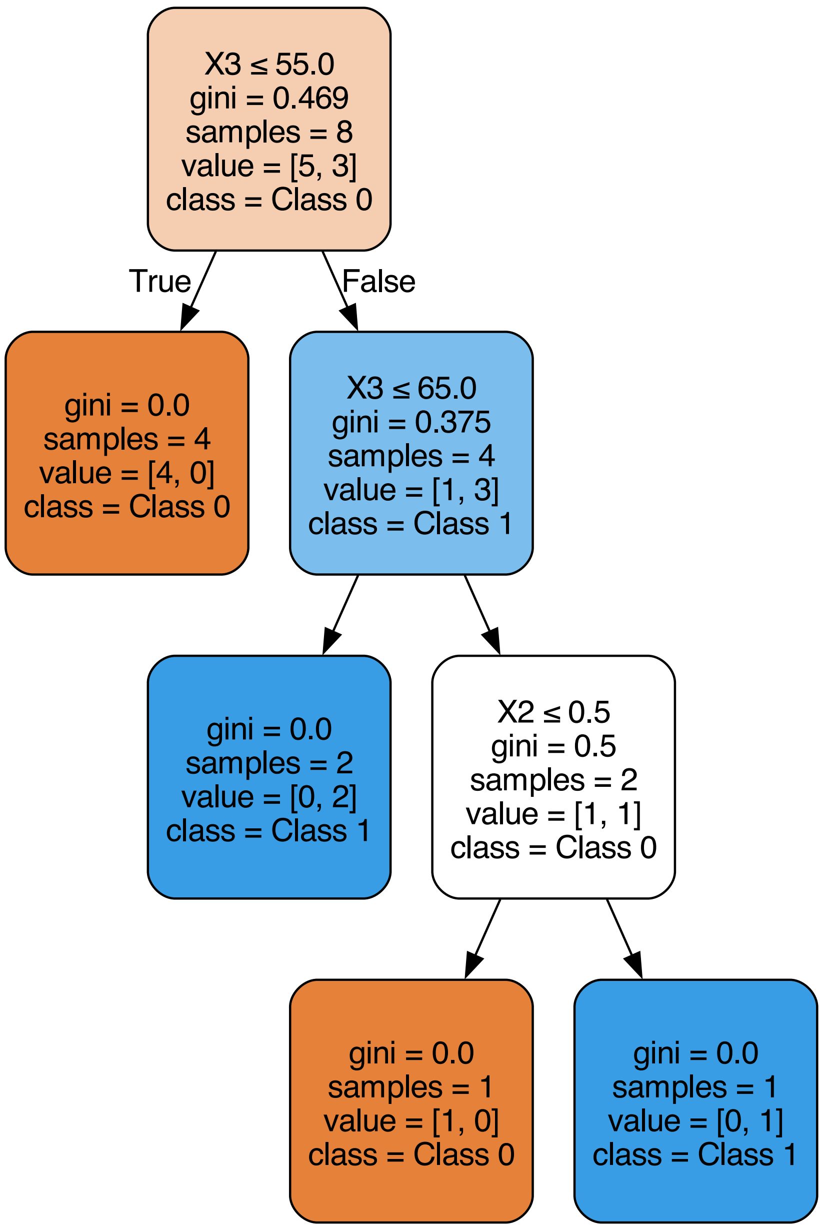

```
Do you want to continue? (yes/no) yes
x1: 60
x2: 0
x3: 50
label: 0
```


```
Do you want to continue? (yes/no) yes
x1: 45
x2: 1
x3: 35
label: 0
```


```
x1: 30
x2: 0
x3: 70
label 0
Do you want to continue? (yes/no) yes
```


```
Do you want to continue? (yes/no) yes
x1: 60
x2: 0
x3: 60
label: 1
```


```
Do you want to continue? (yes/no) yes
x1: 120
x2: 1
x3: 4
label: 0
```


X3 ≤ 55.0
gini = 0.469
samples = 8
value = [5, 3]
class = Class 0
True
False
gini = 0.0
samples = 4
value = [4, 0]
class = Class 0
X3 ≤ 65.0
gini = 0.375
samples = 4
value = [1, 3]
class = Class 1
gini = 0.0
samples = 2
value = [0, 2]
class = Class 1
X2 ≤ 0.5
gini = 0.5
samples = 2
value = [1, 1]
class = Class 0
gini = 0.0
samples = 1
value = [1, 0]
class = Class 0
gini = 0.0
samples = 1
value = [0, 1]
class = Class 1