In [288…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from lifelines import KaplanMeierFitter, WeibullFitter
from scipy.stats import gamma
from scipy.stats import norm
from scipy.stats import chi2
```

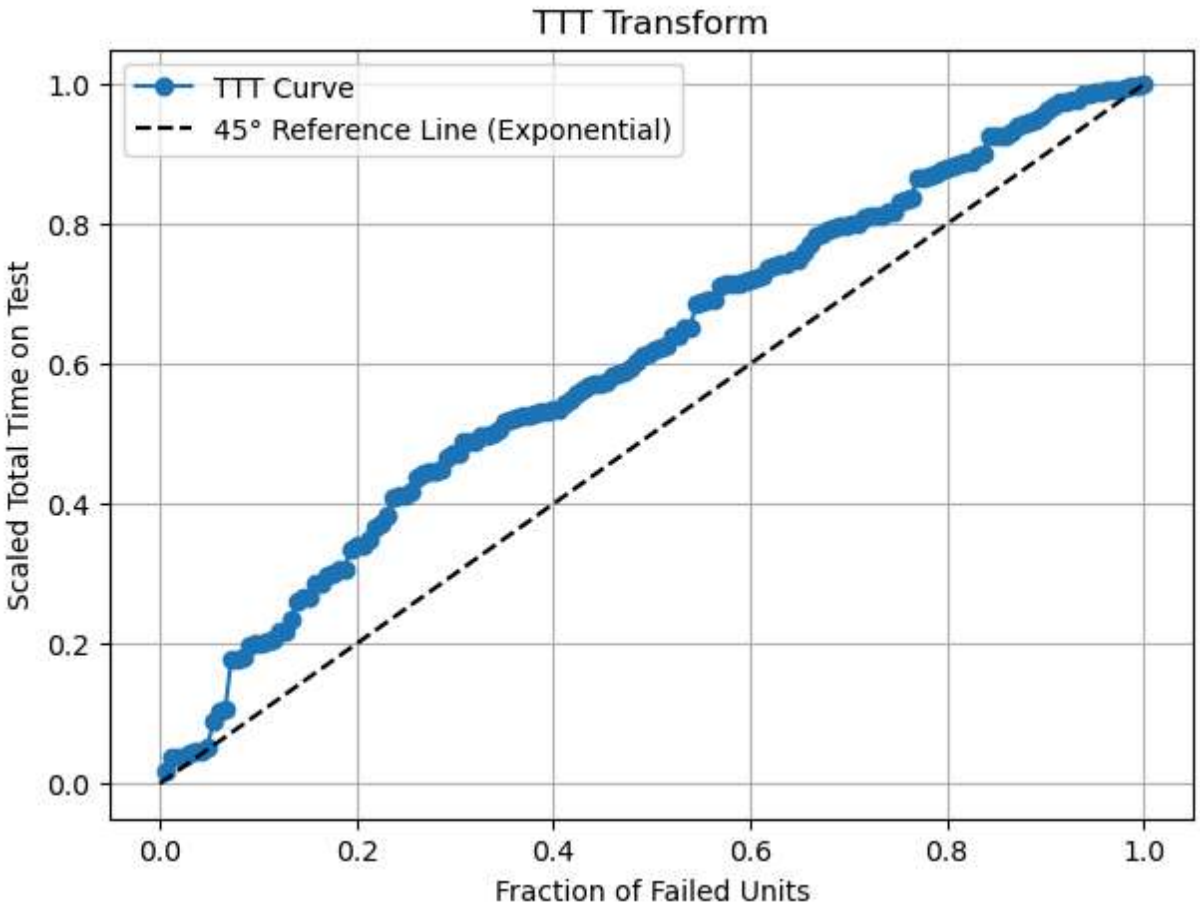
In [289…
```
df = pd.read_csv("C:/Users/singh/Documents/lung_cancer.csv")
```

In [290…
```
# Map categories to digits
mapping = {'M': 0, 'F': 1}
df['sex'].replace(mapping, inplace=True)
```

In [291…
```
# Filter for uncensored data
uncensored_times = df[df['Y'] == 1]['TIME'].sort_values().values
n = len(uncensored_times)
```

In [292…
```
# Compute cumulative TTT values
S = np.zeros(n)
for i in range(n):
    S[i] = np.sum(uncensored_times[:i+1]) + (n - i - 1) * uncensored_times[i]

# Normalize for plotting
x_ttt = np.arange(1, n + 1) / n
y_ttt = S / S[-1]
```

In [293…
```
plt.figure(figsize=(7, 5))
plt.plot(x_ttt, y_ttt, marker='o', label='TTT Curve')
plt.plot([0, 1], [0, 1], 'k--', label='45° Reference Line (Exponential)')
plt.title("TTT Transform")
plt.xlabel("Fraction of Failed Units")
plt.ylabel("Scaled Total Time on Test")
plt.legend()
plt.grid(True)
plt.show()
```



- the given datset is Increasing Failure Rate type. hence we wil use those parametric models that follows IFR.
  - Weibull Distribution
  - Normal Distribution
  - Gamma Distribution

Based on which we will do reliability assesment of the models in parametric method based on our datasets

In [295…
```
# Rename if necessary
T = df['TIME']
E = df['Y']
```

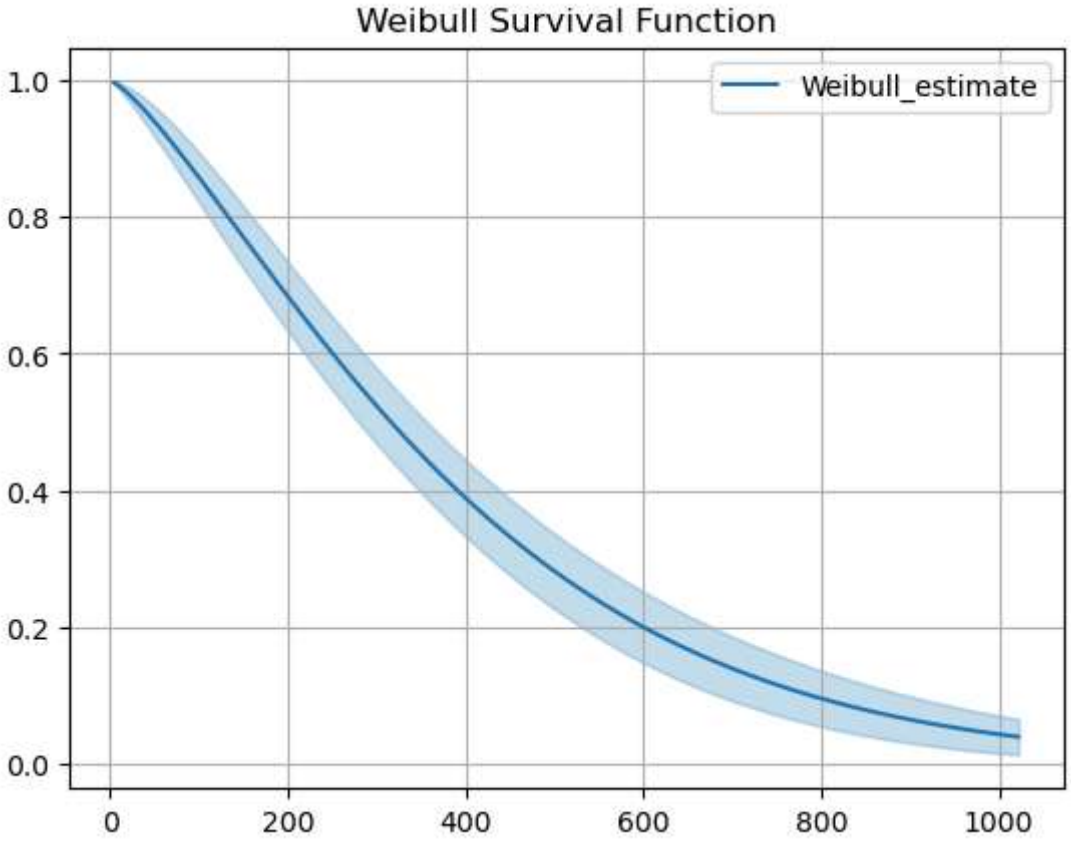
WEIBULL MODEL FOR PARAMETRIC METHOD

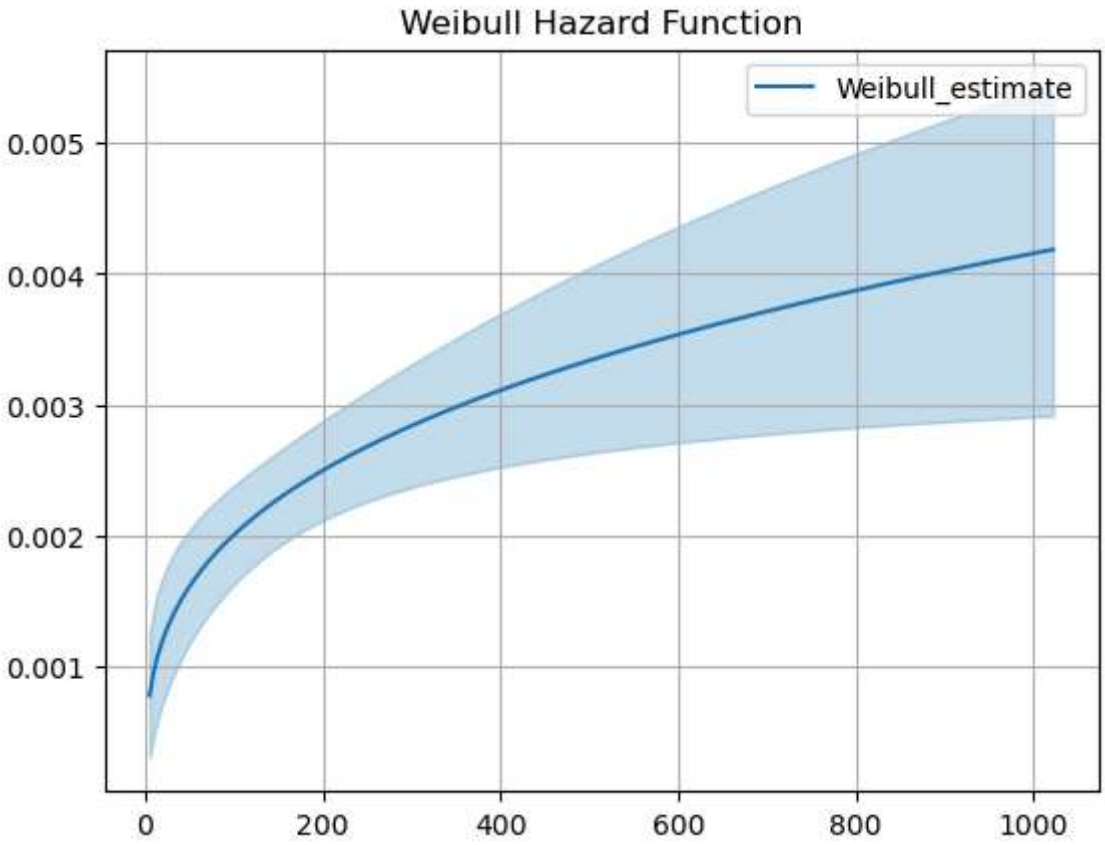
(two parameters)

In [297…
```
# Weibull Distribution Model for Parametric Method
wf = WeibullFitter()
wf.fit(T, event_observed=E)

print("Weibull Parameters:")
print("Lambda (scale):", wf.lambda_)
print("Rho (shape):", wf.rho_)
```

```
Weibull Parameters:
Lambda (scale): 417.7586472606239
Rho (shape): 1.3168399982896235
```

In [298…
```
wf.plot_survival_function()
plt.title("Weibull Survival Function")
plt.grid(True)
plt.show()
wf.plot_hazard()
plt.title("Weibull Hazard Function")
plt.grid(True)
plt.show()
```





In [299…

```
# mean time to failure MTTF
from scipy.special import gamma as gamma_func

lambda_w = wf.lambda_
rho_w = wf.rho_

mttf_weibull = lambda_w * gamma_func(1 + 1 / rho_w)
print(f"Weibull MTTF: {mttf_weibull:.2f}")
print(f"Weibull AIC: {wf.AIC_:.2f}")
print(f"Weibull BIC: {wf.BIC_:.2f}")
```

```
Weibull MTTF: 384.85
Weibull AIC: 2311.70
Weibull BIC: 2318.56
```

NORMAL MODEL FOR PARAMETRIC MODEL

In [301…

```
# Estimate parameters: mean and std of uncensored data
mu, sigma = norm.fit(T)

# Define survival and hazard functions for Normal distribution
def survival_function(T):
    return 1 - norm.cdf(T, mu, sigma)

def hazard_function(T):
    pdf = norm.pdf(T, mu, sigma)
    sf = survival_function(T)
    return np.where(sf > 0, pdf / sf, 0)
    # Mean Time To Failure (MTTF) for normal = mu
MTTF = mu
```

```
# Log-likelihood for AIC and BIC (include both censored and uncensored data)
log_likelihood = np.sum(E * norm.logpdf(T, mu, sigma) +
                        (1 - E) * np.log(survival_function(T)))
k = 2  # Number of parameters: mu and sigma
n = len(T)
AIC = -2 * log_likelihood + 2 * k
BIC = -2 * log_likelihood + k * np.log(n)
```

In [302…

```
# Output results
print("Normal Distribution Model")
print(f"Estimated μ (mean): {mu:.4f}")
print(f"Estimated σ (std dev): {sigma:.4f}")
print(f"MTTF: {MTTF:.4f}")
print(f"AIC: {AIC:.4f}")
print(f"BIC: {BIC:.4f}")
```

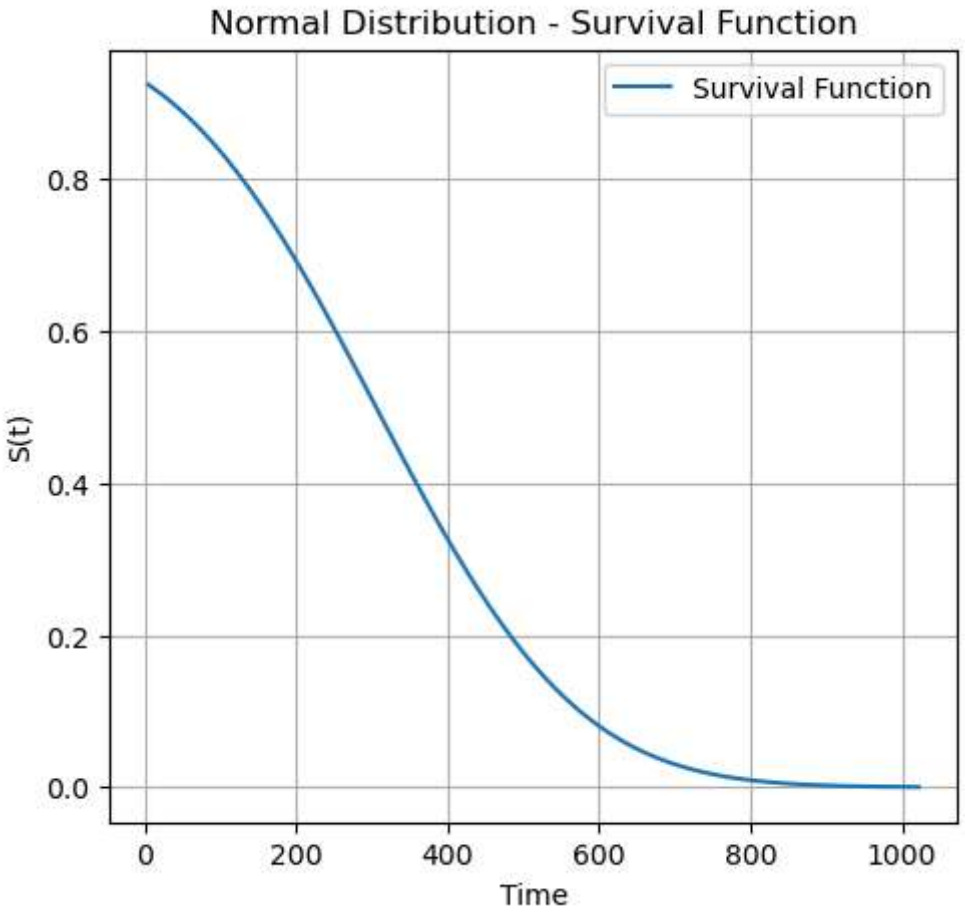
```
Normal Distribution Model
Estimated μ (mean): 305.2325
Estimated σ (std dev): 210.1831
MTTF: 305.2325
AIC: 2396.8012
BIC: 2403.6599
```

In [303…

```
# Optional: plot survival and hazard functions
t_vals = np.linspace(min(T), max(T), 100)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(t_vals, survival_function(t_vals), label='Survival Function')
plt.title('Normal Distribution - Survival Function')
plt.xlabel('Time')
plt.ylabel('S(t)')
plt.grid(True)
plt.legend()
```

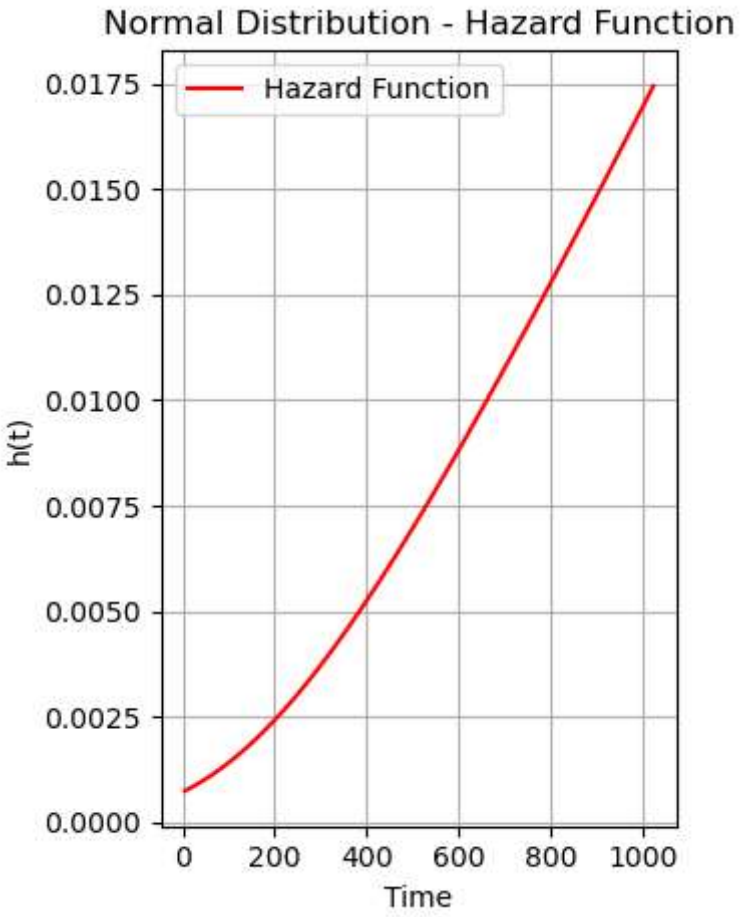
Out[303…

```
<matplotlib.legend.Legend at 0x1be72523710>
```



In [304…

```
plt.subplot(1, 2, 2)
plt.plot(t_vals, hazard_function(t_vals), label='Hazard Function', color='r')
plt.title('Normal Distribution - Hazard Function')
plt.xlabel('Time')
plt.ylabel('h(t)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



GAMMA MODEL FOR PARAMETRIC METHOD

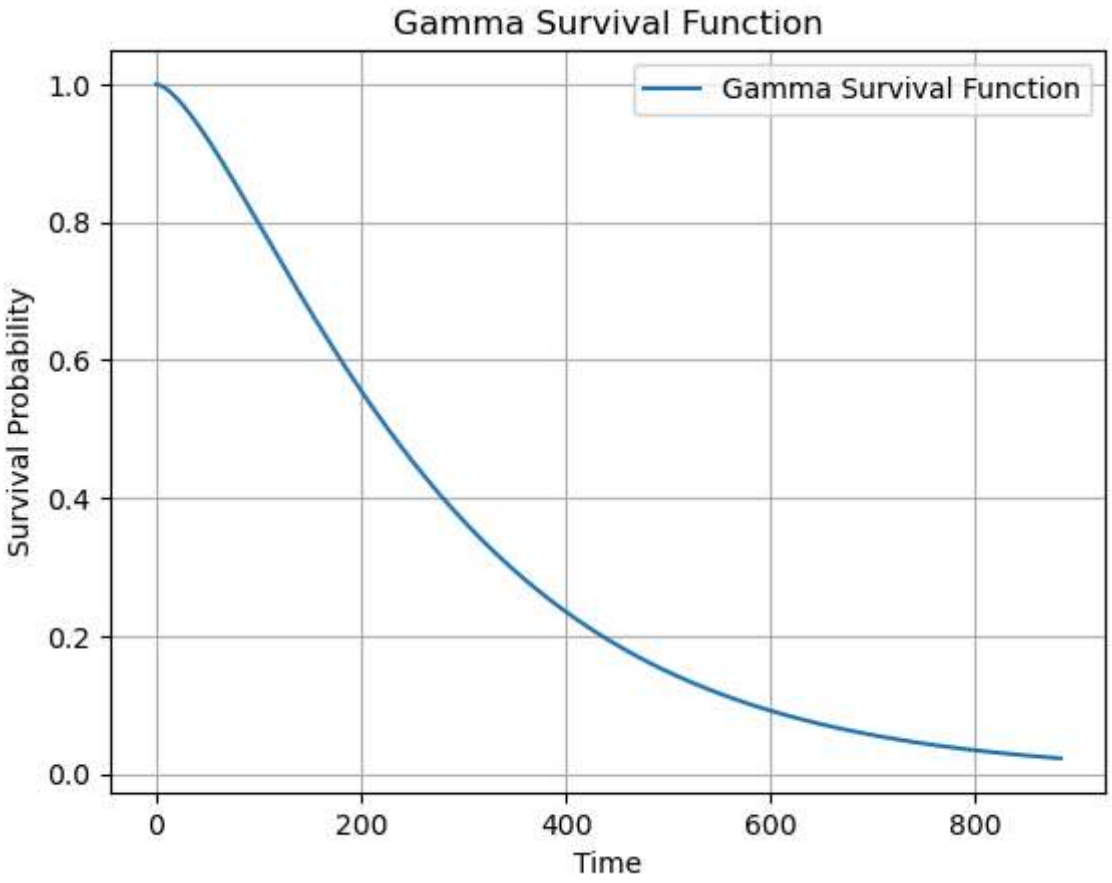
In [306…

```
from scipy.stats import gamma

# Filter uncensored data (event == 1)
uncensored_T = df[df['Y'] == 1]['TIME'].values

# Fit gamma distribution
a_g, loc_g, scale_g = gamma.fit(uncensored_T, floc=0)

print(f"Gamma Parameters:\nShape (a): {a_g:.4f}, Scale: {scale_g:.4f}")
```

```
Gamma Parameters:
Shape (a): 1.5867, Scale: 178.3596
```

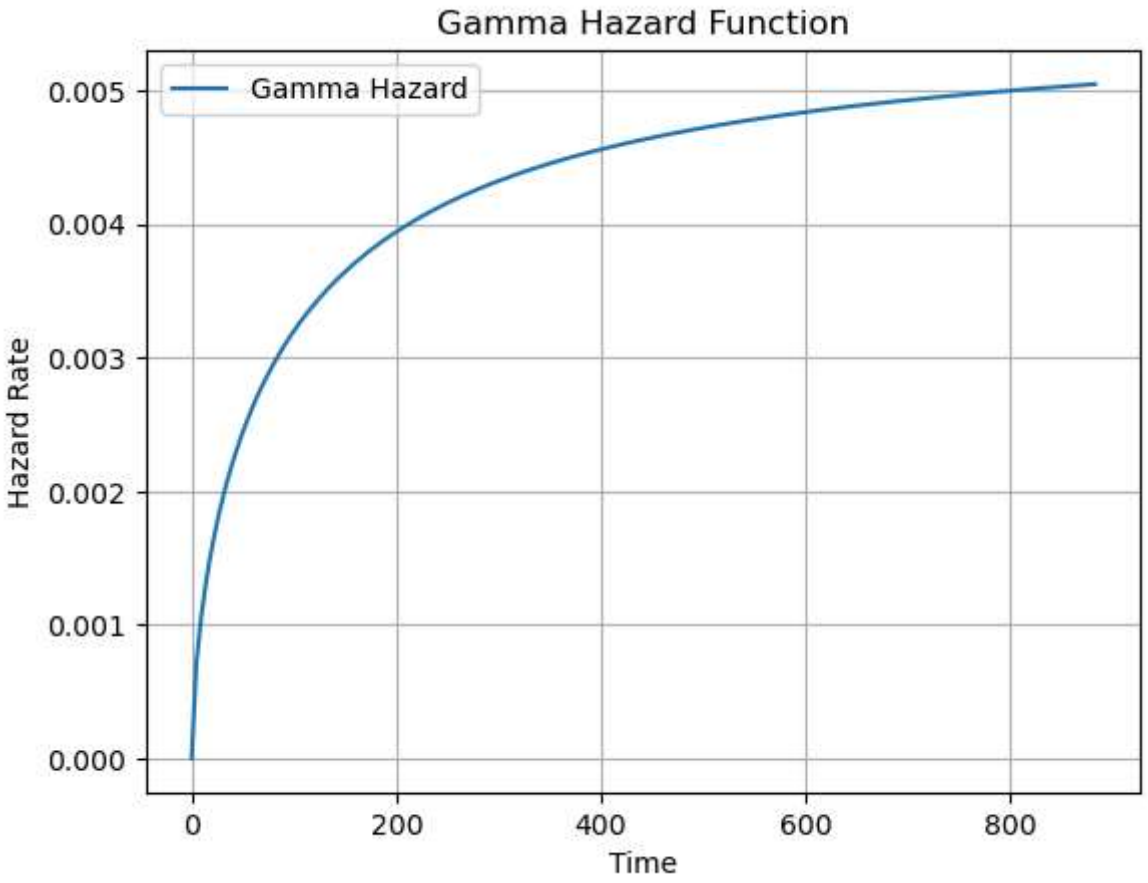
In [307…

```
x_vals = np.linspace(0, uncensored_times.max(), 200)
gamma_surv = gamma.sf(x_vals, a_g, loc=loc_g, scale=scale_g)

plt.plot(x_vals, gamma_surv, label="Gamma Survival Function")
plt.title("Gamma Survival Function")
plt.xlabel("Time")
plt.ylabel("Survival Probability")
plt.grid(True)
plt.legend()
plt.show()
```



In [308…

```
# Estimate hazard h(t) = f(t) / S(t)
pdf_vals = gamma.pdf(x_vals, a_g, loc=loc_g, scale=scale_g)
hazard_gamma = pdf_vals / gamma_surv

plt.plot(x_vals, hazard_gamma, label="Gamma Hazard")
plt.title("Gamma Hazard Function")
plt.xlabel("Time")
plt.ylabel("Hazard Rate")
plt.grid(True)
plt.legend()
plt.show()
```



In [309…
```
mttf_gamma = a_g * scale_g
print(f"Gamma MTTF: {mttf_gamma:.2f}")
```

```
Gamma MTTF: 283.00
```

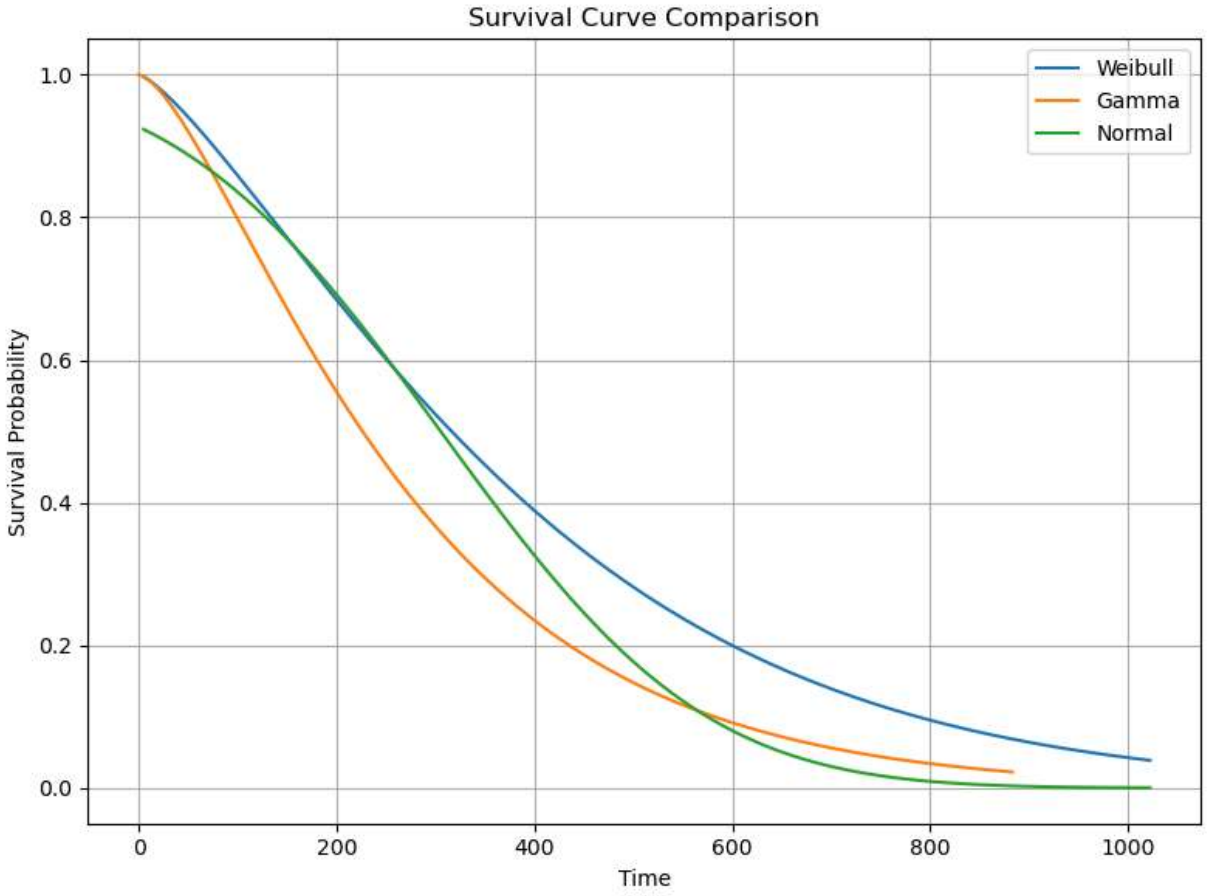
In [310…
```
log_likelihood_gamma = np.sum(gamma.logpdf(uncensored_times, a_g, loc=loc_g,
scale=scale_g))
k_gamma = 2  # shape and scale
n_gamma = len(uncensored_times)

aic_gamma = 2 * k_gamma - 2 * log_likelihood_gamma
bic_gamma = k_gamma * np.log(n_gamma) - 2 * log_likelihood_gamma

print(f"Gamma Log-Likelihood: {log_likelihood_gamma:.2f}")
print(f"Gamma AIC: {aic_gamma:.2f}")
print(f"Gamma BIC: {bic_gamma:.2f}")
```

```
Gamma Log-Likelihood: -1087.20
Gamma AIC: 2178.40
Gamma BIC: 2184.61
```

In [311…
```
# --- Weibull ---
weibull_dict = {
    'Model': 'Weibull',
    'AIC': wf.AIC_,
    'BIC': wf.BIC_,
    'Log-Likelihood': -wf.log_likelihood_,
    'Median Survival Time': wf.median_survival_time_,
    'MTTF': mttf_weibull
}
```

```
# --- Normal ---
normal_dict = {
    'Model': 'Normal',
    'AIC': AIC,
    'BIC': BIC,
    'Log-Likelihood': -log_likelihood,
    'MTTF': MTTF
}

# --- Gamma ---
gamma_dict = {
    'Model': 'Gamma',
    'AIC': aic_gamma,
    'BIC': bic_gamma,
    'Log-Likelihood': -log_likelihood_gamma,
    'Median Survival Time': np.median(uncensored_T),
    'MTTF': mttf_gamma
}
# Combine into a DataFrame
summary_df = pd.DataFrame([weibull_dict, gamma_dict, normal_dict])
summary_df = summary_df[['Model', 'AIC', 'BIC', 'Log-Likelihood', 'Median Survival Time',
'MTTF']]

# Round values
summary_df = summary_df.round(3)

# Show summary
print("📊 Summary Comparison of All Models:")
print(summary_df)

plt.figure(figsize=(8, 6))

# Weibull
wf.plot_survival_function(label="Weibull", ci_show=False)

# Gamma (manual)
plt.plot(x_vals, gamma_surv, label="Gamma")

# Normal
plt.plot(t_vals, survival_function(t_vals), label='Normal')

plt.title("Survival Curve Comparison")
plt.xlabel("Time")
plt.ylabel("Survival Probability")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
📊 Summary Comparison of All Models:
     Model       AIC       BIC  Log-Likelihood  Median Survival Time     MTTF
0  Weibull  2311.702  2318.561        1153.851               316.264  384.853
1    Gamma  2178.398  2184.610        1087.199               226.000  283.000
2   Normal  2396.801  2403.660        1196.401                   NaN  305.232
```



INTERPRETATION

- The Gamma distribution, while having the lowest AIC, is not performing well on the survival curves compared Weibull. This could indicate that the Gamma distribution's assumptions about the data are not well-suited for this specific dataset.

KAPLAN MEIER ESTIMATION METHOD OF SURVIVAL FUNCTION UNDER NON-PARAMETRIC METHOD

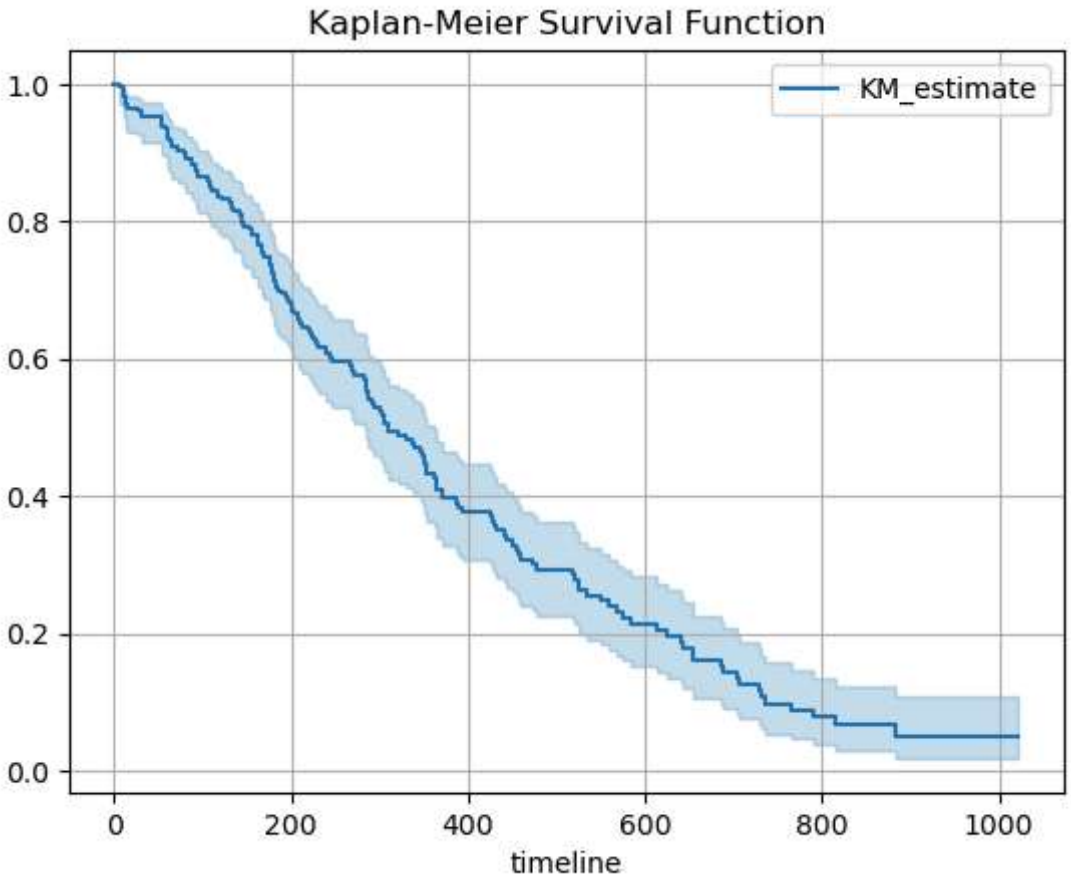
In [314…

```
from lifelines import KaplanMeierFitter

T = df['TIME']
E = df['Y']

kmf = KaplanMeierFitter()
kmf.fit(T, event_observed=E)

print(f"Median Survival Time (KM): {kmf.median_survival_time_:.2f}")
```

```
Median Survival Time (KM): 310.00
```

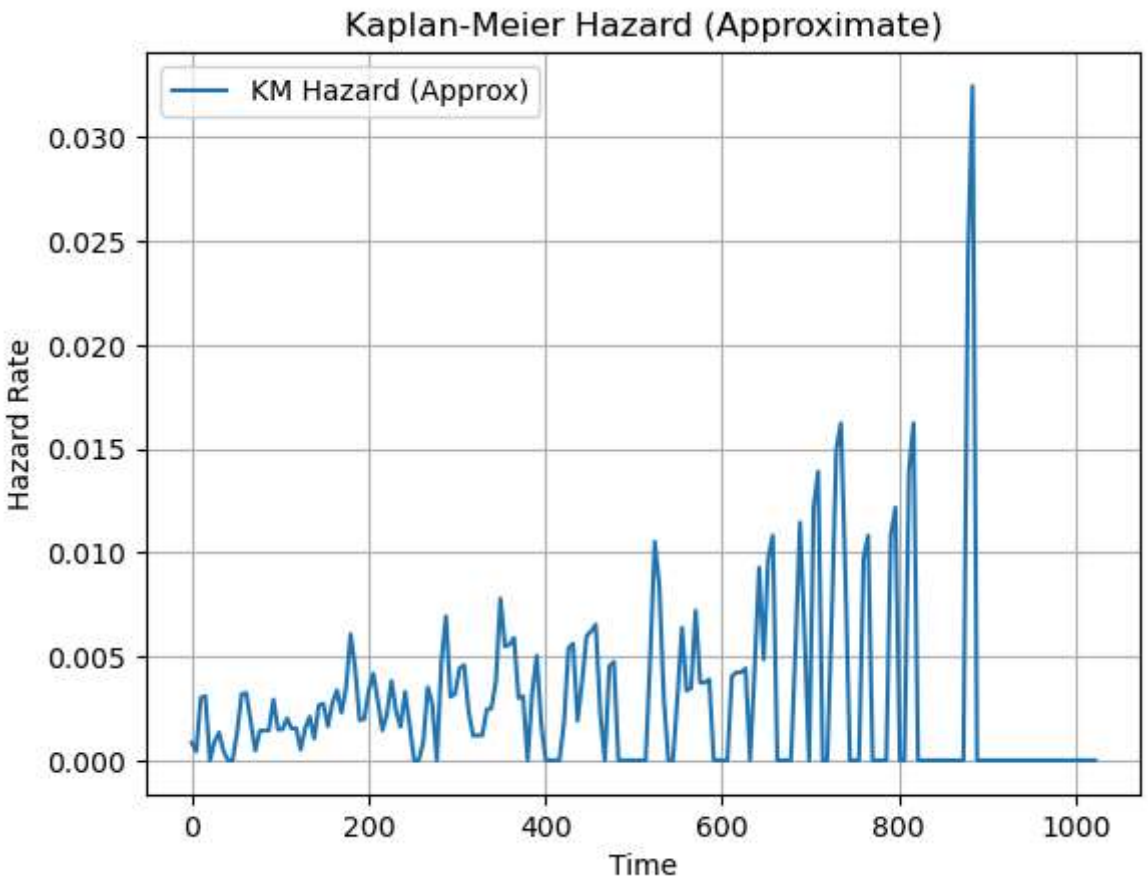
In [315…

```
kmf.plot_survival_function()
plt.title("Kaplan-Meier Survival Function")
plt.grid(True)
plt.show()
```



In [316…

```
# Approximate hazard: -dS/dt / S(t)
timeline = np.linspace(0, T.max(), 200)
surv_vals = kmf.predict(timeline)
hazard_approx = -np.gradient(surv_vals, timeline) / surv_vals

plt.plot(timeline, hazard_approx, label="KM Hazard (Approx)")
plt.title("Kaplan-Meier Hazard (Approximate)")
plt.xlabel("Time")
plt.ylabel("Hazard Rate")
plt.grid(True)
plt.legend()
plt.show()
```



In [317…

```
from numpy import trapz

km_surv = kmf.survival_function_at_times(timeline).values.flatten()
mttf_km = trapz(km_surv, timeline)

print(f"Kaplan-Meier MTTF: {mttf_km:.2f}")
```

```
Kaplan-Meier MTTF: 376.36
```

In [318…

```
# --- Weibull ---
weibull_dict = {
    'Model': 'Weibull',
    'AIC': wf.AIC_,
    'BIC': wf.BIC_,
    'Log-Likelihood': -wf.log_likelihood_,
    'Median Survival Time': wf.median_survival_time_,
    'MTTF': mttf_weibull
}

# --- Normal ---
normal_dict = {
    'Model': 'Normal',
    'AIC': AIC,
    'BIC': BIC,
    'Log-Likelihood': -log_likelihood,
    'MTTF': MTTF
}

# --- Gamma ---
gamma_dict = {
    'Model': 'Gamma',
```

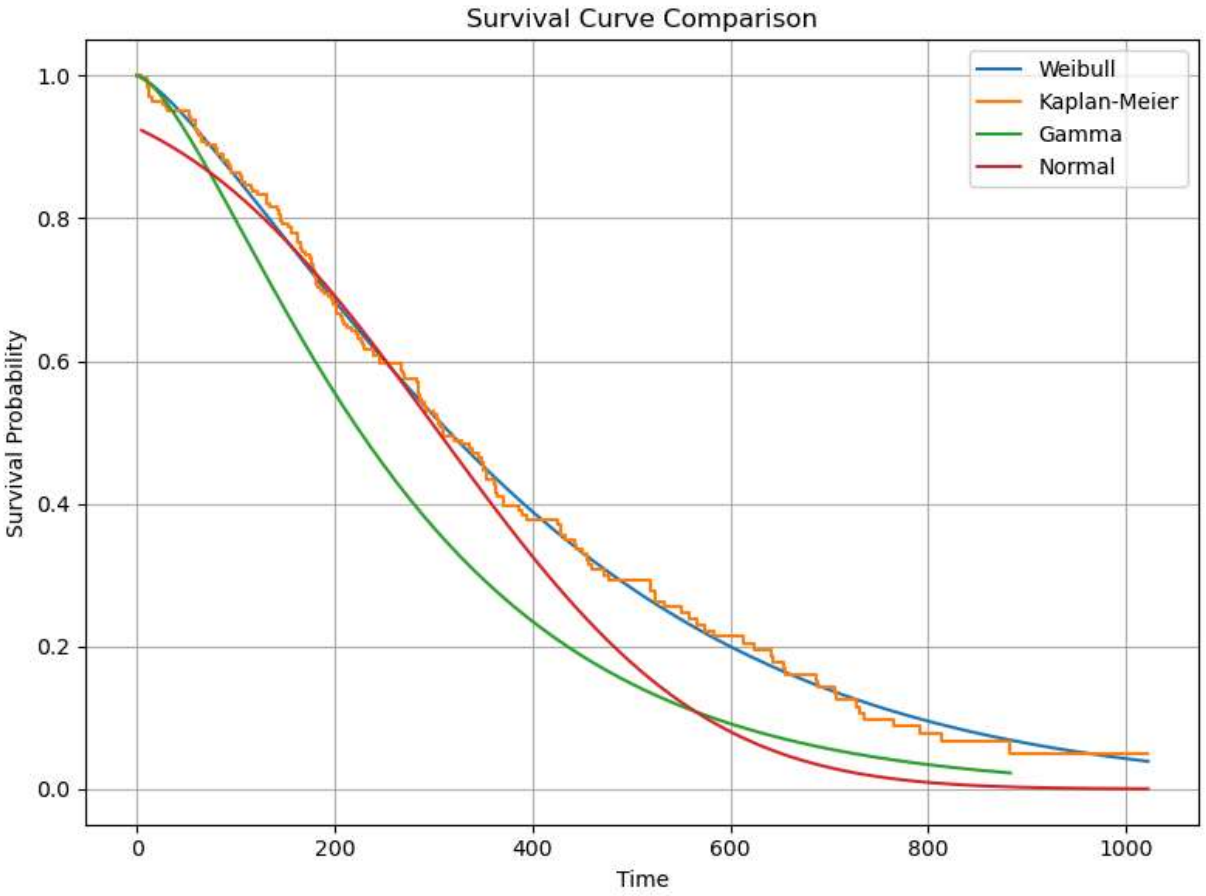
```
    'AIC': aic_gamma,
    'BIC': bic_gamma,
    'Log-Likelihood': -log_likelihood_gamma,
    'Median Survival Time': np.median(uncensored_T),
    'MTTF': mttf_gamma
}

# --- Kaplan-Meier ---
km_dict = {
    'Model': 'Kaplan-Meier',
    'AIC': np.nan,  # Not defined
    'BIC': np.nan,
    'Log-Likelihood': np.nan,
    'Median Survival Time': kmf.median_survival_time_,
    'MTTF': mttf_km
}

# Combine into a DataFrame
summary_df = pd.DataFrame([km_dict, weibull_dict, gamma_dict, normal_dict])
summary_df = summary_df[['Model', 'AIC', 'BIC', 'Log-Likelihood', 'Median Survival Time',
'MTTF']]

# Round values
summary_df = summary_df.round(3)

# Show summary
print("Summary Comparison of All Models:")
print(summary_df)
```

```
Summary Comparison of All Models:
          Model       AIC       BIC  Log-Likelihood  Median Survival Time  \
0  Kaplan-Meier       NaN       NaN             NaN               310.000
1       Weibull  2311.702  2318.561        1153.851               316.264
2         Gamma  2178.398  2184.610        1087.199               226.000
3        Normal  2396.801  2403.660        1196.401                   NaN

      MTTF
0  376.358
1  384.853
2  283.000
3  305.232
```

In [319…

```
plt.figure(figsize=(8, 6))

# Weibull
wf.plot_survival_function(label="Weibull", ci_show=False)

# Kaplan-Meier
kmf.plot_survival_function(label="Kaplan-Meier", ci_show=False)

# Gamma (manual)
plt.plot(x_vals, gamma_surv, label="Gamma")

# Normal
plt.plot(t_vals, survival_function(t_vals), label='Normal')

plt.title("Survival Curve Comparison")
plt.xlabel("Time")
plt.ylabel("Survival Probability")
plt.grid(True)
plt.legend()
```

```
plt.tight_layout()
plt.show()
```



- In this scenario, despite the Gamma distribution having the lowest AIC, the Weibull and Kaplan-Meier survival curves being almost identical, and the Gamma curve being below those, it's recommended to favor the Weibull model and possibly the Kaplan-Meier estimator as the primary methods for analyzing the data. The Kaplan-Meier estimator provides a non-parametric estimate of the survival function, while the Weibull distribution offers a parametric model that can be adjusted for covariates. While the Gamma model shows the best fit based on AIC, its poor performance on the survival curves suggests it might not be the most appropriate model for this specific dataset.

COX PROPORTIONAL METHOD

- exp(coef) is the hazard ratio: a. > 1 = higher risk b. < 1 = protective
- A significant p-value (< 0.05) indicates the variable has a statistically significant effect.

Cause & Treatment Effect

a. Cause/Risk Factors: age, sex, and ecog (performance score)

b. Higher age or worse ecog = higher hazard (worse survival)

c. Treatment Proxy: karnoPH and karnoPAT

Higher scores = better functioning = lower hazard = beneficial treatment effect

In [323…

```
from lifelines import CoxPHFitter

cph = CoxPHFitter()
cph.fit(df[['TIME', 'Y', 'age', 'sex', 'ecog', 'karnoPH', 'karnoPAT']],
duration_col='TIME', event_col='Y')
cph.print_summary()  # Shows hazard ratios, p-values, etc.
```

model	lifelines.CoxPHFitter
duration col	'TIME'
event col	'Y'
baseline estimation	breslow
number of observations	228
number of events observed	165
partial log-likelihood	-732.72
time fit was run	2025-04-25 16:58:38 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p
age	0.01	1.01	0.01	-0.01	0.03	0.99	1.03	0.00	1.28	0.2(
sex	-0.57	0.56	0.17	-0.90	-0.24	0.40	0.78	0.00	-3.40	<0.005
ecog	0.54	1.71	0.18	0.18	0.90	1.20	2.45	0.00	2.95	<0.005
karnoPH	0.01	1.01	0.01	-0.01	0.03	0.99	1.03	0.00	1.40	0.16
karnoPAT	-0.01	0.99	0.01	-0.02	0.00	0.98	1.00	0.00	-1.45	0.15

Concordance	0.65
Partial AIC	1475.43
log-likelihood ratio test	34.39 on 5 df
-log2(p) of ll-ratio test	18.94

Interpretation of Cause & Treatment

a. Cause (Risk Factors):

- sex, age, and ecog—associated with increased hazard.

b. Treatment Effects:

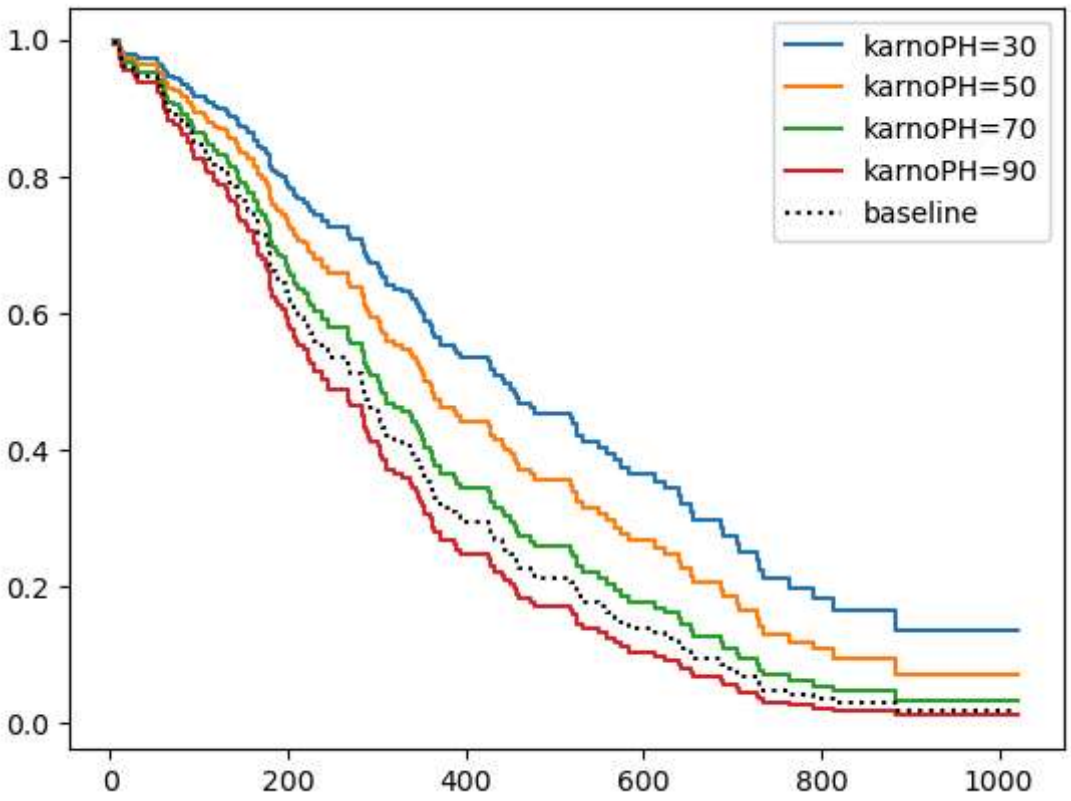
- karnoPH and karnoPAT—higher scores suggest better response to treatment, leading to lower hazard.

Using the Cox model, we can estimate how survival probability changes over time for different levels of physician-assessed Karnofsky score (karnoPH).

- Karnofsky 90: Highest survival probability
- Karnofsky 70: Moderate survival
- Karnofsky 50: Lower survival
- Karnofsky 30: Steep drop in survival

In [326…
```
cph.plot_partial_effects_on_outcome(covariates='karnoPH', values=[30, 50, 70, 90])
```

Out[326… <Axes: >



Interpretation:

- Patients with higher Karnofsky scores are much more likely to survive longer .* This confirms the treatment effect—better functional health (possibly due to treatment or resilience) leads to lower hazar.

In [327…

```
df = df.rename(columns=lambda x: x.lower().strip())
# assuming 'status' = 1 if event occurred, 0 if censored

# Fit Kaplan-Meier estimator
kmf = KaplanMeierFitter()
kmf.fit(durations=T, event_observed=E)

# Fit Weibull model
wf = WeibullFitter()
wf.fit(T, event_observed=E)

# Create bins for time intervals (e.g., deciles)
bins = np.percentile(T, np.linspace(0, 100, 11))
df['time_bin'] = pd.cut(T, bins=bins, include_lowest=True)

# Observed deaths per bin (from actual data)
observed = df[E == 1].groupby('time_bin').size().reindex(df['time_bin'].cat.categories,
fill_value=0)

# Midpoints of bins for estimating expected events
midpoints = [(interval.left + interval.right) / 2 for interval in
df['time_bin'].cat.categories]

# Expected deaths using Weibull model
total_n = len(df)
expected_probs = []

for t1, t2 in zip(bins[:-1], bins[1:]):
    s1 = wf.survival_function_at_times(t1).values[0]
    s2 = wf.survival_function_at_times(t2).values[0]
    expected_probs.append(s1 - s2)  # probability of death in [t1, t2]

expected = np.array(expected_probs) * total_n

# Chi-square statistic
chi_square = np.sum((observed - expected) ** 2 / expected)

# Degrees of freedom = (number of bins - 1 - number of parameters estimated)
dfree = len(observed) - 1 - 2  # Weibull has 2 parameters

# p-value
p_value = 1 - chi2.cdf(chi_square, dfree)

print(f"Chi-square statistic: {chi_square:.4f}")
print(f"Degrees of freedom: {dfree}")
print(f"P-value: {p_value:.4f}")

if p_value > 0.05:
    print("Fail to reject null: Weibull is a good fit.")
else:
    print("Reject null: Weibull is not a good fit.")
```

```
Chi-square statistic: 27.1874
Degrees of freedom: 7
P-value: 0.0003
Reject null: Weibull is not a good fit.
```

```
C:\Users\singh\AppData\Local\Temp\ipykernel_8904\169077401.py:17: FutureWarning: The defaul
t of observed=False is deprecated and will be changed to True in a future version of panda
s. Pass observed=False to retain current behavior or observed=True to adopt the future defa
ult and silence this warning.
  observed = df[E == 1].groupby('time_bin').size().reindex(df['time_bin'].cat.categories, f
ill_value=0)
```

Hence Kaplan-Meier model for estimation of survival function is the best fit model for the chosen datset

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: