Telco customer Churn

- Kaggle Telco customer churn dataset
- Kaggle Customer churn prediction dataset
- Objectives
  - Factor: Whether the customer has a partner or not (Yes, No)
  - Factor: Whether the customer has dependents or not (Yes, No)
  - Factor: Whether the customer has phone service or not (Yes, No)
  - Binary classification: Whether the customer churned or not (Yes, No)
  - Multiclass classification: Whether the customer has multiple lines or not (Yes, No, No phone service)
  - Regression: Monthly charges of the customer (Continuous, $18.25 - $118.75)
  - Clustering: Customers who contract several aspects of service (Yes, No, No phone service)
  - Clustering: Customers who contract several aspects of service (Yes, No, No phone service)
  - Churn: Target, whether they churn or not, Yes/No

```
In [1]: import pandas as pd
        import numpy as np
        import ydata as sp
        import matplotlib.pyplot as plt
        %matplotlib inline
In [2]: from ydata_sdgen.sdgen import *
        from ydata_sdgen.utils.preprocessing import *
        df.describe()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
count	3943-CRVFW	Female	0	Yes	Yes	72	Yes	No phone service	DSL	Yes	...
unique	7043	2	2	2	2	73	2	3	3	3	...
top	0002-ORFBO	Male	0	No	No	1	Yes	No	Fiber optic	No	...
freq	1	3555	5901	3641	4933	613	6361	3390	3096	3498	...

10 rows × 21 columns

```
In [3]: df.columns
```

```
Out[3]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
               'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
               'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
               'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
               'MonthlyCharges', 'TotalCharges', 'Churn'],
              dtype='object')
```

```
In [4]: df.info()
```

```
<class pandas.core.frame.DataFrame>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtype: float64(1), int64(2), object(18)
memory usage: 1.1 MB
```

Observation

- Binary categorical features
- Numerical features are stored as text strings
- Convert/clean and/or Tokenize categorical

```
In [5]: df.groupby('Churn').describe.transpose()
```

```
C:\Users\Administrator\AppDataLocal\Temp\ipykernel_4844\anaconda\envs\ydata\lib\site-packages\numpy\lib\nanfunc
In [6]: df.groupby('MultipleLines').describe.transpose()
```

	count	MonthlyCharges	TotalCharges
MultipleLines			
No	3390.0	36.10	2385.70
No phone service	682.0	18.95	361.83
Yes	2971.0	70.71	4247.37

```
In [7]: df.groupby('Churn').describe.transpose()
```

	count	MonthlyCharges	TotalCharges
Churn			
No	5174.0	61.27	3549.99
Yes	1869.0	74.44	1531.80

```
In [8]: df.groupby('Contract').describe.transpose()
```

	count	MonthlyCharges	TotalCharges
Contract			
Month-to-month	3875.0	66.40	2283.30
One year	1473.0	62.81	3879.91
Two year	1695.0	60.89	4032.35

```
In [9]: df.groupby('PaymentMethod').describe.transpose()
```

	count	MonthlyCharges	TotalCharges
PaymentMethod			
Bank transfer (automatic)	1544.0	67.19	3694.80
Credit card (automatic)	1522.0	67.09	3657.26
Electronic check	2365.0	78.99	2671.96
Mailed check	1612.0	61.07	1060.31

```
In [10]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [11]: df.groupby('PaymentMethod').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
PaymentMethod																
Bank transfer (automatic)	1544.0	67.19	34.599	18.25	25.275	70.350	89.850	118.75	1542.0	3694.804513	2735.951689	21.15	1024.275000	3423.31	5976.640000	8684.8
Credit card (automatic)	1522.0	67.09	34.415	18.40	25.150	70.100	89.400	118.75	1521.0	3657.264011	2733.075602	18.85	1022.950000	3394.55	5973.690000	8672.45
Electronic check	2365.0	78.99	28.512	18.25	56.100	84.450	94.200	118.35	2365.0	2671.963341	2307.198688	18.85	307.660000	1944.30	4054.650000	8684.8
Mailed check	1612.0	61.07	30.090	18.85	25.400	64.425	85.700	118.75	1604.0	1060.314607	1946.796392	18.85	77.093750	411.35	1320.750000	8670.1

```
In [12]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [13]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [14]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [15]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [16]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [17]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [18]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [19]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [20]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [21]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [22]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [23]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [24]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [25]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [26]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [27]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [28]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [29]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [30]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [31]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [32]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [33]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [34]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [35]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [36]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [37]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [38]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [39]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [40]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [41]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531.796094	1890.822994	18.85	134.500000	703.55	2331.300000	8672.45

