wb_km1

March 17, 2022

1 WB - milestone 1 - EDA

1.1 physioNet dataset

1.1.1 Autorzy:

Paulina Jaszczuk
Jędrzej Sokołowski
Filip Szympliński

[200]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import math
warnings.filterwarnings('ignore')

# ustawia domyślną wielkość wykresów
plt.rcParams['figure.figsize'] = (18,12)
# to samo tylko dla tekstu
plt.rcParams['font.size'] = 16
# ustawia wielkość tekstów dla wykresów seaborn zależną od wielkości wykresu
sns.set_context('paper', font_scale=2.5)
```

1.2 Import danych i poglądowe informacje

Zbiór danych medycznych physioNet opisujący pacjentów z oddziałów kariochirurgicznych.

Nasze dane zotały zebrane w formie szeregów czasowych. Każdy z 4000 pacjentów był reprezentowany jako osobny plik składający się z różnych informacji zdrowotnych zbieranych na przestrzeni 48h. Postanowiliśmy jednak pominąć zagadnienie szeregów czasowych i skupić się na klasycznym modelowaniu. W tym celu wyekstrahowaliśmy z danych medycznych średnią, minimum i maksimum każdej zmiennej (uznaliśmy, że będzie to próbka reprezentatywna) oraz zmienne charakteryzujące pacjenta. Dało to nam łącznie 118 zmiennych objaśniających.

[201]:
```
data = pd.read_csv("patients_data.csv", sep=",")
```

Zmienne wynikowe, których było 6 zostały przekształcone w jedną binarną zmienną `Survived` zawierającą wartość 0 w przypadku śmierci pacjenta i 1 w przeciwnym.

[202]:
```
outcomes_df = pd.read_csv("Outcomes-a.txt")
```

[203]:
```
survived_column = outcomes_df.apply(lambda row: 1 if row.Length_of_stay < row.
↪Survival or row.Survival == -1 else 0, axis=1)
```

[204]:
```
outcomes_df = outcomes_df.loc[:, ["RecordID"]]
```

[205]:
```
outcomes_df["Survived"] = survived_column
```

[206]:
```
outcomes_df.head()
```

[206]:

	RecordID	Survived
0	132539	1
1	132540	1
2	132541	1
3	132543	1
4	132545	1

[207]:
```
data = data.merge(outcomes_df, how='inner', on = "RecordID")
```

[208]:
```
data
```

[208]:

	Unnamed: 0	RecordID	Age	Gender	Height	ICUType	Weight	\
0	0	132539.0	54.0	0.0	-1.0	4.0	-1.000000	
1	1	132540.0	76.0	1.0	175.3	2.0	80.670588	
2	2	132541.0	44.0	0.0	-1.0	3.0	56.700000	
3	3	132543.0	68.0	1.0	180.3	3.0	84.600000	
4	4	132545.0	88.0	0.0	-1.0	3.0	-1.000000	
…	…	…	…	…	…	…	…	
3995	3995	142665.0	70.0	0.0	-1.0	4.0	87.000000	
3996	3996	142667.0	25.0	1.0	-1.0	3.0	166.400000	
3997	3997	142670.0	44.0	1.0	-1.0	3.0	109.000000	
3998	3998	142671.0	37.0	1.0	-1.0	3.0	87.400000	
3999	3999	142673.0	78.0	0.0	157.5	4.0	87.838889	

	Cholesterol_mean	Cholesterol_max	Cholesterol_min	…	WBC_mean	\
0	NaN	NaN	NaN	…	10.300000	
1	NaN	NaN	NaN	…	11.266667	
2	NaN	NaN	NaN	…	4.700000	
3	NaN	NaN	NaN	…	9.400000	
4	NaN	NaN	NaN	…	4.300000	
…	…	…	…	…	…	
3995	NaN	NaN	NaN	…	14.500000	
3996	117.0	117.0	117.0	…	4.733333	

```
3997              NaN              NaN              NaN  …  11.066667
3998              NaN              NaN              NaN  …  13.025000
3999              NaN              NaN              NaN  …   9.228571

      WBC_max  WBC_min  HCO3_mean  HCO3_max  HCO3_min  NIMAP_mean  NIMAP_max  \
0        11.2      9.4  27.000000      28.0      26.0   71.559118      92.33
1        13.3      7.4  22.333333      24.0      21.0   75.308571      88.33
2         6.2      3.7  25.000000      26.0      24.0   96.751316     110.00
3        11.5      7.9  27.666667      28.0      27.0   83.885517     100.70
4         4.8      3.8  19.000000      20.0      18.0   74.946512     105.70
…         …        …          …         …         …          …          …
3995     17.5     11.5  21.500000      22.0      21.0   79.144333     104.00
3996      6.5      3.0  23.800000      28.0      20.0   81.471364     100.70
3997     11.8     10.6  24.750000      26.0      23.0   80.529130     108.30
3998     14.7     11.5  26.750000      31.0      21.0         NaN        NaN
3999     11.9      4.8  19.666667      24.0      18.0   73.533333      92.00

      NIMAP_min  Survived
0         58.67         1
1         49.33         1
2         83.33         1
3         68.33         1
4         52.33         1
…         …       …
3995      60.00         1
3996      61.67         1
3997      60.33         1
3998        NaN         0
3999      52.00         1

[4000 rows x 116 columns]
```

[209]:
```
#Usuwamy sztuczną zmienną Unnamed
data = data.drop("Unnamed: 0", axis=1)
```

[210]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4000 entries, 0 to 3999
Columns: 115 entries, RecordID to Survived
dtypes: float64(114), int64(1)
memory usage: 3.5 MB
```

Zmienne możemy podzielić na medyczne oraz te charakteryzujące pacjenta. Wśród tych drugich wyróżniamy: ID, wiek, płeć (0 - kobieta, 1 - mężczyzna), wzrost, wagę oraz rodzaj oddziału, na którym pacjent przebywał (ICUType): 1 - oddział intensywnej opieki kardiologicznej, 2 - oddział ratunkowy kardiochirurgii, 3 - OIOM medyczny, 4 - OIOM chirurchiczny. Wśród danych medycznych są m.in.: cholesterol czy glukoza. Więcej informacji o danych można znaleźć pod linkiem:

data.

Wśród zmiennych medycznych jest zdecydowana przewaga zmiennych ilościowych - jedynie `MechVent` - kolumna odpowiadająca informacji, czy pacjent został poddany wentylacji z użyciem respiratora - jest binarna (0 - nie, 1 - tak)

Postanowiliśmy usunąć jeszcze zmienną `RecordID` - są to po prostu uporządkowane numery nadane pacjentom, które nie niosą żadnej informacji.

[211]:
```
#Usuwamy zmienną RecordID
data = data.drop("RecordID", axis=1)
```

1.3 Brakujące dane

Zanim zajmiemy się kolumnami z brakującymi wartościami, spójrzmy na wiersze.

[212]:
```
#wiersze, dla których brakuje ponad połowy wartości
rows_to_drop = []
for i in data.index:
  number_of_nulls = data.loc[[i]].isna().sum().sum()
  if number_of_nulls > data.shape[1]/2:
    rows_to_drop.append(i)

len(rows_to_drop)
```

[212]:
```
112
```

Doszliśmy do wniosku, że dane takie są mało reprezentatywne, dlatego postanowiliśmy je usunąć.

[213]:
```
data.drop(rows_to_drop, axis=0, inplace=True)
```

Teraz zajmijmy się kolumnami.

Już pierwszy rzut oka na dane pokazuje nam wartości `NaN`. Dodatkowo w opisie danych możemy przeczytać, że wartości brakujące oznaczone są przez -1.

[214]:
```
#To są faktyczne nulle
data[data == -1].count().nlargest(6)
```

[214]:
```
Height              1807
Weight               245
Gender                 2
Age                    0
ICUType                0
Cholesterol_mean       0
dtype: int64
```

Jak widać braków danych oznaczonych przez -1 jest dużo w kolumnie odpowiadającej wzrostowi oraz wadze. W kolumnie odpowiadającej płci są tylko 2. Te dwa wiersze możemy usunąć bez większych konsekwencji, natomiast dane o wysokości i wadze uzupełnimy przez medianę z podziałem na płcie.

[215]:
```
#usunięcie 2 wierszy z brakami danych w kolumnie z płcią
data.drop(data[data["Gender"] == -1].index, axis=0, inplace=True)
```

[216]:
```
#mediana wzrostu kobiet
women_height_median = data["Height"][(data["Gender"] == 0) & (data["Height"] !=␣
↪-1)].median()

#mediana wzrostu mężczyzn
men_height_median = data["Height"][(data["Gender"] == 1) & (data["Height"] !=␣
↪-1)].median()

#mediana wagi kobiet
women_weight_median = data["Weight"][(data["Gender"] == 0) & (data["Weight"] !=␣
↪-1)].median()

#mediana wagi mężczyzn
men_weight_median = data["Weight"][(data["Gender"] == 1) & (data["Weight"] !=␣
↪-1)].median()

#zastąpienie braków danych odpowiednimi medianami
data["Height"][(data["Gender"] == 0) & (data["Height"] == -1)] =␣
↪women_height_median
data["Height"][(data["Gender"] == 1) & (data["Height"] == -1)] =␣
↪men_height_median
data["Weight"][(data["Gender"] == 0) & (data["Weight"] == -1)] =␣
↪women_weight_median
data["Weight"][(data["Gender"] == 1) & (data["Weight"] == -1)] =␣
↪men_weight_median
```

[217]:
```
data[data == -1].count().nlargest(5)
```

[217]:
```
Age        0
Gender     0
Height     0
ICUType    0
Weight     0
dtype: int64
```

Wyeliminowaliśmy nulle pod postacią wartości -1.

[218]:
```
#to też nulle
data.isnull().sum()
```

[218]:
```
Age            0
Gender         0
Height         0
ICUType        0
```

```
Weight          0
               …
HCO3_min       10
NIMAP_mean    468
NIMAP_max     468
NIMAP_min     468
Survived        0
Length: 114, dtype: int64
```

[219]:
```
nulls = pd.DataFrame(data.isnull().sum())
nulls = nulls[nulls[0]>0]
nulls.shape
```

[219]:
```
(102, 1)
```

W danych jest też wiele innych wartości brakujących (występują w 105 kolumnach).

[220]:
```
#50 kolumn z największą liczbą braków danych
data.isna().sum().nlargest(50)
```

[220]:
```
TroponinI_mean      3682
TroponinI_max       3682
TroponinI_min       3682
Cholesterol_mean    3582
Cholesterol_max     3582
Cholesterol_min     3582
TroponinT_mean      3032
TroponinT_max       3032
TroponinT_min       3032
RespRate_mean       2816
RespRate_max        2816
RespRate_min        2816
Albumin_mean        2284
Albumin_max         2284
Albumin_min         2284
ALP_mean            2212
ALP_max             2212
ALP_min             2212
Bilirubin_mean      2184
Bilirubin_max       2184
Bilirubin_min       2184
ALT_mean            2181
ALT_max             2181
ALT_min             2181
AST_mean            2177
AST_max             2177
AST_min             2177
```

```
SaO2_mean          2108
SaO2_max           2108
SaO2_min           2108
Lactate_mean       1718
Lactate_max        1718
Lactate_min        1718
MechVent_mean      1394
MechVent_max       1394
MechVent_min       1394
FiO2_mean          1207
FiO2_max           1207
FiO2_min           1207
MAP_mean           1134
MAP_max            1134
MAP_min            1134
SysABP_mean        1128
SysABP_max         1128
SysABP_min         1128
DiasABP_mean       1128
DiasABP_max        1128
DiasABP_min        1128
PaCO2_mean          906
PaCO2_max           906
dtype: int64
```

Postanowiliśmy usunąć kolumny, w których brakuje więcej niż 50% danych.

[221]:
```
#usuwamy kolumny, w których +50% wartości to nulle
columns_to_drop = nulls[nulls[0]>data.shape[0]/2].index
data.drop(columns_to_drop, axis=1, inplace=True)
```

[222]:
```
data.shape
```

[222]:
```
(3886, 84)
```

Ciekawie sytuacja wygląda ze zmienną `MechVent`. Jest ona binarna. Dodatkowo po głębszej analizie widać, że jedyne wartości nie brakujące jakie przyjmuje to 1 (wentylacja była stosowana). Może to świadczyć o tym, że dane były zapisywane tylko w sytuacji, gdy pacjent rzeczywiście był wentylowany. Dlatego dane brakujące postanawiamy uzupełnić wartościami 0 (pacjent nie był wentylowany) oraz usunąć zmienne `MechVent_min` oraz `MechVent_mean`.

