01.survRandomForest

July 5, 2021

1 Random Survival Forest

Customer churn/attrition, a.k.a the percentage of customers that stop using paying services, is one of the most important metrics for a business, as it usually costs more to acquire new customers than it does to retain existing ones. Indeed, according to a study by Bain & Company, existing customers tend to buy more from a company over time, thus reducing the operating costs of the business and may refer the products they use to others. For example, in financial services, a 5% increase in customer retention produces more than a 25% increase in profit. By using Survival Analysis, not only companies can predict if customers are likely to stop doing business but also when that event might happen.

1.1 Methods

In this study, we adopt random survival forests which have never been used in understanding factors affecting membership in a sport club using existing data in a Sport Club. The analysis is based on the use of random survival forests in the presence of covariates that do not necessarily satisfy the PH assumption. Random Survival Forests does not make the proportional hazards assumption (Ehrlinger, 2016) and has the flexibility to model survivor curves that are of dissimilar shapes for contrastinggroups of subjects. Random Survival Forest is an extension of Random Forest allowing efficient non-parametric analysis of time to event data (Breiman, 2001). This characteristcs allow us to surpass the Cox Regression limitation of the proportional hazard assumption, requiring to exclude variables which not fullfill the model assumption. It was shown by (Breiman, 2001) that ensemble learning can be further improved by injecting randomization into the base learning process - a method called Random Forests.
The random survival forest was developed using the package PySurvival (Fotso & Others, 2019) The most relevant variables predicting the dropout are analysed using the log-rank test. The metric variables are transformed to categorical using the quartiles to provide a statistical comparison of groups. The survival analysis was conducted using the package Lifelines (Davidson-Pilon et al., 2017).

1.2 Results

The initial model has a c-index of 0.92. After removing estadoCivil_outro and ano c-index improved to 0.94. Without idade improved to 0.95. The most relevant variables predicting the dropout are: - mesesUP - valorTotal - anoUltimoPagamento - quotaMensal - escaloesTotalJogos_ate 1 - jogosEpoca - escaloesTotalJogos_56 a 105 - estadoCivil_solteiro - escaloesTotalJogos_21 a 56 - escaloesTotalJogos_mais 105 - sexo_M - estadoCivil_nao definido

TODO: COLOCAR ESTA DESCRIÇÃO PARA TODAS AS VARIÁVEIS: There were identified significative differences between the gender groups (2=194.63, p < .005), wrenew two or more contracts, the survival probability for 12 months is 85.49%

1.3 Methods bibliography

- Ehrlinger, J. (2016). ggRandomForests: Exploring Random Forest Survival. ArXiv:1612.08974 [Stat]. http://arxiv.org/abs/1612.08974
- Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32. https://doi.org/10/d8zjwq
- Fotso, S., & others. (2019). PySurvival: Open source package for Survival Analysis modeling. https://www.pysurvival.io/

2 Dataset

Considering the sport club policies all the customers with payments less than 24 months where considered active:
- dt['abandonou'] = 0 - dt.loc[dt['mesesUP']>=24,'abandonou']=1

The variables extracted from the software correspond to the time interval of becoming a customer until the end of observation (censoring on 31 Maio 2019) or the end of the customer relationship (dropout). The survival time in the dataset is represented by the number of years the customer begin affilated.

We extracted records of 25316 customers (male n=17246, female n=8070) from a sport club; data corresponded to the time period between October 1, 1944 and May 31, 2019.

[4]:
```python
from IPython.display import HTML
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import datetime
import seaborn as sns

df = pd.read_excel('../data/dadosSociosTratados.xlsx',index_col=0)
```

2.1 Description

Variáveis:

- ano: ano inscrição sócio
- idade: numérica
- sexo: M ou F
- estadoCivil: solteiro, não definido, casado, outro
- quotaMensal: valor da quota mensal
- ultimPagamento: Quando é que foi realizado o último pagamento
- valorTotal: Valor total pago
- totalJogos: Total de Jogos que o sócio foi
- jogosEpoca: Jogos vistos na última época
- mesesUP: Quantos meses desde o último pagamento

- anosSocio: Há quantos anos é sócio
- idaEstágio: Vai ou não ao estádio
- abandonou: 1 não é sócio, 0 é sócio censura

Ei is the event indicator such that Ei=1, if an event happens and Ei=0 in case of censoring

As variáveis categóricas foram transformadas em dummies: - sexo - estadoCivil - escaloesTotalJogos

[5]:
```
df.sexo.value_counts()
```

[5]:
```
M    17246
F     8070
Name: sexo, dtype: int64
```

[6]:
```
df.describe()
```

[6]:
```
                ano         idade   quotaMensal    valorTotal    totalJogos  \
count  25316.000000  25316.000000  25316.000000  25316.000000  25316.000000
mean    2007.048033     27.262996      4.356099    316.037984     26.535946
std       10.937818     20.087078      3.550837    493.971528     45.812996
min     1944.000000    -70.000000      0.000000      0.000000      0.000000
25%     2004.000000     13.000000      1.000000      5.000000      0.000000
50%     2010.000000     19.000000      2.500000     53.000000      0.000000
75%     2014.000000     41.000000      6.000000    448.250000     36.000000
max     2019.000000    118.000000     10.000000   2602.000000    197.000000

         jogosEpoca  diasUltimoPagamento       mesesUP     abandonou  \
count  25316.000000         25316.000000  25316.000000  25316.000000
mean       2.171631           586.277033     18.814110      0.221638
std        4.076356           990.398069     32.498248      0.415357
min        0.000000             0.000000      0.000000      0.000000
25%        0.000000            83.994934      2.000000      0.000000
50%        0.000000           122.113031      4.000000      0.000000
75%        2.000000           534.982420     17.000000      0.000000
max       16.000000          4778.034828    156.000000      1.000000

          anosSocio    idaEstadio           mes
count  25316.000000  25316.000000  25316.000000
mean      11.264339      0.401367      6.875454
std       10.908777      0.490185      3.391117
min        0.000000      0.000000      1.000000
25%        5.000000      0.000000      4.000000
50%        8.000000      0.000000      8.000000
75%       14.000000      1.000000      9.000000
max       74.000000      1.000000     12.000000
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 25316 entries, 1 to 25316
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   nome                 25316 non-null  object
 1   dataAdesao           25316 non-null  datetime64[ns]
 2   ano                  25316 non-null  int64
 3   contribuinte         25316 non-null  object
 4   dataNascimento       25316 non-null  object
 5   idade                25316 non-null  int64
 6   sexo                 25316 non-null  object
 7   estadoCivil          25316 non-null  object
 8   categoria            25316 non-null  object
 9   quotaMensal          25316 non-null  float64
 10  profissao            25316 non-null  object
 11  codPostal            25316 non-null  object
 12  ultimaQuota          25316 non-null  object
 13  ultimoPagamento      25316 non-null  object
 14  valorTotal           25316 non-null  float64
 15  totalJogos           25316 non-null  int64
 16  jogosEpoca           25316 non-null  int64
 17  diasUltimoPagamento  25316 non-null  float64
 18  mesesUP              25316 non-null  int64
 19  abandonou            25316 non-null  int64
 20  anosSocio            25316 non-null  int64
 21  idaEstadio           25316 non-null  int64
 22  escaloesTotalJogos   25316 non-null  object
 23  mes                  25316 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(9), object(11)
memory usage: 4.8+ MB
```

