MACHINE LEARNING, EDITORIAL, PROGRAMMING, STATISTICS

# Survival Analysis with Python Tutorial — How, What, When, and Why

Diving into survival analysis with Python — a statistical branch used to predict and calculate the expected duration of time for one or more significant events to occur.

**Author(s): Pratik Shukla**

Last updated, January 8, 2021



This article covers an extensive review with step-by-step explanations and code for how to perform statistical survival analysis used to investigate the time some event takes to occur, such as patient survival during the COVID-19 pandemic, the time to failure of engineering products, or even the time to closing a sale after an initial customer contact.

**This tutorial's code is available on Github and its full implementation on Google Colab.**

**Table of Contents:**

## 1. Survival Analysis Basics:

**Survival analysis** is a set of statistical approaches used to determine the time it takes for an event of interest to occur. We use survival analysis to study the **time** until some **event** of interest occurs. Time is usually measured in years, months, weeks, days, and other time measuring units. The event of interest could be anything of interest. It could be an actual death, a birth, a retirement, along with others.

**For instance, how can Survival Analysis be useful to analyze the ongoing COVID-19 pandemic data?**

1. We can find the number of days until patients showed COVID-19 symptoms.
2. We can find for which age group it is deadlier.
3. We can find which treatment has the highest survival probability.
4. We can find whether a person's sex has a significant effect on their survival time?
5. We can find the median number of days of survival for patients.
6. We can find which factor has more impact on patients' survival.

In this tutorial, we are going to perform a thorough analysis of **patients with lung cancer.** Do not worry if it seems complicated. Once we go through the logic behind it, we will have the ability to perform survival analysis on any data set. **Exciting! Isn't it?**

**Survival analysis** is used in a variety of field such as:

- Cancer studies for patients survival time analyses.
- Sociology for "event-history analysis."
- In Engineering for "failure-time analysis."
- Time until product failure.
- Time until a warranty claim.

- Time until a process reaches a critical level.
- Time from initial sales contact to a sale.
- Time from employee hire to either termination or quit.
- Time from a salesperson hires to their first sale.

In **cancer studies**, typical research questions are:

1. What is the impact of specific clinical characteristics on patient's survival? For example, is there any difference between people who have higher blood sugar and those who do not?
2. What is the probability that an individual survives a specific time (years, months, days)? For example, given a set of cancer patients, we will tell that if 300 days after a cancer diagnosis has been passed, then the probability of that person being alive at that time will be 0.7.
3. Are there differences in survival between groups of patients? For example, Let's say there are two groups of people diagnosed with cancer. Those two groups were given two different kinds of treatments. Our goal here will be to find out if there is a significant difference between the survival time for those two different groups based on the treatment they were given.

#### Objectives

In this tutorial, we will see the following methods of survival analysis in detail:

*1)* ***Kaplan-Meier plots*** to visualize survival curves.

***2) Nelson-Aalen plots*** to visualize the cumulative hazard.

***3) Log-Rank test*** to compare the survival curves of two or more groups

***4) Cox-proportional hazards regression*** finds out the effect of different variables like age, sex, and weight on survival.

#### Fundamental concepts

We will start this tutorial by understanding some basic definitions and concepts related to survival analysis.

#### Survival time and type of events in cancer studies

**Survival Time:** It is usually referred to as an amount of time until when a subject is alive or actively participates in a survey.

There are three main types of events in survival analysis:

**1) Relapse:** Relapse is defined as a deterioration in the subject's state of health after a temporary improvement.

**2) Progression:** Progression is defined as the process of developing or moving gradually towards a more advanced state. It basically means that the health of the subject under observation is improving.

**3) Death:** Death is defined as the destruction or permanent end of something. In our case, death will be our event of interest.

#### Censoring

As we discussed above, survival analysis focuses on the occurrence of an event of interest. The event of interest can be anything like birth, death, or retirement. However, there is still a possibility that the event we are interested in does not occur. Such observations are known as censored observations.

There are three types of censoring:

1. **Right Censoring**: The subject under observation is still alive. In this case, we can not have our timing when our event of interest(death) occurs.
2. **Left Censoring:** In this type of censoring, the event cannot be observed for some reason. It may also include the event that occurred before the experiment started, such as the number of days from birth when the kid started walking.
3. **Interval Censoring:** In this type of data censoring, we only have data for a specific interval, so it is possible that the event of interest does not occur during that time.

Censoring may occur in the following instances:

1. A patient has not (yet) experienced the event of interest (death or relapse in our case) within a period.
2. A patient is not followed anymore.
3. If a patient moves to another city, then follow-up might not be possible for the hospital staff.
4. We only have the data for a specific interval.

#### Survival and hazard functions

We generally use two related probabilities to analyze survival data for a subject.

1) Survival Function(S)

2) Hazard Function (H)

To find the survival probability of a subject, we will use the survival function S(t), the Kaplan-Meier Estimator. The survival function is defined as the probability that an individual (subject) survives from the time origin (diagnosis of a disease) to a specified future time t. Please note that the time can be in various forms like minutes, days, weeks, months, or years. For example, S(200)=0.7 means that after 200 days, a subject's survival probability is 0.7. In many deadly diseases, the survival probability decreases as the period increases. If the subject is alive at the end of an experiment, then that data will be censored.

The hazard probability, denoted by H(t), is the probability that an individual (subject) who is under observation at a time t has an event (death) at that time. For example, If h(200) = 0.7 means that after 200 days or on the 200th day, the probability of being dead is 0.7. One thing to keep in mind here is that the hazard function gives us the cumulative probability. We will discuss this in detail later in this tutorial.

Notice that, in contrast to the survival function, which focuses on the survival of a subject, the hazard function gives us the probability of a subject being dead on a given time. We can note that higher survival probability and lower hazard probability is good for the subject's health.

**Let's move forward to the cool coding part!**

Download the public dataset from the UPC.

**Data Description:**

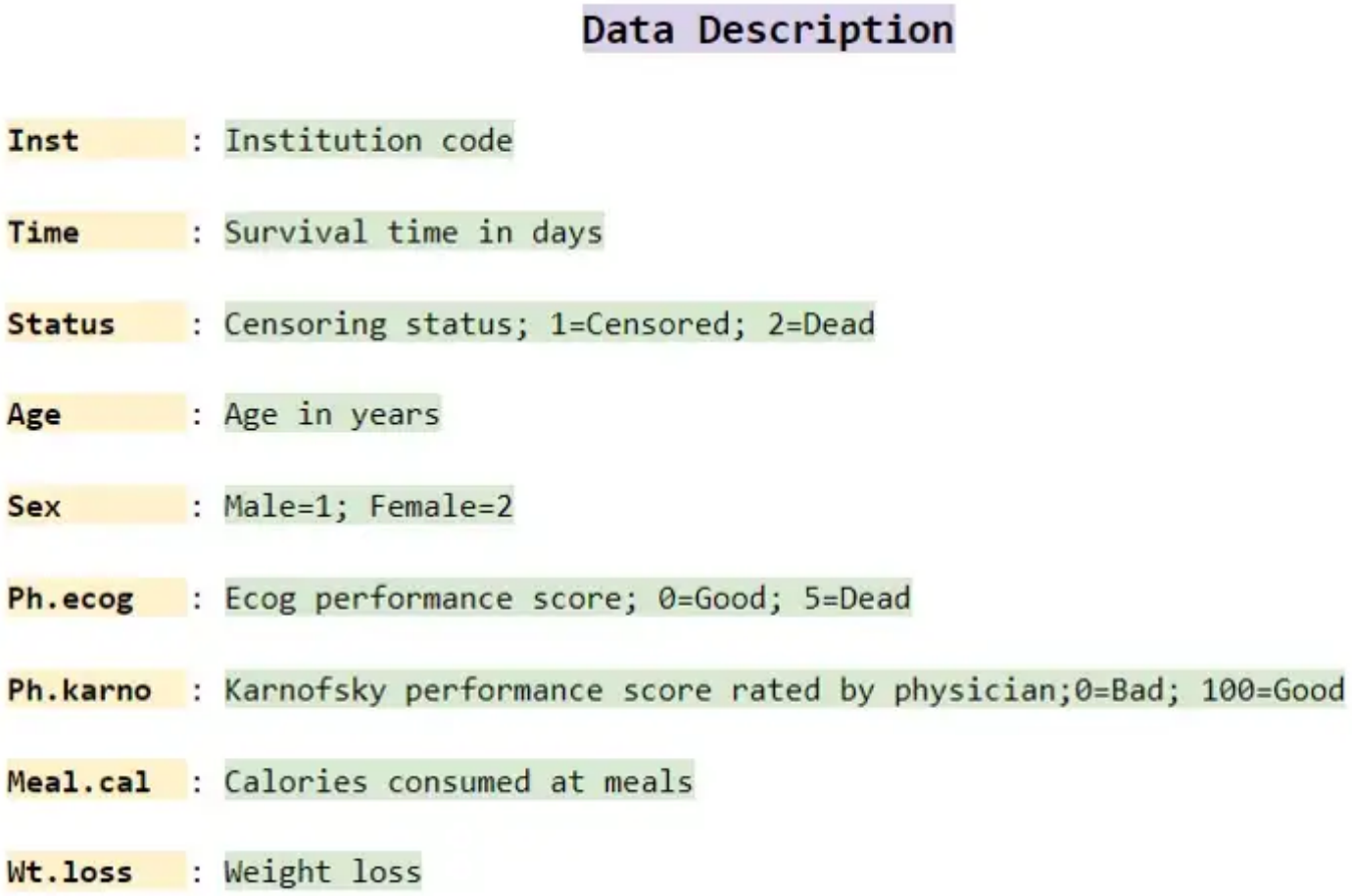


Figure 1: Data description values.

## 2. Kaplan-Meier Estimator Theory and Example

The **Kaplan–Meier estimator** is a non-parametric statistic used to estimate the survival function (probability of a person surviving) from the lifetime data. In medical research, it is often used to measure the fraction of patients living for a specific time after treatment or diagnosis. For example: Calculating the amount of time(year, month, day) a particular patient lived after he/she was diagnosed with cancer or his treatment starts. The estimator is named after **Edward L. Kaplan** and **Paul Meier**, who submitted similar manuscripts to the *American Statistical Association* Journal.

The probability of survival at time ti, which is denoted by S(ti), is calculated as follow:

$$\hat{S}(t) = \prod_{t_i \leq t} \frac{n_i - d_i}{n_i}$$

Where,

$\hat{S}(t)$ = The probability of being alive at time $t$

$n_i$ = The number of subjects alive just before time $t_i$

$d_i$ = The number of events at time $t_i$

Figure 2: Formula to calculate the probability of survival at time ti.

We can also write the equation above in a simple form as follows:

$$S(t_i) = S(t_{i-1}) * \left(1 - \frac{d_i}{n_i}\right)$$

Where,

$S(t_i)$ = The probability of being alive at time $t_i$

$n_i$ = The number of subjects alive just before time $t_i$

$d_i$ = The number of events at time $t_i$

$S(0) = 1$

$t_0 = 0$

Figure 3: Probability of survival time in a simple form S(ti).

For Example:

**1) Survival probability at time t=1:**

$$S(1) = S(0) * \left(1 - \frac{d_1}{n_1}\right) = \left(1 - \frac{d_1}{n_1}\right)$$

Figure 4: Survival probability at time t=1 formula

**2) Survival probability at time t=2:**

$$S(2) = S(1) * \left(1 - \frac{d_2}{n_2}\right)$$

Figure 5: Survival probability at time t=2 formula

### 3) Survival probability at time t=3:

$$S(3) = S(2) * \left(1 - \frac{d_3}{n_3}\right)$$

Figure 6: Survival probability at time t=3 formula

In a more generalized way, the probability of survival for a particular time is given by.

$$S_t = \frac{Number\ of\ subjects\ at\ \ risk\ at\ the\ start - Number\ of\ subjects\ that\ died}{Number\ of\ subjects\ at\ risk\ at\ the\ start}$$

Figure 7: Generalized formula for the probability of survival for a particular time.

From the above equations, we can confidently say that.

$$S(n) = S_0 * S_1 * S_2 * \ldots * S_n$$

Figure 8: Expressing survival generalization.

**Kaplan-Meier Estimator (Without any groups)**

### 1) Import required libraries:

```
#Import required libraries:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifelines import KaplanMeierFitter
```

Figure 9: Importing pandas, numpty, matplotlib.pyplot, and lifelines.