```
In [42]: df.groupby('Churn').describe.transpose()[['MonthlyCharges', 'TotalCharges']]
```

	MonthlyCharges								TotalCharges							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Churn																
No	5174.0	61.27	31.092	18.25	25.100	64.425	88.475	118.75	5163.0	2555.344141	2329.456984	18.8	577.825000	1683.60	4264.275000	8684.8
Yes	1869.0	74.44	24.054	18.85	56.150	79.650	94.226	118.35	1869.0	1531

```
In [24]: CatBoost
```

```
[1]: Name: dtypes:
     0    gender    object
     1    age       float64
     2    hypertens int64
     3    heartdis  int64
     4    smoking   object
     5    bmi       float64
     6    HbA1c     float64
     7    glucose   int64
     8    diabetes  int64
     9    Female    bool
     10   Male      bool
     11   Other     bool
     12   No        bool
     13   Yes       bool
     14   Never     bool
     15   Not       bool
     16   Smokes    bool
     17   former    bool
     18   never     bool
     19   smokes    bool
     20   No.1      bool
     21   Yes.1     bool
     22   Female.1  bool
     23   Male.1    bool
     24   Other.1   bool
     25   No.2      bool
     26   Yes.2     bool
     27   Govt_job  bool
     28   Private   bool
     29   Self-emp  bool
     30   children  bool
     31   Urban     bool
     32   Rural     bool
     33   Unknown   bool
     34   Formerly  bool
     35   never     bool
     36   smokes    bool
     37   Unknown.1 bool
     dtype: int64
```

```
In [25]: df_encoded = df.drop(['Type', 'Residence'], axis=1)
         df_encoded.dtypes
```

```
[2]: gender    object
     age       float64
     hypertens int64
     heartdis  int64
     smoking   object
     bmi       float64
     HbA1c     float64
     glucose   int64
     diabetes  int64
     dtype: object
```

```
In [26]: df_encoded = pd.get_dummies(df_encoded, drop_first=True)
         df_encoded.dtypes
```

```
[3]: gender_Female    bool
     gender_Male      bool
     gender_Other     bool
     smoking_No       bool
     smoking_Yes      bool
     smoking_Never    bool
     smoking_Not      bool
     smoking_Smokes   bool
     smoking_former   bool
     smoking_never    bool
     smoking_smokes   bool
     smoking_No.1     bool
     smoking_Yes.1    bool
     smoking_Female.1 bool
     smoking_Male.1   bool
     smoking_Other.1  bool
     smoking_No.2     bool
     smoking_Yes.2    bool
     smoking_Govt_job bool
     smoking_Private  bool
     smoking_Self-emp bool
     smoking_children bool
     smoking_Urban    bool
     smoking_Rural    bool
     smoking_Unknown  bool
     smoking_Formerly bool
     smoking_never    bool
     smoking_smokes   bool
     smoking_Unknown.1 bool
     dtype: int64
```