[223]:
```
#brak danych o innych wartosciach niz 1
data[(data["MechVent_min"]<1) | (data["MechVent_max"]<1) |␣
↪(data["MechVent_mean"]<1)]
```

[223]:
```
Empty DataFrame
Columns: [Age, Gender, Height, ICUType, Weight, MAP_mean, MAP_max, MAP_min,
HCT_mean, HCT_max, HCT_min, SysABP_mean, SysABP_max, SysABP_min, NIDiasABP_mean,
```

```
NIDiasABP_max, NIDiasABP_min, Lactate_mean, Lactate_max, Lactate_min, HR_mean,
HR_max, HR_min, FiO2_mean, FiO2_max, FiO2_min, Urine_mean, Urine_max, Urine_min,
BUN_mean, BUN_max, BUN_min, Mg_mean, Mg_max, Mg_min, Na_mean, Na_max, Na_min,
MechVent_mean, MechVent_max, MechVent_min, K_mean, K_max, K_min, PaCO2_mean,
PaCO2_max, PaCO2_min, pH_mean, pH_max, pH_min, GCS_mean, GCS_max, GCS_min,
Platelets_mean, Platelets_max, Platelets_min, Temp_mean, Temp_max, Temp_min,
NISysABP_mean, NISysABP_max, NISysABP_min, PaO2_mean, PaO2_max, PaO2_min,
Glucose_mean, Glucose_max, Glucose_min, Creatinine_mean, Creatinine_max,
Creatinine_min, DiasABP_mean, DiasABP_max, DiasABP_min, WBC_mean, WBC_max,
WBC_min, HCO3_mean, HCO3_max, HCO3_min, NIMAP_mean, NIMAP_max, NIMAP_min,
Survived]
Index: []

[0 rows x 84 columns]
```

[224]:
```
#usuwamy kolumny MechVent_min i MechVent_mean
data.drop(["MechVent_min", "MechVent_mean"], axis=1, inplace=True)
```

[225]:
```
#uzupełniamy wartości brakujące w kolumnie MechVent_max zerami
data["MechVent_max"][data["MechVent_max"].isna()] = 0
```

Pozostałe zmienne uzupełnimy średnią z podziałem na płcie (IterativeInputer z pakietu Sklearn niestety nadal zostawiał dużą liczbę kolumn z wieloma brakami danych).

[226]:
```
nulls.drop(columns_to_drop, axis=0, inplace=True)
nulls.drop(["MechVent_min", "MechVent_mean", "MechVent_max"], axis=0,␣
↪inplace=True)
```

[227]:
```
for column in nulls.index:
  women_mean = data[column][(data["Gender"] == 0) & (data[column].notna())].
↪mean()
  men_mean = data[column][(data["Gender"] == 1) & (data[column].notna())].mean()
  data[column][(data["Gender"] == 0) & (data[column].isna())] = women_mean
  data[column][(data["Gender"] == 1) & (data[column].isna())] = men_mean
```

[228]:
```
data.isna().sum().nlargest(5)
```

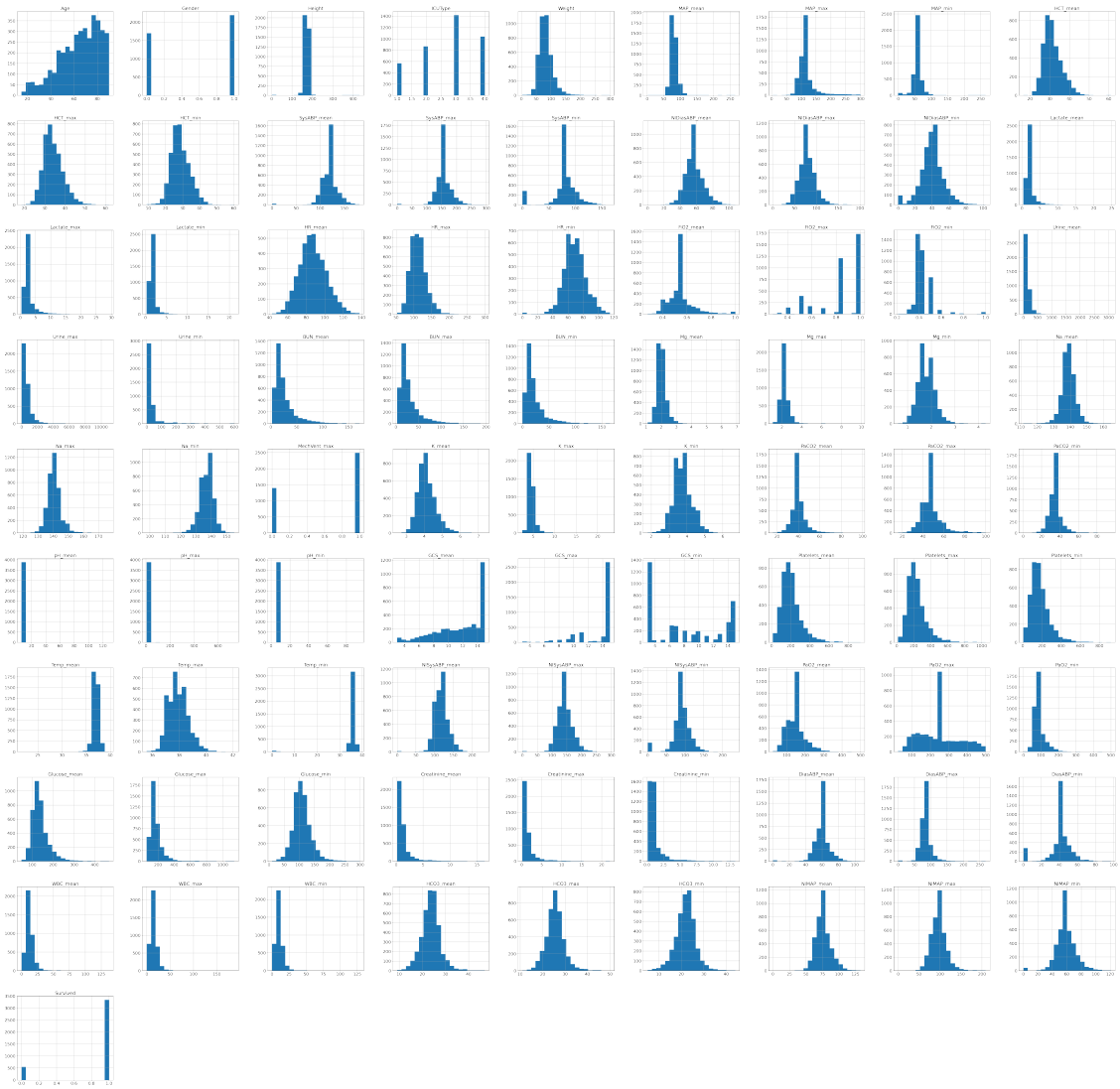
[228]:
```
Age        0
Gender     0
Height     0
ICUType    0
Weight     0
dtype: int64
```

Zero braków danych! :)

1.4 Rozkłady zmiennych

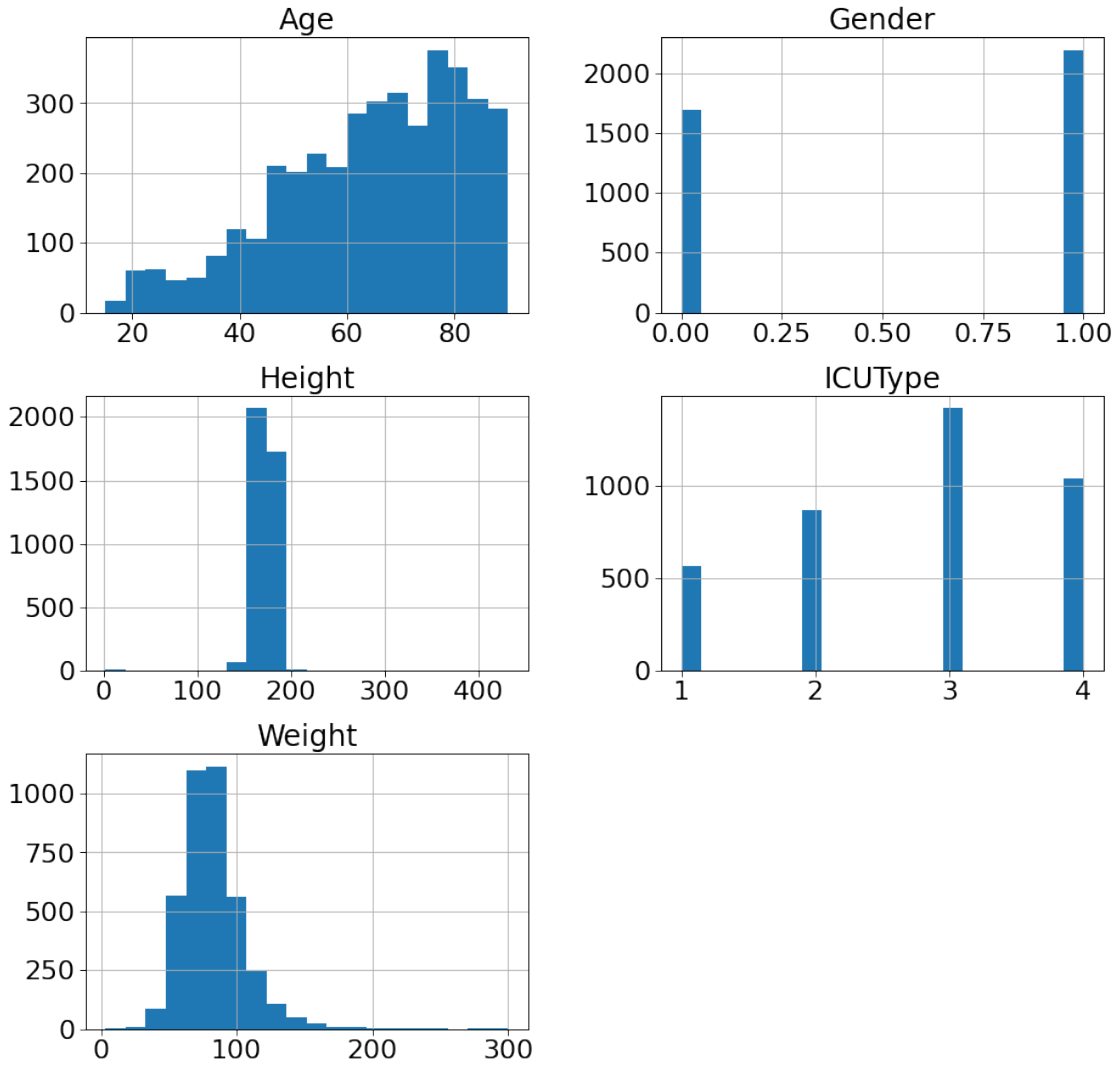
Przyjrzyjmy się najpierw wszystkim danym, by potem rozważać je dokładniej.

[229]:
```
data.hist(bins = 20, figsize=(100, 100))
plt.show()
```



Spójrzmy teraz dokładniej na rozkłady zmiennych charakteryzujących pacjenta.

[230]:
```
static_cols =  ['Age', 'Gender', 'Height', 'ICUType', 'Weight']

data[static_cols].hist(bins = 20, figsize=(15, 15))
plt.show()
```



[231]:
```
for column in static_cols:
  print(column + " min: " + "%.3f" % data[column].min())
  print(column + " max: " + "%.3f" % data[column].max())
```

```
Age min: 15.000
Age max: 90.000
Gender min: 0.000
Gender max: 1.000
Height min: 1.800
Height max: 431.800
ICUType min: 1.000
ICUType max: 4.000
Weight min: 3.500
Weight max: 300.000
```

Jak widać, na oddziale przebywały głównie osoby starsze. Najmłodsza z nich miała 15 lat. Częściej byli to również mężczyźni (co jest intuicyjne - według badań zapadają oni częściej na choroby sercowe). Jeśli chodzi jednak o wzrost i wagę, to możemy pokusić się o stwierdzenie, że występują tam błędy danych - wartości takie jak 1,8cm, 431cm, 0,55kg czy 448 kg z pewnością nie są prawidłowe.

[232]:
```
#usunięcie rekordów z błędnymi danymi wzrostu i wagi
wrong_indexes = data[(data["Height"]<100) | (data["Height"]>250) |␣
↪(data["Weight"]<30) | (data["Weight"]>300)].index
data.drop(wrong_indexes, axis=0, inplace=True)
```

Według opisu w naszych danych występują też dwie zmienne o ograniczonym zakresie - `GCS` (Glasgow Coma Score - skala używana w medycynie w celu oceny poziomu przytomności [3-15]) oraz `pH` (parametr gazometrii krwii tętniczej [0-14]). Sprawdźmy, czy i tu nie mamy błędów danych.

