Fitting the Cox Proportional Hazards model

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

Hazard function and hazard rate

Hazard function $h(t)$: describes the probability that event happens at some time, given survival up to that time.

Hazard rate: the instantaneous rate of event occurring

$$h(t) = -\frac{d}{dt} log S(t)$$

The **hazard function** $h(t)$ and the **survival function** $S(t)$ can be derived from each other.

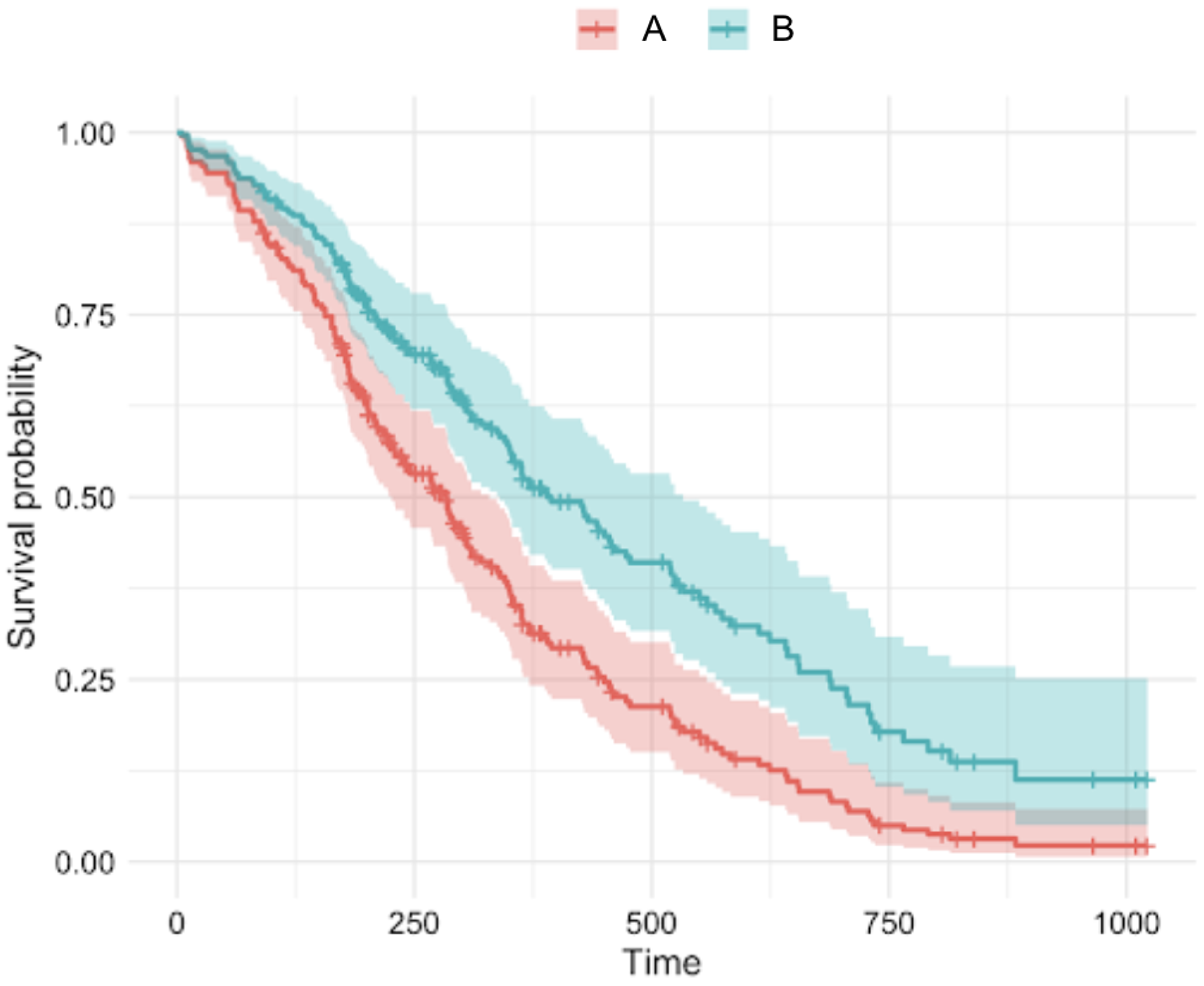
The proportional hazards assumption

The proportional hazards assumption: all individuals' hazards are proportional to one another.

In the case of individual A and individual B:

$$h_A(t) = ch_B(t)$$

1. There is a **baseline hazard function** and other hazards are specified with **scaling factors.**
2. The **relative survival impact** associated with **a variable** does not change with time (time-invariant).



The Cox Proportional Hazards model

Based on the proportional hazards assumption:

$$h(t|x) = b_0(t)exp\left(\sum_{i=1}^{n} b_i(x_i - \overline{x_i}\right)$$

$b_0(t)$: population-level baseline hazard function that changes with time.

$exp\left(\sum_{i=1}^{n} b_i(x_i - \overline{x_i}\right)$: the linear relationship between covariates and the log of hazard, does NOT change with time.

- The Cox Proportional Hazards (Cox PH) model is **a regression model** that regresses **covariates** on time-to-event/duration.

Data requirement for Cox PH model

- **Durations:** the lifetime/duration of the individuals.
- **Events:** whether the event has been observed (1=Yes, 0=No, censored).
  - If not supplied, the model assumes no subjects are censored.
- **Covariates:** continuous or one-hot encoded categorical variables for the regression.

Fitting the Cox PH model

1. Import and instantiate the `CoxPHFitter` class

```
from lifelines import CoxPHFitter
coxph = CoxPHFitter()
```

2. Call `.fit()` to fit the estimator to the data

```
coxph.fit(df, duration_col, event_col)
```

3. Access other properties to check model summary, covariate, coefficients, predict, plot, etc.

```
coxph.summary()
coxph.predict()
```

Example Cox PH model

- DataFrame: `mortgage_df`
- Covariates:
    - `house`
    - `principal`
    - `interest`
    - `property_tax`
    - `credit_score`
- Other columns: `duration` , `paid_off`

```
from lifelines import CoxPHFitter
coxph = CoxPHFitter()
coxph.fit(df=mortgage_df,
          duration_col="duration",
          event_col="paid_off")
```

Custom model

Filter the `DataFrame` :

```
new_df = mortgage_df.loc[:,
          mortgage_df.columns!="house"]
coxph.fit(df=new_df,
          duration_col="duration",
          event_col="paid_off")
```

Use the `formula` parameter:

```
coxph.fit(df=mortgage_df,
          duration_col="duration",
          event_col="paid_off",
          formula="principal + interest
          + property_tax + credit_score")
```

- More convenient and clearer, but doesn't scale to large number of covariates.

