Telco Churn Survival Analysis

Author: Cennen Del Rosario
Date: September 15, 2020

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from lifelines import KaplanMeierFitter
from lifelines.statistics import (logrank_test,
                                  pairwise_logrank_test,
                                  multivariate_logrank_test,
                                  survival_difference_at_fixed_point_in_time_test)
import warnings
warnings.filterwarnings("ignore")
plt.style.use('seaborn')
```

1. Data Exploration

In [2]:
```
telco_df = pd.read_csv('telco_customer_churn.csv')
telco_df['churn'] = [1 if x == 'Yes' else 0 for x in telco_df['Churn']]
column_names = telco_df.columns
column_names
```

Out[2]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn', 'churn'],
      dtype='object')
```

In [3]:
```
telco_df.head(5)
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	Month-to-month	Yes	Electronic check
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	No	No	No	One year	No	Mailed check
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	Month-to-month	Yes	Mailed check
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	No	No	One year	No	Bank transfer (automatic)
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	Month-to-month	Yes	Electronic check

5 rows × 22 columns

In [4]:
```
for feature in column_names[1:]:
    if telco_df[feature].dtypes == object:
        print(feature, '\t', telco_df[feature].unique())
```

```
gender   ['Female' 'Male']
Partner          ['Yes' 'No']
Dependents       ['No' 'Yes']
PhoneService     ['No' 'Yes']
MultipleLines    ['No phone service' 'No' 'Yes']
InternetService          ['DSL' 'Fiber optic' 'No']
OnlineSecurity   ['No' 'Yes' 'No internet service']
OnlineBackup     ['Yes' 'No' 'No internet service']
DeviceProtection         ['No' 'Yes' 'No internet service']
TechSupport      ['No' 'Yes' 'No internet service']
StreamingTV      ['No' 'Yes' 'No internet service']
StreamingMovies          ['No' 'Yes' 'No internet service']
Contract         ['Month-to-month' 'One year' 'Two year']
PaperlessBilling         ['Yes' 'No']
PaymentMethod    ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
TotalCharges     ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
Churn    ['No' 'Yes']
```

2. Univariate Modelling

Kaplan-Meier Estimator

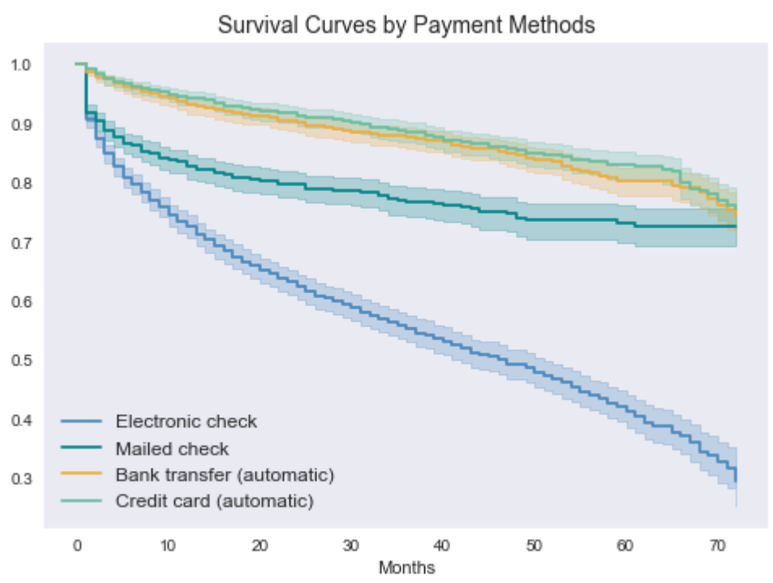
In [5]:
```
from lifelines import KaplanMeierFitter

time = telco_df['tenure']
event = telco_df['churn']

kmf = KaplanMeierFitter()
kmf.fit(time, event_observed=event)

kmf.plot(at_risk_counts=True)
plt.title('Kaplan-Meier Curve');
```

In [6]:
```
kmf.median_survival_time_
```

Out[6]: inf

Logrank Tests

In [7]:
```
colors = ['#4889BF', '#008489', '#ECAE3F', '#66BFA1', '#E65656']

ax = plt.subplot(1,1,1)
plt.style.use('seaborn-whitegrid')

kmf = KaplanMeierFitter()

for i, payment_method in enumerate(telco_df['PaymentMethod'].unique()):
    flag = telco_df['PaymentMethod'] == payment_method