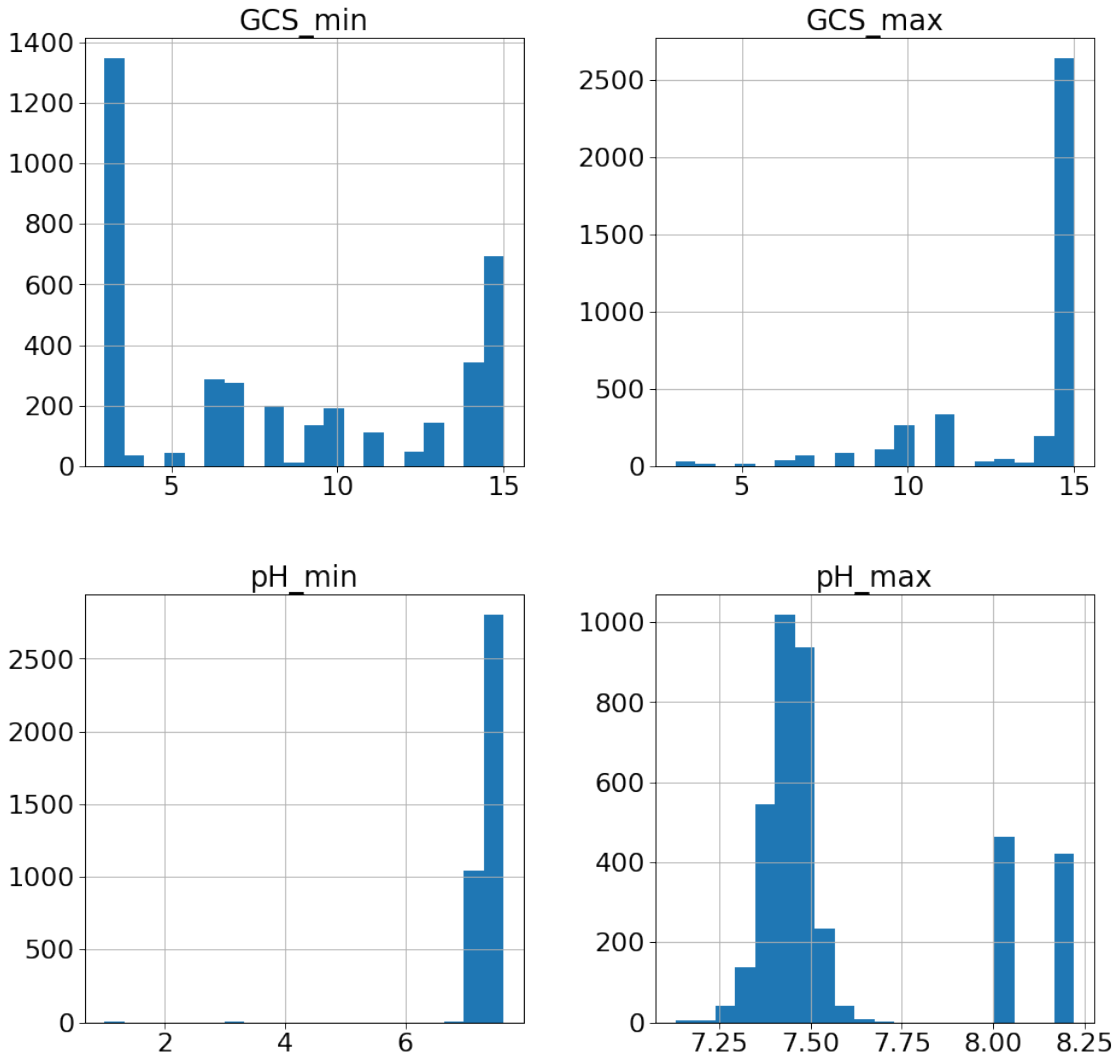
[233]:
```
cols = ["GCS_min", "GCS_max", "pH_min", "pH_max"]
for column in cols:
  print(column + " min: " + "%.3f" % data[column].min())
  print(column + " max: " + "%.3f" % data[column].max())
```

```
GCS_min min: 3.000
GCS_min max: 15.000
GCS_max min: 3.000
GCS_max max: 15.000
pH_min min: 1.000
pH_min max: 94.000
pH_max min: 7.130
pH_max max: 735.000
```

`GSC` jak widać mieści się w zakresie, natomiast pH juz nie - górny kres wychodzi stanowczo ponad 14.

[234]:
```
wrong_indexes_pH = data[(data["pH_min"]>14) | (data["pH_mean"]>14) |␣
↪(data["pH_max"]>14)].index
#jest 6 takich rekordów
data.drop(wrong_indexes_pH, axis=0, inplace=True)
```

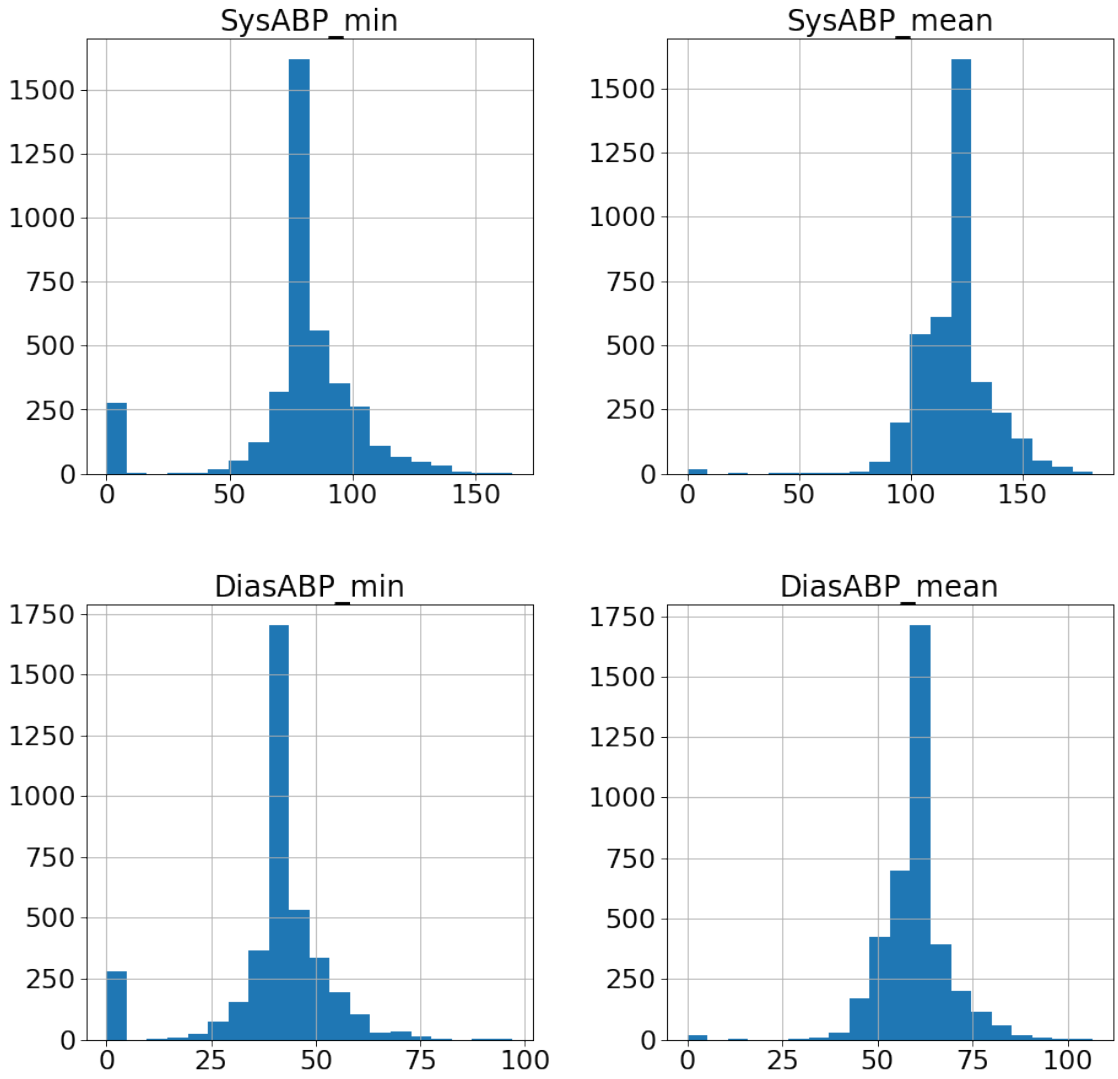
[235]:
```
data[cols].hist(bins = 20, figsize=(15, 15))
plt.show()
```



Jak widzimy dalej - `GCS` często osiąga swoją najwyższą wartość, zaś `pH` niemal nie spada poniżej 7 oraz nie wzrasta powyżej 9.

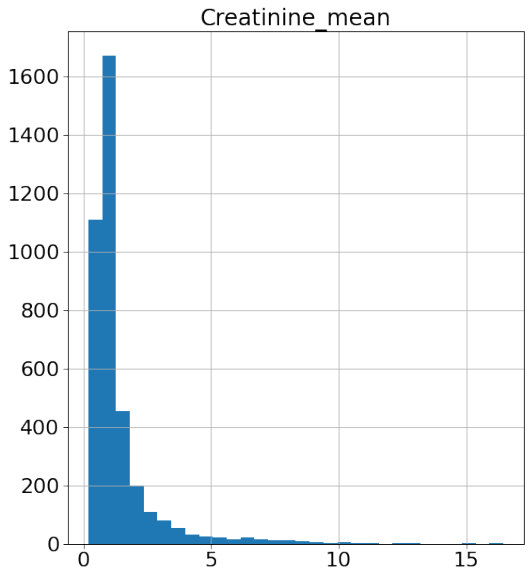
Na wielu histogramach możemy zauważyć, że część wartości nienaturalnie przyjmuje wartość 0 (np. `SysABP` - nieinwazyjne skurczowe ciśnienie tętnicze krwi czy `DiasABP` - inwazyjne rozkurczowe ciśnienie tętnicze krwi). Trudno jednak stwierdzić, czy jest to błąd w danych czy ma to uzasadnienie.

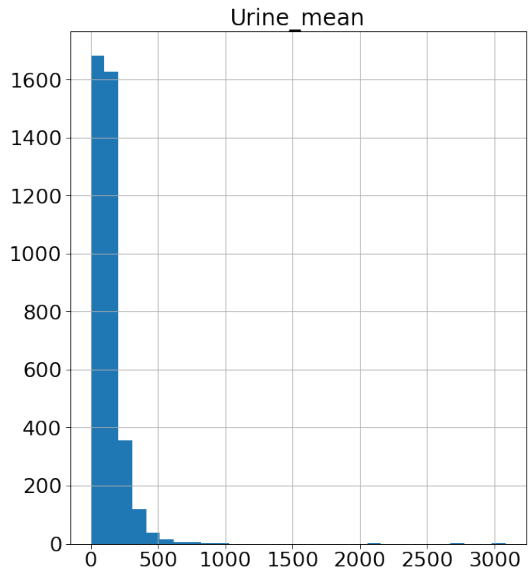
[236]:
```
cols_with_zeros = ['SysABP_min', 'SysABP_mean', 'DiasABP_min', "DiasABP_mean"]
data[cols_with_zeros].hist(bins = 20, figsize=(15, 15))
plt.show()
```



Z kolei inne zmienne skupione są w okolicach zera i mają nieliczne duże wartości (`Creatinine` - kreatynina w surowicy czy `Urine` - wydalanie moczu). Być może warto je zlogarytmować na dalszych etapach pracy.

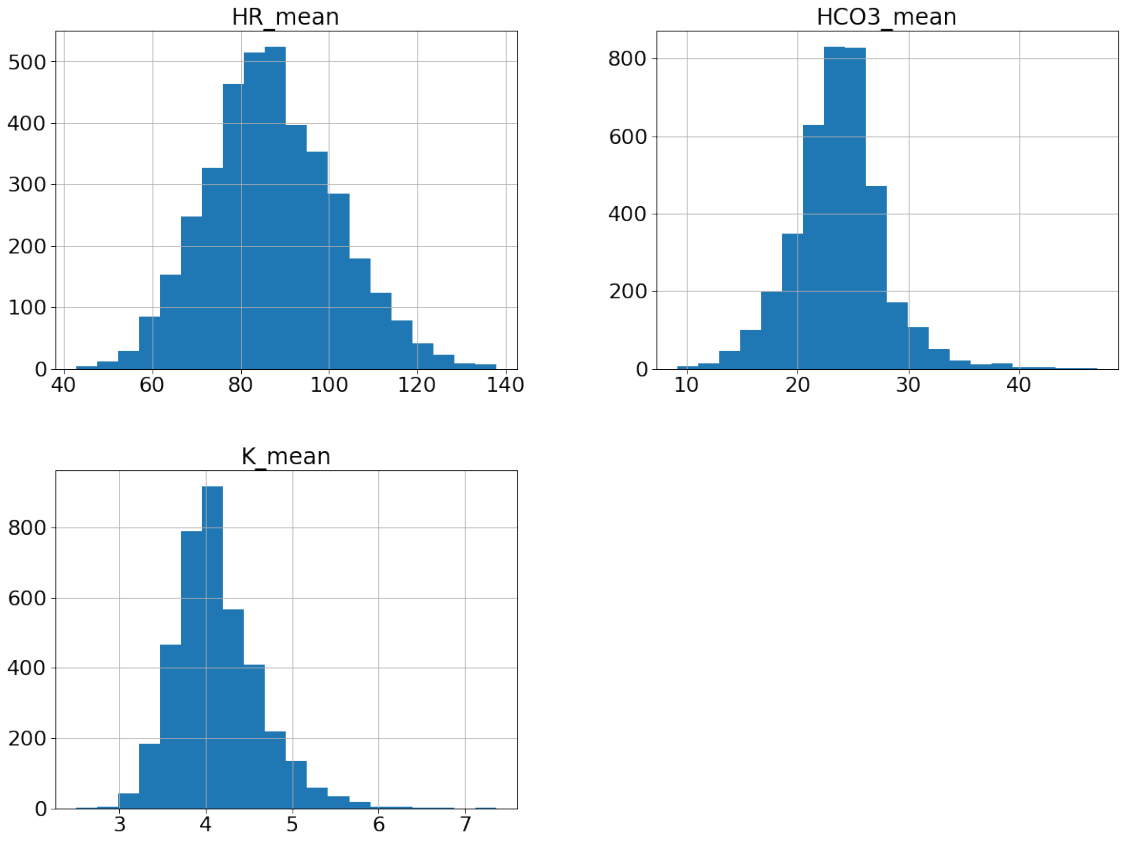
[237]:
```
log_cols = ["Creatinine_mean", "Urine_mean"]
data[log_cols].hist(bins=30, figsize=(20, 10))
plt.show()
```





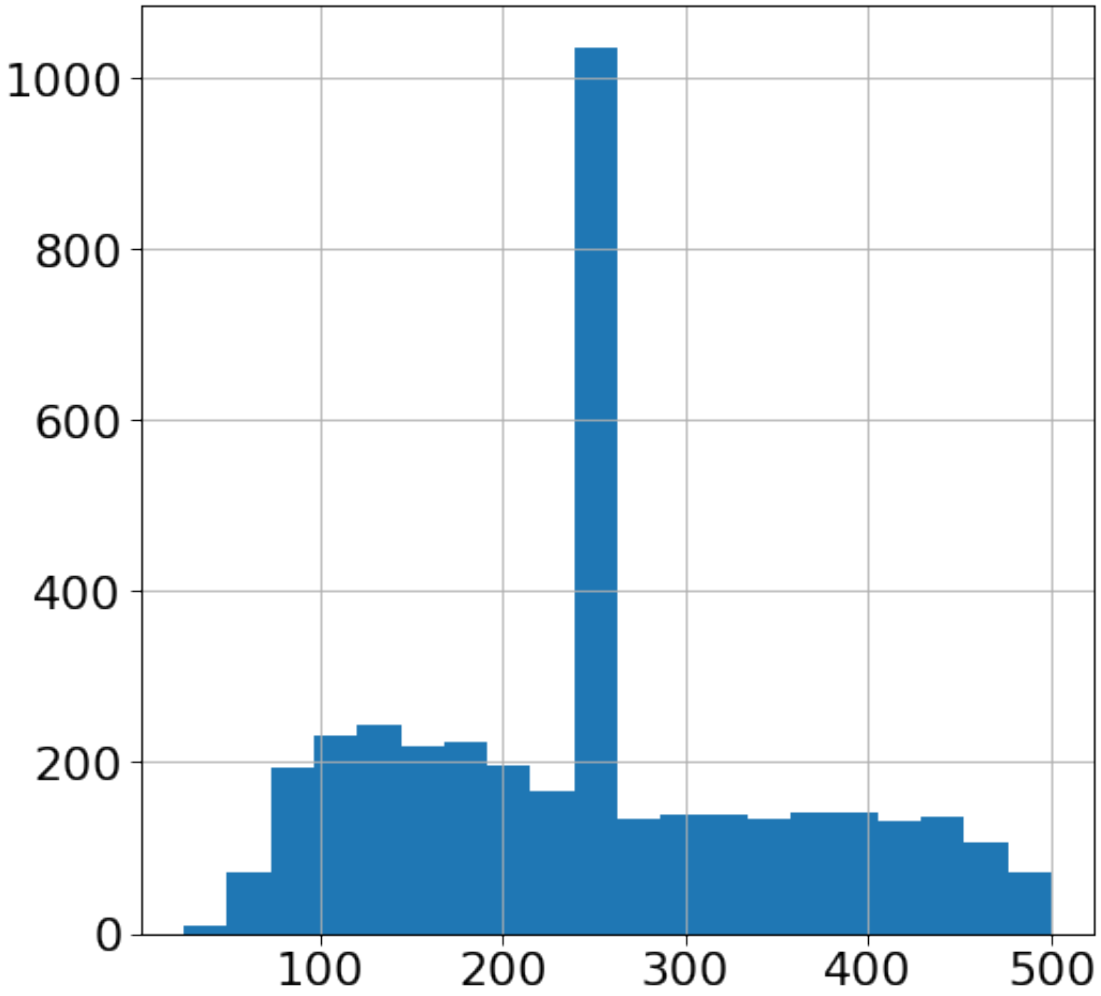
Część zmiennych przyjmuje wartości o rozkładzie zbliżonym do rozkładu normalnego z cięższymi lub lżejszymi ogonami - są to m.in. `HR` - tętno, `HCO3` - wodorowęglan w surowicy czy `K` - potas w surowicy.

[238]:
```
normal_cols = ['HR_mean', 'HCO3_mean', "K_mean"]
data[normal_cols].hist(bins = 20, figsize=(20, 15))
plt.show()
```



Część zmiennych wyróżnia się tym, że dominuje w nich jedna wartość (wartości z jednego kubełka), np. wartość maksymalna `PaO2` - ciśnienie parcjalne tętnicze.

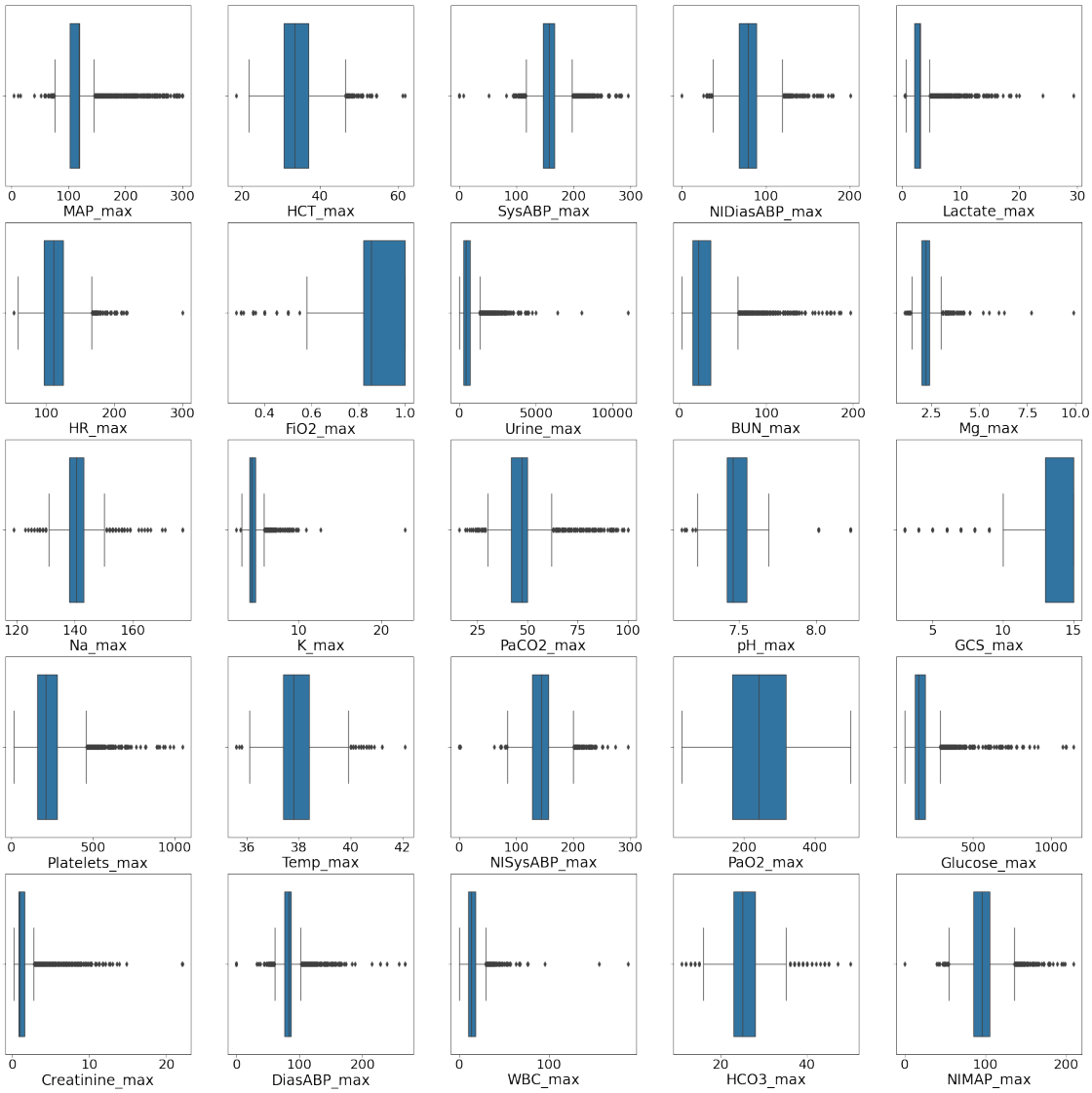
[239]:
```
data["PaO2_max"].hist(bins = 20, figsize=(8, 8))
plt.show()
```



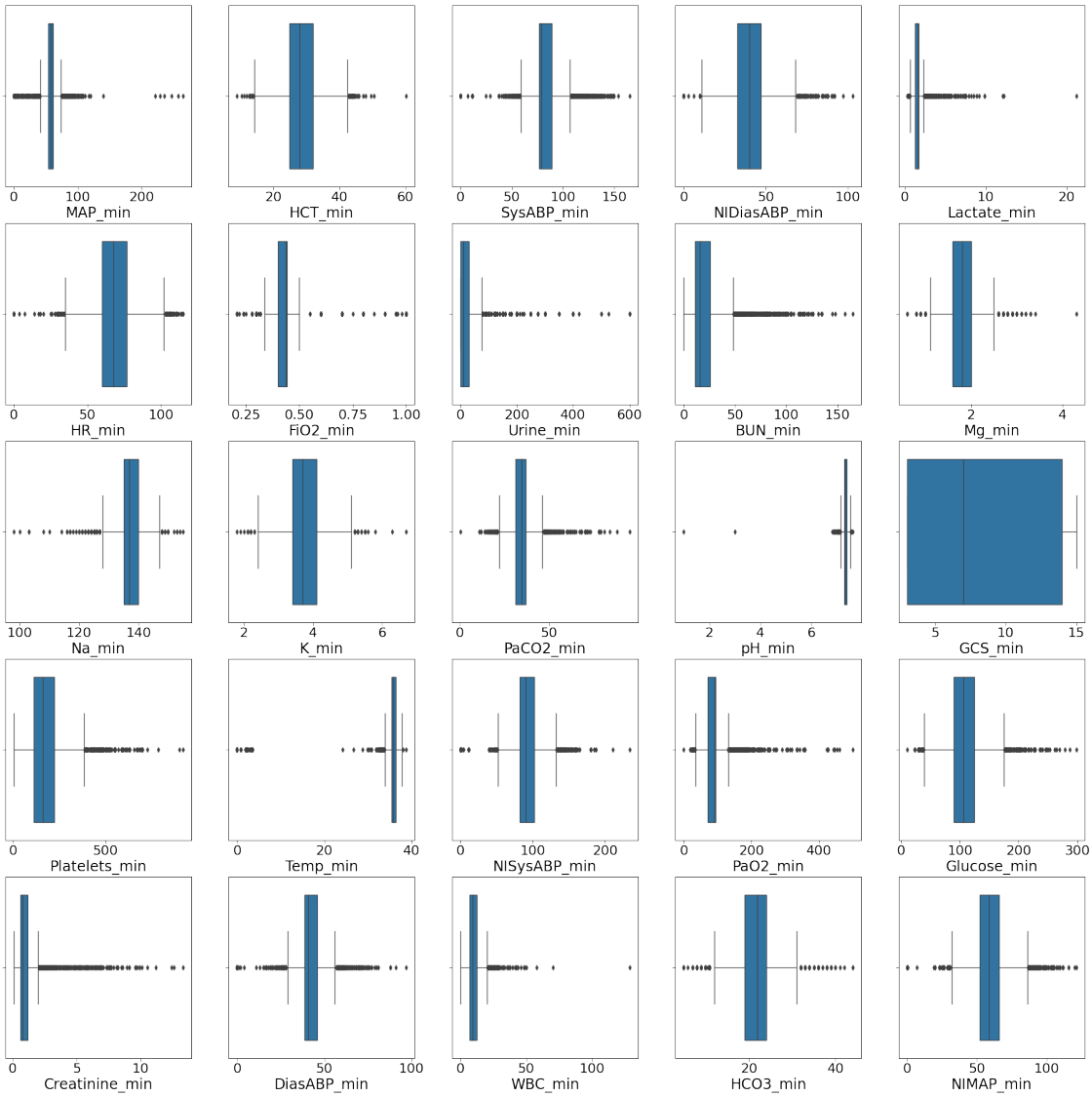
Wyraźnie widać, że w znacznej większości przypadków maksymalna wartość zmiennej oscyuje wokół 250.

1.5 Outliery

[240]:
```
#boxploty zmiennych dynamicznych (outliery prawostronne)
cols = [w for w in data.columns if w.endswith('max')]
cols.remove("MechVent_max")
fig, axes = plt.subplots(5,5, figsize = (32, 32))

for col, ax in zip(cols, axes.flatten()):
    sns.boxplot(x=col, data=data, orient="h", ax=ax)
```



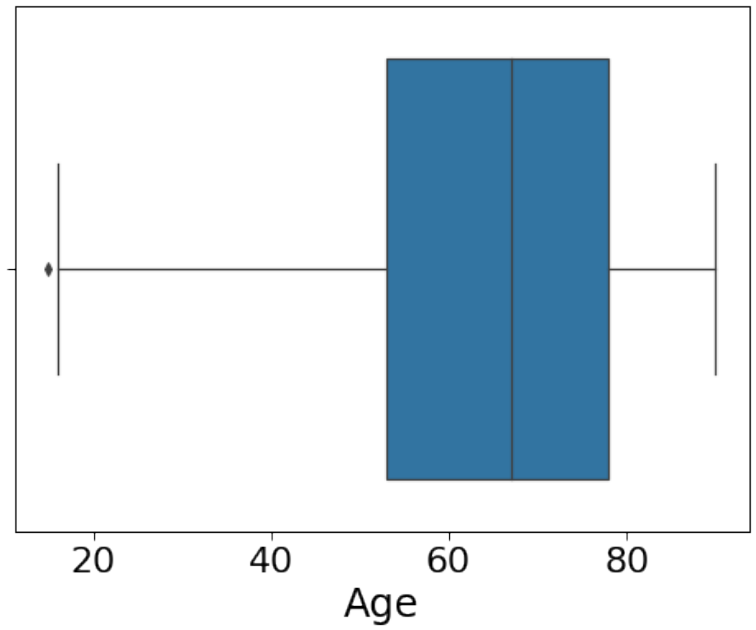
[241]:
```
#boxploty zmiennych dynamicznych (outliery lewostronne)
cols = [w for w in data.columns if w.endswith('min')]
fig, axes = plt.subplots(5,5, figsize = (32, 32))

for col, ax in zip(cols, axes.flatten()):
    sns.boxplot(x=col, data=data, orient="h", ax=ax)
```

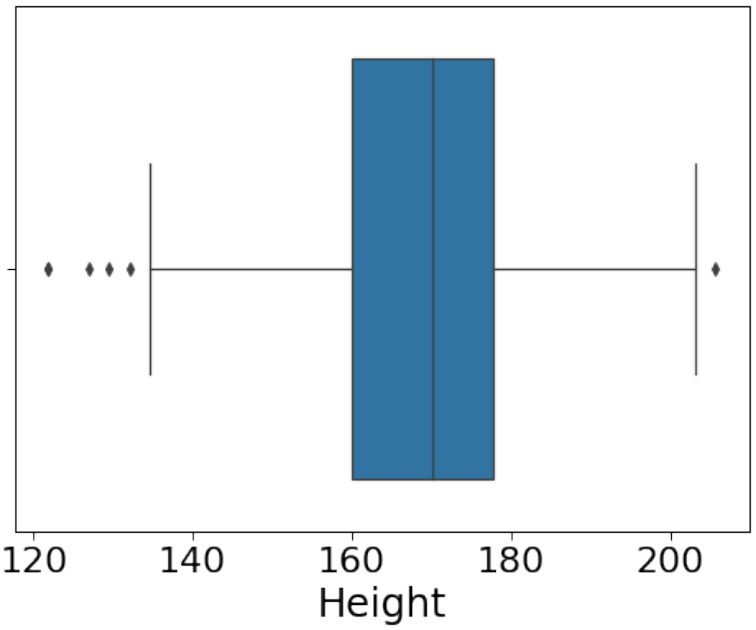


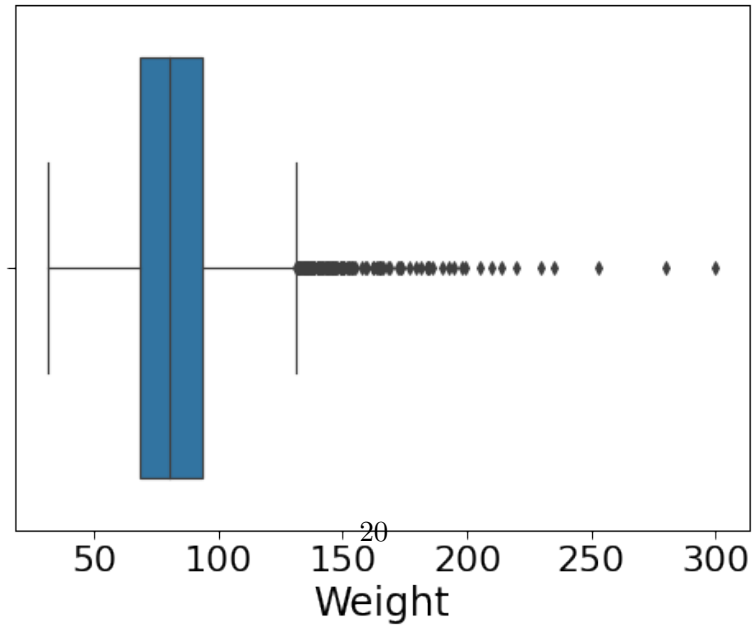
Jak widać niemal w każdej w każdej zmiennej występują outliery (zarówno po prawej jak i lewej stronie rozkładu). Trudno jednak określić, czy stanowią one błąd, czy są ważnymi informacjami w zbiorze danych. Jedynie wśród kilku cech, takich jak **Temperatura** wartości bliskie zeru możemy z dużą dozą prawdopodobieństwa zakwalifikować jako błędne.

[242]:
```
#boxploty zmiennych statycznych
cols = list(data.columns[0:5])
cols.remove("Gender")
cols.remove("ICUType")

fig, axes = plt.subplots(3,1, figsize = (8, 20))

for col, ax in zip(cols, axes.flatten()):
```

```
    sns.boxplot(x=col, data=data, orient="h", ax=ax)
```


20
40
60
80
Age

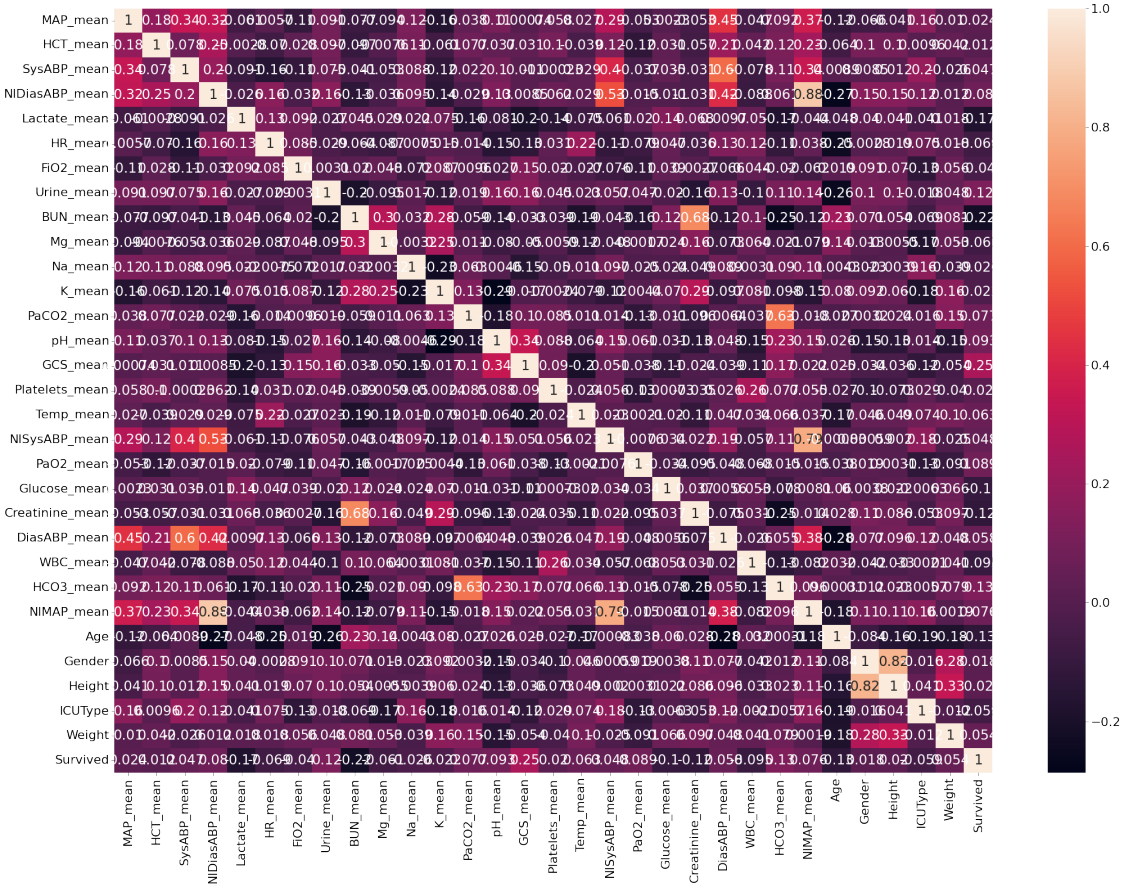

120
140
160
180
200
Height


50
100
150
200
250
300
Weight

Wśród danych statycznych trudno zaklasyfikować którekolwiek wartości jako nieprawidłowe, mimo że wynikają one z niektórych boxplotów. Niestety wysoka waga może okazać się bardzo ważną informacją w kontekście chorób sercowych.

1.6 Korelacje między zmiennymi

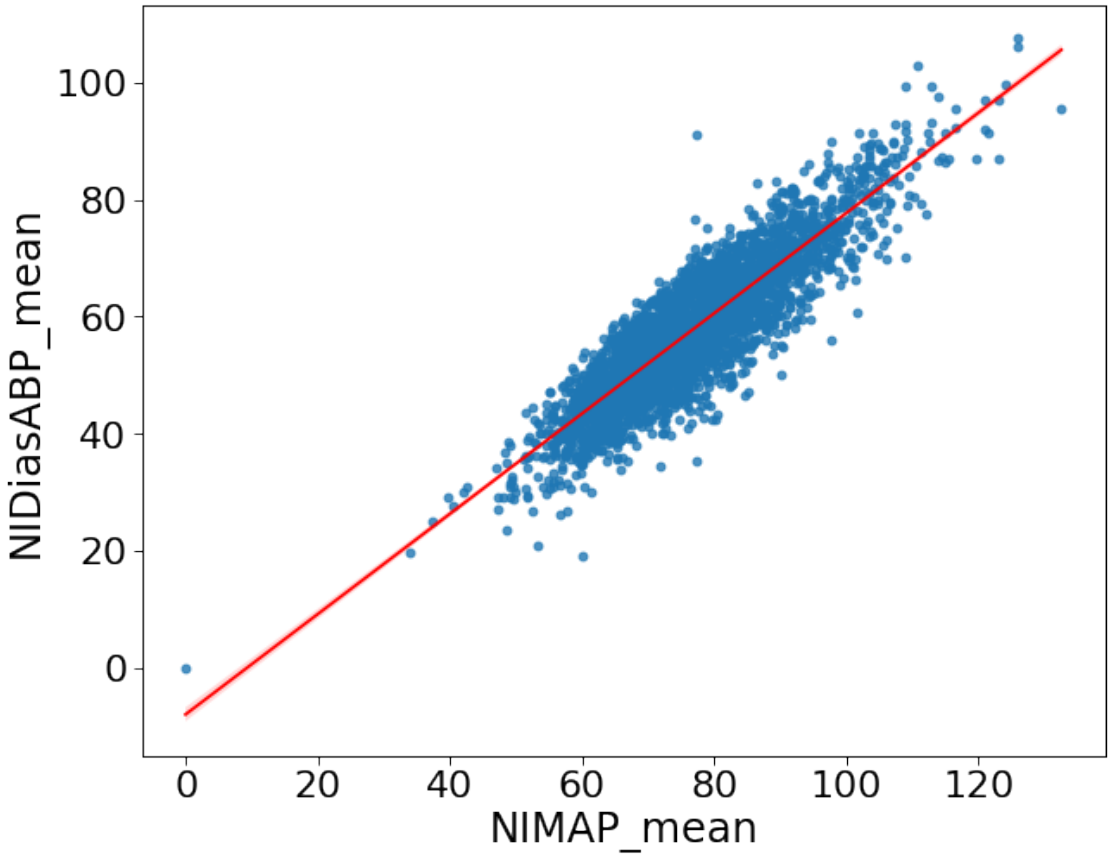
[243]:
```
corr_cols = [w for w in data.columns if w.endswith('mean')]
static_cols =  ['Age', 'Gender', 'Height', 'ICUType', 'Weight']
y = ["Survived"]
corr_cols = corr_cols + static_cols + y
plt.figure(figsize=(35,25))
sns.heatmap(data[corr_cols].corr(), annot=True)
plt.show()
```



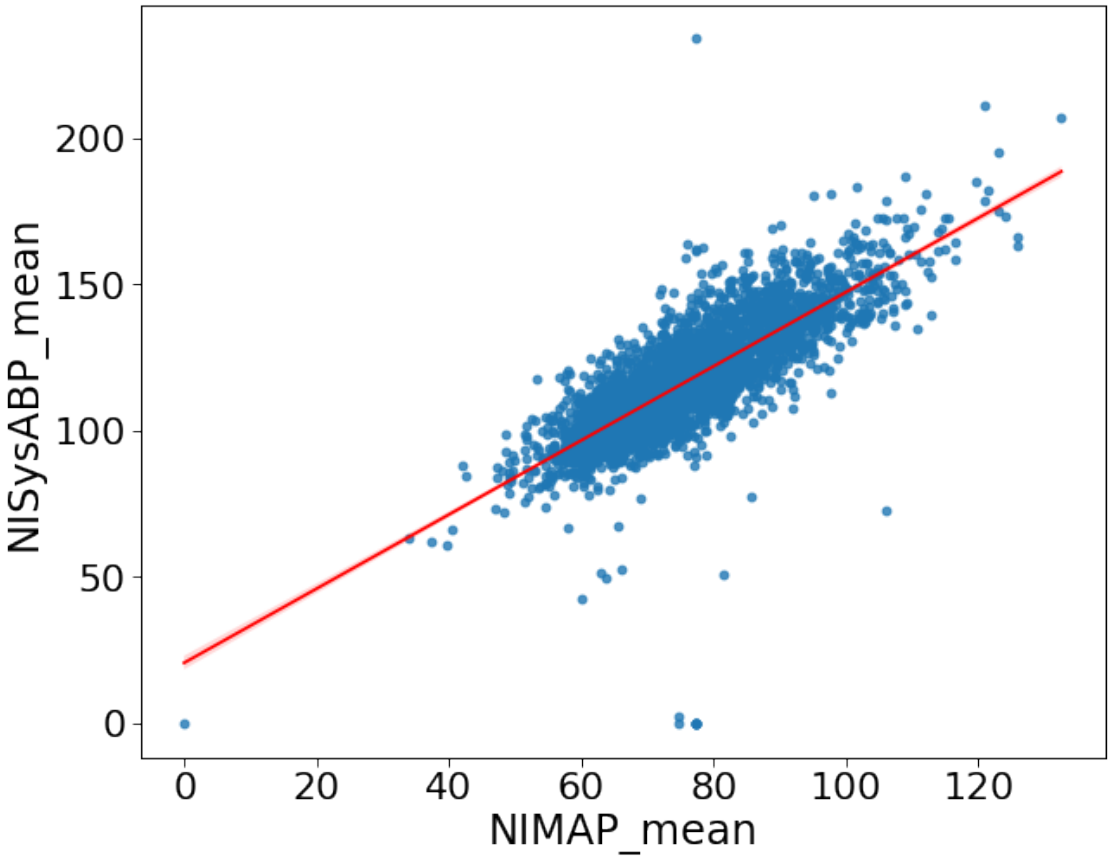
Widać wyraźną dodatnią korelację pomiędzy zmiennymi `NIMAP` - nieinwazyjne średnie ciśnienie tętnicze krwi i `NIDiasABP` - nieinwazyjne rozkurczowe ciśnienie tętnicze krwi (logiczne, jedna wartość jest statystyką drugiej). Ponad to warto odnotować korelację pomiędzy zmiennymi `NIMAP` i `NISysABP` - nieinwazyjne skurczowe ciśnienie tętnicze krwi (jak wyżej). Poza tym widać logiczne

korelacje jak między `Height` i `Gender`. Pozostałe korelacje są w granicach normy. Zmienna celu `Survived` nie jest wysoko skorelowana z którąkolwiek inną zmienną.

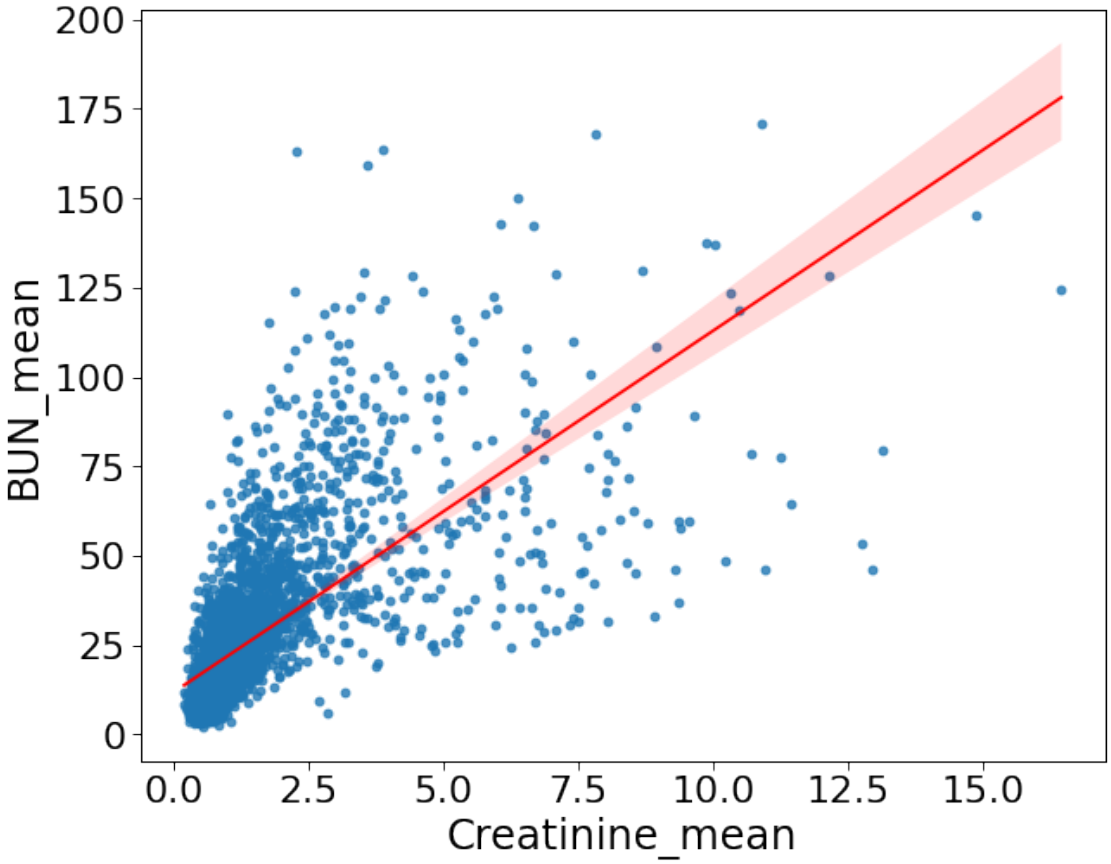
[244]:
```
#NIMAP ~ NIDiasABP
plt.figure(figsize=(10,8))
sns.regplot(data=data, x = "NIMAP_mean", y = "NIDiasABP_mean",␣
 ↪line_kws={"color": "red"})
plt.show()
```



[245]:
```
#NIMAP ~ NISysABP
plt.figure(figsize=(10,8))
sns.regplot(data=data, x = "NIMAP_mean", y = "NISysABP_mean", line_kws={"color":
 ↪ "red"})
plt.show()
```



[246]:
```
#Creatinine ~ BUN
plt.figure(figsize=(10,8))
sns.regplot(data=data, x = "Creatinine_mean", y = "BUN_mean", line_kws={"color":
 ↪ "red"})
plt.show()
```

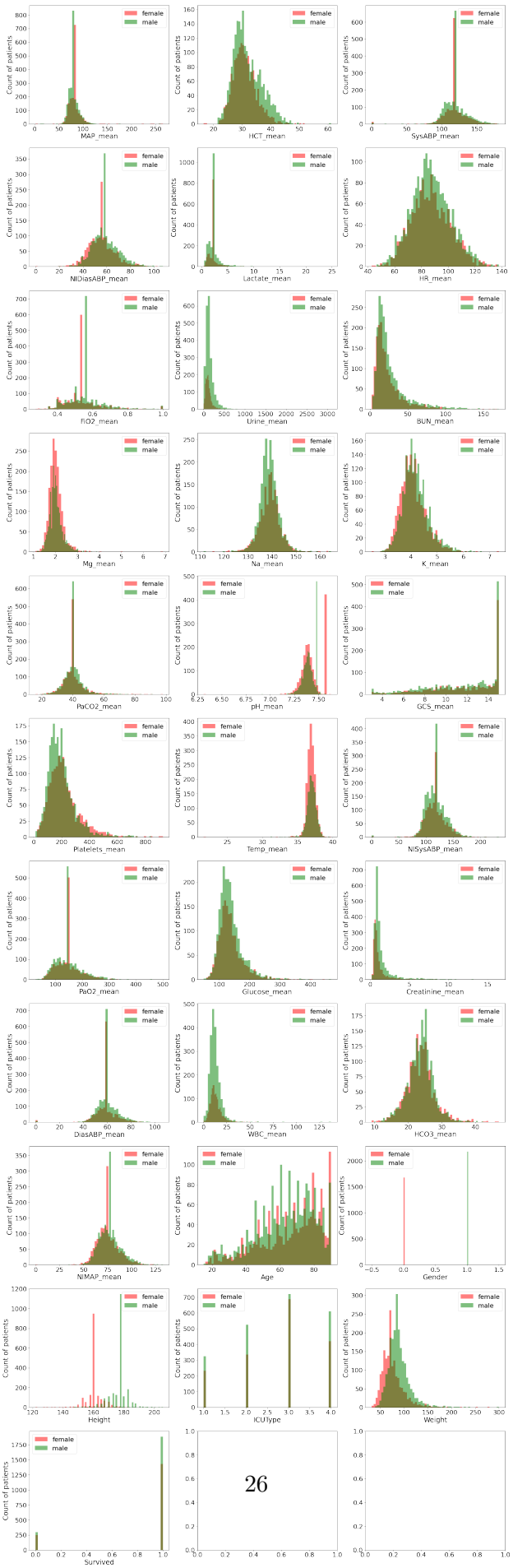


1.7 Rozkłady zmiennych zgrupowanych

[247]:
```
# funkcja rysująca histogramy zmiennych cols pogrupowanych wg zmiennej␣
↪grouping_col
def grouped_hist(grouping_col, cols, labels):
  col_names = cols
  n = len(col_names)
  i = 0
  j = 0
  plt.rcParams['font.size'] = 10
  if n % 3 != 0:
    k = n//3 + 1
  else:
    k = n//3
  fig, ax = plt.subplots(k, 3, figsize=(10*3, 10*(n//3)))

  for col in col_names:
    ax[i,j].hist(data[data[grouping_col]==0][col], bins=60, alpha=0.5,␣
↪color="red", label=labels[0])
```

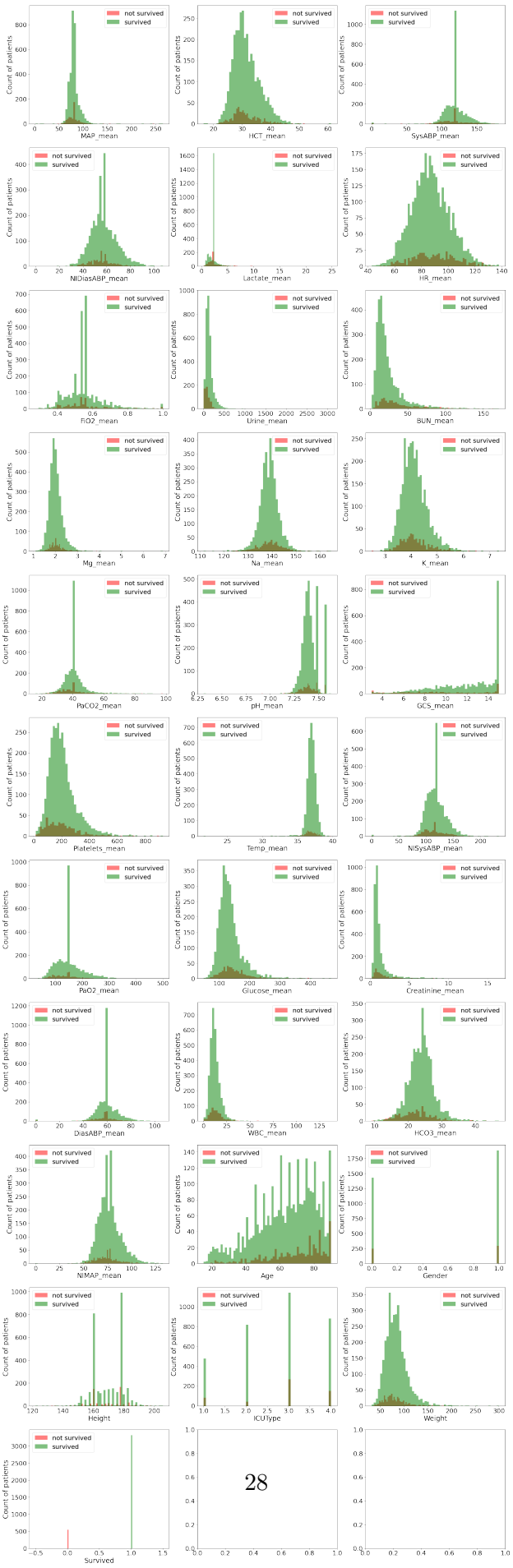
```
    ax[i,j].hist(data[data[grouping_col]==1][col], bins=60, alpha=0.5,␣
↪color="green", label=labels[1])
    ax[i,j].set_xlabel(col)
    ax[i,j].set_ylabel("Count of patients")
    ax[i,j].legend()
    j += 1
    if j == 3:
        i += 1
        j = 0

  plt.show()
```

[248]:
```
# histogramy zmiennych pogrupowanych według płci
grouped_hist("Gender", corr_cols, ["female", 'male'])
```


female
male
Count of patients
MAP_mean
HCT_mean
SysABP_mean
NIDiasABP_mean
Lactate_mean
HR_mean
FiO2_mean
Urine_mean
BUN_mean
Mg_mean
Na_mean
K_mean
PaCO2_mean
pH_mean
GCS_mean
Platelets_mean
Temp_mean
NISysABP_mean
PaO2_mean
Glucose_mean
Creatinine_mean
DiasABP_mean
WBC_mean
HCO3_mean
NIMAP_mean
Age
Gender
Height
ICUType
Weight
Survived

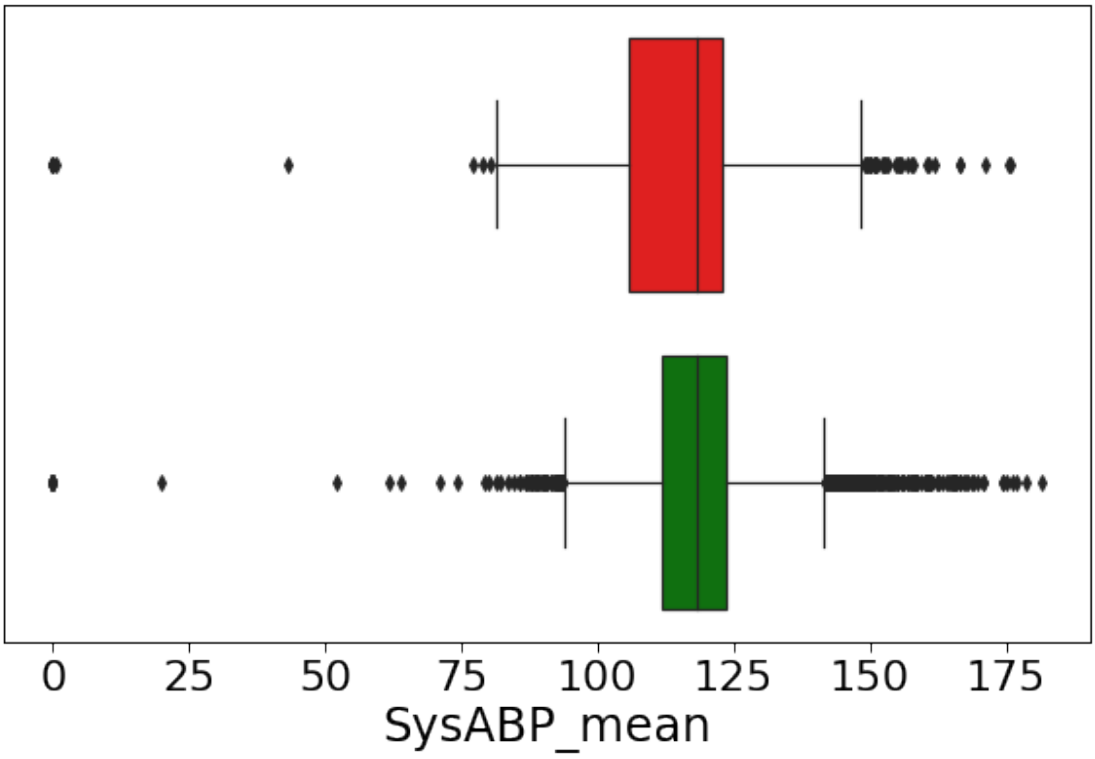
Z powyższych histogramów wynika, że jedynie dla wzrostu i wagi występują widoczne różnice w rozkładnie (zgadzają się one zresztą z naszą intuicją), które polegają na przesunięciu rozkładu w kierunku większych wartości dla mężczyzn, lecz kształ rozkładu zostaje zachowany.

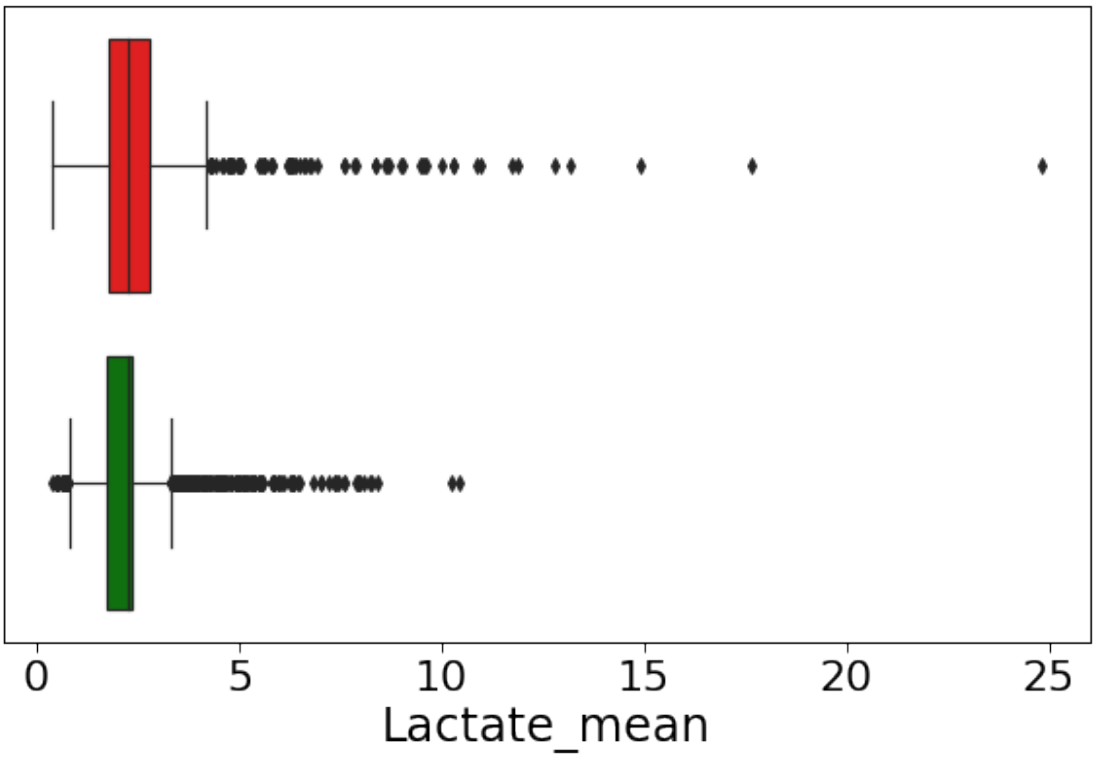
[249]:
```
# histogramy zmiennych pogrupowanych węług zmiennej celu
grouped_hist("Survived", corr_cols, ["not survived", "survived"])
```

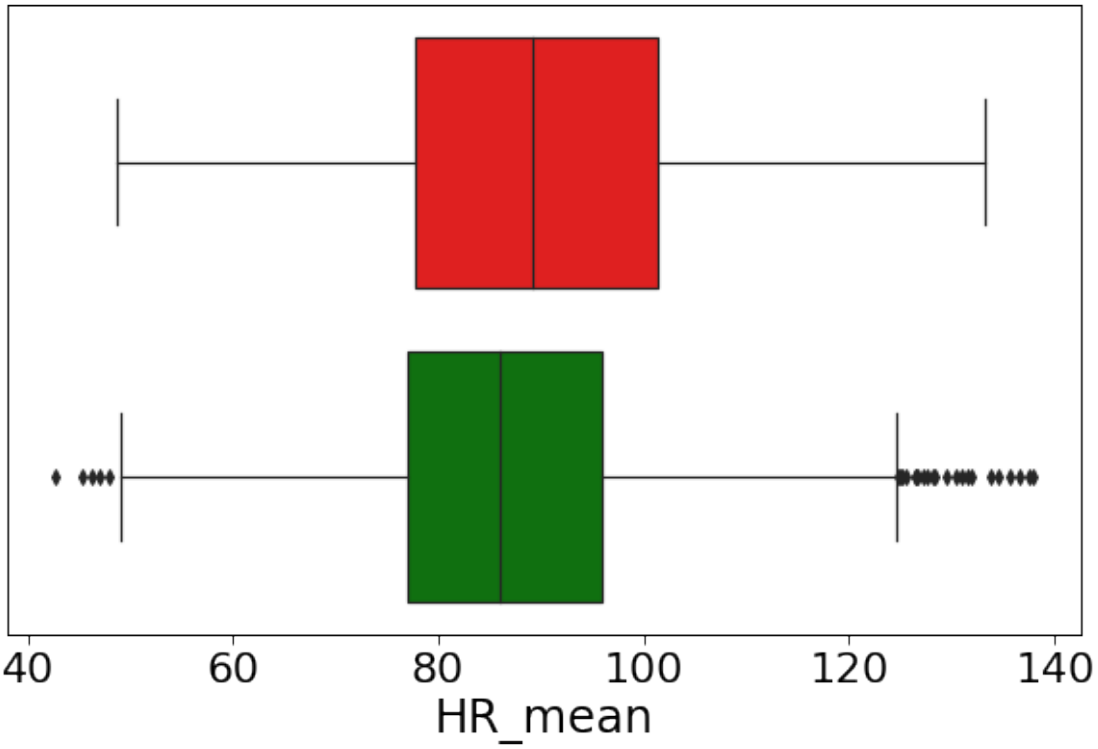

not survived
survived
Count of patients
MAP_mean
HCT_mean
SysABP_mean
NIDiasABP_mean
Lactate_mean
HR_mean
FiO2_mean
Urine_mean
BUN_mean
Mg_mean
Na_mean
K_mean
PaCO2_mean
pH_mean
GCS_mean
Platelets_mean
Temp_mean
NISysABP_mean
PaO2_mean
Glucose_mean
Creatinine_mean
DiasABP_mean
WBC_mean
HCO3_mean
NIMAP_mean
Age
Gender
Height
ICUType
Weight
Survived

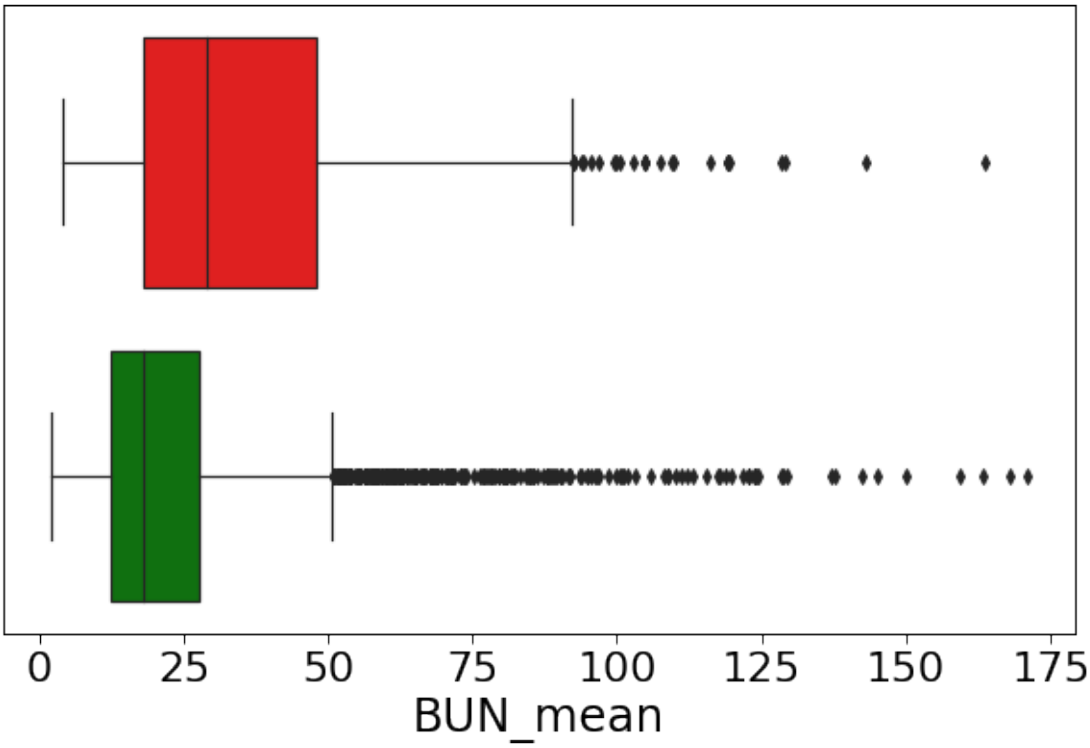
Nie widać znaczących różnic.

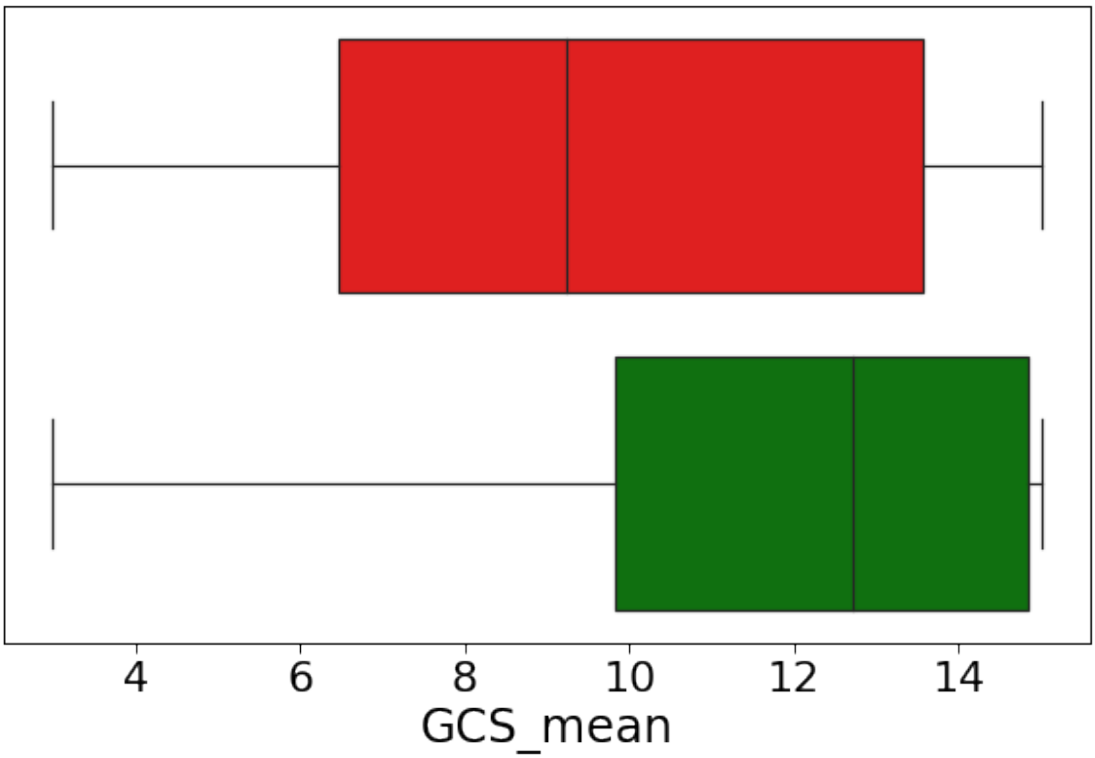
[252]:
```
df = data
col_names = ['SysABP_mean', 'Lactate_mean', 'HR_mean', 'BUN_mean', 'GCS_mean',␣
 ↪'Glucose_mean', 'HCO3_mean', 'Age']
plt.rcParams['figure.figsize'] = (10,6)

for col in col_names:
    fig, ax = plt.subplots()
    sns.boxplot(x=col, y="Survived", data=df, orient="h", palette={0:"red", 1:
 ↪"green"}, ax=ax)
    ax.get_yaxis().set_visible(False)

    plt.show()
```

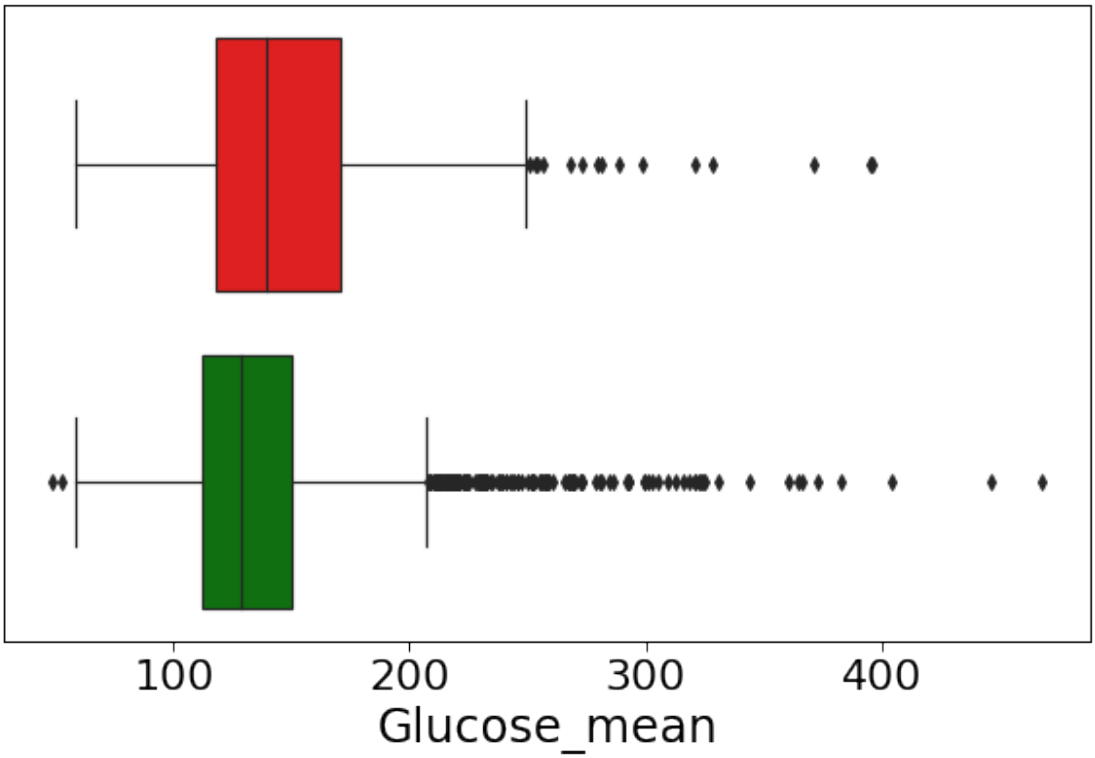


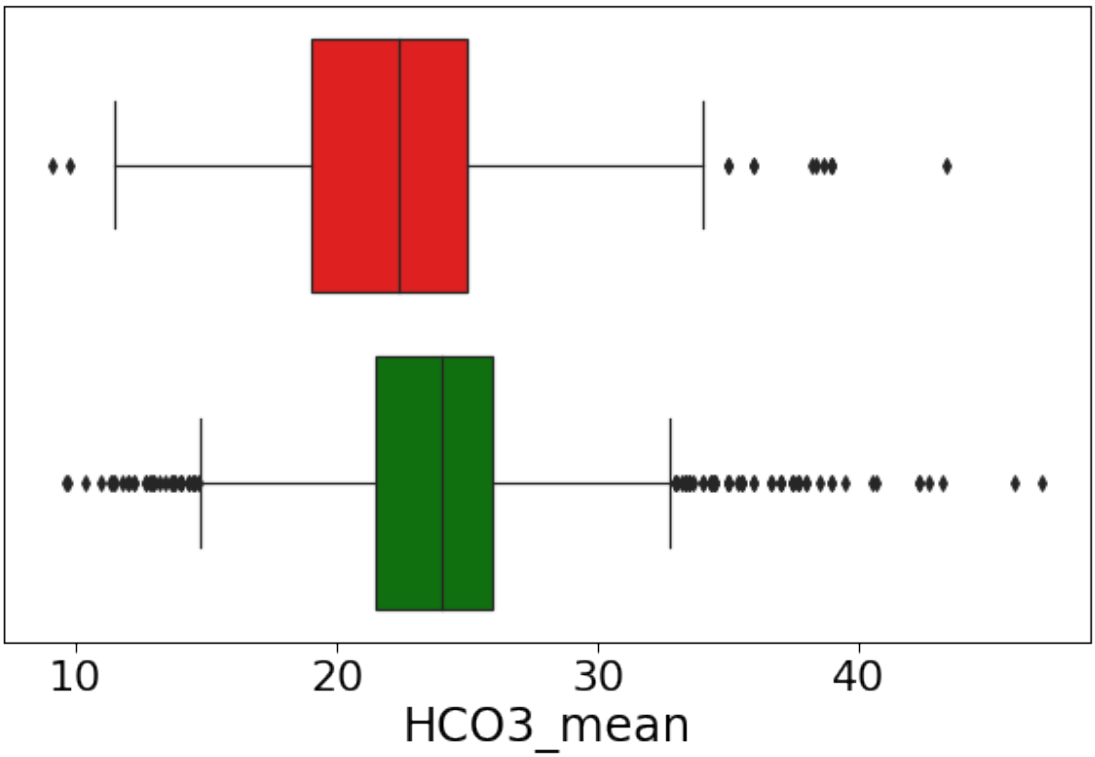

0
5
10
15
20
25
Lactate_mean

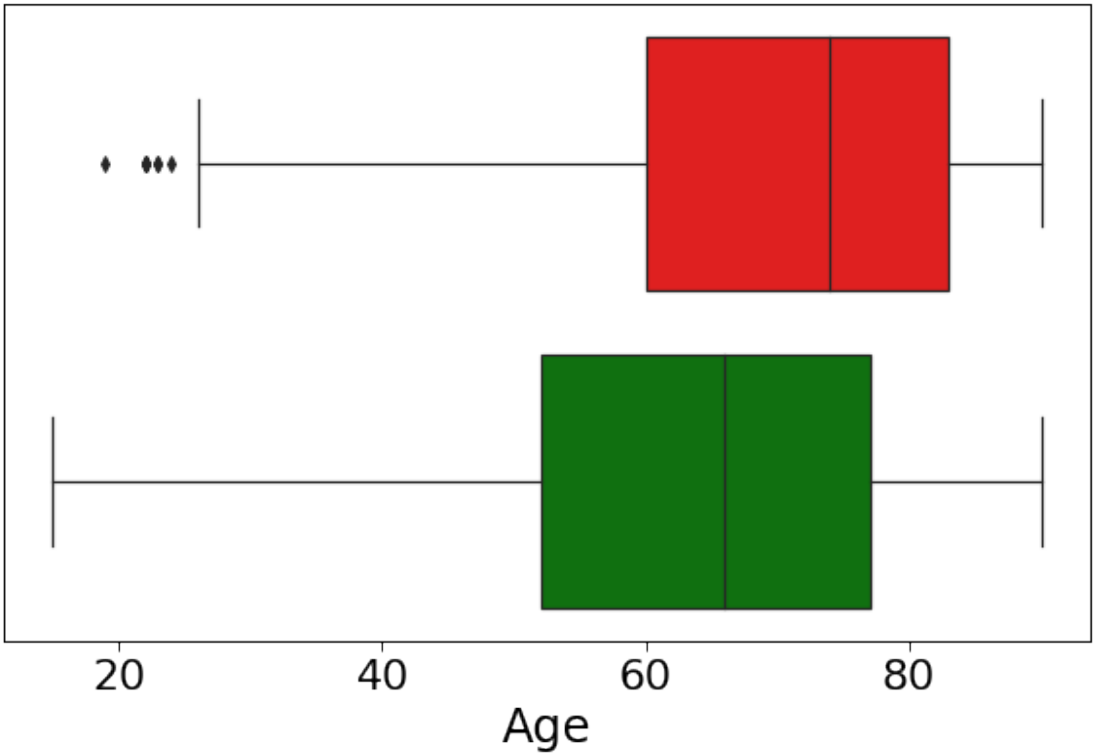

40
60
80
100
120
140
HR_mean


0
25
50
75
100
125
150
175
BUN_mean


4
6
8
10
12
14
GCS_mean


100
200
300
400
Glucose_mean


10
20
30
40
HCO3_mean

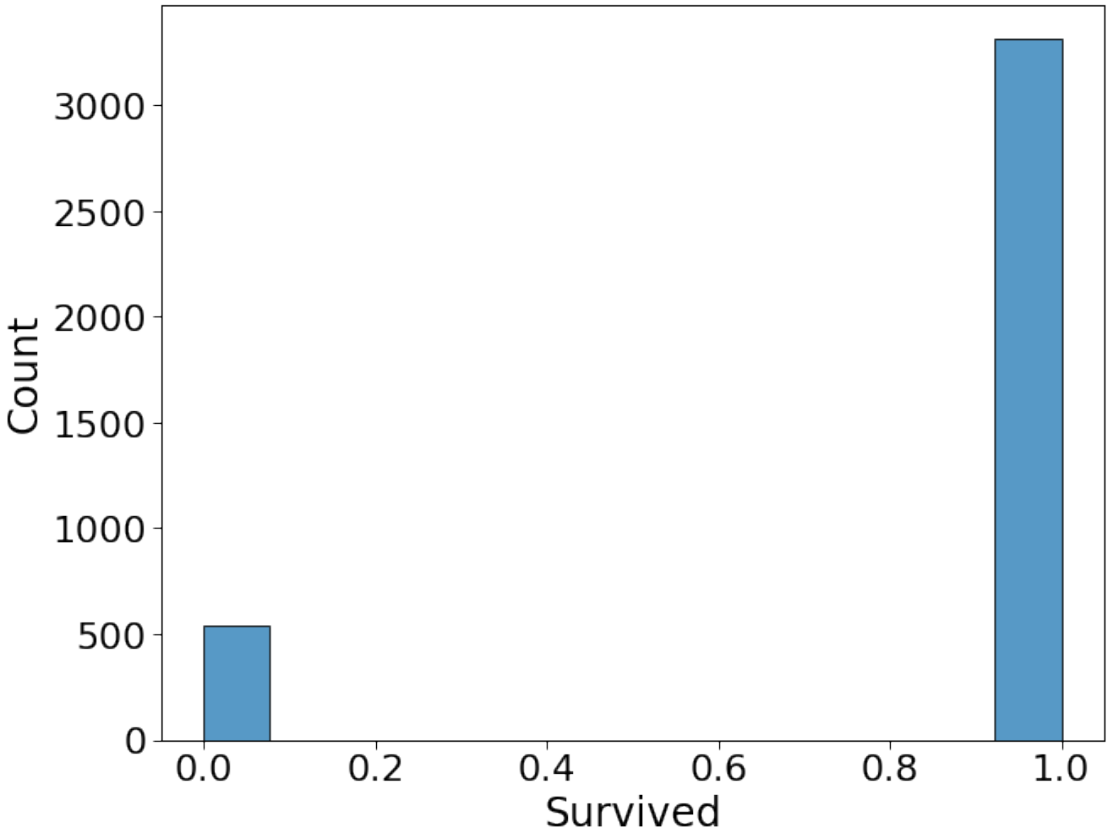

20
40
60
80
Age

Postanowiliśmy też przyjrzeć się boxplotom pogrupowanym względem zmiennej celu. Kolor zielony charakteryzuje pacjenta, który przeżył, zaś czerwony tego, który zmarł. Zwizualizowaliśmy tylko najbardziej interesujące przypadki. Można zaobserwować tu pewne ciekawe zależności, np. Tętno {HR) pacjentów, którzy przeżyli, było statystycznie niższe niż tych, którzy zmarli. Podobnie sytuacja wygląda z azotem mocznikowym we krwi (BUN). Z kolei statystyczna wartość GCS (skala Glasgow - jest używana w medycynie w celu oceny poziomu przytomności) dla pacjentów, którzy nie przeżyli jest zdecydowanie niższa, podobnie jak poziom wodorowęglanu w surowicy (HCO3). Widać też pewne bardzo intuicyjne zjawiska - częściej umierali pacjenci starci.

1.8 Analiza zmiennej celu

Naszą zmienną celu jest Survived. Przyjmuje ona wartości 0, gdy pacjent zmarł i 1, gdy przeżył.

[251]:
```
# Rozkład zmiennej celu
plt.figure(figsize=(10,8))
sns.histplot(data['Survived'])
plt.show()
```



Zdecydowanie nie ma równiej dystrybucji. Fakt ten będzie użyteczny przy wyborze metryk na późniejszym etapie pracy.

1.9 Wnioski

Zbiór zawiera dane medyczne - samo to jest utrudnieniem, gdyż, nie posiadając specjalistycznej wiedzy, trudno jest go interpretować. Po zagregowaniu danych z szeregów czasowych do postaci, która umożliwia wykonanie standardowego modelowania, w danynych ukazało się dużo braków (nie każde pomiary były wykonywane dla każdego pacjenta). Część z nich usunęliśmy, część staraliśmy się zaimputować. Dodatkowo zlokalizowaliśmy błędy w danych (4-metrowy wzrost), które też zostały usunięte. Dane zawierają też dużo outlierów, jednak z racji ich medycznego charakteru, postanowiliśmy ich nie usuwać (dane medyczne dla różnych pacjentów są często bardzo zróżnicowane). Między cechami nie występuje żadna niespodziewana korelacja. Zależności występujące są logiczne i zgodne z intuicją (płeć ~ wzrost czy średnie ciśnienie krwi ~ rozkurczowe ciśnienie krwi). Postanowiliśmy też przyjrzeć się danym pogrupowanym względem płci (jest ona ważnym czynnikiem przy chorobach kardiologicznych) oraz względem samej zmiennej celu. Grupowania te nie pokazały nam niestety żadnych większych zależności z wyjątkiem boxplotów niektórych zmiennych podzielonych względem zmiennej celu. Sama zmienna celu jest mocno niezbalansowana.