sql

November 19, 2021

0.1 Retrieve Data

[ ]:
```
from config import *
import pandas as pd
import mysql.connector

mydb = mysql.connector.connect(
  host="127.0.0.1",
  user=mysql_username,
  password=mysql_password,
  database="bestofu",
)


# u.id, u.name, u.gender, o.data, s.time_in_bed, s.time_asleep, s.
 ↪wakeup_number, s.sleep_score, s.stress_score
```

[ ]:
```
mycursor = mydb.cursor()

sql_1 = """
SELECT id, name, gender, data, time_in_bed, time_asleep, wakeup_number,␣
 ↪sleep_score, stress_score
FROM (
        SELECT id
                ,name
                ,gender
        FROM users
        ) u
JOIN (
        SELECT user_id
                ,data
        FROM daily_overall_data
        ) o ON o.user_id = u.id
JOIN (
        SELECT user_id
                ,date
                ,time_in_bed
                ,time_asleep
```

```
                ,wakeup_number
                ,sleep_score
                ,stress_score
        FROM daily_statistics
        ) s ON s.user_id = u.id
WHERE s.date BETWEEN '2021-11-01'
                AND '2021-11-30'
ORDER BY s.date
"""

col_name = ["u:id", "u:name", "u:gender", "data", "y:time_in_bed",
            "y:time_asleep", "y:wakeup_number", "y:sleep_score", "y:
 ↪stress_score"]

df = []
mycursor.execute(sql_1)
myresult = mycursor.fetchall()

for x in myresult:
  df.append(x)

df = pd.DataFrame(df, columns=col_name)
```

[ ]:
```
set_f = set()
for i in df.data:
    try:
        i_obj = eval(i)
    except NameError as ner:
        i_obj = eval(i.replace("true", "1").replace("false", "0"))

    for j in i_obj:
        if j not in set_f:
            set_f.add(j)

set_f
```

[ ]:
```
{'VO2_max',
 'blood_diastolic_presure',
 'blood_glucose',
 'blood_systolic_presure',
 'body_temperature',
 'exercise_time',
 'heart_beat_series',
 'hight_hr_events',
 'hr_variability_SDNN',
 'low_hr_events',
```

```
 'oxygen_saturation',
 'recent_body_mass_index',
 'resespiratory_rate',
 'resting_hr',
 'walking_hr_verage'}
```

[ ]:
```
mycursor = mydb.cursor()

sql_1 = """
SELECT u.*, o.*, s.*
FROM (
        SELECT id
                ,name
                ,gender
        FROM users
        ) u
JOIN (
        SELECT user_id
                ,data->'$.VO2_max' as VO2_max
                ,data->'$.blood_diastolic_presure' as blood_diastolic_presure
                ,data->'$.blood_glucose' as blood_glucose
                ,data->'$.blood_systolic_presure' as blood_systolic_presure
                ,data->'$.body_temperature' as body_temperature
                ,data->'$.exercise_time' as exercise_time
                ,data->'$.heart_beat_series' as heart_beat_series
                ,data->'$.hight_hr_events' as hight_hr_events
                ,data->'$.hr_variability_SDNN' as hr_variability_SDNN
                ,data->'$.low_hr_events' as low_hr_events
                ,data->'$.oxygen_saturation' as oxygen_saturation
                ,data->'$.recent_body_mass_index' as recent_body_mass_index
                ,data->'$.resespiratory_rate' as resespiratory_rate
                ,data->'$.resting_hr' as resting_hr
                ,data->'$.walking_hr_verage' as walking_hr_verage
        FROM daily_overall_data
        ) o ON o.user_id = u.id
JOIN (
        SELECT user_id
                ,date
                ,time_in_bed
                ,time_asleep
                ,wakeup_number
                ,sleep_score
                ,stress_score
        FROM daily_statistics
        ) s ON s.user_id = u.id
WHERE s.date BETWEEN '2021-09-01'
                AND '2021-11-30'
```

```
ORDER BY s.date
"""

col_name = ["u:id", "u:name", "u:gender", "o.user_id",
                        'd:VO2_max',
                        'd:blood_diastolic_presure',
                        'd:blood_glucose',
                        'd:blood_systolic_presure',
                        'd:body_temperature',
                        'd:exercise_time',
                        'd:heart_beat_series',
                        'd:hight_hr_events',
                        'd:hr_variability_SDNN',
                        'd:low_hr_events',
                        'd:oxygen_saturation',
                        'd:recent_body_mass_index',
                        'd:resespiratory_rate',
                        'd:resting_hr',
                        'd:walking_hr_verage',  "s.user_id", "u:date",
            "y:time_in_bed", "y:time_asleep", "y:wakeup_number", "y:
 ↪sleep_score", "y:stress_score"]

df = []
mycursor.execute(sql_1)
myresult = mycursor.fetchall()

for x in myresult:
  df.append(x)

df = pd.DataFrame(df, columns=col_name)
```

0.2 Data Profile

[ ]:
```
import os
import sys
import numpy as np
import pandas as pd
import altair as alt
sys.path.append("/Users/tuanzai/Desktop/Git/tools/tools")

from data_overview import *
from df_to_highchart import *
```