### 2) Read the data set:

```
#Read the dataset:

data = pd.read_csv("lung.csv")
data.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN |
| 1 | 2 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 |
| 2 | 3 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 |
| 3 | 4 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 |
| 4 | 5 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 |

Figure 10: Reading the dataset.

### 3) Print the columns in our data set:

```
#Print the column names of our data:

print(data.columns)
```

```
Index(['Unnamed: 0', 'inst', 'time', 'status', 'age', 'sex', 'ph.ecog',
       'ph.karno', 'pat.karno', 'meal.cal', 'wt.loss'],
      dtype='object')
```

Figure 11: Printing the data columns.

### 4) Get additional information about the dataset:

It gives us information about the data type of the columns along with their null-value counter. We need to remove the rows with a null value for some of the survival analysis methods.

```
#Additional info about our dataset:

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228 entries, 0 to 227
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  228 non-null    int64
 1   inst        227 non-null    float64
 2   time        228 non-null    int64
 3   status      228 non-null    int64
 4   age         228 non-null    int64
 5   sex         228 non-null    int64
 6   ph.ecog     227 non-null    float64
 7   ph.karno    227 non-null    float64
 8   pat.karno   225 non-null    float64
 9   meal.cal    181 non-null    float64
 10  wt.loss     214 non-null    float64
dtypes: float64(6), int64(5)
memory usage: 19.7 KB
```

Figure 12: Additional info about our dataset.

### 5) Get statistical information about the dataset:

It gives us some statistical information like the total number of rows, mean, standard deviation, minimum value, 25th percentile, 50th percentile, 75th percentile, and maximum value for each column.

```
#Statistical info about our dataset:

data.describe()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 228.00000 | 227.000000 | 228.000000 | 228.000000 | 228.000000 | 228.000000 | 227.000000 | 227.000000 | 225.000000 | 181.000000 | 214.000000 |
| mean | 114.50000 | 11.088106 | 305.232456 | 1.723684 | 62.447368 | 1.394737 | 0.951542 | 81.938326 | 79.955556 | 928.779006 | 9.831776 |
| std | 65.96211 | 8.303491 | 210.645543 | 0.448159 | 9.073457 | 0.489870 | 0.717872 | 12.327955 | 14.623177 | 402.174707 | 13.139902 |
| min | 1.00000 | 1.000000 | 5.000000 | 1.000000 | 39.000000 | 1.000000 | 0.000000 | 50.000000 | 30.000000 | 96.000000 | -24.000000 |
| 25% | 57.75000 | 3.000000 | 166.750000 | 1.000000 | 56.000000 | 1.000000 | 0.000000 | 75.000000 | 70.000000 | 635.000000 | 0.000000 |
| 50% | 114.50000 | 11.000000 | 255.500000 | 2.000000 | 63.000000 | 1.000000 | 1.000000 | 80.000000 | 80.000000 | 975.000000 | 7.000000 |
| 75% | 171.25000 | 16.000000 | 396.500000 | 2.000000 | 69.000000 | 2.000000 | 1.000000 | 90.000000 | 90.000000 | 1150.000000 | 15.750000 |
| max | 228.00000 | 33.000000 | 1022.000000 | 2.000000 | 82.000000 | 2.000000 | 3.000000 | 100.000000 | 100.000000 | 2600.000000 | 68.000000 |

Figure 13: Obtaining statistical info about our dataset.

### 6) Find out sex distribution using histogram:

This gives us a general idea about how our data is distributed. In the following graph, we can see that around 139 values have a status of 1, and approximately 90 values have a status of 2, which means

that there are 139 males and around 90 females in our dataset.

```
#Plot histogram for sex of patient:

print (data["sex"].hist())
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

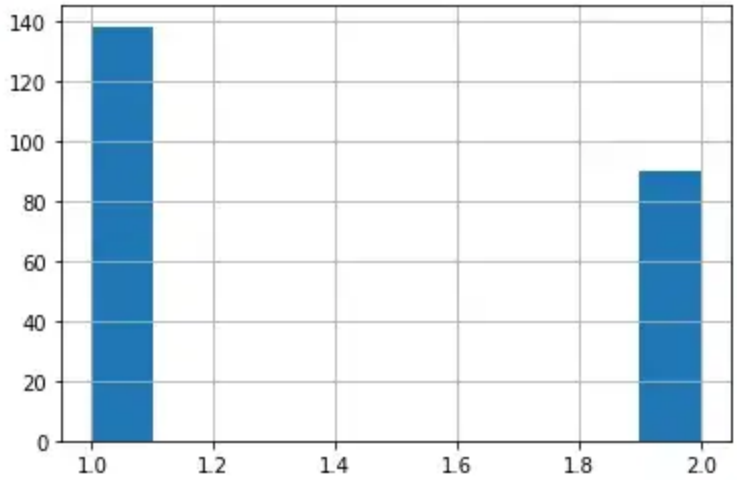


Figure 14: Plotting histogram for the sex of the patient.

## 7) Create an object for Kaplan-Meier-Fitter:

```
#Create an object of KaplanMeierFitter:

kmf = KaplanMeierFitter()
```

Figure 15: Creating an object for the Kaplan-Meier-Fitter.

## 8) Organize the data:

Now we need to organize our data. We will add a new column in our dataset that is called “dead.” It stores the data about whether a person that is a part of our experiment is dead or alive(based on the status value). If our status value is 1, then that person is alive, and if our status value is 2, then the person is dead. It is a crucial step for what we need to do in the next step as we are going to store our data in columns called censored and observed. Where observed data stores the value of dead persons in a specific timeline, and censored data stores the value of alive persons or persons that we are not going to investigate.

```
#Organize our data:

#If status = 1 , then dead = 0
#If status = 2 , then dead = 1

data.loc[data.status == 1, 'dead'] = 0
data.loc[data.status == 2, 'dead'] = 1

data.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN | 1.0 |
| 1 | 2 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 | 1.0 |
| 2 | 3 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 | 0.0 |
| 3 | 4 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 | 1.0 |
| 4 | 5 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 | 1.0 |

Figure 16: Organizing our data.

### 9) Fitting our data into an object:

Here our goal is to find the number of days a patient survived before they died. Our event of interest will be “death,” which is stored in the “dead” column. The first argument it takes is the timeline for our experiment.

```
#Fit the parameter values in our object:

kmf.fit(durations =  data["time"], event_observed = data["dead"])
```

```
<lifelines.KaplanMeierFitter:"KM_estimate", fitted with 228 total observations, 63 right-censored observations>
```

Figure 17: Fitting the parameter values in our object.

### 10) Generate event table:

One of the most crucial methods of the kmf object is the “event_table.” It gives us various information for our survival analysis. Let’s have a look at it column-by-column.

```
#Print the event table:

kmf.event_table

# Removed = Observed + Censored
# Censored = Person that didn't die.(They are of no use to us!)
# Observed = Persons that died.
```

| event_at | removed | observed | censored | entrance | at_risk |
|---|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 228 | 228 |
| 5.0 | 1 | 1 | 0 | 0 | 228 |
| 11.0 | 3 | 3 | 0 | 0 | 227 |
| 12.0 | 1 | 1 | 0 | 0 | 224 |
| 13.0 | 2 | 2 | 0 | 0 | 223 |
| ... | ... | ... | ... | ... | ... |
| 840.0 | 1 | 0 | 1 | 0 | 5 |
| 883.0 | 1 | 1 | 0 | 0 | 4 |
| 965.0 | 1 | 0 | 1 | 0 | 3 |
| 1010.0 | 1 | 0 | 1 | 0 | 2 |
| 1022.0 | 1 | 0 | 1 | 0 | 1 |

187 rows × 5 columns

Figure 18: Printing the event table.

**a) event_at:** It stores the value of the timeline for our dataset. i.e., when was the patient observed in our experiment or when was the experiment conducted. It can be several minutes, days, months, years, and others. In our case, it is going to be for many days. It stores the value of survival days for the subjects.

**b) at_risk:** It stores the number of current patients under observation. In the beginning, it will be the total number of patients we are going to observe in our experiment. If new patients are added at a particular time, then we have to increase their value accordingly. Therefore:

```
at_risk = Current patients at_risk + Entrance - Removed
```

Figure 19: at_risk variable formula.

**c) entrance:** It stores the value of new patients in a given timeline. It is possible that while experimenting, other patients are also diagnosed with the disease. To account for that, we have the entrance column.

**d) censored:** Our ultimate goal is to find the survival probability for a patient. At the end of the experiment, if the person is still alive, we will add him/her to the censored category. We have already discussed the types of censoring.

**e) observed:** It stores the value of the number of subjects that died during the experiment. From a broad perspective, these are the people who met our event of interest.

**f) removed:** It stores the values of patients that are no longer part of our experiment. If a person dies or is censored, then he/she falls into this category. In short, it is an addition of the data in the observed and censored category.

```
removed = Observed + Censored
```

Figure 20: removed variable formula.

### 11) Calculating the probability of survival for individual timelines:

Let's first see the formula for calculating the survival of a particular person at a given time.

$$S_t = \frac{Number\ of\ subjects\ at\ \ risk\ at\ the\ start - Number\ of\ subjects\ that\ died}{Number\ of\ subjects\ at\ risk\ at\ the\ start}$$

Figure 21: Calculating the probability of survival for individual timelines.

### a) Survival probability at t=0 only:

$$S_0 = \frac{228 - 0}{228} = 1$$

Figure 22: Calculating the survival probability of t=0.

```
#Calculating the survival probability for a given time:

event_at_0 = kmf.event_table.iloc[0,:]

#Calculate the survival probability for t=0:

surv_for_0 = (event_at_0.at_risk - event_at_0.observed)/event_at_0.at_risk
surv_for_0
```

```
1.0
```

Figure 23: Calculating the survival probability for a given time and t=0.

**b) Survival probability at t=5 only:**

$$S_5 = \frac{228 - 1}{228} = 0.9956$$

Figure 24: Survival probability of t=5.

```
#Calculating the survival probability for a given time:

event_at_5 = kmf.event_table.iloc[1,:]

#Calculate the survival probability for t=5:
surv_for_5 = (event_at_5.at_risk - event_at_5.observed)/event_at_5.at_risk
surv_for_5
```

```
0.9956140350877193
```

Figure 25: Calculating the survival probability for a given time.

**c) Survival probability at t=11 only:**

$$S_{11} = \frac{227 - 3}{227} = 0.9867$$

Figure 26: Survival probability at t=11.

```
#Calculating the survival probability for a given time:

event_at_11 = kmf.event_table.iloc[2,:]

#Calculate the survival probability for t=11:
surv_for_11 = (event_at_11.at_risk - event_at_11.observed)/event_at_11.at_risk
surv_for_11
```

```
0.986784140969163
```

Figure 27: Calculating the survival probability for a given time.

Now what we found here is the probability for a specific time. What we want is the probability for the entire time for a patient. i.e., the probability of patient surviving all the rounds of the experiment.

In a nutshell, we want to find the probability of a person surviving all of the time he lived after diagnosis. What we just found is the probability of a particular experiment only.

Let us take a straightforward example to understand the concept of conditional probability. For instance, we have a total of 15 balls in a non-transparent box. Out of the 15 balls, we have seven black balls, five red balls, and three green balls. Here is a pictorial view for that.

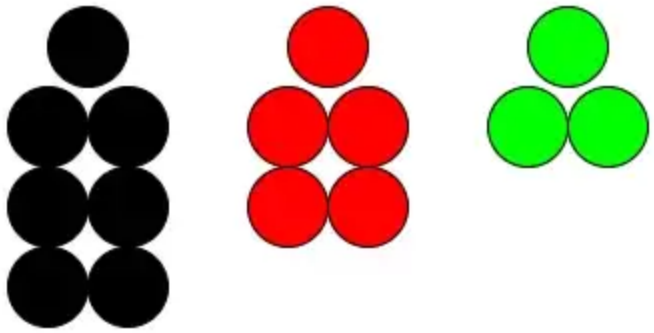

Figure 28: Ball figure example.

**a) Probability of choosing a red ball:**

$$P(r_1) = \frac{5}{15}$$

Figure 29: Probability of choosing a red ball.

**b) Probability of choosing the second red ball:**

Since we've removed a ball that was red, the total number of red balls we have is 4, and the total number of balls we have is 14.

$$P(r_2) = \frac{4}{14}$$

Figure 30: Probability of choosing the second red ball.

If our question is to find the probability of both the balls being red, we will multiply it, and that is precisely what we are going to do in survival analysis. We know that a patient has survived the 1st time interval, and we want to find the probability of him surviving the second time interval given that he has survived the 1st time interval. My point here is we do not want to find the probability of the second time interval only. We want the total probability of him surviving the entire period.

In our example, the probability of both balls being red is as following:

$$P(r_{12}) = P(r_1) * P(r_2) = \frac{5}{15} * \frac{4}{14}$$

Figure 31: Calculating the probability of both balls being red.

In survival analysis, we can write the formula as follows:

$$S(n) = S_0 * S_1 * S_2 * \ldots * S_n$$

Figure 32: Calculating S(n).