Interpret coefficients

```
print(coxph.summary)
```

```
<lifelines.CoxPHFitter: fitted with 1808 observations, 340 censored>
                        coef  exp(coef)  se(coef)      z       p
covariate house        -0.38       0.68      0.19. -1.98    0.05
          principal    -0.06       0.94      0.02  -2.61    0.01
          interest      0.31       1.37      0.31   1.02    0.31
          property_tax -0.15       0.86      0.21  -0.71    0.48
          credit_score -0.43       0.65      0.38  -1.14.   0.26
```

- **Hazard ratio:** e^{coef}
  - A one unit increase in `interest` from its median value -> the hazards change by the a factor of $e^{0.31} = 1.37$, which is a 37% increase compared to the baseline hazards.

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

Interpreting the Cox PH model

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

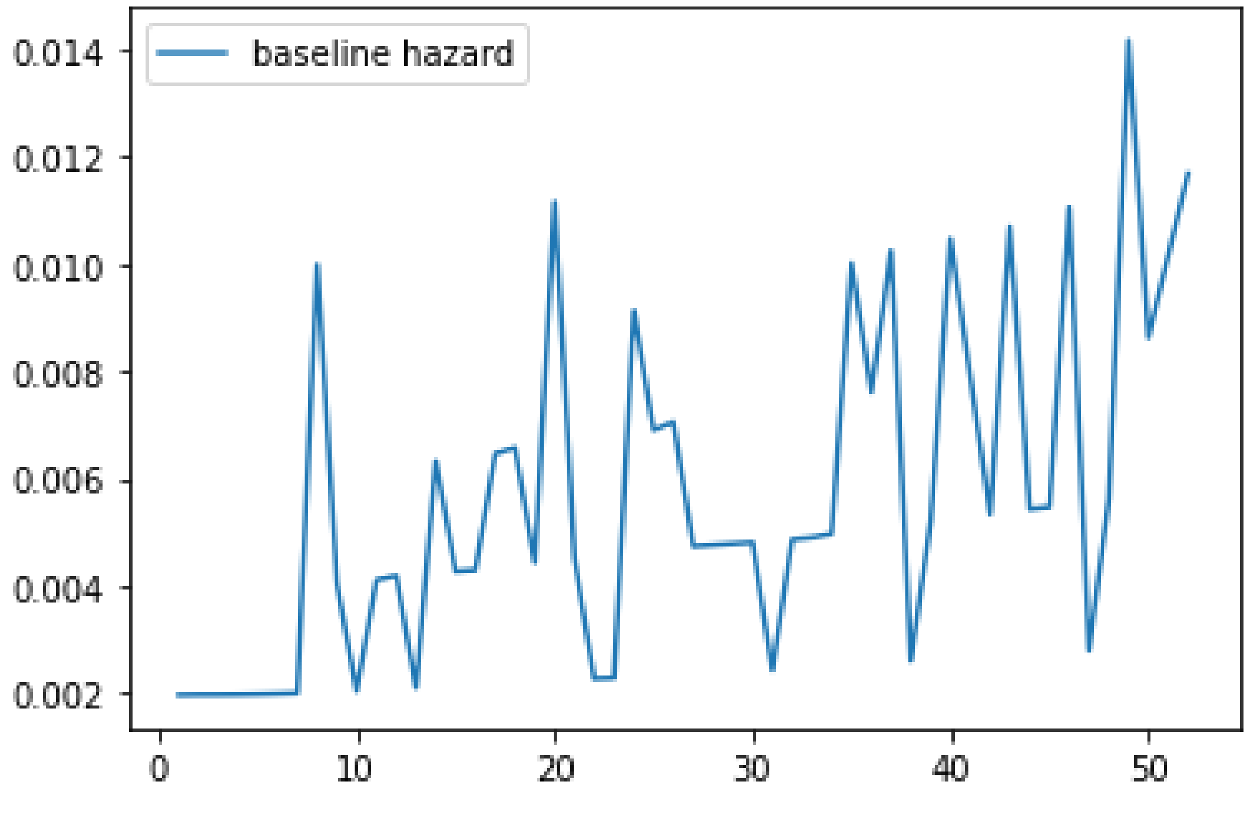
The baseline hazards

- **Hazard ratio:** how much hazard increases or decreases relative to *baseline* hazards.
- **Baseline hazards:** the risk for individuals at the baseline levels of covariates.
  - *Baseline* $\neq$ setting covariates to 0
  - *Baseline* means setting covariates to their averages (median for `lifelines`)

The baseline functions

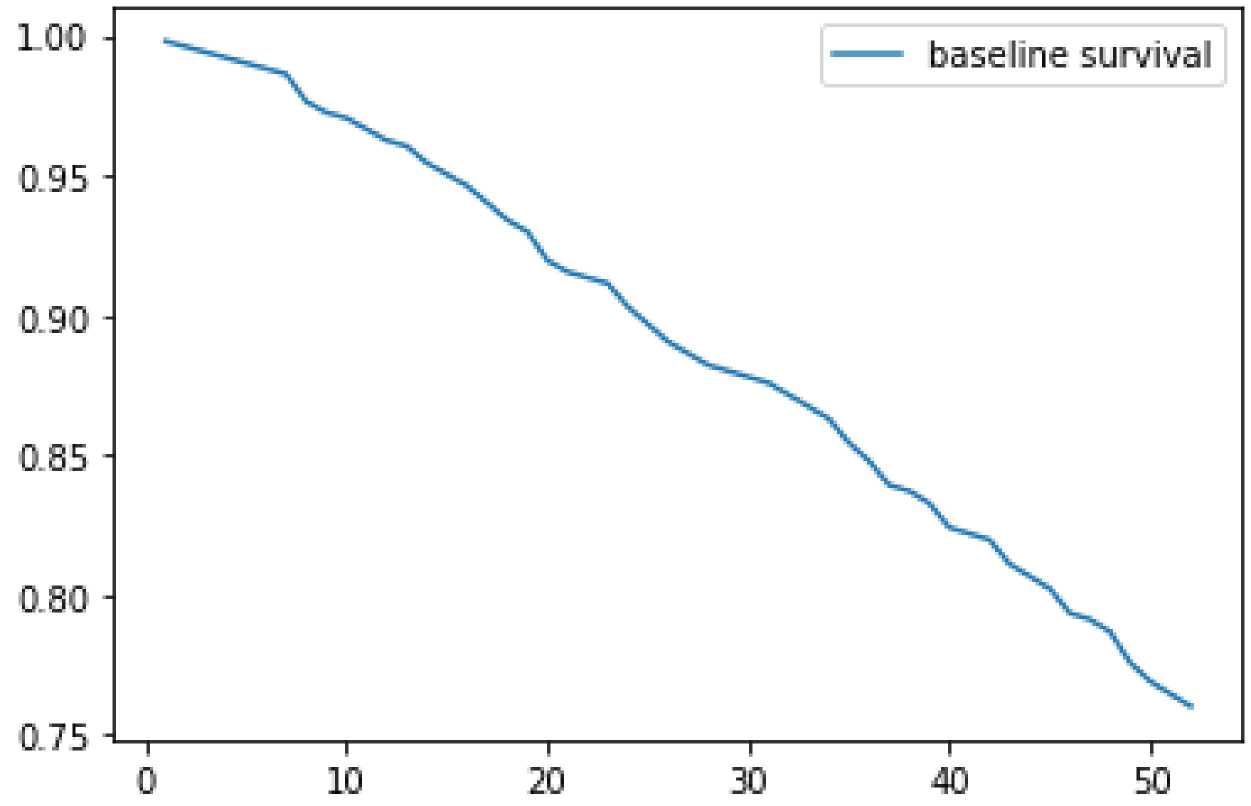
Baseline hazard function

```
model.baseline_hazard_.plot()
plt.show()
```



Baseline survival function

```
model.baseline_survival_.plot()
plt.show()
```



Interpret the hazard ratio

- **Hazard ratio:** e^{coef}, how much hazard changes relative to the *average* individual when covariates change.

	Calculation	Example
Coefficient	x	0.405
Hazard ratio	e^x	$e^{0.405} = 1.5$
Hazards interpretation	$e^x - 1$	$1.5 - 1 = 0.5$ -> 50% increase in hazards
Survival time interpretation	$\frac{1}{e^x} - 1$	$\frac{1}{1.5} - 1 = 0.67 - 1 = -0.23$ -> 23% decrease in survival time

Visualize the hazard ratio

`.plot_partial_effects_on_outcome()`

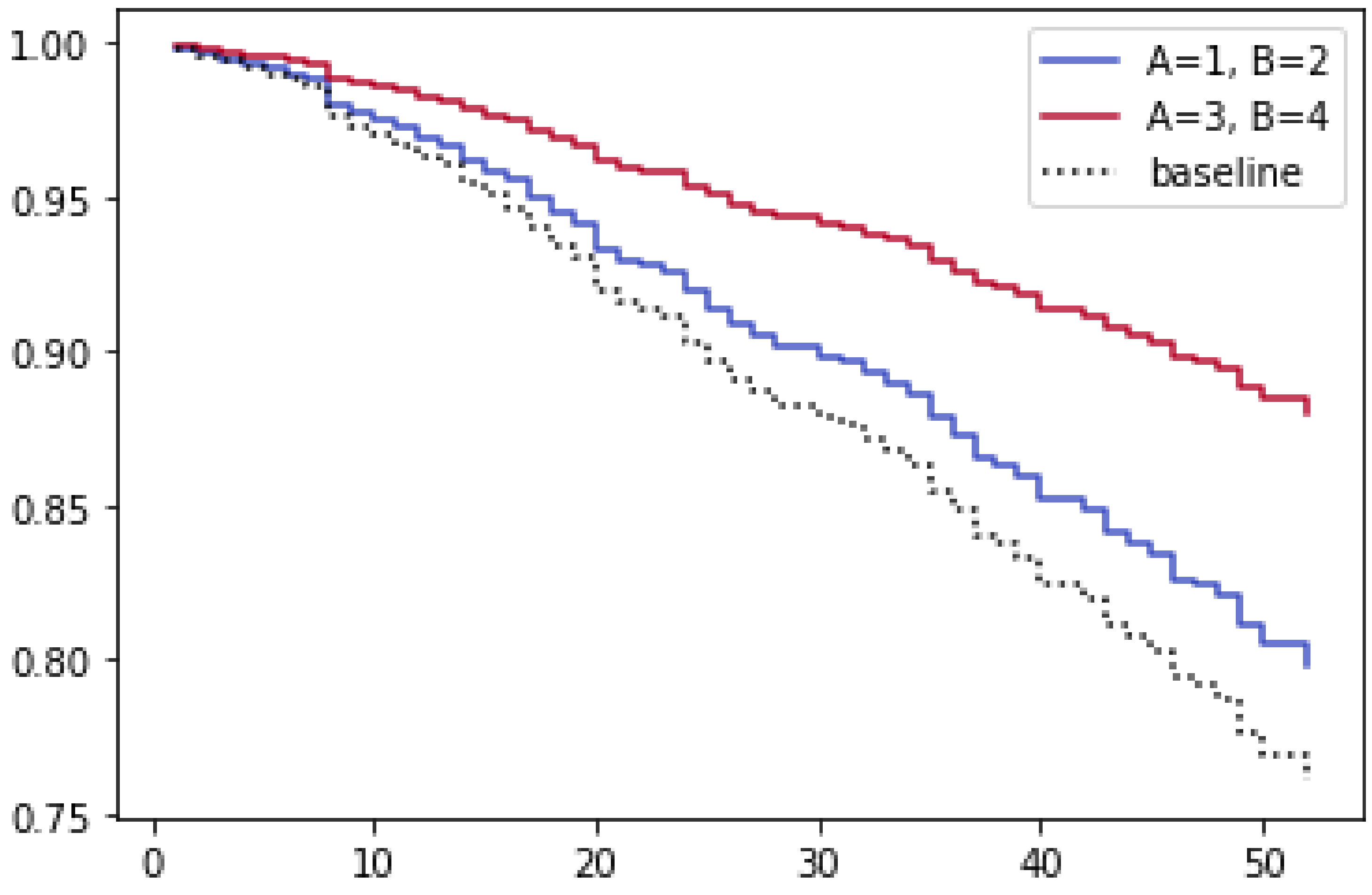
- `covariates` (string or list): name(s) of the covariate(s) in the original dataset that we wish to vary.
  - If there are multiple covariates, pass them in as **a list**.
- `values` (1d or 2d iterable): values we wish the covariate to take on.
  - If there are multiple covariates, pass the values as **pairs/tuples of values**.

Visualize the hazard ratio

The model has covariates `A` , `B` , `C` , and we wish to vary

- `A` over 1, 2
- `B` over 3, 4

```
model.plot_partial_effects_on_outcome(
    covariates=["A","B"],
    values=[[1,2],
            [3,4]]
)
plt.show()
```



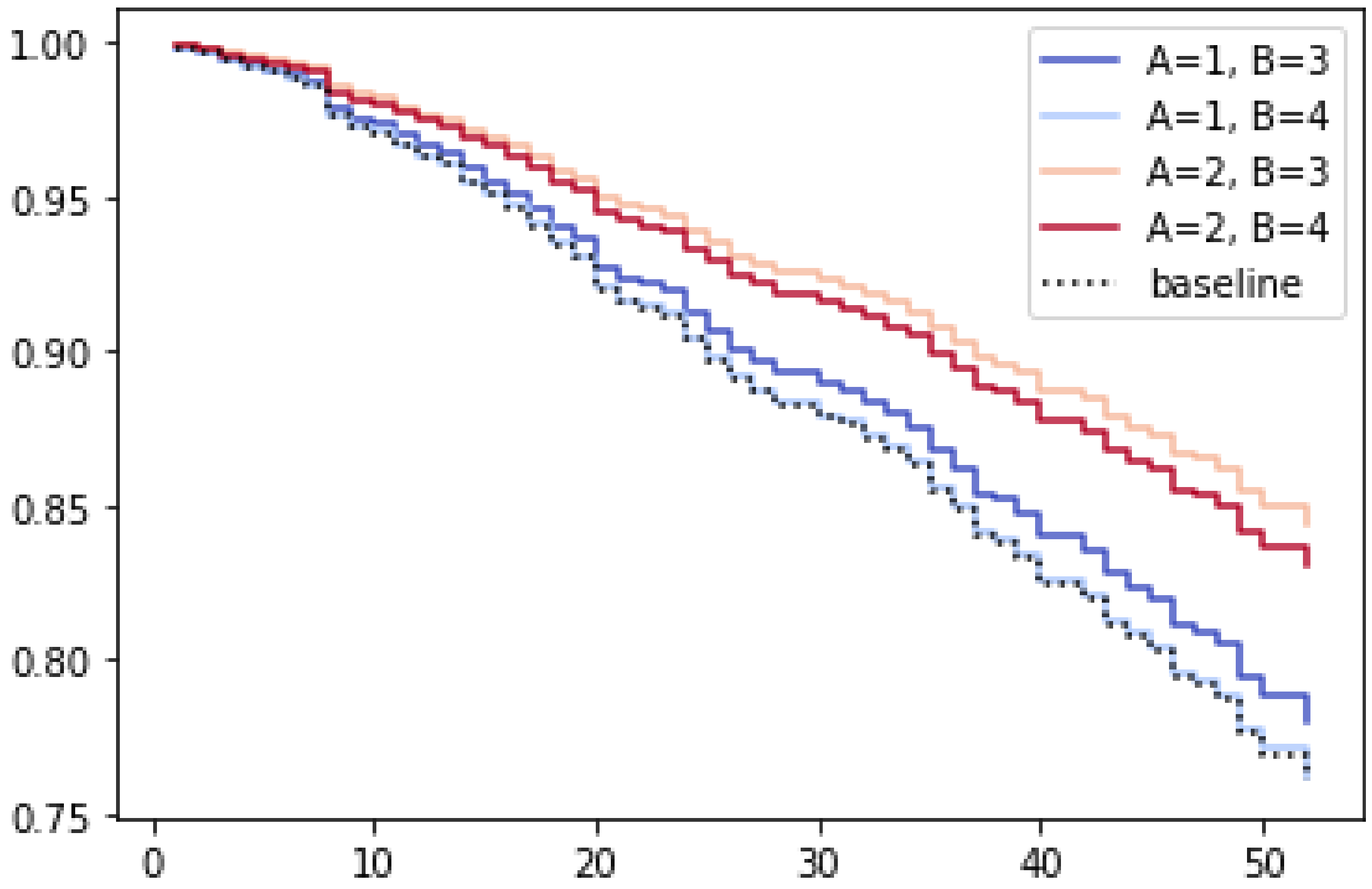
Wrong...

Visualize the hazard ratio

The model has covariates `A`, `B`, `C`, and we wish to vary

- `A` over 1, 2
- `B` over 3, 4

```
model.plot_partial_effects_on_outcome(
    covariates=["A","B"],
    values=[[1,3],
            [1,4],
            [2,3],
            [2,4]]
)
plt.show()
```



Correct!

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

The proportional hazards assumption

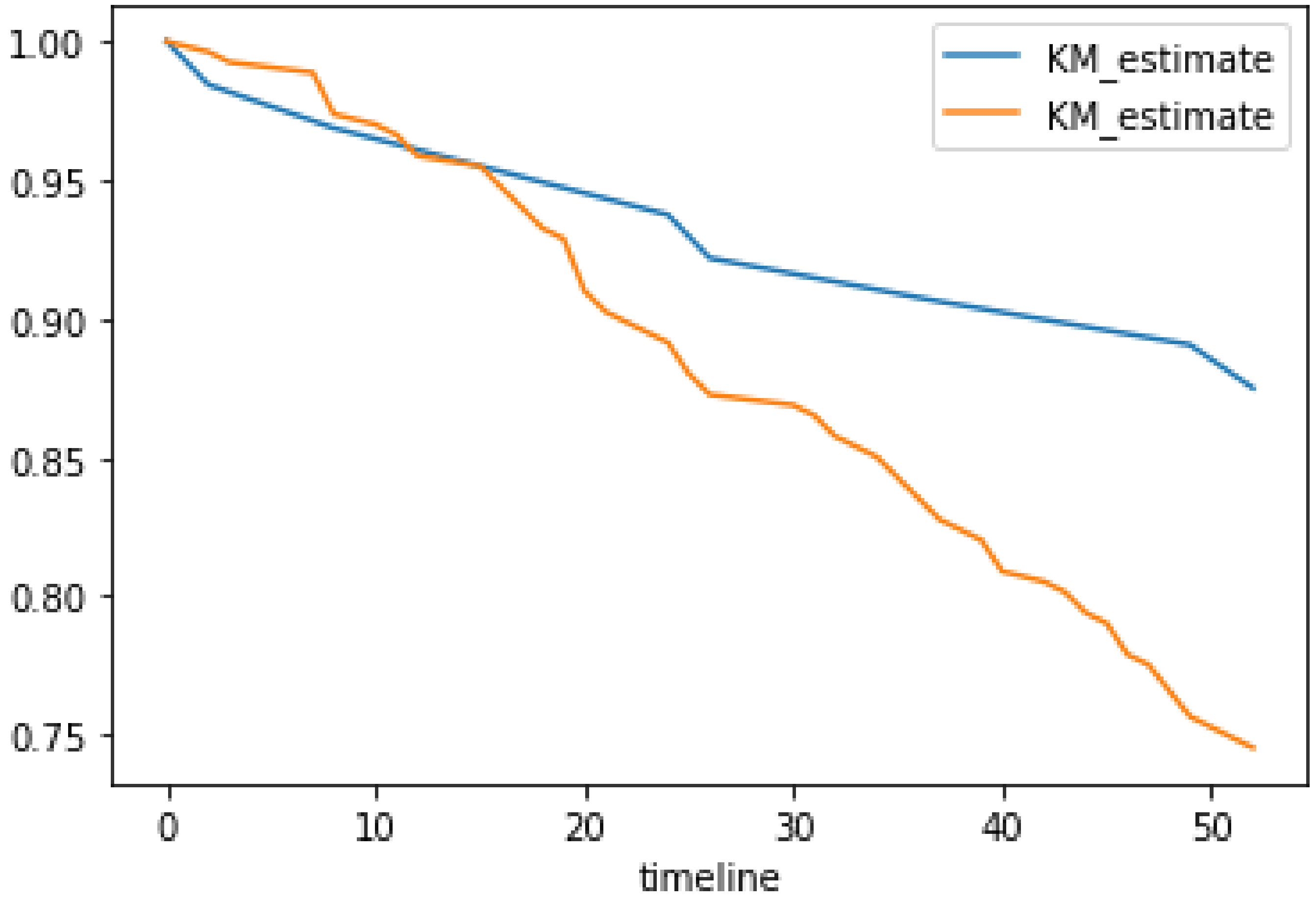
SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

Use the Kaplan-Meier curves

If the covariate only has a few values, inspect each group's Kaplan-Meier curve.

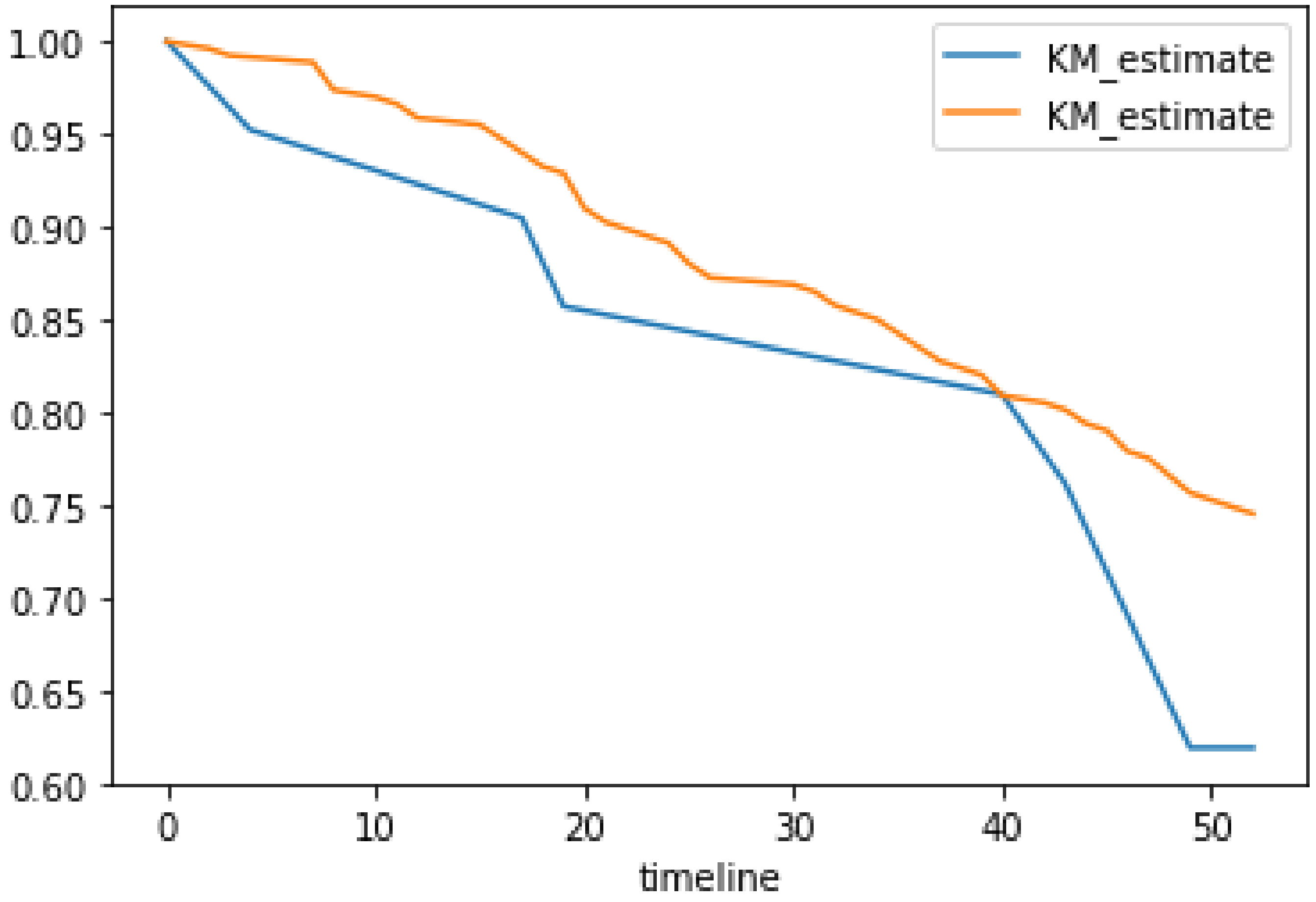
- Curves intersect: **fails** the proportional hazards assumption.



Use the Kaplan-Meier curves

If the covariate only has a few values, inspect each group's Kaplan-Meier curve.

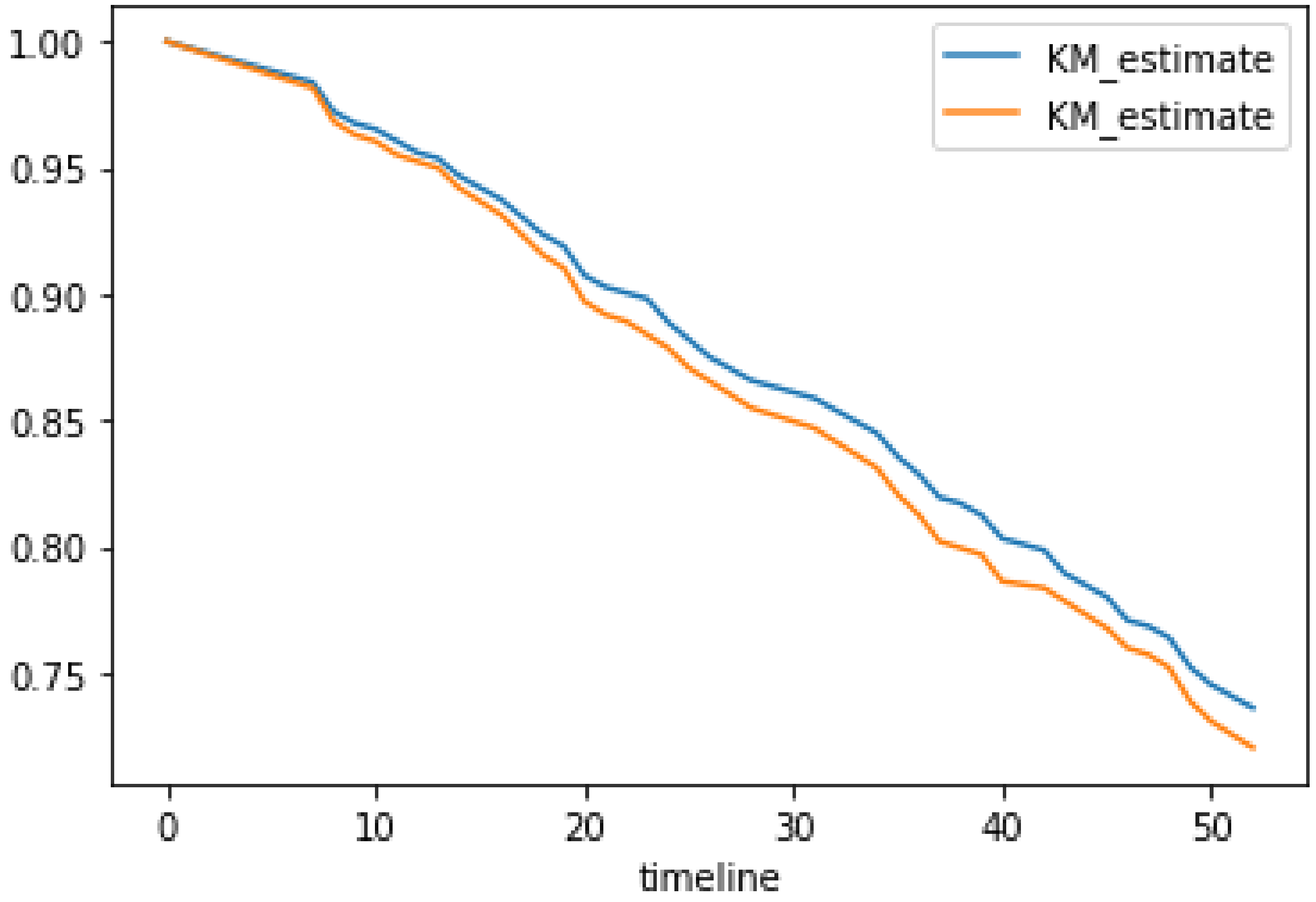
- Curves have different shapes: **fails** the proportional hazards assumption.



Use the Kaplan-Meier curves

If the covariate only has a few values, inspect each group's Kaplan-Meier curve.

- Curves have similar shapes and are parallel: **satisfies** the proportional hazards assumption.



.check_assumptions()

If the covariates are continuous, use the `.check_assumptions()` method.

- Parameters
  - `training_df` : the original DataFrame used in the call to fit the model.
  - `p_value_threshold` : the threshold to use to alert the user of violations (default: 0.01, recommended: 0.05).

.check_assumptions()

```
model.check_assumptions(training_df, p_value_threshold=0.05)
```

```
1. Variable 'A' failed the non-proportional test: p-value is 0.0007.
Advice 1: ...
Advice 2: ...
2. Variable 'B' failed the non-proportional test: p-value is 0.0063.
Advice 1: ...
Advice 2: ...
```

When the proportional hazards assumption fails

- Usually, it's a reasonable assumption and violations do not impact model performance significantly.
- If it fails, try other modeling frameworks, such as the Weibull AFT model, and compare their AIC scores.

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

Predicting with the Cox PH model

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

Predict median survival times

After calling `.fit()` to fit model to the data:

- `.predict_median()` : predicts the median lifetimes for subjects
    - If the survival curve does not cross 0.5, the median survival time is $\inf$.
- Parameters:
    - `X` : the DataFrame to predict with.
    - `conditional_after` : an array or list of values that represent how long subjects have already lived for.

Predict median survival times

```
model.predict_median(X, conditional_after)
```

```
0        inf
1       44.0
2       46.0
3        inf
4       48.0
        ...
500      inf
```

Predict the survival function

- `.predict_survival_function()` : predicts the survival function for subjects, given their covariates.
- Parameters:
  - `X` : the DataFrame to predict with.
  - `conditional_after` : an array or list of values that represent how long subjects have already lived for.

Predict the survival function

```
model.predict_survival_function(X, conditional_after)
```

```
           0         1         2         3         4    ...       500
1.0  0.997616  0.993695  0.994083  0.999045  0.997626   ...  0.998865
2.0  0.995230  0.987411  0.988183  0.998089  0.995250   ...  0.997728
3.0  0.992848  0.981162  0.982314  0.997133  0.992878   ...  0.996592
4.0  0.990468  0.974941  0.976468  0.996176  0.990507   ...  0.995455
5.0  0.988085  0.968739  0.970639  0.995216  0.986392   ...  0.993476
```

Why are survival predictions useful?

- Proactive failure prevention, forecasting models, etc.

Key steps

1. Preprocess the data and one-hot encode any categorical variables.
2. Split data into train and test (common split is 80% train and 20% test).
    - The proportions of censored data should be similar in both sets.
3. Fit the Cox PH model to train.

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

Congratulations!

SURVIVAL ANALYSIS IN PYTHON

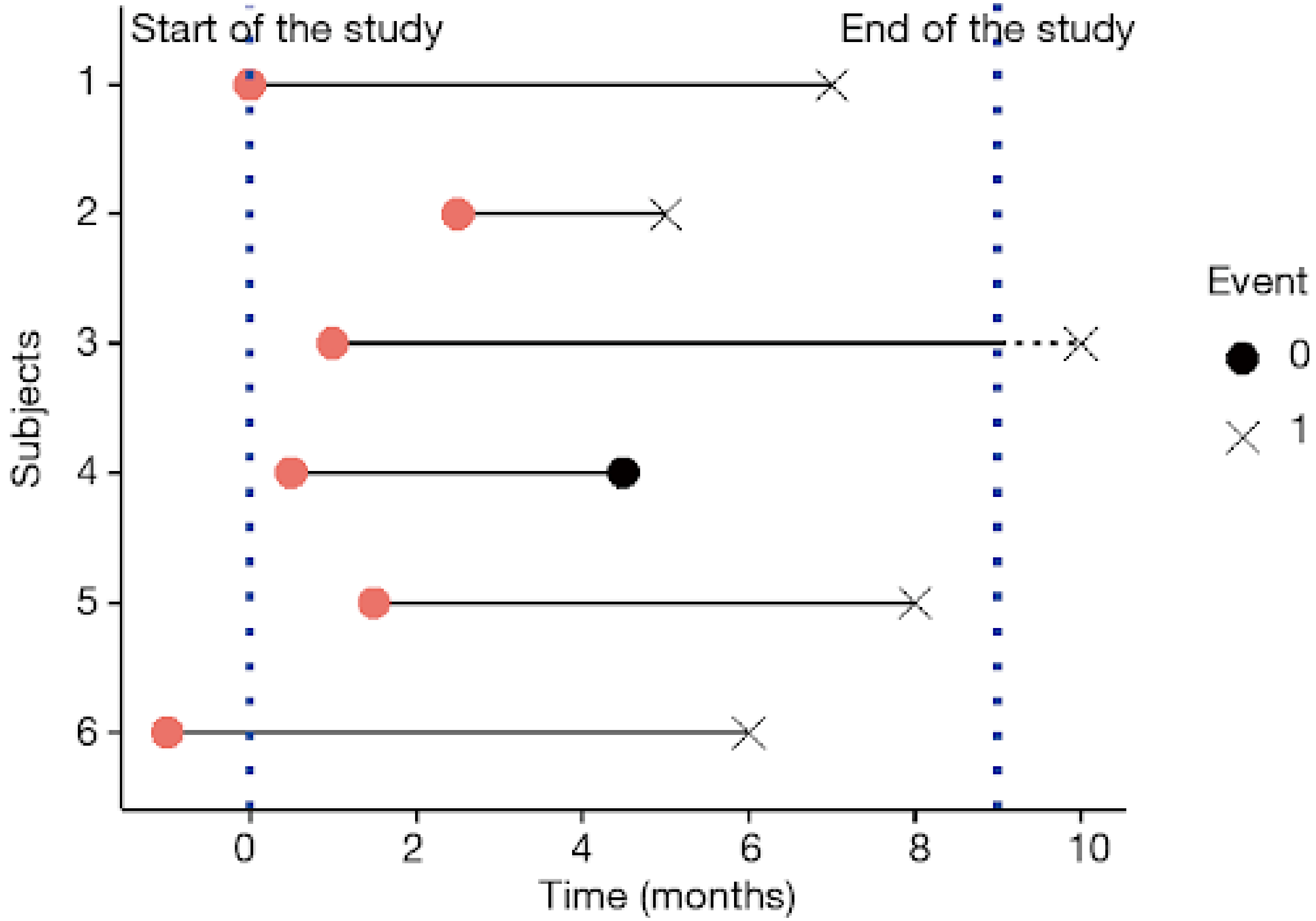
Shae Wang
Senior Data Scientist at Ripple

Why survival analysis?

Use cases

- Estimate time to event
- Measure how factors affect time to event

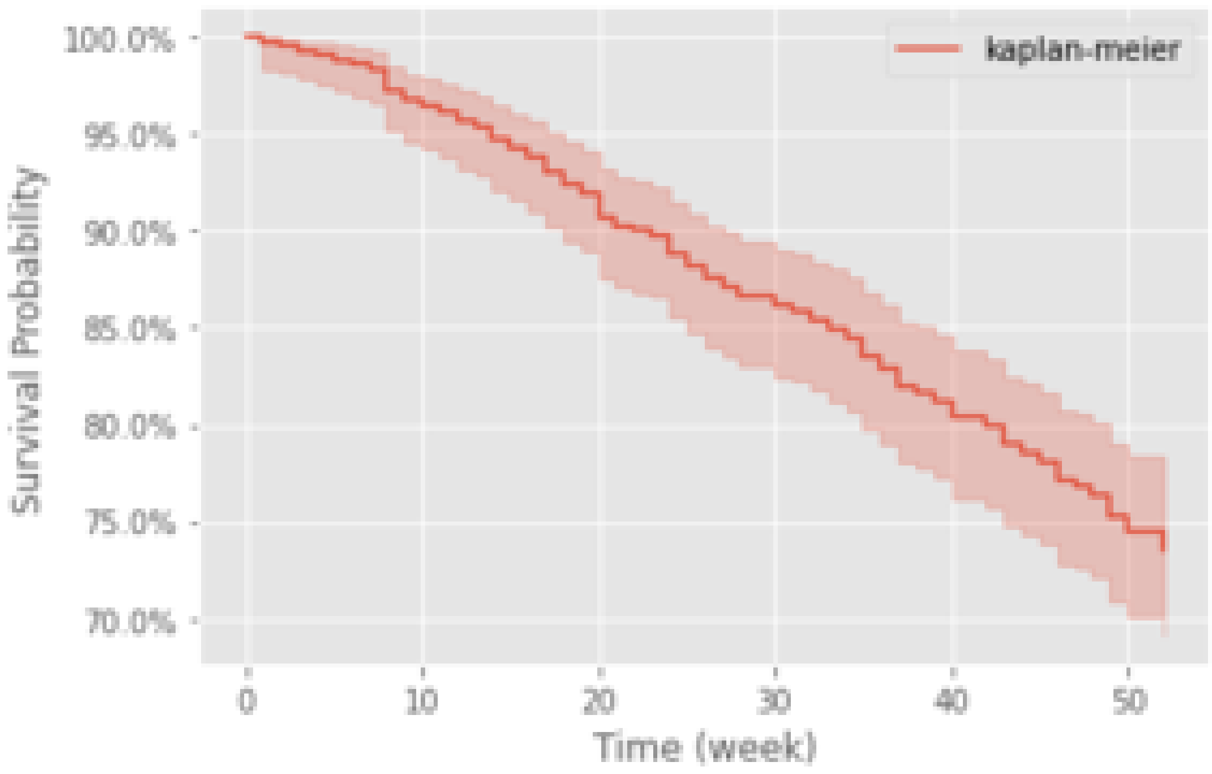
Censoring



Estimate survival curves

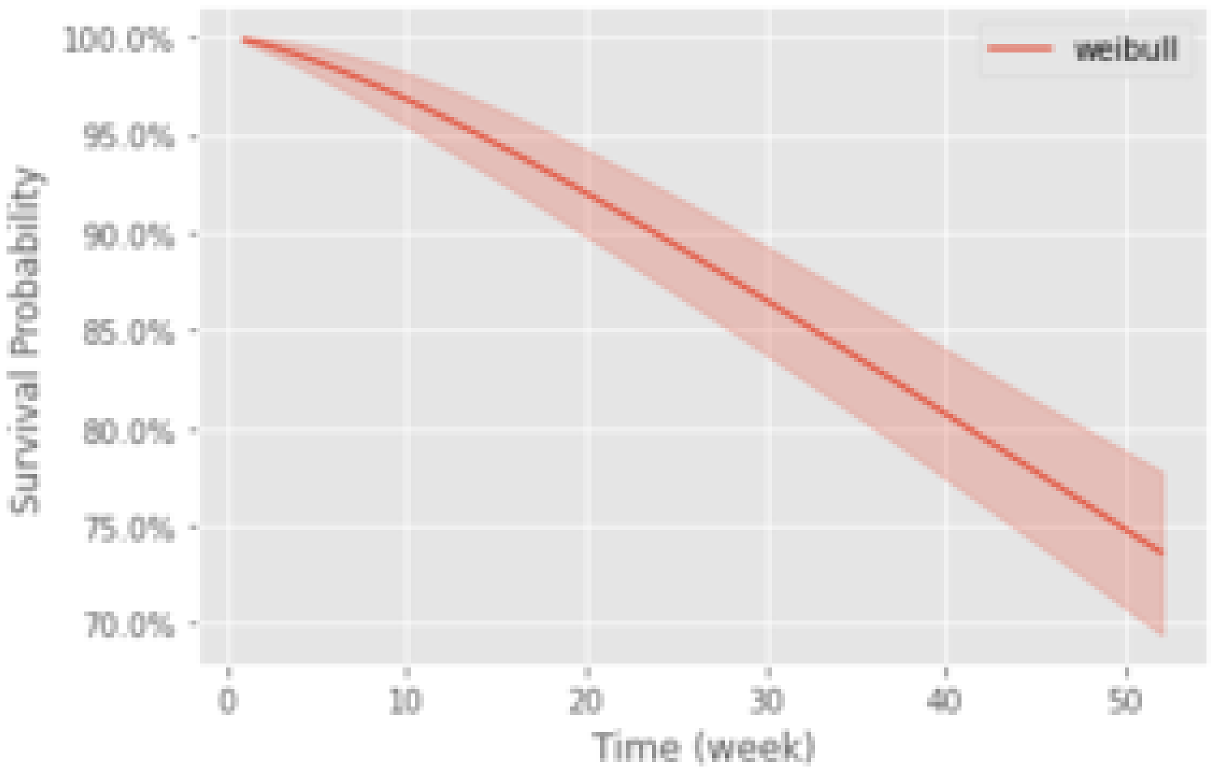
Kaplan-Meier estimator

- Non-parametric model



Weibull model

- Parametric model



The lifelines package

Kaplan-Meier estimator

```
from lifelines import KaplanMeierFitter
```

```
kmf = KaplanMeierFitter()
kmf.fit()
```

Weibull model

```
from lifelines import WeibullFitter
```

```
wb = WeibullFitter()
wb.fit()
```

Survival curve with covariates

Methods

- Log-rank test
- Weibull model
- Cox Proportional-Hazards (Cox PH) model

Thank you!

SURVIVAL ANALYSIS IN PYTHON