[ ]:
```
d_list = [x for x in df if x.startswith("d:")]
df[d_list] = df[d_list].replace("", np.nan).replace("true", 1).replace("false",␣
 ↪0).astype(float)
```

```
num_edd = edd(df[d_list], missing_value=np.nan, ignore_col=[], save_path=None)
display(num_edd)
print("\r\r\r==================== missing rate < 0.9 ====================")
display(num_edd[num_edd["missingrate"] <= 0.9])
```

```
====================== time cost: 0:00:00.119227 ====================
```

	var	type	sample_cnt	unique_values	missingrate	\
0	d:VO2_max	float64	171691	76	0.918918	
1	d:blood_diastolic_presure	float64	171691	2	0.979801	
2	d:blood_glucose	float64	171691	1	0.9899	
3	d:blood_systolic_presure	float64	171691	2	0.979801	
4	d:body_temperature	float64	171691	2	0.979801	
5	d:exercise_time	float64	171691	1	0.404989	
6	d:heart_beat_series	float64	171691	83	0.345965	
7	d:hight_hr_events	float64	171691	1	0.997909	
8	d:hr_variability_SDNN	float64	171691	1006	0.194588	
9	d:low_hr_events	float64	171691	1	0.998288	
10	d:oxygen_saturation	float64	171691	16	0.843323	
11	d:recent_body_mass_index	float64	171691	25	0.97114	
12	d:resespiratory_rate	float64	171691	14	0.987349	
13	d:resting_hr	float64	171691	58	0.208735	
14	d:walking_hr_verage	float64	171691	146	0.275565	

	zerorate	mode	mean	std	min	qt1	qt5	\
0	0.0	NaN	44.511407	5.610902	27.0	29.18	33.12	
1	0.0	NaN	126.5	13.501947	113.0	113.0	113.0	
2	0.0	NaN	36.031176	0.0	36.031176	36.031176	36.031176	
3	0.0	NaN	151.0	39.005624	112.0	112.0	112.0	
4	0.0	NaN	37.6	0.400058	37.2	37.2	37.2	
5	0.0	NaN	1.0	0.0	1.0	1.0	1.0	
6	0.0	NaN	47.345323	13.590376	10.0	10.0	23.0	
7	0.0	NaN	1.0	0.0	1.0	1.0	1.0	
8	0.0	NaN	44.606873	20.966603	6.58597	14.39401	20.879116	
9	0.0	NaN	1.0	0.0	1.0	1.0	1.0	
10	0.0	NaN	0.942086	0.034619	0.85	0.86	0.88	
11	0.0	NaN	21.935246	2.913112	18.069728	18.069728	18.069728	
12	0.0	NaN	18.887431	2.943169	11.5	14.0	14.0	
13	0.0	NaN	62.393749	11.81503	45.0	49.0	52.0	
14	0.0	NaN	101.623566	15.495646	64.0	70.0	77.5	

	qt25	qt50	qt75	qt95	qt99	max
0	42.59	45.04	45.85	55.0	55.0	55.0
1	113.0	126.5	140.0	140.0	140.0	140.0
2	36.031176	36.031176	36.031176	36.031176	36.031176	36.031176
3	112.0	151.0	190.0	190.0	190.0	190.0

```
4        37.2       37.6       38.0       38.0        38.0        38.0
5         1.0        1.0        1.0        1.0         1.0         1.0
6        38.0       50.0       57.0       64.0        77.0       117.0
7         1.0        1.0        1.0        1.0         1.0         1.0
8   29.937435  40.438225  54.056683  78.376495  132.896576  220.633392
9         1.0        1.0        1.0        1.0         1.0         1.0
10       0.92       0.94       0.97       0.99         1.0         1.0
11  18.968052  21.579205  23.794317      26.37   26.407541   26.407541
12       17.0       18.5       21.0       22.5        28.0        28.0
13       57.0       60.0       63.0       83.0       119.0       145.0
14       91.0       99.5      113.0      128.0       138.0       164.0

==================== missing rate < 0.9 ====================

                     var     type  sample_cnt unique_values missingrate  \
5         d:exercise_time  float64      171691             1    0.404989
6     d:heart_beat_series  float64      171691            83    0.345965
8   d:hr_variability_SDNN  float64      171691          1006    0.194588
10    d:oxygen_saturation  float64      171691            16    0.843323
13           d:resting_hr  float64      171691            58    0.208735
14    d:walking_hr_verage  float64      171691           146    0.275565

   zerorate mode        mean        std      min       qt1        qt5  \
5       0.0  NaN         1.0        0.0      1.0       1.0        1.0
6       0.0  NaN   47.345323  13.590376     10.0      10.0       23.0
8       0.0  NaN   44.606873  20.966603  6.58597  14.39401  20.879116
10      0.0  NaN    0.942086   0.034619     0.85      0.86       0.88
13      0.0  NaN   62.393749   11.81503     45.0      49.0       52.0
14      0.0  NaN  101.623566  15.495646     64.0      70.0       77.5

         qt25       qt50       qt75       qt95        qt99         max
5         1.0        1.0        1.0        1.0         1.0         1.0
6        38.0       50.0       57.0       64.0        77.0       117.0
8   29.937435  40.438225  54.056683  78.376495  132.896576  220.633392
10       0.92       0.94       0.97       0.99         1.0         1.0
13       57.0       60.0       63.0       83.0       119.0       145.0
14       91.0       99.5      113.0      128.0       138.0       164.0
```

[ ]:
```
dlist_useful = num_edd[num_edd["missingrate"] <= 0.9]["var"].to_list()
display(dlist_useful)

num_edd_plot = edd(df[dlist_useful], missing_value=np.nan, ignore_col=[],␣
 ↪save_path=None)
num_edd_plot
```

```
['d:exercise_time',
 'd:heart_beat_series',
 'd:hr_variability_SDNN',
 'd:oxygen_saturation',
```

```
 'd:resting_hr',
 'd:walking_hr_verage']

====================== time cost: 0:00:00.140256 ====================
```

[ ]:

	var	type	sample_cnt	unique_values	missingrate	zerorate	mode	mean	std	min	qt1	qt5	qt25	qt50	qt75	qt95	qt99	max
0	d:exercise_time	float64	171691	1	0.404989	0.0	NaN	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	d:heart_beat_series	float64	171691	83	0.345965	0.0	NaN	47.345323	13.590376	10.0	10.0	23.0	38.0	50.0	57.0	64.0	77.0	117.0
2	d:hr_variability_SDNN	float64	171691	1006	0.194588	0.0	NaN	44.606873	20.966603	6.58597	14.39401	20.879116	29.937435	40.438225	54.056683	78.376495	132.896576	220.633392
3	d:oxygen_saturation	float64	171691	16	0.843323	0.0	NaN	0.942086	0.034619	0.85	0.86	0.88	0.92	0.94	0.97	0.99	1.0	1.0
4	d:resting_hr	float64	171691	58	0.208735	0.0	NaN	62.393749	11.81503	45.0	49.0	52.0	57.0	60.0	63.0	83.0	119.0	145.0
5	d:walking_hr_verage	float64	171691	146	0.275565	0.0	NaN	101.623566	15.495646	64.0	70.0	77.5	91.0	99.5	113.0	128.0	138.0	164.0

[ ]:
```
columns = ["missingrate", 'mean', 'std', 'min', 'qt1', 'qt5', 'qt25', 'qt50',␣
↪'qt75', 'qt95', 'qt99', 'max', "sample_cnt"]
```

[ ]:
```
num_edd_plot[columns]
```

[ ]:

	missingrate	mean	std	min	qt1	qt5	qt25	qt50	qt75	qt95	qt99	max	sample_cnt
0	0.404989	1.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	171691
1	0.345965	47.345323	13.590376	10.0	10.0	23.0	38.0	50.0	57.0	64.0	77.0	117.0	171691
2	0.194588	44.606873	20.966603	6.58597	14.39401	20.879116	29.937435	40.438225	54.056683	78.376495	132.896576	220.633392	171691
3	0.843323	0.942086	0.034619	0.85	0.86	0.88	0.92	0.94	0.97	0.99	1.0	1.0	171691
4	0.208735	62.393749	11.81503	45.0	49.0	52.0	57.0	60.0	63.0	83.0	119.0	145.0	171691
5	0.275565	101.623566	15.495646	64.0	70.0	77.5	91.0	99.5	113.0	128.0	138.0	164.0	171691

3	0.94	0.97	0.99	1.0	1.0	171691
4	60.0	63.0	83.0	119.0	145.0	171691
5	99.5	113.0	128.0	138.0	164.0	171691

[ ]:
```
# num_edd_plot = edd(df[dlist_useful].fillna(-1), missing_value=np.nan,␣
↪ignore_col=[], save_path=None)
edd1 = lambda df: edd(df[dlist_useful+[x for x in df if x.startswith("y:")]])
stat_by_date = df[dlist_useful + ["u:date"] + [x for x in df if x.startswith("y:
↪")]].groupby("u:date").apply(edd1)
```

```
====================== time cost: 0:00:00.040932 ======================

====================== time cost: 0:00:00.036135 ======================

====================== time cost: 0:00:00.033289 ======================

====================== time cost: 0:00:00.032040 ======================

====================== time cost: 0:00:00.035283 ======================

====================== time cost: 0:00:00.031200 ======================

====================== time cost: 0:00:00.030274 ======================

====================== time cost: 0:00:00.031897 ======================

====================== time cost: 0:00:00.031490 ======================

====================== time cost: 0:00:00.029812 ======================

====================== time cost: 0:00:00.030034 ======================

====================== time cost: 0:00:00.030868 ======================

====================== time cost: 0:00:00.029769 ======================
```

```
====================== time cost: 0:00:00.029415 ======================

====================== time cost: 0:00:00.030963 ======================

====================== time cost: 0:00:00.032336 ======================

====================== time cost: 0:00:00.029806 ======================

====================== time cost: 0:00:00.030380 ======================

====================== time cost: 0:00:00.032961 ======================

====================== time cost: 0:00:00.029566 ======================

====================== time cost: 0:00:00.029375 ======================

====================== time cost: 0:00:00.043852 ======================

====================== time cost: 0:00:00.039087 ======================

====================== time cost: 0:00:00.040445 ======================

====================== time cost: 0:00:00.036719 ======================

====================== time cost: 0:00:00.039637 ======================

====================== time cost: 0:00:00.032733 ======================

====================== time cost: 0:00:00.034583 ======================

====================== time cost: 0:00:00.034273 ======================
```

```
====================== time cost: 0:00:00.034472 ======================

====================== time cost: 0:00:00.032418 ======================

====================== time cost: 0:00:00.030011 ======================

====================== time cost: 0:00:00.028679 ======================

====================== time cost: 0:00:00.029162 ======================

====================== time cost: 0:00:00.031018 ======================

====================== time cost: 0:00:00.030821 ======================

====================== time cost: 0:00:00.029259 ======================

====================== time cost: 0:00:00.039502 ======================

====================== time cost: 0:00:00.042399 ======================

====================== time cost: 0:00:00.045931 ======================

====================== time cost: 0:00:00.047520 ======================

====================== time cost: 0:00:00.032769 ======================

====================== time cost: 0:00:00.030026 ======================

====================== time cost: 0:00:00.034261 ======================

====================== time cost: 0:00:00.034466 ======================
```

```
====================== time cost: 0:00:00.038625 ======================

====================== time cost: 0:00:00.032446 ======================

====================== time cost: 0:00:00.032200 ======================

====================== time cost: 0:00:00.029296 ======================

====================== time cost: 0:00:00.028916 ======================

====================== time cost: 0:00:00.031080 ======================

====================== time cost: 0:00:00.029797 ======================

====================== time cost: 0:00:00.043324 ======================

====================== time cost: 0:00:00.034606 ======================

====================== time cost: 0:00:00.040209 ======================

====================== time cost: 0:00:00.029450 ======================

====================== time cost: 0:00:00.030281 ======================

====================== time cost: 0:00:00.029174 ======================

====================== time cost: 0:00:00.029550 ======================

====================== time cost: 0:00:00.028616 ======================

====================== time cost: 0:00:00.035429 ======================
```

```
====================== time cost: 0:00:00.033686 ======================

====================== time cost: 0:00:00.030262 ======================

====================== time cost: 0:00:00.030170 ======================

====================== time cost: 0:00:00.028864 ======================

====================== time cost: 0:00:00.031097 ======================

====================== time cost: 0:00:00.030653 ======================

====================== time cost: 0:00:00.031719 ======================

====================== time cost: 0:00:00.044009 ======================

====================== time cost: 0:00:00.042224 ======================

====================== time cost: 0:00:00.069086 ======================

====================== time cost: 0:00:00.066107 ======================

====================== time cost: 0:00:00.039509 ======================

====================== time cost: 0:00:00.030051 ======================

====================== time cost: 0:00:00.030568 ======================

====================== time cost: 0:00:00.028647 ======================

====================== time cost: 0:00:00.028317 ======================
```

```
====================== time cost: 0:00:00.028840 ====================

====================== time cost: 0:00:00.032335 ====================

====================== time cost: 0:00:00.029279 ====================
```