    kmf.fit(time[flag], event_observed=event[flag], label=payment_method)
    kmf.plot(ax=ax, color=colors[i], xlabel = "Months")
    ax.grid(False)
    ax.legend(loc = 'lower left', fontsize = 12)

plt.title("Survival Curves by Payment Methods", fontsize=14);
```



In [8]:
```
credit_card_flag = telco_df['PaymentMethod'] == 'Credit card (automatic)'
bank_transfer_flag = telco_df['PaymentMethod'] == 'Bank transfer (automatic)'

results = logrank_test(time[credit_card_flag],
                       event[bank_transfer_flag],
                       event[credit_card_flag],
                       event[bank_transfer_flag])
results.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

	test_statistic	p	-log2(p)
0	1692.96	<0.005	inf

In [9]:
```
results = pairwise_logrank_test(event_durations = telco_df['tenure'],
                                event_observed = telco_df['churn'],
                                groups = telco_df['PaymentMethod'])
results.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	1
test_name	logrank_test

		test_statistic	p	-log2(p)
Bank transfer (automatic)	**Credit card (automatic)**	0.87	0.35	1.51
	Electronic check	510.04	<0.005	372.74
	Mailed check	51.07	<0.005	40.03
Credit card (automatic)	**Electronic check**	539.74	<0.005	394.21
	Mailed check	64.82	<0.005	50.11
Electronic check	**Mailed check**	152.46	<0.005	113.93

In [10]:
```
results = multivariate_logrank_test(telco_df['tenure'], telco_df['PaymentMethod'], telco_df['churn'])
results.print_summary()
```

t_0	-1
null_distribution	chi squared
degrees_of_freedom	3
test_name	multivariate_logrank_test

	test_statistic	p	-log2(p)
0	865.24	<0.005	619.58

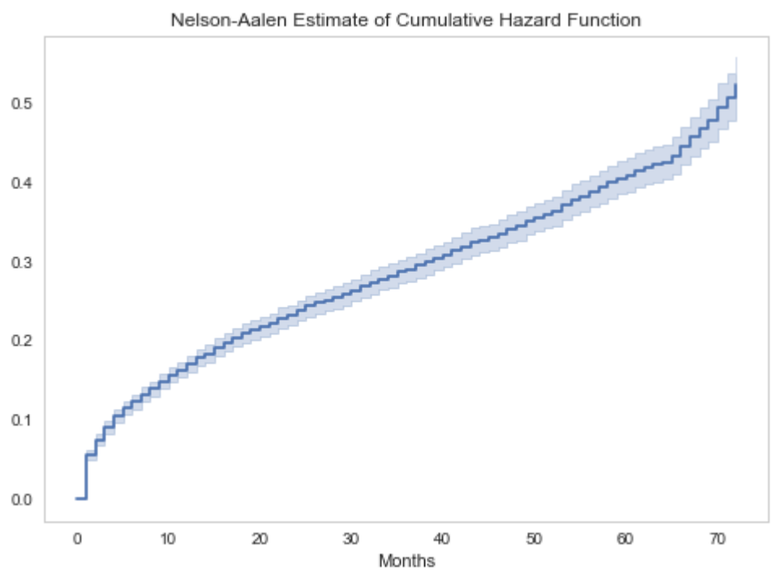
In [52]:
```
kmf_credit_card = KaplanMeierFitter(label="credit_card").fit(time[credit_card_flag], event[credit_card_flag])
kmf_bank_transfer = KaplanMeierFitter(label="bank_transfer").fit(time[bank_transfer_flag], event[bank_transfer_flag])

results = survival_difference_at_fixed_point_in_time_test(60, kmf_credit_card, kmf_bank_transfer)
results.print_summary()
```

null_distribution	chi squared
degrees_of_freedom	1
point_in_time	60
fitterA	<lifelines.KaplanMeierFitter:"credit_card", fi...
fitterB	<lifelines.KaplanMeierFitter:"bank_transfer", ...
test_name	survival_difference_at_fixed_point_in_time_test

	test_statistic	p	-log2(p)
0	2.57	0.11	3.20

Cumulative Hazard Function Estimator

In [12]:
```
from lifelines import NelsonAalenFitter
naf = NelsonAalenFitter()

naf.fit(time, event_observed=event)
naf.plot(title = "Nelson-Aalen Estimate of Cumulative Hazard Function",
         xlabel = "Months",  grid = False, legend=None);
```



3. Survival Regression

Data Preparation

In [13]:
```
for feature in column_names[1:]:
    if telco_df[feature].dtypes == object:
        print(feature, '\t', telco_df[feature].unique())
```

