Customer Churn Analysis

Customer churn analysis is the process of identifying and understanding why customers stop using a company's products or services. This analysis is crucial for both retail and commercial segments as it helps businesses predict and mitigate customer churn, ultimately improving customer retention and profitability.

Importing Libraries

In [ ]:
```
# pip install lifelines

# lifelines package, which is used for survival analysis in Python.
# This package provides tools to estimate survival functions, compare survival curves, and fit survival models,
# helping you analyze the time until events like customer churn occur.
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
import statsmodels.api as st
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()

#Lifelines is a survival analysis package
from lifelines import KaplanMeierFitter
from lifelines.statistics import multivariate_logrank_test
from lifelines.statistics import logrank_test
from lifelines import CoxPHFitter
```

Data Preparation

In [14]:
```
df = pd.read_csv("Telco_churn.csv")
df.head(2)
```

Out[14]:

	Unnamed: 0	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DevicePr
0	0	7590-VHVEG	Female	False	True	False	1	False	NaN	DSL	...	
1	1	5575-GNVDE	Male	False	False	False	34	True	False	DSL	...	

2 rows × 22 columns

In [15]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0        5043 non-null   int64  
 1   customerID        5043 non-null   object 
 2   gender            5043 non-null   object 
 3   SeniorCitizen     5043 non-null   object 
 4   Partner           5043 non-null   object 
 5   Dependents        5043 non-null   object 
 6   tenure            5043 non-null   int64  
 7   PhoneService      5043 non-null   object 
 8   MultipleLines     4774 non-null   object 
 9   InternetService   5043 non-null   object 
 10  OnlineSecurity    4392 non-null   object 
 11  OnlineBackup      4392 non-null   object 
 12  DeviceProtection  4392 non-null   object 
 13  TechSupport       4392 non-null   object 
 14  StreamingTV       4392 non-null   object 
 15  StreamingMovies   4392 non-null   object 
 16  Contract          5043 non-null   object 
 17  PaperlessBilling  5043 non-null   object 
 18  PaymentMethod     5043 non-null   object 
 19  MonthlyCharges    5043 non-null   float64
 20  TotalCharges      5038 non-null   object 
 21  Churn             5042 non-null   object 
dtypes: float64(1), int64(2), object(19)
memory usage: 866.9+ KB
```

In [16]:

```
df.Churn = labelencoder.fit_transform(df.Churn)
df.Churn.value_counts()
```

Out[16]:

```
Churn
0    2219
1    1487
2     780
3     556
4       1
Name: count, dtype: int64
```

In [18]:
```
# For the analysis, I will need to create dummy variables for all categorical variables.
eventvar = df['Churn']
timevar = df['tenure']
```

In [19]:
```
categorical = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod']

survivaldata = pd.get_dummies(df, columns = categorical, drop_first= True)
survivaldata.head()
```

Out[19]:

	Unnamed: 0	customerID	tenure	MonthlyCharges	TotalCharges	Churn	gender_Male	SeniorCitizen_1	SeniorCitizen_False	Senior
0	0	7590-VHVEG	1	29.850000	29.850000381469727	0	False	False	True	
1	1	5575-GNVDE	34	56.950001	1889.5	0	True	False	True	
2	2	3668-QPYBK	2	53.849998	108.1500015258789	2	True	False	True	
3	3	7795-CFOCW	45	42.299999	1840.75	0	True	False	True	
4	4	9237-HQITU	2	70.699997	151.64999389648438	2	False	False	True	

5 rows × 57 columns

In [ ]:
```
I droped the variables such as customerID, tenure, Churn as they are not needed in survival data. Also, we need to add constan
```

In [20]:
```
survivaldata.drop(['customerID', 'tenure', 'Churn'], axis = 1, inplace= True)
survivaldata = st.add_constant(survivaldata, prepend=False)
survivaldata.head()
```

Out[20]:

	Unnamed: 0	MonthlyCharges	TotalCharges	gender_Male	SeniorCitizen_1	SeniorCitizen_False	SeniorCitizen_True	Partner_No	Par
0	0	29.850000	29.850000381469727	False	False	True	False	False	
1	1	56.950001	1889.5	True	False	True	False	False	
2	2	53.849998	108.1500015258789	True	False	True	False	False	
3	3	42.299999	1840.75	True	False	True	False	False	
4	4	70.699997	151.64999389648438	False	False	True	False	False	

5 rows × 55 columns

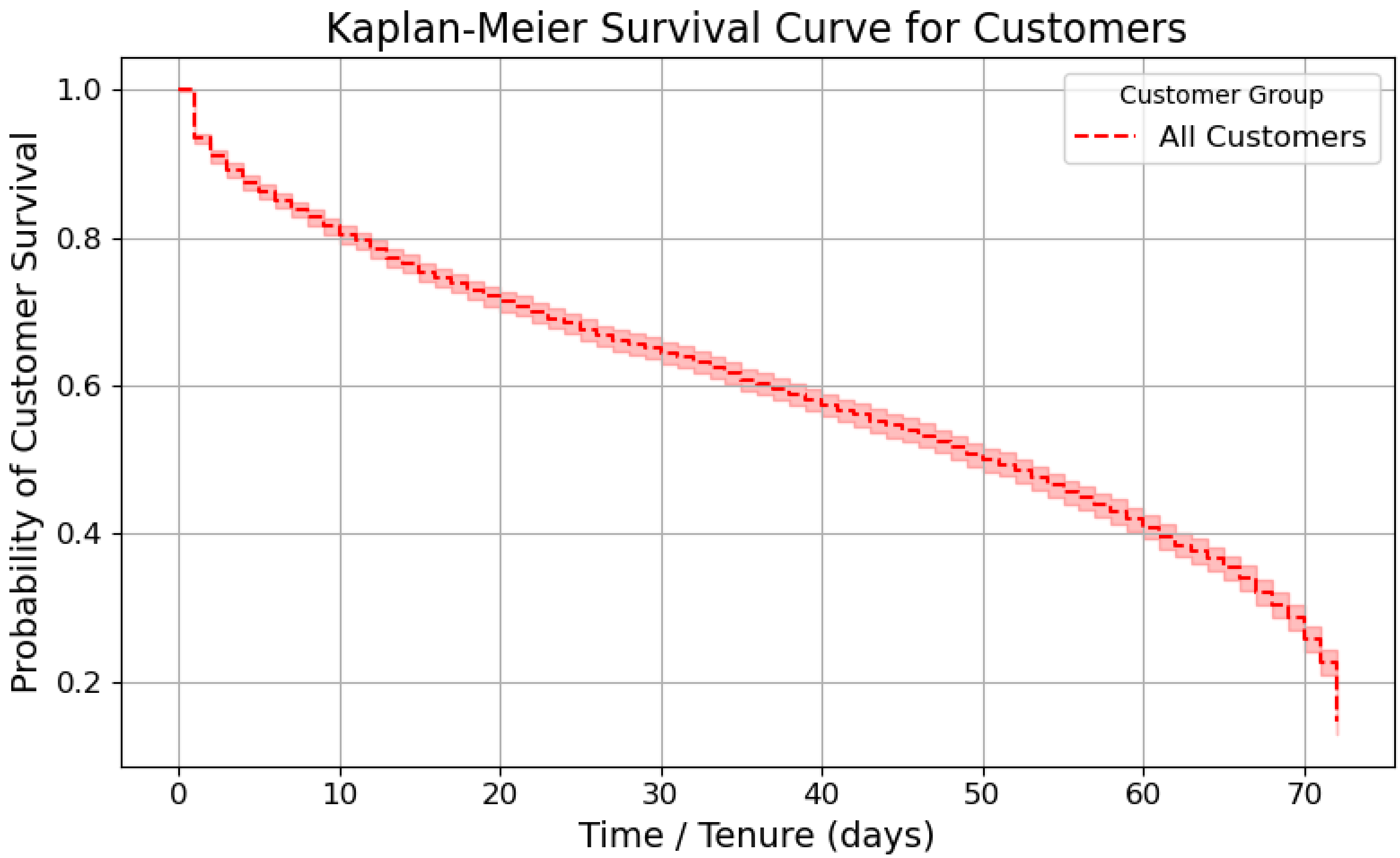
In this context, churn represents the event of a customer leaving, while tenure indicates the duration a customer has stayed with our service. Both variables are crucial for conducting customer survival analysis.

Method1 : Kaplan-Meier Curve

In [34]:

```
# Create a Kaplan-Meier object
kmf = KaplanMeierFitter()

# Fit the model using time and event variables
kmf.fit(timevar, event_observed=eventvar, label="All Customers")

# Plot the survival function
plt.figure(figsize=(8, 5))
kmf.plot(color='red', style='--', ci_show=True)  # Show confidence intervals
plt.title('Kaplan-Meier Survival Curve for Customers', fontsize=16)
plt.xlabel('Time / Tenure (days)', fontsize=14)
plt.ylabel('Probability of Customer Survival', fontsize=14)
plt.grid(True)
plt.legend(title='Customer Group', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
```

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



Findings

I observed a sharp initial drop, indicating that customers begin churning rapidly after their first tenure. However, the churn rate decreases over time. To address this, we could offer more discounts on long-term plans to encourage customers to subscribe for extended periods.

Method 2 : Log-Rank Test

The Log-Rank Test compares the survival times of different groups to see if there are significant differences. For example, it helps determine if one group of customers stays with your service longer than another. It's like comparing the lifespans of two types of light bulbs to see which lasts longer. If the test shows a significant difference, it means the groups have different patterns of staying or leaving.

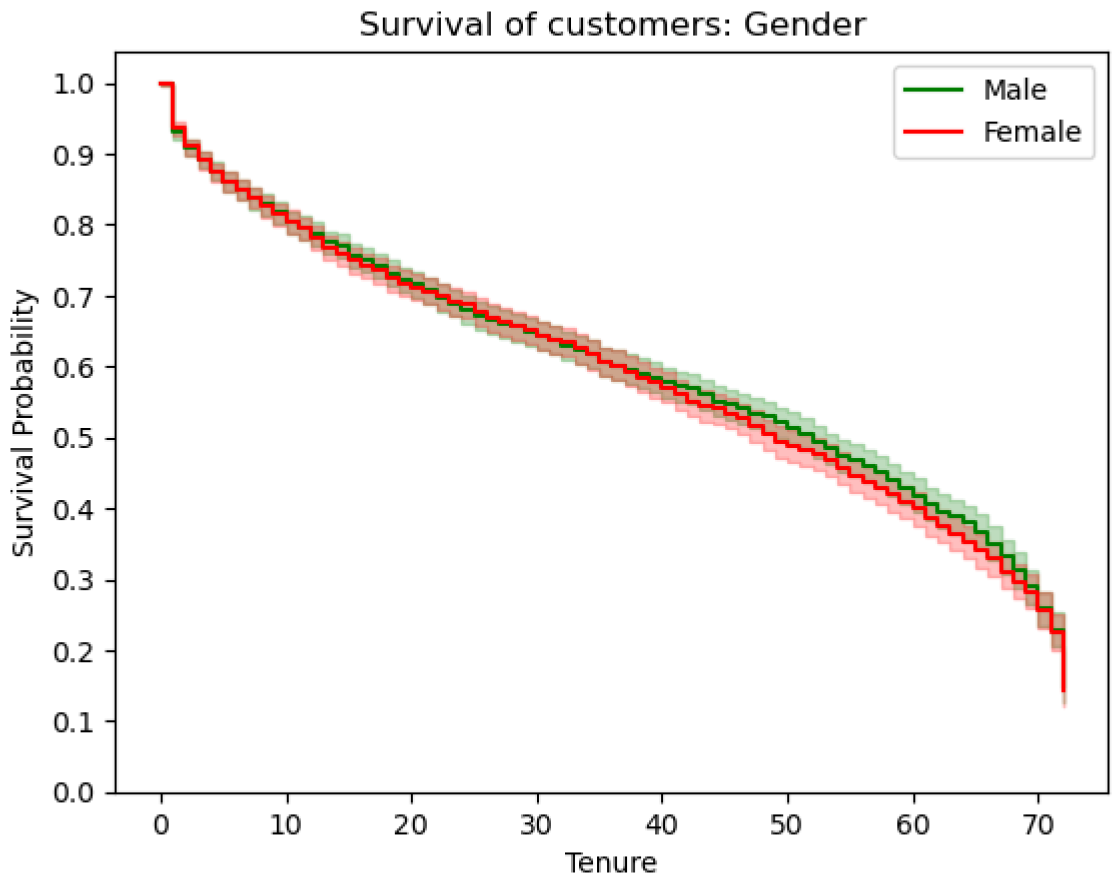
Gender

In [71]:

```
male = (survivaldata['gender_Male'] == 1)
female = (survivaldata['gender_Male'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[male],event_observed = eventvar[male],label = "Male")
plot1 = kmf.plot(ax = ax, color='green')

kmf.fit(timevar[female],event_observed = eventvar[female],label = "Female")
plot2 = kmf.plot(ax = plot1, color='red')

plt.title('Survival of customers: Gender')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[male], timevar[female], event_observed_A=eventvar[male], event_observed_B=eventvar[female])
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	0.63	0.43	1.22

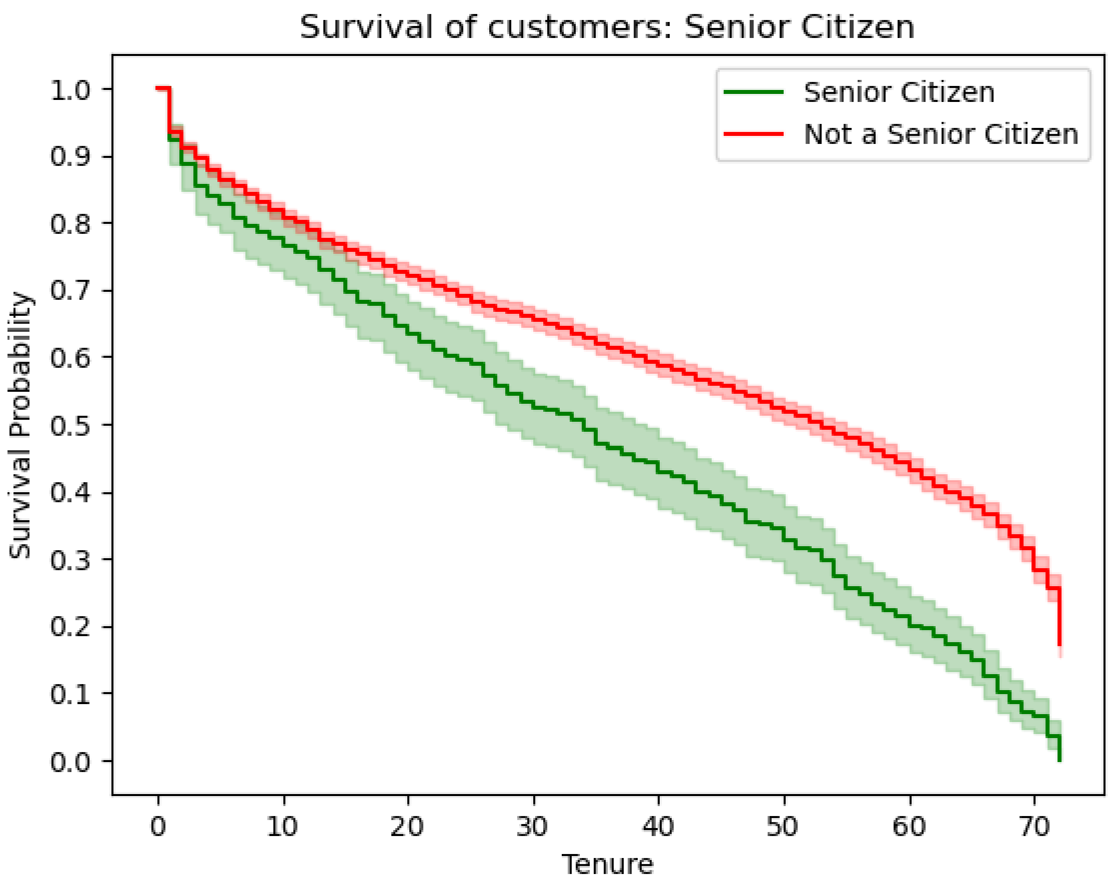


Senior Citizen

In [70]:
```
SeniorCitizen = (survivaldata['SeniorCitizen_1'] == 1)
no_SeniorCitizen = (survivaldata['SeniorCitizen_1'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[SeniorCitizen],event_observed = eventvar[SeniorCitizen],label = "Senior Citizen")
plot1 = kmf.plot(ax = ax, color='green')

kmf.fit(timevar[no_SeniorCitizen],event_observed = eventvar[no_SeniorCitizen],label = "Not a Senior Citizen")
plot2 = kmf.plot(ax = plot1, color='red')

plt.title('Survival of customers: Senior Citizen')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[SeniorCitizen], timevar[no_SeniorCitizen], event_observed_A=eventvar[SeniorCitizen], event_obser
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	122.27	<0.005	92.00



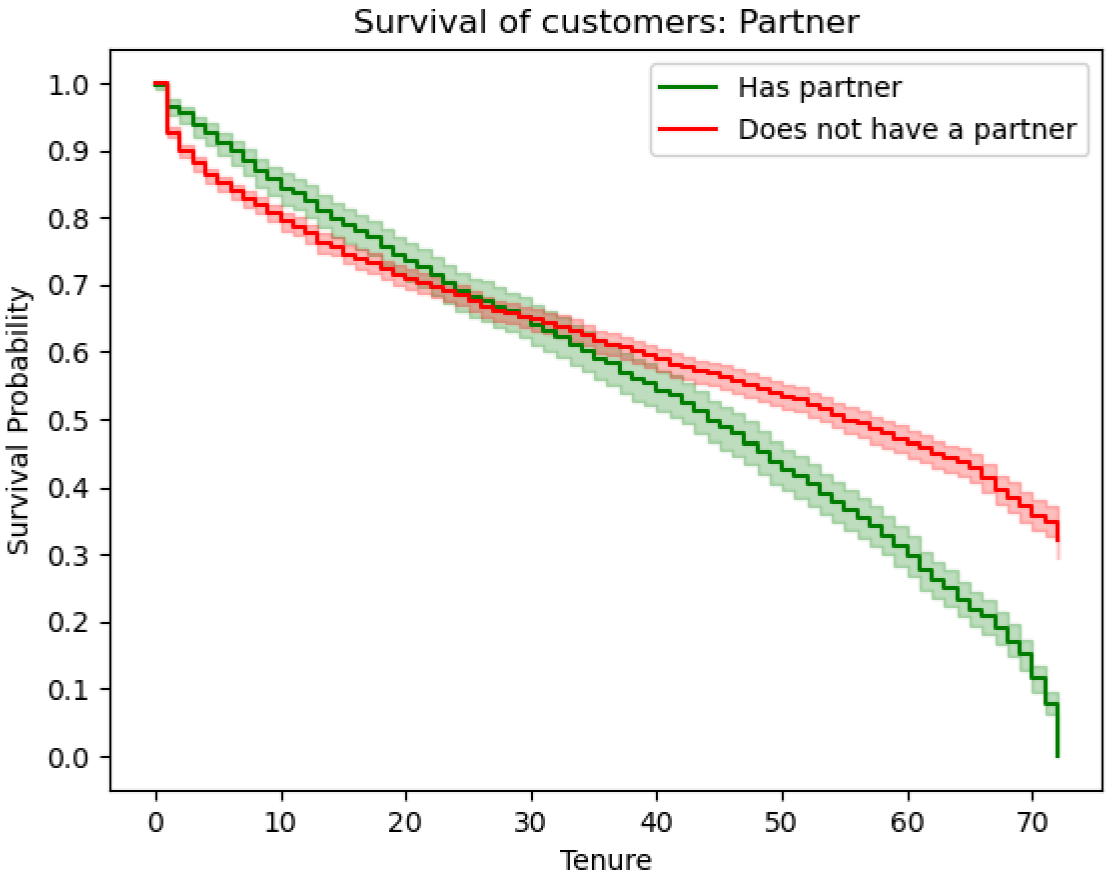
Partner

In [69]:
```
partner = (survivaldata['Partner_Yes'] == 1)
no_partner = (survivaldata['Partner_Yes'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[partner],event_observed = eventvar[partner],label = "Has partner")
plot1 = kmf.plot(ax = ax, color='green')

kmf.fit(timevar[no_partner],event_observed = eventvar[no_partner],label = "Does not have a partner")
```

```
plot2 = kmf.plot(ax = plot1, color='red')

plt.title('Survival of customers: Partner')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[partner], timevar[no_partner], event_observed_A=eventvar[partner], event_observed_B=eventvar[no_
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	166.13	<0.005	123.86



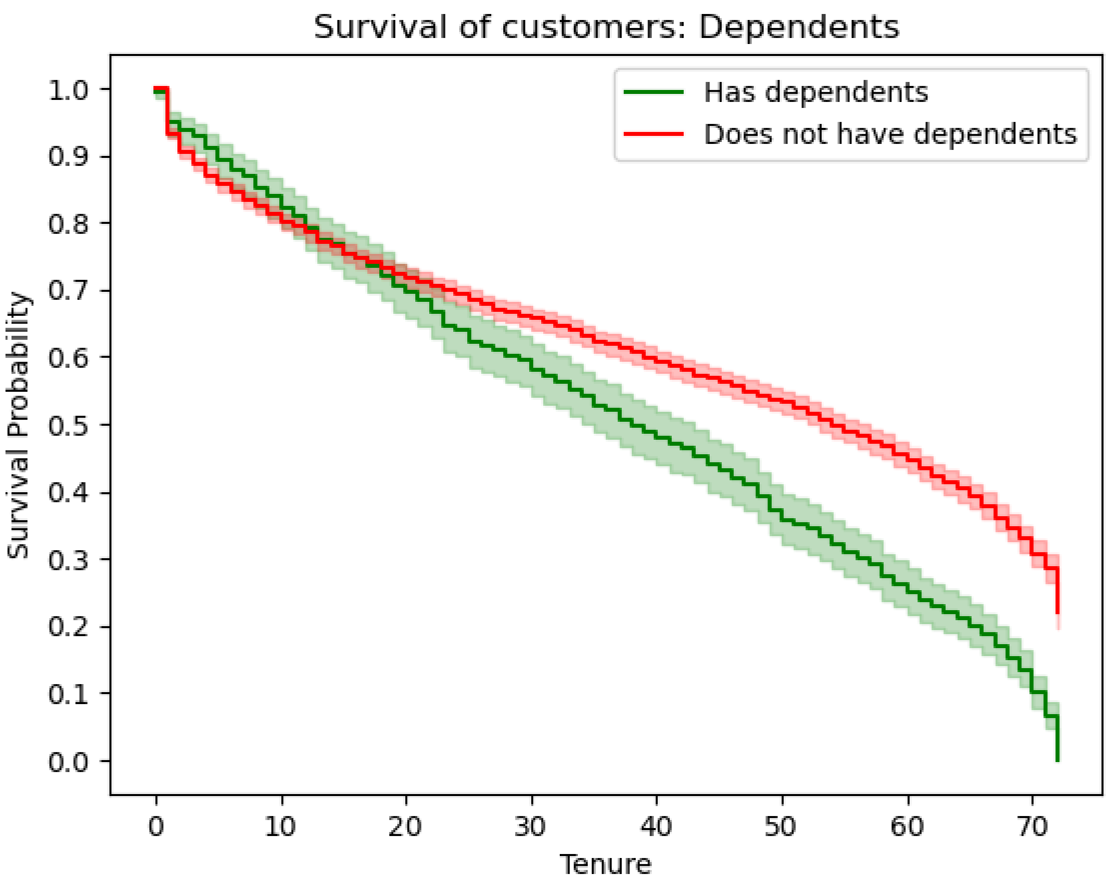
Dependents

In [68]:
```
Dependents = (survivaldata['Dependents_Yes'] == 1)
no_Dependents = (survivaldata['Dependents_Yes'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[Dependents],event_observed = eventvar[Dependents],label = "Has dependents")
plot1 = kmf.plot(ax = ax, color='green')

kmf.fit(timevar[no_Dependents],event_observed = eventvar[no_Dependents],label = "Does not have dependents")
```

```
plot2 = kmf.plot(ax = plot1, color='red')

plt.title('Survival of customers: Dependents')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[Dependents], timevar[no_Dependents], event_observed_A=eventvar[Dependents], event_observed_B=eve
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	140.22	<0.005	105.05



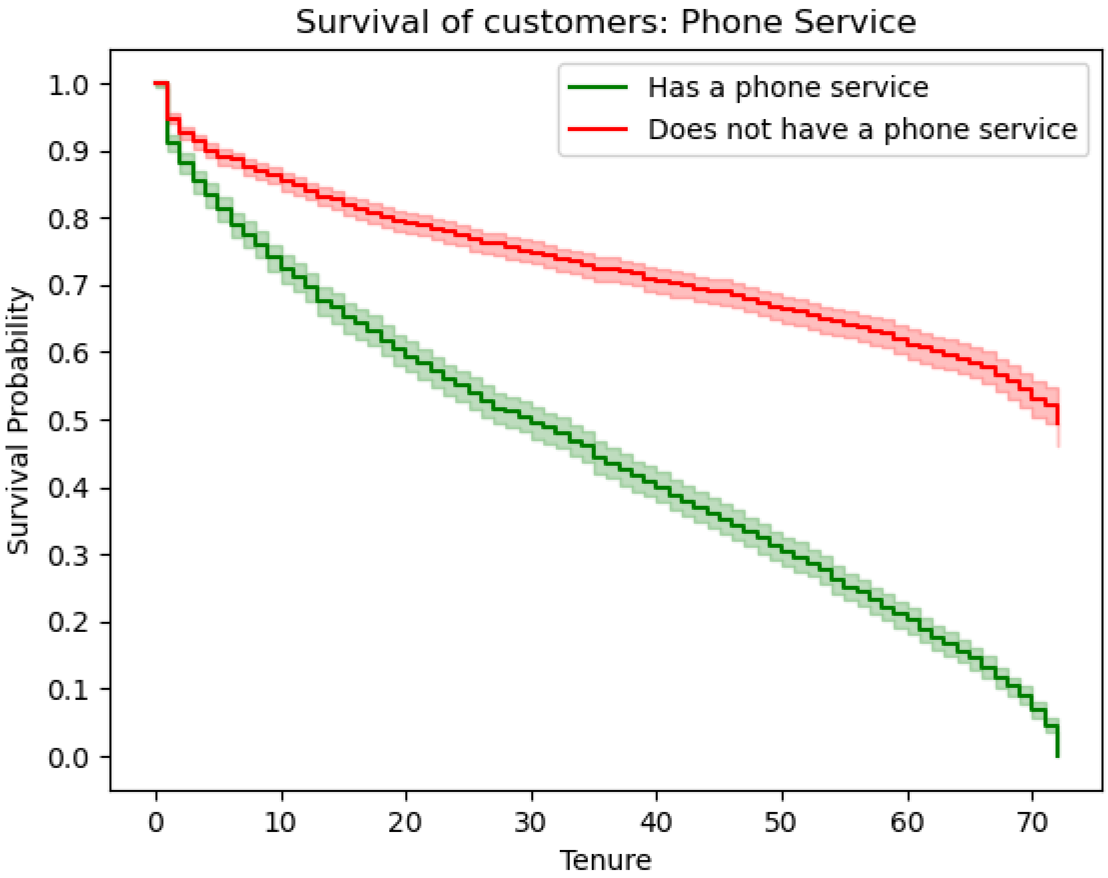
In [2]:
```
### PhoneService
```

In [67]:
```
PhoneService = (survivaldata['PhoneService_Yes'] == 1)
no_PhoneService = (survivaldata['PhoneService_Yes'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[PhoneService],event_observed = eventvar[PhoneService],label = "Has a phone service")
plot1 = kmf.plot(ax = ax, color='green')

kmf.fit(timevar[no_PhoneService],event_observed = eventvar[no_PhoneService],label = "Does not have a phone service")
```

```
plot2 = kmf.plot(ax = plot1, color='red')

plt.title('Survival of customers: Phone Service')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[PhoneService], timevar[no_PhoneService], event_observed_A=eventvar[PhoneService], event_observed
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	1039.25	<0.005	754.99



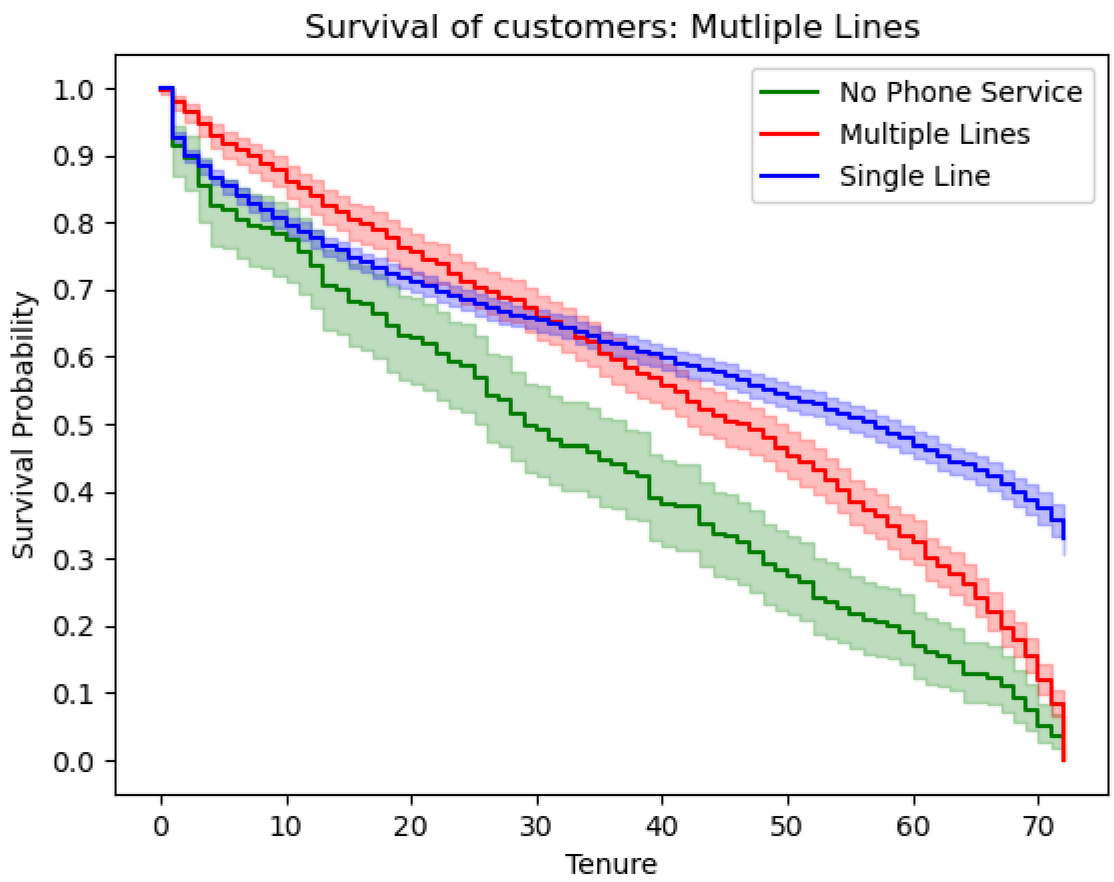
MultipleLines

In [66]:
```
no_phone = (survivaldata['MultipleLines_No phone service'] == 1)
multiLines = (survivaldata['MultipleLines_Yes'] == 1)
no_multiLines = ((survivaldata['MultipleLines_Yes'] == 0) & (survivaldata['MultipleLines_No phone service'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_phone],event_observed = eventvar[no_phone],label = "No Phone Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[multiLines],event_observed = eventvar[multiLines],label = "Multiple Lines")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_multiLines],event_observed = eventvar[no_multiLines],label = "Single Line")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Mutliple Lines')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['MultipleLines'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1496.10	<0.005	inf



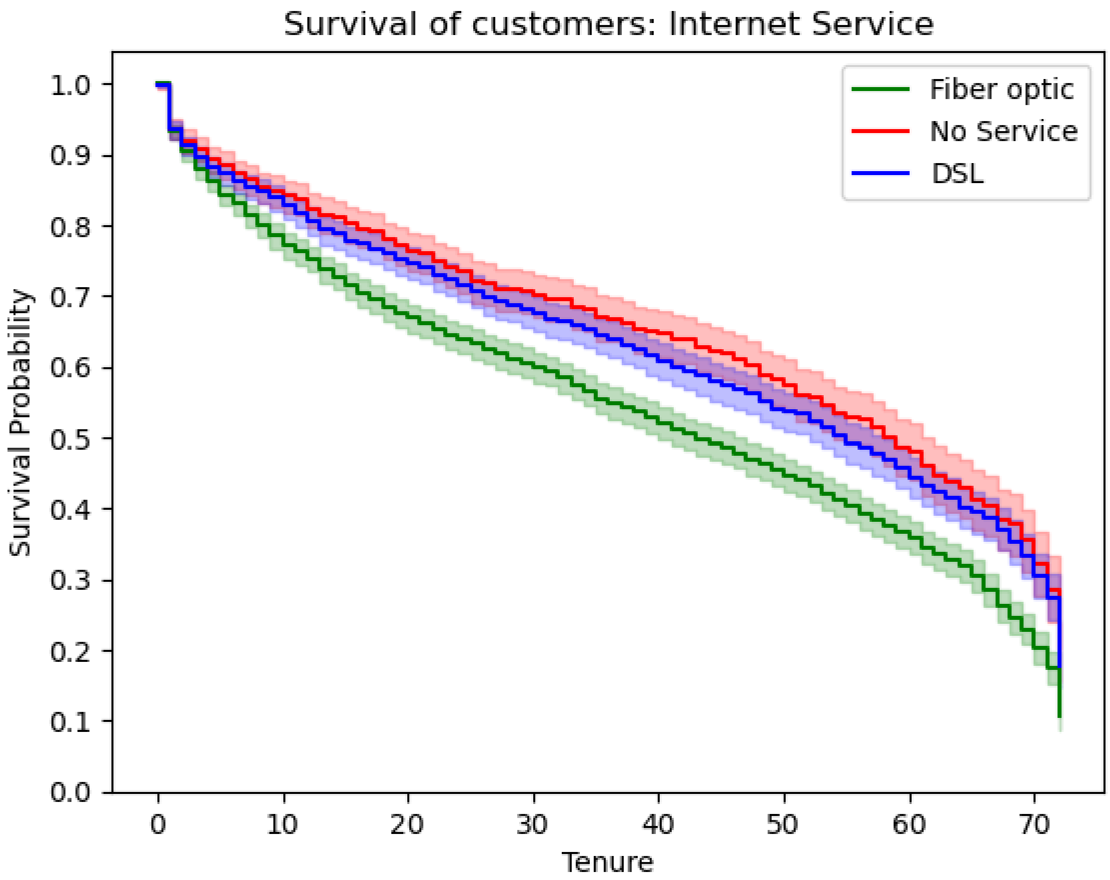
Internet Service

```
In [79]: Fiber_optic = (survivaldata['InternetService_Fiber optic'] == 1)
         No_Service = (survivaldata['InternetService_No'] == 1)
         DSL = ((survivaldata['InternetService_Fiber optic'] == 0) & (survivaldata['InternetService_No'] == 0))

         plt.figure()
         ax = plt.subplot(1,1,1)

         kmf.fit(timevar[Fiber_optic],event_observed = eventvar[Fiber_optic],label = "Fiber optic")
         plot1 = kmf.plot(ax = ax, color='green')
```

```python
kmf.fit(timevar[No_Service],event_observed = eventvar[No_Service],label = "No Service")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[DSL],event_observed = eventvar[DSL],label = "DSL")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Internet Service')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['InternetService'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	2
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	65.15	<0.005	46.99



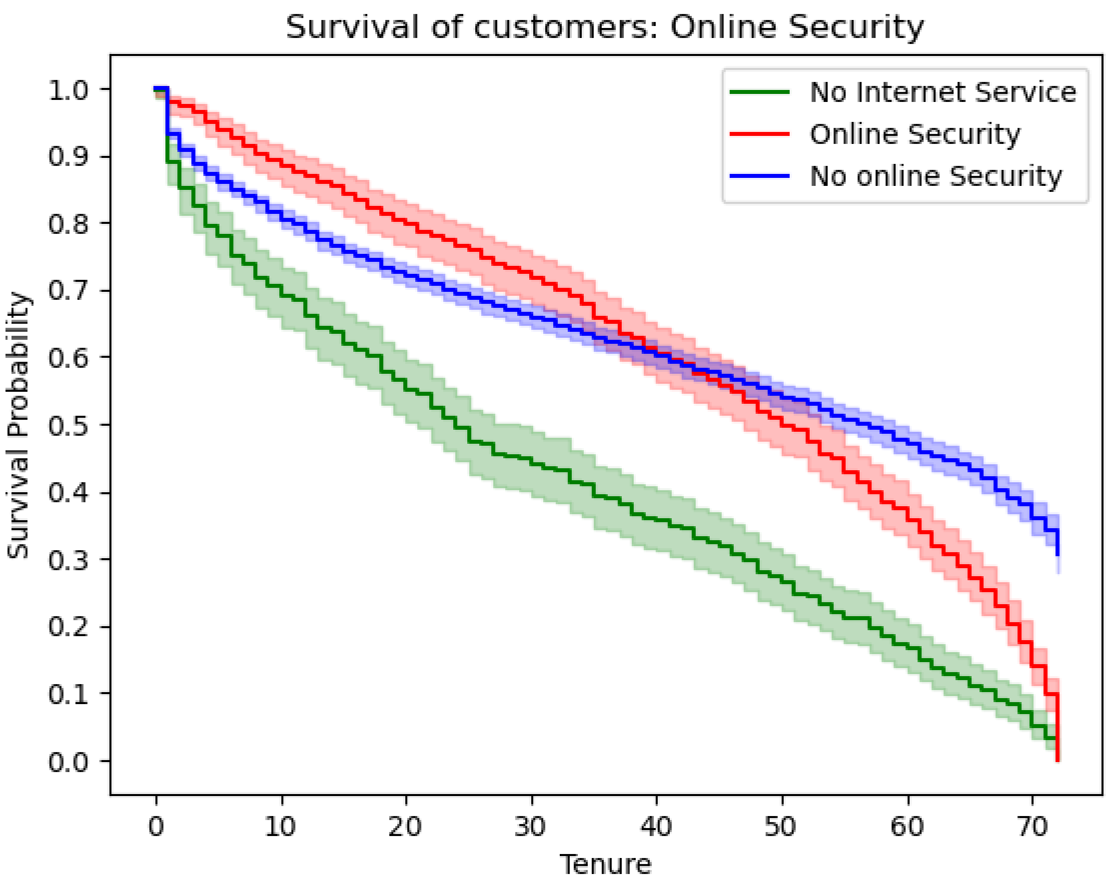
Online Security

In [80]:
```
no_internetService = (survivaldata['OnlineSecurity_No internet service'] == 1)
onlineSecurity = (survivaldata['OnlineSecurity_Yes'] == 1)
no_onlineSecurity = ((survivaldata['OnlineSecurity_No internet service'] == 0) & (survivaldata['OnlineSecurity_Yes'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[onlineSecurity],event_observed = eventvar[onlineSecurity],label = "Online Security")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_onlineSecurity],event_observed = eventvar[no_onlineSecurity],label = "No online Security")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Online Security')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['OnlineSecurity'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1316.79	<0.005	936.23



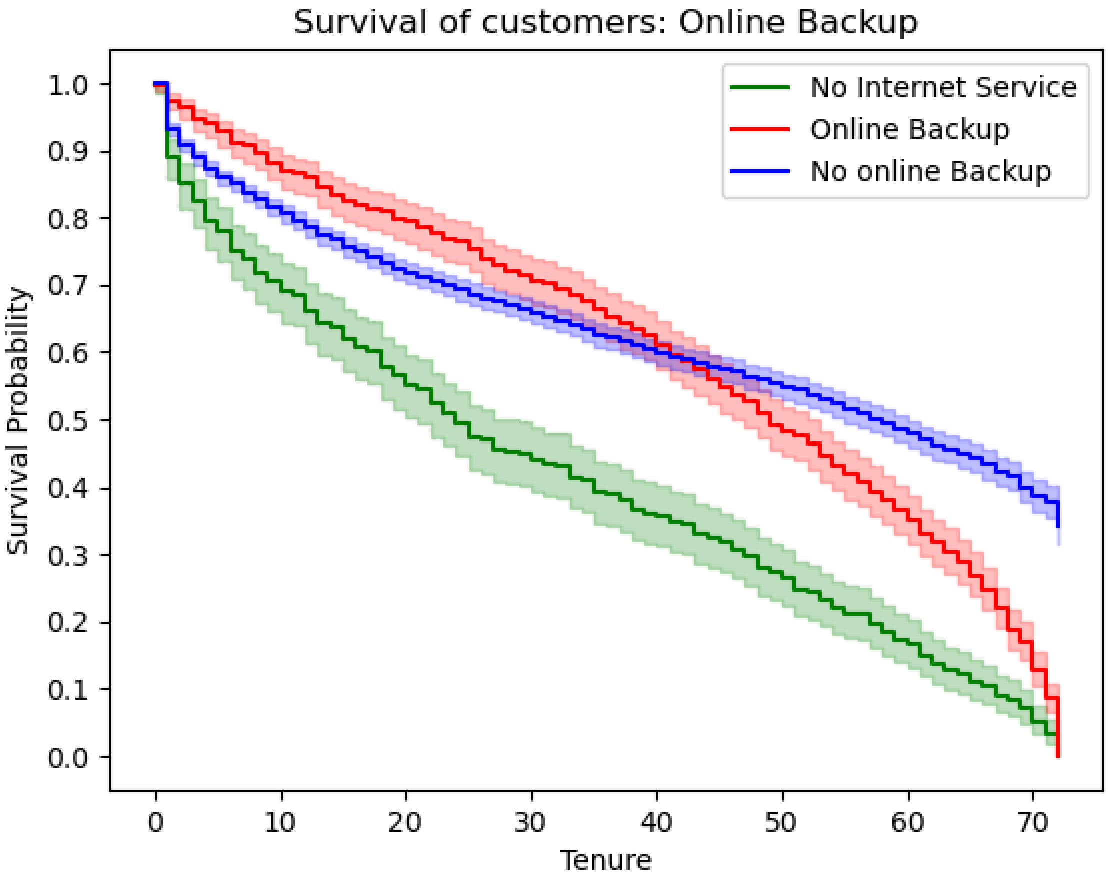
Online Backup

In [81]:
```
no_internetService = (survivaldata['OnlineBackup_No internet service'] == 1)
onlineBackup = (survivaldata['OnlineBackup_Yes'] == 1)
no_onlineBackup = ((survivaldata['OnlineBackup_No internet service'] == 0) & (survivaldata['OnlineBackup_Yes'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[onlineBackup],event_observed = eventvar[onlineBackup],label = "Online Backup")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_onlineBackup],event_observed = eventvar[no_onlineBackup],label = "No online Backup")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Online Backup')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['OnlineBackup'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1340.67	<0.005	953.41



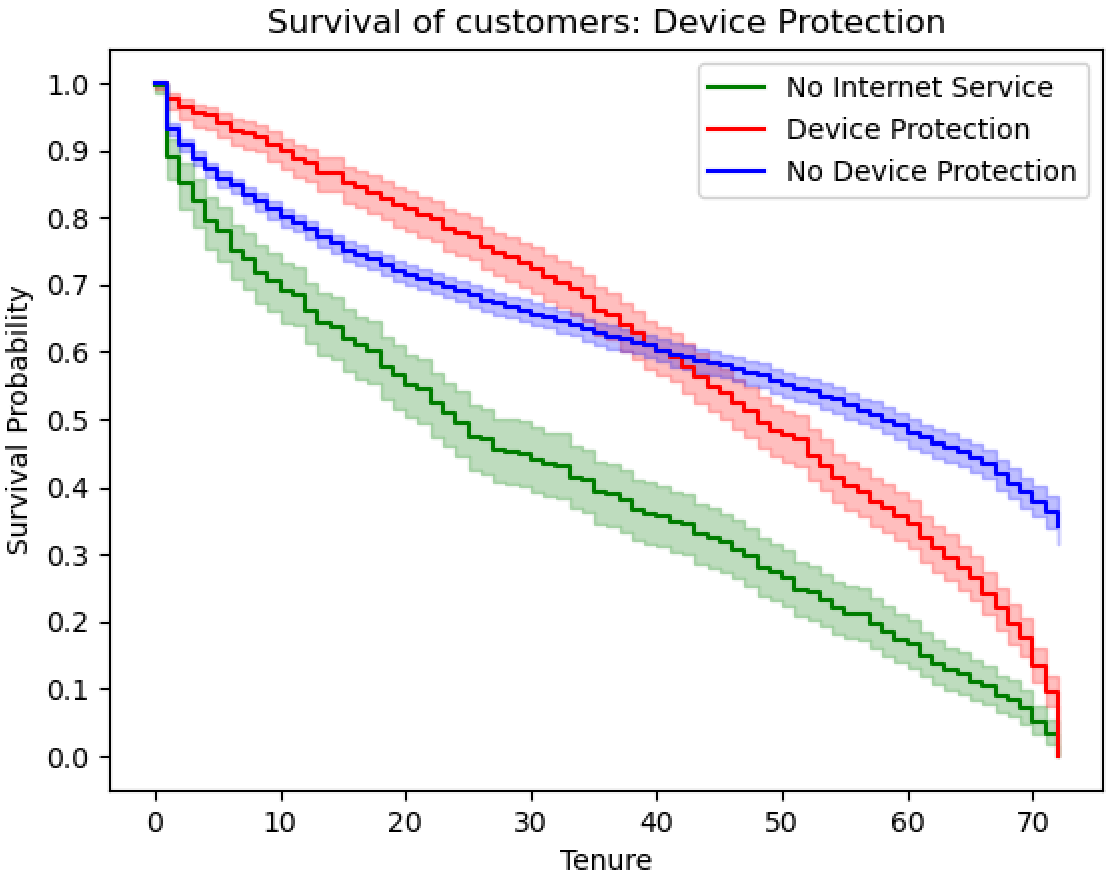
Device Protection

```
In [82]: no_internetService = (survivaldata['DeviceProtection_No internet service'] == 1)
DeviceProtection = (survivaldata['DeviceProtection_Yes'] == 1)
no_DeviceProtection = ((survivaldata['DeviceProtection_No internet service'] == 0) & (survivaldata['DeviceProtection_Yes'] ==

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[DeviceProtection],event_observed = eventvar[DeviceProtection],label = "Device Protection")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_DeviceProtection],event_observed = eventvar[no_DeviceProtection],label = "No Device Protection")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Device Protection')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['DeviceProtection'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1336.84	<0.005	950.66



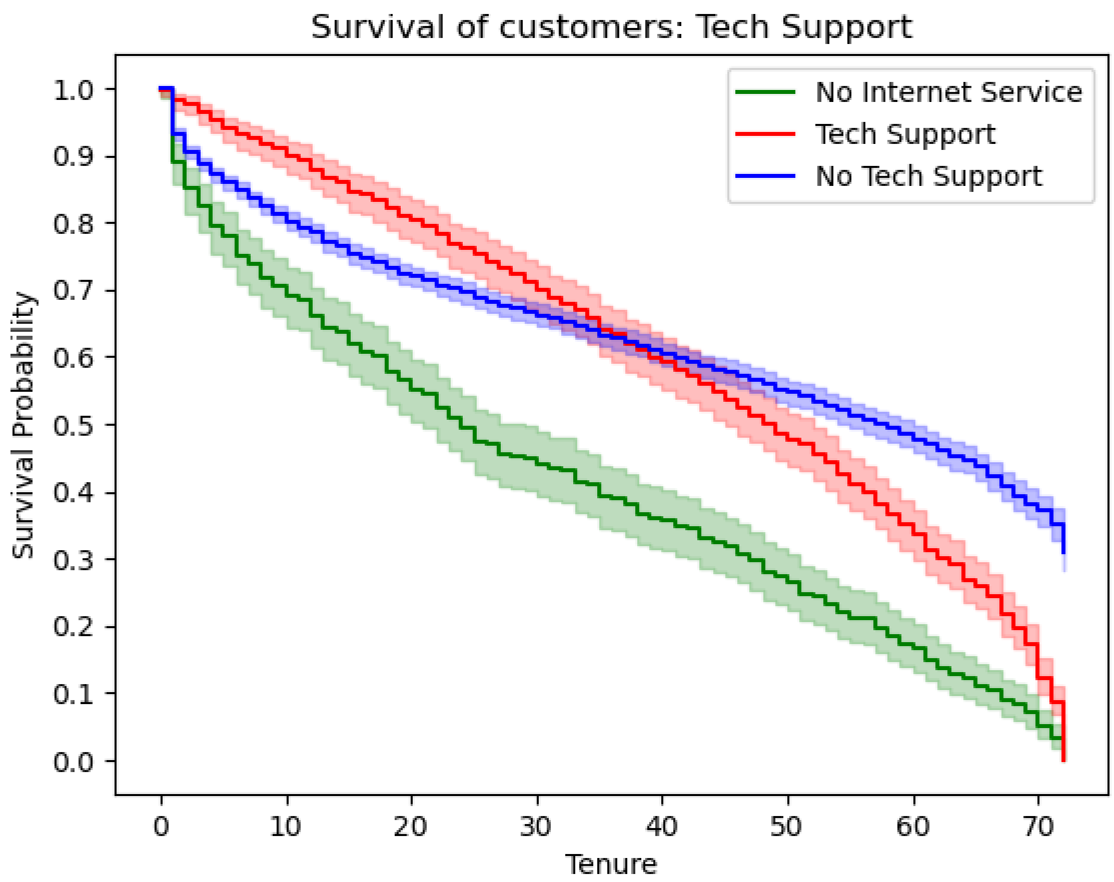
Tech Support

In [85]:
```
no_internetService = (survivaldata['TechSupport_No internet service'] == 1)
TechSupport = (survivaldata['TechSupport_Yes'] == 1)
no_TechSupport = ((survivaldata['TechSupport_No internet service'] == 0) & (survivaldata['TechSupport_Yes'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```python
kmf.fit(timevar[TechSupport],event_observed = eventvar[TechSupport],label = "Tech Support")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_TechSupport],event_observed = eventvar[no_TechSupport],label = "No Tech Support")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Tech Support')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['TechSupport'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1313.83	<0.005	934.10



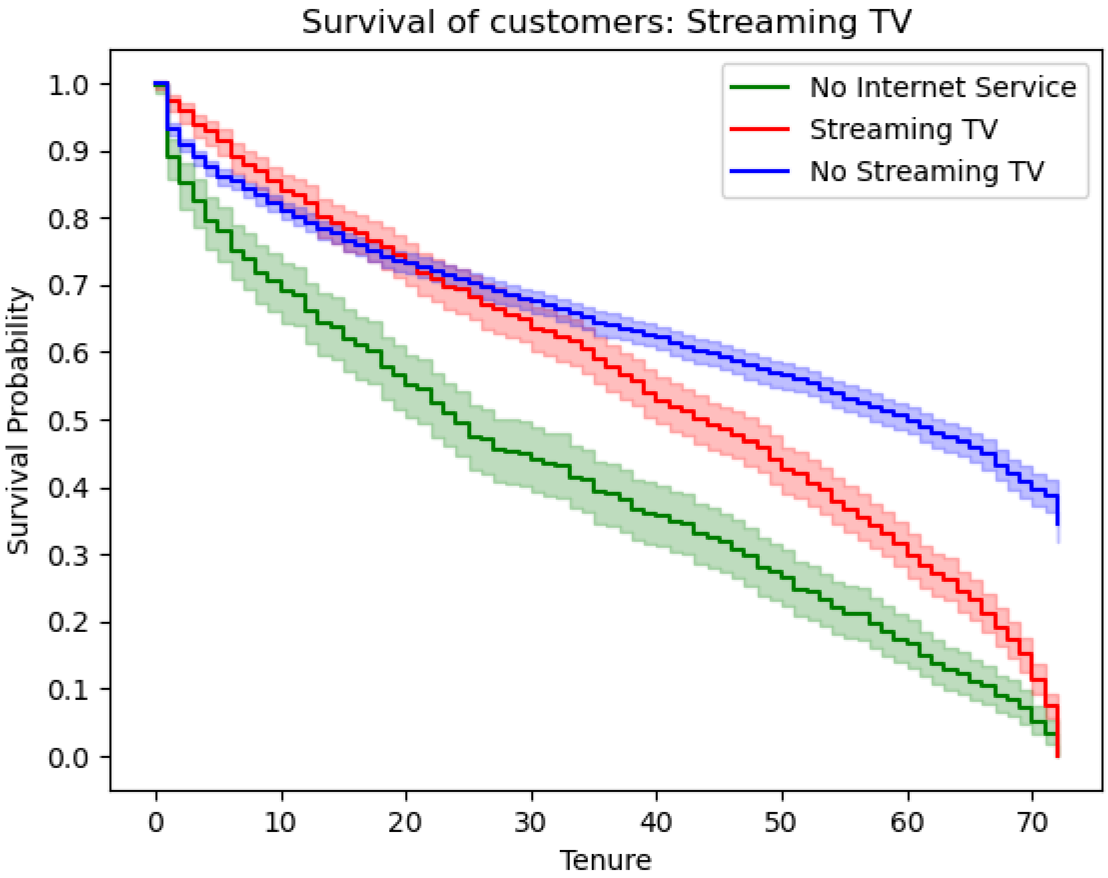
Streaming TV

In [87]:
```
no_internetService = (survivaldata['StreamingTV_No internet service'] == 1)
StreamingTV = (survivaldata['StreamingTV_Yes'] == 1)
no_StreamingTV = ((survivaldata['StreamingTV_No internet service'] == 0) & (survivaldata['StreamingTV_Yes'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[StreamingTV],event_observed = eventvar[StreamingTV],label = "Streaming TV")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_StreamingTV],event_observed = eventvar[no_StreamingTV],label = "No Streaming TV")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Streaming TV')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['StreamingTV'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1090.63	<0.005	773.49



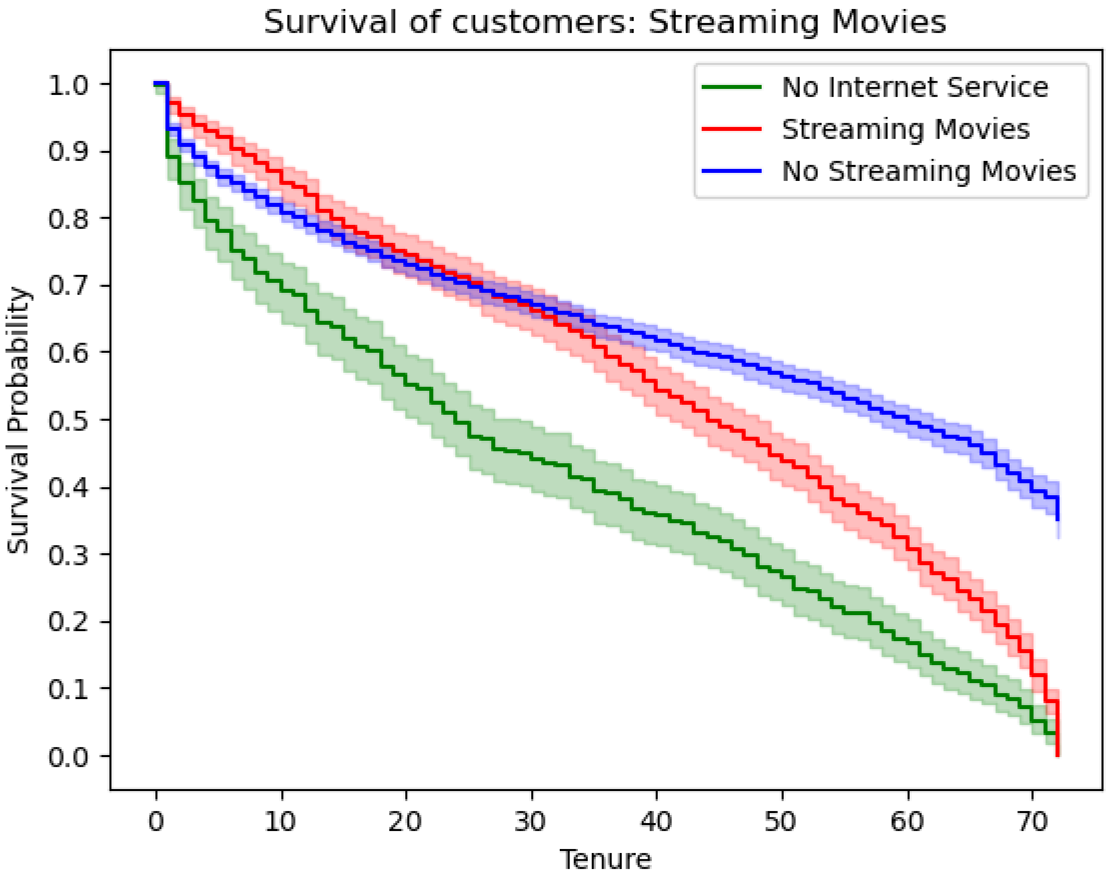
Streaming Movies

In [88]:
```
no_internetService = (survivaldata['StreamingMovies_No internet service'] == 1)
StreamingMovies = (survivaldata['StreamingMovies_Yes'] == 1)
no_StreamingMovies = ((survivaldata['StreamingMovies_No internet service'] == 0) & (survivaldata['StreamingMovies_Yes'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[no_internetService],event_observed = eventvar[no_internetService],label = "No Internet Service")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[StreamingMovies],event_observed = eventvar[StreamingMovies],label = "Streaming Movies")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[no_StreamingMovies],event_observed = eventvar[no_StreamingMovies],label = "No Streaming Movies")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Streaming Movies')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['StreamingMovies'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	5
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	1112.03	<0.005	788.89



Contract

In [90]:
```
Contract_One_year = (survivaldata['Contract_One year'] == 1)
Contract_Two_year = (survivaldata['Contract_Two year'] == 1)
Contract_month_to_month = ((survivaldata['Contract_One year'] == 0) & (survivaldata['Contract_Two year'] == 0))

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[Contract_One_year],event_observed = eventvar[Contract_One_year],label = "One year Contract")
plot1 = kmf.plot(ax = ax, color='green')
```

```
kmf.fit(timevar[Contract_Two_year],event_observed = eventvar[Contract_Two_year],label = "Two year Contract")
plot2 = kmf.plot(ax = plot1, color='red')

kmf.fit(timevar[Contract_month_to_month],event_observed = eventvar[Contract_month_to_month],label = "Month to month Contract")
plot3 = kmf.plot(ax = plot2, color='blue')

plt.title('Survival of customers: Contract')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['Contract'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	2
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	2030.38	<0.005	inf

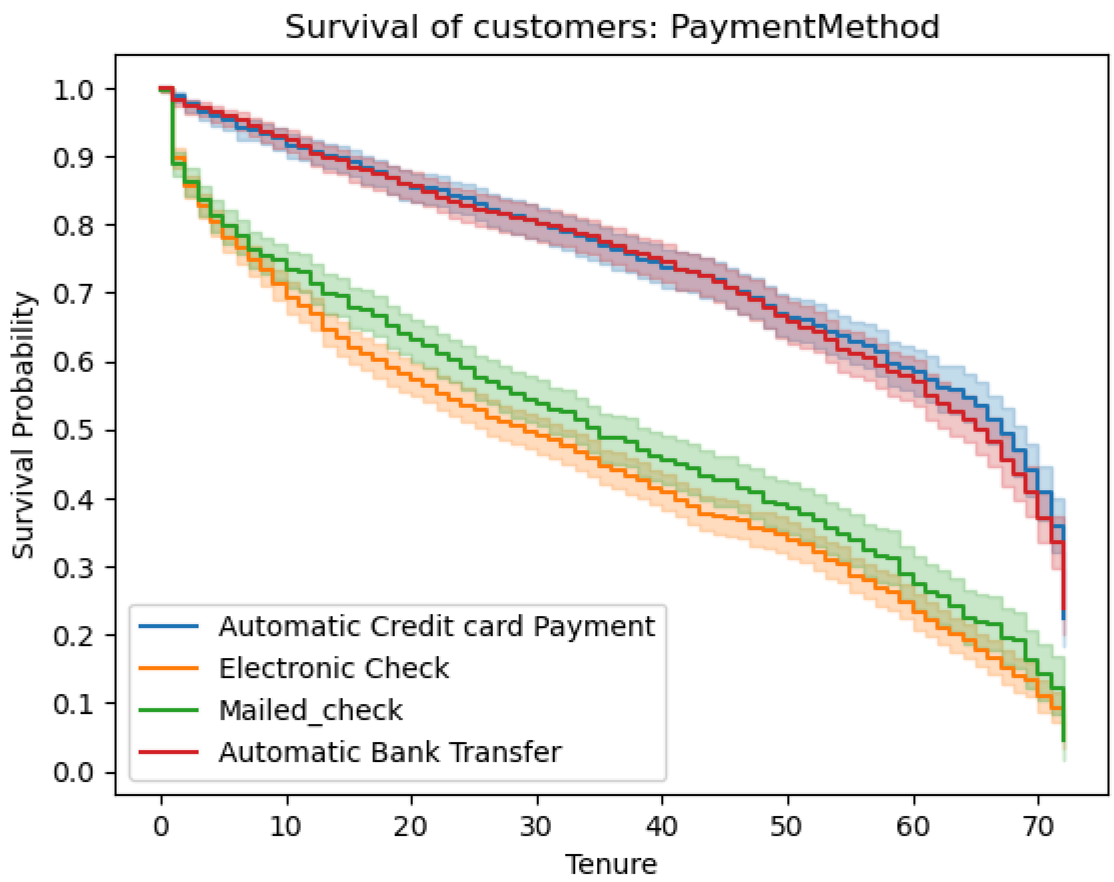
Payment Method

In [91]:
```
automatic_Credit_Card = (survivaldata['PaymentMethod_Credit card (automatic)'] == 1)
electronic_check = (survivaldata['PaymentMethod_Electronic check'] == 1)
mailed_check = (survivaldata['PaymentMethod_Mailed check'] == 1)
automatic_Bank_Transfer = ((survivaldata['PaymentMethod_Credit card (automatic)'] == 0) & (survivaldata['PaymentMethod_Electro

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[automatic_Credit_Card],event_observed = eventvar[automatic_Credit_Card],label = "Automatic Credit card Payment
plot1 = kmf.plot(ax = ax)
```

```python
kmf.fit(timevar[electronic_check],event_observed = eventvar[electronic_check],label = "Electronic Check")
plot2 = kmf.plot(ax = plot1)

kmf.fit(timevar[mailed_check],event_observed = eventvar[mailed_check],label = "Mailed_check")
plot3 = kmf.plot(ax = plot2)

kmf.fit(timevar[automatic_Bank_Transfer],event_observed = eventvar[automatic_Bank_Transfer],label = "Automatic Bank Transfer")
plot4 = kmf.plot(ax = plot3)

plt.title('Survival of customers: PaymentMethod')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
twoplusgroups_logrank = multivariate_logrank_test(df['tenure'], df['PaymentMethod'], df['Churn'], alpha = 0.95)
twoplusgroups_logrank.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	3
alpha	0.95
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	604.63	<0.005	431.85



Paperless Billing

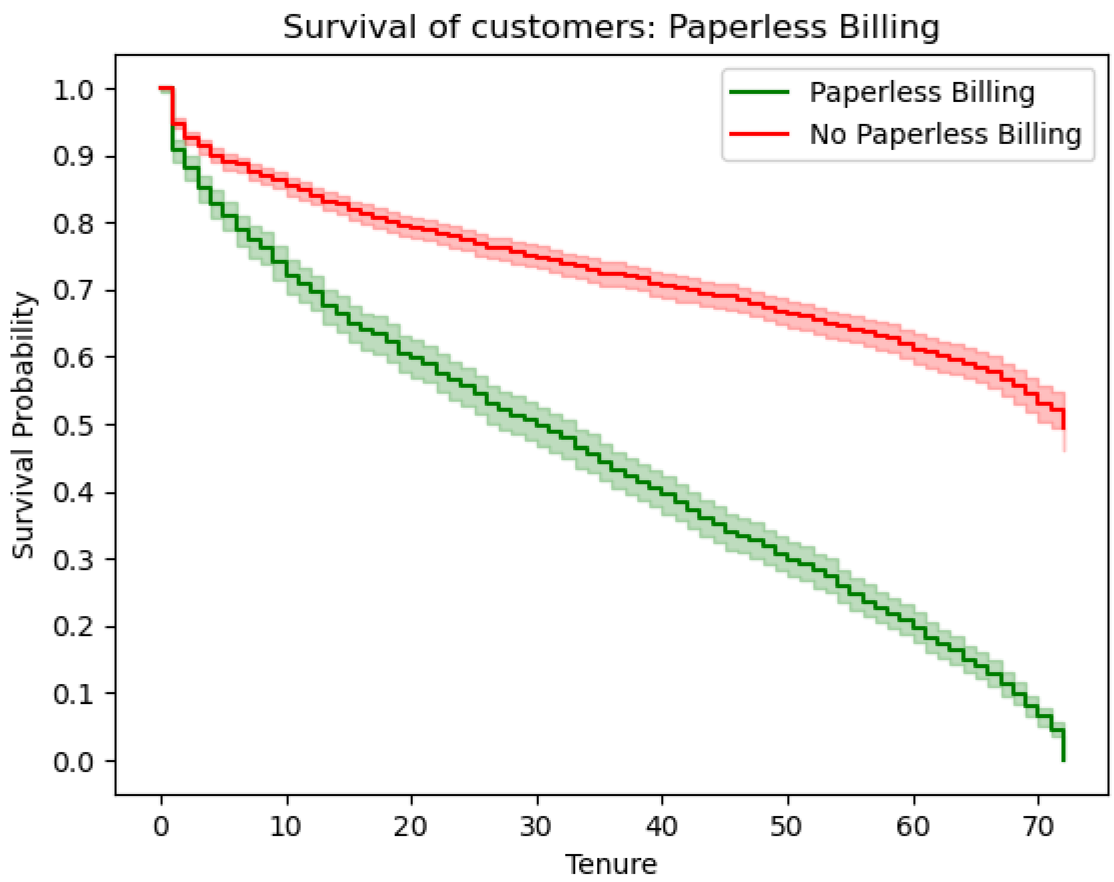
In [93]:

```
PaperlessBilling = (survivaldata['PaperlessBilling_Yes'] == 1)
no_PaperlessBilling = (survivaldata['PaperlessBilling_Yes'] == 0)

plt.figure()
ax = plt.subplot(1,1,1)

kmf.fit(timevar[PaperlessBilling],event_observed = eventvar[PaperlessBilling],label = "Paperless Billing")
plot1 = kmf.plot(ax = ax,  color='green')

kmf.fit(timevar[no_PhoneService],event_observed = eventvar[no_PhoneService],label = "No Paperless Billing")
```

```python
plot2 = kmf.plot(ax = plot1,  color='red')

plt.title('Survival of customers: Paperless Billing')
plt.xlabel('Tenure')
plt.ylabel('Survival Probability')
plt.yticks(np.linspace(0,1,11))
groups = logrank_test(timevar[PaperlessBilling], timevar[no_PaperlessBilling], event_observed_A=eventvar[PaperlessBilling], ev
groups.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	601.95	<0.005	439.16



Survival Regression

In [142…

```
def datapreparation(filepath):

    df = pd.read_csv(filepath)
    df.drop(["customerID"], inplace = True, axis = 1)

    df.TotalCharges = df.TotalCharges.replace(" ",np.nan)
    #df.TotalCharges.fillna(0, inplace = True)
    df['TotalCharges'] = df['TotalCharges'].fillna(0)
    df.TotalCharges = df.TotalCharges.astype(float)
```

```python
    cols1 = ['SeniorCitizen','Partner', 'Dependents', 'PaperlessBilling', 'Churn', 'PhoneService']
    for col in cols1:
        df[col] = df[col].apply(lambda x: 0 if x == "No" else 1)

    df.gender = df.gender.apply(lambda x: 0 if x == "Male" else 1)
    df.MultipleLines = df.MultipleLines.map({'No phone service': 0, 'No': 0, 'Yes': 1})

    cols2 = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']
    for col in cols2:
        df[col] = df[col].map({'No internet service': 0, 'No': 0, 'Yes': 1})

    df = pd.get_dummies(df, columns=['InternetService', 'Contract', 'PaymentMethod'], drop_first=True)

    return df
```

In [143…

```python
df2 = datapreparation("Telco_churn.csv")
#df2.head()
```

In [144…

```python
# Convert True/False values to 1/0 in the DataFrame
bool_cols = df2.select_dtypes(include=['bool']).columns
df2[bool_cols] = df2[bool_cols].astype(int)
```

In [145…

```python
df2.head()
```

Out[145...

	Unnamed: 0	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	...	Month
0	0	1	1	1	1	1	1	NaN	NaN	NaN	...	
1	1	0	1	1	1	34	1	NaN	NaN	NaN	...	
2	2	0	1	1	1	2	1	NaN	NaN	NaN	...	
3	3	0	1	1	1	45	1	NaN	NaN	NaN	...	
4	4	1	1	1	1	2	1	NaN	NaN	NaN	...	

5 rows × 25 columns

In [149...

```
# Drop rows with NaNs
df2 = df2.dropna()
df2.head()
```

Out[149...

	Unnamed: 0	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	...	Mc
3000	0	0	1	0	0	6	1	0.0	0.0	0.0	...	
3001	1	0	1	0	0	19	1	0.0	0.0	0.0	...	
3002	2	1	1	1	1	69	0	0.0	1.0	0.0	...	
3003	3	0	1	1	1	11	1	1.0	0.0	0.0	...	
3004	4	1	1	1	0	64	1	1.0	0.0	1.0	...	

5 rows × 25 columns

Survival Regression Ananlysis using Cox Proportional Hazard model

In [156…

```
cph = CoxPHFitter()
cph.fit(df2, duration_col='tenure', event_col='Churn')

cph.print_summary()
```

model	lifelines.CoxPHFitter
duration col	'tenure'
event col	'Churn'
baseline estimation	breslow
number of observations	2043
number of events observed	556
partial log-likelihood	-3115.66
time fit was run	2025-02-19 18:56:55 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
Unnamed: 0	0.00	1.00	0.00	-0.00	0.00	1.00	1.00	0.00	0.78	0.44	1.19
gender	0.01	1.01	0.09	-0.15	0.18	0.86	1.20	0.00	0.16	0.87	0.20
Partner	-0.19	0.83	0.10	-0.38	0.00	0.68	1.00	0.00	-1.93	0.05	4.21
Dependents	-0.17	0.84	0.12	-0.41	0.07	0.66	1.07	0.00	-1.38	0.17	2.58
PhoneService	-0.31	0.73	0.87	-2.01	1.38	0.13	3.98	0.00	-0.36	0.72	0.48
MultipleLines	-0.22	0.80	0.24	-0.68	0.24	0.51	1.27	0.00	-0.94	0.35	1.53
OnlineSecurity	-0.43	0.65	0.25	-0.91	0.06	0.40	1.06	0.00	-1.72	0.09	3.54
OnlineBackup	-0.33	0.72	0.23	-0.78	0.12	0.46	1.12	0.00	-1.45	0.15	2.77
DeviceProtection	-0.13	0.88	0.23	-0.58	0.33	0.56	1.39	0.00	-0.55	0.58	0.78
TechSupport	-0.20	0.82	0.24	-0.67	0.27	0.51	1.32	0.00	-0.83	0.41	1.29
StreamingTV	-0.37	0.69	0.43	-1.21	0.47	0.30	1.61	0.00	-0.86	0.39	1.35
StreamingMovies	-0.28	0.76	0.43	-1.12	0.55	0.33	1.74	0.00	-0.66	0.51	0.97

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
PaperlessBilling	0.24	1.28	0.10	0.04	0.45	1.04	1.56	0.00	2.35	0.02	5.72
MonthlyCharges	0.07	1.07	0.04	-0.01	0.15	0.99	1.17	0.00	1.70	0.09	3.49
TotalCharges	-0.00	1.00	0.00	-0.00	-0.00	1.00	1.00	0.00	-21.15	<0.005	327.52
InternetService_Fiber optic	-0.35	0.71	1.05	-2.40	1.71	0.09	5.53	0.00	-0.33	0.74	0.43
InternetService_No	-0.26	0.77	1.08	-2.38	1.86	0.09	6.43	0.00	-0.24	0.81	0.30
Contract_One year	-1.23	0.29	0.18	-1.57	-0.88	0.21	0.41	0.00	-7.00	<0.005	38.47
Contract_Two year	-3.50	0.03	0.35	-4.18	-2.82	0.02	0.06	0.00	-10.07	<0.005	76.87
PaymentMethod_Credit card (automatic)	-0.06	0.94	0.16	-0.38	0.26	0.68	1.30	0.00	-0.36	0.72	0.48
PaymentMethod_Electronic check	0.23	1.26	0.14	-0.03	0.50	0.97	1.65	0.00	1.72	0.09	3.55
PaymentMethod_Mailed check	0.40	1.49	0.16	0.08	0.71	1.09	2.03	0.00	2.48	0.01	6.27

Concordance	0.93
Partial AIC	6275.32
log-likelihood ratio test	1690.85 on 22 df
-log2(p) of ll-ratio test	inf

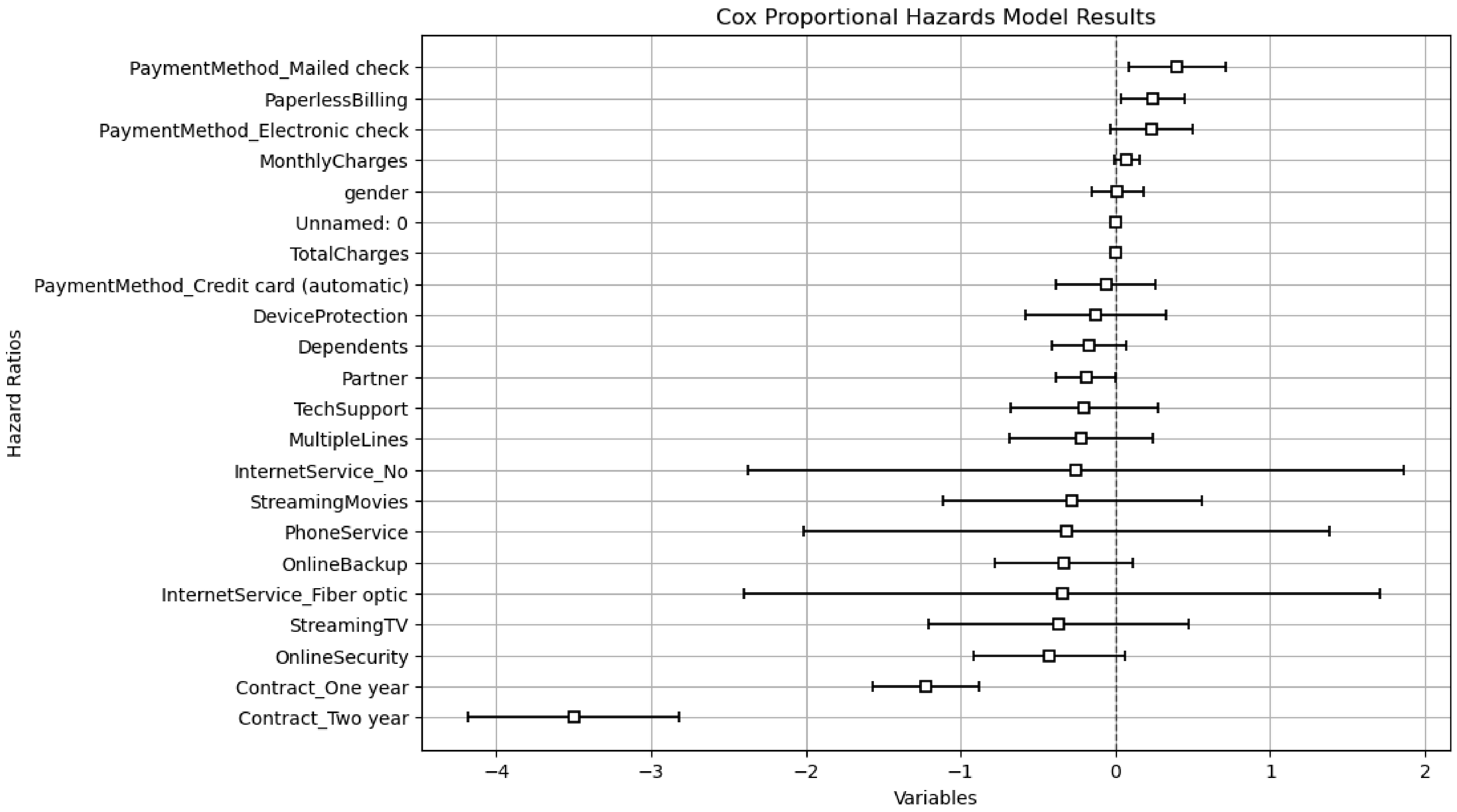
In [160…
```
#cph is your CoxPHFitter instance
#print(dir(cph))
```

In [159…
```
# To get the concordance index
c_index = cph.concordance_index_
print(f"Concordance Index: {c_index}")
```

```
Concordance Index: 0.9270444000831447
```

In [162…

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 7))

# Plot the CoxPHFitter results
cph.plot(ax=ax)
ax.set_title('Cox Proportional Hazards Model Results')
ax.set_xlabel('Variables')
ax.set_ylabel('Hazard Ratios')
ax.grid(True)
plt.show()
```



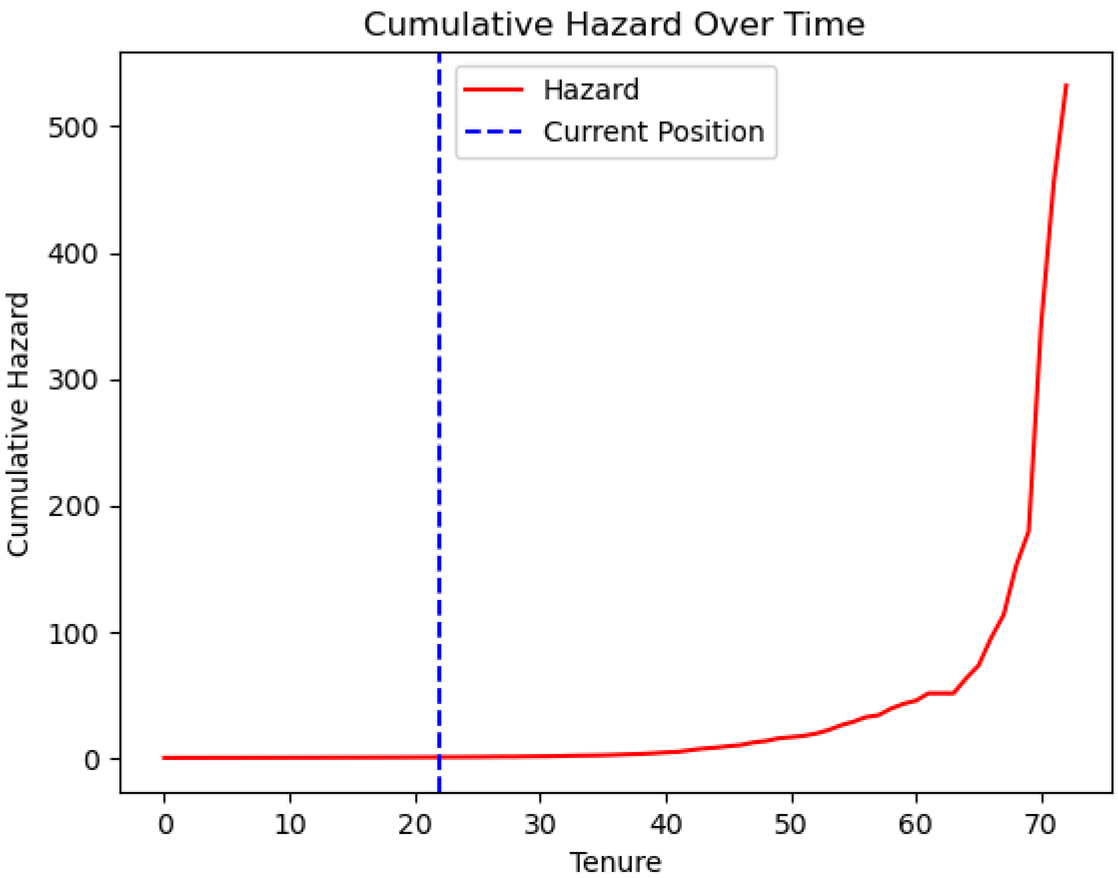
In [163…

```
test_id = df2.sample(1)
```

In [166…

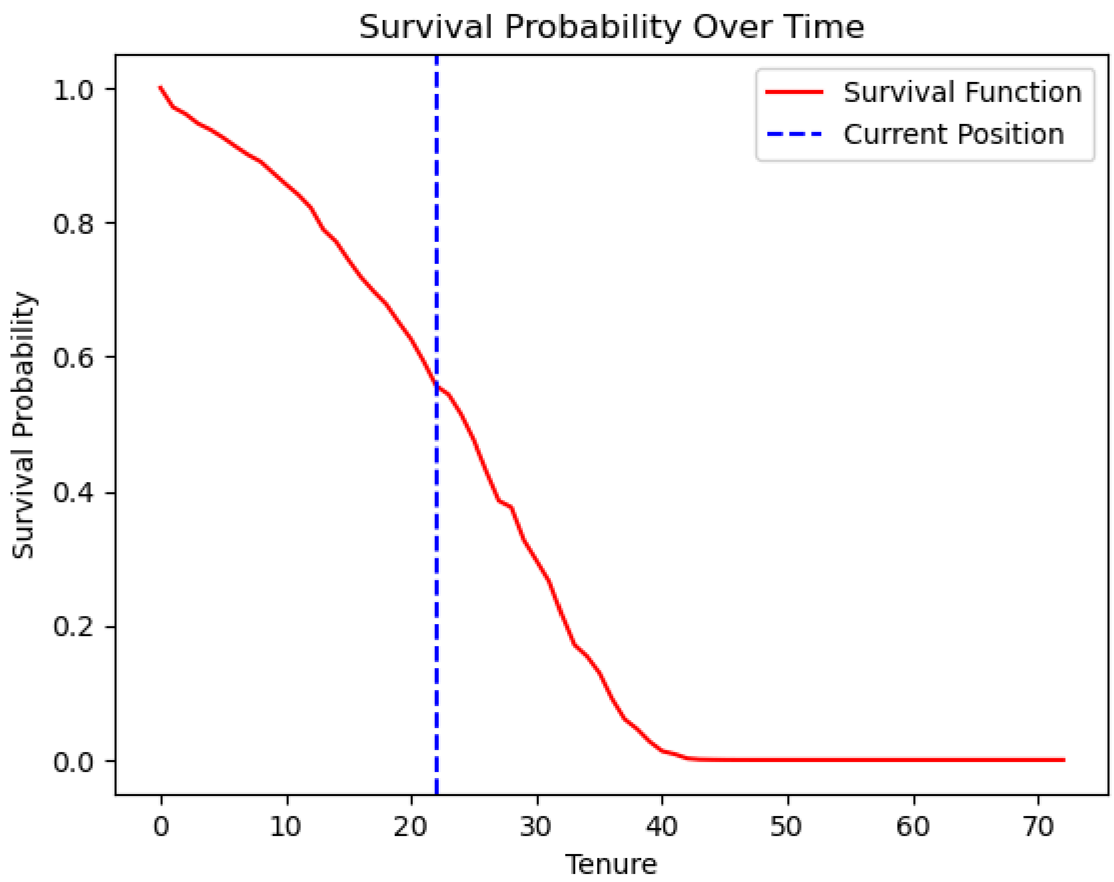
```
fig, ax = plt.subplots()
cph.predict_cumulative_hazard(test_id).plot(ax = ax, color = 'red')
plt.axvline(x=test_id.tenure.values[0], color = 'blue', linestyle='--')
plt.legend(labels=['Hazard','Current Position'])
ax.set_xlabel('Tenure', size = 10)
```

```python
ax.set_ylabel('Cumulative Hazard', size = 10)
ax.set_title('Cumulative Hazard Over Time');
```



In [167…

```python
fig, ax = plt.subplots()
cph.predict_survival_function(test_id).plot(ax = ax, color = 'red')
plt.axvline(x=test_id.tenure.values[0], color = 'blue', linestyle='--')
plt.legend(labels=['Survival Function','Current Position'])
ax.set_xlabel('Tenure', size = 10)
ax.set_ylabel('Survival Probability', size = 10)
ax.set_title('Survival Probability Over Time');
```



In [168…

```
# save the model
import pickle
pickle.dump(cph, open('survivemodel.pkl','wb'))
```

Customer Lifetime Value

To calculate the customer lifetime value (CLV), I multiply the monthly charges a customer pays to the telecom company by their expected lifetime.

I use the survival function to estimate a customer's expected lifetime. To be a bit conservative, I consider a customer to have churned when their survival probability drops to 10%.

In [169…

```
def LTV(info):
    life = cph.predict_survival_function(info).reset_index()
    life.columns = ['Tenure', 'Probability']
    max_life = life.Tenure[life.Probability > 0.1].max()

    LTV = max_life * info['MonthlyCharges'].values[0]
    return LTV
```

In [170…

```
print('LTV of a testid is:', LTV(test_id), 'dollars.')
```

```
LTV of a testid is: 3129.0 dollars.
```

Findings

The estimated customer lifetime value (CLV) for the given test customer is 3129.0 dollars. This means that the telecom company can expect to earn a total of 3129.0 dollars from this customer over their expected lifetime, assuming the customer will churn when their survival probability drops to 10%.