```
In [27]: df_encoded.head()
```

	gender	age	hypertens	heartdis	smoking	bmi	HbA1c	glucose	diabetes	gender_Female	gender_Male	gender_Other	smoking_No	smoking_Yes	smoking_Never	smoking_Not	smoking_Smokes	smoking_former	smoking_never	smoking_smokes	smoking_No.1	smoking_Yes.1	smoking_Female.1	smoking_Male.1	smoking_Other.1	smoking_No.2	smoking_Yes.2	smoking_Govt_job	smoking_Private	smoking_Self-emp	smoking_children	smoking_Urban	smoking_Rural	smoking_Unknown	smoking_Formerly	smoking_never	smoking_smokes	smoking_Unknown.1
0	Female	80.0	0	1	never	25.19	6.6	140	0	True	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	Male	54.0	0	0	No	22.29	5.7	60	0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	Female	28.0	0	0	never	27.32	5.0	159	0	True	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	Male	36.0	0	0	former	23.45	5.0	155	0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	Female	76.0	1	1	smokes	20.14	4.8	70	0	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

```
In [28]: df.info()
```

```
[4]: <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 43400 entries, 0 to 43399
     Data columns (total 10):
     #   Column      Non-Null Count  Dtype
     ---  ------      --------------  -----
     0   gender      43400 non-null  object
     1   age         43400 non-null  float64
     2   hypertens   43400 non-null  int64
     3   heartdis    43400 non-null  int64
     4   smoking     43400 non-null  object
     5   bmi         43400 non-null  float64
     6   HbA1c       43400 non-null  float64
     7   glucose     43400 non-null  int64
     8   diabetes    43400 non-null  int64
     9   Type        43400 non-null  object
     dtypes: float64(4), int64(5), object(1)
     memory usage: 3.3 MB
```

Check feature importance

```
In [29]: CatBoostClassifier(iterations=100, verbose=False, random_state=42)
         model.fit(X_train, y_train)
         feature_importances = model.get_feature_importance()
         feature_names = X_train.columns
         for name, importance in zip(feature_names, feature_importances):
             print(f'{name}: {importance}')
```

```
[5]: hypertens: 0.4195
     heartdis: 0.3125
     HbA1c: 0.1429
     glucose: 0.1251
     gender_Female: 0.0000
     gender_Male: 0.0000
     gender_Other: 0.0000
     smoking_No: 0.0000
     smoking_Yes: 0.0000
     smoking_Never: 0.0000
     smoking_Not: 0.0000
     smoking_Smokes: 0.0000
     smoking_former: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_No.1: 0.0000
     smoking_Yes.1: 0.0000
     smoking_Female.1: 0.0000
     smoking_Male.1: 0.0000
     smoking_Other.1: 0.0000
     smoking_No.2: 0.0000
     smoking_Yes.2: 0.0000
     smoking_Govt_job: 0.0000
     smoking_Private: 0.0000
     smoking_Self-emp: 0.0000
     smoking_children: 0.0000
     smoking_Urban: 0.0000
     smoking_Rural: 0.0000
     smoking_Unknown: 0.0000
     smoking_Formerly: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_Unknown.1: 0.0000
     age: 0.0000
     bmi: 0.0000
     diabetes: 0.0000
```

```
In [30]: CatBoostClassifier(iterations=100, verbose=False, random_state=42)
         model.fit(X_train, y_train)
         feature_importances = model.get_feature_importance()
         feature_names = X_train.columns
         for name, importance in zip(feature_names, feature_importances):
             print(f'{name}: {importance}')
```

```
[6]: hypertens: 0.4195
     heartdis: 0.3125
     HbA1c: 0.1429
     glucose: 0.1251
     gender_Female: 0.0000
     gender_Male: 0.0000
     gender_Other: 0.0000
     smoking_No: 0.0000
     smoking_Yes: 0.0000
     smoking_Never: 0.0000
     smoking_Not: 0.0000
     smoking_Smokes: 0.0000
     smoking_former: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_No.1: 0.0000
     smoking_Yes.1: 0.0000
     smoking_Female.1: 0.0000
     smoking_Male.1: 0.0000
     smoking_Other.1: 0.0000
     smoking_No.2: 0.0000
     smoking_Yes.2: 0.0000
     smoking_Govt_job: 0.0000
     smoking_Private: 0.0000
     smoking_Self-emp: 0.0000
     smoking_children: 0.0000
     smoking_Urban: 0.0000
     smoking_Rural: 0.0000
     smoking_Unknown: 0.0000
     smoking_Formerly: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_Unknown.1: 0.0000
     age: 0.0000
     bmi: 0.0000
     diabetes: 0.0000
```