```
gender   ['Female' 'Male']
Partner          ['Yes' 'No']
Dependents       ['No' 'Yes']
PhoneService     ['No' 'Yes']
MultipleLines    ['No phone service' 'No' 'Yes']
InternetService          ['DSL' 'Fiber optic' 'No']
OnlineSecurity   ['No' 'Yes' 'No internet service']
OnlineBackup     ['Yes' 'No' 'No internet service']
DeviceProtection         ['No' 'Yes' 'No internet service']
TechSupport      ['No' 'Yes' 'No internet service']
StreamingTV      ['No' 'Yes' 'No internet service']
StreamingMovies          ['No' 'Yes' 'No internet service']
Contract         ['Month-to-month' 'One year' 'Two year']
PaperlessBilling         ['Yes' 'No']
PaymentMethod    ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
TotalCharges     ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
Churn    ['No' 'Yes']
```

In [14]:
```
pd.to_numeric(telco_df["TotalCharges"], errors='coerce').dropna()
```

Out[14]:
```
0        29.85
1      1889.50
2       108.15
3      1840.75
4       151.65
        ...
7038   1990.50
7039   7362.90
7040    346.45
7041    306.60
7042   6844.50
Name: TotalCharges, Length: 7032, dtype: float64
```

In [17]:
```
from sklearn.preprocessing import OneHotEncoder
import re
import numpy as np

telco_clean = telco_df
telco_clean['TotalCharges'] = [x if x != np.NaN else 0 for x in pd.to_numeric(telco_df["TotalCharges"], errors='coerce')]


def to_binary(col_name):
    if telco_clean[col_name].dtypes == object:
        if col_name == 'gender':
            telco_clean[col_name] = [0 if x == 'Male' else 1 for x in telco_clean[col_name]]
        else:
            telco_clean[col_name] = [0 if 'No' in x else 1 for x in telco_clean[col_name]]

# encode binary columns
non_binary_cols = ["customerID", "tenure", "InternetService", "Contract", "PaymentMethod", "Churn"]
binary_columns = list(set(column_names) - set(non_binary_cols))
for name in binary_columns:
    to_binary(name)

# encode columns with more than 2 categories
encoder = OneHotEncoder(drop="first", sparse=False)
multivalued_df = pd.DataFrame(encoder.fit_transform(telco_clean[["InternetService", "Contract", "PaymentMethod"]]))
multivalued_df.columns = [re.sub('[\s\(\)]','_', n) for n in encoder.get_feature_names()]

# replace them with encoded values
telco_clean["Tenure"] = telco_clean['tenure']
telco_clean["Churn"] = telco_clean['churn']
telco_clean = telco_clean.drop(["InternetService", "Contract", "PaymentMethod", "tenure", "churn"], axis=1)
telco_clean = pd.concat([multivalued_df,telco_clean], axis=1)
telco_clean.head(5)
```

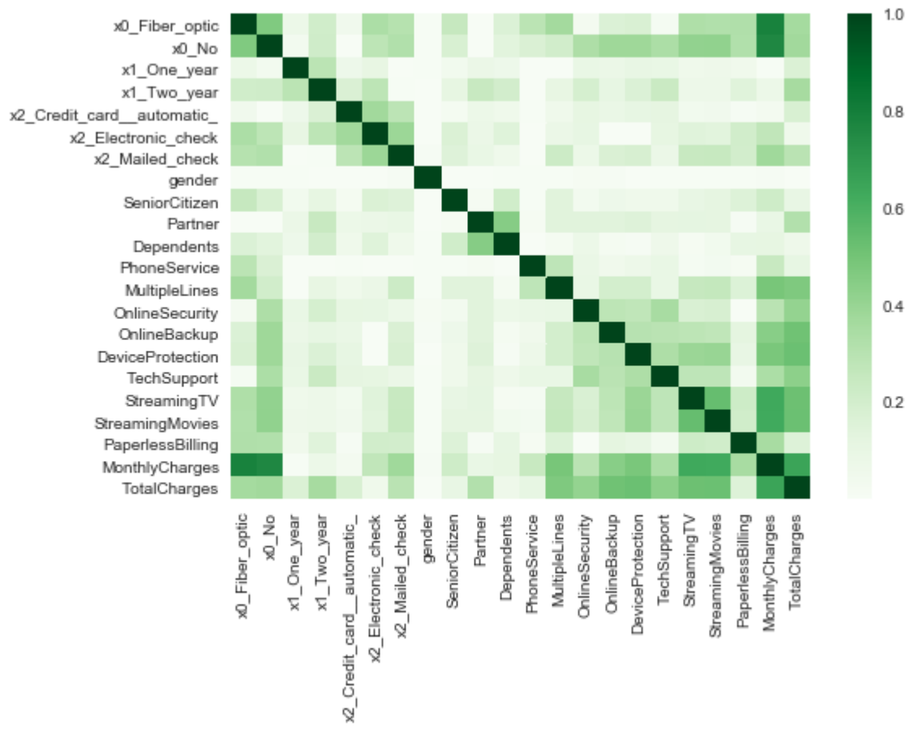
Out[17]:

	x0_Fiber_optic	x0_No	x1_One_year	x1_Two_year	x2_Credit_card__automatic_	x2_Electronic_check	x2_Mailed_check	customerID	gender	SeniorCitizen	...	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	7590-VHVEG	1	0	...	1	0	0	0
1	0.0	0.0	1.0	0.0	0.0	0.0	1.0	5575-GNVDE	0	0	...	0	1	0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3668-QPYBK	0	0	...	1	0	0	0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	7795-CFOCW	0	0	...	0	1	1	0
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	9237-HQITU	1	0	...	0	0	0	0

5 rows × 25 columns

Check Collinearity

In [19]:
```
import seaborn as sns

corr = telco_clean[telco_clean.columns[:-2]].corr()
corr = np.abs(corr)

# plot the heatmap
sns.heatmap(corr, cmap="Greens",
        xticklabels=corr.columns,
        yticklabels=corr.columns);
```



In [297]:
```
telco_clean.columns[:-2]
```

Out[297]:
```
Index(['x0_Fiber_optic', 'x0_No', 'x1_One_year', 'x1_Two_year',
       'x2_Credit_card__automatic_', 'x2_Electronic_check', 'x2_Mailed_check',
       'customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges'],
      dtype='object')
```

Fitting Cox PH Models

In [20]:
```
form1 = " + ".join(['gender', 'SeniorCitizen', 'Partner',
                    'Dependents', 'PhoneService', 'MultipleLines',
                    'TechSupport', 'StreamingTV', 'StreamingMovies',
                    'PaperlessBilling', 'MonthlyCharges',
                    'TotalCharges'])

form2 = " + ".join(['SeniorCitizen', 'Partner',
                    'Dependents', 'PhoneService', 'MultipleLines',
                    'TechSupport', 'StreamingTV', 'StreamingMovies',
                    'PaperlessBilling', 'MonthlyCharges',
                    'TotalCharges'])

form3 = " + ".join(['SeniorCitizen', 'Partner', 'Dependents',
                    'MultipleLines', 'TechSupport',
                    'PaperlessBilling', 'MonthlyCharges',
                    'TotalCharges'])
```

In [21]:
```
from lifelines import CoxPHFitter

cph = CoxPHFitter(penalizer=0.1)
cph.fit(telco_clean.fillna(0), duration_col='Tenure', event_col='Churn',
        formula = form3)
cph.print_summary()
```

model	lifelines.CoxPHFitter
duration col	'Tenure'
event col	'Churn'
penalizer	0.1
l1 ratio	0
baseline estimation	breslow
number of observations	7043
number of events observed	1869
partial log-likelihood	-14591.39
time fit was run	2020-09-21 07:37:41 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
SeniorCitizen	0.26	1.29	0.05	0.16	0.36	1.17	1.43	5.21	<0.005	22.37
Partner	-0.40	0.67	0.04	-0.48	-0.31	0.62	0.73	-9.21	<0.005	64.68
Dependents	-0.31	0.73	0.05	-0.41	-0.21	0.67	0.81	-6.13	<0.005	30.04
MultipleLines	-0.10	0.90	0.04	-0.19	-0.02	0.83	0.98	-2.45	0.01	6.14
TechSupport	-0.52	0.60	0.05	-0.61	-0.42	0.54	0.66	-10.58	<0.005	84.52
PaperlessBilling	0.42	1.53	0.04	0.34	0.51	1.40	1.67	9.63	<0.005	70.56
MonthlyCharges	0.02	1.02	0.00	0.02	0.02	1.02	1.02	22.46	<0.005	368.58
TotalCharges	-0.00	1.00	0.00	-0.00	-0.00	1.00	1.00	-33.91	<0.005	835.06

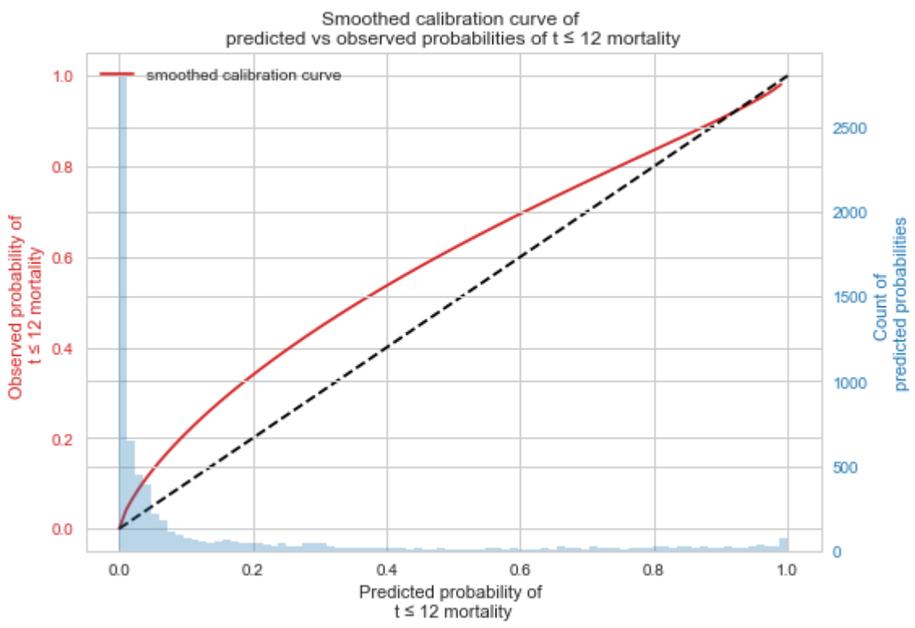
Concordance	0.88
Partial AIC	29198.78
log-likelihood ratio test	2123.30 on 8 df
-log2(p) of ll-ratio test	inf

Model Probability Calibration

In [22]:
```
x = telco_clean.fillna(0)
x[x['Tenure'] == 0] = 0.01
```

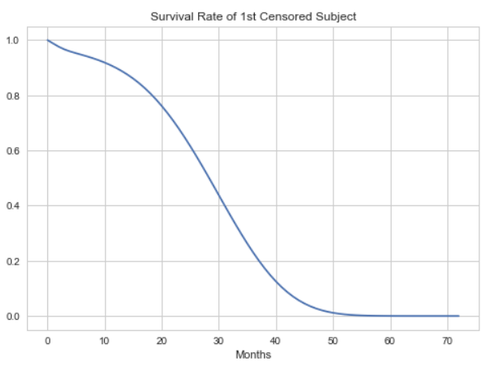
In [24]:
```
from lifelines.calibration import survival_probability_calibration

cph = CoxPHFitter(baseline_estimation_method="spline", n_baseline_knots=3)
cph.fit(x, duration_col='Tenure', event_col='Churn',
        formula = form3)

survival_probability_calibration(cph, x, t0=12);
```

```
ICI =  0.053278121098122716
E50 =  0.040031382327195O5
```



Cox PH Prediction

In [25]:
```
cph_new = cph.fit(x, duration_col='Tenure', event_col='Churn',
        formula = form3)

# filter down to just censored subjects to predict remaining survival
censored_subjects = x.loc[x['Churn'] == 0]
censored_subjects_last_obs = censored_subjects['Tenure']

# predict new survival function
ps = cph_new.predict_survival_function(censored_subjects,
                                       conditional_after=censored_subjects_last_obs)
ps[0].plot(xlabel="Months", title="Survival Rate of 1st Censored Subject")

# predict median remaining life
pred_median_time = cph_new.predict_median(censored_subjects,
                                       conditional_after=censored_subjects_last_obs)
```



In [27]:
```
x.loc[x['Churn'] == 0].index[0:6]
display = x.iloc[0:10,-2:]
display['Remaining Life'] = pred_median_time[0:6]
display['Customer Lifetime'] = display['Tenure'] + display['Remaining Life']
display
```

Out[27]:

	Churn	Tenure	Remaining Life	Customer Lifetime
0	0.0	1.0	29.0	30.0
1	0.0	34.0	15.0	49.0
2	1.0	2.0	NaN	NaN
3	0.0	45.0	24.0	69.0
4	1.0	2.0	NaN	NaN
5	1.0	8.0	NaN	NaN
6	0.0	22.0	8.0	30.0
7	0.0	10.0	26.0	36.0
8	1.0	28.0	NaN	NaN
9	0.0	62.0	41.0	103.0