

```
credentials= credentials)
return
```

## Asset Failure Interface

based on the initial Survival Analysis of phase 1, the idea in this phase is industrialize the model and introduce state of the art NN and loss reduction functions like log negative likelihood to calculate the probability of failure of an asset. Additionally refresh the model with new incidents and re-think the input features as the current ones (those that describe weather 24 hours before the incident) we believe that are not really relevant for the model and we think that better change them for some that describe maximum and minimum during life of the asset

some background about Survival Analysis

<https://lifelines.readthedocs.io/en/latest/Survival%20Analysis%20intro.html>

repository

<https://gitlab.com/juan.huertas/afi-http-restful-interface>

## Survival analysis (Phase 1)

**Survival Analysis Master\_0-Copy1.1.ipynb:** the initial version of Cox regression and contains all the pre-processing.

**Cox\_regression\_v1.1.ipynb:** using processed dataset 'df\_cox\_master.csv', adding new weather features and median, with Cox regression and prediction code

**Asset\_failure\_result.ipynb:** python code for the API

**df\_cox\_master.csv:** the pre-processed dataset without weather features.

**cox\_master.csv:** contains processed weather features.

**cox\_master\_dummy.csv:** the same file as cox\_master.csv but has transformed categorical variables to dummy variables.

**Prediction\_table.csv:** the table contains the survival probability of each asset and their each instance, from day 1 to day 543. I transformed the table and added in instances and cumulative duration, now each column represents a day and each row represent one asset and its corresponding instance.

To run Cox regression, load cox\_master\_dummy as X1 and use below code:

```
X1=pd.read_csv('cox_master_dummy.csv')
```

```
cph = CoxPHFitter()
```









```
cph.fit(X1, 'T', 'event', cluster_col='Asset_Number',show_progress=True, step_size=0.50)
```

```
cph.print_summary() # the summary of Cox regression
```

```
prediction = cph.predict_survival_function(X1, times=[1., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20., 21., 22., 23.,
24., 25., 26., 27., 28., 29., 30., 31., 32., 33., 34., 35., 36., 37., 38., 39., 40., 41., 42., 43., 44., 45., 46., 47., 48., 49., 50., 51., 52., 53., 54., 55., 56.,
57., 58., 59., 60., 61., 62., 63., 64., 65., 66., 67., 68., 69., 70., 71., 72., 73., 74., 75., 76., 77., 78., 79., 80., 81., 82., 83., 84., 85., 86., 87., 88., 89.,
90., 91., 92., 93., 94., 95., 96., 97., 98., 99., 100., 101., 102., 103., 104., 105., 106., 107., 108., 109., 110., 111., 112., 113., 114., 115., 116., 117.,
118., 119., 120., 121., 122., 123., 124., 125., 126., 127., 128., 129., 130., 131., 132., 133., 134., 135., 136., 137., 138., 139., 140., 141., 142.,
143., 144., 145., 146., 147., 148., 149., 150., 151., 152., 153., 154., 155., 156., 157., 158., 159., 160., 161., 162., 163., 164., 165., 166., 167.,
168., 169., 170., 171., 172., 173., 174., 175., 176., 177., 178., 179., 180., 181., 182., 183., 184., 185., 186., 187., 188., 189., 190., 191., 192.,
193., 194., 195., 196., 197., 198., 199., 200., 201., 202., 203., 204., 205., 206., 207., 208., 209., 210., 211., 212., 213., 214., 215., 216., 217.,
218., 219., 220., 221., 222., 223., 224., 225., 226., 227., 228., 229., 230., 231., 232., 233., 234., 235., 236., 237., 238., 239., 240., 241., 242.,
243., 244., 245., 246., 247., 248., 249., 250., 251., 252., 253., 254., 255., 256., 257., 258., 259., 260., 261., 262., 263., 264., 265., 266., 267.,
268., 269., 270., 271., 272., 273., 274., 275., 276., 277., 278., 279., 280., 281., 282., 283., 284., 285., 286., 287., 288., 289., 290., 291., 292.,
293., 294., 295., 296., 297., 298., 299., 300., 301., 302., 303., 304., 305., 306., 307., 308., 309., 310., 311., 312., 313., 314., 315., 316., 317.,
318., 319., 320., 321., 322., 323., 324., 325., 326., 327., 328., 329., 330., 331., 332., 333., 334., 335., 336., 337., 338., 339., 340., 341., 342.,
343., 344., 345., 346., 347., 348., 349., 350., 351., 352., 353., 354., 355., 356., 357., 358., 359., 360., 361., 362., 363., 364., 365., 366., 367.,
368., 369., 370., 371., 372., 373., 374., 375., 376., 377., 378., 379., 380., 381., 382., 383., 384., 385., 386., 387., 388., 389., 390., 391., 392.,
393., 394., 395., 396., 397., 398., 399., 400., 401., 402., 403., 404., 405., 406., 407., 408., 409., 410., 411., 412., 413., 414., 415., 416., 417.,
418., 419., 420., 421., 422., 423., 424., 425., 426., 427., 428., 429., 430., 431., 432., 433., 434., 435., 436., 437., 438., 439., 440., 441., 442.,
443., 444., 445., 446., 447., 448., 449., 450., 451., 452., 453., 454., 455., 456., 457., 458., 459., 460., 461., 462., 463., 464., 465., 466., 467.,
468., 469., 470., 471., 472., 473., 474., 475., 476., 477., 478., 479., 480., 481., 482., 483., 484., 485., 486., 487., 488., 489., 490., 491., 492.,
493., 494., 495., 496., 497., 498., 499., 500., 501., 502., 503., 504., 505., 506., 507., 508., 509., 510., 511., 512., 513., 514., 515., 516., 517.,
518., 519., 520., 521., 522., 523., 524., 525., 526., 527., 528., 529., 530., 531., 532., 533., 534., 535., 536., 537., 538., 539., 540., 541., 542.,
```

543.])

prediction # the survival probability of each asset and their each instance, from day 1 to day 543

File	Modified
>  LDM Data required by SIM_v2_DS_JT.xlsx	Jan 25, 2019 by jialitao@deloitte.co.uk
>  tpr_asset_v0.1.xlsx	Jan 25, 2019 by jialitao@deloitte.co.uk
>  cox_master.csv	Mar 16, 2019 by jialitao@deloitte.co.uk
>  cox_master_dummy.csv	Mar 16, 2019 by jialitao@deloitte.co.uk
>  prediction_table.csv	Mar 16, 2019 by jialitao@deloitte.co.uk
>  Survival Analysis Master_0-Copy1.1.ipynb	Mar 16, 2019 by jialitao@deloitte.co.uk
>  cox_regression_v1.1.ipynb	Mar 16, 2019 by jialitao@deloitte.co.uk
>  df_cox_master.csv	Mar 20, 2019 by jialitao@deloitte.co.uk

Drag and drop to upload or [browse for files](#)

 Download All

[Working document] **Survival Analysis - Asset defect**

## Assumptions:

### 1. Asset type

- Decision:** Use 'Grouping\_Full\_Name' as the basis of high level asset class mapping, and when 'Grouping\_Full\_Name' for a specific asset is N/A, use the information from Engineering\_Suffix, System\_Asset\_Type, EQUIP\_CLASS\_DESC to populate Asset\_Class\_Grouping.
- Rationale:** Grouping\_Full\_Name is a column from Ellipse system for asset classes, and it aligns with the 10 categories of assets listed in Network Rail Asset Management Strategy October 2014 (<https://cdn.networkrail.co.uk/wp-content/uploads/2016/11/Asset-Management-Strategy.pdf>). High\_Level\_Asset\_Class was populated by using Asset\_Class\_Grouping, and Point and Crossing (P&C), Signalling and Telecom (S&T) were grouped as two higher classes.
- Asset type mapping reference** - check Survival\_Analysis\_asset\_mapping\_v0.1.xlsx
  - Grouping\_Full\_Name: Directly mapped in from Ellipse
  - Asset\_Class\_Grouping: The same as Grouping\_Full\_Name; When Grouping\_Full\_Name is N/A, use the combined info from Engineering\_Suffix, System\_Asset\_Type, and EQUIP\_CLASS\_DESC
  - High\_Level\_Asset\_Class: Higher level grouping by us based on Asset\_Class\_Grouping; Fences and Barriers, Structures manually grouped with track for better model performance
  - Engineering\_Suffix: From FMS, not accurate due to multiple types for a single asset
  - System\_Asset\_Type: From Ellipse
  - EQUIP\_CLASS\_DESC: From Ellipse

### 2. Asset failure probability

- Decision:** For any past and future date beyond the dataset timeline, assume history repeat itself. As a result, the survival probability will always be a value in the prediction table based on which day the sim date is in the cycle (1-543), and which asset

it is

- b. **Rationale:** We assume all the assets entered our survival analysis on 2017 Jan 1st, and left on 2018 June 28th, due to the data constraints and model performance. The survival probability will remain the same after day 543 (2018 June 28th). But in real life, also for simulation purpose, we need to have a more realistic number.

### 3. Delay minutes and time window

- a. **Decision:** For delay minute and delay time window that associated with asset defect, delay minutes = the median all the delays of one asset in one instance, delay time window = defect occurred time of that asset to 2 hours after defect occurred time
- b. **Rationale:** We don't know how the delay minutes were calculated, nor if there's a logic for computing delay time window. Instead of assigning randomly generated numbers, we use historical values.

#### Method:

- Survival analysis – Kaplan–Meier estimator and Cox regression

#### Sample data:

- 940 Asset defect records for 637 assets from FMS
- Dataset timeline: First observed asset defect on 2017 January 1st, the last observed asset defect on 2018 June 28th
- Analysis timeline: 2017 January 1st to 2019 March 16th
- Assets grouped into high-level asset classes: Electrification and Plant (E&P), Points and Crossings (P&C), Signalling and Telecom (S&T) and Track
- Weather features mapped in using the information from the closest weather station

#### Explanatory variables:

- High\_level\_Asset\_Class: Asset type
- Weather\_Station: Approximate closest weather location of each asset
- temp\_24h\_d: Median temperature 24 hours before asset defect happened (unit:Kelvins)
- pressure\_24h\_d : Median pressure 24 hours before asset defect happened (unit: hPa)
- humidity\_24h\_d : Median humidity 24 hours before asset defect happened (unit: %)
- wind\_speed\_24h\_d: Median wind speed 24 hours before asset defect happened (Unit: meter/sec)
- wind\_speed\_24h\_d\_h : Maximum wind speed 24 hours before asset defect happened (Unit: meter/sec)
- wind\_speed\_24h\_d\_l: Minimum wind speed 24 hours before asset defect happened (Unit: meter/sec)
- temp\_24h\_range: The difference between the median max temperature and the median min temperature 24 hours before asset defect
- temp\_1w\_d: Median temperature 1 week before asset defect
- pressure\_1w\_d: Median pressure 1 week before asset defect
- humidity\_1w\_d: Median humidity 1 week before asset defect
- wind\_speed\_1w\_d: Median wind speed 1 week before asset defect
- wind\_speed\_1w\_d\_h: Max wind speed 1 week before asset defect
- wind\_speed\_1w\_d\_l: Min wind speed 1 week before asset defect
- Temp\_1w\_range: The difference between the median max temperature and the median min temperature 1 week before asset defect

\*Note: if there's no defect for an asset, weather features calculated as 24 hours/1 week before the last day in the weather dataset (2019 Jan 20th)

#### Output:

- Kaplan-Meier curves by asset classes and weather station
- The summary of Cox regression
- Predicted survival probability across dataset timeline

#### Results:

<lifelines.CoxPHFitter: fitted with 940 observations, 637 censored>

duration col = 'T'  
event col = 'event'  
cluster col = 'Asset\_Number'  
robust variance = True  
number of subjects = 940  
number of events = 303  
log-likelihood = -1609.47  
time fit was run = 2019-03-16 13:19:45 UTC

	coef	exp(coef)	se(coef)	z	p	-log2(p)	lower 0.95	upper 0.95
High_Level_Asset_Class[T.P&C]	-0.11	0.9	0.22	-0.51	0.61	0.71	-0.53	0.31
High_Level_Asset_Class[T.S&T]	-0.13	0.88	0.19	-0.7	0.48	1.05	-0.5	0.24
High_Level_Asset_Class[T.Track]	0.29	1.34	0.15	1.88	0.06	4.06	-0.01	0.59
weather_station[T.Leeds]	-0.68	0.51	0.24	-2.82	<0.005	7.72	-1.16	-0.21
weather_station[T.Manchester]	-0.96	0.38	0.31	-3.14	<0.005	9.2	-1.56	-0.36
weather_station[T.Mirfield]	-0.73	0.48	0.29	-2.54	0.01	6.49	-1.29	-0.17
weather_station[T.York]	-0.62	0.54	0.31	-1.97	0.05	4.36	-1.23	0
temp_24h_d	0.03	1.03	0.03	0.86	0.39	1.35	-0.03	0.08
pressure_24h_d	-0.01	0.99	0.01	-0.9	0.37	1.45	-0.03	0.01
humidity_24h_d	-0.02	0.98	0.01	-2.26	0.02	5.38	-0.05	0
wind_speed_24h_d	-0.04	0.96	0.07	-0.66	0.51	0.98	-0.17	0.08
wind_speed_24h_d_h	0.31	1.36	0.04	8.72	<0.005	58.34	0.24	0.38
wind_speed_24h_d_l	-0.08	0.92	0.08	-0.97	0.33	1.6	-0.24	0.08
temp_24h_range	-0.11	0.9	0.09	-1.23	0.22	2.21	-0.28	0.06
temp_1w_d	0.03	1.03	0.03	0.77	0.44	1.17	-0.04	0.09
pressure_1w_d	0	1	0.02	-0.19	0.85	0.24	-0.03	0.03
humidity_1w_d	0.02	1.02	0.01	1.59	0.11	3.15	-0.01	0.05
wind_speed_1w_d	0.13	1.14	0.11	1.15	0.25	1.99	-0.09	0.35
wind_speed_1w_d_h	-0.36	0.7	0.04	-9.93	<0.005	74.73	-0.43	-0.29
wind_speed_1w_d_l	-0.29	0.75	0.25	-1.16	0.25	2.02	-0.77	0.2
temp_1w_range	0.32	1.37	0.15	2.06	0.04	4.65	0.01	0.62

Concordance = 0.91

Likelihood ratio test = 797.76 on 21 df, -log2(p)=513.46

Likelihood ratio test = 797.76 on 21 df, -log2(p)=513.46

**exp(coef) = Hazard Ratio (HR)**

**HR = 1: lack of association**

**HR > 1: an increased risk**

**HR < 1: a smaller risk**

Comparing Track asset group to the default baseline group E&P, there is a 34% higher risk of track asset group failing (HR = 1.34, p-value = 0.06). However, the association is slightly weak due to p-value is not small than 0.05.

The p-value for each weather station is smaller than 0.05, suggesting statistically significant associations between weather stations and time to asset failure. For example, the baseline group for weather station is Huddersfield by default. For the assets belong to weather station Leeds, they have 49% lower risk of failing compared to the assets belong to Huddersfield (HR = 0.51, p-value < 0.005).

Wind\_speed\_24h\_d\_h has a p-value smaller than 0.005 and an HR of 1.36, indicating a strong relationship between the high wind speed and increased risk of asset failing (HR = 1.36, p-value < 0.005).

temp\_1w\_range has a p-value smaller than 0.05 and an HR of 1.37, showing that the risk of assets failing is significantly higher by 37% when the temperature range 1 week before asset failing increases by 1 degree.

## Asset Data

Asset Criticality Tool High Level Asset Class

Point and Crossing (P&C)

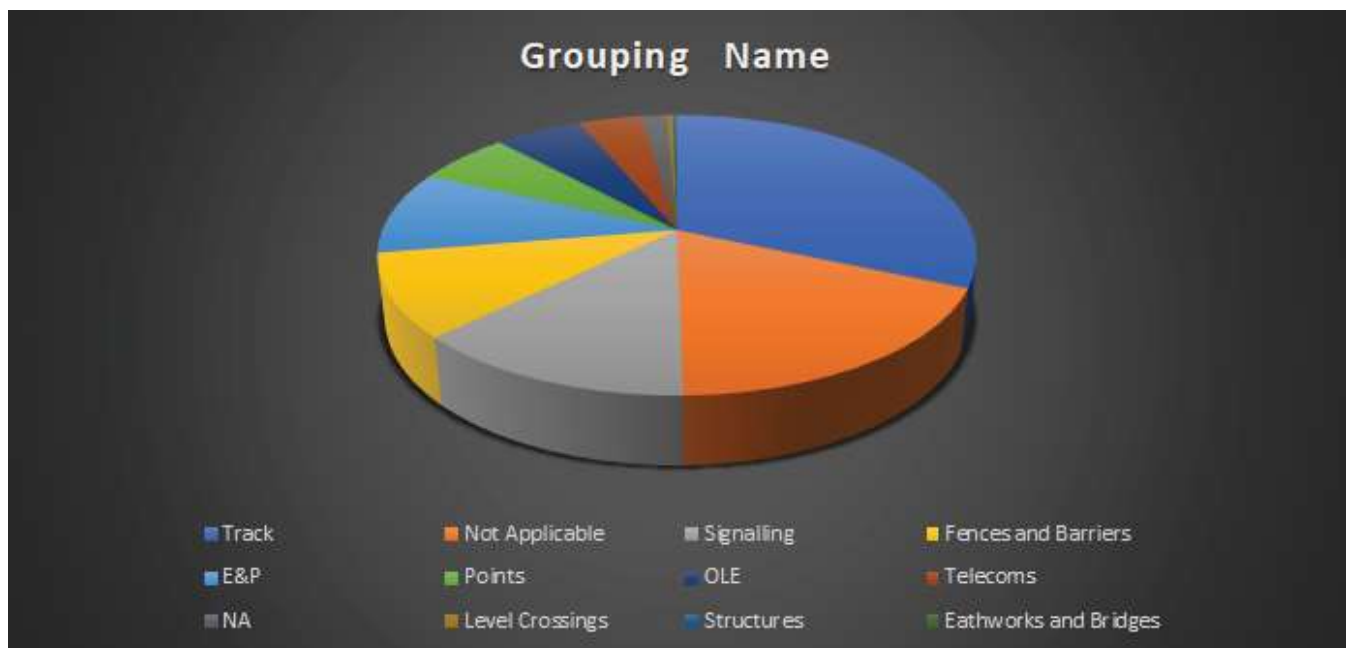
Signalling and Telecom (S&T)

Electrical & Power (E&P)

Tracks

NR\_asset Classification

Grouping_Full_Name	Total
Track	24288
Not Applicable	13969
Signalling	10036
Fences and Barriers	7426
E&P	7404
Points	4640
OLE	4310
Telecoms	3083
NA	1314
Level Crossings	266
Structures	118
Eathworks and Bridges	98



Grouping_Full_Name	ITEM_NAME_2	Total
--------------------	-------------	-------

Track	INSULATED BLOCK JOINT	4383
Track	CHAMBER - CATCHPIT	4015
Track	PIPE - SURFACE WATER	3654
E&P	DISCONNECTION BOX - TRACK CIRCUIT	2885
OLE	OLE STRUCTURE - CANTILEVER	2601
Points	S&C - STRETCHER BAR	2318
Track	DRAINAGE - 1/8 MILE SECTION	2136
Not Applicable	VEGETATION ? 1/8 MILE - UP	2121
Not Applicable	VEGETATION - 1/8 MILE - DOWN	2120
Not Applicable	LOCATION CASE	2043
E&P	CABLE - SIGNAL SUPPLY	1945
Fences and Barriers	BOUNDARY: POST & WIRE	1704
Not Applicable	SIGNALLING POWER SUPPLY - BUSBAR - DC	1624
Fences and Barriers	BOUNDARY: VERTICAL BAR 1	1536
Signalling	FUNCTIONAL SUPPLY POINT	1243
Signalling	SIGNALLING RELAYS	1222
Not Applicable	VEGETATION - HAZARDOUS TREE	1177
Signalling	TRACK CIRC - DC MED VOLT - AC IMM	1160
Fences and Barriers	BOUNDARY: NON BOUNDARY	971
Telecoms	SIGNAL POST TELEPHONE	940
Fences and Barriers	BOUNDARY: MASONRY	861
E&P	DISCONNECTION BOX - AWS	850
Signalling	AWS - AUTOMATIC WARNING SYSTEM	831
Points	S&C - SWITCH PANEL	753
Track	S&C - COMMON CROSSING	737
Track	S&C - LEFT HAND HALF SET	737
Track	S&C - RIGHT HAND HALF SET	737
Telecoms	DATA LOGGER	713
Track	S&C - TURNOUT	672
Track	OUTFALL - WATERCOURSE-ART'ICIAL	658
Not Applicable	EQUIP SUPPORT STRUCTURE - SIGNAL POST	605
Points	POINT HEATER - STRIP	602
Signalling	SSI - DATA LINK MODULE	592
Signalling	SIG HEAD - COLOUR LIGHT - LED	579
Not Applicable	MARKER POST - MILE	560
Track	CULVERT	517
Not Applicable	SUPPLEMENTARY DRIVE - MECHANICAL	511

Track	UNDERBRIDGE - CULVERT	509
Fences and Barriers	BOUNDARY: MESH 1	489
Signalling	SSI - TFM - SIGNAL	474
	LINESIDE CABLE - FIBRE	468
Fences and Barriers	BOUNDARY: POST & RAIL	452
Points	POINT OP EQUIP - HYDRAULIC/PNEUMATIC	446
Track	UNDERBRIDGE - NON VIADUCT	437
Track	TRACK ID	424
Track	LEFT RAIL - PLAIN LINE	423
Track	RIGHT RAIL - PLAIN LINE	420
OLE	OLE STRUCTURE - PORTAL	418
E&P	RESIDUAL CURRENT DEVICE	402
OLE	OLE STRUCTURE - CANTILEVER ANCHOR	369
	CABLE JOINT	325
Not Applicable	DECKING	318
Track	POINT DRAINAGE NODE	316
Not Applicable	AUTHORISED ACCESS POINT - PEDESTRIAN	312
Signalling	TRACK CIRC - DC MED VOLT - AC IMM - DCT	312
Signalling	TRACK CIRC - TI21	292
Track	OVERBRIDGE - NON VIADUCT	278
OLE	OLE STRUCTURE - HEADSPAN	276
Not Applicable	SIGN - PSR FULL SIZE	265
E&P	AIR CONDITIONING SYSTEM	257
Points	POINT OP EQUIP - POINT MACHINE	250
Signalling	SIGNALLING - MULTI-CORE CABLE	242
Signalling	TPWS - TSS	237
Telecoms	POINT ZONE TELEPHONE	236
Not Applicable	VEGETATION AREA - UP	229
Not Applicable	VEGETATION AREA - DOWN	229
E&P	ELECTRICITY SUPPLY - METERED - DNO	226
Fences and Barriers	BOUNDARY AREA: DOWN	220
Fences and Barriers	BOUNDARY AREA: UP	214
Track	CHAMBER - MANHOLE	206
Fences and Barriers	BOUNDARY: MISSING BOUNDARY	205
Not Applicable	SIGN - PSR MINI-SIZE	202
Signalling	TPWS - OSS & TSS	197
Not Applicable	AUTHORISED ACCESS POINT - VEHICLE	195

	LINESIDE RADIO SIGN	185
Track	CHANNEL - ARTIFICIAL DITCH	184
Track	PLATFORM	184
Fences and Barriers	BOUNDARY: DRY STONE WALL	178
Signalling	SIGNAL HEAD - COLOUR LIGHT - 4 ASPECT	173
Not Applicable	SIGNALLING POWER SUPPLY - BUSBAR - AC	169
Track	ADJUSTMENT SWITCH	165
Fences and Barriers	BOUNDARY: CHAIN LINK	157
Signalling	POSITION LIGHT SIGNAL-LED	153
Signalling	TRACK CIRC - DC LOW VOLT - PLAIN	153
Signalling	ZONE CONTROLLER COMPONENT	152
Track	DRAINAGE AREA	151
	TELECOMS LOCATION CASE	139
OLE	OLE STRUCTURE - TUNNEL	139
Track	SIGN - 'PSR'	138
Fences and Barriers	BOUNDARY: TIMBER	136
Telecoms	TRAIN READY TO START (TRTS) SYSTEM	135
E&P	SURGE PROTECTION DEVICE	134
Signalling	GROUND POSITION LIGHT - LED	130
Track	INTERMEDIATE DRAINAGE NODE	127
Telecoms	VIRTUAL CABLE (FMS ONLY)	125
Signalling	SSI - TFM - POINTS	122
OLE	OLE STRUCTURE - PORTAL ANCHOR	121
Fences and Barriers	BOUNDARY: MESH 2	120
Track	RAIL LUBRICATOR	119
OLE	OLE STRUCTURE - MAST	117
Points	POINT HEATING CONTROL CABINET	115
Signalling	BUILDING - TELECOM EQUIP ROOM	112
Not Applicable	TREADLE	111
Not Applicable	CATTLE GRID	110
OLE	OLE STRUCTURE - BRIDGE UNDERARM	110
E&P	TRANSFORMER BOOSTER - SEALED OIL	109
Signalling	TPWS - OSS	106
Signalling	RELAY ROOM	100
Track	CROSSOVER (S&C)	100
Track	INLET OR OUTLET STRUCTURE	100
Track	CHANNEL - NATURAL DITCH	99



Not Applicable	EQUIP SUPPORT STRUCTURE - GANTRY/PORTAL	96
Telecoms	ROUTE INDICATOR - STENCIL - NFO	93
Telecoms	ROUTE IND - STENCIL - FIBRE OPTIC	92
	LINESIDE CABLE - COPPER	92
Track	BOUNDARY DRAINAGE NODE	91
Signalling	SIGNAL POST REPLACEMENT SWITCH	91
E&P	DISCONNECTION BOX - POINTS	89
Signalling	SSI - LONG DISTANCE TERMINAL	89
Track	OVERBRIDGE - FOOT	89
Track	UNDERTRACK CROSSING	86
Track	LUBRICATOR ELECTRIC	85
Not Applicable	ROUTE INDICATOR FEATHER - LED	81
Points	POINT DETECTOR - ELECTRICAL	80
Structures	TUNNEL SHAFT	80
Track	UNDERBRIDGE - PIPE	79
Signalling	VIRTUAL LINESIDE CABLE	78
Fences and Barriers	BOUNDARY: CONCRETE PANEL	75
Signalling	TRACK CIRC - IMPULSE	71
Track	ULTRASONICS JUNCTION AREA	70
Signalling	SIGNAL HEAD - COLOUR LIGHT - 3 ASPECT	70
Signalling	SSI - MANAGEMENT PROCESSOR MODULE	69
Telecoms	SECTION TELEPHONE	69
Telecoms	ACCOMMODATION CROSSING TELEPHONE	69
Not Applicable	AUTHORISED ACCESS POINT - ROAD/RAIL VEH	67
Signalling	AXLE COUNTER-FRAUSCHER-WHEEL DETECTOR	66
Signalling	TRACK CIRC - ASTER SF15/U TYPE	65
Signalling	TRAIN OPERATED WARNING SYSTEM	65
Level Crossings	LEVEL XING - WARNING LIGHTS	65
Track	UNDERBRIDGE - VIADUCT	64
Not Applicable	MARKER POST - GRADIENT	62
Telecoms	ROUTE INDICATOR - THEATRE - NFO	61
Not Applicable	CABLE ROUTE - GROUND TROUGHING	59
OLE	OLE STRUCTURE - MAST - ANCHOR	58
Track	OTHER REFLECTIVE SIGN	56
E&P	CHANGEOVER PANEL	55
Track	COVERED CHANNEL - OTHER	55
Eathworks and Bridges	FORMER BRIDGE STRUCTURE	54

Track	SIGN - 'WHISTLE'	54
Track	S&C - CATCH / TRAP POINT	54
Level Crossings	LEVEL XING - BARRIER MECHANISM	53
Not Applicable	CABLE ROUTE - MAINTAINABLE UNIT	52
Track	INFLOW - LAND DRAINAGE - FARM	51
E&P	DISCONNECTION BOX - EQUIPMENT HOUSING	49
Signalling	SSI - PANEL PROCESSOR	47
Not Applicable	SIGN - PSR - ADVANCE BOARD - FULL SIZE	46
Signalling	SOLID STATE INTERLOCKING	46
Signalling	TRACK CIRC - DC LOW VOLT - AC IMM	46
Not Applicable	VEGETATION UP	46
Not Applicable	VEGETATION DOWN	46
OLE	TRACTION: ELECTRICAL SECTION	45
Not Applicable	ROUTE IND - FEATHER - 1 ROUTE	44
Eathworks and Bridges	SIDEBRIDGE	44
Track	S&C - OBTUSE CROSSING	42
Track	UNABLE TO LOCATE DRAINAGE NODE	42
Track	LUBRICATOR HYDRAULIC	42
E&P	DOMESTIC DISTRIBUTION PANEL	40
Not Applicable	WATER SUPPLY	40
Signalling	SIG HEAD - DORMAN - LED	39
OLE	OLE STRUCTURE - HEADSPAN ANCHOR	39
Signalling	TRACK CIRCUIT - REED JOINTED	38
Structures	TUNNEL BORE	38
Telecoms	RADIO - COVERAGE (FMS ONLY)	37
Fences and Barriers	BOUNDARY: UP	36
Fences and Barriers	BOUNDARY: DOWN	36
Track	DRAINAGE - UP	36
Track	DRAINAGE - DOWN	36
Level Crossings	LEVEL XING - FOOTPATH CROSSING	36
Not Applicable	PSR MAGNET	36
Track	HYDRAULIC LUBRICATOR	36
Telecoms	ROUTE IND - THEATRE - FIBRE OPTIC	36
Not Applicable	SSI - INTERLOCKING CUBICLE	35
Track	INFLOW - GROUNDWATER	35
Telecoms	RADIO - GSM-R MOBILE (FMS ONLY)	35
E&P	EMERGENCY LIGHTING SYSTEM	34

Signalling	PRINCIPAL SUPPLY POINT	33
Not Applicable	END STOP - BUFFER STOP (FIXED)	32
Telecoms	REMOTE CONTROL - TDM - SYSTEM	32
Telecoms	SIGNAL - BANNER REPEATER - LED	32
Track	POINT OF INTEREST DRAINAGE NDE	31
Track	LONGITUDINAL TIMBERS	31
Track	SIGN - 'COUNT DOWN MARKER'	31
E&P	PRINCIPAL UPS	31
Track	INFLOW - UNKNOWN	30
Telecoms	RA / CD INDICATOR - STENCIL	30
E&P	OLE - RED BONDS	30
Track	PIPE - FOUL WATER	29
Signalling	INTERLOCKING - RELAY - NON-GEOGRAPHICAL	29
Signalling	POSITION LIGHT - STD - 3 ASPECT	29
E&P	TRANSFORMER-SIGNAL SUPPLY	29
Not Applicable	SBK - LOOSE LIFTING EQUIP	28
Signalling	STAFF PROTECTION LOCKOUT PATROL SWITCH	27
Telecoms	REMOTE CONTROL - TDM - REMOTE	27
Telecoms	EVENT RECORDER	27
Signalling	EQUIPMENT ROOM (NON SPECIFIC)	26
Signalling	INTERLOCKING - ELECTRONIC	26
Signalling	SSI - DIAGNOSTIC MODULE	26
Track	CHANNEL - CASCADE	25
Not Applicable	END STOP - FRICTION	25
Signalling	SSI - TERMINATION UNIT	25
Telecoms	CONTROLLED CROSSING TELEPHONE	25
Telecoms	VIRTUAL CABLE ROUTE (FMS ONLY)	25
Signalling	TPWS+ - TSS - OSS - OSS+ LOOPS	24
Signalling	PSP ENCLOSURE	23
Level Crossings	LEVEL XING - USER WORKED WITH TELEPHONES	23
Telecoms	'OFF' INDICATOR - FIBRE OPTIC	23
Track	SIGN - 'ELECTRIFICATION WARNING'	23
Not Applicable	SUPPLEMENTARY DRIVE - HYDRAULIC	23
E&P	HV TRANSFORMER - OIL - NON TRACTION	23
Telecoms	'OFF' INDICATOR - LED	22
	WESTCAD CONTROL SYSTEM	22
Telecoms	INSULATION RESISTANCE MONITORING DEVICE	22

E&P	BATTERY	21
Track	OUTFALL - PUBLIC SEWER	21
Not Applicable	CIRCUIT CONTROLLER	21
E&P	TRANSFORMER-PROTECTION VOLTAGE	21
Points	POINT OP EQUIP - MECHANICAL	20
Track	S&C - FIXED DIAMOND	20
E&P	DEHUMIDIFIER	19
Track	ELECTRIC/ELECTRONIC LUBRICATOR	19
Track	SIGN - 'LIMITED CLEARANCE'	19
Signalling	SIGNAL BOX	19
Track	SWITCH DIAMOND - WING RAIL	18
E&P	DIESEL ALTERNATOR SET - 650V	18
E&P	DISCONNECTION BOX - SIGNAL	18
Track	COVERED CHANNEL - COLLECTOR	18
Signalling	POSITION LIGHT - STD - 2 ASPECT	18
Signalling	TRACK CIRC - DC MED VOLT - PLAIN	18
Telecoms	HOT AXLE BEARING DETECTOR - SIG BOX END	17
Points	S&C - LEFT SWITCH BLADE	17
Points	S&C - RIGHT SWITCH BLADE	17
Signalling	SIGNAL HEAD - COLOUR LIGHT - 2 ASPECT	17
Signalling	TRACK CIRC - EBI 400	17
Signalling	TRAIN DESCRIBER	17
Level Crossings	LEVEL XING - AUDIBLE WARNING SYSTEM	16
Signalling	EBI TRACK 400 DOUBLE RAIL APPLICATION	16
Level Crossings	SMS-OD01C-MCB_OBSTACLE-DETECTOR-RADAR	15
Telecoms	RA / CD INDICATOR - FIBRE OPTIC	15
Track	STATIC SANDITE UNIT	15
Telecoms	TUNNEL TELEPHONE	15
Track	CHAMBER - INTERCEPTOR	14
Not Applicable	EQUIP SUPPORT STRUCTURE - MAST	14
Not Applicable	SIGN - PSR - ADVANCE BOARD - MINI	14
E&P	CHARGER	14
Track	TRACTION GEL APPLICATOR	14
Telecoms	REMOTE CONTROL - TDM - LOCAL	14
Signalling	SIGNAL - COLOUR LIGHT-LED-DORMAN INTERGR	14
Not Applicable	RELAY RACK	13
Track	OVERBRIDGE - PIPE	13

Signalling	AXLE COUNTER - FRAUSCHER-EVALUATOR	12
E&P	CONTROL PANEL	12
Not Applicable	GAS SUPPLY	12
Not Applicable	WINDING AND SUPPORT EQUIPMENT - CCTV	12
	REMOTE CONTROL - NON-REED FDM - LOCAL	12
Track	SIGN - 'STOP/WAIT'	12
E&P	TRANSFORMER-BOOSTER-FREE BREATHING-OIL	12
E&P	LIGHTING SYSTEM - LEVEL CROSSING	11
Track	SIGN - 'HIGHWAY SIGNAGE'	11
Signalling	SIGNAL CONTROL - ELECTROMECHANICAL	11
Track	CHANNEL - FLUME	10
Track	PIPE - COMBINED	10
Track	INFLOW - SURFACE WATER DRAIN	10
Not Applicable	FIRE PROTECTION SYSTEM	10
Telecoms	HOT AXLE BEARING DETECTOR	10
Not Applicable	LEVER LOCK	10
Not Applicable	DRIVER (SPT) WALKWAY	10
Telecoms	RTU: SCADA DISTRIBUTED	10
Signalling	NON TRACTION SUBSTATION	9
Not Applicable	CABLE ROUTE - PIPED	9
Track	INFLOW - WATERCOURSE	9
Track	OUTFALL - WATERCOURSE-NATURAL	9
Signalling	SSI - TECHNICIANS TERMINAL	9
Signalling	EMERGENCY REPLACEMENT SWITCH	9
Not Applicable	LEVER FRAME - SIGNAL BOX	9
Not Applicable	COMBINED LOCK	9
	PROTECTION & CONTROL DEVICES	9
Not Applicable	NEUTRAL SECTION - ARTHUR FLURY SKIDLESS	9
Track	TOP OF RAIL FRICTION MODIFIER	9
Telecoms	STENCIL - 'OFF' INDICATOR	9
Telecoms	STENCIL - LIMIT OF SHUNT	9
Telecoms	SIGNAL - BANNER REPEATER - FIBRE OPTIC	9
Signalling	SEMAPHORE DISC - MECHANICAL	9
	OPERATIONAL LINESIDE CCTV POWER	9
	VDU/PC TECHNICIAN TERMINAL	9
Level Crossings	LEVEL XING - BARROW CROSSING LIGHTS	9
E&P	DIESEL ALTERNATOR SET - 400V	8

Level Crossings	SMS-OD02B-MCB_OBSTAC-DETECT-LIDAR	8
Signalling	SPAD INDICATOR	8
Track	PLATFORM 1	8
Track	PLATFORM 2	8
Track	S&C - SWITCH DIAMOND	8
Signalling	POSITION LIGHT - FIBRE - 3 ASPECT	8
Signalling	SEMAPHORE - FIXED ARM	8
Signalling	SIGNAL CONTROL - NX	8
Points	S&C - LEFT HALF SIDE	8
Points	S&C - RIGHT HALF SIDE	8
Telecoms	INSULATION RESISTANCE MON DEV	8
Telecoms	GROUND FRAME TELEPHONE	8
Telecoms	BRIDGE / VIADUCT TELEPHONE	8
Track	UNDERBRIDGE - INTERSECTION	8
Fences and Barriers	BOUNDARY - POST & WIRE (BS1722-2)	7
Signalling	BUILDING - SSP/PSP EQUIP ROOM	7
Track	INFLOW - LAND DRAINAGE-GARDEN	7
Track	POND	7
Not Applicable	EQUIP SUPPORT STRUCTURE - CANTILEVER	7
Level Crossings	LEVEL XING - USER WORKED WITH MSL	7
Level Crossings	LEVEL CROSSING GATE - HAND OPERATED	7
Not Applicable	LEVER FRAME - GROUND FRAME	7
Track	SIGN - 'PROHIBITED ACCESS'	7
Signalling	COL LIGHT (LED) / 4 MODULE	7
Signalling	POSITION LIGHT - STD - 4 ASPECT	7
Not Applicable	STATION	7
Not Applicable	HABD HOUSING	6
Fences and Barriers	BOUNDARY: <TYPE UP HERE>	6
Fences and Barriers	BOUNDARY: <TYPE DOWN HERE>	6
Telecoms	CCTV - LEVEL CROSSING - SYSTEM	6
Telecoms	CCTV SYSTEM - CAMERA	6
Telecoms	CCTV SYSTEM - MONITOR	6
Level Crossings	LEVEL XING - MCB WITH OBSTACLE DETECTION	6
Level Crossings	LEVEL XING - MCB WITH CLOSED CIRCUIT TV	6
Not Applicable	BLOCK APPARATUS	6
Track	SIGN - 'CLEARANCE'	6
Track	SIGN - 'LEVEL CROSSING'	6

Not Applicable	ROUTE IND - FEATHER - 2 ROUTE	6
Signalling	TRACK CIRC INTERRUPTER	6
E&P	DC SECONDARY CELL - ALKALINE	6
Signalling	AWS - YELLOW - PERMANENT MAGNET	5
Fences and Barriers	BOUNDARY: WATERCOURSE	5
Not Applicable	CABLE ROUTE - UNDERTRACK CROSSING	5
E&P	HEY	5
Points	POINT DETECTOR - MECHANICAL	5
Signalling	INTERLOCKING - MECHANICAL	5
E&P	CABLE - DOMESTIC MAINS	5
E&P	LIGHTING SYSTEM - BUFFER STOP	5
E&P	ELECTRICITY SUPPLY - UNMETERED - DNO	5
Not Applicable	NEUT SECT - CERAMIC BEAD-SKIDLESS	5
Telecoms	REMOTE CONTROL - DIRECT WIRED - SYSTEM	5
Telecoms	REMOTE CONTROL - DIRECT WIRED - LOCAL	5
Telecoms	REMOTE CONTROL - DIRECT WIRED - REMOTE	5
Signalling	SIGNAL HEAD - COLOUR LIGHT - 1 ASPECT	5
Signalling	SEMAPHORE STOP - MECHANICAL	5
Not Applicable	SBK - PORTABLE LADDER	5
Signalling	AXLE COUNTER	4
Track	S&C - COMPOUND CROSSING	4
Not Applicable	LIGHT MAINTENANCE DEPOT	4
E&P	DISBOX FITTED TPWS	4
Track	INFLOW - COMBINED WATER DRAIN	4
Not Applicable	WINDING & SUPPORT EQUIPMENT - GENERAL	4
E&P	ELECTRICITY SUPPLY - UNMETERED NOT DNO	4
OLE	OLE ISOLATOR - MANUAL	4
E&P	TRANSFORMER-SIGNAL SUPPLY-LV-DRY	4
E&P	HV TRANSFORMER - AIR - NON TRACTION	4
OLE	OLE STRUCTURE / CANTILEVER	4
Fences and Barriers	BOUNDARY: HEDGE	3
Fences and Barriers	BOUNDARY: VERTICAL BAR 2	3
Signalling	BUILDING / HARMONIC FILTER BUILDING	3
	BUSBAR - OLE - TRACTION	3
	BUSBAR - OLE - RETURN CURRENT	3
Not Applicable	CABLE ROUTE - ELEVATED TROUGHING	3
Not Applicable	END STOP - TEMPORARY BUFFER	3

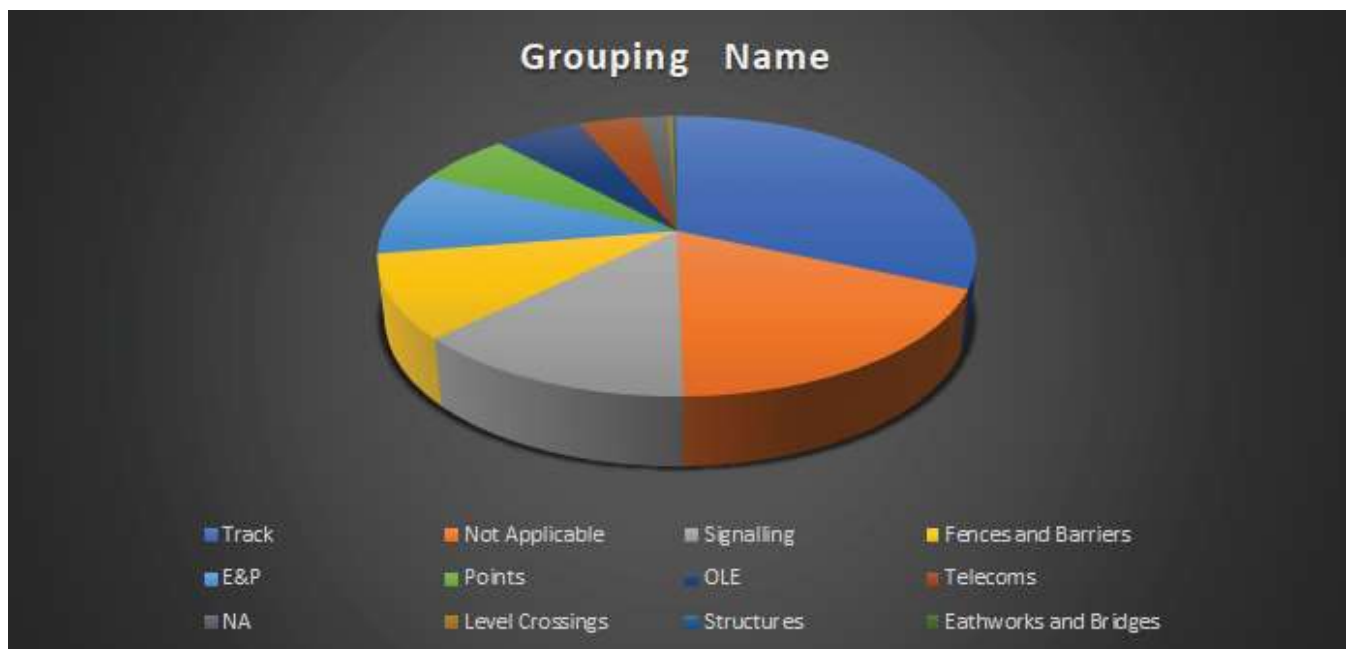
Not Applicable	SIGN - EQUIPMENT BATTERIES	3
E&P	LIGHTING SYSTEM - WALKWAY	3
Signalling	STAFF PROTECTION SYSTEM - LOCKOUT - BIDI	3
Track	OVERBRIDGE - VIADUCT	3
E&P	BATTERY CHARGER - TRIPPING	3
Track	LUBRICATOR CARTRIDGE	3
Telecoms	REMOTE CONTROL SYSTEM - TDM - PMUX	3
Track	SIGN - ST ANDREW CROSS	3
OLE	OLE ISOLATOR - MOTORISED	3
OLE	EARTHING SYSTEM - LOCAL EARTH	3
Signalling	SEMAPHORE DISTANT - MECHANICAL	3
Signalling	SIGNAL CONTROL - COMBINATION	3
	OPERATIONAL LINESIDE CCTV	3
E&P	UNINTERRUPTABLE POWER SUPPLY	3
	ANEMOMETER	2
Not Applicable	SHELF	2
Not Applicable	CUPBOARD	2
Not Applicable	TERMINAL BOX	2
Fences and Barriers	BOUNDARY - MESH 1 (BS1722-10)	2
	BUSBAR HIGH VOLTAGE AC	2
Not Applicable	CABLE ROUTE - BRACKETS/HANGERS	2
Track	CHANNEL - AQUEDUCT	2
Track	CHAMBER - PUMPING STATION	2
Track	OUTFALL - SOAKAWAY	2
Not Applicable	END STOP - SAND DRAG	2
	BOUNDARY FENCE	2
	GROUND SWITCH PANEL	2
Telecoms	HOT AXLE BEARING DETECTOR - T'SIDE END	2
Signalling	INTERLOCKING - ELECTROMECHANICAL	2
Level Crossings	LEVEL XING - USER WORKED CROSSING	2
Level Crossings	LEVEL XING - FOOTPATH WITH MSL	2
Level Crossings	LEVEL XING - MANUAL CONTROLLED BARRIERS	2
Level Crossings	LEVEL CROSSING GATE - WICKET - LOCKED	2
Not Applicable	AWS - AUTOMATIC WARNING SYSTEM	2
Not Applicable	PILOT CABLES	2
Not Applicable	NEUTRAL SECTION - BICC SKIDLESS	2
E&P	BATTERY CHARGER - SCADA RTU ELECTRONIC	2



Track	LUBRICATOR MECHANICAL	2
Track	SIGN - 'JUNCTION/LOCATION/NAME'	2
Track	SIGN - 'RADIO'	2
Track	SIGN - 'STOP/WAIT' + OTW	2
Not Applicable	ROUTE IND - FEATHER - 3 ROUTE	2
Telecoms	ROUTE INDICATOR - STENCIL - LED	2
Telecoms	ROUTE IND - FEATHER - 1 ROUTE	2
Track	S&C - SCISSORS	2
Track	S&C - DERAILER	2
	SUP: SCADA MASTER STATION	2
	RTU: SCADA POINT TO POINT	2
Not Applicable	SBK - VENTILATION FAN	2
	TRACK CIRC - AC 50 HZ VANE	2
Signalling	TRACK CIRCUIT/DC MED VOLT - PLAIN	2
	TURNING CHAMBER	2
Telecoms	CONTROLLED CROSSING TELEPHONE - CB	2
	LEVELS OF VISUAL TRACK INSPECTION	2
Track	PUMPING SYSTEM	2
Not Applicable	AIR RECEIVER	1
Not Applicable	LEVEL XING CONTROL CABINET	1
Fences and Barriers	BOUNDARY: DITCH	1
Fences and Barriers	BOUNDARY: POST & NETTING	1
Fences and Barriers	BOUNDARY: IRON RAILINGS	1
Fences and Barriers	BOUNDARY - VERTICAL BAR 1 (BS1722-12)	1
Signalling	CROSSING KEEPERS CABIN	1
E&P	DIESEL ALTERNATOR SET - DUAL	1
Track	INFLOW - TRADE EFFLUENT DRAIN	1
	ENGINE	1
Not Applicable	FIRE CONTROL PANEL	1
Not Applicable	SIGN - TSR - COMMENCEMENT BOARD	1
Signalling	ERTMS PROJECT	1
	INTERLOCKING - RELAY - GEOGRAPHICAL	1
E&P	CABLE - LOW VOLTAGE	1
Level Crossings	LEVEL XING - AUTOMATIC HALF BARRIERS	1
Level Crossings	LEVEL XING - MANNED GATES	1
Level Crossings	LEVEL XING - STATION FOOTPATH WITH WL	1
Level Crossings	LEVEL CROSSING GATE - BLACKS LOCK	1

Level Crossings	LEVEL CROSSING GATE - WICKET - UNLOCKED	1
Level Crossings	LEVEL CROSSING GATE - ELECTRICAL LOCK	1
E&P	LIGHTING SYSTEM - NAVIGATION	1
	VEHICLE HEALTH MONITOR	1
E&P	ELECTRICITY SUPPLY - METERED - NOT DNO	1
E&P	ELECTRICAL DISTRIBUTION CABINET	1
E&P	GRID SUPPLY POINT	1
	PANCHEX	1
Track	PLATFORM 3	1
Points	Point Heater Control Cabinet/Monitored	1
E&P	BATTERY CHARGER	1
	REMOTE CONTROL - REED FDM - SYSTEM	1
Track	SIGN - 'TRESPASS'	1
	REMOTE EQUIPMENT ROOM MONITORING	1
Track	S&C - TANDEM OR THREE THROW	1
	SIGNAL CONTROL - IECC	1
	ELECTRONIC ROUTE SETTING EQUIPMENT	1
OLE	EARTHING SYSTEM - SUPPLY AUTH EARTH	1
OLE	EARTHING SYSTEM - MULTI-ROD	1
	CCTV - SECURITY - SYSTEM	1
	INTRUDER ALARM SYSTEM	1
	SECURITY ACCESS PANEL	1
Signalling	SIGNAL - POINT SET INDICATOR	1
Signalling	COL LIGHT (LED) / 2 MODULE	1
	SIGNAL - DRIVERS WHITE LIGHT	1
Signalling	POSITION LIGHT - STD - LED	1
Signalling	SIGNAL - LIGHT EMITTING DIODE	1
Signalling	SEMAPHORE MINIATURE - MECHANICAL	1
Signalling	SIGNAL CONTROL - SWITCH PANEL	1
Signalling	SIGNALLING POWER SUPPLY - BUSBAR - DC	1
Signalling	SIGNALLING SUPPLY POINT	1
Not Applicable	SBK - UNINTERRUPTIBLE P S	1
Not Applicable	SBK - FUEL TRANSFER SYSTEM	1
Not Applicable	SBK - DIESEL BOWSER	1
	SWITCHBOARDS	1
	LV DISTRIBUTION BOARD	1
Signalling	TRACK CIRCUIT/DC MED VOLT - AC IMM	1

Signalling	TRACK CIRC - DC LOW VOLT - AC IMM - DCT	1
	SCREENING CABLE EARTH FARM	1
	FIBRE BAY	1
Telecoms	ACCESS POINT TELEPHONE	1
E&P	TRANSFORMER-SIGNAL SUPPLY REGULATING	1
OLE	OLE STRUCTURE / PORTAL - ANCHOR	1
E&P	DC SECONDARY CELL - CYCLON	1
	STANDBY GENERATOR MONITOR	1
Track	SWING OR DRAW BRIDGE	1
Track	PUMP SET	1
Level Crossings	LEVEL XING/FLASHING LIGHT RD SIGNAL LED	1



## Kaplan Meier Estimator

The **Kaplan–Meier estimator**,<sup>[1][2]</sup> also known as the **product limit estimator**, is a **non-parametric statistic** used to estimate the **survival function** from lifetime data. In medical research, it is often used to measure the fraction of patients living for a certain amount of time after treatment. In other fields, Kaplan–Meier estimators may be used to measure the length of time people remain unemployed after a job loss,<sup>[3]</sup> the time-to-failure of machine parts, or how long fleshy fruits remain on plants before they are removed by *frugivores*. The **estimator** is named after **Edward L. Kaplan** and **Paul Meier**, who each submitted similar manuscripts to the *Journal of the American Statistical Association*. The journal editor, **John Tukey**, convinced them to combine their work into one paper, which has been cited about 55,000 times since its publication.<sup>[4][5]</sup>

The **estimator** of the **survival function**  $\{S(t)\}$  (the probability that life is longer than  $\{t\}$ ) is given by:

## Jupyter Notebook



cox\_regression\_v... Estimator.ipynb

our Dataset

```

df=pd.read_csv('df_cox_master.csv')
df.columns

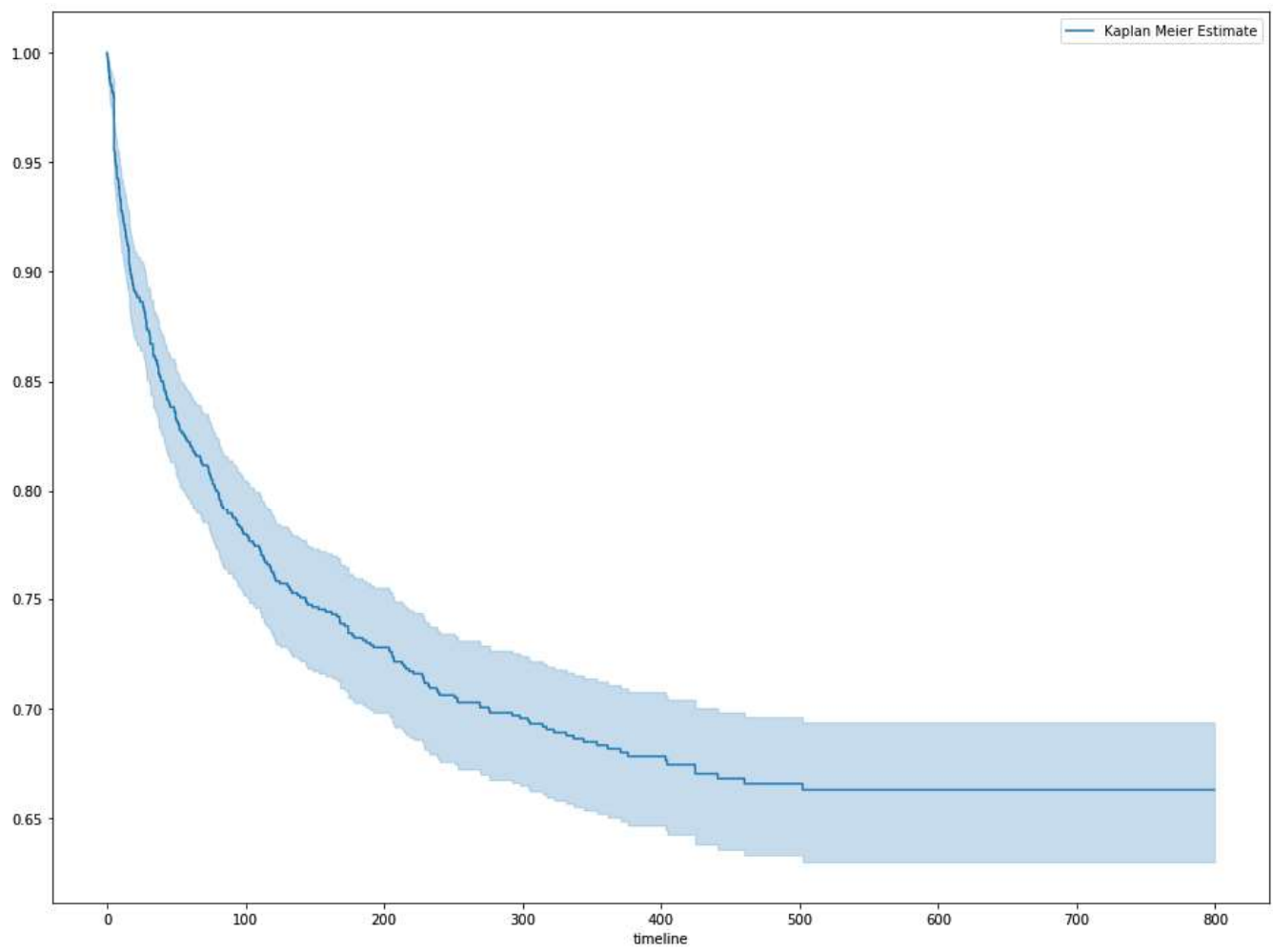
Index(['Asset_Number', 'ELR', 'High_Level_Asset_Class',
      'Asset_Class_Grouping',
      'Grouping_Full_Name', 'Engineering_Suffix', 'System_Asset_Type',
      'EQUIP_CLASS_DESC', 'birth', 'death', 'event', 'duration', 'T',
      'Start_Latitude', 'Start_Longitude', 'End_Latitude',
      'End_Longitude',
      'midpoint_Latitude', 'midpoint_Longitude', 'weather_station',
      'Sum'],
      dtype='object')

df.isnull().sum()
Asset_Number          0
ELR                   0
High_Level_Asset_Class 0
Asset_Class_Grouping  0
Grouping_Full_Name    0
Engineering_Suffix    0
System_Asset_Type     0
EQUIP_CLASS_DESC      0
birth                 0
death                 637
event                 0
duration              0
T                     0
Start_Latitude        0
Start_Longitude       0
End_Latitude          34
End_Longitude         34
midpoint_Latitude     34
midpoint_Longitude    34
weather_station       0
Sum                   293
dtype: int64

kmf = KaplanMeierFitter()
kmf.fit(df['T'], df['event'],label='Kaplan Meier Estimate')
<lifelines.KaplanMeierFitter:"Kaplan Meier Estimate",
fitted with 940 total observations, 637 right-censored observations>

## Create an estimate
plt.figure(figsize=(16,12))
kmf.plot(ci_show=True

```



KPF 4 cohorts using High\_Level\_Asset\_Class feature

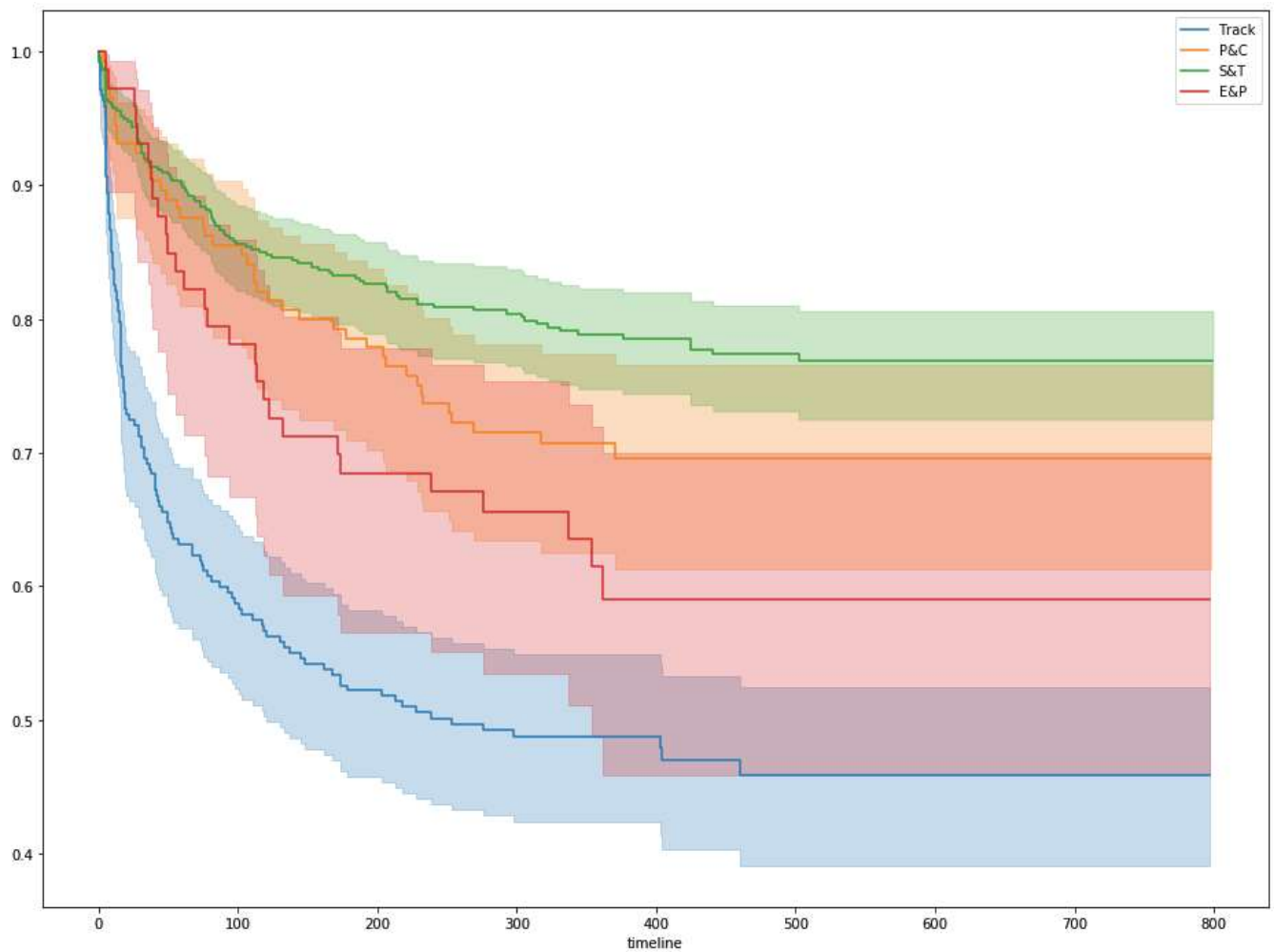
```
T=df['T']
E=df['event']
groups = df['High_Level_Asset_Class']
groups.unique()

#array(['Track', 'P&C', 'S&T', 'E&P'], dtype=object)

i1 = (groups == 'Track')      ## group i1 , having the pandas series
for the 1st cohort
i2 = (groups == 'P&C')        ## group i2 , having the pandas series  for
the 2nd cohort
i3 = (groups == 'S&T')        ## group i1 , having the pandas series  for
the 1st cohort
i4 = (groups == 'E&P')        ## group i2 , having the pandas series  for
the 2nd cohort

kmf1 = KaplanMeierFitter()

plt.figure(figsize=(16,12))
kmf1.fit(T[i1], E[i1], label='Track')
a1 = kmf1.plot()
kmf1.fit(T[i2], E[i2], label='P&C')
kmf1.plot(ax=a1)
kmf1.fit(T[i3], E[i3], label='S&T')
kmf1.plot(ax=a1)
kmf1.fit(T[i4], E[i4], label='E&P')
kmf1.plot(ax=a1)
```



### Estimating hazard rates using Nelson-Aalen

The survival functions is a great way to summarize and visualize the survival dataset, however it is not the only way. If we are curious about the hazard function  $h(t)$  of a population, we unfortunately cannot transform the Kaplan Meier estimate – statistics doesn't work quite that well. Fortunately, there is a proper non-parametric estimator of the *cumulative* hazard function:

$$H(t) = \int_0^t \lambda(z) dz$$

The estimator for this quantity is called the Nelson Aalen estimator:

$$\hat{H}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i}$$

where  $d_i$  is the number of deaths at time  $t_i$  and  $n_i$  is the number of susceptible individuals.

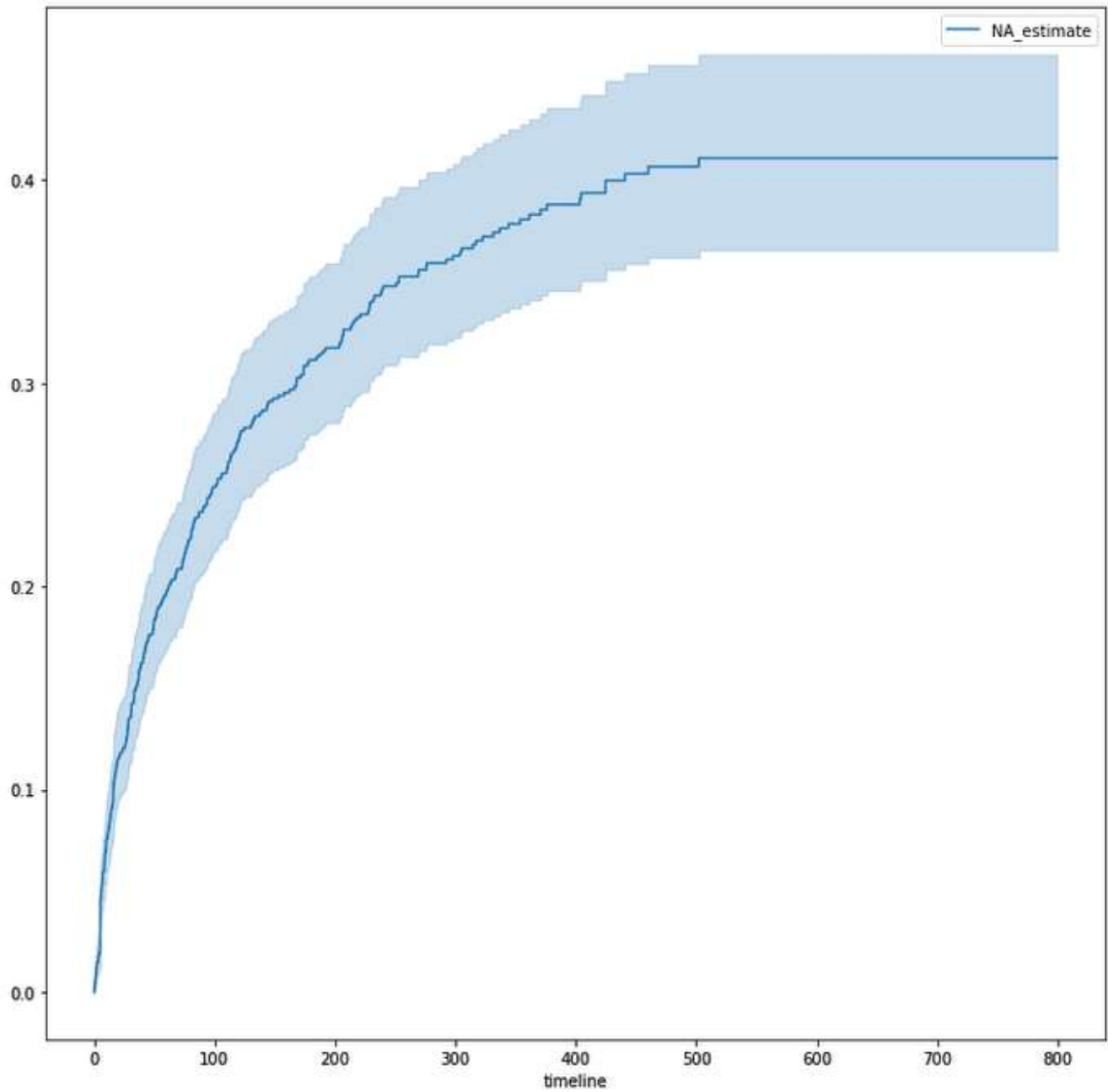


```
T=df['T']
E=df['event']

from lifelines import NelsonAalenFitter
naf = NelsonAalenFitter()

naf.fit(T,event_observed=E)
print(naf.cumulative_hazard_.head())
naf.plot(figsize=(12, 12))
```

	NA_estimate
timeline	
0.000000	0.000000
0.041667	0.002129
0.427778	0.003195
0.459028	0.004262
0.791435	0.005331



#### Other parametric models: Exponential, Log-Logistic, Log-Normal and Splines

Similarly, there are other parametric models in *lifelines*. Generally, which parametric model to choose is determined by either knowledge of the distribution of duration, or some sort of model goodness-of-fit. Below are the built-in parametric models, and the Nelson-Aalen non-parametric model, of the same data.

```

from lifelines import (WeibullFitter, ExponentialFitter,
LogNormalFitter, LogLogisticFitter, NelsonAalenFitter,
PiecewiseExponentialFitter, GeneralizedGammaFitter, SplineFitter)

from lifelines.datasets import load_waltons
data = load_waltons()

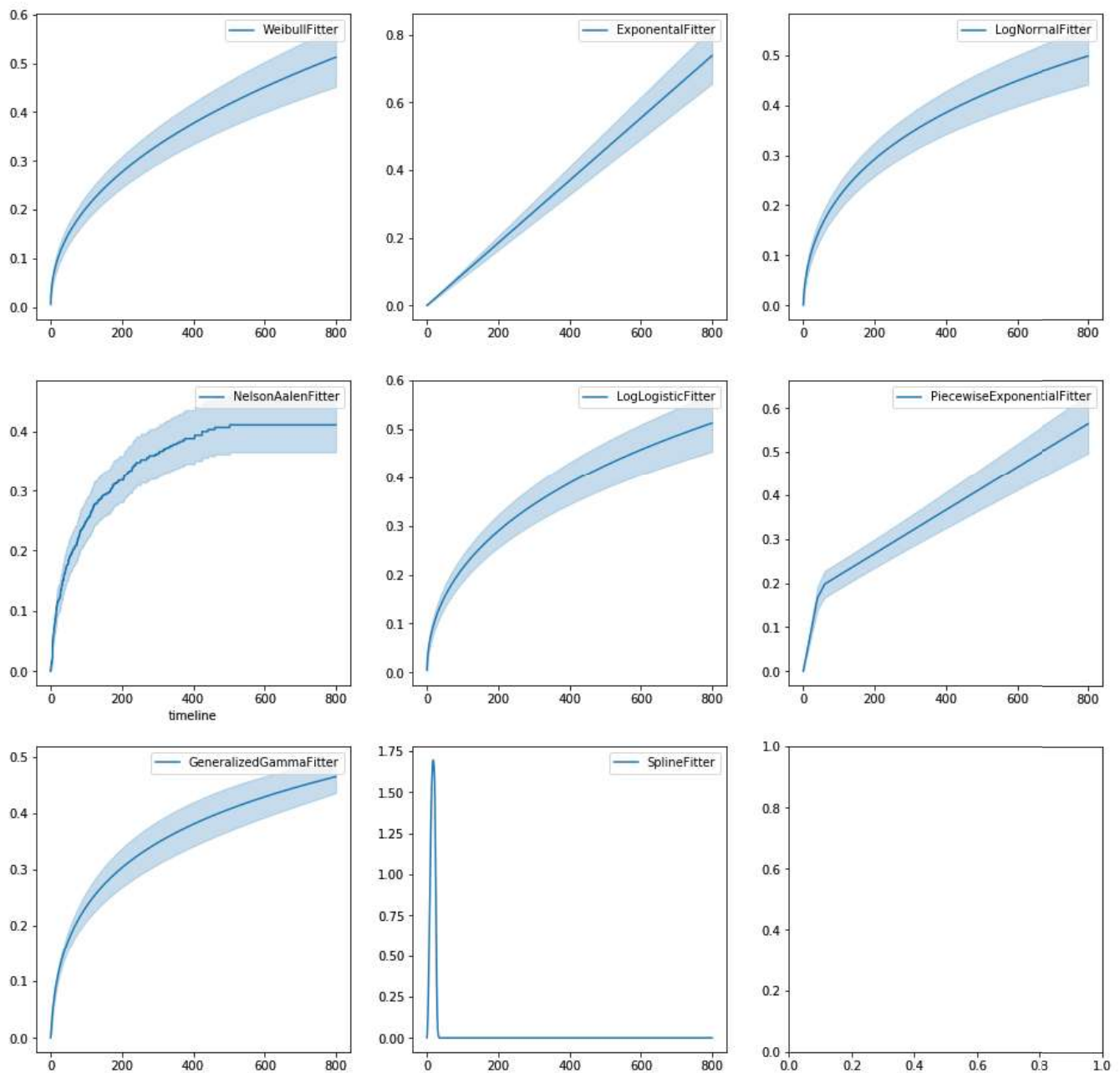
fig, axes = plt.subplots(3, 3, figsize=(16, 16))

T=df['T']
E=df['event']

wbf = WeibullFitter().fit(T, E, label='WeibullFitter')
exf = ExponentialFitter().fit(T, E, label='ExponentialFitter')
lnf = LogNormalFitter().fit(T, E, label='LogNormalFitter')
naf = NelsonAalenFitter().fit(T, E, label='NelsonAalenFitter')
llf = LogLogisticFitter().fit(T, E, label='LogLogisticFitter')
pwf = PiecewiseExponentialFitter([40, 60]).fit(T, E,
label='PiecewiseExponentialFitter')
gg = GeneralizedGammaFitter().fit(T, E, label='GeneralizedGammaFitter')
spf = SplineFitter([6, 20, 40, 75]).fit(T, E, label='SplineFitter')

wbf.plot_cumulative_hazard(ax=axes[0][0])
exf.plot_cumulative_hazard(ax=axes[0][1])
lnf.plot_cumulative_hazard(ax=axes[0][2])
naf.plot_cumulative_hazard(ax=axes[1][0])
llf.plot_cumulative_hazard(ax=axes[1][1])
pwf.plot_cumulative_hazard(ax=axes[1][2])
gg.plot_cumulative_hazard(ax=axes[2][0])
spf.plot_cumulative_hazard(ax=axes[2][1])

```



## Survival regression

Often we have additional data aside from the duration that we want to use. The technique is called *survival regression* – the name implies we regress covariates (e.g., age, country, etc.) against another variable – in this case duration. Similar to the logic in the first part of this tutorial, we cannot use traditional methods like linear regression because of censoring.

There are a few popular models in survival regression: Cox's model, accelerated failure models, and Aalen's additive model. All models attempt to represent the hazard rate  $h(t|x)h(t|x)$  as a function of  $t$  and some covariates  $xx$ . We explore these models next

### Cox's proportional hazard model

The idea behind Cox's proportional hazard model model is that the log-hazard of an individual is a linear function of their static covariates *and* a population-level baseline hazard that changes over time. Mathematically:

$$\underbrace{h(t|x)}_{\text{hazard}} = \underbrace{\widehat{b_0(t)}}_{\text{baseline hazard}} \underbrace{\exp\left(\sum_{i=1}^n b_i(x_i - \bar{x}_i)\right)}_{\text{partial hazard}}^{\text{log-partial hazard}}$$

## Jupyter Notebook



cox\_regression\_...ard Model.ipynb

```
df.columns
Index(['Asset_Number', 'ELR', 'High_Level_Asset_Class',
      'Asset_Class_Grouping',
      'Grouping_Full_Name', 'Engineering_Suffix', 'System_Asset_Type',
      'EQUIP_CLASS_DESC', 'birth', 'death', 'event', 'duration', 'T',
      'Start_Latitude', 'Start_Longitude', 'End_Latitude',
      'End_Longitude',
      'midpoint_Latitude', 'midpoint_Longitude', 'weather_station',
      'Sum'],
      dtype='object')

df_r=df[['event', 'T', 'Asset_Number','ELR', 'High_Level_Asset_Class',
      'Asset_Class_Grouping',
      'Grouping_Full_Name', 'Engineering_Suffix', 'System_Asset_Type',
      'EQUIP_CLASS_DESC','birth', 'death','Sum', 'weather_station']]

df_r['Sum'].fillna(0, inplace=True)
df_r.isnull().sum()

event          0
T              0
Asset_Number   0
ELR            0
High_Level_Asset_Class  0
Asset_Class_Grouping  0
```

```

Grouping_Full_Name          0
Engineering_Suffix          0
System_Asset_Type           0
EQUIP_CLASS_DESC            0
birth                        0
death                        637
Sum                           0
weather_station              0
dtype: int64

le1 = LabelEncoder()
le2 = LabelEncoder()
le3 = LabelEncoder()
le4 = LabelEncoder()
le6 = LabelEncoder()
le5 = LabelEncoder()
le7 = LabelEncoder()
le8 = LabelEncoder()
df_r['ELR_enc'] = le1.fit_transform(df_r['ELR'].astype(str).values)
df_r['HLC_enc'] =
le2.fit_transform(df_r['High_Level_Asset_Class'].astype(str).values)
df_r['Asset_Class_Grouping_enc'] =
le3.fit_transform(df_r['Asset_Class_Grouping'].astype(str).values)
df_r['Grouping_Full_Name_enc'] =
le4.fit_transform(df_r['Grouping_Full_Name'].astype(str).values)
df_r['Engineering_Suffix_enc'] =
le5.fit_transform(df_r['Engineering_Suffix'].astype(str).values)
df_r['System_Asset_Type_enc'] =
le6.fit_transform(df_r['System_Asset_Type'].astype(str).values)
df_r['EQUIP_CLASS_DESC_enc'] =
le7.fit_transform(df_r['EQUIP_CLASS_DESC'].astype(str).values)
df_r['weather_station_enc'] =
le8.fit_transform(df_r['weather_station'].astype(str).values)

df_dummy=df_r[['event', 'T', 'ELR_enc', 'HLC_enc',
                'Asset_Class_Grouping_enc', 'Grouping_Full_Name_enc',
                'Engineering_Suffix_enc', 'System_Asset_Type_enc',
                'EQUIP_CLASS_DESC_enc', 'weather_station_enc', 'Sum',
                'Asset_Number']]

# Using Cox Proportional Hazards model
cph = CoxPHFitter() ## Instantiate the class to create a cph object
cph.fit(df_dummy, 'T', event_col='event',cluster_col='Asset_Number')

```

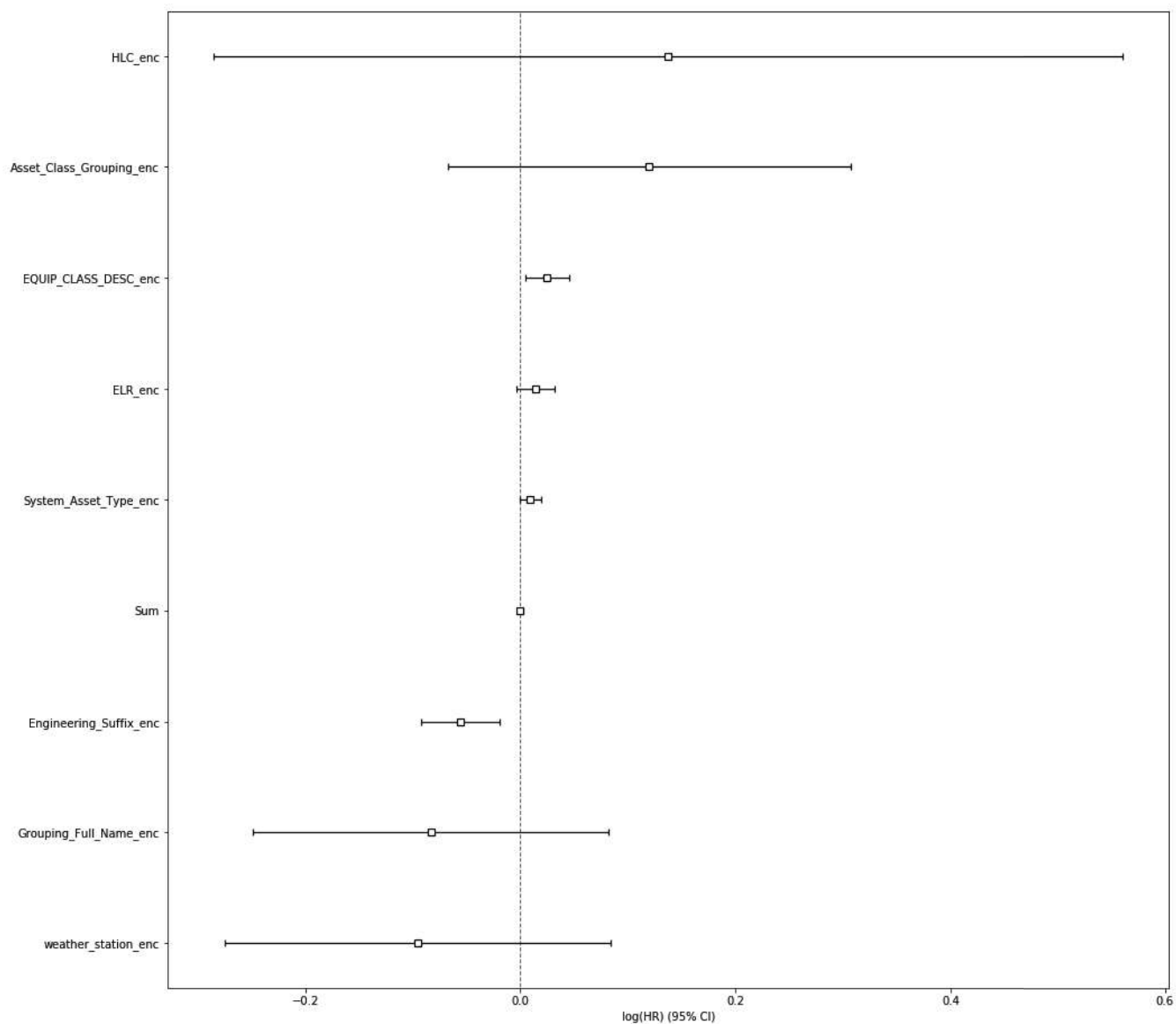
```
## Fit the data to train the model
cph.print_summary()
```

model	lifelines.CoxPHFitter
duration col	'T'
event col	'event'
cluster col	'Asset_Number'
robust variance	True
number of observations	940
number of events observed	303
partial log-likelihood	-1980.45
time fit was run	2020-02-18 11:20:00 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
ELR_enc	0.01	1.01	0.01	-0.00	0.03	1.00	1.03	1.54	0.12	3.02
HLC_enc	0.14	1.15	0.22	-0.29	0.56	0.75	1.75	0.64	0.52	0.93
Asset_Class_Grouping_enc	0.12	1.13	0.10	-0.07	0.31	0.94	1.36	1.26	0.21	2.26
Grouping_Full_Name_enc	-0.08	0.92	0.08	-0.25	0.08	0.78	1.09	-0.98	0.32	1.62
Engineering_Suffix_enc	-0.06	0.95	0.02	-0.09	-0.02	0.91	0.98	-3.01	<0.005	8.60
System_Asset_Type_enc	0.01	1.01	0.01	-0.00	0.02	1.00	1.02	1.79	0.07	3.75
EQUIP_CLASS_DESC_enc	0.03	1.03	0.01	0.00	0.05	1.00	1.05	2.43	0.01	6.06
weather_station_enc	-0.10	0.91	0.09	-0.27	0.08	0.76	1.09	-1.04	0.30	1.74
Sum	-0.00	1.00	0.00	-0.00	0.00	1.00	1.00	-0.38	0.71	0.50

Concordance	0.63
Log-likelihood ratio test	55.78 on 9 df, -log2(p)=26.79

Plotting the coefficients

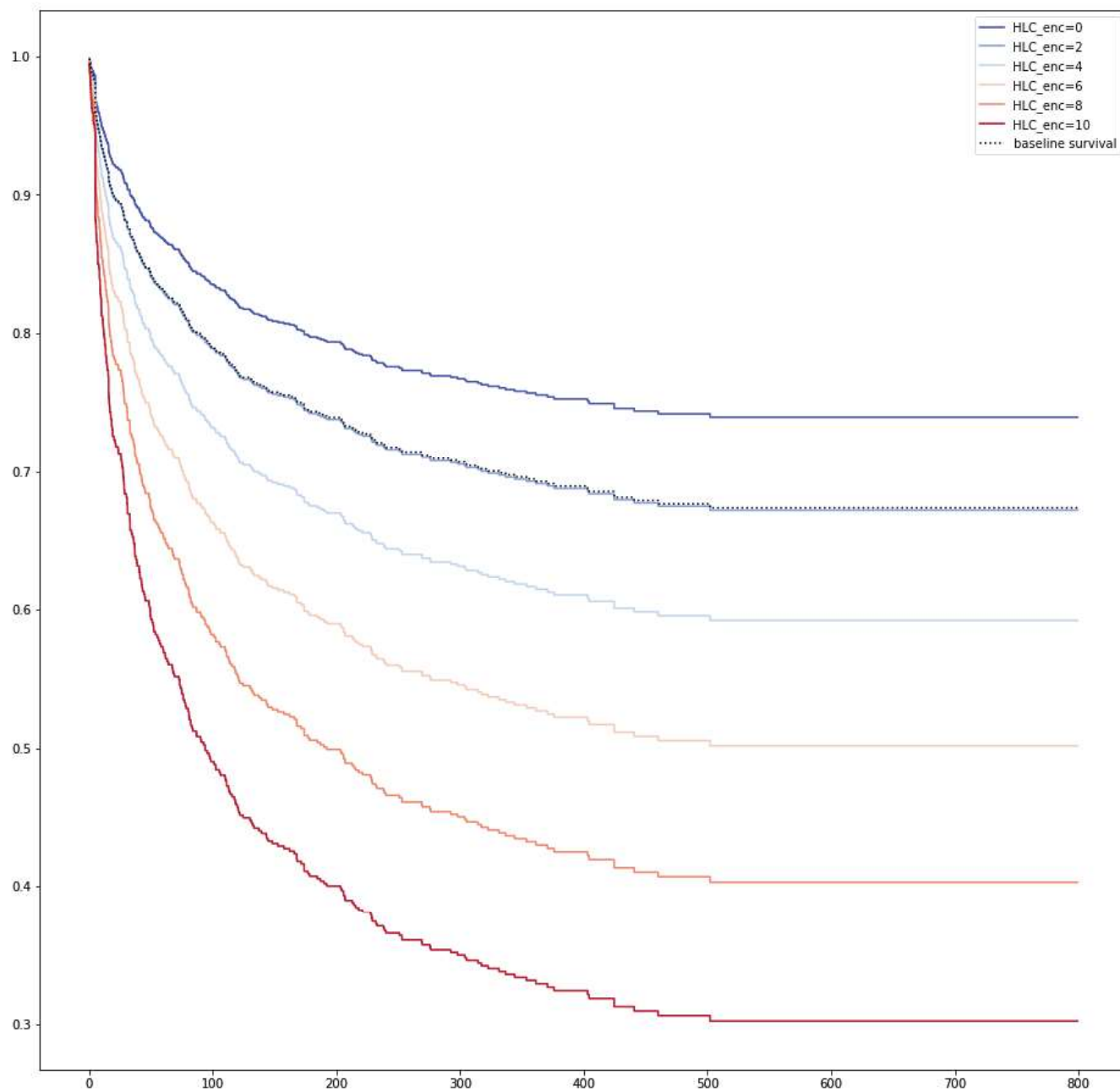


### Plotting the effect of varying a covariate

After fitting, we can plot what the survival curves look like as we vary a single covariate while holding everything else equal. This is useful to understand the impact of a covariate, *given the model*

```
plt.figure(figsize=(16,16))
cph.plot_covariate_groups('HLC_enc', [0, 2, 4, 6, 8, 10],
cmap='coolwarm',figsize=(16,16))
```





`cph.check_assumptions(df_dummy)`

To make proper inferences, we should ask if our Cox model is appropriate for our dataset. Recall from above that when using the Cox model, we are implicitly applying the proportional hazard assumption

The ``p\_value\_threshold`` is set at 0.01. Even under the null hypothesis of no violations, some covariates will be below the threshold by chance. This is compounded when there are many covariates. Similarly, when there are lots of observations, even minor deviances from the proportional hazard assumption will be flagged.

With that in mind, it's best to use a combination of statistical tests and visual tests to determine the most serious violations. Produce visual plots using ``check\_assumptions(..., show\_plots=True)`` and looking for non-constant lines. See link [A] below for a full example.

null_distribution		chi squared	
degrees_of_freedom		1	
test_name		proportional_hazard_test	
		test_statistic	p
Asset_Class_Grouping_enc	km	0.60	0.44
	rank	0.57	0.45
ELR_enc	km	0.09	0.77
	rank	0.08	0.78
EQUIP_CLASS_DESC_enc	km	0.03	0.86
	rank	0.02	0.89
Engineering_Suffix_enc	km	0.15	0.70
	rank	0.14	0.71
Grouping_Full_Name_enc	km	0.10	0.75
	rank	0.10	0.75
HLC_enc	km	0.38	0.54
	rank	0.48	0.49
Sum	km	3.84	0.05
	rank	3.68	0.05
System_Asset_Type_enc	km	2.03	0.15
	rank	1.87	0.17
weather_station_enc	km	13.63	<0.005
	rank	13.63	<0.005

1. Variable 'weather\_station\_enc' failed the non-proportional test: p-value is 0.0002.

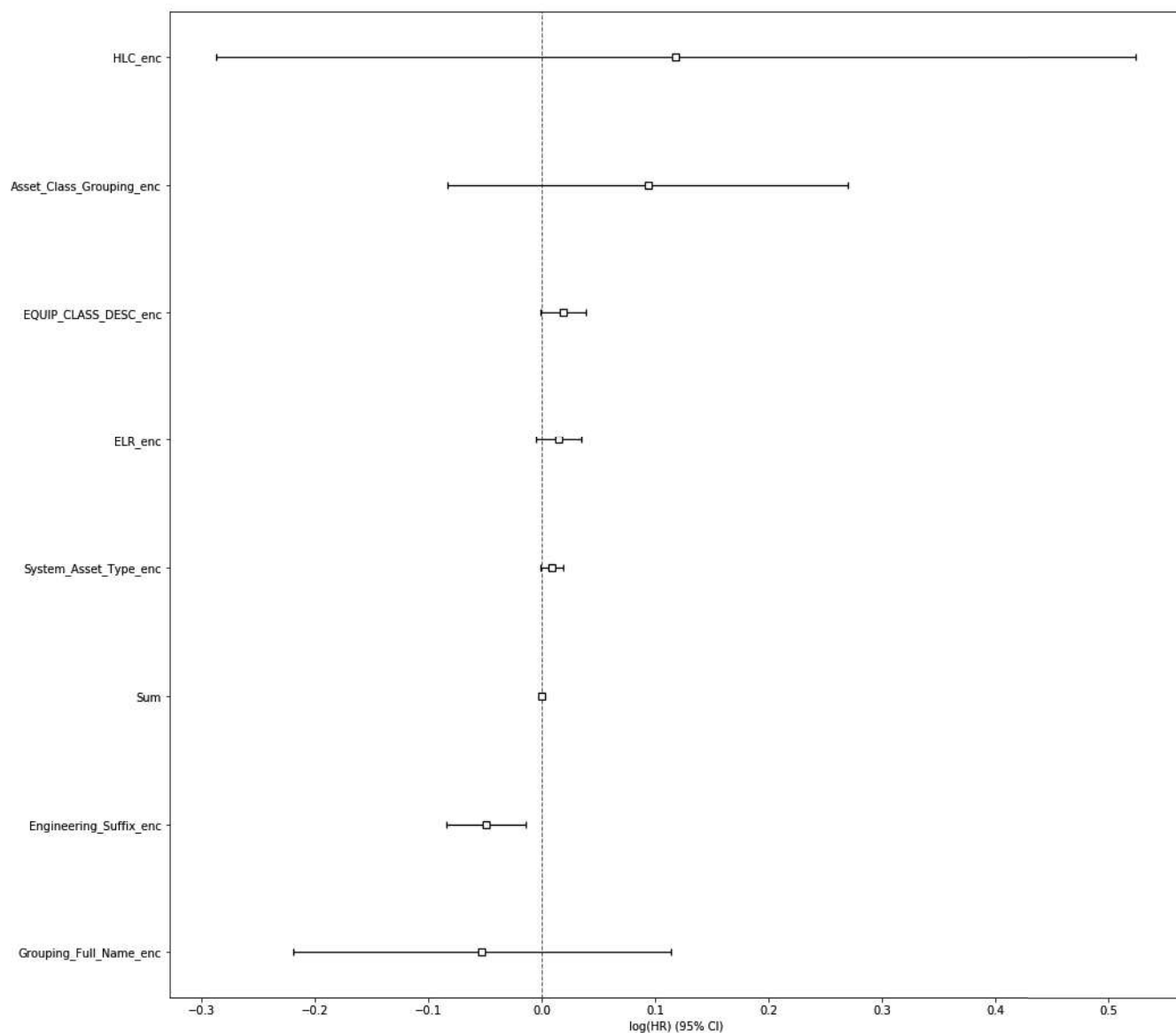
Advice: with so few unique values (only 5), you can include `strata=['weather\_station\_enc', ...]` in the call in `.fit`. See documentation in link [E] below.

```
# Using Cox Proportional Hazards model
cph = CoxPHFitter() ## Instantiate the class to create a cph object
cph.fit(df_dummy, 'T', event_col='event', cluster_col='Asset_Number',
strata=['weather_station_enc']) ## Fit the data to train the model
cph.print_summary()
```

model	lifelines.CoxPHFitter
duration col	'T'
event col	'event'
cluster col	'Asset_Number'
robust variance	True
strata	{weather_station_enc}
number of observations	940
number of events observed	303
partial log-likelihood	-1498.90
time fit was run	2020-02-18 11:33:48 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
ELR_enc	0.01	1.01	0.01	-0.01	0.03	0.99	1.04	1.44	0.15	2.73
HLC_enc	0.12	1.13	0.21	-0.29	0.52	0.75	1.69	0.57	0.57	0.82
Asset_Class_Grouping_enc	0.09	1.10	0.09	-0.08	0.27	0.92	1.31	1.04	0.30	1.74
Grouping_Full_Name_enc	-0.05	0.95	0.06	-0.22	0.11	0.80	1.12	-0.82	0.53	0.91
Engineering_Suffix_enc	-0.05	0.95	0.02	-0.08	-0.01	0.92	0.99	-2.73	0.01	7.32
System_Asset_Type_enc	0.01	1.01	0.01	-0.00	0.02	1.00	1.02	1.75	0.08	3.65
EQUIP_CLASS_DESC_enc	0.02	1.02	0.01	-0.00	0.04	1.00	1.04	1.82	0.07	3.88
Sum	-0.00	1.00	0.00	-0.00	0.00	1.00	1.00	-0.10	0.92	0.12

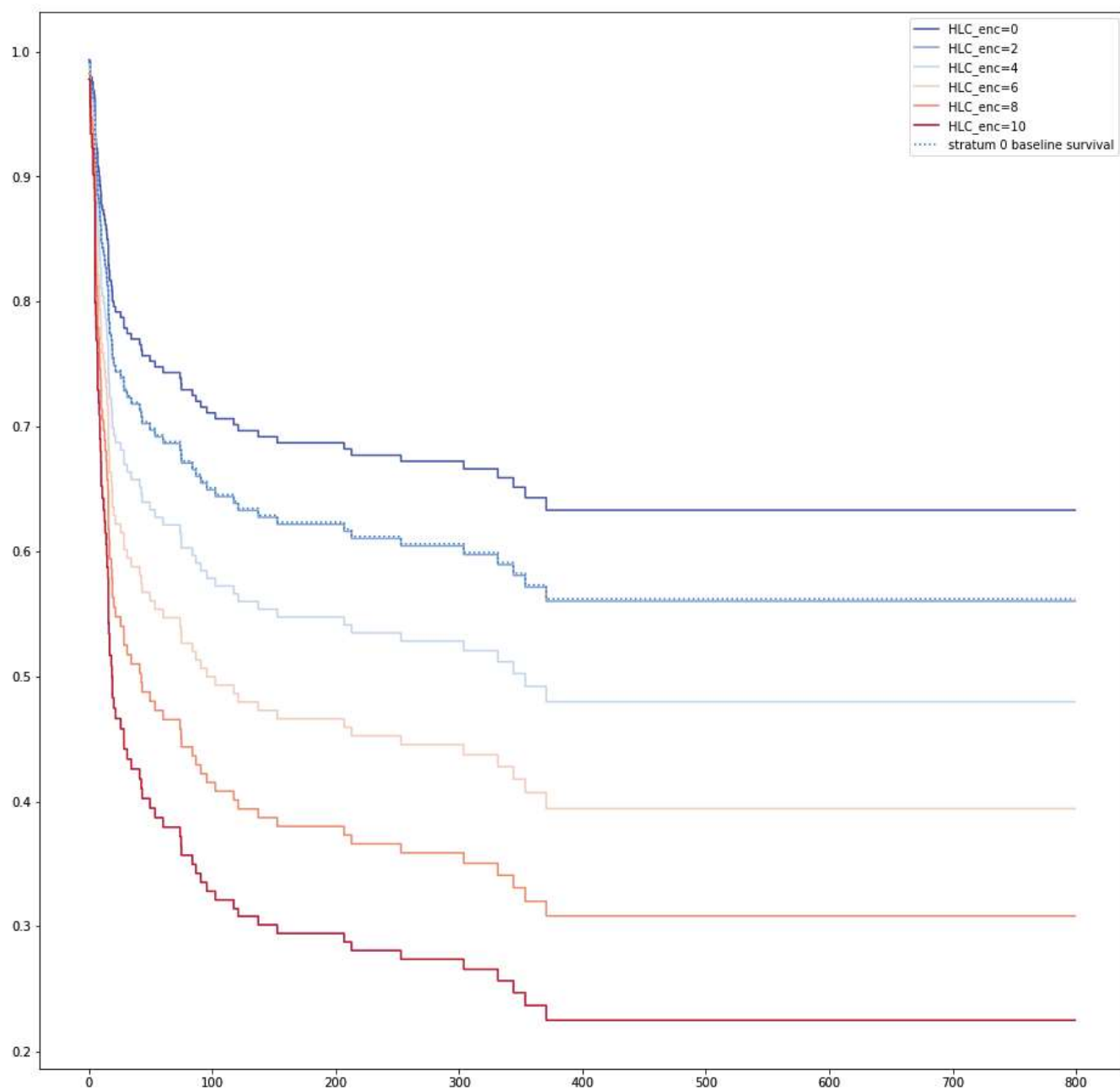
Concordance	0.61
Log-likelihood ratio test	37.13 on 8 df, -log2(p)=16.48

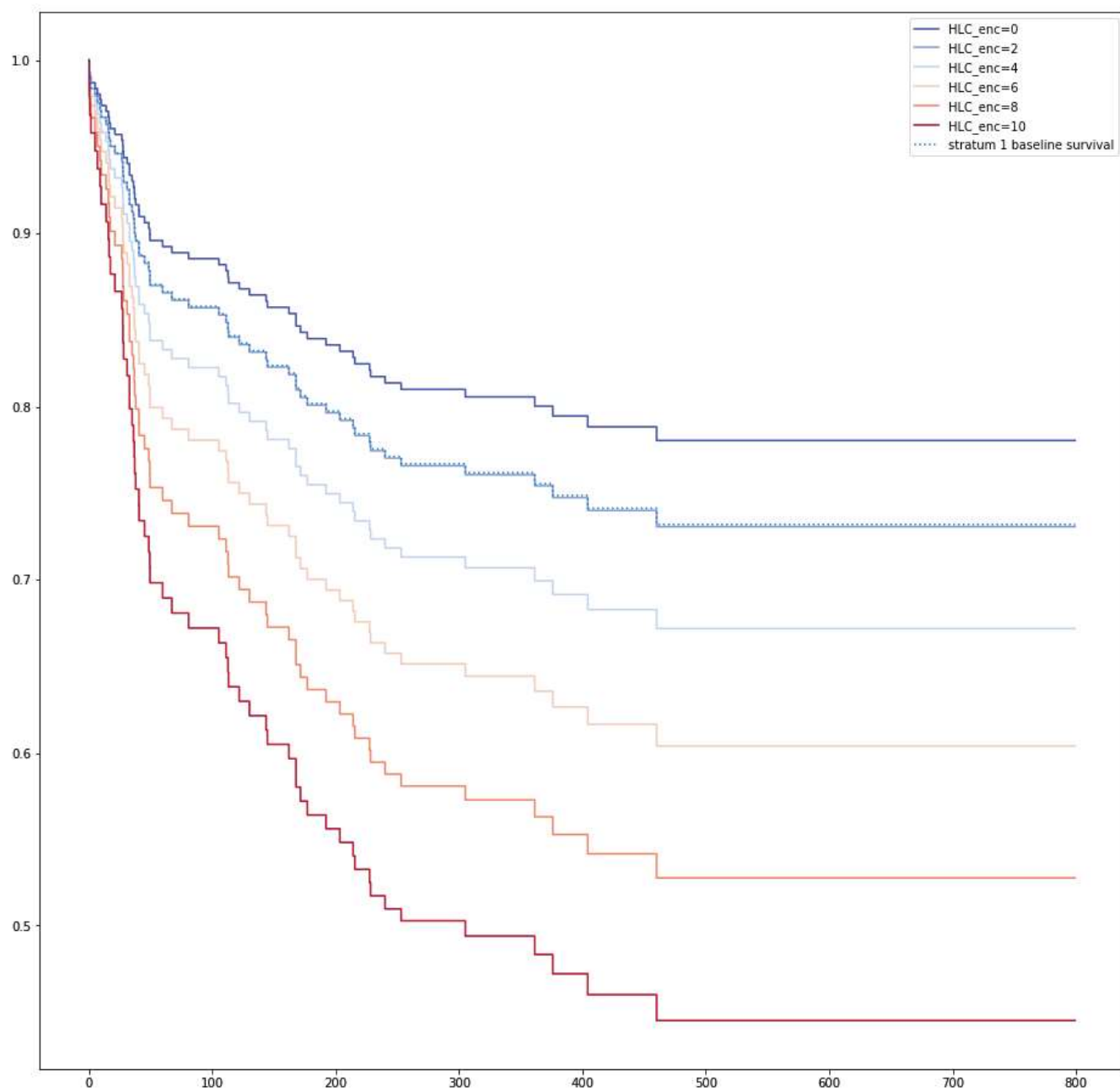


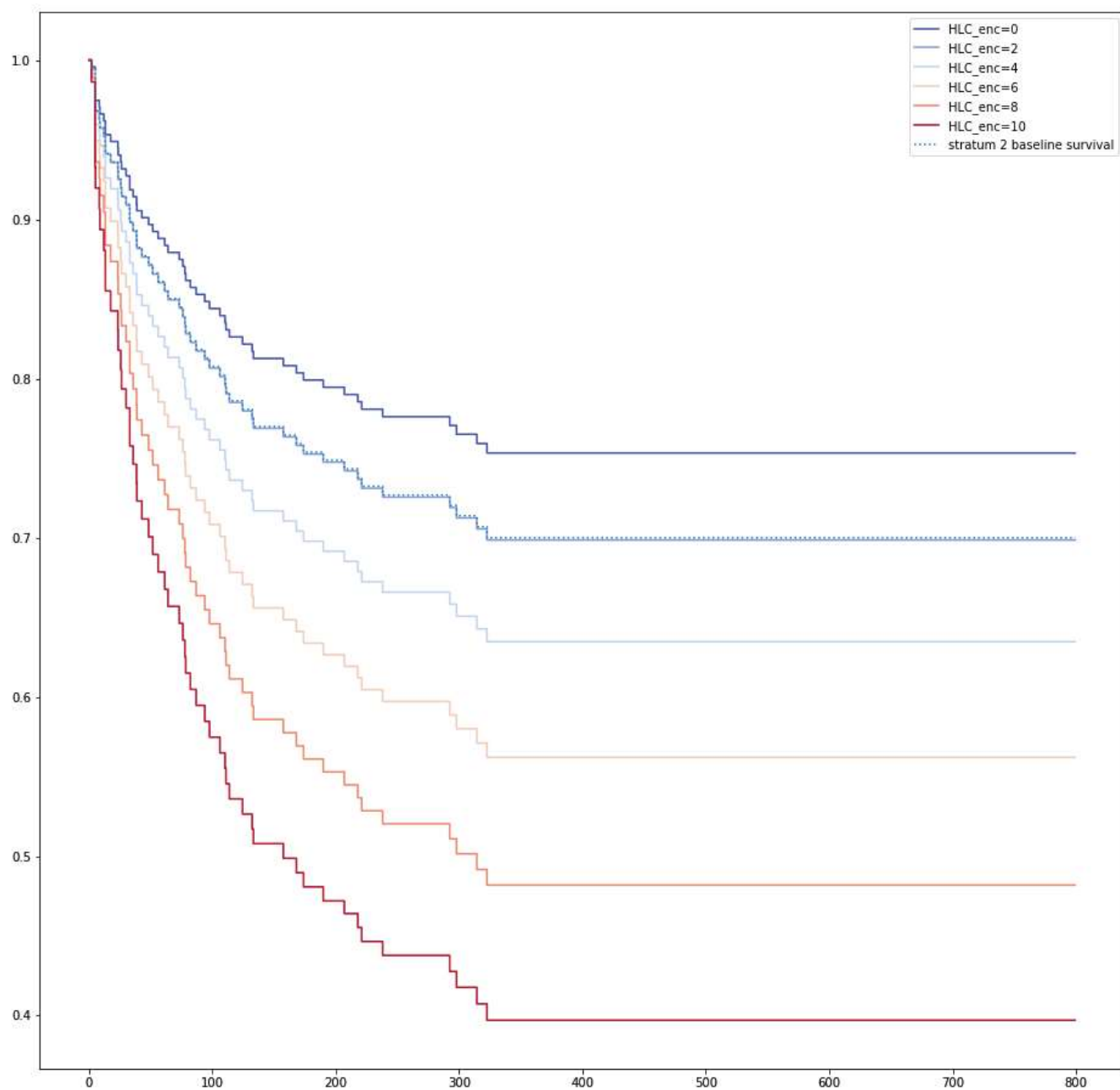
### Plot Covariate groups after stratification (weather\_station\_enc)

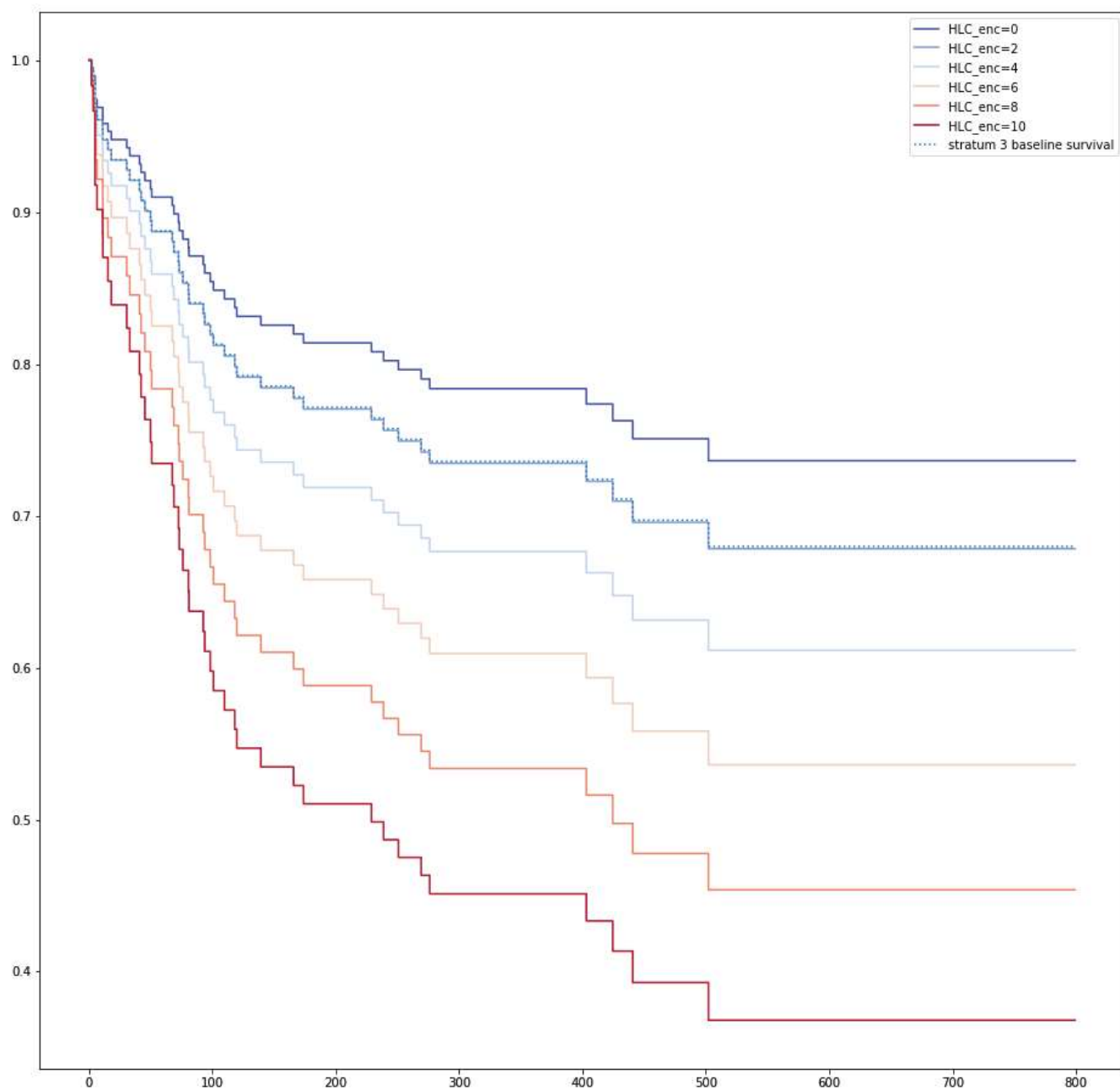
5 strata corresponding to the 5 weather locations use on the regression

```
['Leeds', 'York', 'Manchester', 'Mirfield', 'Huddersfield']
```

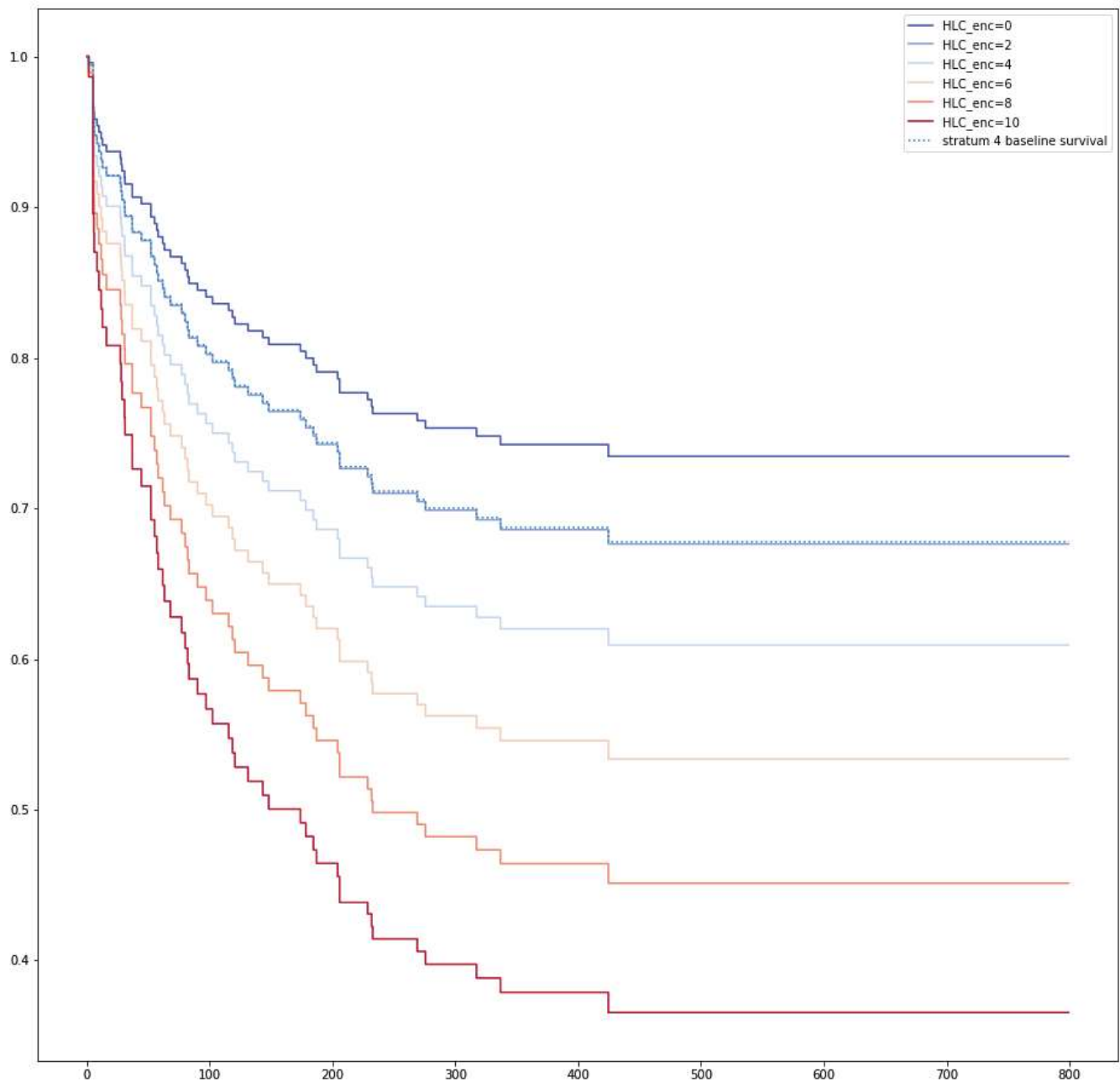












### Using Cox Proportional Hazards model. Clustering

we use the property clustering when a single individual having multiple occurrences, and hence showing up in the data-set more than once. In our case we do the clustering using **Asset number**

```
cph = CoxPHFitter()    ## Instantiate the class to create a cph object
cph.fit(df_dummy, 'T', event_col='event', cluster_col='Asset_Number',
strata=['weather_station_enc'])    ## Fit the data to train the model
cph.print_summary()
```

model	lifelines.CoxPHFitter											
duration col	'T'											
event col	'event'											
cluster col	'Asset_Number'											
robust variance	True											
strata	[weather_station_enc]											
number of observations	940											
number of events observed	303											
partial log-likelihood	-1498.90											
time fit was run	2020-02-19 08:59:42 UTC											
	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)		
ELR_enc	0.01	1.01	0.01	-0.01	0.03	0.99	1.04	1.44	0.15	2.73		
HLC_enc	0.12	1.13	0.21	-0.29	0.52	0.75	1.69	0.57	0.57	0.82		
Asset_Class_Grouping_enc	0.09	1.10	0.09	-0.08	0.27	0.92	1.31	1.04	0.30	1.74		
Grouping_Full_Name_enc	-0.05	0.95	0.08	-0.22	0.11	0.80	1.12	-0.62	0.53	0.91		
Engineering_Suffix_enc	-0.05	0.95	0.02	-0.08	-0.01	0.92	0.99	-2.73	0.01	7.32		
System_Asset_Type_enc	0.01	1.01	0.01	-0.00	0.02	1.00	1.02	1.75	0.08	3.65		
EQUIP_CLASS_DESC_enc	0.02	1.02	0.01	-0.00	0.04	1.00	1.04	1.82	0.07	3.88		
Sum	-0.00	1.00	0.00	-0.00	0.00	1.00	1.00	-0.10	0.92	0.12		
Concordance	0.61											
Log-likelihood ratio test	37.13 on 8 df, -log2(p)=16.48											

## Model selection based on predictive power

If censoring is present, it's not appropriate to use a loss function like mean-squared-error or mean-absolute-loss. Instead, one measure is the concordance-index, also known as the c-index. This measure evaluates the accuracy of the *ranking* of predicted time. It is in fact a generalization of AUC, another common loss function, and is interpreted similarly:

- 0.5 is the expected result from random predictions,
- 1.0 is perfect concordance and,
- 0.0 is perfect anti-concordance (multiply predictions with -1 to get 1.0)

our concordance-index on the baseline hazard model is

**cph.score\_** 0.6132267317407313

## Prediction

A common use case is to predict the event time of censored subjects. This is easy to do, but we first have to calculate an important conditional probability. Let  $T$  be the (random) event time for some subject, and  $S(t) := P(T > t)$  be their survival function. We are interested to answer the following: *What is a subject's new survival function given I know the subject has lived past time  $s$ ?* Mathematically:

$$\begin{aligned} P(T > t \mid T > s) &= \frac{P(T > t \text{ and } T > s)}{P(T > s)} \\ &= \frac{P(T > t)}{P(T > s)} \\ &= \frac{S(t)}{S(s)} \end{aligned}$$

Thus we scale the original survival function by the survival function at time  $s$  (everything prior to  $s$  should be mapped to 1.0 as well, since we are working with probabilities and we know that the subject was alive before  $s$ ).

Back to our original problem of predicting the event time of censored individuals, *lifelines* has all this math and logic built in when using the `conditional_after` kwarg.

as our data-set contains a column 'T' with number of days since birth we can predict the survival rate on different scale time frames

```
censored_subjects = df_dummy.loc[~df_dummy['event'].astype(bool)]
censored_subjects_last_obs = censored_subjects['T']
cph.predict_survival_function(censored_subjects, times=[0.04, 5., 25., 50.],
conditional_after=censored_subjects_last_obs)
```

We are interested specially for our purposes the survival probability on Time 0.04 days (next hour) and we can see mostly is 100%. (This study is limited to 11 March 2019)

	59	77	99	129	138	139	171	172	173	219	...	866	872	884	885	899	904	917	919	924	930
0.04	1.000000	1.000000	1.000000	1.000000	1.0	1.000000	1.0	1.0	1.0	1.000000	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	1.000000
5.00	1.000000	0.977985	1.000000	1.000000	1.0	1.000000	1.0	1.0	1.0	1.000000	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	1.000000
25.00	0.957980	0.953970	1.000000	0.981782	1.0	0.972036	1.0	1.0	1.0	0.984397	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	0.988259
50.00	0.934456	0.925593	0.979371	0.951300	1.0	0.944315	1.0	1.0	1.0	0.984397	...	1.0	1.0	1.0	1.0	1.0	0.994789	1.0	1.0	1.0	0.975485

4 rows × 637 columns

## Adding New features to Regression Model

Jupyter Notebook



cox\_regression\_v...r features.ipynb

We created 8 new features to the dataset considering the following requirements  
looking at the weather conditions of the close cities among the available on our dataset

```
['Leeds', 'York', 'Manchester', 'Mirfield', 'Huddersfield']
```

we calculate max and min of the following weather features:

Pressure, Wind\_speed, Pressure, temperature

on the following ranges

**for censored data**

birth date to 11/march /2019

**for non-censored data**

from birth to dead

Example calculate temperature min and max

```

def temp_min(event, death, end_e, weather, station, birth):
    if event == 1:
        if birth >= death:
            mask = (weather['dt_iso'] <= birth) & (weather['dt_iso'] >=
death ) & (weather['city_name'] == station)
            return weather[mask]['temp'].min()
        else:
            mask = (weather['dt_iso'] <= death) & (weather['dt_iso'] >=
birth ) & (weather['city_name'] == station)
            return weather[mask]['temp'].min()
    else:
        if birth >= end_e:

            mask2 = (weather['dt_iso'] <= birth) & (weather['dt_iso'] >=
end_e ) & (weather['city_name'] == station)
            return weather[mask2]['temp'].min()
        else:
            mask2 = (weather['dt_iso'] <= end_e) & (weather['dt_iso'] >=
birth ) & (weather['city_name'] == station)
            return weather[mask2]['temp'].min()

def temp_max(event, death, end_e, weather, station, birth):
    if event == 1:
        if birth >= death:
            mask = (weather['dt_iso'] <= birth) & (weather['dt_iso'] >=
death ) & (weather['city_name'] == station)
            return weather[mask]['temp'].max()
        else:
            mask = (weather['dt_iso'] <= death) & (weather['dt_iso'] >=
birth ) & (weather['city_name'] == station)
            return weather[mask]['temp'].max()
    else:
        if birth >= end_e:

            mask2 = (weather['dt_iso'] <= birth) & (weather['dt_iso'] >=
end_e ) & (weather['city_name'] == station)
            return weather[mask2]['temp'].max()
        else:
            mask2 = (weather['dt_iso'] <= end_e) & (weather['dt_iso'] >=
birth ) & (weather['city_name'] == station)
            return weather[mask2]['temp'].max()

```

adding the features to the data-set

```

tim1= datetime.datetime.now()
for i in data.index:
#     if i > 2:
#         break

    death= df_cox2.at[i,'death']
    #print(type(death))
    birth= df_cox2.at[i,'birth']
    event= df_cox2.at[i,'event']
    station= df_cox2.at[i,'weather_station']
    end_e= df_cox2.at[i,'end_e']
    # min max life period
    df_cox2.at[i,'temp_min']=temp_min(event, death, end_e, weather,
station, birth)
    df_cox2.at[i,'temp_max']=temp_max(event, death, end_e, weather,
station,birth)

    # min max life period
    df_cox2.at[i,'humidity_min']=humidity_min(event, death, end_e,
weather, station, birth)
    df_cox2.at[i,'humidity_max']=humidity_max(event, death, end_e,
weather, station,birth)

    # min max life period
    df_cox2.at[i,'pressure_min']=pressure_min(event, death, end_e,
weather, station, birth)
    df_cox2.at[i,'pressure_max']=pressure_max(event, death, end_e,
weather, station,birth)

    # min max life period
    df_cox2.at[i,'wind_speed_min']=wind_speed_min(event, death, end_e,
weather, station, birth)
    df_cox2.at[i,'wind_speed_max']=wind_speed_max(event, death, end_e,
weather, station,birth)
tim2= datetime.datetime.now()
print(tim2-tim1)

```

### Cox's proportional hazard model

```

cph = CoxPHFitter(penalizer=0.1)    ## Instantiate the class to create a
cph object
cph.fit(df_dummy, 'T', event_col='event',cluster_col='Asset_Number',
strata=['weather_station'])    ## Fit the data to train the model
cph.print_summary()

```

model	lifelines.CoxPHFitter
duration col	'T'
event col	'event'
cluster col	'Asset_Number'
penalizer	0.1
robust variance	True
strata	[weather_station]
number of observations	940
number of events observed	303
partial log-likelihood	-1152.47
time fit was run	2020-02-19 11:39:05 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
ELR_enc	0.03	1.03	0.01	0.00	0.05	1.00	1.05	2.21	0.03	5.21
HLC_enc	0.08	1.08	0.15	-0.22	0.38	0.80	1.46	0.51	0.61	0.71
Asset_Class_Grouping_enc	0.14	1.15	0.08	-0.02	0.30	0.98	1.35	1.76	0.08	3.67
Grouping_Full_Name_enc	-0.09	0.92	0.08	-0.25	0.08	0.78	1.08	-1.04	0.30	1.74
Engineering_Suffix_enc	-0.08	0.92	0.02	-0.11	-0.04	0.89	0.96	-4.42	<0.005	16.64
System_Asset_Type_enc	0.01	1.01	0.00	0.01	0.02	1.01	1.02	3.72	<0.005	12.29
EQUIP_CLASS_DESC_enc	0.00	1.00	0.01	-0.02	0.02	0.98	1.02	0.10	0.92	0.12
weather_station_enc	0.00	1.00	0.00	0.00	0.00	1.00	1.00	1.99	0.05	4.41
Sum	0.00	1.00	0.00	-0.00	0.00	1.00	1.00	1.72	0.08	3.56
temp_min	-0.01	0.99	0.03	-0.07	0.06	0.93	1.06	-0.24	0.81	0.30
temp_max	-0.05	0.95	0.02	-0.08	-0.01	0.92	0.99	-2.71	0.01	7.21
humidity_min	-0.02	0.98	0.01	-0.03	-0.00	0.97	1.00	-2.39	0.02	5.88
humidity_max	0.07	1.08	0.07	-0.07	0.22	0.93	1.24	1.00	0.32	1.65
pressure_min	0.01	1.01	0.02	-0.03	0.05	0.97	1.05	0.40	0.69	0.53
pressure_max	-0.21	0.81	0.02	-0.26	-0.16	0.77	0.85	-8.58	<0.005	56.50
wind_speed_min	-0.56	0.57	0.61	-1.76	0.64	0.17	1.90	-0.91	0.36	1.47
wind_speed_max	-0.05	0.95	0.07	-0.19	0.09	0.83	1.10	-0.66	0.51	0.97

Concordance	0.92
Log-likelihood ratio test	729.98 on 17 df, -log2(p)=476.48

```

cph.params_
ELR_enc          2.637444e-02
HLC_enc          7.765308e-02
Asset_Class_Grouping_enc 1.436312e-01
Grouping_Full_Name_enc -8.592828e-02
Engineering_Suffix_enc -7.942993e-02
System_Asset_Type_enc 1.479956e-02
EQUIP_CLASS_DESC_enc 1.077999e-03
weather_station_enc 1.159160e-13
Sum              4.369060e-04
temp_min         -7.886119e-03
temp_max         -4.911626e-02
humidity_min     -1.915675e-02
humidity_max      7.370109e-02
pressure_min     7.642093e-03
pressure_max     -2.133512e-01
wind_speed_min   -5.591996e-01
wind_speed_max   -4.787618e-02

```

```
cph.baseline_survival_
```

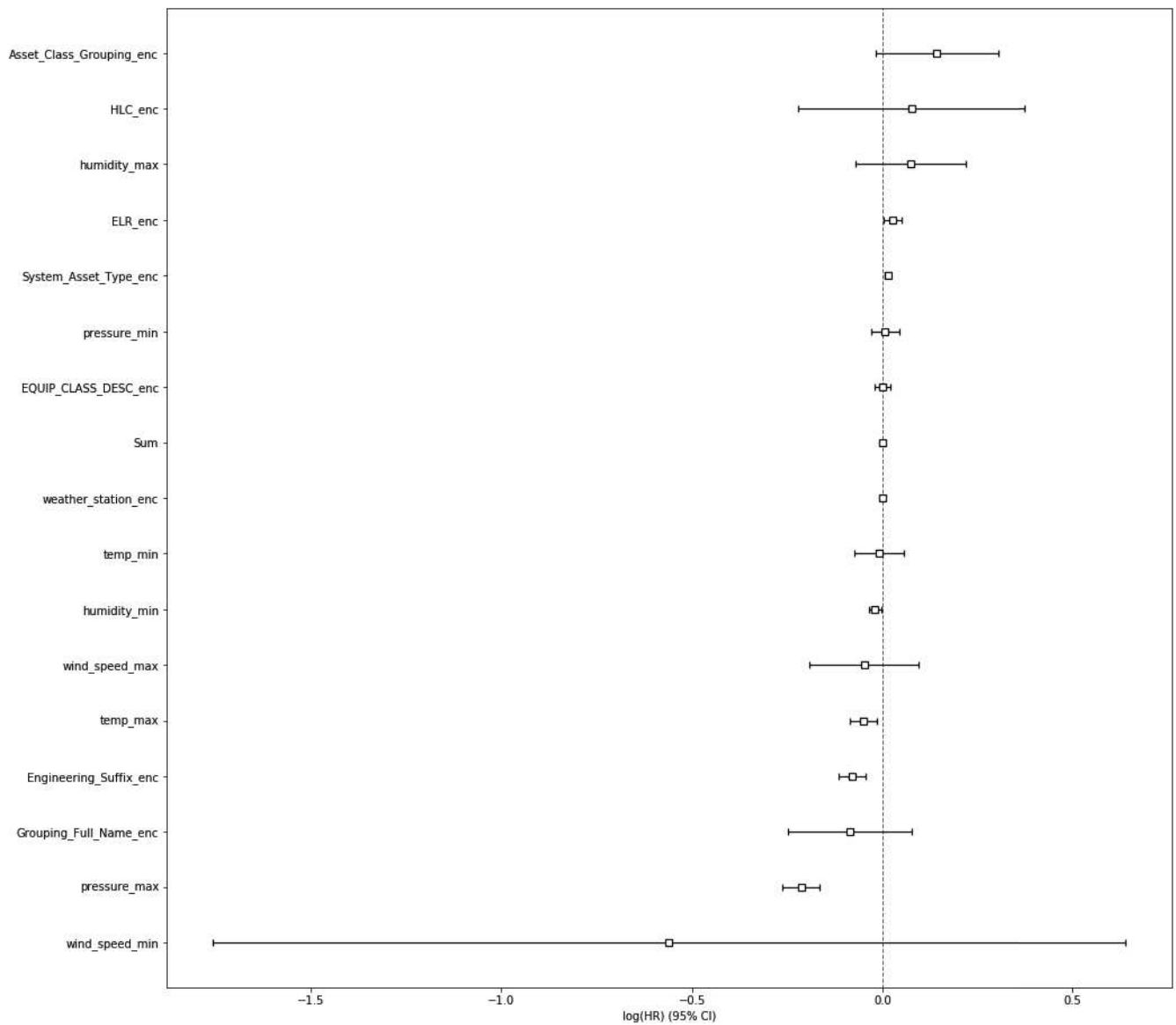
	Huddersfield	Leeds	Manchester	Mirfield	York
0.041667	0.999301	1.000000	1.000000	1.000000	1.000000
0.427778	0.999301	0.998818	1.000000	1.000000	1.000000
0.459028	0.999301	0.997615	1.000000	1.000000	1.000000
0.791435	0.999301	0.996244	1.000000	1.000000	1.000000
0.942361	0.998710	0.996244	1.000000	1.000000	1.000000
...	...	...	...	...	...
796.970777	0.686204	0.716675	0.701583	0.69596	0.726256
797.092303	0.686204	0.716675	0.701583	0.69596	0.726256
797.158276	0.686204	0.716675	0.701583	0.69596	0.726256
797.999005	0.686204	0.716675	0.701583	0.69596	0.726256
799.324942	0.686204	0.716675	0.701583	0.69596	0.726256

915 rows × 5 columns



```
cph.score_  
0.9182185974826612
```

```
plt.figure(figsize=(16,16))  
cph.plot()
```



survival predictions

```

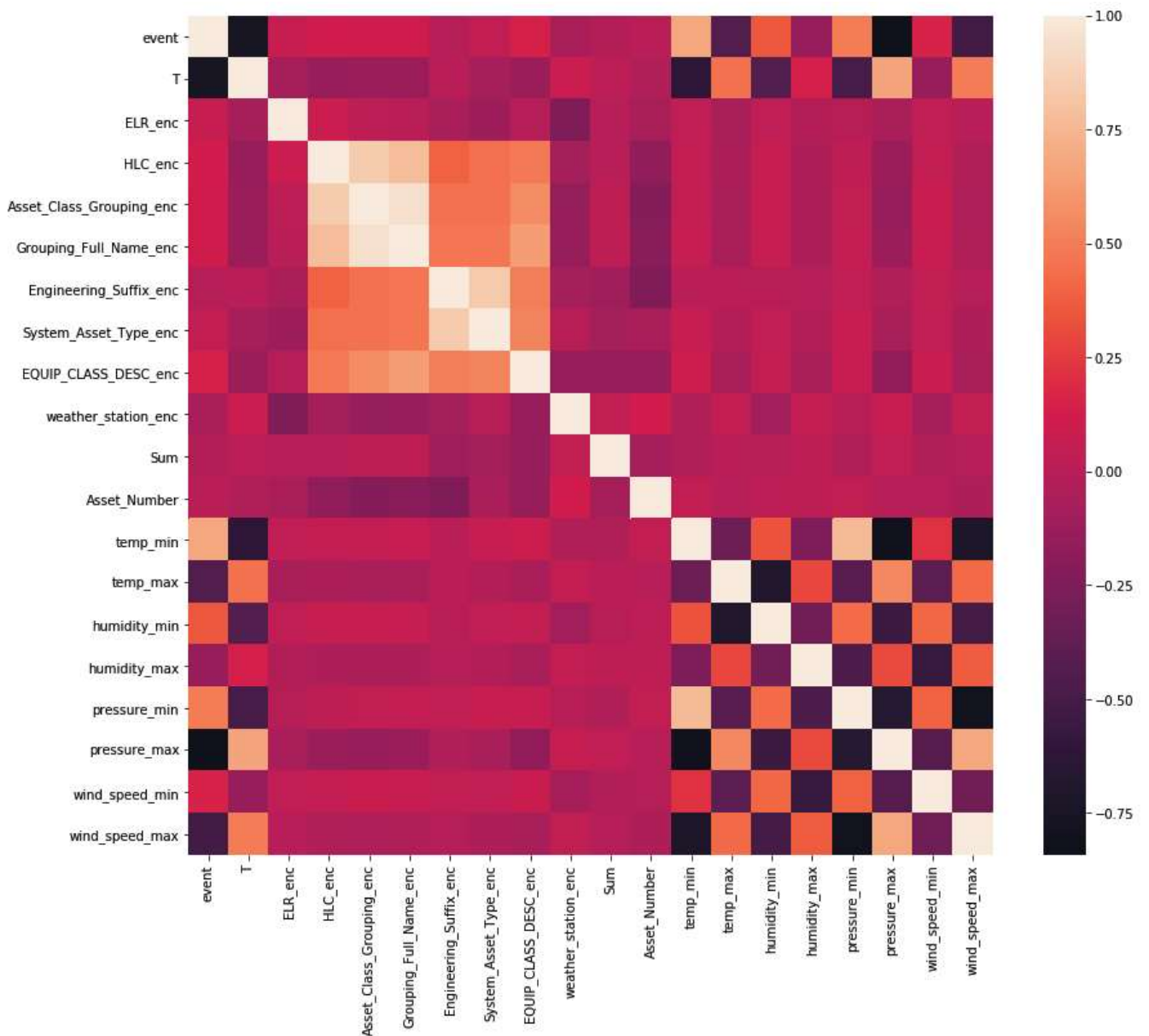
censored_subjects = df_dummy.loc[~df_dummy['event'].astype(bool)]
censored_subjects_last_obs = censored_subjects['T']
cph.predict_survival_function(censored_subjects, times=[0.04, 5., 25.,
50.], conditional_after=censored_subjects_last_obs)

```

	59	77	99	129	138	139	171	172	173	219	...	866	872	884	885	899	904	917	919	924	930
0.04	1.000000	1.000000	1.000000	1.000000	1.0	1.000000	1.0	1.0	1.0	1.000000	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	1.000000
5.00	1.000000	0.986940	1.000000	1.000000	1.0	1.000000	1.0	1.0	1.0	1.000000	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	1.000000
25.00	0.975269	0.971592	1.000000	0.993606	1.0	0.987419	1.0	1.0	1.0	0.991226	...	1.0	1.0	1.0	1.0	1.0	1.000000	1.0	1.0	1.0	0.987716
50.00	0.960102	0.952417	0.992262	0.974953	1.0	0.973166	1.0	1.0	1.0	0.991226	...	1.0	1.0	1.0	1.0	1.0	0.994839	1.0	1.0	1.0	0.974244

4 rows × 637 columns

Correlation Matrix



## Deep Survival Neural Network (DSNN)

### Jupyter Notebook



Loss Function: negative log of Breslow Approximation partial likelihood function

[https://en.wikipedia.org/wiki/Proportional\\_hazards\\_model](https://en.wikipedia.org/wiki/Proportional_hazards_model)



#### Tied times [ edit ]

Several approaches have been proposed to handle situations in which there are ties in the time data. *Breslow's method* describes the approach in which the procedure described above is used unmodified, even when ties are present. An alternative approach that is considered to give better results is *Efron's method*<sup>[6]</sup>. Let  $t_j$  denote the unique times, let  $H_j$  denote the set of indices  $i$  such that  $Y_i = t_j$  and  $C_i = 1$ , and let  $m_j = |H_j|$ . Efron's approach maximizes the following partial likelihood.

$$L(\beta) = \prod_j \frac{\prod_{i \in H_j} \theta_i}{\prod_{\ell=0}^{m-1} \left[ \sum_{i: Y_i \geq t_j} \theta_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i \right]}.$$

The corresponding log partial likelihood is

$$\ell(\beta) = \sum_j \left( \sum_{i \in H_j} X_i \cdot \beta - \sum_{\ell=0}^{m-1} \log \left( \sum_{i: Y_i \geq t_j} \theta_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i \right) \right),$$

the score function is

$$\ell'(\beta) = \sum_j \left( \sum_{i \in H_j} X_i - \sum_{\ell=0}^{m-1} \frac{\sum_{i: Y_i \geq t_j} \theta_i X_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i X_i}{\sum_{i: Y_i \geq t_j} \theta_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i} \right),$$

and the Hessian matrix is

$$\ell''(\beta) = - \sum_j \sum_{\ell=0}^{m-1} \left( \frac{\sum_{i: Y_i \geq t_j} \theta_i X_i X_i' - \frac{\ell}{m} \sum_{i \in H_j} \theta_i X_i X_i'}{\phi_{j,\ell,m}} - \frac{Z_{j,\ell,m} Z_{j,\ell,m}'}{\phi_{j,\ell,m}^2} \right),$$

where

$$\begin{aligned} \phi_{j,\ell,m} &= \sum_{i: Y_i \geq t_j} \theta_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i \\ Z_{j,\ell,m} &= \sum_{i: Y_i \geq t_j} \theta_i X_i - \frac{\ell}{m} \sum_{i \in H_j} \theta_i X_i. \end{aligned}$$

Note that when  $H_j$  is empty (all observations with time  $t_j$  are censored), the summands in these expressions are treated as zero.



```

def _create_loss(self):
    """
    Define the loss function.

    Notes
    -----
    The negative log of Breslow Approximation partial
    likelihood function. See more in "Breslow N.. See more in
    "Breslow N., 'Covariance analysis of censored
    survival data, ' Biometrics 30.1(1974):89-99.".
    """
    with tf.name_scope("loss"):
        # Obtain T and E from self.Y
        # NOTE: negative value means E = 0
        Y_c = tf.squeeze(self.Y)
        Y_hat_c = tf.squeeze(self.Y_hat)
        Y_label_T = tf.abs(Y_c)
        Y_label_E = tf.cast(tf.greater(Y_c, 0), dtype=tf.float32)
        Obs = tf.reduce_sum(Y_label_E)

        Y_hat_hr = tf.exp(Y_hat_c)
        Y_hat_cumsum = tf.log(tf.cumsum(Y_hat_hr))

        # Start Computation of Loss function

        # Get Segment from T
        unique_values, segment_ids = tf.unique(Y_label_T)
        # Get Segment_max
        loss_s2_v = tf.segment_max(Y_hat_cumsum, segment_ids)
        # Get Segment_count
        loss_s2_count = tf.segment_sum(Y_label_E, segment_ids)
        # Compute S2
        loss_s2 = tf.reduce_sum(tf.multiply(loss_s2_v,
loss_s2_count))
        # Compute S1
        loss_s1 = tf.reduce_sum(tf.multiply(Y_hat_c, Y_label_E))
        # Compute Breslow Loss
        loss_breslow = tf.divide(tf.subtract(loss_s2, loss_s1), Obs)

        # Compute Regularization Term Loss
        reg_item =
tf.contrib.layers.l1_l2_regularizer(self.config["L1_reg"],
self.config["L2_reg"])
        loss_reg = tf.contrib.layers.apply_regularization(reg_item,
tf.get_collection("var_weight"))

        # Loss function = Breslow Function + Regularization Term
        self.loss = tf.add(loss_breslow, loss_reg)

```

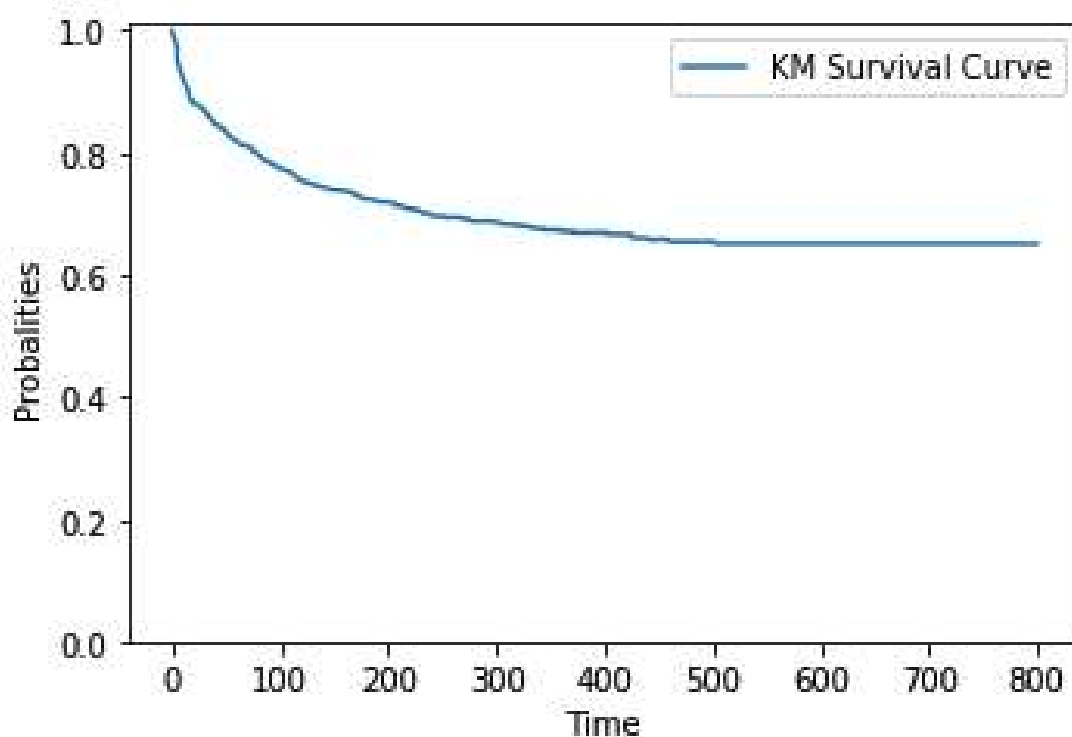
```

train_data, test_data = train_test_split(datan, shuffle =True)
train_data.shape, test_data.shape
((705, 20), (235, 20))

plt.figure(figsize=(12,12))
survival_stats(train_data, t_col=colname_t, e_col=colname_e, plot=True)

----- Survival Data Statistics -----
# Rows: 705
# Columns: 18 + e + t
# Events Ratio: 0.33%
# Min Time: 0.041666667
# Max Time: 799.324942

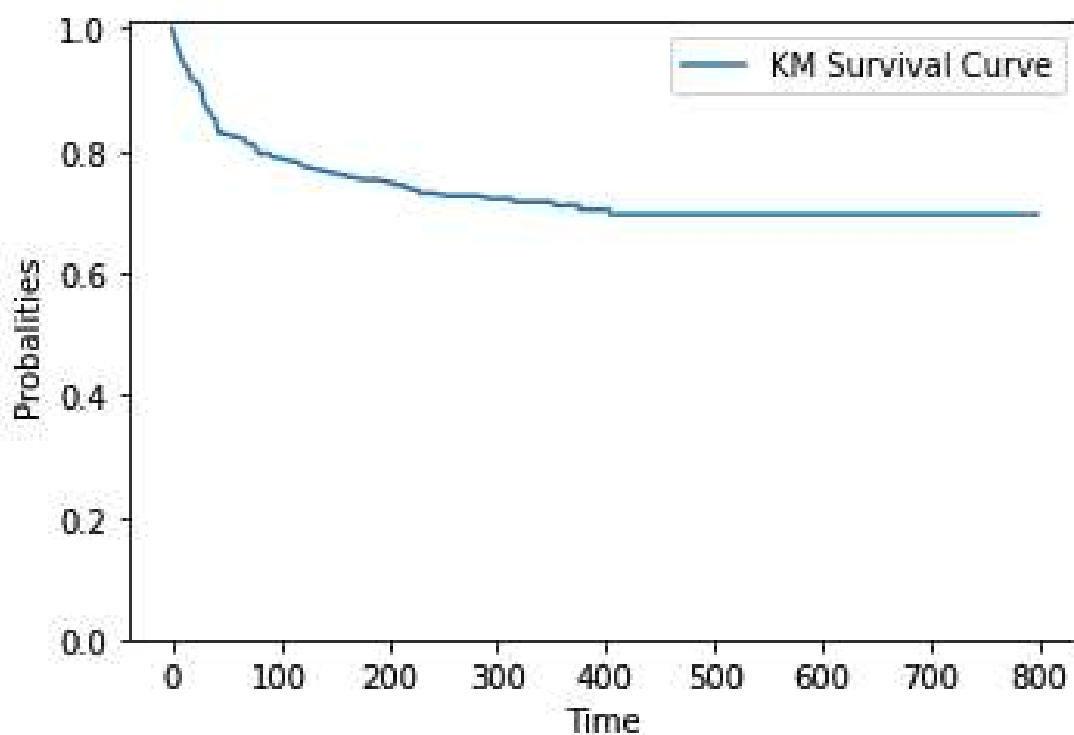
```



	0	100	200	300	400	500	600	700	800
At risk	705	544	499	407	263	171	144	70	0

```
plt.figure(figsize=(12,12))
survival_stats(test_data, t_col=colname_t, e_col=colname_e, plot=True)
```

```
----- Survival Data Statistics -----
# Rows: 235
# Columns: 18 + e + t
# Events Ratio: 0.29%
# Min Time: 0.459027778
# Max Time: 797.0923029
```



	At risk								
KM Survival Curve	235	185	173	145	94	67	60	27	0

```

from tfdeepsurv import dsnn

# Number of features in your dataset
input_nodes = len(surv_train.columns) - 1
# Specify your neural network structure
hidden_layers_nodes = [6, 3, 1]

# the arguments of dsnn can be obtained by Bayesian Hyperparameters
Tuning.
# It would affect your model performance largely!
nn_config = {
    "learning_rate": 0.07,
    "learning_rate_decay": 1.0,
    "activation": 'tanh',
    "L1_reg": 3.4e-5,
    "L2_reg": 8.8e-5,
    "optimizer": 'sgd',
    "dropout_keep_prob": 1.0,
    "seed": 1
}

# ESSENTIAL STEP-1: Pass arguments
model = dsnn(
    input_nodes,
    hidden_layers_nodes,
    nn_config
)

# ESSENTIAL STEP-2: Build Computation Graph
model.build_graph()

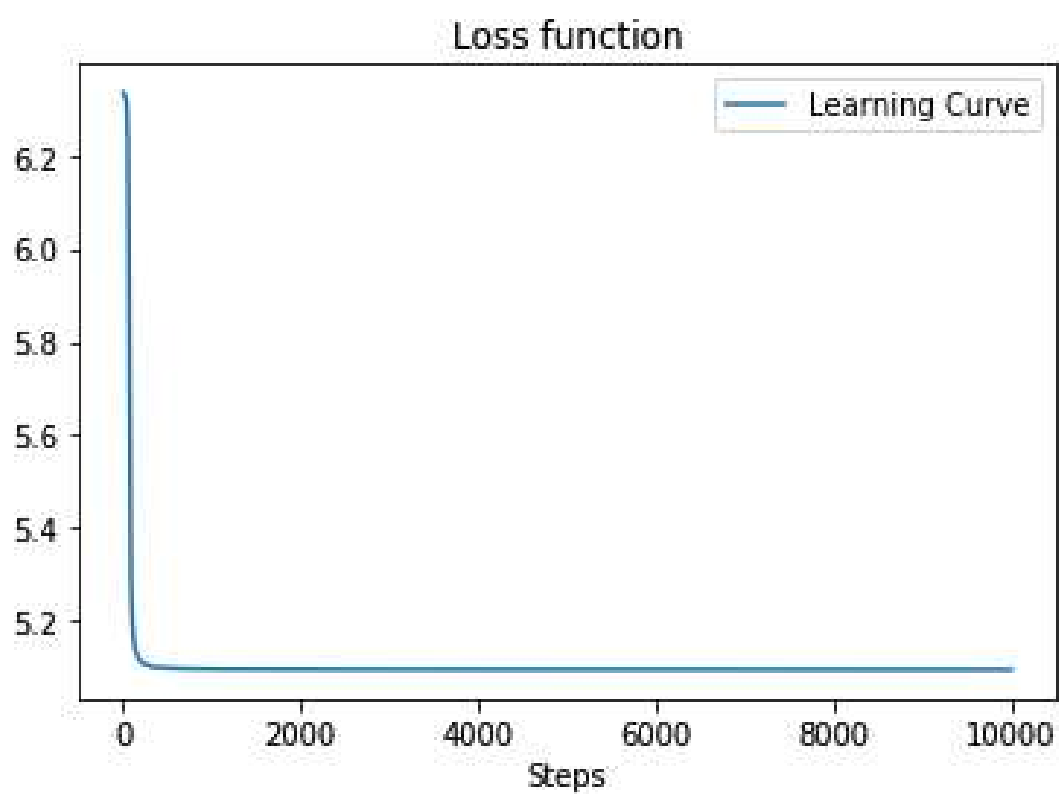
Y_col = ["Y"]
X_cols = [c for c in surv_train.columns if c not in Y_col]

# model saving and loading is also supported!
# read comments of `train()` function if necessary.

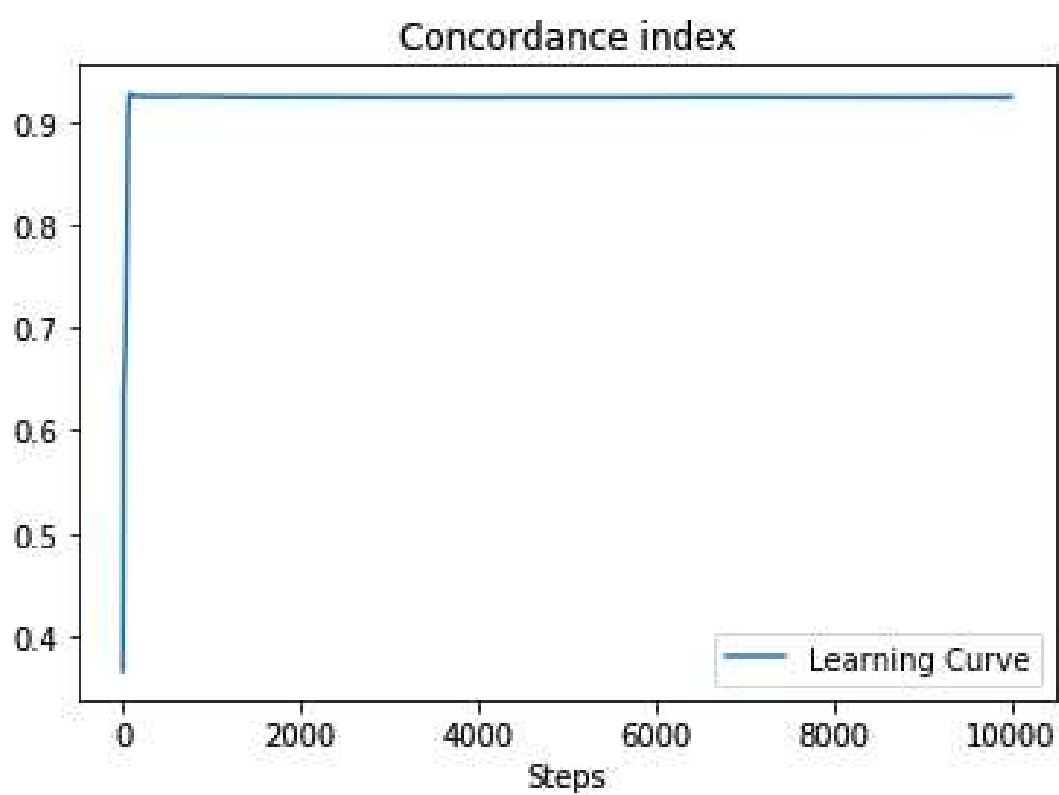
# ESSENTIAL STEP-3: Train Model
# `num_steps` is also a important parameters
watch_list = model.train(
    surv_train[X_cols], surv_train[Y_col],
    num_steps=10000,
    num_skip_steps=100,
    plot=True,
    save_model='./model/afe'
)

```

Learning Curve



Concordance Index





```
print("CI on training data:", model.evals(surv_train[X_cols],
surv_train[Y_col]))
print("CI on test data:", model.evals(surv_test[X_cols],
surv_test[Y_col]))
```

CI on training data: 0.9243192958335824

CI on test data: 0.9469242727070841

## Prediction

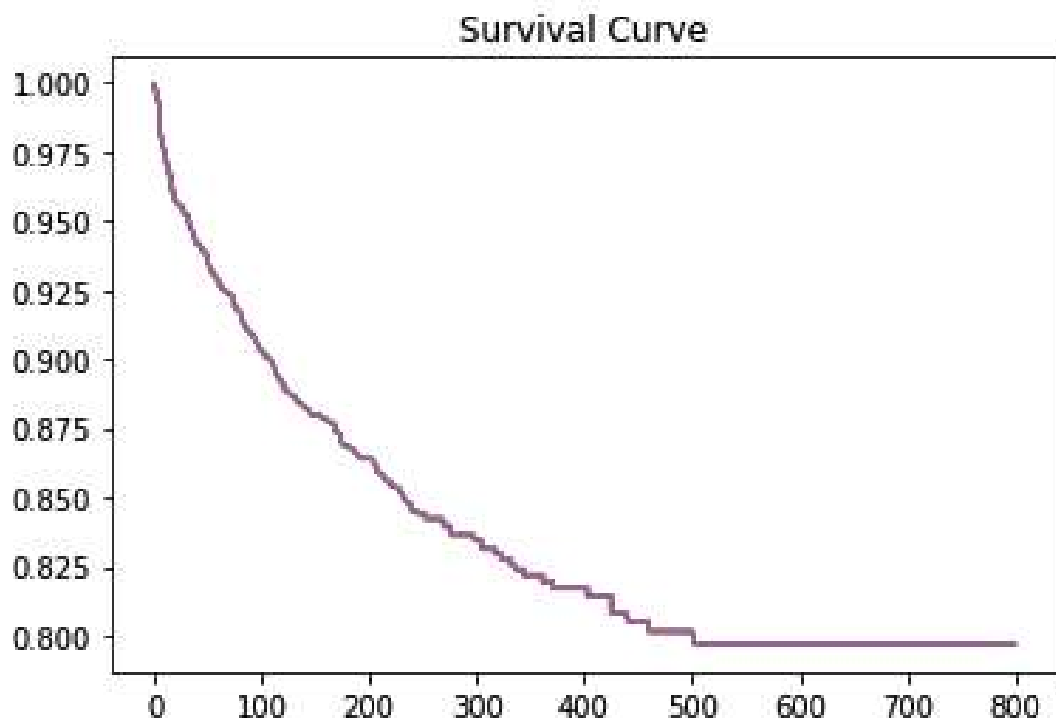
prediction survival function probabilities time scale 1 hour to 800 days. Column 0.041 is next hour

```
# predict survival function
model.predict_survival_function(stest[0:5], plot=True)
```

	0.041667	0.427778	0.791435	0.942361	1.354861	1.466667	1.629167	1.682639	2.081134	2.096528	...	789.321471	790.179109	790.213138	790.258971
0	0.999098	0.998644	0.998189	0.997733	0.997275	0.996816	0.996355	0.995894	0.995431	0.994966	...	0.797034	0.797034	0.797034	0.797034
1	0.999098	0.998644	0.998189	0.997733	0.997275	0.996816	0.996356	0.995894	0.995431	0.994966	...	0.797048	0.797048	0.797048	0.797048
2	0.999098	0.998644	0.998189	0.997732	0.997274	0.996815	0.996355	0.995893	0.995430	0.994965	...	0.796996	0.796996	0.796996	0.796996
3	0.999098	0.998644	0.998189	0.997733	0.997275	0.996816	0.996355	0.995893	0.995430	0.994966	...	0.797019	0.797019	0.797019	0.797019
4	0.999098	0.998644	0.998189	0.997733	0.997275	0.996816	0.996355	0.995894	0.995430	0.994966	...	0.797030	0.797030	0.797030	0.797030

5 rows × 685 columns

## Survival Curve



```

# predict log hazard ratio
print(model.predict(stest[0:10]))

[[-0.9998013 ]
 [-0.9998823 ]
 [-0.99959344]
 [-0.9997211 ]
 [-0.99978375]
 [-0.01272986]
 [-0.9998494 ]
 [-0.99980116]
 [ 0.9999072 ]
 [-0.999845  ]]

# predict hazard ratio
print(model.predict(stest[0:10], output_margin=False))

[[0.36795256]
 [0.36792275]
 [0.36802903]
 [0.36798206]
 [0.367959  ]
 [0.9873508 ]
 [0.36793485]
 [0.3679526 ]
 [2.7180295 ]
 [0.36793646]]

```

## Restful HTTP API

repository

<https://gitlab.com/juan.huertas/afi-http-restful-interface>

service running in Gunicorn or Cherrypy webser

entry point

```

url = 'http://35.246.52.204:15436/afi/predict'
headers = {'Content-Type': 'application/json'}

body = """{"columns":["Asset_Number"],
"index": [0,4,6,8,9,11,15,16,21,22,24,25,26,27,30,31,34,37,38,44,45,48,49,
59,77,82,83,99,100,101,102,129,133,134,135,136,137,138,139,141,142,147,
148,156,157,159,160,161,162,163,164,165,166,167,169,170,171,172,173,175,
176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,193,194,
195,198,199,202,204,205,206,207,208,209,211,212,213,214,215,216,217,218,

```

219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,  
237,238,239,240,241,242,243,244,245,246,247,249,251,252,253,254,255,256,  
257,259,260,261,262,263,264,265,266,267,268,269,270,271,272,274,275,276,  
277,279,280,281,282,283,284,285,286,287,288,289,291,293,295,297,298,299,  
300,301,304,307,308,309,310,311,313,314,315,316,317,318,321,322,323,324,  
325,327,329,330,331,332,333,334,335,337,338,339,341,342,343,344,346,348,  
350,351,353,354,355,356,357,358,359,360,361,362,363,365,366,367,368,370,  
371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,  
389,390,391,393,396,397,398,399,400,401,402,403,404,405,406,407,408,409,  
410,411,412,413,414,415,416,418,419,421,422,423,424,425,426,427,428,429,  
430,431,432,433,434,435,436,437,438,439,440,441,443,444,445,446,447,448,  
449,450,451,452,454,455,456,457,458,459,460,462,465,467,468,469,470,471,  
473,474,475,476,477,478,479,480,481,482,483,484,486,487,488,489,490,491,  
492,493,494,495,496,497,498,501,502,503,504,505,506,507,508,509,510,511,  
513,514,518,519,520,521,522,523,525,526,527,528,529,533,535,536,537,538,  
539,540,541,542,543,544,545,546,548,549,550,551,552,553,554,555,556,557,  
558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,  
576,577,578,579,581,582,583,584,585,586,587,588,591,592,593,594,595,598,  
601,608,611,612,613,615,616,618,621,622,625,626,627,631,632,633,634,635,  
637,638,639,640,642,643,644,645,646,647,648,649,650,651,655,656,657,659,  
661,662,663,664,672,673,674,677,678,679,683,684,686,687,688,690,692,693,  
696,697,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,  
717,718,719,720,721,723,724,725,726,728,729,730,731,733,734,735,737,738,  
739,741,742,743,744,747,748,749,750,751,752,758,759,760,761,766,767,768,  
769,771,778,781,783,785,786,787,788,789,790,791,794,795,796,797,798,799,  
800,802,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,821,  
822,823,824,825,829,832,833,838,839,840,841,842,843,844,845,846,848,849,  
850,859,860,861,862,864,865,866,868,869,871,872,874,875,876,877,878,879,  
880,882,883,884,885,886,887,890,891,892,893,895,897,898,899,900,904,905,  
906,907,908,910,911,912,917,919,924,930,931,932,933,934,935,936,937,938,  
939], "data": [[48075], [48081], [48353], [48354], [48355], [48356], [48358], [48  
367], [49828], [49829], [49830], [49831], [49832], [50378], [50383], [50384], [50  
387], [50811], [51863], [51864], [51865], [53991], [53992], [53999], [54002], [54  
014], [54015], [54035], [54036], [54040], [54041], [54058], [54060], [54062], [54  
499], [54502], [54503], [55016], [55017], [58134], [58135], [58136], [58145], [58  
146], [145644], [159598], [183741], [186801], [186984], [187011], [187013], [193  
696], [194094], [194102], [864101], [865676], [955168], [955171], [955617], [967  
632], [971600], [971602], [971609], [971662], [971665], [971670], [971671], [971  
672], [971684], [971697], [971699], [971700], [971806], [971807], [971813], [973  
382], [973483], [973523], [973524], [973737], [973809], [974250], [974459], [974  
467], [974468], [974469], [974486], [974584], [974688], [975814], [975880], [977  
985], [978419], [979889], [979905], [979951], [990439], [991872], [992941], [993  
429], [1002304], [1002332], [1009571], [1009760], [1009764], [1009767], [101018  
8], [1010390], [1011838], [1011867], [1012077], [1012958], [1012960], [1012968]  
, [1012969], [1013046], [1013085], [1013087], [1013090], [1013115], [1013162], [1  
013173], [1013185], [1013203], [1013236], [1013251], [1013252], [1013478], [10  
13507], [1013521], [1013611], [1013665], [1013815], [1013819], [1014831], [1015  
924], [1017203], [1017581], [1017584], [1017585], [1017595], [1017877], [101788  
5], [1017935], [1018042], [1018046], [1018063], [1018064], [1018073], [1018520]  
, [1018778], [1018858], [1018861], [1018886], [1018888], [1018891], [1018920], [

1018966], [1018989], [1018995], [1019177], [1019183], [1019232], [1019404], [1019628], [1019646], [1019652], [1019659], [1019662], [1019749], [1020097], [1020106], [1020122], [1020593], [1020596], [1020799], [1020801], [1020811], [1020816], [1020833], [1020927], [1020930], [1020931], [1021037], [1021066], [1021072], [1021079], [1021261], [1021381], [1021737], [1022412], [1022441], [1022470], [1022478], [1022479], [1022484], [1022486], [1022496], [1022499], [1022505], [1022509], [1022541], [1022588], [1022652], [1022657], [1022673], [1022676], [1022788], [1023057], [1023178], [1023319], [1023333], [1023335], [1024150], [1024163], [1024172], [1024189], [1024203], [1024209], [1024211], [1024212], [1024213], [1024259], [1024271], [1024273], [1024964], [1025178], [1025218], [1025423], [1025451], [1026338], [1026486], [1027187], [1027372], [1027883], [1041932], [1045266], [1046522], [1053761], [1054716], [1054743], [1055237], [1055280], [1059206], [1059621], [1059622], [1059623], [1061027], [1063020], [1115064], [1115082], [1119140], [1123155], [1123163], [1146544], [1162828], [1187649], [1238186], [1238432], [1239326], [1239329], [1239402], [1239404], [1239445], [1245364], [1245645], [1245654], [1245687], [1245688], [1245690], [1245695], [1246123], [1310453], [1310457], [1348667], [1348681], [1348686], [1348689], [1348707], [1348713], [1348870], [1348871], [1348886], [1348887], [1396883], [1396887], [1396889], [1396907], [1396959], [1396979], [1396990], [1397370], [1450622], [1450667], [1450680], [1450789], [1450806], [1450808], [1450811], [1451012], [1451013], [1451911], [1451912], [1452212], [1452260], [1452339], [1452340], [1452455], [1452479], [1452482], [1452484], [1452488], [1452499], [1452532], [1452569], [1455609], [1455848], [1455859], [1455957], [1455966], [1455984], [1455994], [1456217], [1458187], [1458223], [1458232], [1458481], [1458795], [1458798], [1458809], [1458826], [1458831], [1458849], [1458912], [1458939], [1458953], [1458966], [1458991], [1459128], [1459143], [1459323], [1459376], [1459379], [1459690], [1459743], [1459747], [1459765], [1459770], [1459796], [1459856], [1460155], [1460171], [1465071], [1495417], [1496028], [1518823], [1519167], [1522007], [1523674], [1525481], [1525490], [1525659], [1525663], [1525802], [1526206], [1526220], [1526240], [1526364], [1527447], [1527479], [1527541], [1527647], [1527855], [1527930], [1529223], [1529710], [1529906], [1529969], [1529974], [1532112], [1576723], [1576821], [1578891], [1624685], [1624700], [1648347], [1667658], [1670641], [1694189], [1697000], [1697974], [1730405], [1770159], [1773689], [1790838], [1800438], [1801500], [1811508], [1817562], [1819980], [1821012], [1821306], [1821309], [1821313], [1821315], [1822826], [1822997], [1823006], [1831312], [1833196], [1836380], [1842955], [1843087], [1849974], [1855659], [1858063], [1865260], [1870000], [1872049], [1872562], [1873345], [1873365], [1873391], [1873893], [1953026], [1962828], [1962829], [1967076], [1973292], [1979945], [1979946], [1982026], [1987737], [1993388], [1993396], [1993399], [1998913], [1999272], [1999658], [1999769], [2001752], [2001754], [2001756], [2001958], [2004129], [2024163], [2024263], [2024343], [2024345], [2024347], [2029276], [2029304], [2037528], [2061893], [2061928], [2061932], [2061933], [2061944], [2061960], [2061969], [2061970], [2061985], [2062015], [2062044], [2071743], [2095932], [2116896], [2116904], [2116911], [2124750], [2183184], [2241758], [2241760], [2242019], [2242593], [2402964], [2404191], [2404227], [2410736], [2439848], [2439895], [2439901], [2440031], [2440037], [2440328], [2440334], [2440516], [2440583], [2446408], [2446447], [2446624], [2446640], [2452647], [2460039], [2460476], [2462317], [2464458], [2464537], [2489239], [2489246], [2504880], [2513472], [2513669], [2513689], [2513730], [2513756], [2513767], [2514361], [2514783], [2514855], [2514858], [2514859], [2537900], [2558343], [2558608], [2558617], [2644712], [

2670795], [2702765], [2702767], [2702870], [2702952], [2703295], [2703339], [2703796], [2703801], [2703803], [2703805], [2703921], [2703922], [2704032], [2704237], [2704241], [2704259], [2704380], [2704910], [2710421], [2710468], [2710821], [2711146], [2711164], [2711165], [2729430], [2751765], [2753760], [2753766], [2771604], [2837110], [2842446], [2908190], [2933283], [2937907], [2989165], [3166736], [3268171], [3674674], [3739043], [4692453], [4692455], [4692460], [4692466], [6439281], [6701175], [6785164], [6785165], [7694144], [7780568], [7789743], [7789747], [7789748], [7796429], [7878616], [7919932], [9247647], [9247659], [9256904], [9263878], [9265336], [9266156], [10159829], [10164079], [10164087], [10164143], [10169828], [10290324], [10290424], [10290474], [10290512], [10292644], [10292768], [10294778], [10294783], [10294790], [10294791], [10330725], [10341377], [10353641], [10363615], [10376835], [10377357], [10377420], [10391812], [10391813], [10392091], [10392092], [10392094], [10392383], [10392449], [10392814], [10411607], [10432733], [10432734], [10432899], [10683554], [10685342], [11352224], [14105964], [17006519], [17055406], [17901673], [18227081], [18227082], [18241930], [18241931], [18248068], [18248069], [18248070], [18248071], [18652961], [18654037], [18666843], [18674707], [18697685], [18778080], [18789880], [18804192], [18808004]]}""body = ""{"columns": ["Asset\_Number"], "index": [0, 4, 6, 8, 9, 11, 15, 16, 21, 22, 24, 25, 26, 27, 30, 31, 34, 37, 38, 44, 45, 48, 49, 59, 77, 82, 83, 99, 100, 101, 102, 129, 133, 134, 135, 136, 137, 138, 139, 141, 142, 147, 148, 156, 157, 159, 160, 161, 162, 163, 164, 165, 166, 167, 169, 170, 171, 172, 173, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 193, 194, 195, 198, 199, 202, 204, 205, 206, 207, 208, 209, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 249, 251, 252, 253, 254, 255, 256, 257, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 274, 275, 276, 277, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 291, 293, 295, 297, 298, 299, 300, 301, 304, 307, 308, 309, 310, 311, 313, 314, 315, 316, 317, 318, 321, 322, 323, 324, 325, 327, 329, 330, 331, 332, 333, 334, 335, 337, 338, 339, 341, 342, 343, 344, 346, 348, 350, 351, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 365, 366, 367, 368, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 393, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 418, 419, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 454, 455, 456, 457, 458, 459, 460, 462, 465, 467, 468, 469, 470, 471, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 513, 514, 518, 519, 520, 521, 522, 523, 525, 526, 527, 528, 529, 533, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 581, 582, 583, 584, 585, 586, 587, 588, 591, 592, 593, 594, 595, 598, 601, 608, 611, 612, 613, 615, 616, 618, 621, 622, 625, 626, 627, 631, 632, 633, 634, 635, 637, 638, 639, 640, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 655, 656, 657, 659, 661, 662, 663, 664, 672, 673, 674, 677, 678, 679, 683, 684, 686, 687, 688, 690, 692, 693, 696, 697, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 717, 718, 719, 720, 721, 723, 724, 725, 726, 728, 729, 730, 731, 733, 734, 735, 737, 738, 739, 741, 742, 743, 744, 747, 748, 749, 750, 751, 752, 758, 759, 760, 761, 766, 767, 768, 769, 771, 778, 781, 783, 785, 786, 787, 788, 789, 790, 791, 794, 795, 796, 797, 798, 799, 800, 802, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 821, 822, 823, 824, 825, 829, 832, 833, 838, 839, 840, 8

41,842,843,844,845,846,848,849,850,859,860,861,862,864,865,866,868,869,871,872,874,875,876,877,878,879,880,882,883,884,885,886,887,890,891,892,893,895,897,898,899,900,904,905,906,907,908,910,911,912,917,919,924,930,931,932,933,934,935,936,937,938,939]

, "data": [48075], [48081], [48353], [48354], [48355], [48356], [48358], [48367], [49828], [49829], [49830], [49831], [49832], [50378], [50383], [50384], [50387], [50811], [51863], [51864], [51865], [53991], [53992], [53999], [54002], [54014], [54015], [54035], [54036], [54040], [54041], [54058], [54060], [54062], [54499], [54502], [54503], [55016], [55017], [58134], [58135], [58136], [58145], [58146], [145644], [159598], [183741], [186801], [186984], [187011], [187013], [193696], [194094], [194102], [864101], [865676], [955168], [955171], [955617], [967632], [971600], [971602], [971609], [971662], [971665], [971670], [971671], [971672], [971684], [971697], [971699], [971700], [971806], [971807], [971813], [973382], [973483], [973523], [973524], [973737], [973809], [974250], [974459], [974467], [974468], [974469], [974486], [974584], [974688], [975814], [975880], [977985], [978419], [979889], [979905], [979951], [990439], [991872], [992941], [993429], [1002304], [1002332], [1009571], [1009760], [1009764], [1009767], [1010188], [1010390], [1011838], [1011867], [1012077], [1012958], [1012960], [1012968], [1012969], [1013046], [1013085], [1013087], [1013090], [1013115], [1013162], [1013173], [1013185], [1013203], [1013236], [1013251], [1013252], [1013478], [1013507], [1013521], [1013611], [1013665], [1013815], [1013819], [1014831], [1015924], [1017203], [1017581], [1017584], [1017585], [1017595], [1017877], [1017885], [1017935], [1018042], [1018046], [1018063], [1018064], [1018073], [1018520], [1018778], [1018858], [1018861], [1018886], [1018888], [1018891], [1018920], [1018966], [1018989], [1018995], [1019177], [1019183], [1019232], [1019404], [1019628], [1019646], [1019652], [1019659], [1019662], [1019749], [1020097], [1020106], [1020122], [1020593], [1020596], [1020799], [1020801], [1020811], [1020816], [1020833], [1020927], [1020930], [1020931], [1021037], [1021066], [1021072], [1021079], [1021261], [1021381], [1021737], [1022412], [1022441], [1022470], [1022478], [1022479], [1022484], [1022486], [1022496], [1022499], [1022505], [1022509], [1022541], [1022588], [1022652], [1022657], [1022673], [1022676], [1022788], [1023057], [1023178], [1023319], [1023333], [1023335], [1024150], [1024163], [1024172], [1024189], [1024203], [1024209], [1024211], [1024212], [1024213], [1024259], [1024271], [1024273], [1024964], [1025178], [1025218], [1025423], [1025451], [1026338], [1026486], [1027187], [1027372], [1027883], [1041932], [1045266], [1046522], [1053761], [1054716], [1054743], [1055237], [1055280], [1059206], [1059621], [1059622], [1059623], [1061027], [1063020], [1115064], [1115082], [1119140], [1123155], [1123163], [1146544], [1162828], [1187649], [1238186], [1238432], [1239326], [1239329], [1239402], [1239404], [1239445], [1245364], [1245645], [1245654], [1245687], [1245688], [1245690], [1245695], [1246123], [1310453], [1310457], [1348667], [1348681], [1348686], [1348689], [1348707], [1348713], [1348870], [1348871], [1348886], [1348887], [1396883], [1396887], [1396891], [1396907], [1396959], [1396979], [1396990], [1397370], [1450622], [1450667], [1450680], [1450789], [1450806], [1450808], [1450811], [1451012], [1451013], [1451911], [1451912], [1452212], [1452260], [1452339], [1452340], [1452455], [1452479], [1452482], [1452484], [1452488], [1452499], [1452532], [1452569], [1455609], [1455848], [1455859], [1455957], [1455966], [1455984], [1455994], [1456217], [1458187], [1458223], [1458232], [1458481], [1458795], [1458798], [1458809], [1458826], [1458831], [1458849], [1458912], [1458939], [1458953], [1458966], [1458991], [1459128], [1459143], [1459323], [1459376], [1459379], [1459690], [145974

3], [1459747], [1459765], [1459770], [1459796], [1459856], [1460155], [1460171], [1465071], [1495417], [1496028], [1518823], [1519167], [1522007], [1523674], [1525481], [1525490], [1525659], [1525663], [1525802], [1526206], [1526220], [1526240], [1526364], [1527447], [1527479], [1527541], [1527647], [1527855], [1527930], [1529223], [1529710], [1529906], [1529969], [1529974], [1532112], [1576723], [1576821], [1578891], [1624685], [1624700], [1648347], [1667658], [1670641], [1694189], [1697000], [1697974], [1730405], [1770159], [1773689], [1790838], [1800438], [1801500], [1811508], [1817562], [1819980], [1821012], [1821306], [1821309], [1821313], [1821315], [1822826], [1822997], [1823006], [1831312], [1833196], [1836380], [1842955], [1843087], [1849974], [1855659], [1858063], [1865260], [1870000], [1872049], [1872562], [1873345], [1873365], [1873391], [1873893], [1953026], [1962828], [1962829], [1967076], [1973292], [1979945], [1979946], [1982026], [1987737], [1993388], [1993396], [1993399], [1998913], [1999272], [1999658], [1999769], [2001752], [2001754], [2001756], [2001958], [2004129], [2024163], [2024263], [2024343], [2024345], [2024347], [2029276], [2029304], [2037528], [2061893], [2061928], [2061932], [2061933], [2061944], [2061960], [2061969], [2061970], [2061985], [2062015], [2062044], [2071743], [2095932], [2116896], [2116904], [2116911], [2124750], [2183184], [2241758], [2241760], [2242019], [2242593], [2402964], [2404191], [2404227], [2410736], [2439848], [2439895], [2439901], [2440031], [2440037], [2440328], [2440334], [2440516], [2440583], [2446408], [2446447], [2446624], [2446640], [2452647], [2460039], [2460476], [2462317], [2464458], [2464537], [2489239], [2489246], [2504880], [2513472], [2513669], [2513689], [2513730], [2513756], [2513767], [2514361], [2514783], [2514855], [2514858], [2514859], [2537900], [2558343], [2558608], [2558617], [2644712], [2670795], [2702765], [2702767], [2702870], [2702952], [2703295], [2703339], [2703796], [2703801], [2703803], [2703805], [2703921], [2703922], [2704032], [2704237], [2704241], [2704259], [2704380], [2704910], [2710421], [2710468], [2710821], [2711146], [2711164], [2711165], [2729430], [2751765], [2753760], [2753766], [2771604], [2837110], [2842446], [2908190], [2933283], [2937907], [2989165], [3166736], [3268171], [3674674], [3739043], [4692453], [4692455], [4692460], [4692466], [6439281], [6701175], [6785164], [6785165], [7694144], [7780568], [7789743], [7789747], [7789748], [7796429], [7878616], [7919932], [9247647], [9247659], [9256904], [9263878], [9265336], [9266156], [10159829], [10164079], [10164087], [10164143], [10169828], [10290324], [10290424], [10290474], [10290512], [10292644], [10292768], [10294778], [10294783], [10294790], [10294791], [10330725], [10341377], [10353641], [10363615], [10376835], [10377357], [10377420], [10391812], [10391813], [10392091], [10392092], [10392094], [10392383], [10392449], [10392814], [10411607], [10432733], [10432734], [10432899], [10683554], [10685342], [11352224], [14105964], [17006519], [17055406], [17901673], [18227081], [18

```
227082],[18241930],[18241931],[18248068],[18248069],[18248070],[18248071],
[18652961],[18654037],[18666843],[18674707],[18697685],[18778080],[187
89880],[18804192],[18808004]]}"""
```

the Json request file contains the Asset numbers of element on the route to calculate its survival probability

the answer in json payload format

contains Asset number, survival probability in next hour , ELR and asset description

```
{\"0\":{\"Asset_Number\":48075,\"prob_survival\":0.9992737934,\"ELR\": \"
DOL2\", \"Asset_Class_Grouping\": \"Track\"},
\"1\":{\"Asset_Number\":48081,\"prob_survival\":0.9992737934,\"ELR\": \"D
OL2\", \"Asset_Class_Grouping\": \"Track\"},
\"2\":{\"Asset_Number\":48353,\"prob_survival\":0.9992738136,\"ELR\": \"E
CM3\", \"Asset_Class_Grouping\": \"Track\"},
\"3\":{\"Asset_Number\":48354,\"prob_survival\":0.9992738223,\"ELR\": \"E
CM3\", \"Asset_Class_Grouping\": \"Track\"}}
```

## Time Table Prediction Model and Recommendation Engine

### Phase 1 Definition

[Timetable Delay \(TD\) Predictive Engine](#)

### Phase 2 Upgrade

- Update of NN Model and Genetic Algorithm as fit function on the Recommendation Engine
- Replace LSTM NN with RTPMS DNN as Fit function
- taking necessary steps to industrialize the model

### ML ETL Process Time Table Prediction Model / Recommendation Engine

#### Time Table prediction Model & Recommendation Engine

There are 3 main tables involved on the ETL process, ccf\_enriched, wttmaster and wttchild.

We use Big-query and google.cloud libraries to connect Postgress database. In the following Jupyter Notebook we have an examples of how we are accessing the database

The following Jupyter Ipython Notebook contains the latest version of ETL and Data-set transformation for this model. Here we input an stacked LSTM NN and out input is a variable number of sequences of variable length. each sample inputting the Neural Network is a variable sequences of rows with n features(columns)

[TTPM\\_LSTM\\_per Station\\_v7\\_for\\_Rec\\_ENGINE.ipynb](#)

Code In Jupyter Notebook	Comments
--------------------------	----------