### 12) Finding survival probability:

We want to find the probability that a patient has survived through all the timeline till now. Now we need to find the actual survival probability for a patient.

### a) Survival probability for t=0:

$$S(0) = S_0 = 1$$

Figure 33: Calculating the survival probability for t=0.

```
#Calculating the actual survival probability at a given time:

surv_after_0 = surv_for_0
print("Survival Probability After 0 Days: ",surv_after_0)
```

```
Survival Probability After 0 Days:  1.0
```

Figure 34: Calculating the actual survival probability at a given time.

### b) Survival probability for t=5:

$$S(5) = S_5 * S_0 = 1 * 0.9956 = 0.9956$$

Figure 35: Calculating the survival probability for t=5.

```
#Calculating the actual survival probability at a given time:

surv_after_5 = surv_for_0 * surv_for_5
print("Survival Probability After 5 Days: ",surv_after_5)
```

```
Survival Probability After 5 Days:  0.9956140350877193
```

Figure 36:

## c) Survival probability for t=11:

$$S(11) = S_{11} * S_5 * S_0 = 0.9867 * 0.9965 * 1 = 0.9823$$

Figure 37: Calculating the survival probability for t=11.

```
#Calculating the actual survival probability at a given time:

surv_after_11 = surv_for_0 * surv_for_5 * surv_for_11
print("Survival Probability After 11 Days: ",surv_after_11)
```

```
Survival Probability After 11 Days:  0.9824561403508771
```

Figure 38: Calculating the actual survival probability at a given time.

## 13) Predicting the probability:

Now the kmf object's predict function does all of this work for us. However, it is always good practice to know the logic behind it.

```
#Get the probability values the easy way!

print("Survival probability for t=0: ",kmf.predict(0))
print("Survival probability for t=5: ",kmf.predict(5))
print("Survival probability for t=11: ",kmf.predict(11))
```

```
Survival probability for t=0:  1.0
Survival probability for t=5:  0.9956140350877193
Survival probability for t=11:  0.9824561403508766
```

Figure 39: Displaying the probability values the easy way.

## 14) Finding the survival probability for an array of the timeline:

```
#Predicting the surviaval probability for an array of value:

kmf.predict([0,5,11,12])
```

```
0     1.000000
5     0.995614
11    0.982456
12    0.978070
Name: KM_estimate, dtype: float64
```

Figure 40: Predicting the survival probability for an array of values.

### 15) Get survival probability for the whole timeline:

```
#To get the full list:

print(kmf.survival_function_)
```

```
          KM_estimate
timeline
0.0          1.000000
5.0          0.995614
11.0         0.982456
12.0         0.978070
13.0         0.969298
...               ...
840.0        0.067127
883.0        0.050346
965.0        0.050346
1010.0       0.050346
1022.0       0.050346

[187 rows x 1 columns]
```

Figure 41: Getting the survival probability for the whole timeline.

The survival probability for a patient at timeline 0 is 1. Grasping our thoughts, then we gather that the probability that a person dies on the 1st day of diagnosis is near equals to 0. So we can say that the survival probability is as high as possible. As the timeline increases, the probability of survival decreases for a patient.

### 16) Plot the graph:

```
#Plot the graph:

kmf.plot()
plt.title("The Kaplan-Meier Estimate")
plt.xlabel("Number of days")
plt.ylabel("Probability of survival")
```

```
Text(0, 0.5, 'Probability of survival')
```

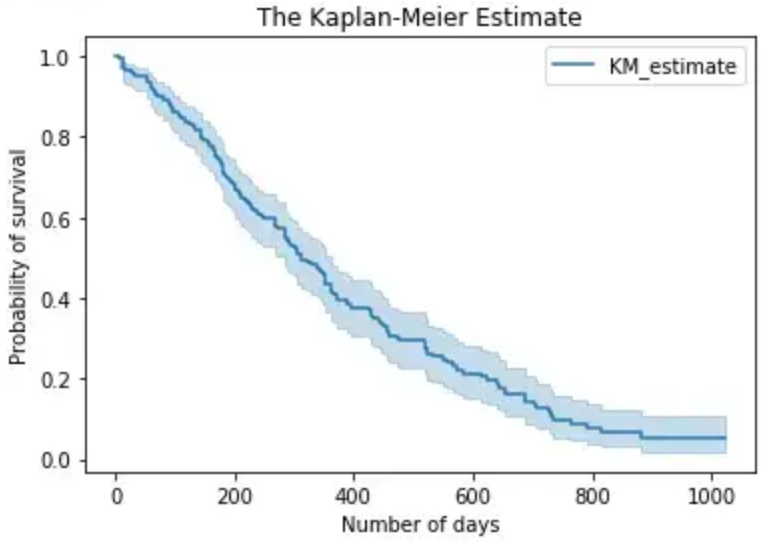


Figure 42: Plotting the probability of survival.

## 17) The median number of survival days:

It provides the number of days where, on average, 50% of the patients survived.

```
#The median number of days:

print("The median survival time: ",kmf.median_survival_time_)
```

```
The median survival time:  310.0
```

Figure 43: Displaying the median number of days.

From the code above, we can say that on average, a person lived 310 days after the day of diagnosis.

## 18) Survival probability with confidence interval:

```
#Survival probability with confidence interval:

kmf.confidence_interval_survival_function_
```

| | KM_estimate_lower_0.95 | KM_estimate_upper_0.95 |
|---|---|---|
| **0.0** | 1.000000 | 1.000000 |
| **5.0** | 0.969277 | 0.999381 |
| **11.0** | 0.953935 | 0.993379 |
| **12.0** | 0.948120 | 0.990813 |
| **13.0** | 0.936682 | 0.985244 |
| **...** | ... | ... |
| **840.0** | 0.030728 | 0.123060 |
| **883.0** | 0.017866 | 0.108662 |
| **965.0** | 0.017866 | 0.108662 |
| **1010.0** | 0.017866 | 0.108662 |
| **1022.0** | 0.017866 | 0.108662 |

187 rows × 2 columns

Figure 44: Estimating the survival probability with a confidence interval.

## 19) Graph for survival probability with confidence interval:

```
#Plot survival function with confidence interval:

confidence_surv_func = kmf.confidence_interval_survival_function_

plt.plot(confidence_surv_func["KM_estimate_lower_0.95"],label="Lower")
plt.plot(confidence_surv_func["KM_estimate_upper_0.95"],label="Upper")
plt.title("Survival Function With Confidence Interval")
plt.xlabel("Number of days")
plt.ylabel("Survival Probability")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc659ef1cc8>
```

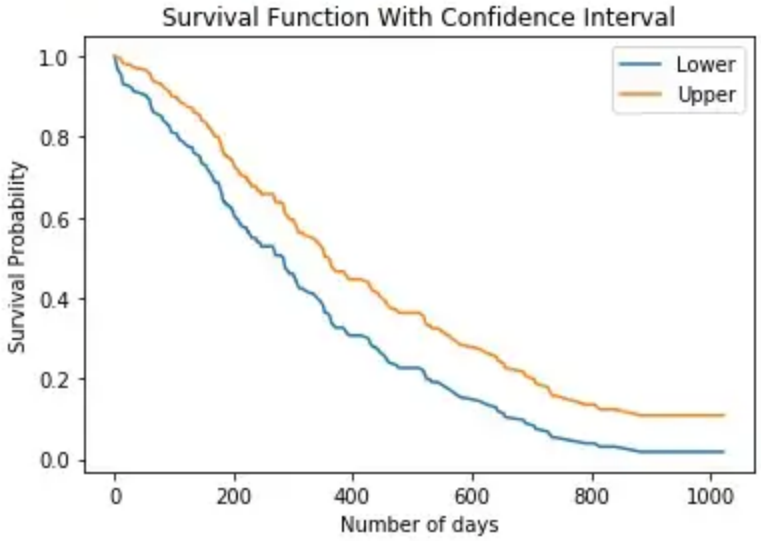


Figure 45: Plotting the survival function with a confidence interval.

Now all the information we have is for the survival of a person. Now we will see what is the probability for a person to die at a specific timeline. Here notice that a higher survival probability is suitable for a person, but higher cumulative density (probability of a person to die) is not so good!

**20) Probability of a person dying:**

$$P_{die(t)} = \frac{Number\ of\ subjects\ that\ died\ at\ time\ t}{Number\ of\ subjects\ at\ risk\ just\ before\ time\ t}$$

Figure 46: Probability of a person dying.

Here the denominator value is subjected at risk in the previous row.

The formula for cumulative density:

$$P\big(die(t_i)\big) = \sum_{t_i \le t} P_{die(t_i)}$$

Figure 47: The formula for cumulative density.

a) Probability of a person dying at t=0:

$$P_{die(0)} = \frac{0}{228} = 0$$

Figure 48: Probability of a person dying at t=0.

b) Probability of a person dying at t=5:

$$P_{die(5)} = \frac{1}{228} = 0.004385$$

Figure 49: Probability of a person dying at t=5.

c) Probability of a person dying at t=11:

$$P_{die(11)} = \frac{3}{228} = 0.013157$$

Figure 50: Probability of a person dying at t=11.

Find the cumulative density:

d) Cumulative density at t=0:

$$P\big(die(0)\big) = P_{die(0)} = 0$$

Figure 51: Calculating the cumulative density at t=0.

e) Cumulative density at t=5:

$$P\big(die(5)\big) = P_{die(0)} + P_{die(5)} = 0 + 0.004385 = 0.004385$$

Figure 52: Calculating the cumulative density at t=5.

f) Cumulative density at t=11:

$$P(die(11)) = P_{die(0)} + P_{die(5)} + P_{die(11)} = 0 + 0.004385 + 0.01315$$
$$= 0.01754$$

Figure 53: Calculate the cumulative density at t=11.

```
#Probabaility of a subject dying:
#p(1022) = p(0) +.....+p(1022)

kmf.cumulative_density_
```

| timeline | KM_estimate |
|---|---|
| 0.0 | 0.000000 |
| 5.0 | 0.004386 |
| 11.0 | 0.017544 |
| 12.0 | 0.021930 |
| 13.0 | 0.030702 |
| ... | ... |
| 840.0 | 0.932873 |
| 883.0 | 0.949654 |
| 965.0 | 0.949654 |
| 1010.0 | 0.949654 |
| 1022.0 | 0.949654 |

187 rows × 1 columns

Figure 54: Displaying the probability of a subject dying.

## 21) Plot the graph for cumulative density:

```
#Plot the cumulative density graph:

kmf.plot_cumulative_density()
plt.title("Cumulative Density Plot")
plt.xlabel("Number of days")
plt.ylabel("Probability of person's death")
```

```
Text(0, 0.5, "Probability of person's death")
```

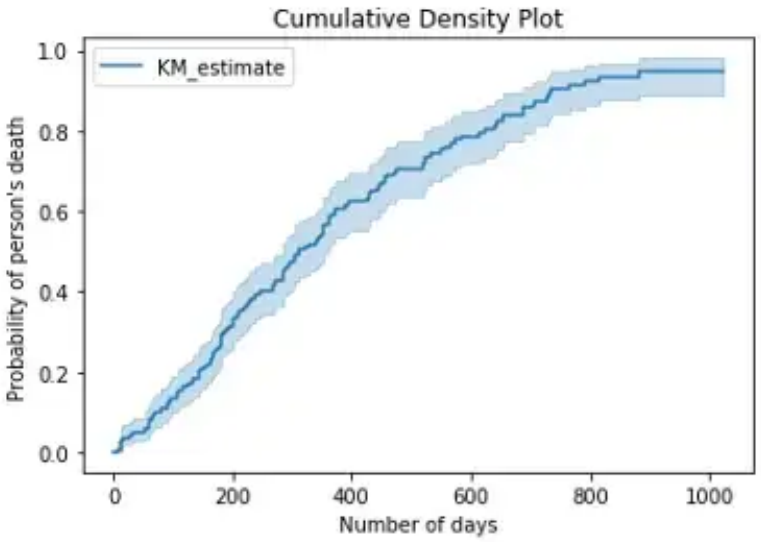


Figure 55: Plotting the cumulative density.

Notice that, as the number of survival days increases the probability of a person dying increases.

## 22) The cumulative density with confidence interval:

```
#Cumulative density with confidence interval:

kmf.confidence_interval_cumulative_density_
```

| | KM_estimate_lower_0.95 | KM_estimate_upper_0.95 |
|---|---|---|
| **0.0** | 0.000000 | 0.000000 |
| **5.0** | 0.030723 | 0.000619 |
| **11.0** | 0.046065 | 0.006621 |
| **12.0** | 0.051880 | 0.009187 |
| **13.0** | 0.063318 | 0.014756 |
| **...** | ... | ... |
| **840.0** | 0.969272 | 0.876940 |
| **883.0** | 0.982134 | 0.891338 |
| **965.0** | 0.982134 | 0.891338 |
| **1010.0** | 0.982134 | 0.891338 |
| **1022.0** | 0.982134 | 0.891338 |

187 rows × 2 columns

Figure 56: Calculate the cumulative density with a confidence interval.