[ ]:
```
import altair as alt
import pandas as pd
alt.data_transformers.enable('data_server')
alt.renderers.enable('mimetype')


# url = ('https://raw.githubusercontent.com/UBC-MDS/exploratory-data-viz/main/'␣
 ↪+
#       'chapters/en/slides/module2/data/world-data-gapminder.csv')
# gm = pd.read_csv(url, parse_dates=['year'])
# gm



# alt.Chart(gm).mark_line().encode(
#     x='year',
#     y='sum(population)')
```

[ ]:
```
RendererRegistry.enable('mimetype')
```

0.2.1 stablity of Feature

[ ]:
```
def stat_distribution(source, title):

    print(title)

    bar = alt.Chart(source).mark_bar(opacity=0.45, color='#57A44C').encode(
        x = 'date:T',
        y = alt.Y("sample_cnt", axis = alt.Axis(title='Total Count',␣
 ↪titleColor='#57A44C')),
    ).properties(
        width = 600
    ).interactive()

    source = source[["date", "mean", "std", "min", "qt1", "qt25", "qt75",␣
 ↪"max", "missingrate"]].melt("date", var_name='category', value_name='y')
```

```
    selection = alt.selection_multi(fields=['category'], bind='legend')
    line = alt.Chart(source).mark_circle(interpolate='monotone').encode(
        x = 'date:T',
        y=alt.Y("y:Q", axis = alt.Axis(title='Percentage stats',␣
↪titleColor='#5276A7')),
        color=alt.Color('category:N'),
        # opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),
        tooltip="y:Q"
    )

    line = (line + line.mark_line()).properties(
        width = 600
    ).interactive()

    display(alt.layer(bar, line).resolve_scale(
        y = 'independent'
    ))
```

[ ]:
```
stat_by_date["missingrate"] = stat_by_date["missingrate"]*100
```

[ ]:
```
for i in dlist_useful:
    source = stat_by_date.loc[stat_by_date["var"] == i, columns].reset_index().
↪rename(columns={"u:date":"date"})
    stat_distribution(source, i)
```

```
d:exercise_time

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html


d:heart_beat_series

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html


d:hr_variability_SDNN

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
```

```
https://altair-viz.github.io/user_guide/troubleshooting.html


d:oxygen_saturation

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html


d:resting_hr

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html


d:walking_hr_verage

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html
```

[ ]:
```
for i in [x for x in df if x.startswith("y:")]:
    source = stat_by_date.loc[stat_by_date["var"] == i, columns].reset_index().
 ↪rename(columns={"u:date":"date"})
    stat_distribution(source, i)
```

```
y:time_in_bed

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html


y:time_asleep

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html
```

y:wakeup_number

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled for the frontend that you are using. For more information, see https://altair-viz.github.io/user_guide/troubleshooting.html

y:sleep_score

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled for the frontend that you are using. For more information, see https://altair-viz.github.io/user_guide/troubleshooting.html

y:stress_score

<VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled for the frontend that you are using. For more information, see https://altair-viz.github.io/user_guide/troubleshooting.html

0.2.2 feature and score

[ ]:
```
df_plots = df[dlist_useful+[x for x in df if x.startswith("y:")]]
```

[ ]:
```
corr_df = df_plots[[x for x in df_plots if x != "d:exercise_time"]].dropna().
 ↪corr()
display(corr_df)

source = corr_df.reset_index().melt("index")
alt.Chart(source).mark_square().encode(
    x='index:O',
    y='variable:O',
    color='value:Q',
    size='value:Q',
    tooltip='value:Q',
).properties(
    width = 400,
    height = 400
)
```

```
                      d:heart_beat_series  d:hr_variability_SDNN  \
d:heart_beat_series              1.000000              -0.006587
d:hr_variability_SDNN           -0.006587               1.000000
d:oxygen_saturation             -0.037140              -0.145225
```

```
d:resting_hr                      0.145861             -0.140667
d:walking_hr_verage              -0.024763             -0.047746
y:time_in_bed                    -0.090222              0.024514
y:time_asleep                    -0.066581              0.018091
y:wakeup_number                  -0.105162              0.028574
y:sleep_score                     0.005007             -0.001360
y:stress_score                   -0.124725              0.033889

                       d:oxygen_saturation  d:resting_hr  d:walking_hr_verage  \
d:heart_beat_series              -0.037140      0.145861            -0.024763
d:hr_variability_SDNN            -0.145225     -0.140667            -0.047746
d:oxygen_saturation               1.000000      0.020778             0.052204
d:resting_hr                      0.020778      1.000000             0.328875
d:walking_hr_verage               0.052204      0.328875             1.000000
y:time_in_bed                    -0.018202     -0.154540            -0.109377
y:time_asleep                    -0.013432     -0.114044            -0.080716
y:wakeup_number                  -0.021216     -0.180130            -0.127489
y:sleep_score                     0.001010      0.008576             0.006070
y:stress_score                   -0.025163     -0.213638            -0.151205

                       y:time_in_bed  y:time_asleep  y:wakeup_number  \
d:heart_beat_series        -0.090222      -0.066581        -0.105162
d:hr_variability_SDNN       0.024514       0.018091         0.028574
d:oxygen_saturation        -0.018202      -0.013432        -0.021216
d:resting_hr               -0.154540      -0.114044        -0.180130
d:walking_hr_verage        -0.109377      -0.080716        -0.127489
y:time_in_bed               1.000000       0.898565         0.649318
y:time_asleep               0.898565       1.000000         0.530634
y:wakeup_number             0.649318       0.530634         1.000000
y:sleep_score               0.227563       0.482138         0.065688
y:stress_score              0.040018      -0.069236         0.208551

                       y:sleep_score  y:stress_score
d:heart_beat_series         0.005007       -0.124725
d:hr_variability_SDNN      -0.001360        0.033889
d:oxygen_saturation         0.001010       -0.025163
d:resting_hr                0.008576       -0.213638
d:walking_hr_verage         0.006070       -0.151205
y:time_in_bed               0.227563        0.040018
y:time_asleep               0.482138       -0.069236
y:wakeup_number             0.065688        0.208551
y:sleep_score               1.000000       -0.095462
y:stress_score             -0.095462        1.000000
```

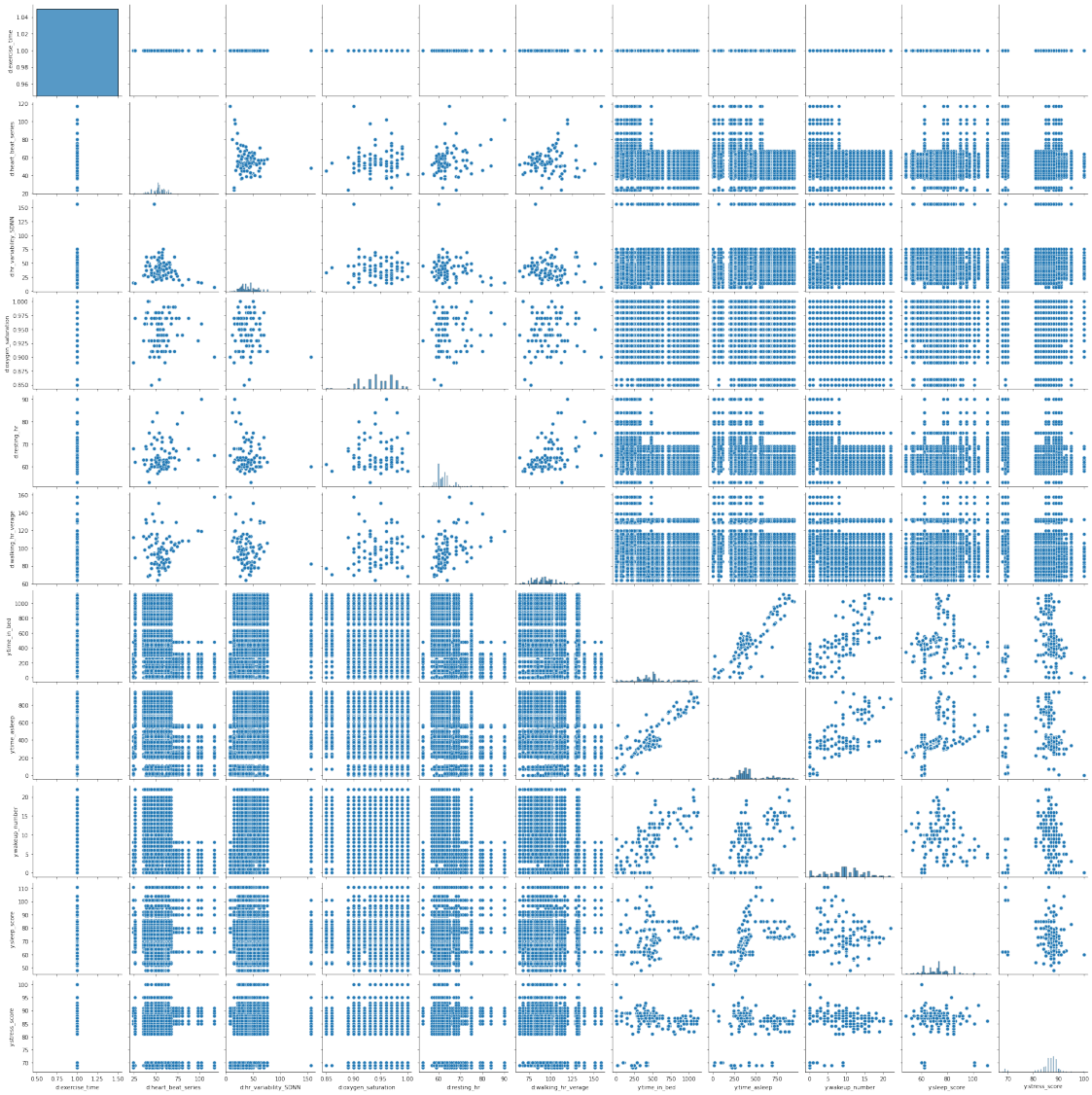
[ ]: <VegaLite 4 object>

If you see this message, it means the renderer has not been properly enabled

```
for the frontend that you are using. For more information, see
https://altair-viz.github.io/user_guide/troubleshooting.html
```

[ ]:
```
import seaborn as sns
sns.pairplot(df_plots.dropna())
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7fb61f242820>
```



0.3 IV related

[ ]:
```
df[["y:sleep_score"]].dropna().max()
```

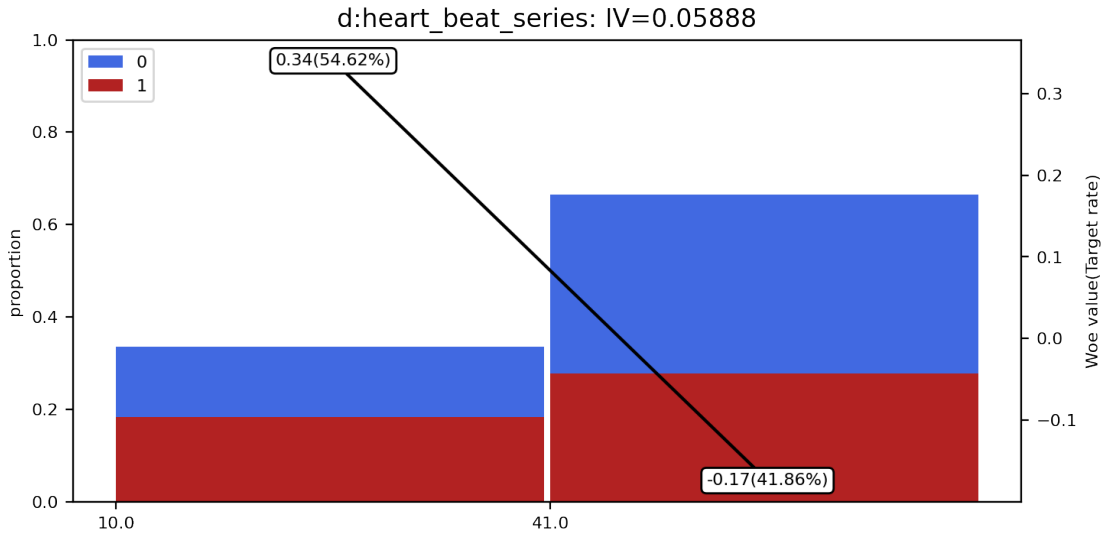
[ ]:
```
y:sleep_score    111.0
dtype: float64
```

[ ]:
```
qmin, qmax = df["y:sleep_score"].quantile(0.25), df["y:sleep_score"].quantile(0.
↪75)
```

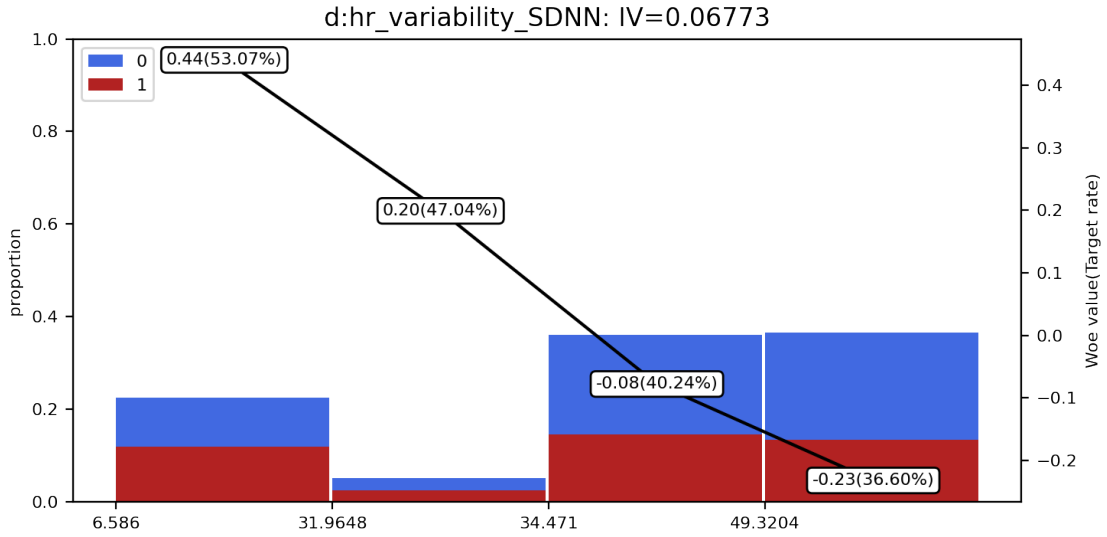
[ ]:
```
check_list = [
#  'd:exercise_time',
 'd:heart_beat_series',
 'd:hr_variability_SDNN',
 'd:oxygen_saturation',
 'd:resting_hr',
 'd:walking_hr_verage',
 'y:time_in_bed',
 'y:time_asleep',
 'y:wakeup_number',
#  'y:sleep_score',
#  'y:stress_score'
 ]

for i in check_list:
    pass
```

[ ]:
```
from woe_tools import *


def handle_label(x):
    if x<qmin:
        return 1
    elif x>qmax:
        return 0
    else:
        return -1


for i in check_list:
    df_tmp = df[["y:sleep_score", i]].dropna()
    df_tmp["y:sleep_score"] = df_tmp["y:sleep_score"].apply(handle_label)
    df_tmp = df_tmp[df_tmp["y:sleep_score"] != -1]
    numwoe_autobinning(df_tmp, i, "y:sleep_score")
```

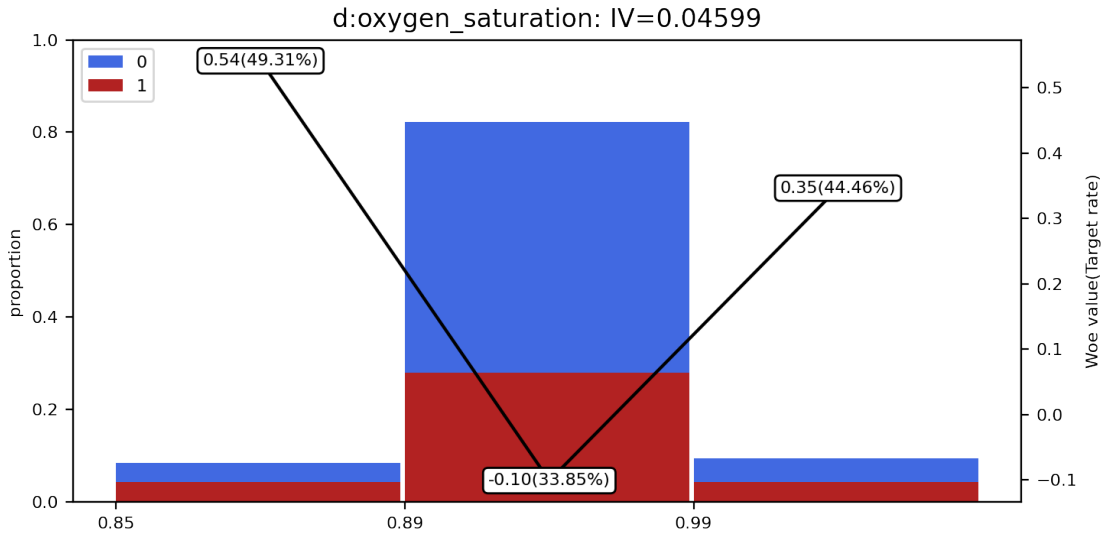
```
WARNING:root:Expect variable has missing value but actually no missing
```



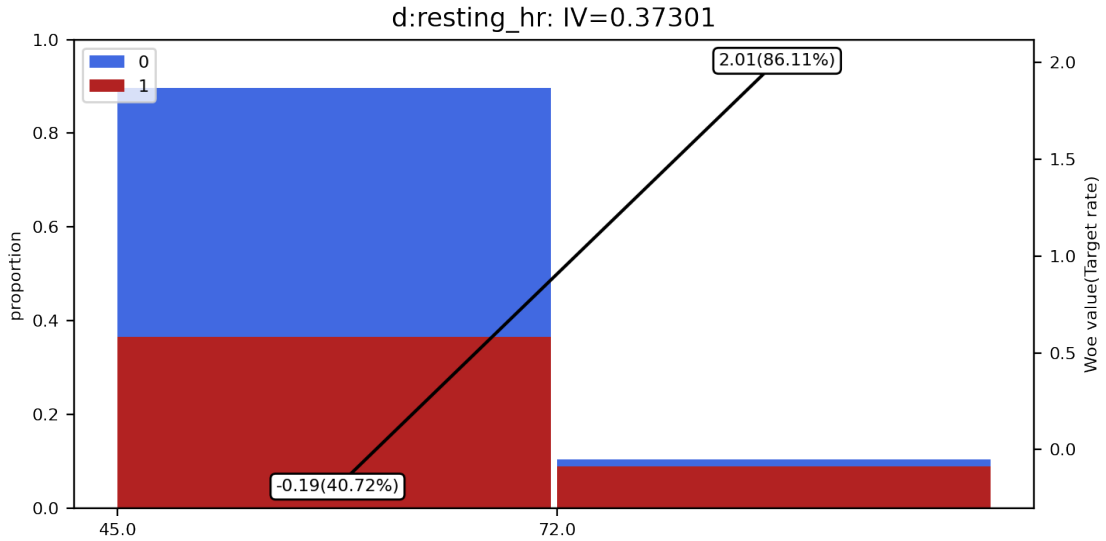

```
WARNING:root:Expect variable has missing value but actually no missing
```



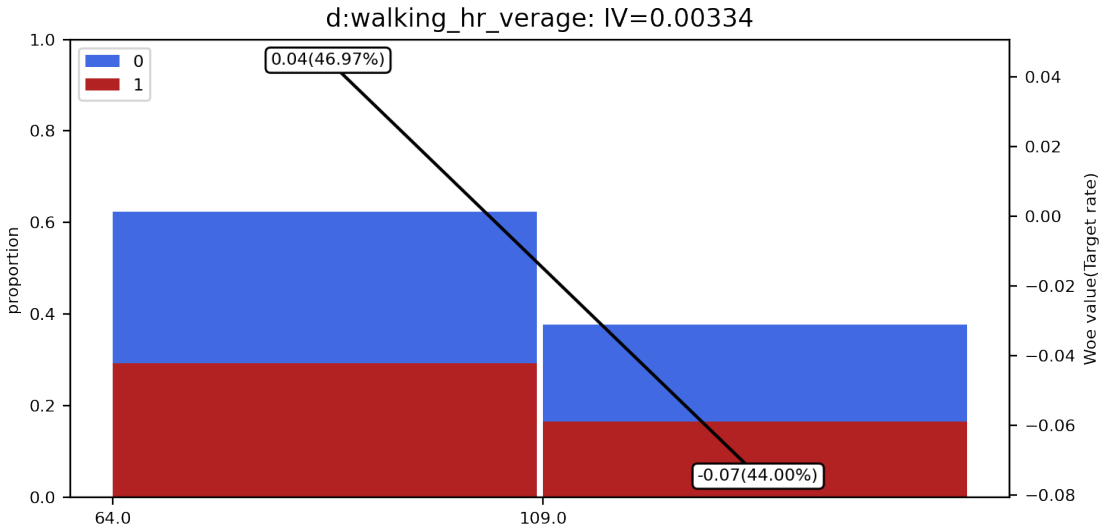

```
WARNING:root:Expect variable has missing value but actually no missing
```



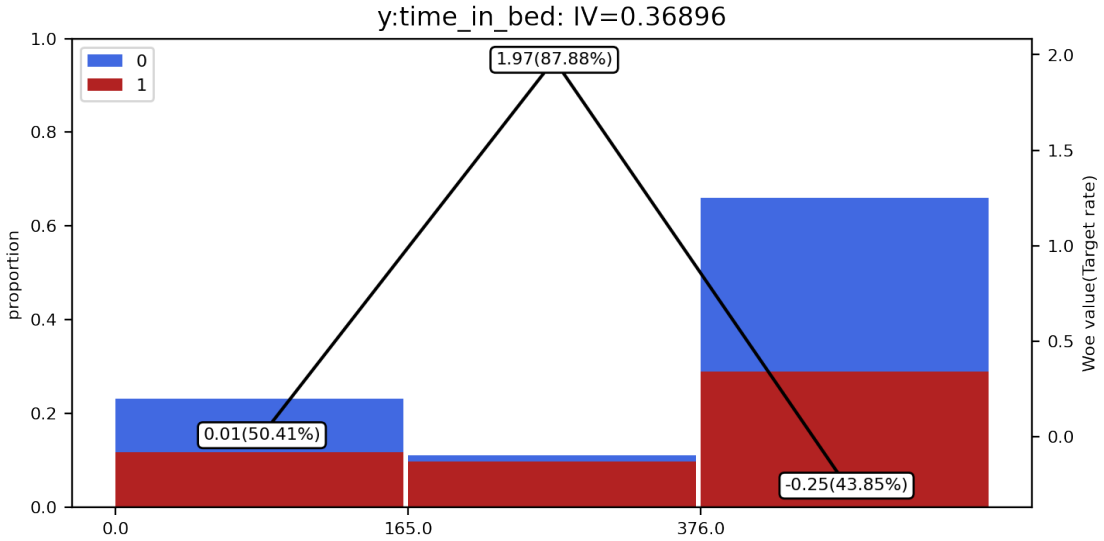

```
WARNING:root:Expect variable has missing value but actually no missing
```



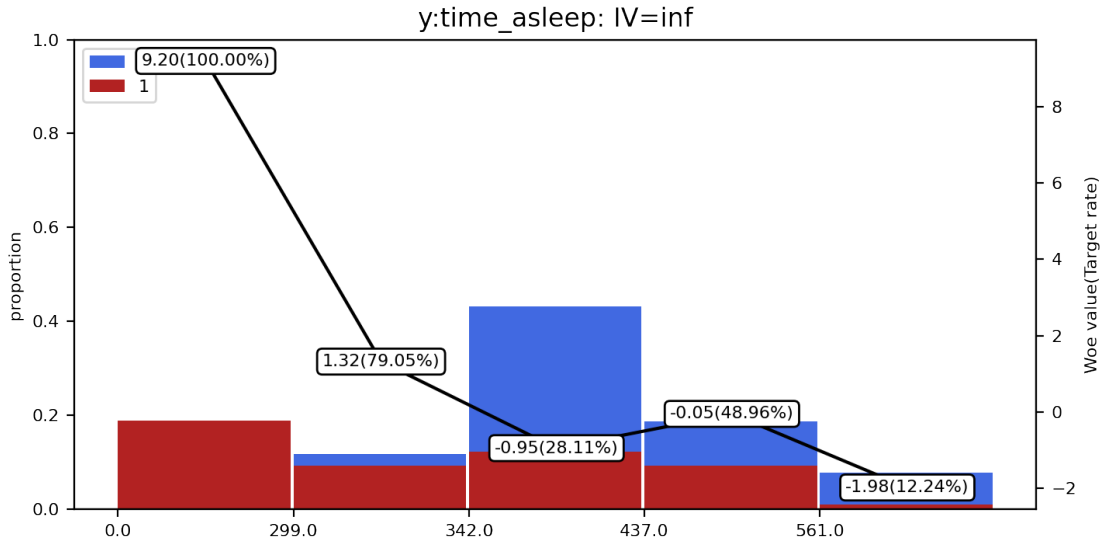

```
WARNING:root:Expect variable has missing value but actually no missing
```



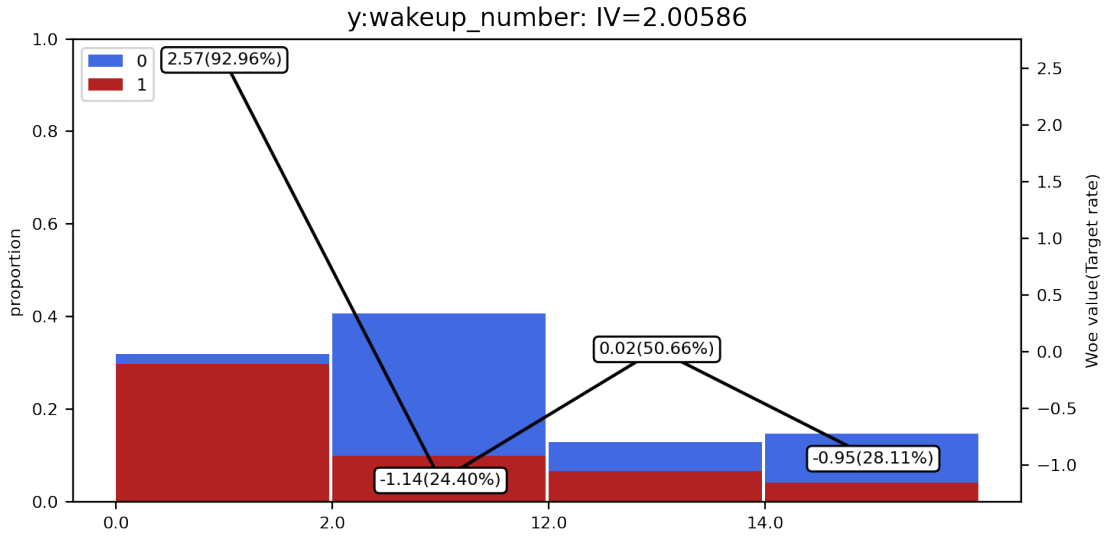
WARNING:root:Expect variable has missing value but actually no missing



WARNING:root:Expect variable has missing value but actually no missing




```
WARNING:root:Expect variable has missing value but actually no missing
```



[ ]: