PANDAS(Python)

Nageswara Rao Gandham
Syneos Health

Indian Association for Statistics in Clinical Trials

Introduction

As we all know data analytics are playing vital role in all industries there are lot of analytical tools available in the market. Out of those I would like to present on the PANDAS which stands for "**Python** Data Analysis Library".

What is PANDAS? Productio and Distributd Analysis System

PANDAS is a **Python** package providing fast, flexible, and expressive data structures designed to make working with relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. It is built on top of two core Python libraries— matplotlib for data visualization and NumPy for mathematical operations.

Data Frames and Series

While series are useful, most analysts work with the majority of their data in Data Frames. Data Frames store data in the familiar table format of rows and columns, much like a spreadsheet or dataset. Data Frames makes a lot of analytical tasks easier, such as **finding the averages per column** in a dataset. You can also think of Data Frames as a collection of series.

Handling Missing data with Pandas

Missing data is common in most data analysis applications. I find drop na and fill na function very useful while handling missing data.

Ex: df.fillna(0)

Summary statistics and Data Visualization

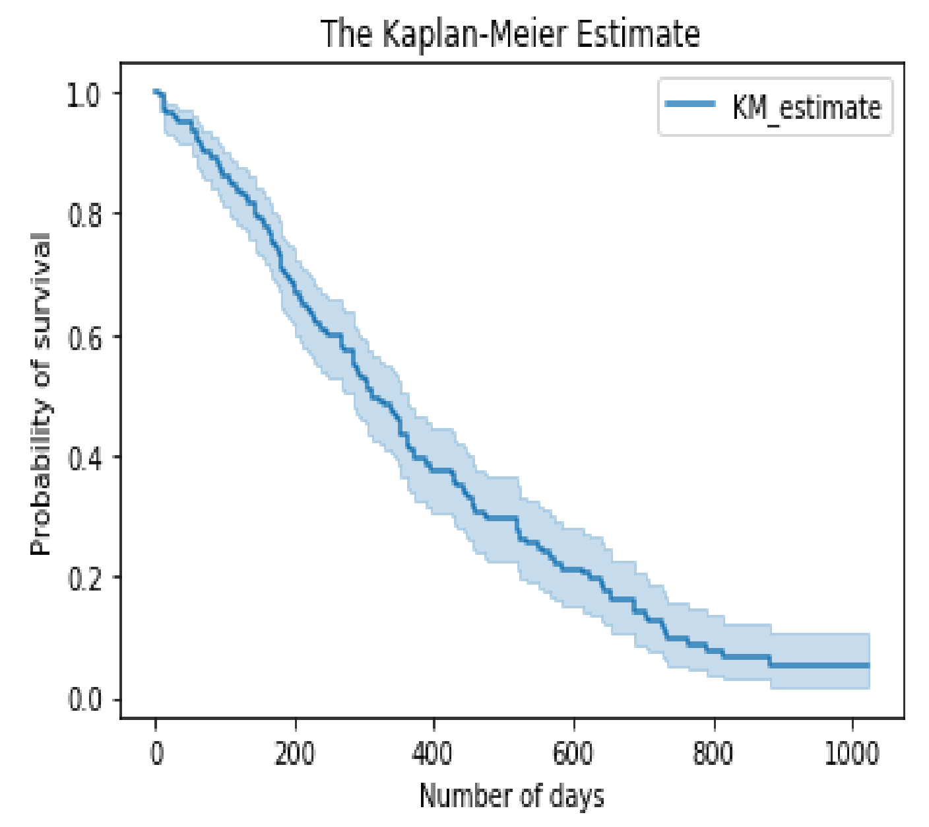
Creating Summary statistics and graphs are very easy with Pandas. Visualization also we have multiple packages depend on our requirement we can use them.

Ex: df.["Score"].describe, df.["Score"].mean

```
#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')
```

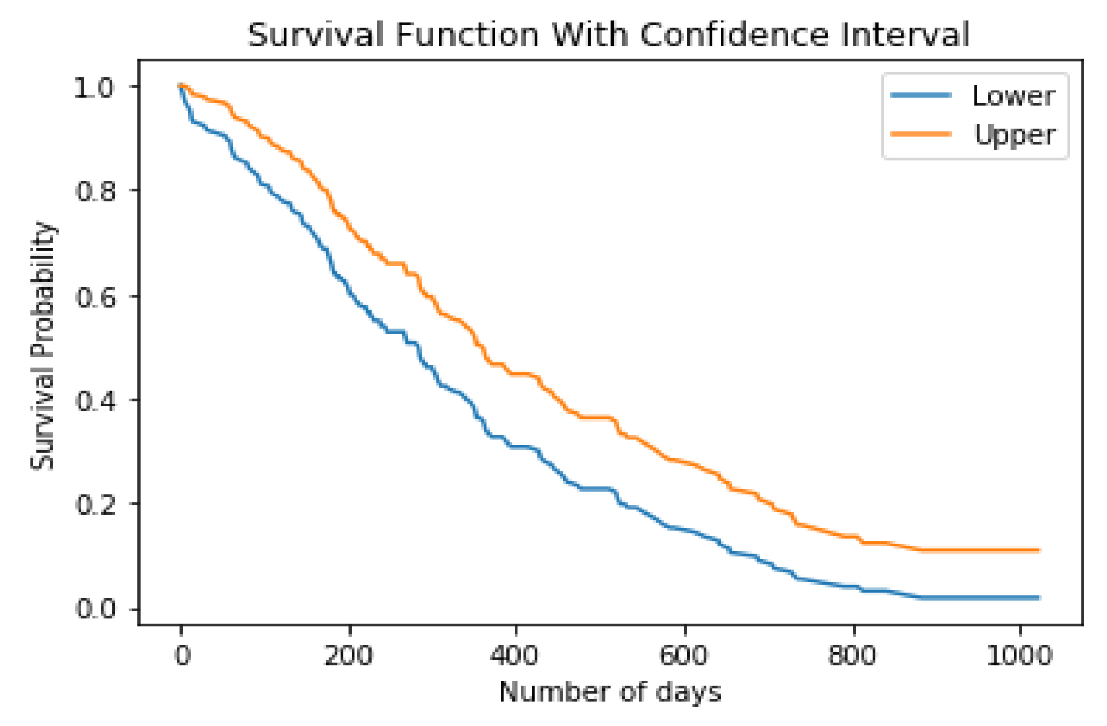


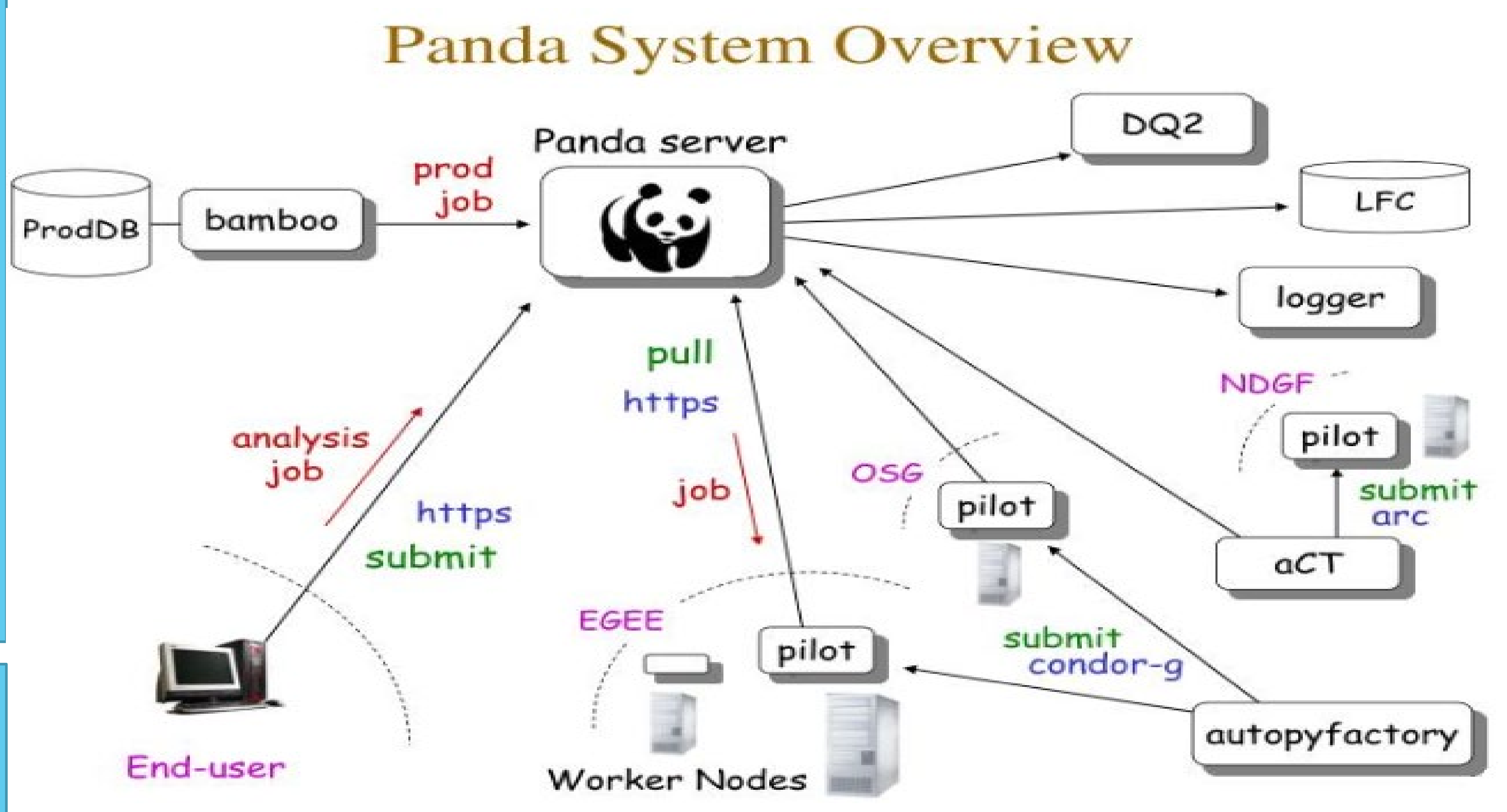
```
#Plot survival function with confidence interval:

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>
```





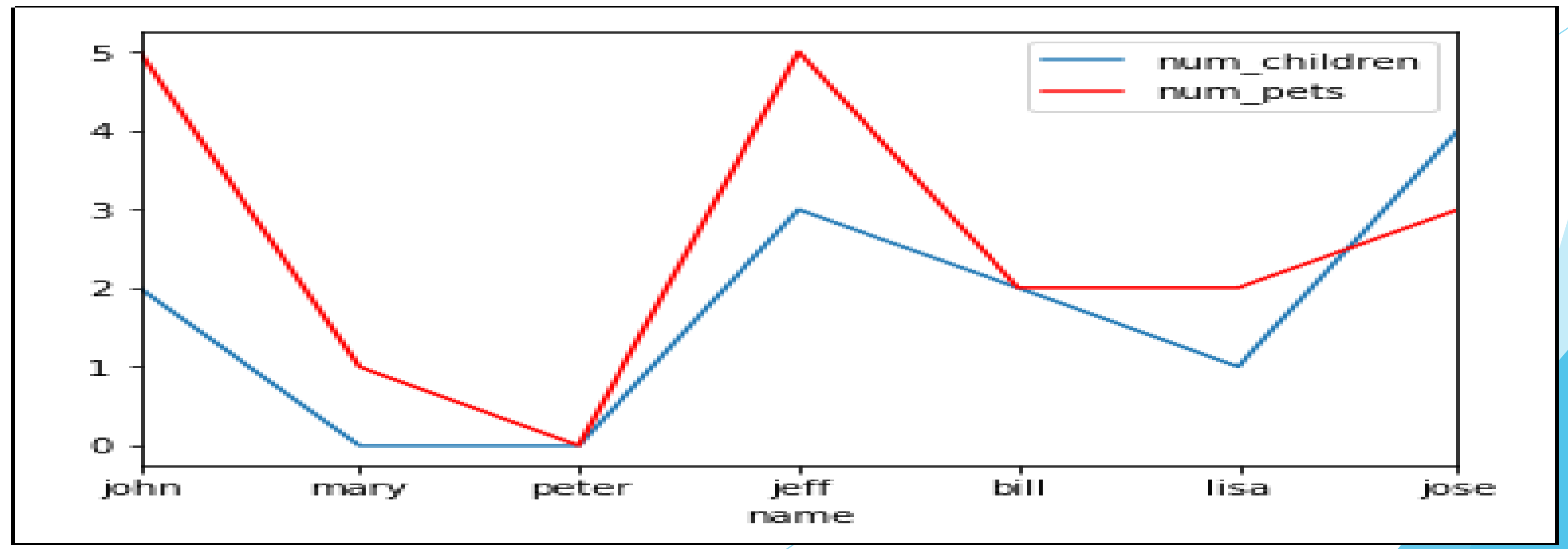
```
df_no_missing = pd.data.dropna()
df_no_missing
```

```
import matplotlib.pyplot as plt
import pandas as pd

# gca stands for 'get current axis'
ax = plt.gca()

df.plot(kind='line',x='name',y='num_children',ax=ax)
df.plot(kind='line',x='name',y='num_pets', color='red', ax=ax)

plt.show()
```



Loading data to Python environment

We need to import modules to load the data set into Python environment.

```
import pandas as pd
import numpy as np
data = pd.read_csv("train.csv")
```

Filtering data by using Pandas

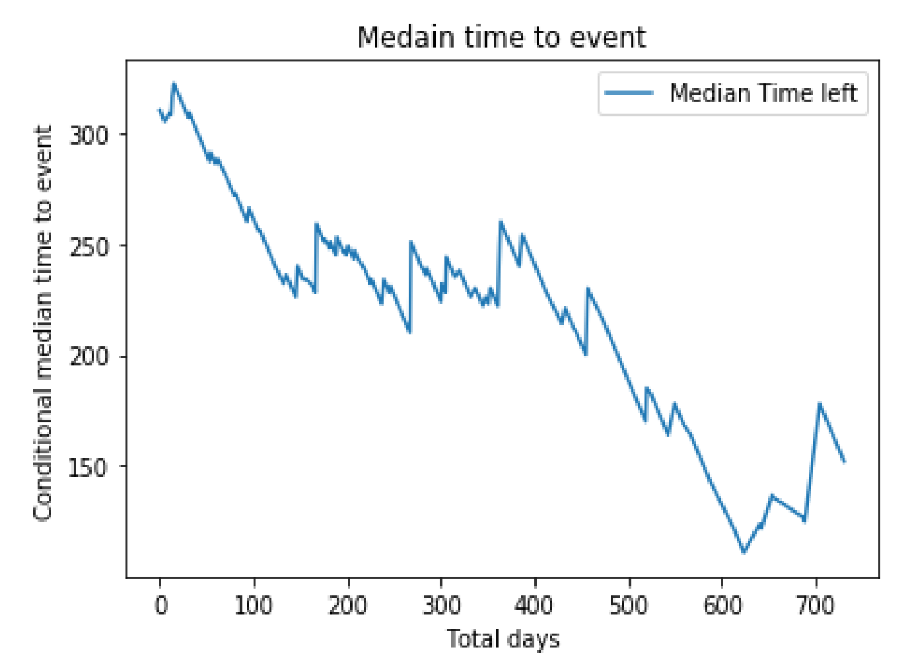
We can filter the data in Pandas very easily.

df_filtered = df [df . Score >=2]
print(df_filtered)

```
#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>
```



Conclusion

- When compared to other tools very less code and any one with basic knowledge can work on it.
- Visualization is very easy and within fraction of seconds, you can visualize the data with the use of Matplotlib.
- You can take advantage of the data analysis features of Pandas to create custom big data solutions without putting extra time and effort.
- It is an open source and effective tool for data analysis.