## 23) Graph for cumulative density with a confidence interval:

```
#Plot cumulative density with confidence interval:

confidence_cumulative_density = kmf.confidence_interval_cumulative_density_

plt.plot(kmf.confidence_interval_cumulative_density_["KM_estimate_lower_0.95"],label="Lower")
plt.plot(kmf.confidence_interval_cumulative_density_["KM_estimate_upper_0.95"],label="Upper")
plt.title("Cumulative Density With Confidence Interval")
plt.xlabel("Number of days")
plt.ylabel("Cumulative Density")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc65b6c2148>
```

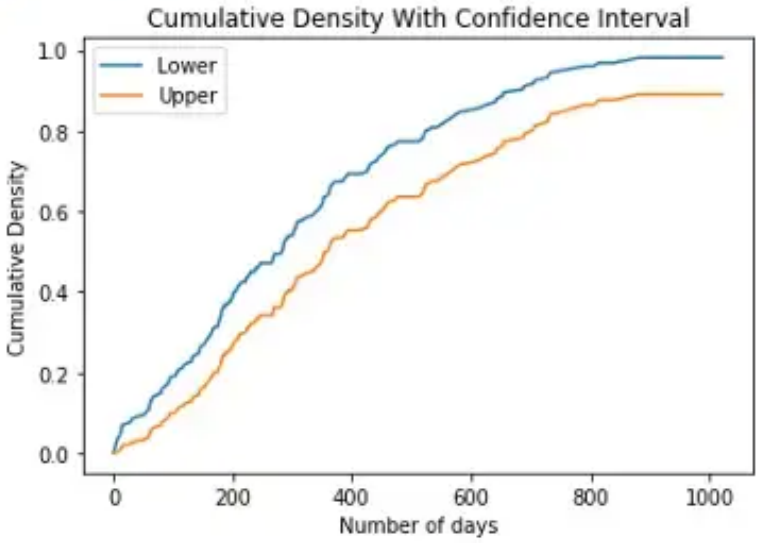


Figure 57: Plotting the cumulative density with a confidence interval.

## 24) Get cumulative density for a particular day:

```
#Find cumulative density at a specific time:

kmf.cumulative_density_at_times(times=1022)
```

```
1022    0.949654
Name: KM_estimate, dtype: float64
```

Figure 58: Find cumulative density at a specific time.

## 25) The median time to an event:

We can get the amount of time remaining from the median survival time.

```
#Conditional median time to event of interest:

kmf.conditional_time_to_event_
```

| timeline | KM_estimate - Conditional median duration remaining to event |
|---|---|
| 0.0 | 310.0 |
| 5.0 | 305.0 |
| 11.0 | 309.0 |
| 12.0 | 308.0 |
| 13.0 | 316.0 |
| ... | ... |
| 840.0 | inf |
| 883.0 | inf |
| 965.0 | inf |
| 1010.0 | inf |
| 1022.0 | inf |

187 rows × 1 columns

Figure 59: Calculating the conditional median time to an event of interest.

## 26) Graph for the median time to the event:

```
#Conditional median time left for event:

median_time_to_event = kmf.conditional_time_to_event_
plt.plot(median_time_to_event,label="Median Time left")
plt.title("Medain time to event")
plt.xlabel("Total days")
plt.ylabel("Conditional median time to event")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc65a0461c8>
```

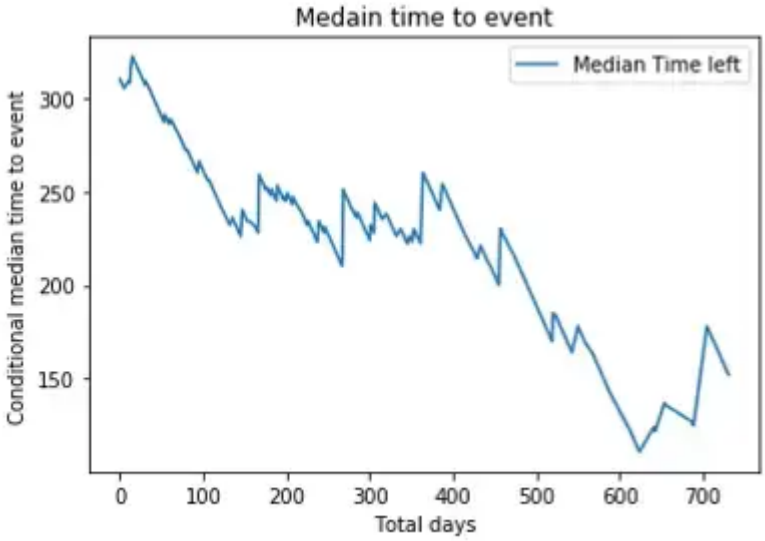


Figure 60: Plotting the graph for the median time to an event.

## 3. Estimating hazard rates using Nelson-Aalen

**Hazard function H(t):**

Until now, we discussed the Kaplan-Meier survival function. Using that, we can get the probability of the event of interest (death in our case) not occurring by that time. The survival functions are a great way to summarize and visualize the survival dataset; however, it is not the only way. We can visualize the aggregate information on survival using the Nelson-Aalen hazard function h(t). The hazard function h(t) gives us the probability that a subject under observation at time t has an event of interest (death) at that time. To get the information about the hazard function, we cannot transform the Kaplan-Meier estimator. For that, there is a proper nonparametric estimator of the *cumulative* hazard function:

Cumulative Hazard Function:

$$\hat{H}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i}$$

Where,

$\hat{H}(t)$ = Cumulative hazard probability

$d_i$ = The number of events at time $t_i$

$n_i$ = The number of subjects alive at time $t_i$

Figure 61: Formula to calculate the cumulative hazard function.

### 1) Import required libraries:

```
#Hazard function:

from lifelines import NelsonAalenFitter
```

Figure 62: Importing NelsonAalenFitter from lifelines.

### 2) Create an object of Nelson-Aalen-Fitter:

```
#Create an object of NelsonAalenFitter:

naf = NelsonAalenFitter()
```

Figure 63: Creating an object of Nelson-Aalen-Fitter.

### 3) Fitting the data:

```
#Fit our data into the object:

naf.fit(data["time"], event_observed=data["dead"])
```

```
<lifelines.NelsonAalenFitter:"NA_estimate", fitted with 228 total observations, 63 right-censored observations>
```

Figure 64: Fitting the data into the object.

### 4) Finding the cumulative hazard:

Here we'll use the event table generated in the previous part to understand how the hazard function actually works.

```
#Print the event table:

kmf.event_table

# Removed = Observed + Censored
# Censored = Person that didn't die.(They are of no use to us!)
# Observed = Persons that died.
```

| event_at | removed | observed | censored | entrance | at_risk |
|---|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 228 | 228 |
| 5.0 | 1 | 1 | 0 | 0 | 228 |
| 11.0 | 3 | 3 | 0 | 0 | 227 |
| 12.0 | 1 | 1 | 0 | 0 | 224 |
| 13.0 | 2 | 2 | 0 | 0 | 223 |
| ... | ... | ... | ... | ... | ... |
| 840.0 | 1 | 0 | 1 | 0 | 5 |
| 883.0 | 1 | 1 | 0 | 0 | 4 |
| 965.0 | 1 | 0 | 1 | 0 | 3 |
| 1010.0 | 1 | 0 | 1 | 0 | 2 |
| 1022.0 | 1 | 0 | 1 | 0 | 1 |

187 rows × 5 columns

Figure 65: Finding the cumulative hazard.

Here is the formula to find the non-cumulative hazard probability at a specific time:

$$H_t = \frac{Number\ of\ subjects\ that\ died}{Number\ of\ subjects\ at\ risk}$$

Figure 66: Formula to calculate the non-cumulative hazard probability at a specific time.

a) Finding the hazard probability at t=0:

$$H_0 = \frac{0}{228} = 0$$

Figure 67: Finding the hazard probability at t=0.

b) Finding the hazard probability at t=5:

$$H_5 = \frac{1}{228} = 0.004385$$

Figure 68: Finding the hazard probability at t=5.

c) Finding the hazard probability at t=11:

$$H_{11} = \frac{3}{227} = 0.013215$$

Figure 69: Finding the hazard probability at t-11.

d) Finding the cumulative hazard probability at t=0:

$$H(0) = H_0 = 0$$

Figure 70: Finding the cumulative hazard probability at t=0.

e) Finding the cumulative hazard probability at t=5:

$$H(5) = H_5 + H_0 = 0.004385 + 0 = 0.004385$$

Figure 71: Finding the cumulative hazard probability at t=5.

f) Finding the cumulative hazard probability at t=11:

$$H(11) = H_{11} + H_5 + H_0 = 0.013215 + 0.004385 + 0 = 0.01766$$

Figure 72: Finding the cumulative hazard probability at t=11.

```
#Print the cumulative hazard:

naf.cumulative_hazard_
```

| timeline | NA_estimate |
|---|---|
| 0.0 | 0.000000 |
| 5.0 | 0.004386 |
| 11.0 | 0.017660 |
| 12.0 | 0.022125 |
| 13.0 | 0.031114 |
| ... | ... |
| 840.0 | 2.641565 |
| 883.0 | 2.891565 |
| 965.0 | 2.891565 |
| 1010.0 | 2.891565 |
| 1022.0 | 2.891565 |

187 rows × 1 columns

Figure 73: Displaying the cumulative hazard.

## 5) Plot the graph for cumulative hazard:

```
#Plot the cumulative hazard grpah:

naf.plot_cumulative_hazard()
plt.title("Cumulative Probability for Event of Interest")
plt.xlabel("Number of days")
plt.ylabel("Cumulative Probability of person's death")
```

```
Text(0, 0.5, "Cumulative Probability of person's death")
```

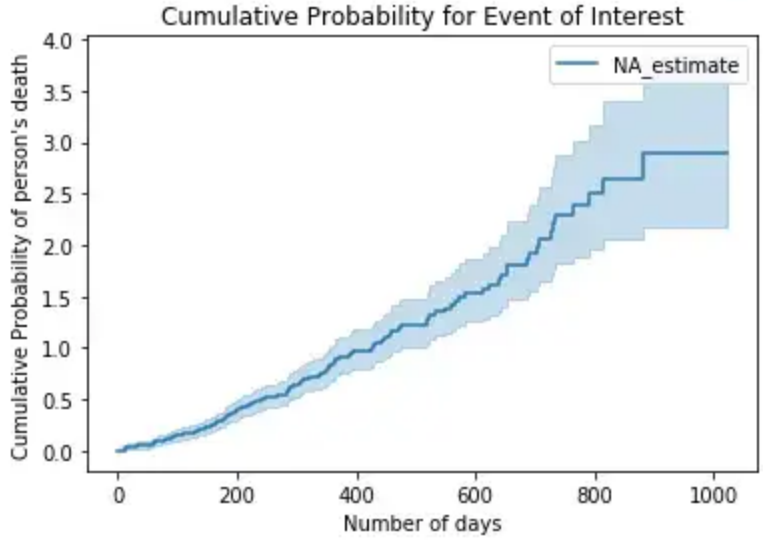

Figure 74: Plot the graph for cumulative hazards.

The cumulative hazard has a less clear understanding than the survival functions, but the hazard functions are based on more advanced survival analysis techniques.

## 6) Predict a value:

```
#We can predict the value at a certain point :

print("Time = 500 days: ",naf.predict(500))
print("Time = 1022 days: ",naf.predict(1022))
```

```
Time = 500 days:  1.219546171331098
Time = 1022 days:  2.8915648373471052
```

Figure 75: Predicting the value of a certain point.

## 7) Cumulative hazard probability with confidence interval:

```
#Cumulative hazard with confidence interval:

naf.confidence_interval_
```

| | NA_estimate_lower_0.95 | NA_estimate_upper_0.95 |
|---|---|---|
| **0.0** | 0.000000 | 0.000000 |
| **5.0** | 0.000618 | 0.031136 |
| **11.0** | 0.006628 | 0.047055 |
| **12.0** | 0.009209 | 0.053156 |
| **13.0** | 0.014832 | 0.065266 |
| **...** | ... | ... |
| **840.0** | 2.058738 | 3.389389 |
| **883.0** | 2.176987 | 3.840697 |
| **965.0** | 2.176987 | 3.840697 |
| **1010.0** | 2.176987 | 3.840697 |
| **1022.0** | 2.176987 | 3.840697 |

187 rows × 2 columns

Figure 76: Calculating the cumulative hazard probability with a confidence interval.

## 8) Graph for cumulative hazard probability with confidence interval:

```
#Plot cumulative hazard with confidence interval:

confidence_interval = naf.confidence_interval_

plt.plot(confidence_interval["NA_estimate_lower_0.95"],label="Lower")
plt.plot(confidence_interval["NA_estimate_upper_0.95"],label="Upper")
plt.title("Cumulative hazard With Confidence Interval")
plt.xlabel("Number of days")
plt.ylabel("Cumulative hazard")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc65a1c6fc8>
```

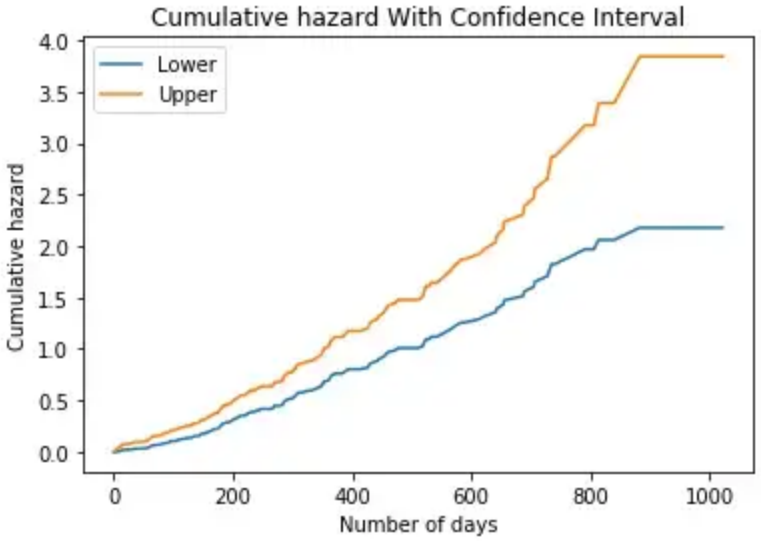


Figure 77: Plotting the confidence interval.

## 9) Cumulative hazard vs. cumulative density:

```
#Plot the cumulative_hazard and cumulative density:

kmf.plot_cumulative_density(label="Cumulative Hazard")
naf.plot_cumulative_hazard(label="Cumulative Density")
plt.xlabel("Number of Days")
```

```
Text(0.5, 0, 'Number of Days')
```

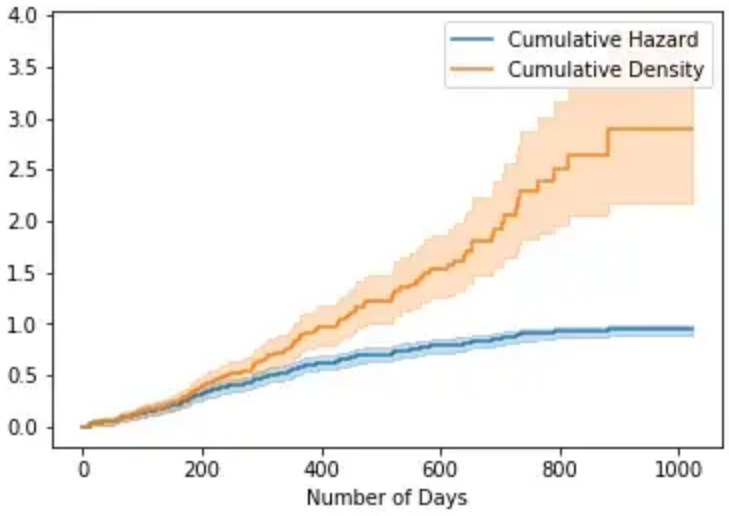

Figure 78: Plotting the cumulative hazard and cumulative density

## 4. Kaplan-Meier Estimator with groups

Until now, we saw how we could find the survival probability and hazard probability for all of our observations. Now it is time to perform some analysis on our data to determine whether there is any difference in survival probability if we divide our data into groups based on specific characteristics. Let's divide our data into two groups based on sex: Male and Female. Our goal here is to check is there any significant difference in survival rate if we divide our dataset based on sex. Later in this tutorial, we will see on what basis do we divide the data into groups.

### 1) Import required libraries:

```
#Import required libraries:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifelines import KaplanMeierFitter
```

Figure 79: Importing pandas, numpy, matplotlib.pyplot, and KaplanMeierFitter from lifelines in Python.

### 2) Read the dataset:

```
#Read the dataset:

data = pd.read_csv("lung.csv")
data.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN |
| 1 | 2 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 |
| 2 | 3 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 |
| 3 | 4 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 |
| 4 | 5 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 |

Figure 80: Reading the dataset.

### 3) Organize our data:

```
#Organize our data:

#If status = 1 , then dead = 0
#If status = 2 , then dead = 1

data.loc[data.status == 1, 'dead'] = 0
data.loc[data.status == 2, 'dead'] = 1

data.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN | 1.0 |
| 1 | 2 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 | 1.0 |
| 2 | 3 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 | 0.0 |
| 3 | 4 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 | 1.0 |
| 4 | 5 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 | 1.0 |

Figure 81: Organizing the data.

## 4) Create two objects of Kaplan-Meier-Fitter():

```
#Create two objects for groups:

#kmf_m for male data:
#kmf_f for female data:

kmf_m = KaplanMeierFitter()
kmf_f = KaplanMeierFitter()
```

Figure 82: Creating the two objects of the Kaplan-Meier-Fitter.

## 5) Divide the data into groups:

```
#Dividing data into groups:

Male = data.query("sex == 1")
Female = data.query("sex == 2")
```

Figure 83: Dividing the data into groups.

## 6) Male data:

```
#View data of Male group:

Male.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN | 1.0 |
| 1 | 2 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 | 1.0 |
| 2 | 3 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 | 0.0 |
| 3 | 4 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 | 1.0 |
| 4 | 5 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 | 1.0 |

Figure 84: Viewing the data of the male group.

## 7) Female data:

```
#View data of Female group:

Female.head()
```

| | Unnamed: 0 | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 7 | 7.0 | 310 | 2 | 68 | 2 | 2.0 | 70.0 | 60.0 | 384.0 | 10.0 | 1.0 |
| 7 | 8 | 11.0 | 361 | 2 | 71 | 2 | 2.0 | 60.0 | 80.0 | 538.0 | 1.0 | 1.0 |
| 11 | 12 | 16.0 | 654 | 2 | 68 | 2 | 2.0 | 70.0 | 70.0 | NaN | 23.0 | 1.0 |
| 12 | 13 | 11.0 | 728 | 2 | 68 | 2 | 1.0 | 90.0 | 90.0 | NaN | 5.0 | 1.0 |
| 18 | 19 | 1.0 | 61 | 2 | 56 | 2 | 2.0 | 60.0 | 60.0 | 238.0 | 10.0 | 1.0 |

Figure 85: Displaying the data of the female group.

## 8) Fit data into our objects:

```
#Fit data into objects:

kmf_m.fit(durations =  Male["time"],event_observed = Male["dead"] ,label="Male")
kmf_f.fit(durations =  Female["time"],event_observed = Female["dead"], label="Female")
```

```
<lifelines.KaplanMeierFitter:"Female", fitted with 90 total observations, 37 right-censored observations>
```

Figure 86: Fitting the male and female data into objects.

## 9) Event table for the male group:

```
#Event table for male group:

kmf_m.event_table
```

| event_at | removed | observed | censored | entrance | at_risk |
|---|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 138 | 138 |
| 11.0 | 3 | 3 | 0 | 0 | 138 |
| 12.0 | 1 | 1 | 0 | 0 | 135 |
| 13.0 | 2 | 2 | 0 | 0 | 134 |
| 15.0 | 1 | 1 | 0 | 0 | 132 |
| ... | ... | ... | ... | ... | ... |
| 814.0 | 1 | 1 | 0 | 0 | 5 |
| 840.0 | 1 | 0 | 1 | 0 | 4 |
| 883.0 | 1 | 1 | 0 | 0 | 3 |
| 1010.0 | 1 | 0 | 1 | 0 | 2 |
| 1022.0 | 1 | 0 | 1 | 0 | 1 |

120 rows × 5 columns

Figure 87: Event table for the male group.

## 10) Event table for the female group:

```
#Event table for female group:

kmf_f.event_table
```

| event_at | removed | observed | censored | entrance | at_risk |
|---|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 90 | 90 |
| 5.0 | 1 | 1 | 0 | 0 | 90 |
| 60.0 | 1 | 1 | 0 | 0 | 89 |
| 61.0 | 1 | 1 | 0 | 0 | 88 |
| 62.0 | 1 | 1 | 0 | 0 | 87 |
| ... | ... | ... | ... | ... | ... |
| 735.0 | 1 | 1 | 0 | 0 | 5 |
| 740.0 | 1 | 0 | 1 | 0 | 4 |
| 765.0 | 1 | 1 | 0 | 0 | 3 |
| 821.0 | 1 | 0 | 1 | 0 | 2 |
| 965.0 | 1 | 0 | 1 | 0 | 1 |

88 rows × 5 columns

Figure 88: Event table for the female group.

## 11) Predicting survival probabilities:

Now we can predict the survival probability for both the groups.

```
#Predict value based on time:

kmf_m.predict(11)
```

```
0.9782608695652172
```

Figure 89: Predicting the value based on time.

```
#Predict value based on time:

kmf_f.predict(11)
```

```
0.9888888888888887
```

Figure 90: Predicting the value based on time.

## 12) Get the complete list of survival probabilities:

## a) Survival probability for a male group:

```
#Get complete data of survival function for male group:

kmf_m.survival_function_
```

| | Male |
|---|---|
| timeline | |
| 0.0 | 1.000000 |
| 11.0 | 0.978261 |
| 12.0 | 0.971014 |
| 13.0 | 0.956522 |
| 15.0 | 0.949275 |
| ... | ... |
| 814.0 | 0.053571 |
| 840.0 | 0.053571 |
| 883.0 | 0.035714 |
| 1010.0 | 0.035714 |
| 1022.0 | 0.035714 |

120 rows × 1 columns

Figure 91: Get complete data of the survival function for the male group.

## b) Survival probability for the female group:

```
#Get complete data of survival function for female group:

kmf_f.survival_function_
```

| timeline | Female |
|---|---|
| 0.0 | 1.000000 |
| 5.0 | 0.988889 |
| 60.0 | 0.977778 |
| 61.0 | 0.966667 |
| 62.0 | 0.955556 |
| ... | ... |
| 735.0 | 0.124822 |
| 740.0 | 0.124822 |
| 765.0 | 0.083214 |
| 821.0 | 0.083214 |
| 965.0 | 0.083214 |

88 rows × 1 columns

Figure 92: Get complete data of the survival function for the male group.

## 13) Plot the graph for survival probabilities:

```
#Plot the survival_function data:

kmf_m.plot()
kmf_f.plot()

plt.xlabel("Days Passed")
plt.ylabel("Survival Probability")
plt.title("KMF")
```

```
Text(0.5, 1.0, 'KMF')
```

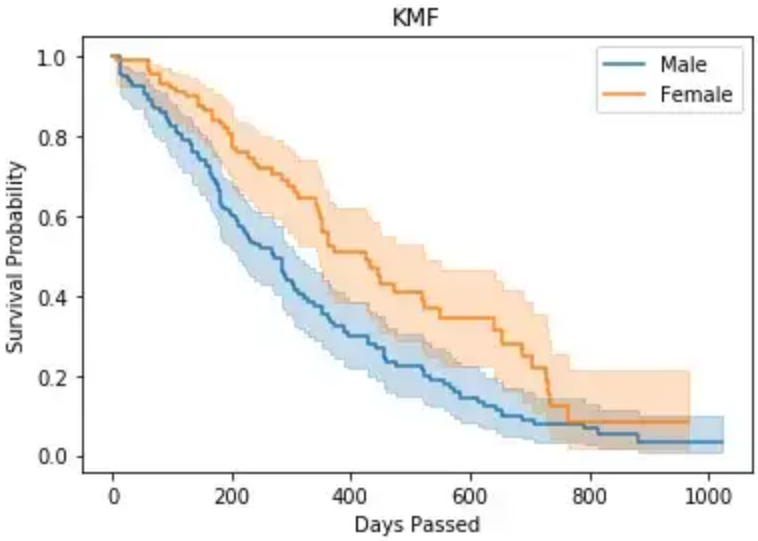


Figure 93: Plotting the graph for survival probabilities.

Here we can notice that the probability of females surviving lung cancer is higher than that of males. Therefore, from this data, we can say that medical researchers should focus more on the factors that lead to male patients' poor survival rates.

**14) Get the cumulative density:**

**a) For the male group:**

```
#Cumulative density for male group:

kmf_m.cumulative_density_
```

| timeline | Male |
|---|---|
| 0.0 | 0.000000 |
| 11.0 | 0.021739 |
| 12.0 | 0.028986 |
| 13.0 | 0.043478 |
| 15.0 | 0.050725 |
| ... | ... |
| 814.0 | 0.946429 |
| 840.0 | 0.946429 |
| 883.0 | 0.964286 |
| 1010.0 | 0.964286 |
| 1022.0 | 0.964286 |

120 rows × 1 columns

Figure 94: Cumulative density for the male group.

## b) For the female group:

```
#Cumulative density for female group:

kmf_f.cumulative_density_
```

| timeline | Female |
|---|---|
| 0.0 | 0.000000 |
| 5.0 | 0.011111 |
| 60.0 | 0.022222 |
| 61.0 | 0.033333 |
| 62.0 | 0.044444 |
| ... | ... |
| 735.0 | 0.875178 |
| 740.0 | 0.875178 |
| 765.0 | 0.916786 |
| 821.0 | 0.916786 |
| 965.0 | 0.916786 |

88 rows × 1 columns

Figure 95: Cumulative density for the female group.

## 15) Plot the graph for cumulative density:

```
#PLot the graph for cumulative density for both groups:

kmf_m.plot_cumulative_density()
kmf_f.plot_cumulative_density()
plt.title("Cumulative Density")
plt.xlabel("Number of days")
plt.ylabel("Probability")
```

```
Text(0, 0.5, 'Probability')
```

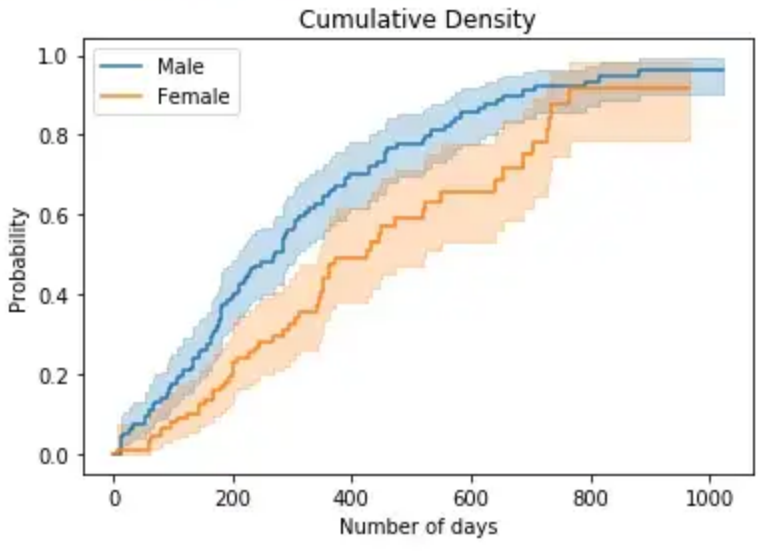

Figure 96: Plotting the graph for cumulative density for both groups.

## 16) Hazard function:

```
#Hazard Function:

from lifelines import NelsonAalenFitter
```

Figure 97: Importing the NelsonAalenFitter.

## 17) Fit the data into our objects:

```
#Fitting the data into objects:

naf_m = NelsonAalenFitter()
naf_f = NelsonAalenFitter()

naf_m.fit(Male["time"],event_observed = Male["dead"])
naf_f.fit(Female["time"],event_observed = Female["dead"])
```

```
<lifelines.NelsonAalenFitter:"NA_estimate", fitted with 90 total observations, 37 right-censored observations>
```

Figure 98: Fitting the data into our objects.

## 18) Cumulative hazard probability:

## a) For the male group:

```
#Cumulative hazard for male group:

naf_m.cumulative_hazard_
```

| timeline | NA_estimate |
|---|---|
| 0.0 | 0.000000 |
| 11.0 | 0.021899 |
| 12.0 | 0.029306 |
| 13.0 | 0.044287 |
| 15.0 | 0.051863 |
| ... | ... |
| 814.0 | 2.831337 |
| 840.0 | 2.831337 |
| 883.0 | 3.164670 |
| 1010.0 | 3.164670 |
| 1022.0 | 3.164670 |

120 rows × 1 columns

Figure 99: Cumulative hazard for the male group.

## b) For the female group:

```
#Cumulative hazard for female group:

naf_f.cumulative_hazard_
```

| timeline | NA_estimate |
|---|---|
| 0.0 | 0.000000 |
| 5.0 | 0.011111 |
| 60.0 | 0.022347 |
| 61.0 | 0.033711 |
| 62.0 | 0.045205 |
| ... | ... |
| 735.0 | 1.988977 |
| 740.0 | 1.988977 |
| 765.0 | 2.322310 |
| 821.0 | 2.322310 |
| 965.0 | 2.322310 |

88 rows × 1 columns

Figure 100: Cumulative hazard for the female group.

## 19) Plot the graph for cumulative hazard probability:

```
#Plot the graph for cumulative hazard:

naf_m.plot_cumulative_hazard(label="Male")
naf_f.plot_cumulative_hazard(label="Female")
plt.title("Cumulative Hazard Plot")
plt.xlabel("Number of Days")
plt.ylabel("Cumulative Hazard")
```

```
Text(0, 0.5, 'Cumulative Hazard')
```

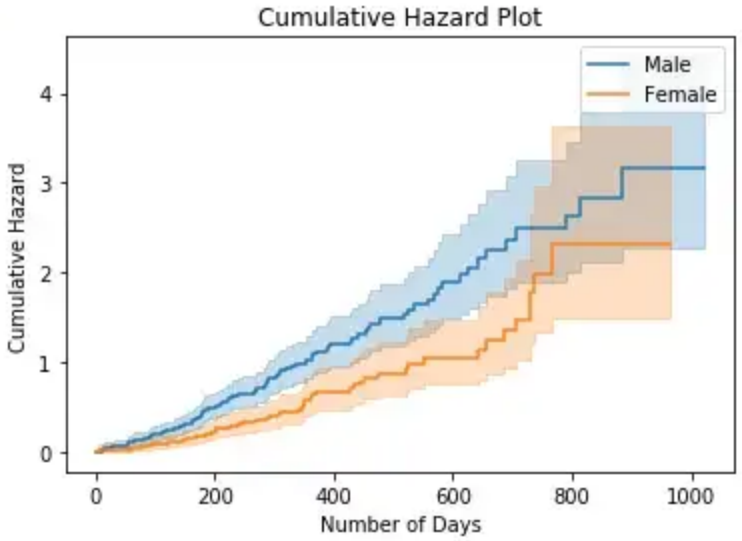


Figure 101: Plotting the graph for cumulative hazard.

## 20) The median time to event for the male group:

```
#Conditional median time to event of interest:

kmf_m.conditional_time_to_event_
```

| timeline | Male - Conditional median duration remaining to event |
|---|---|
| 0.0 | 270.0 |
| 11.0 | 272.0 |
| 12.0 | 271.0 |
| 13.0 | 271.0 |
| 15.0 | 270.0 |
| ... | ... |
| 814.0 | inf |
| 840.0 | inf |
| 883.0 | inf |
| 1010.0 | inf |
| 1022.0 | inf |

120 rows × 1 columns

Figure 102: Finding the conditional median to the event of interest.

## 21) The median time to event graph for the male group:

```
#Conditional median time left for event for male group:

median_time_to_event = kmf_m.conditional_time_to_event_
plt.plot(median_time_to_event,label="Median Time left")
plt.title("Medain time to event")
plt.xlabel("Total days")
plt.ylabel("Conditional median time to event")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc50dfd6808>
```

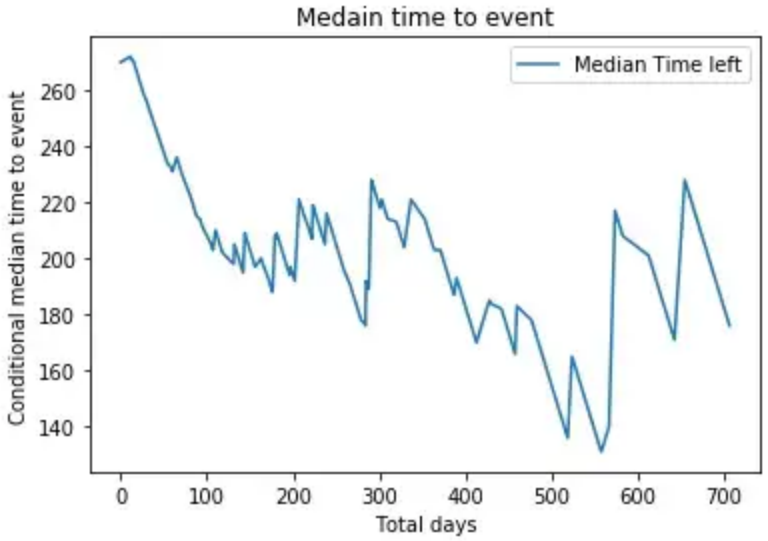


Figure 103: Plotting the conditional time to event for the male group.

## 22) The median time to event for the female group:

```
#Conditional median time to event of interest for female group:

kmf_f.conditional_time_to_event_
```

Female - Conditional median duration remaining to event

| timeline | |
|---|---|
| 0.0 | 426.0 |
| 5.0 | 421.0 |
| 60.0 | 373.0 |
| 61.0 | 372.0 |
| 62.0 | 371.0 |
| ... | ... |
| 735.0 | inf |
| 740.0 | inf |
| 765.0 | inf |
| 821.0 | inf |
| 965.0 | inf |

88 rows × 1 columns

Figure 104: Finding the conditional median time to event of interest for the female group.

## 23) The median time to event graph for the female group:

```
#Conditional median time left for event for female group:

median_time_to_event = kmf_f.conditional_time_to_event_
plt.plot(median_time_to_event,label="Median Time left")
plt.title("Medain time to event")
plt.xlabel("Total days")
plt.ylabel("Conditional median time to event")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc50e14a748>
```

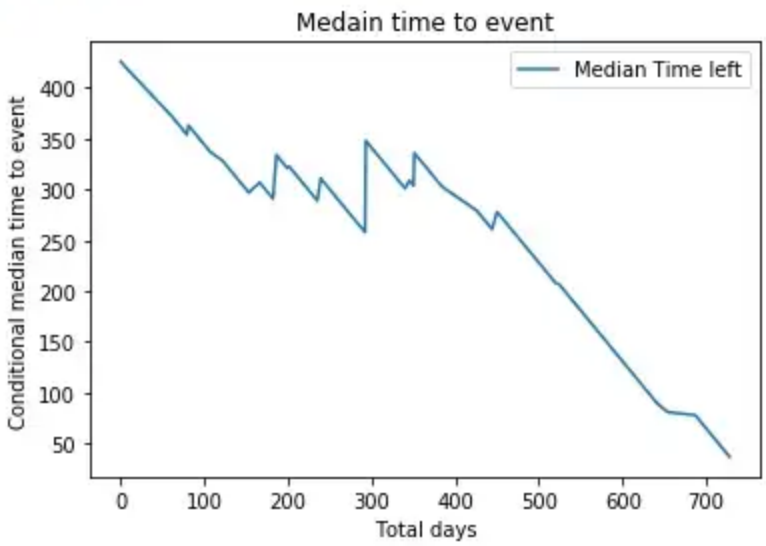

Figure 105: Conditional median time left for an event for the female group.

## 24) Survival probability with a confidence interval for the male group:

```
#Survival probability with confidence interval for male group:

kmf_m.confidence_interval_survival_function_
```

| | Male_lower_0.95 | Male_upper_0.95 |
|---|---|---|
| **0.0** | 1.000000 | 1.000000 |
| **11.0** | 0.934122 | 0.992937 |
| **12.0** | 0.924619 | 0.989022 |
| **13.0** | 0.905787 | 0.980229 |
| **15.0** | 0.896549 | 0.975490 |
| **...** | ... | ... |
| **814.0** | 0.019335 | 0.114081 |
| **840.0** | 0.019335 | 0.114081 |
| **883.0** | 0.008603 | 0.096828 |
| **1010.0** | 0.008603 | 0.096828 |
| **1022.0** | 0.008603 | 0.096828 |

120 rows × 2 columns

Figure 106: Calculating the survival probability with a confidence interval for the male group.

## 25) Survival probability graph with a confidence interval for the male group:

```
#Plot survival function with confidence interval for male group:

confidence_surv_func = kmf_m.confidence_interval_survival_function_

plt.plot(confidence_surv_func["Male_lower_0.95"],label="Lower")
plt.plot(confidence_surv_func["Male_upper_0.95"],label="Upper")
plt.title("Survival Function With Confidence Interval")
plt.xlabel("Number of days")
plt.ylabel("Survival Probability")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc50e3b7dc8>
```

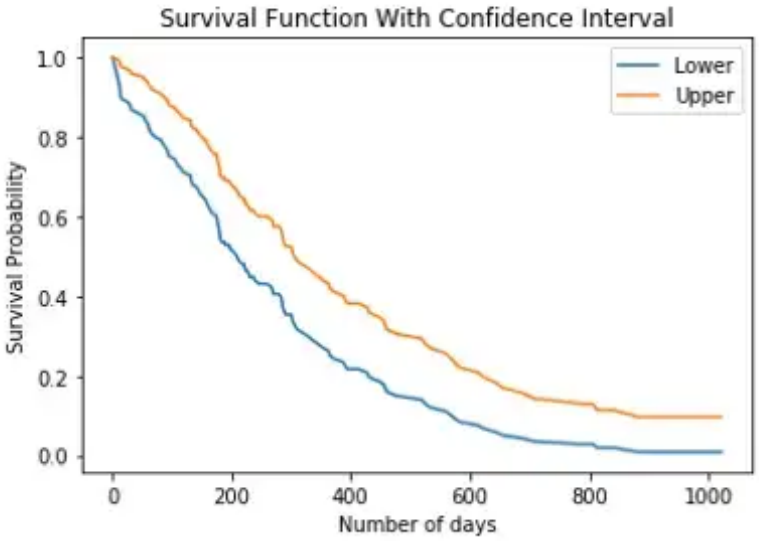


Figure 107: Confidence survival function and plot with a confidence interval for the male group.

## 26) Survival probability with a confidence interval for the female group:

```
#Survival probability with confidence interval for female group:

kmf_f.confidence_interval_survival_function_
```

| | Female_lower_0.95 | Female_upper_0.95 |
|---|---|---|
| **0.0** | 1.000000 | 1.000000 |
| **5.0** | 0.923743 | 0.998427 |
| **60.0** | 0.914060 | 0.994396 |
| **61.0** | 0.900217 | 0.989126 |
| **62.0** | 0.885909 | 0.983083 |
| **...** | ... | ... |
| **735.0** | 0.042931 | 0.252736 |
| **740.0** | 0.042931 | 0.252736 |
| **765.0** | 0.018505 | 0.212364 |
| **821.0** | 0.018505 | 0.212364 |
| **965.0** | 0.018505 | 0.212364 |

88 rows × 2 columns

Figure 108: Survival probability with a confidence interval for the female group.

## 27) Survival probability graph with a confidence interval for the female group:

```
#Plot survival function with confidence interval for female group:

confidence_surv_func = kmf_f.confidence_interval_survival_function_

plt.plot(confidence_surv_func["Female_lower_0.95"],label="Lower")
plt.plot(confidence_surv_func["Female_upper_0.95"],label="Upper")
plt.title("Survival Function With Confidence Interval")
plt.xlabel("Number of days")
plt.ylabel("Survival Probability")
plt.legend()
```

```
<matplotlib.legend.Legend at 0xc50e43ec88>
```

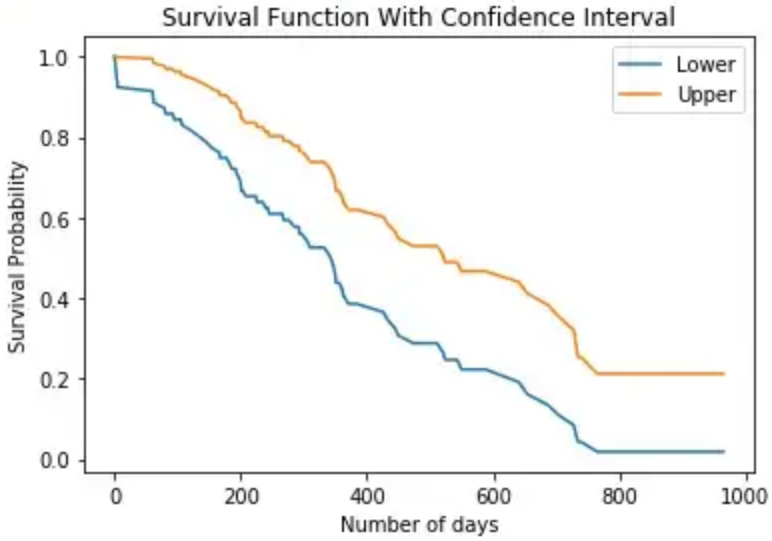


Figure 109: Plotting the survival function with a confidence interval for the female group.

## 28) Comparison of cumulative density vs. cumulative hazard:

## a) For the male group:

```
#Plot the cumulative_hazard and cumulative density:

kmf_m.plot_cumulative_density(label="Male Density")
naf_m.plot_cumulative_hazard(label="Male Hazard")
plt.xlabel("Number of Days")
```

```
Text(0.5, 0, 'Number of Days')
```

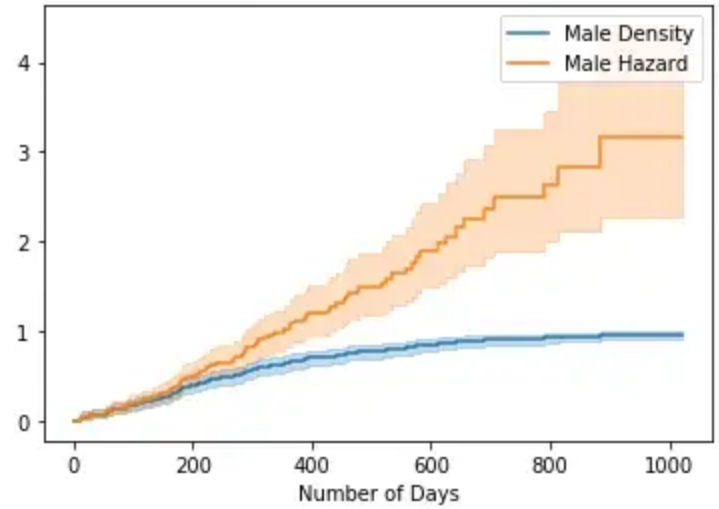


Figure 110: Plot the cumulative hazard and cumulative density.

## b) For the female group:

```
#Plot the cumulative_hazard and cumulative density:

kmf_f.plot_cumulative_density(label="Female Density")
naf_f.plot_cumulative_hazard(label="Female Hazard")
plt.xlabel("Number of Days")
```

```
Text(0.5, 0, 'Number of Days')
```

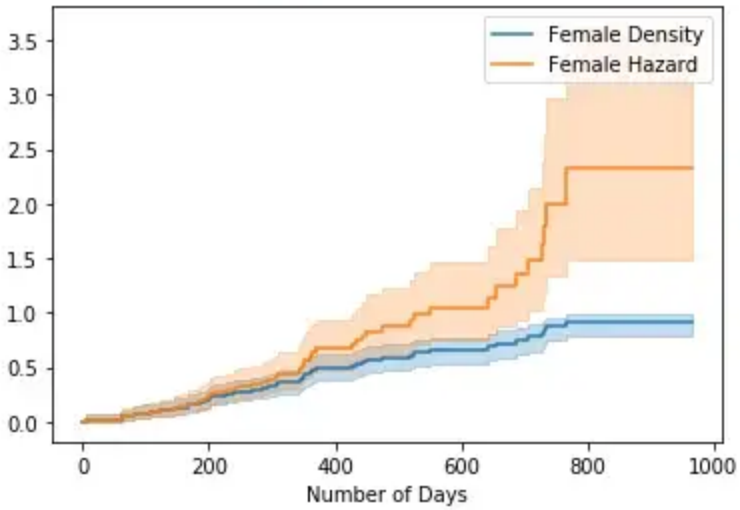


Figure 111: Plotting the cumulative hazard and cumulative density.

## 5. Log-Rank Test:

The log-rank test is a hypothesis test that is used to compare the survival distribution of two samples.

Goal: Our goal is to see if there is any significant difference between the groups being compared.

**Null Hypothesis**: The null hypothesis states that there is no significant difference between the groups being studied. If there is a significant difference between those groups, then we have to reject our null hypothesis.

**How do we say that there is a significant difference?**

A p-value between 0 and 1 denotes the statistical significance. The smaller the p-value, the more significant the statistical difference between groups being studied is. Notice that our goal is to find if there is any difference between the groups we are comparing. If yes, we can do more research on why there are lower survival chances for a particular group based on various information like their diet, lifestyle, and others.

**Less than (5% = 0.05) P-value** means there is a significant difference between the groups we compared. We can partition our groups based on their sex, age, race, treatment method, and others.

**It's a test to find out the value of P.**

**1) Get the variables for the Log-rank test:**

```
#Define variables for log-rank test:

Time_A = Male['time']
Event_A = Male['dead']

Time_B = Female['time']
Event_B = Female['dead']
```

Figure 112: Defining the variables for the log-rank test.

**2) Performing the Log-rank test:**

```
#Performing the Log-Rank test:

from lifelines.statistics import logrank_test

results = logrank_test(Time_A, Time_B, event_observed_A=Event_A, event_observed_B=Event_B)
results.print_summary()
```

| | |
|---|---|
| t_0 | -1 |
| null_distribution | chi squared |
| degrees_of_freedom | 1 |
| test_name | logrank_test |

| | test_statistic | p |
|---|---|---|
| 0 | 10.33 | <0.005 |

Figure 113: Performing the log-rank test.

### 3) Print the p-value:

```
#Print the P-value:

print("P-value :",results.p_value)
```

```
P-value : 0.0013111645203554667
```

Figure 114: Printing the p-value.

We have compared the survival distributions of two different groups using the famous statistical method, the Log-rank test. Here we can notice that the p-value is 0.00131(<0.005) for our groups, which denotes that we have to reject the null hypothesis and admit that the survival function for both groups is significantly different. The p-values give us strong evidence that “sex” was associated with the number of survival days. In short, we can say that the “sex” of a person makes a significant difference in survival probability.

## 6. Cox-proportional hazard model

The cox-proportional hazard model is a regression model generally used by medical researchers to determine the relationship between the survival time of a subject and one or more predictor variables. In short, we want to find out how different parameters like age, sex, weight, height affects the survival time of a subject.

In the previous section, we saw Kaplan-Meier, Nelson-Aalen, and Log-Rank-Test. However, in that, we were only able to consider one variable at a time, and one more thing to notice is that we were performing operations only on categorical variables like sex, status, and others. It can not be used for non-categorical data like age, weight, or height. As a solution for that, we use the Cox proportional hazards regression analysis, **which works for both quantitative predictors non-categorical variables and for categorical variables.**

#### Why do we need it?

In medical research, we are generally considering more than one factor to diagnose a person's health or survival time. i.e., we generally make use of their sex, age, blood pressure, and blood sugar to find out if there is any significant difference between those in different groups. For example, if we are grouping our data based on a person's age, our goal will be to determine which age group has a higher survival chance. Is that children's group, adult group, or old persons' group? Now what we need to find is on what basis we make a group? To find that, we use cox regression and find the coefficients of different parameters. Let's see how that works!

#### Basics of the Cox-proportional Hazard Method

The ultimate purpose of the cox-proportional hazard method is to notice how different factors in our dataset impact the event of interest.

### Hazard function:

$$h(t) = h_0(t) * \exp(b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)$$

*Where,*

$t$ = *survival time*

$h(t)$ = *the hazard function*

$x_1, x_2, \ldots, x_n$ = *covariates*

$b_1, b_2, \ldots, b_n$ = *measures the impact of covariates*

Figure 115: Hazard function formula.

The value of exp(bi) is called the Hazard Ratio (HR). We will understand this by taking an example.

```
HR = 1 : No Effects

HR < 1 : Reduction in Hazard

HR > 1 : Increase in Hazard
```

Figure 116: Defining the hazard ratio.

**Let's code:**

## 1) Import required libraries:

```
#Import required libraries:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifelines import KaplanMeierFitter
from lifelines import CoxPHFitter
```

Figure 117: Importing the required libraries with Python.

## 2) Read the CSV file:

```
#Read the data file:

data = pd.read_csv("lung.csv")
data =  data.drop(["Unnamed: 0"],axis=1)
data.head()
```

| | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3.0 | 306 | 2 | 74 | 1 | 1.0 | 90.0 | 100.0 | 1175.0 | NaN |
| 1 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 |
| 2 | 3.0 | 1010 | 1 | 56 | 1 | 0.0 | 90.0 | 90.0 | NaN | 15.0 |
| 3 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 |
| 4 | 1.0 | 883 | 2 | 60 | 1 | 0.0 | 100.0 | 90.0 | NaN | 0.0 |

Figure 118: Read the CSV file.

## 3) Delete rows that contain null values:

Next, we need to delete the rows which have null values. Our model cannot work on rows which has null values. If we do not preprocess our data, then we might get an error.

```
#Drop rows with null values:

data= data.dropna(subset=['inst', 'time', 'status', 'age', 'sex',
                          'ph.ecog','ph.karno', 'pat.karno', 'meal.cal', 'wt.loss'])
data.head()
```

| | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 |
| 3 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 |
| 5 | 12.0 | 1022 | 1 | 74 | 1 | 1.0 | 50.0 | 80.0 | 513.0 | 0.0 |
| 6 | 7.0 | 310 | 2 | 68 | 2 | 2.0 | 70.0 | 60.0 | 384.0 | 10.0 |
| 7 | 11.0 | 361 | 2 | 71 | 2 | 2.0 | 60.0 | 80.0 | 538.0 | 1.0 |

Figure 119: Drop rows with null values.

## 4) Create an object for the KapanMeierFitter:

```
#Create an object:

kmf = KaplanMeierFitter()
```

Figure 120: Create an object for the KapanMeierFitter.

## 5) Organize the data:

```
#Organize the data:

data.loc[data.status == 1, 'dead'] = 0
data.loc[data.status == 2, 'dead'] = 1
data.head()
```

| | inst | time | status | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.0 | 455 | 2 | 68 | 1 | 0.0 | 90.0 | 90.0 | 1225.0 | 15.0 | 1.0 |
| 3 | 5.0 | 210 | 2 | 57 | 1 | 1.0 | 90.0 | 60.0 | 1150.0 | 11.0 | 1.0 |
| 5 | 12.0 | 1022 | 1 | 74 | 1 | 1.0 | 50.0 | 80.0 | 513.0 | 0.0 | 0.0 |
| 6 | 7.0 | 310 | 2 | 68 | 2 | 2.0 | 70.0 | 60.0 | 384.0 | 10.0 | 1.0 |
| 7 | 11.0 | 361 | 2 | 71 | 2 | 2.0 | 60.0 | 80.0 | 538.0 | 1.0 | 1.0 |

Figure 121: Organizing the data.

## 6) Fit the data into an object:

```
#Fit data into our object:

kmf.fit(durations =  data["time"], event_observed = data["dead"])
```

```
<lifelines.KaplanMeierFitter:"KM_estimate", fitted with 167 total observations, 47 right-censored observations>
```

Figure 122: Fitting the data into an object.

## 7) Generating the event table:

```
#Get the event table:

kmf.event_table
```

| event_at | removed | observed | censored | entrance | at_risk |
|---|---|---|---|---|---|
| 0.0 | 0 | 0 | 0 | 167 | 167 |
| 5.0 | 1 | 1 | 0 | 0 | 167 |
| 11.0 | 1 | 1 | 0 | 0 | 166 |
| 12.0 | 1 | 1 | 0 | 0 | 165 |
| 13.0 | 1 | 1 | 0 | 0 | 164 |
| ... | ... | ... | ... | ... | ... |
| 814.0 | 1 | 1 | 0 | 0 | 5 |
| 821.0 | 1 | 0 | 1 | 0 | 4 |
| 840.0 | 1 | 0 | 1 | 0 | 3 |
| 965.0 | 1 | 0 | 1 | 0 | 2 |
| 1022.0 | 1 | 0 | 1 | 0 | 1 |

150 rows × 5 columns

Figure 123: Generating the event table.

## 8) Get the required columns:

```
#Get required columns from the data:

data = data[[ 'time', 'age', 'sex', 'ph.ecog','ph.karno',
              'pat.karno', 'meal.cal', 'wt.loss', 'dead']]
```

Figure 124: Get the required columns from the data.

## 9) Fit the data and print the summary:

```
#Get the summary using CoxPHFitter:

cph = CoxPHFitter()
cph.fit(data,"time",event_col="dead")
cph.print_summary()
```

| | |
|---|---|
| **model** | lifelines.CoxPHFitter |
| **duration col** | 'time' |
| **event col** | 'dead' |
| **baseline estimation** | breslow |
| **number of observations** | 167 |
| **number of events observed** | 120 |
| **partial log-likelihood** | -494.03 |
| **time fit was run** | 2020-09-07 13:39:24 UTC |

Figure 125: Get the summary using CoxPHFitter.

| | coef | exp(coef) | se(coef) | coef lower 95% | coef upper 95% | exp(coef) lower 95% | exp(coef) upper 95% | z | p | -log2(p) |
|---|---|---|---|---|---|---|---|---|---|---|
| **age** | 0.01 | 1.01 | 0.01 | -0.01 | 0.03 | 0.99 | 1.03 | 0.93 | 0.35 | 1.51 |
| **sex** | -0.55 | 0.57 | 0.20 | -0.95 | -0.16 | 0.39 | 0.85 | -2.75 | 0.01 | 7.37 |
| **ph.ecog** | 0.74 | 2.09 | 0.22 | 0.30 | 1.18 | 1.35 | 3.26 | 3.29 | <0.005 | 9.95 |
| **ph.karno** | 0.02 | 1.02 | 0.01 | 0.00 | 0.04 | 1.00 | 1.05 | 2.00 | 0.05 | 4.45 |
| **pat.karno** | -0.01 | 0.99 | 0.01 | -0.03 | 0.00 | 0.97 | 1.00 | -1.49 | 0.14 | 2.87 |
| **meal.cal** | 0.00 | 1.00 | 0.00 | -0.00 | 0.00 | 1.00 | 1.00 | 0.11 | 0.91 | 0.13 |
| **wt.loss** | -0.01 | 0.99 | 0.01 | -0.03 | 0.00 | 0.97 | 1.00 | -1.83 | 0.07 | 3.89 |

| | |
|---|---|
| **Concordance** | 0.65 |
| **Partial AIC** | 1002.07 |
| **log-likelihood ratio test** | 28.16 on 7 df |
| **-log2(p) of ll-ratio test** | 12.25 |

Figure 126: Displaying the data.

In the picture above, notice the p-value for each column in our dataset. Next, we know that p-value<0.05 is considered statistically significant. Here we can see that “sex” and “ph.ecog” have p-values less than 0.05. So we can say that while grouping our data for analysis, we should focus on dividing the data based on these two factors.

$$Hazard\ Ratio\ (HR) = \exp(b_i)$$

Figure 127: Hazard Ratio formula.

```
HR = 1 : No Effects

HR < 1 : Reduction in Hazard

HR > 1 : Increase in Hazard
```

Figure 128: Hazard Ratio values.

Here notice the p-value for “sex” is 0.01, and the Hazard Ratio(HR) is 0.57, which indicates a strong relationship between the patients’ sex and decreased risk of death. For example, holding the other covariates constant, **being female (sex=2) reduces the hazard by a factor of 0.57, or 43%.** That means that females have higher survival chances. Next, the p-value for ph.ecog is <0.005, and the Hazard Ratio(HR) is 2.09, which indicates a strong relationship between the ph.ecog value and the increased risk of death. Holding the other covariates constant, a higher value of ph.ecog is associated with poor survival. **Here person with higher ph.ecog value has a 109% higher risk of death.** So, in short, we can say that doctors should try to reduce the value of ph.ecog in patients by providing relevant medicines. Next, notice that the Hazard Ratio(HR) for Age is 1.01, suggesting only a 1% increase for a higher age group. So we can say that there is no significant difference between different age groups.

### 10) Check which factor affects the most from the graph:

In the following graph, we can notice the difference in “sex” and “ph.ecog” data.

```
#Plot the result on graph:

cph.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9a934c3c88>
```

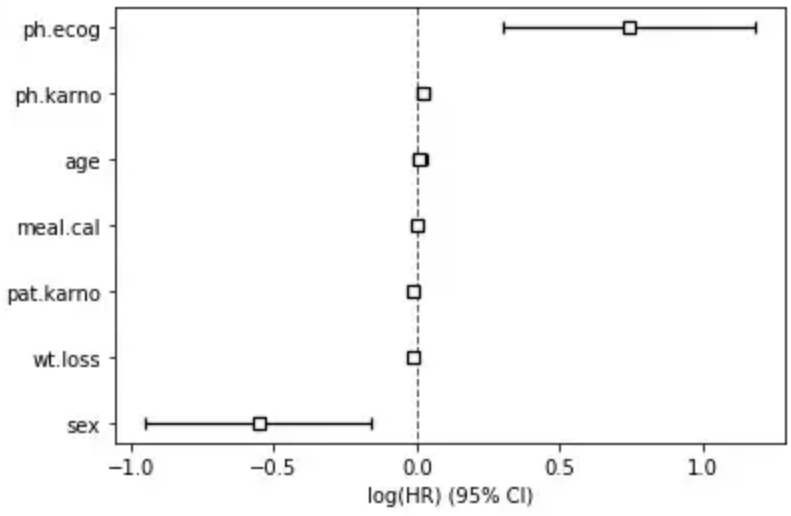

Figure 129: Plot the result on a graph.

## 11) Check our theory with real observations:

Let's check our conclusions with real data from our observations.

```
data.iloc[10:15,:]
```

| | time | age | sex | ph.ecog | ph.karno | pat.karno | meal.cal | wt.loss | dead |
|---|---|---|---|---|---|---|---|---|---|
| **17** | 707 | 63 | 1 | 2.0 | 50.0 | 70.0 | 1025.0 | 22.0 | 1.0 |
| **18** | 61 | 56 | 2 | 2.0 | 60.0 | 60.0 | 238.0 | 10.0 | 1.0 |
| **20** | 301 | 67 | 1 | 1.0 | 80.0 | 80.0 | 1025.0 | 17.0 | 1.0 |
| **21** | 81 | 49 | 2 | 0.0 | 100.0 | 70.0 | 1175.0 | -8.0 | 1.0 |
| **23** | 371 | 58 | 1 | 0.0 | 90.0 | 100.0 | 975.0 | 13.0 | 1.0 |

Figure 130: Conclusion table.

```
#Plotting the data:

d_data = data.iloc[10:15,:]
cph.predict_survival_function(d_data).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9a935ffb08>
```

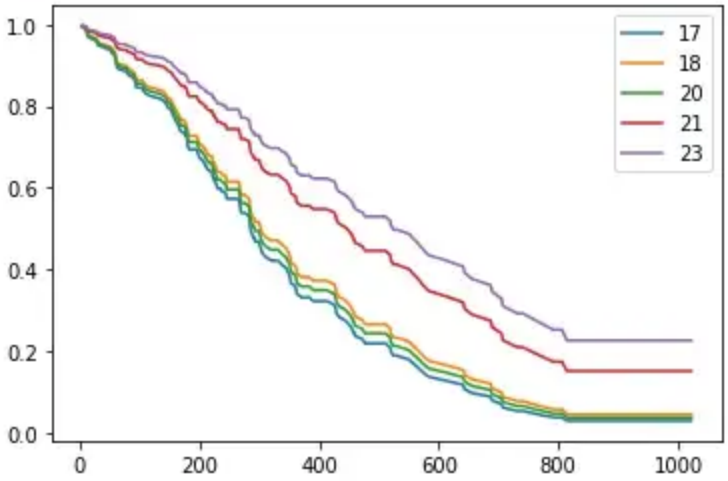


Figure 131: Plotting our data.

In the above graph, we can see that person 23 has the highest chance of survival, while person 17 has the least chance of survival. By checking the main table, we can notice a significant change in the ph.ecog value. We can also see that people 21 and 23 have higher chances of survival as they have the least value of ph.ecog.

Okay, so this is it for this tutorial. Thank you for reading. Your feedback is always welcome.

This is a revamped version of the original article published on **KDNuggets**.

> 📚 If you enjoyed this piece, check out our tutorial on neural networks from scratch with Python code and math in detail.📚


Buy Pratik a Coffee!

**DISCLAIMER:** The views expressed in this article are those of the author(s) and do not represent the views of Carnegie Mellon University. These writings do not intend to be final products, yet rather a reflection of current thinking, along with being a catalyst for discussion and improvement.

Published via Towards AI

**Citation**

For attribution in academic contexts, please cite this work as:

```
Shukla, et al., “Survival Analysis with Python Tutorial – How, What, When, and
Why”, Towards AI, 2020
```

**BibTex citation:**

```
@article{pratik_iriondo_2020,
 title={Survival Analysis with Python Tutorial – How, What, When, and Why},
 url={https://towardsai.net/survival-analysis-with-python},
 journal={Towards AI},
 publisher={Towards AI Co.},
 author={Pratik, Shukla},
```

```
  editor={Iriondo, Roberto},
  year={2020},
  month={Sep}
}
```

## 7. Resources

Google Colab Implementation

Github Repository

## 8. References:


[1] Lifelines Example, https://lifelines.readthedocs.io/en/latest/Examples.html

[2] Kaplan — Meier Estimator,
Wikipedia, https://en.wikipedia.org/wiki/Kaplan%E2%80%93Meier_estimator

[3] Lifelines, Univariate
NelsonAalenFilter, https://lifelines.readthedocs.io/en/latest/fitters/univariate/NelsonAalenFitter.html

[4] STHDA, Statistical Tools for High-throughput Data
Analysis, http://www.sthda.com/english/wiki/cox-proportional-hazards-model