[8]:
```
df.head()
```

[8]:
```
                                nome dataAdesao   ano contribuinte  \
Sócio
1        DURVAL MANUEL BELO MOREIRA 1944-10-01  1944    105910465
2       ANTONIO ALBINO BELO MOREIRA 1944-10-01  1944    152586199
3             MARIO GONCALVES BRAGA 1945-08-24  1945    999999990
4          MANUEL BATISTA CERQUEIRA 1945-09-01  1945    124938060
5           JOAQUIM MANUEL FERREIRA 1945-09-01  1945    108239110

            dataNascimento  idade sexo   estadoCivil  categoria  quotaMensal  \
Sócio
1      1935-05-11 00:00:00     83    M        casado      homem         10.0
2      1930-09-29 00:00:00     88    M      solteiro      homem         10.0
3      1945-08-24 00:00:00     73    M  nao definido      homem         10.0
4      1921-05-27 00:00:00     97    M        casado  reformado          5.0
```

```
5      1921-03-08 00:00:00     97    M         outro      homem         10.0

       … valorTotal totalJogos jogosEpoca diasUltimoPagamento  mesesUP  \
Sócio  …
1      …     1906.0          0          0          103.308984        3
2      …     1906.0          0          0          113.056309        3
3      …     1553.0          0          0         1100.031203       36
4      …      790.0          0          0          264.945987        8
5      …     1466.0          0          0         1089.943393       35

       abandonou  anosSocio  idaEstadio  escaloesTotalJogos  mes
Sócio
1              0         74           0               ate 1   10
2              0         74           0               ate 1   10
3              1         73           0               ate 1    8
4              0         73           0               ate 1    9
5              1         73           0               ate 1    9

[5 rows x 24 columns]
```

[9]:
```
df.tail()
```

[9]:
```
                                        nome dataAdesao   ano contribuinte  \
Sócio
25312             MICAEL YTWEVA MASSANO MOREIRA 2019-02-21  2019    286625296
25313  GONÇALO MARIA FERR. SENDIM POLIDO PIRES 2019-02-21  2019    275230511
25314               TOMAS MIGUEL SOARES RIBEIRO 2019-02-21  2019    290670578
25315     MIGUEL MARIA REBELO CORSINO DA SILVA 2019-02-21  2019    259420697
25316                   JOAO MANUEL JESUS VIDAL 2019-02-21  2019    257110887

            dataNascimento  idade sexo estadoCivil categoria  quotaMensal  \
Sócio
25312  2011-04-14 00:00:00      7    M    solteiro     sub14          1.0
25313  2010-05-26 00:00:00      8    M    solteiro    atleta          1.0
25314  2016-11-24 00:00:00      2    M    solteiro     sub14          1.0
25315  2004-06-30 00:00:00     14    M    solteiro     sub14          1.0
25316  1990-05-23 00:00:00     28    M    solteiro     homem         10.0

       … valorTotal totalJogos jogosEpoca diasUltimoPagamento  mesesUP  \
Sócio  …
25312  …       17.0          0          0           72.296706        2
25313  …       12.0          0          0           72.269791        2
25314  …       17.0          0          0           72.095076        2
25315  …       17.0          0          0           71.969059        2
25316  …        0.0          0          0           71.966639        2

       abandonou  anosSocio  idaEstadio  escaloesTotalJogos  mes
```

```
Sócio
25312           0          0           0              ate 1     2
25313           0          0           0              ate 1     2
25314           0          0           0              ate 1     2
25315           0          0           0              ate 1     2
25316           0          0           0              ate 1     2

[5 rows x 24 columns]
```

[10]:
```
df['ultimoPagamento'] = pd.to_datetime(df['ultimoPagamento'],format='%Y-%m-%d␣
↪%H:%M', errors='coerce')
```

[11]:
```
df['anoUltimoPagamento']=df['ultimoPagamento'].apply(lambda x: x.year)
```

[12]:
```
df.anoUltimoPagamento.unique()
```

[12]:
```
array([2019., 2016., 2018., 2015., 2017., 2014., 2008.,   nan, 2009.,
       2007., 2010., 2011., 2012., 2013., 2006.])
```

[13]:
```
df.anoUltimoPagamento=df.anoUltimoPagamento.fillna(0)
```

[13]:
```
df['anoUltimoPagamento']=df.anoUltimoPagamento.astype(int)
```

[14]:
```
df.
↪drop(columns=['nome','dataAdesao','ultimoPagamento','contribuinte','dataNascimento','mes','
                ␣
↪'codPostal','categoria','ultimaQuota','diasUltimoPagamento'],inplace=True)
```

[15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25316 entries, 1 to 25316
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   ano                 25316 non-null  int64
 1   idade               25316 non-null  int64
 2   sexo                25316 non-null  object
 3   estadoCivil         25316 non-null  object
 4   quotaMensal         25316 non-null  float64
 5   valorTotal          25316 non-null  float64
 6   totalJogos          25316 non-null  int64
 7   jogosEpoca          25316 non-null  int64
 8   mesesUP             25316 non-null  int64
 9   abandonou           25316 non-null  int64
 10  anosSocio           25316 non-null  int64
 11  idaEstadio          25316 non-null  int64
```

```
 12  escaloesTotalJogos  25316 non-null  object
 13  anoUltimoPagamento  25316 non-null  float64
dtypes: float64(3), int64(8), object(3)
memory usage: 2.9+ MB
```

[16]:
```
df.estadoCivil.value_counts()
```

[16]:
```
solteiro        12065
nao definido     7667
casado           5085
outro             499
Name: estadoCivil, dtype: int64
```

Variáveis:

- ano: ano inscrição sócio
- idade
- sexo
- estadoCivil: solteiro, não definido, casado, outro
- quotaMensal: valor da quota mensal
- ultimPagamento: Quando é que foi realizado o último pagamento
- valorTotal: Valor total pago
- totalJogos: Total de Jogos que o sócio foi
- jogosEpoca: Jogos vistos na última época
- mesesUP: Quantos meses desde o último pagamento
- anosSocio: Há quantos anos é sócio
- idaEstágio: Vai ou não ao estádio
- abandonou: 1 não é sócio, 0 é sócio censura

Ei is the event indicator such that Ei=1, if an event happens and Ei=0 in case of censoring

2.2 Converting from categorical to numerical

There are several categorical features that need to be encoded into one-hot vectors:

```
sexo
estadoCivil
escaloesTotalJogos
```

[17]:
```
dfCurvas = df.copy()
df = pd.get_dummies(df,␣
↪columns=['sexo','estadoCivil','escaloesTotalJogos'],drop_first=True)
```

[18]:
```
# Creating the time and event columns
time_column = 'anosSocio'
event_column = 'abandonou'

# Extracting the features
features = np.setdiff1d(df.columns, [time_column, event_column] ).tolist()
```

2.3 Verificar null values e duplicates

[19]:
```
# Checking for null values
N_null = sum(df[features].isnull().sum())
print("The raw_dataset contains {} null values".format(N_null)) #0 null values
```

```
The raw_dataset contains 0 null values
```

[20]:
```
# Removing duplicates if there exist
N_dupli = sum(df.duplicated(keep='first'))
df = df.drop_duplicates(keep='first').reset_index(drop=True)
print("The raw_dataset contains {} duplicates".format(N_dupli))

# Number of samples in the dataset
N = df.shape[0]
```

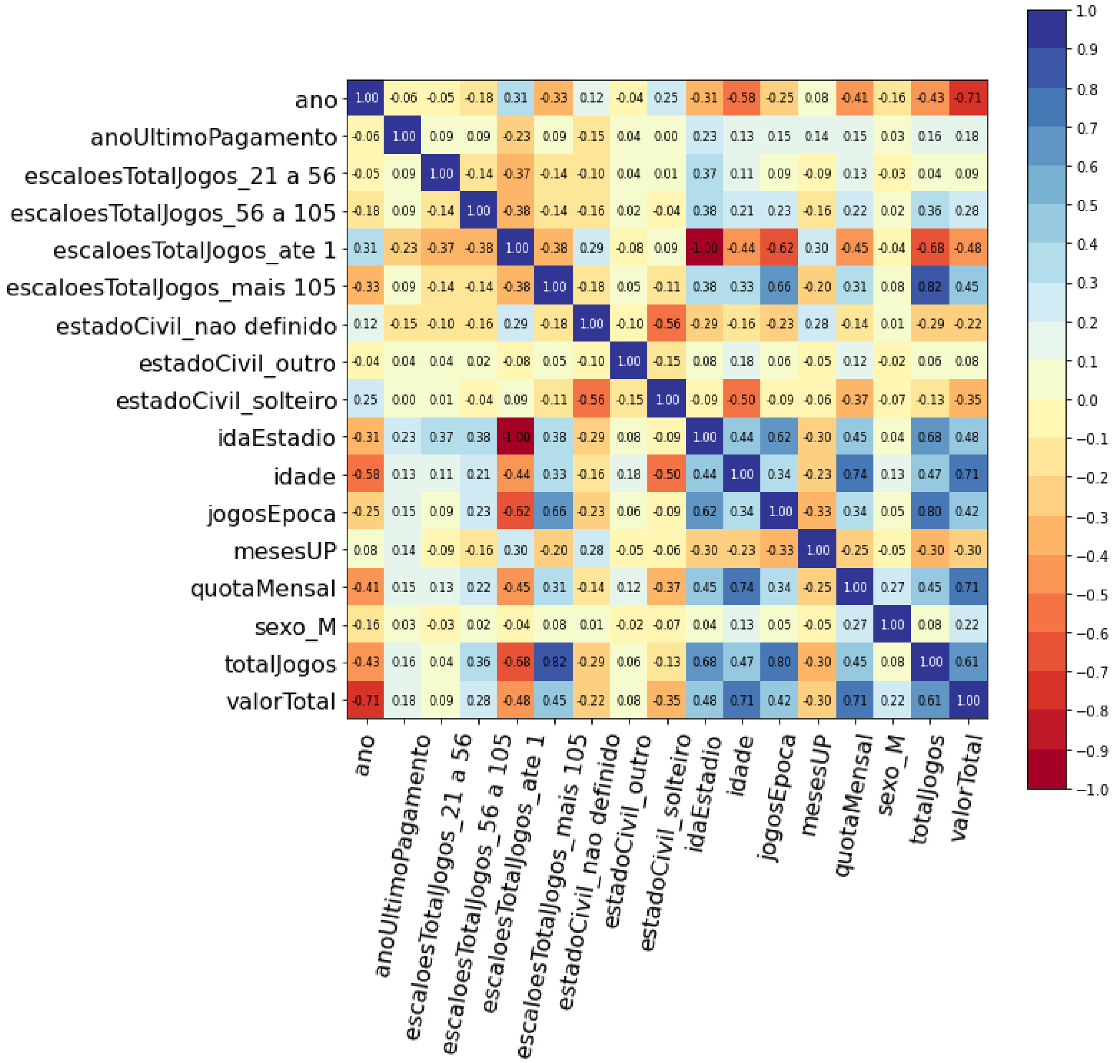
```
The raw_dataset contains 4928 duplicates
```

3 Exploratory Data Analysis

[21]:
```
df.columns
```

[21]:
```
Index(['ano', 'idade', 'quotaMensal', 'valorTotal', 'totalJogos', 'jogosEpoca',
       'mesesUP', 'abandonou', 'anosSocio', 'idaEstadio', 'anoUltimoPagamento',
       'sexo_M', 'estadoCivil_nao definido', 'estadoCivil_outro',
       'estadoCivil_solteiro', 'escaloesTotalJogos_21 a 56',
       'escaloesTotalJogos_56 a 105', 'escaloesTotalJogos_ate 1',
       'escaloesTotalJogos_mais 105'],
      dtype='object')
```

[22]:
```
from pysurvival.utils.display import correlation_matrix
correlation_matrix(df[features], figure_size=(10,10), text_fontsize=8)
```



Vamos remover as variáveis com correlações maiores

[23]:
```
to_remove = ['totalJogos', 'idaEstadio']
features = np.setdiff1d(features, to_remove).tolist()
```

4 Modeling

So as to perform cross-validation later on and assess the performances of the model, let's split the dataset into training and testing sets.

4.1 Building the model

[26]:
```
# Building training and testing sets
from sklearn.model_selection import train_test_split
index_train, index_test = train_test_split( range(N), test_size = 0.4)
data_train = df.loc[index_train].reset_index( drop = True )
data_test  = df.loc[index_test].reset_index( drop = True )

# Creating the X, T and E inputs
X_train, X_test = df[features], data_test[features]
T_train, T_test = df[time_column], data_test[time_column]
E_train, E_test = df[event_column], data_test[event_column]
```

[25]:
```
#from pysurvival.models.survival_forest import ConditionalSurvivalForestModel
from pysurvival.models.survival_forest import RandomSurvivalForestModel
# Fitting the model
csf = RandomSurvivalForestModel(num_trees=200)
csf.fit(X_train, T_train, E_train, max_features='sqrt',
        max_depth=5, min_node_size=20)
```

[25]:
```
RandomSurvivalForestModel
```

4.1.1 Features importance

[28]:
```
csf.variable_importance_table
```

[28]:

	feature	importance	pct_importance
0	mesesUP	25.444019	0.315903
1	anoUltimoPagamento	12.009933	0.149111
2	valorTotal	8.408662	0.104399
3	jogosEpoca	6.308750	0.078327
4	quotaMensal	5.297568	0.065772
5	estadoCivil_solteiro	4.335004	0.053822
6	idade	3.832753	0.047586
7	escaloesTotalJogos_56 a 105	3.559232	0.044190
8	escaloesTotalJogos_ate 1	3.279739	0.040720
9	escaloesTotalJogos_mais 105	2.490875	0.030926
10	sexo_M	2.381270	0.029565
11	escaloesTotalJogos_21 a 56	1.967675	0.024430
12	estadoCivil_nao definido	1.228350	0.015251
13	estadoCivil_outro	-0.957972	0.000000
14	ano	-12.725045	0.000000

A negative number means that the model performs better without estadoCivil_Outro and ano: https://stackoverflow.com/questions/27918320/what-does-negative-incmse-in-randomforest-package-mean

The variable mesesUP explains the survival 30%, year last payment 16.8%, totalAmount 12%, number of games 7.7%....

4.1.2 Model performance

We are going to access the model performance using the training and test set. Previously defined. C-index close to 1, the model has an powerfull discriminatory; but if it is close to 0.5, it has no ability to discriminate between low and high risk subjects.

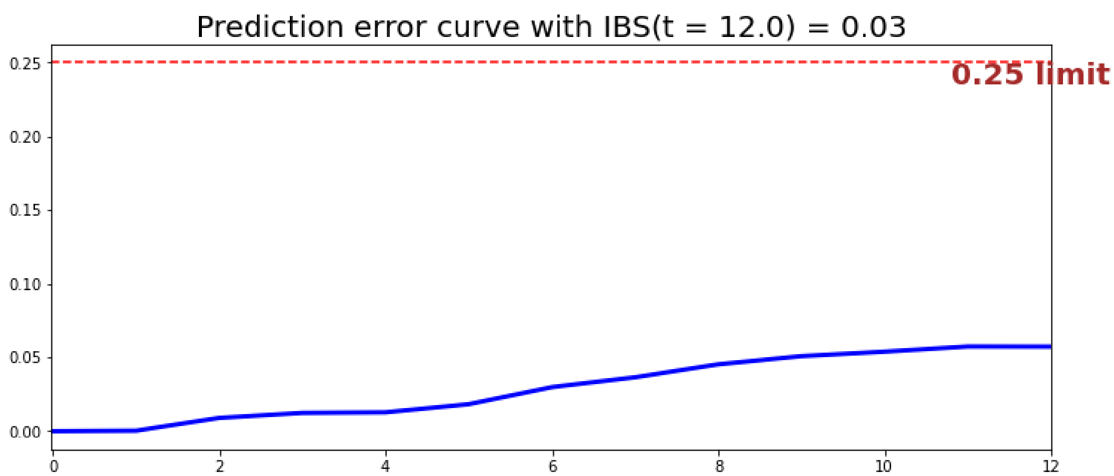
4.1.3 C-index

[29]:
```
from pysurvival.utils.metrics import concordance_index
c_index = concordance_index(csf, X_test, T_test, E_test)
print('C-index: {:.2f}'.format(c_index)) #0.83
```

```
C-index: 0.91
```

4.1.4 Brier Score

[30]:
```
from pysurvival.utils.display import integrated_brier_score
ibs = integrated_brier_score(csf, X_test, T_test, E_test, t_max=12,
    figure_size=(12,5))
print('IBS: {:.2f}'.format(ibs))
```




```
IBS: 0.03
```

The IBS is equal to 0.03 on the entire model time axis. This indicates that the model will have very good predictive abilities.

4.2 Building model without estadoCivil_outro and ano

[31]:
```
to_remove = ['estadoCivil_outro', 'ano']
features = np.setdiff1d(features, to_remove).tolist()
```

[32]:
```
# Creating the X, T and E inputs
X_train, X_test = df[features], data_test[features]
T_train, T_test = df[time_column], data_test[time_column]
E_train, E_test = df[event_column], data_test[event_column]
```

[33]:
```
csf = RandomSurvivalForestModel(num_trees=200)
csf.fit(X_train, T_train, E_train, max_features='sqrt',
        max_depth=5, min_node_size=20)
```

[33]:
```
RandomSurvivalForestModel
```

4.2.1 Features importance

[34]:
```
csf.variable_importance_table
```

[34]:

	feature	importance	pct_importance
0	mesesUP	26.109648	0.305522
1	anoUltimoPagamento	12.090799	0.141480
2	valorTotal	10.921663	0.127800
3	jogosEpoca	7.009034	0.082016
4	quotaMensal	5.877116	0.068771
5	estadoCivil_solteiro	4.837993	0.056612
6	escaloesTotalJogos_ate 1	4.230473	0.049503
7	escaloesTotalJogos_56 a 105	3.820484	0.044705
8	sexo_M	3.293989	0.038545
9	escaloesTotalJogos_mais 105	2.755716	0.032246
10	escaloesTotalJogos_21 a 56	1.918743	0.022452
11	estadoCivil_nao definido	1.827835	0.021388
12	idade	0.765702	0.008960

4.2.2 Model performance

We are going to access the model performance using the training and test set. Previously defined. C-index close to 1, the model has an powerfull discriminatory; but if it is close to 0.5, it has no ability to discriminate between low and high risk subjects.

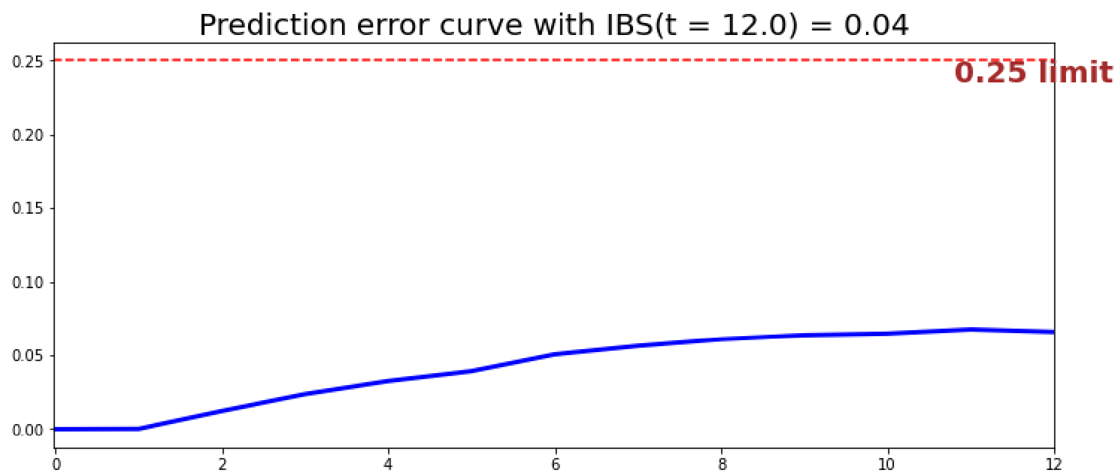
4.2.3 C-index

[35]:
```
from pysurvival.utils.metrics import concordance_index
c_index = concordance_index(csf, X_test, T_test, E_test)
print('C-index: {:.2f}'.format(c_index)) #0.83
```

```
C-index: 0.94
```

4.2.4 Brier Score

[36]:
```
from pysurvival.utils.display import integrated_brier_score
ibs = integrated_brier_score(csf, X_test, T_test, E_test, t_max=12,
    figure_size=(12,5))
print('IBS: {:.2f}'.format(ibs))
```




```
IBS: 0.04
```

4.3 Building model without idade

[39]:
```
to_remove = ['idade']
features = np.setdiff1d(features, to_remove).tolist()
```

[40]:
```
# Creating the X, T and E inputs
X_train, X_test = df[features], data_test[features]
T_train, T_test = df[time_column], data_test[time_column]
E_train, E_test = df[event_column], data_test[event_column]
```

[41]:
```
csf = RandomSurvivalForestModel(num_trees=200)
csf.fit(X_train, T_train, E_train, max_features='sqrt',
        max_depth=5, min_node_size=20)
```

[41]: RandomSurvivalForestModel

4.3.1 Features importance

[42]:
```
csf.variable_importance_table
```

[42]:

	feature	importance	pct_importance
0	mesesUP	26.059067	0.279085

```
1            anoUltimoPagamento   13.272150        0.142141
2                    valorTotal   11.317420        0.121206
3                   quotaMensal   10.881023        0.116532
4                    jogosEpoca    6.070691        0.065015
5      escaloesTotalJogos_ate 1    5.221843        0.055924
6           estadoCivil_solteiro    5.091730        0.054531
7   escaloesTotalJogos_56 a 105    4.685484        0.050180
8    escaloesTotalJogos_21 a 56    3.973545        0.042555
9                        sexo_M    3.178554        0.034041
10  escaloesTotalJogos_mais 105    2.510664        0.026888
11      estadoCivil_nao definido    1.111203        0.011901
```

4.3.2 Model performance

We are going to access the model performance using the training and test set. Previously defined. C-index close to 1, the model has an powerfull discriminatory; but if it is close to 0.5, it has no ability to discriminate between low and high risk subjects.

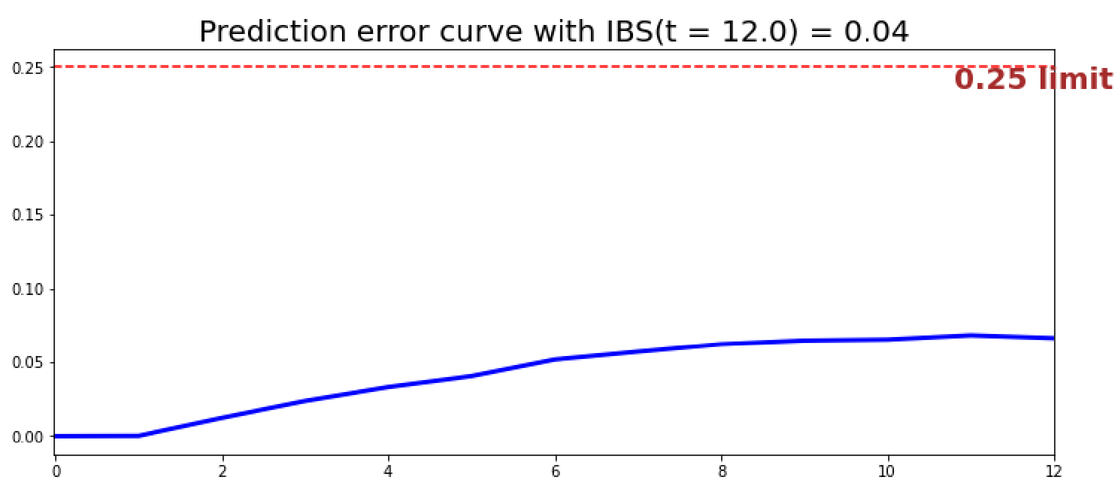
4.3.3 C-index

[43]:
```
from pysurvival.utils.metrics import concordance_index
c_index = concordance_index(csf, X_test, T_test, E_test)
print('C-index: {:.2f}'.format(c_index)) #0.83
```

```
C-index: 0.94
```

4.3.4 Brier Score

[44]:
```
from pysurvival.utils.display import integrated_brier_score
ibs = integrated_brier_score(csf, X_test, T_test, E_test, t_max=12,
    figure_size=(12,5))
print('IBS: {:.2f}'.format(ibs))
```

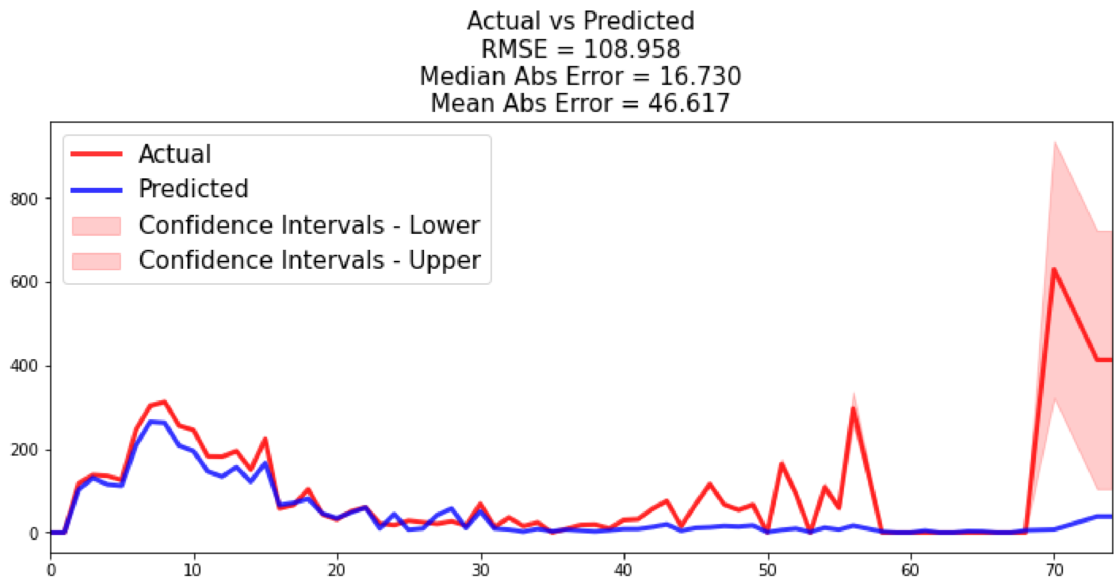


IBS: 0.04

5 Predictions

Lets compare the timeseries of actual and predicted customers who leave for each time t.

[45]:
```
from pysurvival.utils.display import compare_to_actual
results = compare_to_actual(csf, X_test, T_test, E_test, is_at_risk = False,
figure_size=(12, 5), metrics = ['rmse', 'mean', 'median'])
```

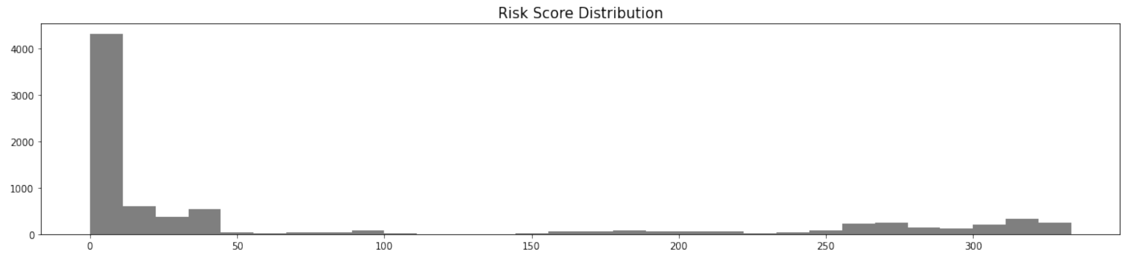


The model only makes an average absolute error of ~33 customers.

5.1 Individual predictions

Compute the probability of remaining a customer for all times t

[46]:
```
from pysurvival.utils.display import create_risk_groups

risk_groups = create_risk_groups(model=csf, X=X_test,
    use_log = False, num_bins=30, figure_size=(20, 4))
```



6 Curvas de sobrevivência

[47]:
```
def curvaSobrevivencia(dados,coluna):
    ax = plt.subplot(111)
    plt.rcParams['figure.figsize'] = [12, 5]
    for item in dados[coluna].unique():
        ix = dados[coluna] == item
        kmf.fit(T.loc[ix], C.loc[ix], label=str(item))
        ax = kmf.plot(ax=ax)
```

6.1 Kaplan-Meier main curve

[49]:
```
from lifelines import KaplanMeierFitter
from lifelines.statistics import multivariate_logrank_test
from lifelines.statistics import pairwise_logrank_test

kmf = KaplanMeierFitter()
T = dfCurvas['anosSocio']
C = dfCurvas['abandonou']
kmf.fit(T,C,label="Abandono dos sócios");
```

[50]:
```
tabela=pd.concat([kmf.event_table.reset_index(),
           kmf.conditional_time_to_event_.reset_index(),
           kmf.survival_function_.reset_index()],axis=1)
```

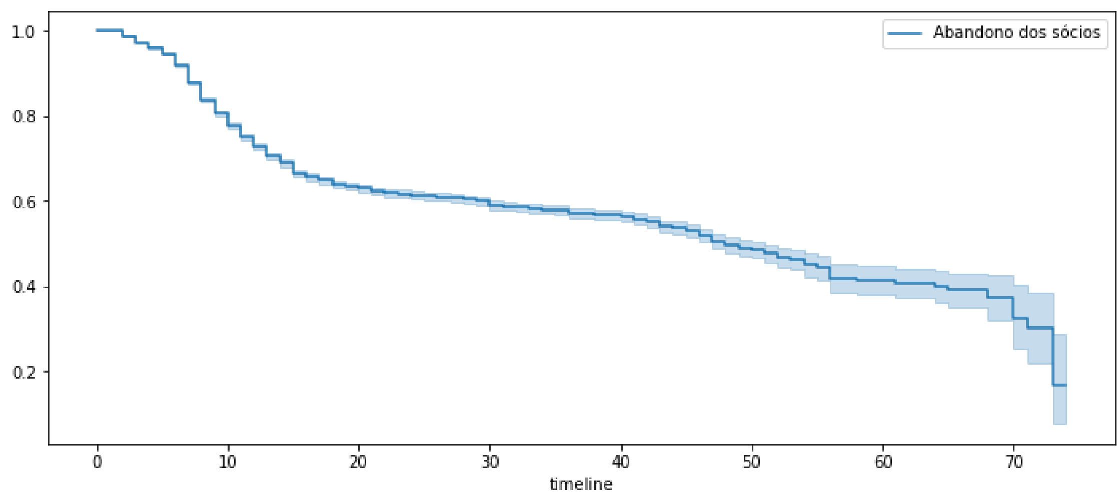
[51]:
```
tabela.columns = ['event_at', 'removed', 'observed', 'censored', 'entrance',␣
 ↪'at_risk','timeline',
                   'median duration remaining to event','timeline', 'Abandono␣
 ↪dos sócios']
```

[52]:
```
tabela.head(12)
```

[52]:
```
    event_at  removed  observed  censored  entrance  at_risk  timeline  \
0          0     1595         0      1595     25316    25316       0.0
1          1     1809         0      1809         0    23721       1.0
```

```
2           2    1132      261      871        0    21912       2.0
3           3    1019      318      701        0    20780       3.0
4           4     630      260      370        0    19761       4.0
5           5     827      264      563        0    19131       5.0
6           6    2111      534     1577        0    18304       6.0
7           7    1988      719     1269        0    16193       7.0
8           8    1942      652     1290        0    14205       8.0
9           9    1241      459      782        0    12263       9.0
10         10    1946      397     1549        0    11022      10.0
11         11     978      310      668        0     9076      11.0

    median duration remaining to event  timeline  Abandono dos sócios
0                                 48.0       0.0             1.000000
1                                 47.0       1.0             1.000000
2                                 47.0       2.0             0.988089
3                                 48.0       3.0             0.972968
4                                 47.0       4.0             0.960166
5                                 47.0       5.0             0.946916
6                                 48.0       6.0             0.919291
7                                 49.0       7.0             0.878473
8                                 48.0       8.0             0.838151
9                                 55.0       9.0             0.806780
10                                58.0      10.0             0.777720
11                                57.0      11.0             0.751157
```

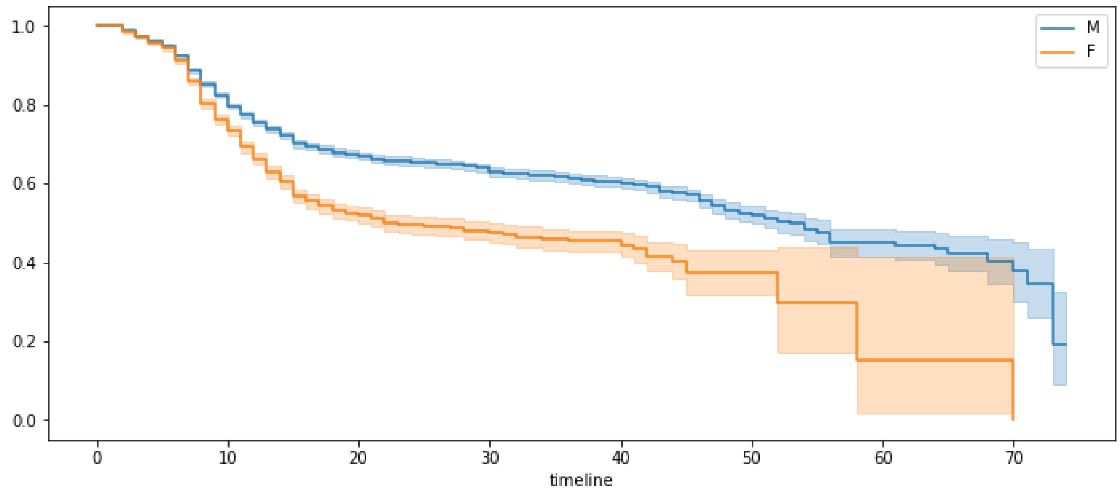
[53]:
```python
plt.rcParams['figure.figsize'] = [12, 5]

kmf.plot();
```



6.2 Por género

[54]:
```
print(dfCurvas.sexo.value_counts())
curvaSobrevivencia(dfCurvas,'sexo')
```

```
M    17246
F     8070
Name: sexo, dtype: int64
```



[55]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas.
↪sexo,event_observed=C)
results.print_summary()
```

	test_statistic	p	-log2(p)
0	194.625277	3.110248e-44	144.527807

[56]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas.
↪sexo,event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
F	M	194.625277	3.110248e-44	144.527807

6.3 MesesUP

[57]:
```
dfCurvas.mesesUP.describe()
```

[57]:
```
count    25316.000000
mean        18.814110
std         32.498248
```

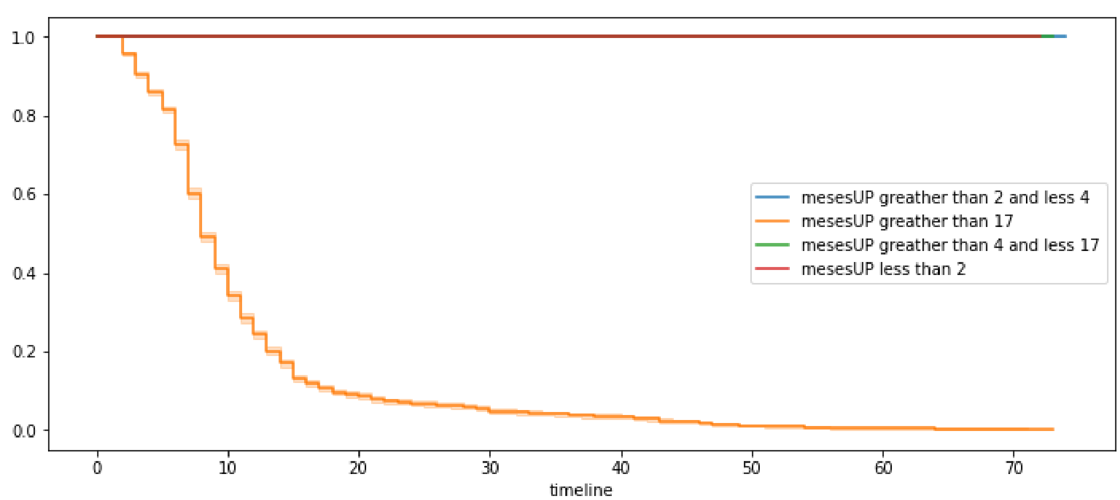
```
min           0.000000
25%           2.000000
50%           4.000000
75%          17.000000
max         156.000000
Name: mesesUP, dtype: float64
```

[58]:
```
var='mesesUP'
varEscalao='escMesesUP'
dfCurvas[varEscalao]=''
for index, cliente in dfCurvas.iterrows():
    #se a variável tiver o valor 1 colocar na nova variável a descrição da␣
 ↪atividade
    if cliente[var] <= 2:
        dfCurvas.at[index,varEscalao]=var+' less than 2'
    elif (cliente[var] > 2) & (cliente[var] <= 4):
        dfCurvas.at[index,varEscalao]=var+' greather than 2 and less 4'
    elif (cliente[var] > 4) & (cliente[var] <= 17):
        dfCurvas.at[index,varEscalao]=var + ' greather than 4 and less 17'
    elif (cliente[var] > 17):
        dfCurvas.at[index,varEscalao]=var + ' greather than 17'
```

[59]:
```
dfCurvas.escMesesUP.value_counts()
```

[59]:
```
mesesUP less than 2                      8814
mesesUP greather than 17                 6303
mesesUP greather than 4 and less 17      5246
mesesUP greather than 2 and less 4       4953
Name: escMesesUP, dtype: int64
```

[52]:
```
curvaSobrevivencia(dfCurvas,varEscalao)
```



[60]:

```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed
results.print_summary()
```

	test_statistic	p	-log2(p)
0	19998.897283	0.0	inf

[61]:

```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
mesesUP greather than 17	mesesUP greather than 2 and less 4	6442.374708	0.0	inf
	mesesUP greather than 4 and less 17	4353.192689	0.0	inf
	mesesUP less than 2	10330.892545	0.0	inf
mesesUP greather than 2 and less 4	mesesUP greather than 4 and less 17	0.000000	1.0	-0.0
	mesesUP less than 2	0.000000	1.0	-0.0
mesesUP greather than 4 and less 17	mesesUP less than 2	0.000000	1.0	-0.0

6.4 ValorTotal

[62]:

```
dfCurvas.valorTotal.describe()
```

[62]:

```
count    25316.000000
mean       316.037984
std        493.971528
min          0.000000
25%          5.000000
50%         53.000000
75%        448.250000
max       2602.000000
Name: valorTotal, dtype: float64
```

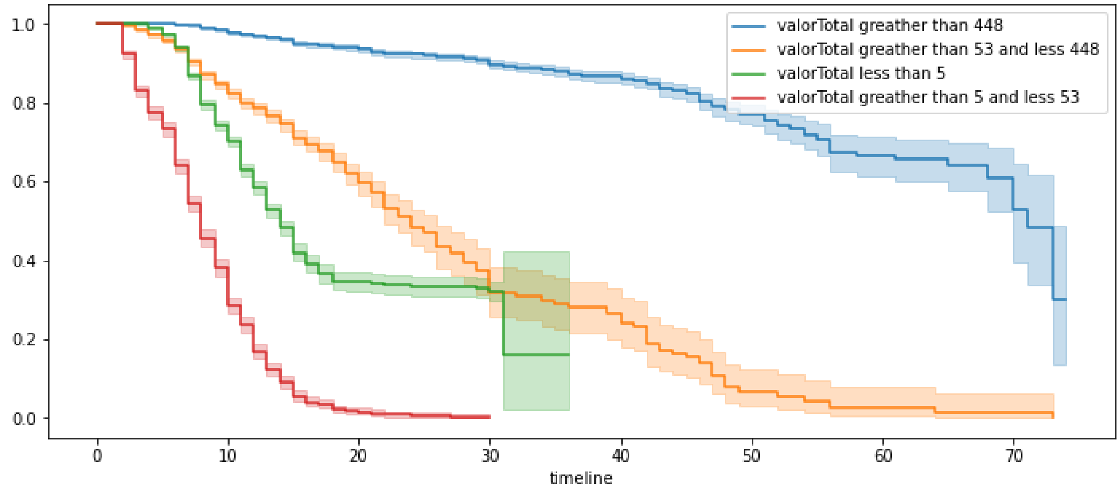
[63]:

```
var='valorTotal'
varEscalao='escValorTotal'
dfCurvas[varEscalao]=''
for index, cliente in dfCurvas.iterrows():
    #se a variável tiver o valor 1 colocar na nova variável a descrição da␣
 ↪atividade
    if cliente[var] <= 5:
        dfCurvas.at[index,varEscalao]=var+' less than 5'
    elif (cliente[var] > 5) & (cliente[var] <= 53):
        dfCurvas.at[index,varEscalao]=var+' greather than 5 and less 53'
    elif (cliente[var] > 53) & (cliente[var] <= 448):
        dfCurvas.at[index,varEscalao]=var + ' greather than 53 and less 448'
    elif (cliente[var] > 448):
        dfCurvas.at[index,varEscalao]=var + ' greather than 448'
```

[64]:
```
dfCurvas[varEscalao].value_counts()
```

[64]:
```
valorTotal less than 5                     7060
valorTotal greather than 448               6329
valorTotal greather than 53 and less 448   6280
valorTotal greather than 5 and less 53     5647
Name: escValorTotal, dtype: int64
```

[65]:
```
curvaSobrevivencia(dfCurvas,varEscalao)
```



[66]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed
results.print_summary()
```

	test_statistic	p	-log2(p)
0	9517.829603	0.0	inf

[67]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed=C)
results.print_summary()
```

		test_statistic	
valorTotal greather than 448	valorTotal greather than 5 and less 53	8318.461705	0.000000e-
	valorTotal greather than 53 and less 448	1527.147254	0.000000e-
	valorTotal less than 5	3177.425216	0.000000e-
valorTotal greather than 5 and less 53	valorTotal greather than 53 and less 448	2997.582565	0.000000e-
	valorTotal less than 5	2005.351350	0.000000e-
valorTotal greather than 53 and less 448	valorTotal less than 5	274.052087	1.485246e

6.5 quotaMensal

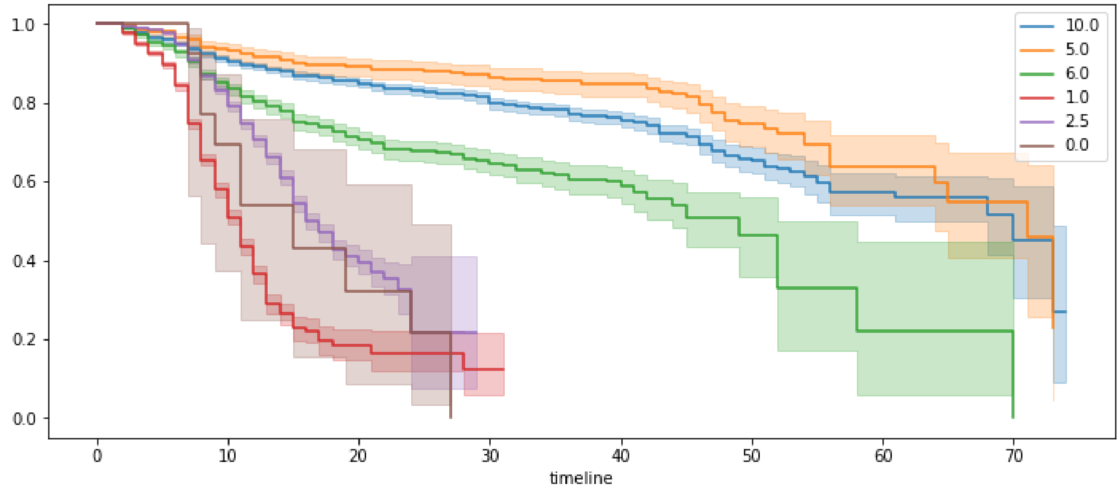
[68]:
```
varEscalao='quotaMensal'
dfCurvas[varEscalao].describe()
```

[68]:
```
count    25316.000000
mean         4.356099
std          3.550837
min          0.000000
25%          1.000000
50%          2.500000
75%          6.000000
max         10.000000
Name: quotaMensal, dtype: float64
```

[69]:
```
dfCurvas[varEscalao].value_counts()
```

[69]:
```
1.0     8016
2.5     7168
10.0    6126
6.0     3123
5.0      869
0.0       14
Name: quotaMensal, dtype: int64
```

[70]:
```
curvaSobrevivencia(dfCurvas,varEscalao)
```



[71]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed
results.print_summary()
```

	test_statistic	p	-log2(p)
0	3373.682348	0.0	inf

[72]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
0.0	1.0	0.837357	3.601539e-01	1.473315
	2.5	0.727220	3.937856e-01	1.344518
	5.0	76.734174	1.955894e-18	58.826878
	6.0	19.484981	1.013938e-05	16.589671
	10.0	53.939692	2.067388e-13	42.137256
1.0	2.5	1031.075028	3.160656e-226	749.095525
	5.0	653.856846	3.238374e-144	476.662376
	6.0	846.919933	3.397667e-186	616.114081
	10.0	2167.783976	0.000000e+00	inf
2.5	5.0	340.212952	5.734471e-76	249.946875
	6.0	178.329728	1.122322e-40	132.710637
	10.0	957.333303	3.378086e-210	695.848694
5.0	6.0	109.345324	1.363389e-25	82.601005
	10.0	16.291769	5.429930e-05	14.168707
6.0	10.0	159.387160	1.540097e-36	118.966390

6.6 jogosEpoca

[73]:
```
var='jogosEpoca'
varEscalao='escJogosEpoca'
dfCurvas[var].describe()
```

[73]:
```
count    25316.000000
mean         2.171631
std          4.076356
min          0.000000
25%          0.000000
50%          0.000000
75%          2.000000
max         16.000000
Name: jogosEpoca, dtype: float64
```

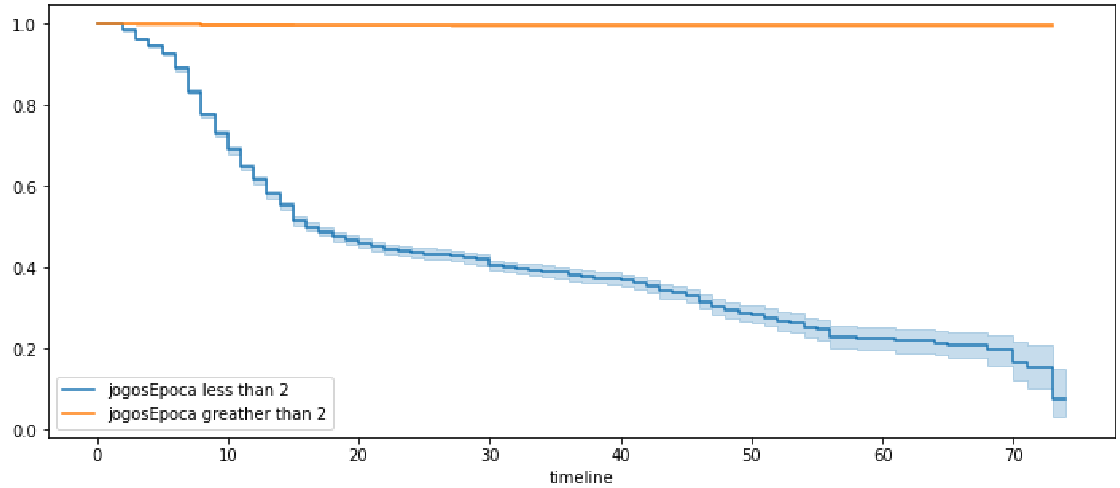
[74]:
```
dfCurvas[varEscalao]=''
for index, cliente in dfCurvas.iterrows():
    #se a variável tiver o valor 1 colocar na nova variável a descrição da␣
 ↪atividade
    if cliente[var] <= 2:
        dfCurvas.at[index,varEscalao]=var+' less than 2'
    elif (cliente[var] > 2):
```

```
        dfCurvas.at[index,varEscalao]=var + ' greather than 2'
```

[75]:
```
dfCurvas[varEscalao].value_counts()
```

[75]:
```
jogosEpoca less than 2        19015
jogosEpoca greather than 2     6301
Name: escJogosEpoca, dtype: int64
```

[76]:
```
curvaSobrevivencia(dfCurvas,varEscalao)
```



[77]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed
results.print_summary()
```

	test_statistic	p	-log2(p)
0	3270.332736	0.0	inf

[78]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[varEscalao],event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
jogosEpoca greather than 2	jogosEpoca less than 2	3270.332736	0.0	inf

6.7 escalaoTotalJogos

[79]:
```
var='escaloesTotalJogos'
dfCurvas[var].describe()
```

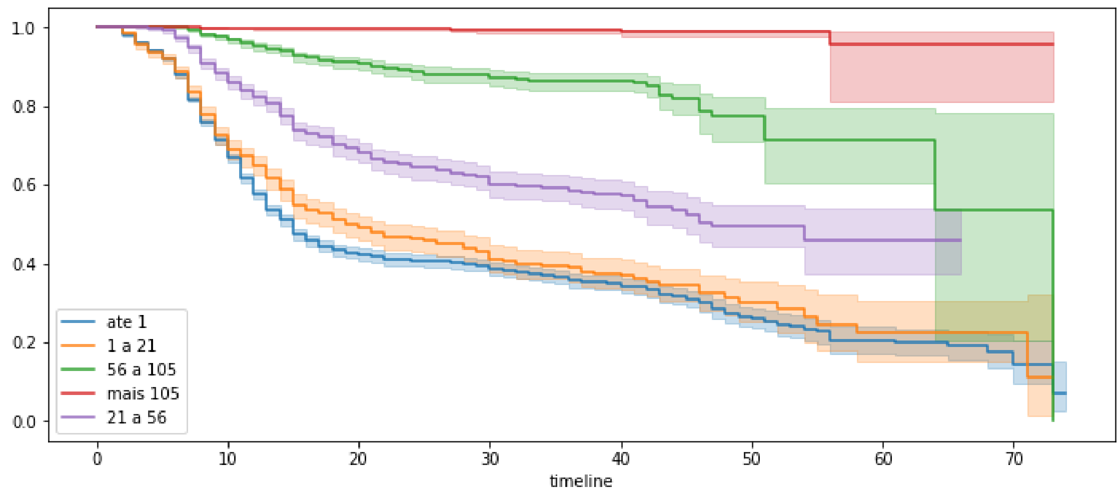
[79]:
```
count      25316
unique         5
```

```
top       ate 1
freq      15155
Name: escaloesTotalJogos, dtype: object
```

[80]:
```
dfCurvas[var].value_counts()
```

[80]:
```
ate 1        15155
1 a 21        2627
mais 105      2527
56 a 105      2519
21 a 56       2488
Name: escaloesTotalJogos, dtype: int64
```

[81]:
```
curvaSobrevivencia(dfCurvas,var)
```



[82]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[var],event_observed=C)
results.print_summary()
```

	test_statistic	p	-log2(p)
0	3147.499074	0.0	inf

[83]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[var],event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
1 a 21	21 a 56	159.947442	1.161801e-36	119.373049
	56 a 105	816.389467	1.474826e-179	594.064585
	ate 1	11.448851	7.153824e-04	10.448998
	mais 105	1574.871681	0.000000e+00	inf
21 a 56	56 a 105	279.881955	7.967468e-63	206.287349
	ate 1	361.252035	1.502998e-80	265.166405
	mais 105	851.984446	2.692429e-187	619.771645
56 a 105	ate 1	1204.146733	7.657725e-264	874.052101
	mais 105	217.912019	2.581281e-49	161.406390
ate 1	mais 105	1894.318290	0.000000e+00	inf

6.8 estadoCivil

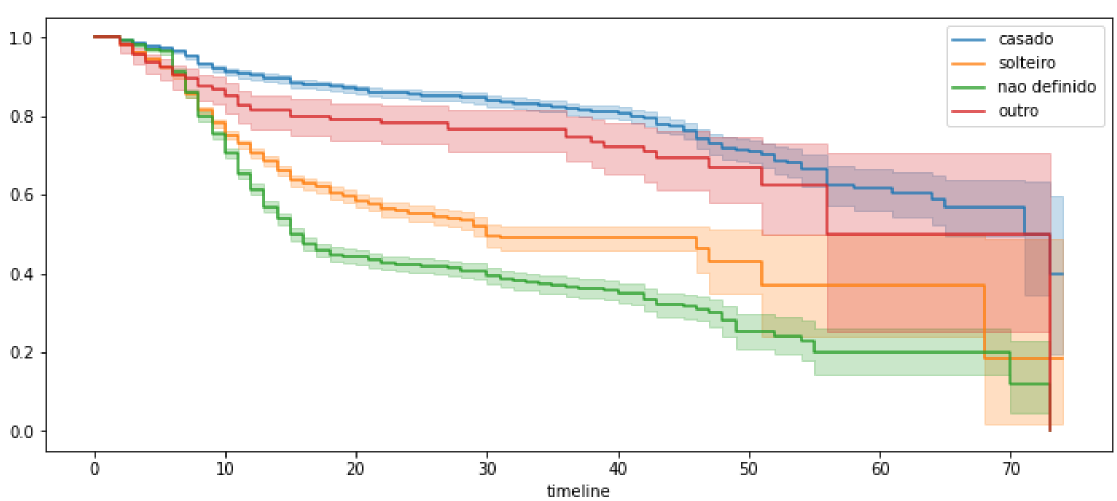
[84]:
```
var='estadoCivil'
dfCurvas[var].describe()
```

[84]:
```
count          25316
unique             4
top        solteiro
freq           12065
Name: estadoCivil, dtype: object
```

[85]:
```
dfCurvas[var].value_counts()
```

[85]:
```
solteiro         12065
nao definido      7667
casado            5085
outro              499
Name: estadoCivil, dtype: int64
```

[86]:
```
curvaSobrevivencia(dfCurvas,var)
```



[87]:
```
results=multivariate_logrank_test(event_durations=T,groups=dfCurvas[var],event_observed=C)
results.print_summary
```

[87]:
```
<bound method StatisticalResult.print_summary of <lifelines.StatisticalResult:
multivariate_logrank_test>
               t_0 = -1
 null_distribution = chi squared
degrees_of_freedom = 3
         test_name = multivariate_logrank_test

---
 test_statistic      p  -log2(p)
        1350.15 <0.005    969.05>
```

[88]:
```
results=pairwise_logrank_test(event_durations=T,groups=dfCurvas[var],event_observed=C)
results.print_summary()
```

		test_statistic	p	-log2(p)
casado	nao definido	1387.589094	1.045572e-303	1006.479921
	outro	16.817280	4.115683e-05	14.568509
	solteiro	763.130292	5.603198e-168	555.597669
nao definido	outro	84.704006	3.465475e-20	64.645509
	solteiro	86.339240	1.515709e-20	65.838569
outro	solteiro	35.301134	2.824676e-09	28.399268