```
In [31]: CatBoostClassifier(iterations=100, verbose=False, random_state=42)
         model.fit(X_train, y_train)
         feature_importances = model.get_feature_importance()
         feature_names = X_train.columns
         for name, importance in zip(feature_names, feature_importances):
             print(f'{name}: {importance}')
```

```
[7]: hypertens: 0.4195
     heartdis: 0.3125
     HbA1c: 0.1429
     glucose: 0.1251
     gender_Female: 0.0000
     gender_Male: 0.0000
     gender_Other: 0.0000
     smoking_No: 0.0000
     smoking_Yes: 0.0000
     smoking_Never: 0.0000
     smoking_Not: 0.0000
     smoking_Smokes: 0.0000
     smoking_former: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_No.1: 0.0000
     smoking_Yes.1: 0.0000
     smoking_Female.1: 0.0000
     smoking_Male.1: 0.0000
     smoking_Other.1: 0.0000
     smoking_No.2: 0.0000
     smoking_Yes.2: 0.0000
     smoking_Govt_job: 0.0000
     smoking_Private: 0.0000
     smoking_Self-emp: 0.0000
     smoking_children: 0.0000
     smoking_Urban: 0.0000
     smoking_Rural: 0.0000
     smoking_Unknown: 0.0000
     smoking_Formerly: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_Unknown.1: 0.0000
     age: 0.0000
     bmi: 0.0000
     diabetes: 0.0000
```

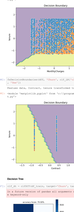
```
In [32]: CatBoostClassifier(iterations=100, verbose=False, random_state=42)
         model.fit(X_train, y_train)
         feature_importances = model.get_feature_importance()
         feature_names = X_train.columns
         for name, importance in zip(feature_names, feature_importances):
             print(f'{name}: {importance}')
```

```
[8]: hypertens: 0.4195
     heartdis: 0.3125
     HbA1c: 0.1429
     glucose: 0.1251
     gender_Female: 0.0000
     gender_Male: 0.0000
     gender_Other: 0.0000
     smoking_No: 0.0000
     smoking_Yes: 0.0000
     smoking_Never: 0.0000
     smoking_Not: 0.0000
     smoking_Smokes: 0.0000
     smoking_former: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_No.1: 0.0000
     smoking_Yes.1: 0.0000
     smoking_Female.1: 0.0000
     smoking_Male.1: 0.0000
     smoking_Other.1: 0.0000
     smoking_No.2: 0.0000
     smoking_Yes.2: 0.0000
     smoking_Govt_job: 0.0000
     smoking_Private: 0.0000
     smoking_Self-emp: 0.0000
     smoking_children: 0.0000
     smoking_Urban: 0.0000
     smoking_Rural: 0.0000
     smoking_Unknown: 0.0000
     smoking_Formerly: 0.0000
     smoking_never: 0.0000
     smoking_smokes: 0.0000
     smoking_Unknown.1: 0.0000
     age: 0.0000
     bmi: 0.0000
     diabetes: 0.0000
```

Check SHAP summary

```
In [33]: CatBoostClassifier(iterations=100, verbose=False, random_state=42)
         model.fit(X_train, y_train)
         explainer = shap.TreeExplainer(model)
         shap_values = explainer.shap_values(X_test)
         shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
[9]: C:\Users\HP\AppDATA\Local\Temp\ipykernel_8\2211.1\1\ipykernel_1\3\site-packages\sklearn\base\
     _estimator_type_ deprecated; check .._estimator_type_ instead.
     warnings.warn(f"{type(self).__name__._estimator_type_ deprecated; "
     C:\Users\HP\AppDATA\Local\Temp\ipykernel_8\2211.1\1\ipykernel_1\3\site-packages\shap\utils\
     _legacy.py:103: FutureWarning: Pass feature_perturbation="interventional" explicitly to silence this
     warning.
     warnings.warn("Pass feature_perturbation="interventional" explicitly to silence this warning.")
```



Observations: Contract Features are Monthly Charges, Gender, Contract

Check Power Predictive Scores

```
In [34]: df_encoded = df_encoded.drop(['Type', 'Residence'], axis=1)
         df_encoded.dtypes
```

```
[10]: gender    object
      age       float64
      hypertens int64
      heartdis  int64
      smoking   object
      bmi       float64
      HbA1c     float64
      glucose   int64
      diabetes  int64
      dtype: object
```

```
In [35]: df = df_encoded.drop(['Type', 'Residence'], axis=1)
         df.dtypes
```

```
[11]: gender    object
      age       float64
      hypertens int64
      heartdis  int64
      smoking   object
      bmi       float64
      HbA1c     float64
      glucose   int64
      diabetes  int64
      dtype: object
```

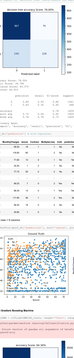
```
In [36]: df_encoded = df_encoded.drop(['Type', 'Residence'], axis=1)
         df_encoded.dtypes
```

```
[12]: gender    object
      age       float64
      hypertens int64
      heartdis  int64
      smoking   object
      bmi       float64
      HbA1c     float64
      glucose   int64
      diabetes  int64
      dtype: object
```

gender
age
hypertens
heartdis
smoking
bmi
HbA1c
glucose
diabetes
1.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00
0.00
1.00
0.00
0

```
In [20]: pycm_B = ConfusionMatrix(actual_vector=y_test, predict_vector=y_pred_B)
         pycm_B.plot(cmap=plt.cmap_hot,number_of_decimals=2,sort_by="TPR")
         pycm_B.stat(overall_param=["Accuracy"],class_param=["TPR"])
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```




```
Accuracy: 0.7000
Kappa: 0.4000
Bias: 0.7000
Prevalence: 0.5000
Overall Accuracy: 0.7000

                 class1  class2  TP Score  support
0                   0.7     0.7   0.70   10.00     10
1                   0.7     0.7   0.70   10.00     10
accuracy            0.7     0.7   0.70   20.00     20
macro avg           0.7     0.7   0.70   20.00     20
weighted avg        0.7     0.7   0.70   20.00     20

Overall Statistics :
{'Accuracy': 0.7000000000000002, 'Kappa': 0.4000000000000004, 'Bias': 0.7000000000000002, 'Prevalence': 0.5000000000000004}
```

```
In [21]: pycm_B.table[0][1]
```

```
Out[21]: 3
```

```
In [22]: pycm_B.table[1][0]
```

```
Out[22]: 3
```

```
In [23]: pycm_B.table[1][1]
```

```
Out[23]: 7
```

```
In [24]: pycm_B.print_stat(sort_by="TPR")
```

```
Overall Statistics :

Accuracy                                     0.7
Kappa                                        0.4
Bias                                         0.7
Prevalence                                   0.5
Overall Accuracy                             0.7

Class Statistics :

Class                                        class1  class2
TP                                           7       7
TN                                           7       7
FP                                           3       3
FN                                           3       3
TPR                                          0.7     0.7
TNR                                          0.7     0.7
PPV                                          0.7     0.7
NPV                                          0.7     0.7
FPR                                          0.3     0.3
FDR                                          0.3     0.3
FOR                                          0.3     0.3
FNR                                          0.3     0.3
ACC                                          0.7     0.7
F1                                           0.7     0.7
MCC                                          0.4     0.4
LR+                                          2.3     2.3
LR-                                          0.43    0.43
DOR                                          5.4     5.4
AUC                                          0.7     0.7
AGM                                          0.7     0.7
OP                                           0.4     0.4
IS                                           0.4     0.4
TOP                                          7       7
P                                            10      10
N                                            10      10
PRE                                          0.5     0.5
G                                            0.7     0.7
S                                            1.4     1.4
BM                                           0.4     0.4
MK                                           0.4     0.4
LH                                           7       7
PH                                           3       3
SBC                                          0.4     0.4
GI                                           0.4     0.4
AUCI                                         Very Good  Very Good
GM                                           0.7     0.7
DP                                           0.85    0.85
DC                                           0.7     0.7
TI                                           0.4     0.4
LD                                           0.3     0.3
CEN                                          0.4     0.4
MEN                                          0.4     0.4
DF                                           0.4     0.4
IBA                                          0.7     0.7
GAI                                          0.4     0.4
LIG                                          0.7     0.7
WRA                                          0.7     0.7
IDI                                          0.3     0.3
COP                                          0.4     0.4
KG                                           0.4     0.4
KD                                           0.7     0.7
VP                                           0.7     0.7
TPR Weighted                                 0.7
TNR Weighted                                 0.7
PPV Weighted                                 0.7
NPV Weighted                                 0.7
TP Macro                                     7
TN Macro                                     7
FP Macro                                     3
FN Macro                                     3
TPR Macro                                    0.7
TNR Macro                                    0.7
PPV Macro                                    0.7
NPV Macro                                    0.7
FPR Macro                                    0.3
FDR Macro                                    0.3
FOR Macro                                    0.3
FNR Macro                                    0.3
ACC Macro                                    0.7
F1 Macro                                     0.7
MCC Macro                                    0.4
LR+ Macro                                    2.3
LR- Macro                                    0.43
DOR Macro                                    5.4
AUC Macro                                    0.7
AGM Macro                                    0.7
OP Macro                                     0.4
IS Macro                                     0.4
TOP Macro                                    7
P Macro                                      10
N Macro                                      10
PRE Macro                                    0.5
G Macro                                      0.7
S Macro                                      1.4
BM Macro                                     0.4
MK Macro                                     0.4
LH Macro                                     7
PH Macro                                     3
SBC Macro                                    0.4
GI Macro                                     0.4
AUCI Macro                                   Very Good
GM Macro                                     0.7
DP Macro                                     0.85
DC Macro                                     0.7
TI Macro                                     0.4
LD Macro                                     0.3
CEN Macro                                    0.4
MEN Macro                                    0.4
DF Macro                                     0.4
IBA Macro                                    0.7
GAI Macro                                    0.4
LIG Macro                                    0.7
WRA Macro                                    0.7
IDI Macro                                    0.3
COP Macro                                    0.4
KG Macro                                     0.4
KD Macro                                     0.7
VP Macro                                     0.7
```

```
In [25]: pycm_B.save_stat("pycm_B",sort_by="TPR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

```
In [26]: pycm_B.save_vector("pycm_B",sort_by="TPR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

```
In [27]: pycm_B.save_html("pycm_B",sort_by="TPR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

```
In [28]: pycm_B.save_latex("pycm_B",sort_by="TPR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

```
In [29]: pycm_B.save_obj("pycm_B",sort_by="TPR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

```
In [30]: pycm_B.save_obj("pycm_B",sort_by="TPR",pickle=True)
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'pycm_B' is not defined
```

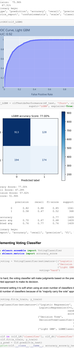
100 cm 5 Kaggle

```
In [31]: y_100kfold_pycm_B=ConfusionMatrix(actual_vector=y, predict_vector=y_100kfold)
```

Light Gradient Boosting Machine

```
In [32]: y_100kfold_pycm_B.plot(cmap=plt.cmap_hot,number_of_decimals=2,sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
ZeroDivisionError: Number of classes in classification or binarization mode should be one except for chi-squared, Tetrachoric and V-Cramer's Diagnostic.
```

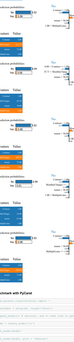



```
Accuracy: 0.9990
Kappa: 0.9980
Bias: 0.0010
Prevalence: 0.5000
Overall Accuracy: 0.9990

                 class1  class2  TNR Score  support
0                   1.0     1.0       1.00   100.00     100
1                   1.0     1.0       1.00   100.00     100
accuracy            1.0     1.0       1.00   200.00     200
macro avg           1.0     1.0       1.00   200.00     200
weighted avg        1.0     1.0       1.00   200.00     200

Overall Statistics :
{'Accuracy': 0.9990000000000001, 'Kappa': 0.9980000000000002, 'Bias': 0.0010000000000000002, 'Prevalence': 0.5000000000000004}
```

```
In [33]: y_100kfold_pycm_B=ConfusionMatrix(actual_vector=y, predict_vector=y_100kfold)
         y_100kfold_pycm_B.plot(cmap=plt.cmap_hot,number_of_decimals=2,sort_by="TNR")
```




```
Accuracy: 0.7700
Kappa: 0.5400
Bias: 0.2300
Prevalence: 0.5000
Overall Accuracy: 0.7700

                 precision  recall  F1 Score  support
0                     0.73    0.85      0.79   100.00     100
1                     0.82    0.69      0.75   100.00     100
accuracy              0.77    0.77      0.77   200.00     200
macro avg             0.78    0.77      0.77   200.00     200
weighted avg          0.78    0.77      0.77   200.00     200

Overall Statistics :
{'Accuracy': 0.7700000000000002, 'Kappa': 0.5400000000000004, 'Bias': 0.2300000000000002, 'Prevalence': 0.5000000000000004}
```

```
In [34]: y_100kfold_pycm_B.table[0][1]
```

```
Out[34]: 15
```

```
In [35]: y_100kfold_pycm_B.table[1][0]
```

```
Out[35]: 31
```

```
In [36]: y_100kfold_pycm_B.table[1][1]
```

```
Out[36]: 69
```

```
In [37]: y_100kfold_pycm_B.print_stat(sort_by="TNR")
```

```
Overall Statistics :

Accuracy                                     0.77
Kappa                                        0.54
Bias                                         0.23
Prevalence                                   0.5
Overall Accuracy                             0.77

Class Statistics :

Class                                        class1  class2
TP                                           85      69
TN                                           69      85
FP                                           31      15
FN                                           15      31
TNR                                          0.69    0.85
TPR                                          0.85    0.69
PPV                                          0.73    0.82
NPV                                          0.82    0.73
FPR                                          0.31    0.15
FDR                                          0.27    0.18
FOR                                          0.18    0.27
FNR                                          0.15    0.31
ACC                                          0.77    0.77
F1                                           0.79    0.75
MCC                                          0.54    0.54
LR+                                          2.7     4.6
LR-                                          0.22    0.36
DOR                                          12.3    12.8
AUC                                          0.77    0.77
AGM                                          0.77    0.77
OP                                           0.54    0.54
IS                                           0.54    0.54
TOP                                          85      69
P                                            100     100
N                                            100     100
PRE                                          0.5     0.5
G                                            0.79    0.75
S                                            1.5     1.4
BM                                           0.54    0.54
MK                                           0.54    0.54
LH                                           85      69
PH                                           31      15
SBC                                          0.54    0.54
GI                                           0.54    0.54
AUCI                                         Very Good  Very Good
GM                                           0.77    0.77
DP                                           0.85    0.85
DC                                           0.77    0.77
TI                                           0.54    0.54
LD                                           0.27    0.18
CEN                                          0.54    0.54
MEN                                          0.54    0.54
DF                                           0.54    0.54
IBA                                          0.77    0.77
GAI                                          0.54    0.54
LIG                                          0.77    0.77
WRA                                          0.77    0.77
IDI                                          0.27    0.18
COP                                          0.54    0.54
KG                                           0.54    0.54
KD                                           0.77    0.77
VP                                           0.77    0.77
TNR Weighted                                 0.77
TPR Weighted                                 0.77
PPV Weighted                                 0.78
NPV Weighted                                 0.78
TN Macro                                     77
TP Macro                                     77
FN Macro                                     23
FP Macro                                     23
TNR Macro                                    0.77
TPR Macro                                    0.77
PPV Macro                                    0.78
NPV Macro                                    0.78
FPR Macro                                    0.23
FDR Macro                                    0.22
FOR Macro                                    0.22
FNR Macro                                    0.23
ACC Macro                                    0.77
F1 Macro                                     0.77
MCC Macro                                    0.54
LR+ Macro                                    3.6
LR- Macro                                    0.29
DOR Macro                                    12.5
AUC Macro                                    0.77
AGM Macro                                    0.77
OP Macro                                     0.54
IS Macro                                     0.54
TOP Macro                                    77
P Macro                                      100
N Macro                                      100
PRE Macro                                    0.5
G Macro                                      0.77
S Macro                                      1.4
BM Macro                                     0.54
MK Macro                                     0.54
LH Macro                                     77
PH Macro                                     23
SBC Macro                                    0.54
GI Macro                                     0.54
AUCI Macro                                   Very Good
GM Macro                                     0.77
DP Macro                                     0.85
DC Macro                                     0.77
TI Macro                                     0.54
LD Macro                                     0.22
CEN Macro                                    0.54
MEN Macro                                    0.54
DF Macro                                     0.54
IBA Macro                                    0.77
GAI Macro                                    0.54
LIG Macro                                    0.77
WRA Macro                                    0.77
IDI Macro                                    0.22
COP Macro                                    0.54
KG Macro                                     0.54
KD Macro                                     0.77
VP Macro                                     0.77
```

```
In [38]: y_100kfold_pycm_B.save_stat("pycm_B",sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [39]: y_100kfold_pycm_B.save_vector("pycm_B",sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [40]: y_100kfold_pycm_B.save_html("pycm_B",sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [41]: y_100kfold_pycm_B.save_latex("pycm_B",sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [42]: y_100kfold_pycm_B.save_obj("pycm_B",sort_by="TNR")
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [43]: y_100kfold_pycm_B.save_obj("pycm_B",sort_by="TNR",pickle=True)
```

```
Error: Traceback (most recent call last):
  File "/usr/local/lib/python3.6/site-packages/IPyKernel/zmqshell.py", line 537, in run_cell_magic
    exec(compile(code, filename, "exec"), ns, ns)
  File "<ipyKernel_launcher_1>", line 1, in <module>
NameError: name 'y_100kfold_pycm_B' is not defined
```

```
In [44]:
```

Benchmark with PyCorrel

```
In [45]: from pycorrel import Correlation,Matrix,Table,Chi2,Cramer,T
```

```
In [46]: Correlation(x=y_100kfold, y=y_100kfold,bias=T)
```

```
Out[46]: {'Correlation': 0.99999999999999978, 'Chi2': 0.0, 'Cramer': 0.0, 'T': 0.0}
```

```
In [47]: Table(x=y_100kfold, y=y_100kfold,bias=T,sort=True,dic=True)
```

```
Out[47]: {'Table': array([[100, 100],
         [100, 100]]), 'Sum_of_rows': array([200, 200]), 'Sum_of_columns': array([200, 200]), 'Total': 400}
```

```
In [48]: Chi2(x=y_100kfold, y=y_100kfold,bias=T,c=False,dic=True)
```

```
Out[48]: {'Chi2': 0.0}
```

```
In [49]: Cramer(x=y_100kfold, y=y_100kfold,bias=T,c=False)
```

```
Out[49]: 0.0
```

```
In [50]: T(x=y_100kfold, y=y_100kfold,bias=T,c=False)
```

```
Out[50]: 0.0
```

Survival Analysis

- How to Fit Pocket and Prevent Cookiness Chunk

```
In [51]:
```