Using Linear Transformations to Predict a Diagnosis through Biomarkers

Author: Addy Moran

Disclaimer: I used Gemini to help with the plotting of the data and to check my work.

The Application

Biomarkers have been used for decades in the diagnosis of illnesses and in drug effectiveness testing. Biomarkers can come from a variety of sources, for example, blood samples or through non-invasive methods to gather things like heart rate or blood pressure. Biomarkers are in constant development and investigation, this week they've found new blood biomarkers to help predict lung cancer before it's noticeable in CT scans [1]. Biomarkers have also been used in emergency situations like battle field operations and in the emergency room [2].

The combination of multiple biomarkers often increase accuracy. The accuracy of biomarker predictions is commonly discussed in terms of sensitivity and specificity, where sensitivity is the true positive rate and the specificity as the true negative rate [3]. However, especially in urgent situations, some tests take too long and having too much data may lead to analysis paralysis. Therefore, it's important to understand how each biomarker impacts the diagnosis, thereby enabling more informed decisions if a test or data source isn't realistic. It is common to use statistics to combine and determine the importance of certain biomarkers for a diagnosis, and many statistical approaches rely on the distribution to be known (for example that the relationship between the biomarker and the diagnosis is Poisson). The primary paper investigated [3] relies on normal distributions, however, papers [4] and [5] expand on [3] and experiment with this assumption.

The Math

The approach discussed by Su and Liu in 1993 [3] relies on normal distributions (and uses Box-Cox to help with that) to suggest the optimal coefficients (or weights) for each biomarker.

Let $N = (N_1, N_2, ..., N_P)$ be the biomarker values in the "non-diseased" group and $D = (D_1, D_2, ..., D_Q)$ be the biomarker values in the diseased group. Assume both N and D are normally distributed random variables.

These coefficients can be expressed using this projection vector $a = (\mu_N - \mu_D)^T \Sigma^{-1}$ where μ_Y is the average biomarker value for the non-diseased, μ_D is the average for the diseased, and Σ is the covariance of the two.

$$U = a^T N \sim Normal(a^T \mu_N, a^T \Sigma_N a)$$

$$V = a^T D \sim Normal(a^T \mu_D, a^T \Sigma_D a)$$

The specificity is then the cumulative distribution function (cdf) of U at some cut off c and the sensitivity is $1 - cdf(V)$ at the same c.

Simple Example

Here is a simple, concrete example using fake data for simplicity.

Table 1 shows the raw data of two biomarkers collected for four people, two of which were diagnosed as a zombie and two were not.

Table 1: Raw Data

Biomarker	**Diseased 1**	**Diseased 1**	**Non-Diseased 1**	**Non-Diseased 2**
Blood Oxygen (%)	89	87	93	97
Glucose (mg/dL)	180	178	99	110

The average of each biomarker for both the diseased and non-diseased group can be found in Table 2.

Table 2: Biomarker Averages

Biomarker	**Diseased (μ_D)**	**Non-Diseased (μ_N)**
Blood Oxygen (%)	88	95
Glucose (mg/dL)	179	104.5

As matrices, they look like this:

$$\mu_D = \begin{bmatrix} 0.88 \\ 179 \end{bmatrix}, \mu_N = \begin{bmatrix} 0.95 \\ 104.5 \end{bmatrix}$$

Then we find the covariance matrix for both the diseased and non-diseased groups.

$$\Sigma_D = \begin{bmatrix} Var(Oxy) & Cov(Oxy, Glu) \\ (Cov(Glu, Oxy) & Var(Glu) \end{bmatrix} = \begin{bmatrix} 0.0002 & 0.02 \\ 0.02 & 2 \end{bmatrix}$$

$$\Sigma_N = \begin{bmatrix} 0.0008 & 0.22 \\ 0.22 & 60.5 \end{bmatrix}$$

We then find the average covariance matrix (and it's inverse) across the two groups:

$$\Sigma = \frac{\Sigma_D + \Sigma_N}{2} = \begin{bmatrix} 0.0005 & 0.12 \\ 0.12 & 31.25 \end{bmatrix}$$

$$\Sigma^{-1} = \begin{bmatrix} 25510.2041 & -97.9592 \\ -97.9592 & 0.4082 \end{bmatrix}$$

From there we can find the ideal coefficients for each biomarker based on this definition: $a = (a_1, ..., a_n)^T : i = 0...n, a_i = (\mu_N - \mu_D)^T \Sigma^{-1}$.

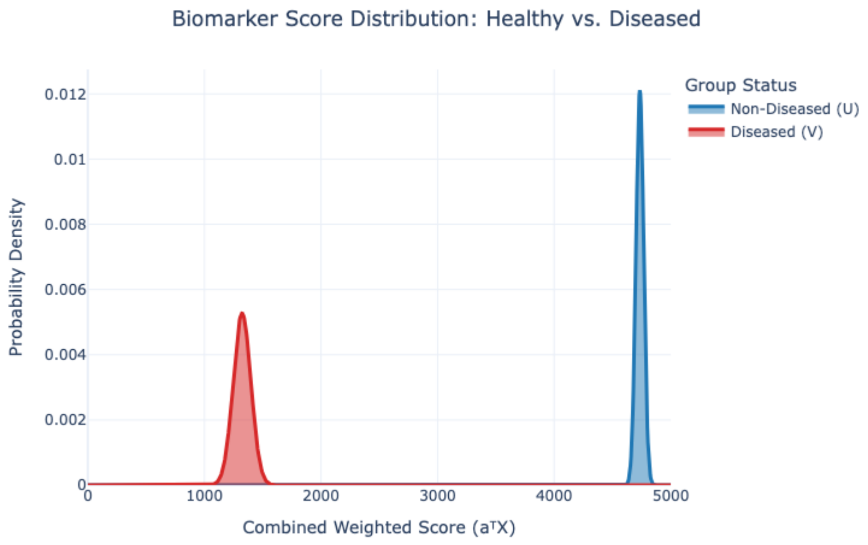
$$(\mu_N - \mu_D)^T = \begin{bmatrix} 0.07 & -74.5 \end{bmatrix}$$

$$(\mu_N - \mu_D)^T \Sigma^{-1} = \begin{bmatrix} 0.07 & -74.5 \end{bmatrix} \begin{bmatrix} 25510.2041 & -97.9592 \\ -97.9592 & 0.4082 \end{bmatrix} = \begin{bmatrix} 9083.6735 \\ -37.2653 \end{bmatrix}$$

Therefore the optimal weights of the oxygen saturation is 9083.67 and for glucose levels it's -37.27. From there we can find the sensitivity and specificity.

$$\begin{gathered} U = a^T X = N(a^T \mu_N, a^T \Sigma_N a) \\ = N(\begin{bmatrix} 9083.6735 & -37.2653 \end{bmatrix} \begin{bmatrix} 0.95 \\ 104.5 \end{bmatrix}, \begin{bmatrix} 9083.6735 & -37.2653 \end{bmatrix} \begin{bmatrix} 0.0008 & 0.22 \\ 0.22 & 60.5 \end{bmatrix} \begin{bmatrix} 9083.6735 \\ -37.2653 \end{bmatrix}) \\ = N(4735.265306122423, 32.93097295240176) \end{gathered}$$

$$\begin{gathered} V = a^T Y = N(a^T \mu_D, a^T \Sigma_D a) \\ = N(\begin{bmatrix} 9083.6735 & -37.2653 \end{bmatrix} \begin{bmatrix} 0.88 \\ 179 \end{bmatrix}, \begin{bmatrix} 9083.6735 & -37.2653 \end{bmatrix} \begin{bmatrix} 0.0002 & 0.02 \\ 0.02 & 2 \end{bmatrix} \begin{bmatrix} 9083.6735 \\ -37.2653 \end{bmatrix}) \\ = N(866.3792, 2278.584) \end{gathered}$$



Since the two PDFs don't intersect, we'll calculate the cut off value based on the space between the two. The non-diseased PDF is practically 0 when the weighted score is less than 4625 and the diseased PDF is practically 0 when the weighted score is greater than 1542, so we'll find the middle point, which is about 3000. When $c = 3000$ the CDF of U (the non-diseased) is 1 and the CDF of V is 0. Therefore the specificity is > 99% and the sensitivity is >99%.

Now that we've seen that the classifier is accurate (based on our small sample set), if we had a third person, Gladys, and we wanted to predict if she was infected with the zombie disease based on her biomarker values (glucose of 165 and blood oxygen level of 91%),

$$\begin{bmatrix} 9083.6735 \\ -37.2653 \end{bmatrix} \begin{bmatrix} 0.91 \\ 165 \end{bmatrix} = 2117.3673$$

Because the 2117 is less than 3000 we can confidently say Gladys is a zombie.

Transforming Sepsis Biomarkers

In the previous example we had a small sample size and only two biomarkers. We'll now look at a dataset from the PhysioNet Computing in Cardiology Challenge 2019 [6]. The dataset has biomarkers from 40,336 people (a total of 1,552,210 rows). This dataset records health statistics marked by an hour so you can see the progression to sepsis (if there is one). As part of the preprocessing, we determine when the patient gets labeled as sepsis (if at all), Table 3 shows an example row but we need to consolidate the list of biomarker values down to a single number.

Table 3: Example Data for Patient 8

gender	**age**	**sepsis_index**	**HR**	**O2Sat**	**SBP**	**DBP**	**MAP**	**Resp**	**FiO2**	**Sepsis Label**
1	27.92	248	[nan, 117.0, nan, nan, nan, nan, 120.0, 109.5, ...	[nan, 99.0, nan, nan, nan, nan, 100.0, 98.5, 9...	[nan, 116.0, nan, nan, nan, nan, 118.0, 106.0, ...	[nan, 81.0, nan, nan, nan, nan, 64.0, 59.0, 58...	[nan, 97.0, nan, nan, nan, nan, 84.0, 74.0, 72...	[nan, 20.0, nan, nan, nan, nan, 30.0, 27.0, 24...	[nan, nan, nan, nan, nan, nan, 1.0, nan, 1.0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

If the patient does not go into sepsis we average their vitals, for example if they have six non-null HR records, we'll average them. If they do go into sepsis, we'll capture the closest HR measurement at the time of labeling, for example, if they get marked as sepsis in their 6th hour admitted to the hospital, we'll find the HR measurement in the same hour (or as close as possible). See Table 4 for an example.

Table 4: Augmented Data for Patient 8

gender	age	sepsis_index	HR	O2Satsingle_pt	SBPsingle_pt	DBPsingle_pt	MAPsingle_pt	Respsingle_pt	FiO2single_pt
1	27.92	248	119.0	100.0	140.0	85.0	106.0	26.5	0.35

We want to verify these columns are normal. When plotting and visually looking for normality we find that the age, HR, SBP, DBP, MAP and Resp fields look closest to a normal distribution (see Appendix for graphs).

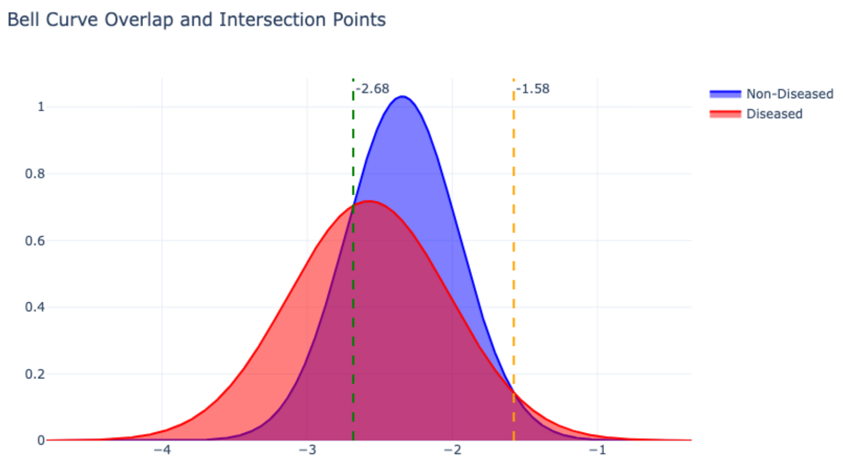
Following the same steps as in the simple example, we find the average of each biomarker across the diseased and non-diseased group, then find the covariance matrix for both.

```
cols = ['age', 'HRsingle_pt', 'SBPsingle_pt', 'DBPsingle_pt', 'MAPsingle_pt',
    'Respsingle_pt']

# dropping the columns where there was no data for a column (i.e. the average is NaN),
    this prevents NaN's in the covariance matrix
diseased = diseased[cols].dropna() # 2,425 total records
non_diseased = non_diseased[cols].dropna() # 30,376 records

mu_d = []
mu_n = []

diseased_summary = diseased.describe()
nondiseased_summary = non_diseased.describe()

for col in cols:
    mu_d.append(diseased_summary[col]['mean'])
    mu_n.append(nondiseased_summary[col]['mean'])

mu_d = np.array(mu_d)
mu_n = np.array(mu_n)

cov_n = np.cov(non_diseased.T)
cov_d = np.cov(diseased.T)

cov = (cov_d + cov_n)/2
cov_inverse = np.linalg.inv(cov)

a = (mu_n - mu_d).T @ cov_inverse
```

$$a = \begin{bmatrix} Age = 0.0011 \\ HR = -0.0223 \\ SBP = -0.0085 \\ DBP = 0.0134 \\ MAP = 0.0061 \\ Resp = -0.0472 \end{bmatrix}$$

Notice, that all of these weights are very close to 0, hinting that there isn't much correlation. In practice, we'd likely want to remove any weights close to 0 since they don't add much value. Out of curiosity I used a Spearman Correlation heat map to see what fields correlated the most with the sepsis index, and there was very little (Fi02 was the closest but it wasn't normal so wasn't included in this experiment). This heatmap can be found in the appendix.

The weights (a) result in the following PDF bell curves



There is a large overlap, solidifying that with these biomarkers, we'll need significant compromise in either sensitivity or specificity. Since this is a medical application, we'll focus on sensitivity (to make sure we don't miss sepsis and accept we'll likely label patients as sepsis when they aren't). So we'll set a cutoff value of -3.25, which results in a sensitivity of 88% and a specificity of 1.6%.

If we apply the weights to a test subject, Steve, who has the following biomarker values,

$$\begin{bmatrix} age = 65 \\ HR = 100 \\ SBP = 122 \\ DBP = 61 \\ MAP = 82 \\ Resp = 15 \end{bmatrix}$$

we'd classify him as not having Sepsis because when multiplying his stats by the calculated weights (a) we get -2.587 which is larger than the cutoff value threshold of -3.25.

Further Research

There has been additional research ([4], [5]) in a distribution-free approach, it would be neat explore these concepts, potentially allowing me to get rid of some of the near 0 weights.

There are likely flaws in my assumptions and preprocessing steps (i.e. averaging vs. selecting a single data point if they're sepsis). These decisions were made to enable rapid testing of [3] on real data, in real life, there should be different decisions made.

References

[1] Optica. (2026, February 16). This new blood test could detect cancer before it shows up on scans. ScienceDaily. https://www.sciencedaily.com/releases/2026/02/260216044002.htm

[2] Hong, Y., Li, L.-H., Kuo, T.-H., Lee, Y.-T., & Hsu, C.-C. (2025). Early Prediction of Septic Shock in Emergency Department Using Serum Metabolites. Journal of the American Society for Mass Spectrometry, 36(6), 1264–1276. https://doi.org/10.1021/jasms.5c00009

[3] Su, J. Q., & Liu, J. S. (1993). Linear Combinations of Multiple Diagnostic Markers. Journal of the American Statistical Association, 88(424), 1350–1355. https://doi.org/10.2307/2291276

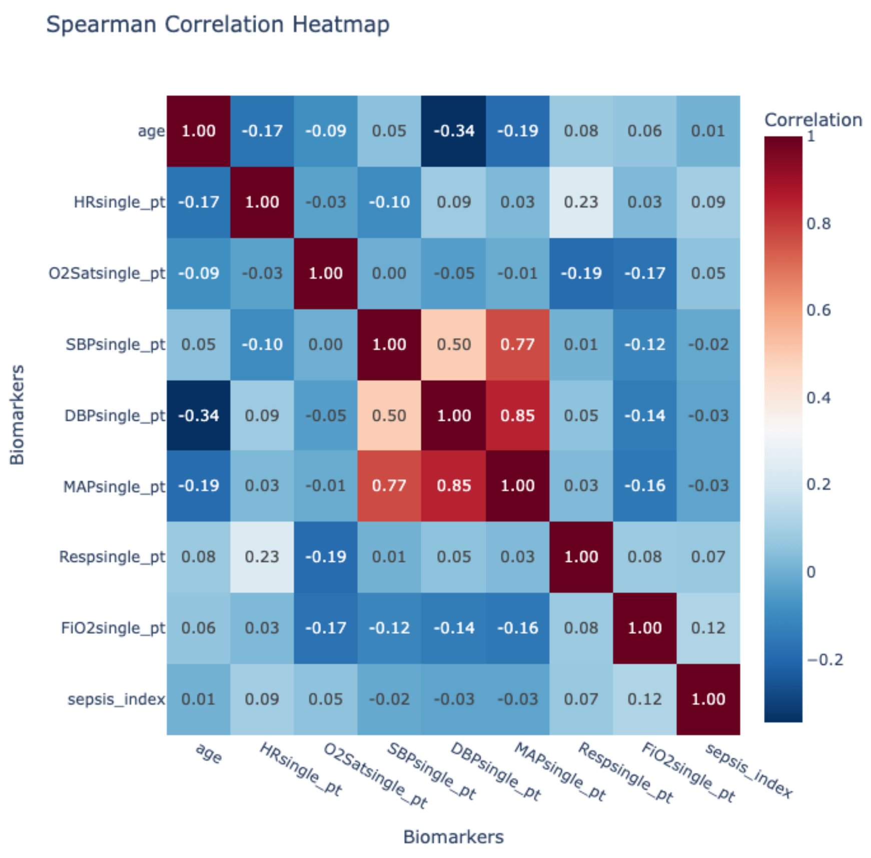
[4] Liu, A., Schisterman, E.F. and Zhu, Y. (2005), On linear combinations of biomarkers to improve diagnostic accuracy. Statist. Med., 24: 37-47. https://doi.org/10.1002/sim.1922

[5] Pfeiffer, R.M. and Bur, E. (2008), A Model Free Approach to Combining Biomarkers. Biom. J., 50: 558-570. https://doi.org/10.1002/bimj.200710428

[6] Hussaini, S. (2021). Prediction of sepsis [Data set] Kaggle. https://www.kaggle.com/datasets/salikhussaini49/prediction-of-sepsis

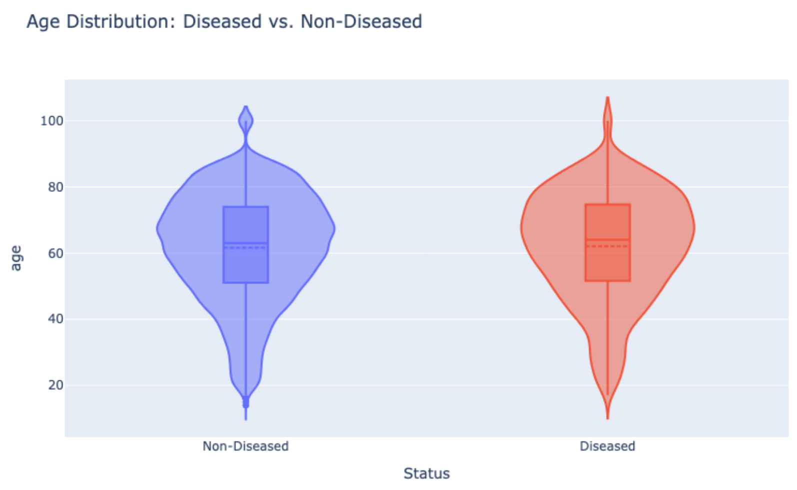
Appendix

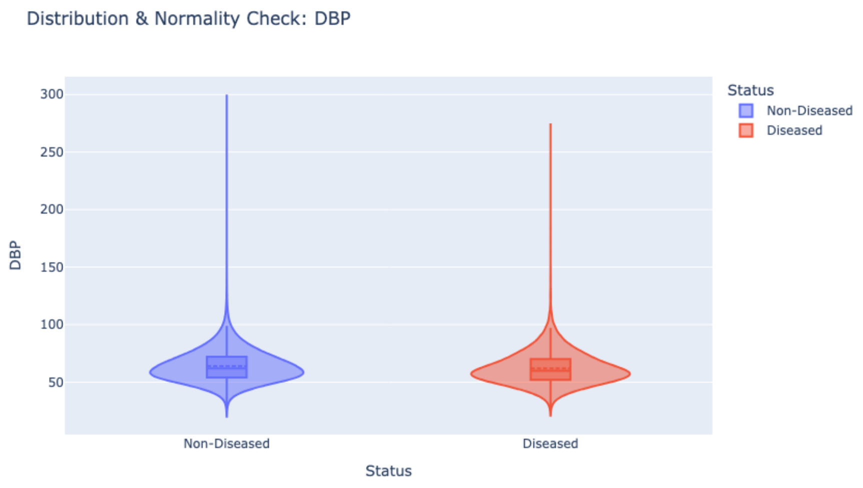
Spearman's

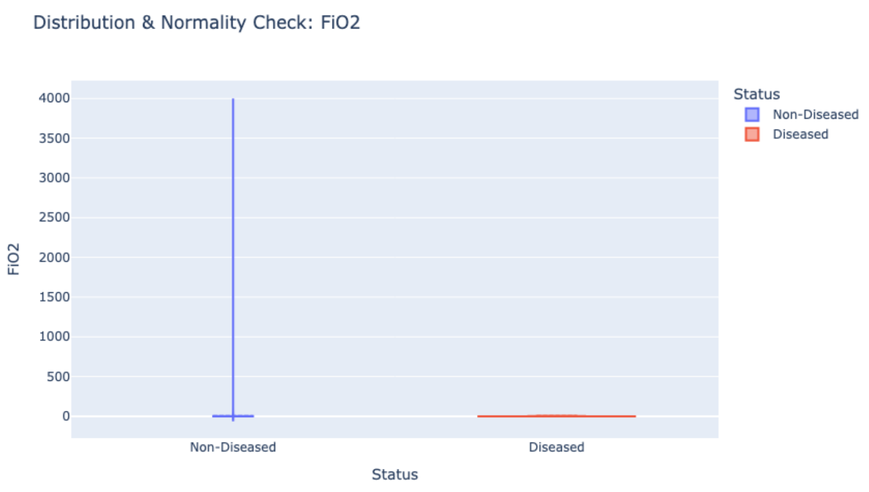


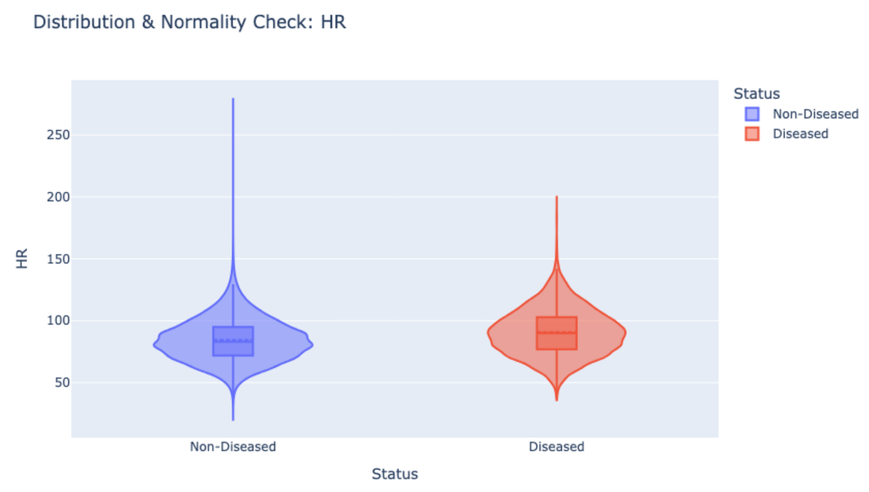
	age	HRsingle_pt	O2Satsingle_pt	SBPsingle_pt	DBPsingle_pt	MAPsingle_pt	Respsingle_pt	FiO2single_pt	sepsis_index
age	1.00	-0.17	-0.09	0.05	-0.34	-0.19	0.08	0.06	0.01
HRsingle_pt	-0.17	1.00	-0.03	-0.10	0.09	0.03	0.23	0.03	0.09
O2Satsingle_pt	-0.09	-0.03	1.00	0.00	-0.05	-0.01	-0.19	-0.17	0.05
SBPsingle_pt	0.05	-0.10	0.00	1.00	0.50	0.77	0.01	-0.12	-0.02
DBPsingle_pt	-0.34	0.09	-0.05	0.50	1.00	0.85	0.05	-0.14	-0.03
MAPsingle_pt	-0.19	0.03	-0.01	0.77	0.85	1.00	0.03	-0.16	-0.03
Respsingle_pt	0.08	0.23	-0.19	0.01	0.05	0.03	1.00	0.08	0.07
FiO2single_pt	0.06	0.03	-0.17	-0.12	-0.14	-0.16	0.08	1.00	0.12
sepsis_index	0.01	0.09	0.05	-0.02	-0.03	-0.03	0.07	0.12	1.00

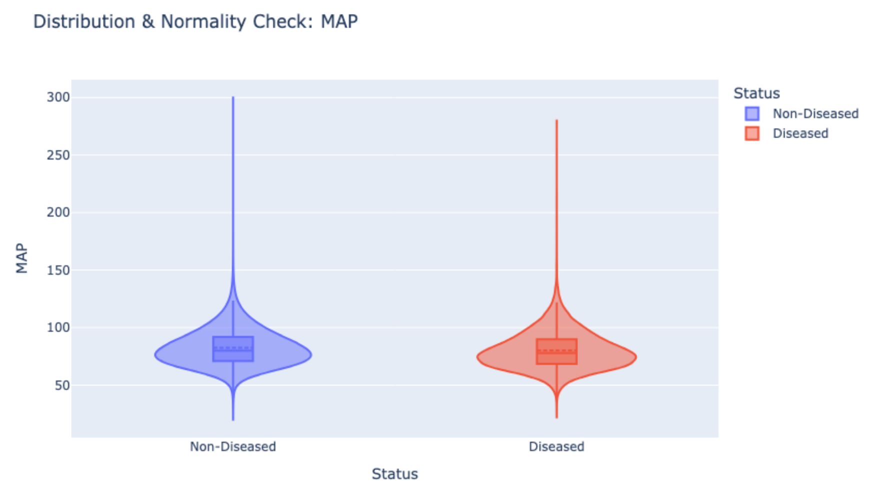
Violin Graphs

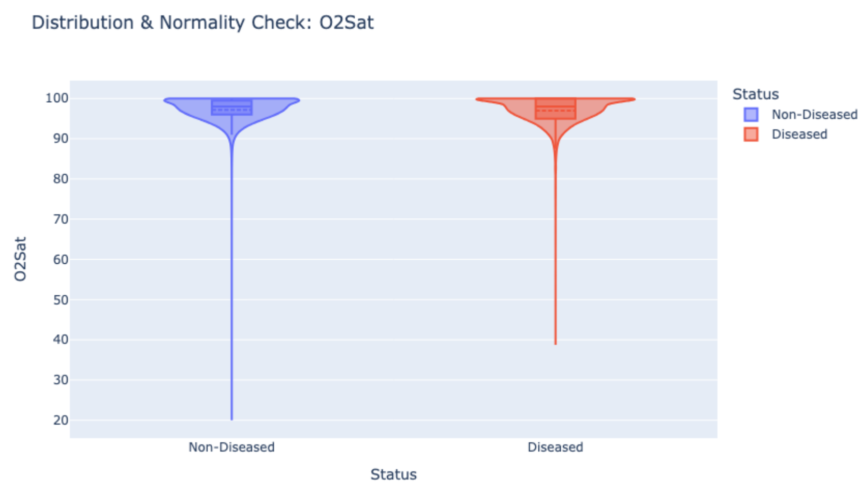


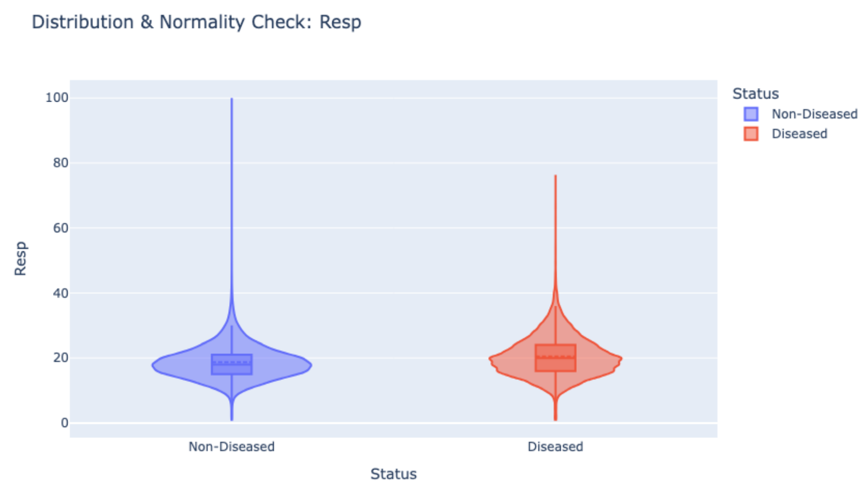


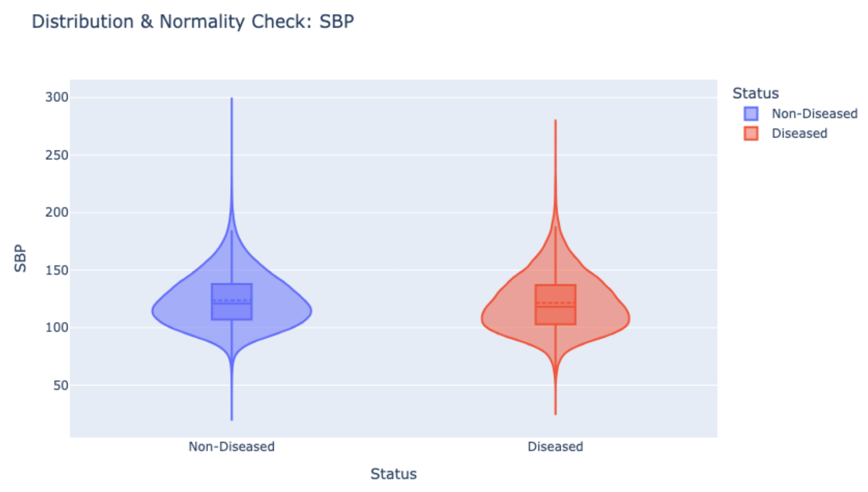

Distribution & Normality Check: FiO2
Status
Non-Diseased
Diseased
FiO2
4000
3500
3000
2500
2000
1500
1000
500
0
Non-Diseased
Diseased
Status


Distribution & Normality Check: HR
Status
Non-Diseased
Diseased
HR
250
200
150
100
50
Non-Diseased
Diseased
Status


Distribution & Normality Check: MAP
Status
Non-Diseased
Diseased
MAP
300
250
200
150
100
50
Non-Diseased
Diseased
Status


Distribution & Normality Check: O2Sat
O2Sat
100
90
80
70
60
50
40
30
20
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased


Distribution & Normality Check: Resp
Resp
100
80
60
40
20
0
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased


Distribution & Normality Check: SBP
SBP
300
250
200
150
100
50
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased

Jupyter Notebooks

See the iPy notebook exports below.

simple_example

February 22, 2026

0.1 Simple Example

Disclaimer: Gemini helped with some of the graphing.

[77]:
```
import plotly.io as pio
pio.renderers.default = "notebook+pdf"
import numpy as np

np.set_printoptions(precision=4, suppress=True)
```

[78]:
```
raw = np.array([[0.89, 0.87, 0.93, 0.97], [180, 178, 99, 110]])
raw
```

[78]:
```
array([[  0.89,   0.87,   0.93,   0.97],
       [180.  , 178.  ,  99.  , 110.  ]])
```

[79]:
```
diseased = np.array([raw[0][:2], raw[1][:2]])
non_diseased = np.array([raw[0][2:], raw[1][2:]])
diseased, non_diseased
```

[79]:
```
(array([[  0.89,   0.87],
        [180.  , 178.  ]]),
 array([[  0.93,   0.97],
        [ 99.  , 110.  ]]))
```

[80]:
```
mu_d = np.array([diseased[0].mean(), diseased[1].mean()])
mu_n = np.array([non_diseased[0].mean(), non_diseased[1].mean()])
mu_d, mu_n
```

[80]:
```
(array([  0.88, 179.  ]), array([  0.95, 104.5 ]))
```

[81]:
```
sigma_d = np.cov(diseased)
sigma_n = np.cov(non_diseased)
(sigma_d, sigma_n)
```

[81]:
```
(array([[0.0002, 0.02  ],
        [0.02  , 2.    ]]),
 array([[ 0.0008,  0.22  ],
        [ 0.22  , 60.5   ]]))
```

[82]:
```
sigma = (sigma_d + sigma_n)/2
sigma
```

[82]:
```
array([[ 0.0005,  0.12  ],
       [ 0.12  , 31.25  ]])
```

[83]:
```
sigma_inverse = np.linalg.inv(sigma)
sigma_inverse
```

[83]:
```
array([[25510.2041,   -97.9592],
       [  -97.9592,     0.4082]])
```

[84]:
```
a = (mu_n - mu_d).T@sigma_inverse
a
```

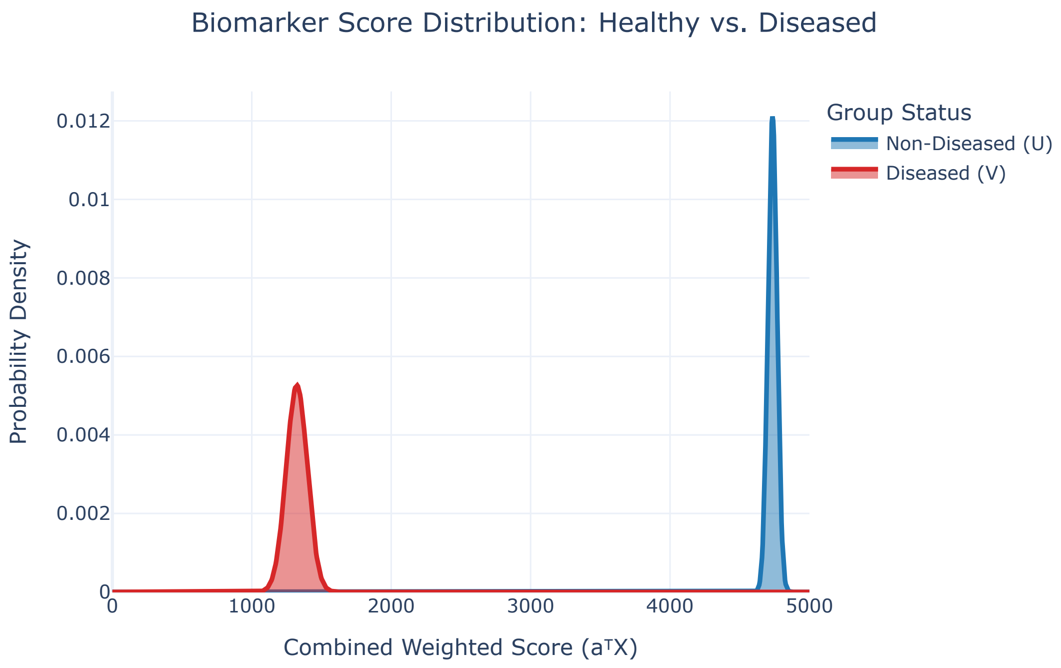
[84]:
```
array([9083.6735,  -37.2653])
```

[85]:
```
mu_u, std_u = a.T@mu_n, np.sqrt(a.T@sigma_n@a)
mu_v, std_v = a.T@mu_d, np.sqrt(a.T@sigma_d@a)
(mu_u, std_u), (mu_v, std_v)
```

[85]:
```
((np.float64(4735.265306122423), np.float64(32.93097295240176)),
 (np.float64(1323.142857142855), np.float64(75.76144084141548)))
```

[86]:
```
from scipy.stats import norm

x = np.linspace(0, 5000, 1000)
u = norm.pdf(x, mu_u, std_u)
v = norm.pdf(x, mu_v, std_v)
```

[87]:
```
import plotly.graph_objects as go

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=x, y=u,
    mode='lines',
    name='Non-Diseased (U)',
    line=dict(color='#1f77b4', width=3),
    fill='tozeroy',
    hovertemplate='Score: %{x:.2f}<br>Density: %{y:.4f}'
))

fig.add_trace(go.Scatter(
    x=x, y=v,
    mode='lines',
    name='Diseased (V)',
    line=dict(color='#d62728', width=3),
    fill='tozeroy',
    hovertemplate='Score: %{x:.2f}<br>Density: %{y:.4f}'
))

fig.update_layout(
    title={
        'text': 'Biomarker Score Distribution: Healthy vs. Diseased',
        'y':0.9, 'x':0.5, 'xanchor': 'center', 'yanchor': 'top'
    },
    xaxis_title='Combined Weighted Score (a X)',
    yaxis_title='Probability Density',
    legend_title='Group Status',
    template='plotly_white',
    hovermode='x unified'
)

fig.show()
```



[88]:
```
norm.cdf(3000, loc=mu_u, scale=std_u)
```

[88]:
```
np.float64(0.0)
```

[89]:
```
norm.cdf(3000, loc=mu_v, scale=std_v)
```

[89]:
```
np.float64(1.0)
```

[90]:
```
1 - norm.cdf(3000, loc=mu_v, scale=std_v)
```

[90]:
```
np.float64(0.0)
```

[91]:
```
a@np.array([[0.91], [165]])
```

[91]:
```
array([2117.3673])
```

sepsis_linear_transforms

February 22, 2026

0.1 Sepsis Biomarker Transformation

Disclaimer: Gemini helped with some of the graphing.

[1]:
```
import numpy as np
import pandas as pd

import plotly.io as pio
pio.renderers.default = "notebook+pdf"

np.set_printoptions(precision=4, suppress=True)
```

[2]:
```
# Dataset Source:
#    Hussaini, S. (2021). Prediction of sepsis [Data set](#). Kaggle.
#    https://www.kaggle.com/datasets/salikhussaini49/prediction-of-sepsis

df = pd.read_csv('./sepsis.csv')
df.head()
```

[2]:
```
   Unnamed: 0  Hour    HR  O2Sat  Temp    SBP   MAP   DBP  Resp  EtCO2  …  \
0           0     0   NaN    NaN   NaN    NaN   NaN   NaN   NaN    NaN  …
1           1     1  65.0  100.0   NaN    NaN  72.0   NaN  16.5    NaN  …
2           2     2  78.0  100.0   NaN    NaN  42.5   NaN   NaN    NaN  …
3           3     3  73.0  100.0   NaN    NaN   NaN   NaN  17.0    NaN  …
4           4     4  70.0  100.0   NaN  129.0  74.0  69.0  14.0    NaN  …

   Fibrinogen  Platelets    Age  Gender  Unit1  Unit2  HospAdmTime  ICULOS  \
0         NaN        NaN  68.54       0    NaN    NaN        -0.02       1
1         NaN        NaN  68.54       0    NaN    NaN        -0.02       2
2         NaN        NaN  68.54       0    NaN    NaN        -0.02       3
3         NaN        NaN  68.54       0    NaN    NaN        -0.02       4
4         NaN      330.0  68.54       0    NaN    NaN        -0.02       5

   SepsisLabel  Patient_ID
0            0       17072
1            0       17072
2            0       17072
3            0       17072
```

```
4              0       17072

[5 rows x 44 columns]
```

[3]:
```
patients = []

biomarker_columns = ['HR', 'O2Sat', 'SBP', 'DBP', 'MAP', 'Resp', 'FiO2',␣
 ↪'SepsisLabel'] # keeping SepsisLabel so we know which hour they go into␣
 ↪Sepsis
for i, patient in df.groupby('Patient_ID'):
    # want an ordered list of vitals by hour, drop any nan (only get columns␣
 ↪were all of the fields are filled for each hour)
    p = {'gender': patient['Gender'].iloc[0], 'age': patient['Age'].iloc[0]}

    p['sepsis_index'] = patient['SepsisLabel'].values.argmax() if␣
 ↪(patient['SepsisLabel'] == 1).any() else -1

    for column in biomarker_columns:
        p[column] = list(patient[column])

    patients.append(p)
patients = pd.DataFrame(patients)

print(f"Number of patients in dataset {len(patients)} and {len(df)} total rows")
patients.head(10)
```

```
Number of patients in dataset 40336 and 1552210 total rows
```

[3]:
```
   gender    age  sepsis_index  \
0       0  83.14            -1
1       0  75.91            -1
2       0  45.82            -1
3       0  65.71            -1
4       1  28.09            -1
5       1  52.01            -1
6       1  64.24            -1
7       1  87.08            -1
8       1  27.92           248
9       0  76.71            -1


                                                   HR  \
0  [nan, 97.0, 89.0, 90.0, 103.0, 110.0, 108.0, 1…
1  [nan, 61.0, 64.0, 56.0, 66.0, 94.0, 58.0, 57.0…
2  [nan, 87.0, 93.0, 90.0, 89.0, 84.0, 82.0, 84.0…
3  [nan, 103.5, 108.0, 107.5, 113.0, 107.0, 93.0,…
4  [84.0, 80.0, 74.0, 73.0, 71.0, nan, 67.0, 75.0…
5  [109.0, 111.0, 107.0, 106.0, 103.0, 100.0, 96…
```

```
6  [nan, 155.5, 146.0, 154.0, 122.0, 124.0, 122.5…
7  [nan, 73.0, 73.0, 80.0, 77.0, 75.0, 75.0, 79.0…
8  [nan, 117.0, nan, nan, nan, nan, 120.0, 109.5,…
9  [82.0, 81.0, 80.0, 63.0, 70.0, 68.0, 69.0, 70…

                                               O2Sat  \
0  [nan, 95.0, 99.0, 95.0, 88.5, 91.0, 92.0, 90.5…
1  [nan, 99.0, 98.0, 100.0, 99.0, 100.0, 99.0, 10…
2  [nan, 96.0, 97.0, 95.0, 97.0, 97.0, 97.0, 97.0…
3  [nan, 97.0, 98.5, 96.5, 100.0, 100.0, 100.0, 1…
4  [97.5, 99.0, 97.0, 98.0, 97.0, nan, 97.0, 97.0…
5  [100.0, 100.0, 99.0, 99.0, 97.0, 99.0, 98.0, 9…
6  [nan, 94.5, 100.0, 97.0, 97.0, 95.0, 94.0, 93…
7  [nan, 99.0, 100.0, 95.0, 100.0, 100.0, 100.0, …
8  [nan, 99.0, nan, nan, nan, nan, 100.0, 98.5, 9…
9  [100.0, 100.0, 100.0, 100.0, 99.0, 96.0, 95.0,…

                                                 SBP  \
0  [nan, 98.0, 122.0, nan, 122.0, nan, 123.0, 93…
1  [nan, 124.0, 125.0, 123.0, 120.0, 194.0, 133.0…
2  [nan, 131.0, 130.0, 128.0, 137.0, 144.0, 140.0…
3  [nan, 107.5, 124.5, 117.5, 125.0, 116.0, 132.5…
4  [140.5, 150.0, 142.0, 144.0, 144.0, nan, 136.0…
5  [118.0, 127.0, 122.0, 101.0, 112.0, 122.0, 111…
6  [nan, 147.5, 133.0, 121.0, 126.0, 114.0, 112.2…
7  [nan, 100.0, 108.0, 89.0, 105.0, 108.5, 108.0,…
8  [nan, 116.0, nan, nan, nan, nan, 118.0, 106.0,…
9  [112.0, 103.0, 120.0, 99.0, 121.0, 112.0, 101…

                                                 DBP  \
0  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
1  [nan, 43.0, 41.0, 41.0, 43.0, 66.0, 43.0, 37.0…
2  [nan, nan, nan, nan, 50.0, 52.0, 50.0, 50.0, 5…
3  [nan, 52.0, 61.5, 58.5, 61.0, 52.0, 58.5, 44.0…
4  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
5  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
6  [nan, nan, 82.0, 73.0, 73.0, 65.0, 61.0, 53.5,…
7  [nan, 49.5, 56.0, 47.0, 51.0, 52.0, 51.0, 52.0…
8  [nan, 81.0, nan, nan, nan, nan, 64.0, 59.0, 58…
9  [63.0, 55.0, 65.0, 55.0, 65.0, 58.0, 47.0, 54…

                                                 MAP  \
0  [nan, 75.33, 86.0, nan, 91.33, nan, 77.0, 76.3…
1  [nan, 65.0, 64.0, 65.0, 67.0, 116.0, 68.0, 62…
2  [nan, 70.33, 69.33, 69.33, 62.67, 81.0, 77.0, …
3  [nan, 70.5, 82.0, 77.5, 80.0, 72.0, 84.0, 34.0…
4  [94.5, 99.0, 103.0, 99.0, nan, nan, 86.0, 87.0…
```

```
5  [86.0, 90.33, 78.0, 73.0, 78.0, 85.33, 75.67, …
6  [nan, 102.0, 87.0, 67.0, 89.0, 77.0, 70.5, 67…
7  [nan, 67.0, 74.0, 62.0, 70.0, 65.0, 69.0, 63.0…
8  [nan, 97.0, nan, nan, nan, nan, 84.0, 74.0, 72…
9  [79.5, 70.5, 83.0, 71.0, 85.0, 74.0, 63.0, 71…

                                              Resp  \
0  [nan, 19.0, 22.0, 30.0, 24.5, 22.0, 29.0, 29.0…
1  [nan, 17.5, 27.0, 9.0, 23.0, 14.0, 13.0, 18.0,…
2  [nan, 29.0, 40.0, 23.0, 26.0, 19.0, 26.0, 24.0…
3  [nan, 18.0, 19.5, 17.0, 26.0, 26.0, 15.0, 14.0…
4  [17.5, 18.0, 19.0, 17.0, 17.0, nan, 19.0, 16.0…
5  [18.5, 43.0, nan, nan, nan, nan, nan, nan, nan…
6  [nan, 33.0, 28.0, 22.0, 15.0, 19.0, 31.0, 26.0…
7  [nan, 16.5, 16.0, 22.0, 20.0, 19.0, 13.0, 16.0…
8  [nan, 20.0, nan, nan, nan, nan, 30.0, 27.0, 24…
9  [14.0, 14.0, 14.0, 15.0, 19.0, 17.0, 23.0, 21…

                                              FiO2  \
0  [nan, nan, nan, nan, 0.28, nan, nan, nan, nan,…
1  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
2  [nan, nan, nan, nan, nan, nan, 0.5, nan, nan, …
3  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
4  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
5  [0.4, 0.4, nan, nan, nan, nan, nan, nan, nan, …
6  [nan, 1.0, nan, 0.4, nan, nan, nan, nan, 0.4, …
7  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
8  [nan, nan, nan, nan, nan, nan, 1.0, nan, 1.0, …
9  [1.0, 0.5, 0.5, nan, nan, 0.4, 0.4, nan, nan, …

                                       SepsisLabel
0  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
1  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
2  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
3  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
4  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
5  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
6  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
7  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
8  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
9  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
```

```
[4]: diseased = patients[patients['sepsis_index'] != -1]
     non_diseased = patients[patients['sepsis_index'] == -1]
```

```
[5]: diseased.head()
```

```
[5]:    gender    age  sepsis_index  \
    8        1  27.92           248
    10       1  65.79            24
    14       0  58.54             5
    17       1  39.28           125
    21       0  77.26             9

                                                       HR  \
    8   [nan, 117.0, nan, nan, nan, nan, 120.0, 109.5,…
    10  [81.0, 82.0, 81.0, 82.0, 84.0, 84.0, 85.0, 85…
    14  [nan, 85.0, 89.5, 97.0, 90.0, 86.0, 87.0, 88.0…
    17  [nan, 104.0, 102.0, 90.0, 90.0, 87.0, 88.0, 89…
    21  [77.0, 75.0, 83.0, 80.0, 79.5, 85.0, 69.0, 66…

                                                     O2Sat  \
    8   [nan, 99.0, nan, nan, nan, nan, 100.0, 98.5, 9…
    10  [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100…
    14  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
    17  [nan, 98.0, 100.0, 100.0, 98.0, 98.0, 99.0, 99…
    21  [100.0, 99.5, 100.0, 99.0, 100.0, 100.0, 95.0,…

                                                      SBP  \
    8   [nan, 116.0, nan, nan, nan, nan, 118.0, 106.0,…
    10  [nan, 136.5, 114.0, 114.0, 117.0, 103.0, 123.0…
    14  [nan, 117.0, 122.5, 127.0, 110.0, 107.0, 112.0…
    17  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
    21  [126.0, 115.0, 129.0, 89.0, 143.0, 161.0, 91.0…

                                                      DBP  \
    8   [nan, 81.0, nan, nan, nan, nan, 64.0, 59.0, 58…
    10  [nan, 71.0, 61.0, 62.0, 61.0, 58.0, 66.0, 72.0…
    14  [nan, 74.0, 75.5, 79.0, 70.0, 69.0, 68.0, 72.0…
    17  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
    21  [53.0, 46.5, 50.0, 41.0, 52.5, 56.0, 43.0, 40…

                                                      MAP  \
    8   [nan, 97.0, nan, nan, nan, nan, 84.0, 74.0, 72…
    10  [nan, 90.0, 87.0, 79.0, 79.0, 73.5, 72.0, 96.0…
    14  [nan, 90.0, 93.0, 97.0, 85.0, 83.0, 80.0, 89.3…
    17  [nan, 83.0, 72.0, 72.0, 74.0, 72.0, 77.0, 76.0…
    21  [77.0, 69.5, 76.0, 57.0, 84.0, 92.0, 56.0, 61…

                                                     Resp  \
    8   [nan, 20.0, nan, nan, nan, nan, 30.0, 27.0, 24…
    10  [12.5, 12.0, 12.0, 12.0, 16.0, 20.0, 18.0, 14…
    14  [nan, 11.0, 9.5, 12.0, 14.0, 10.0, 14.0, 5.0, …
    17  [nan, 16.0, 24.0, 16.5, 14.0, 16.0, 15.5, 16.0…
```

```
21  [16.0, 16.0, 17.0, 18.0, 19.0, 18.0, 15.0, 20…

                                                FiO2  \
8   [nan, nan, nan, nan, nan, nan, 1.0, nan, 1.0, …
10  [0.6, nan, 0.4, nan, nan, nan, nan, nan, nan, …
14  [nan, nan, nan, 0.4, nan, nan, nan, nan, 0.4, …
17  [nan, nan, nan, 0.5, nan, nan, nan, 0.5, 0.4, …
21  [1.0, 0.5, nan, nan, 0.5, nan, nan, nan, nan, …

                                         SepsisLabel
8   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
10  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
14      [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
17  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
21  [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, …
```

[6]:
```python
import statistics, math

# for the non-diseased, take average of HR across the hours, average across␣
 ↪what data we have (i.e. if one NaN out of the 10, find average of the 9)
# for the diseased, use the first HR where it was sepsis

biomarkers = ['HR', 'O2Sat', 'SBP', 'DBP', 'MAP', 'Resp', 'FiO2']

def get_sepsis_value(row, col):
    idx = row['sepsis_index']
    values = row[col]

    # 1. Try to get the value at the sepsis index
    if idx < len(values) and not math.isnan(values[idx]):
        return values[idx]

    # 2. Fallback: Get the closest non-NaN value available
    clean_values = [v for v in values if not math.isnan(v)]
    if clean_values:
        # if they were labeled as sepsis ~50% through their stay, find the
        #   biomarker value ~50% through their stay
        estimated_placement_of_idx = math.floor(idx/len(row[col]))
        return clean_values[estimated_placement_of_idx]

    # 3. Final Fallback: If the entire list is NaNs, we'll ignore this if that␣
 ↪happens
    return None

for column in biomarkers:
    non_diseased[column + "single_pt"] = non_diseased[column].apply(lambda x:␣
↪statistics.mean([x_i for x_i in x if not math.isnan(x_i)]) if [x_i for x_i␣
↪in x if not math.isnan(x_i)] else None)
    # use the index of the first classification of sepsis and if there isn't a␣
↪value at that index (i.e. column data wasn't collected in that hour, take␣
↪the last one in the list)
    diseased[column + "single_pt"] = diseased.apply(lambda x:␣
↪get_sepsis_value(x, column), axis=1)
```

```
/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:28
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_39307/253624440.py:30
: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[7]:
```
diseased
```

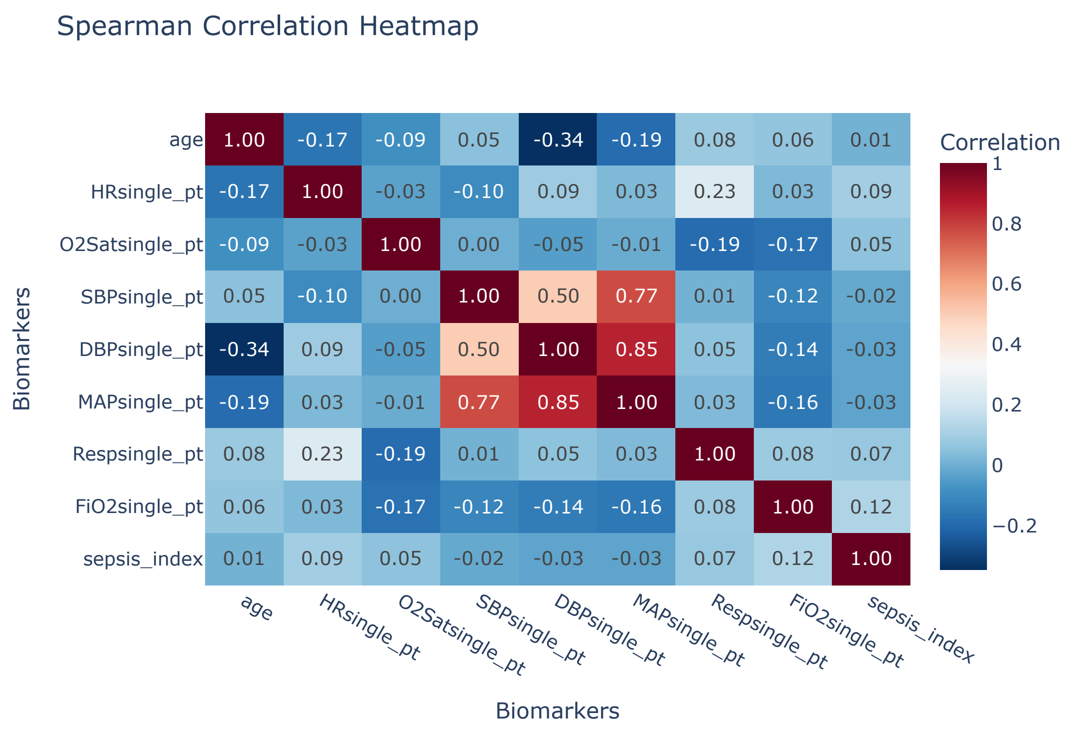
[7]:
```
       gender    age  sepsis_index  \
8           1  27.92           248
10          1  65.79            24
14          0  58.54             5
17          1  39.28           125
21          0  77.26             9
…         …     …            …
40223       1  69.00             2
40245       0  76.00            41
40252       0  71.00            58
40273       0  69.00           101
40294       0  56.00            31

                                                      HR  \
8      [nan, 117.0, nan, nan, nan, nan, 120.0, 109.5,…
10     [81.0, 82.0, 81.0, 82.0, 84.0, 84.0, 85.0, 85…
14     [nan, 85.0, 89.5, 97.0, 90.0, 86.0, 87.0, 88.0…
17     [nan, 104.0, 102.0, 90.0, 90.0, 87.0, 88.0, 89…
21     [77.0, 75.0, 83.0, 80.0, 79.5, 85.0, 69.0, 66…
…                                                  …
40223  [nan, 106.5, 105.0, 122.5, 104.0, 100.5, 107.0…
40245  [91.0, 96.5, 101.0, 104.0, 103.0, 90.0, 85.0, …
40252  [100.0, nan, nan, 102.0, 102.0, nan, nan, nan,…
40273  [90.0, 96.0, 90.0, 93.0, 86.0, 85.0, 79.0, 86…
40294  [nan, 77.0, 80.0, nan, 77.0, 79.0, 83.0, 83.0,…

                                                   O2Sat  \
8      [nan, 99.0, nan, nan, nan, nan, 100.0, 98.5, 9…
10     [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100…
14     [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
17     [nan, 98.0, 100.0, 100.0, 98.0, 98.0, 99.0, 99…
21     [100.0, 99.5, 100.0, 99.0, 100.0, 100.0, 95.0,…
…                                                  …
40223  [nan, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0…
40245  [96.5, 99.0, 100.0, 98.0, 99.0, 100.0, 96.0, 9…
40252  [100.0, nan, nan, 98.0, 98.0, nan, nan, nan, n…
40273  [100.0, 100.0, 100.0, 100.0, 98.0, 99.0, 97.0,…
40294  [nan, 98.0, 97.0, nan, 98.0, nan, nan, nan, na…
```

```
                                                       SBP  \
8      [nan, 116.0, nan, nan, nan, nan, 118.0, 106.0,…
10     [nan, 136.5, 114.0, 114.0, 117.0, 103.0, 123.0…
14     [nan, 117.0, 122.5, 127.0, 110.0, 107.0, 112.0…
17     [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
21     [126.0, 115.0, 129.0, 89.0, 143.0, 161.0, 91.0…
…                                                   …
40223  [nan, 148.0, 162.5, 164.0, 139.5, 146.0, 158.0…
40245  [113.0, 91.5, 116.0, 100.0, 128.0, 103.0, 127…
40252  [102.0, nan, nan, 95.0, 95.0, nan, nan, nan, n…
40273  [97.0, 129.0, 132.0, 128.0, 107.0, 120.0, 127…
40294  [nan, 120.0, 108.0, nan, 132.0, 122.0, 124.0, …

                                                       DBP  \
8      [nan, 81.0, nan, nan, nan, nan, 64.0, 59.0, 58…
10     [nan, 71.0, 61.0, 62.0, 61.0, 58.0, 66.0, 72.0…
14     [nan, 74.0, 75.5, 79.0, 70.0, 69.0, 68.0, 72.0…
17     [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
21     [53.0, 46.5, 50.0, 41.0, 52.5, 56.0, 43.0, 40…
…                                                   …
40223  [nan, 79.5, 94.0, 91.0, 71.5, 74.0, 94.0, 83.0…
40245  [69.0, 61.5, 73.0, 65.0, 75.0, 54.0, 64.5, 64…
40252  [64.0, nan, nan, 64.0, 64.0, nan, nan, nan, na…
40273  [37.0, 46.0, 42.0, 43.0, 47.0, 51.0, 51.0, 56…
40294  [nan, 61.0, 58.0, nan, 67.0, 75.0, 77.0, 73.0,…

                                                       MAP  \
8      [nan, 97.0, nan, nan, nan, nan, 84.0, 74.0, 72…
10     [nan, 90.0, 87.0, 79.0, 79.0, 73.5, 72.0, 96.0…
14     [nan, 90.0, 93.0, 97.0, 85.0, 83.0, 80.0, 89.3…
17     [nan, 83.0, 72.0, 72.0, 74.0, 72.0, 77.0, 76.0…
21     [77.0, 69.5, 76.0, 57.0, 84.0, 92.0, 56.0, 61…
…                                                   …
40223  [nan, 104.5, 121.5, 124.0, 97.5, 104.0, 117.0,…
40245  [83.0, 69.5, 87.0, 76.0, 91.0, 70.0, 82.5, 81…
40252  [nan, nan, nan, 74.0, 74.0, nan, nan, nan, nan…
40273  [64.0, 79.0, 76.0, 78.0, 75.0, 81.0, 86.0, 92…
40294  [nan, 85.0, 77.0, nan, 93.0, 94.0, 95.0, 93.0,…

                                                      Resp  \
8      [nan, 20.0, nan, nan, nan, nan, 30.0, 27.0, 24…
10     [12.5, 12.0, 12.0, 12.0, 16.0, 20.0, 18.0, 14…
14     [nan, 11.0, 9.5, 12.0, 14.0, 10.0, 14.0, 5.0, …
17     [nan, 16.0, 24.0, 16.5, 14.0, 16.0, 15.5, 16.0…
21     [16.0, 16.0, 17.0, 18.0, 19.0, 18.0, 15.0, 20…
…                                                   …
```

```
40223  [nan, 28.0, 28.0, 31.0, 20.0, 26.5, 20.0, 20.0…
40245  [nan, 12.0, 18.0, 22.0, 20.0, 22.0, 22.0, 22.0…
40252  [20.0, nan, nan, 20.0, 20.0, nan, nan, nan, na…
40273  [nan, nan, nan, nan, nan, 20.0, 20.0, 20.0, 20…
40294  [nan, 19.0, 18.0, nan, 24.0, 19.0, 23.0, nan, …

                                                    FiO2  \
8      [nan, nan, nan, nan, nan, nan, 1.0, nan, 1.0, …
10     [0.6, nan, 0.4, nan, nan, nan, nan, nan, nan, …
14     [nan, nan, nan, 0.4, nan, nan, nan, nan, 0.4, …
17     [nan, nan, nan, 0.5, nan, nan, nan, 0.5, 0.4, …
21     [1.0, 0.5, nan, nan, 0.5, nan, nan, nan, nan, …
…                                                    …
40223  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
40245  [1.0, 1.0, 1.0, nan, nan, 0.8, nan, nan, 0.8, …
40252  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …
40273  [1.0, 0.8, nan, 0.8, nan, nan, 0.8, nan, nan, …
40294  [nan, nan, nan, nan, nan, nan, nan, nan, nan, …

                                             SepsisLabel  HRsingle_pt  \
8      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …        119.0
10     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …         98.0
14         [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]         86.0
17     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …        105.0
21     [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, …         73.0
…                                                    …          …
40223               [0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]        105.0
40245  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …         84.0
40252  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …        100.0
40273  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …         82.0
40294  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …         92.0

       O2Satsingle_pt  SBPsingle_pt  DBPsingle_pt  MAPsingle_pt  \
8               100.0         140.0          85.0         106.0
10              100.0         147.0          59.0          86.0
14                NaN         107.0          69.0          83.0
17               96.0         166.0          66.5          75.0
21               97.0         117.0          44.0          65.0
…                 …            …             …            …
40223           100.0         162.5          94.0         121.5
40245            97.0          75.0          66.0          71.0
40252           100.0         102.0          64.0          74.0
40273            96.0         156.0          47.0          70.0
40294            95.0         115.0          57.0          81.0

       Respsingle_pt  FiO2single_pt
8               26.5           0.35
```

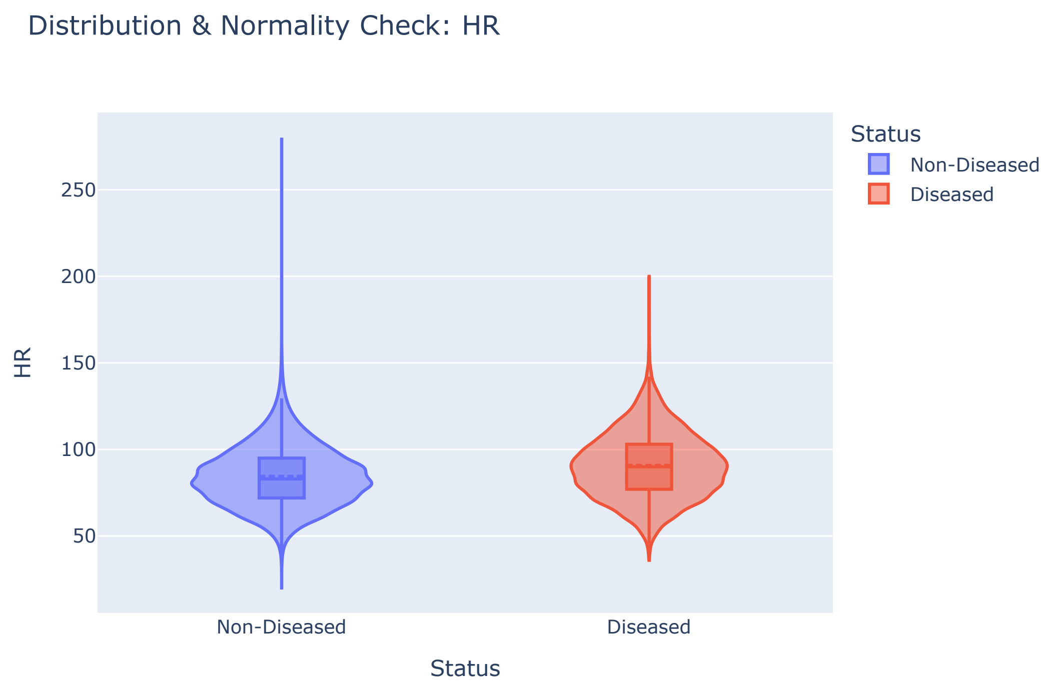
```
10               12.5           0.60
14               10.0           0.40
17               20.0           0.50
21               14.0           0.50
…              …             …
40223            28.0            NaN
40245            26.0           1.00
40252            20.0            NaN
40273            20.0           1.00
40294            22.0            NaN

[2932 rows x 18 columns]
```

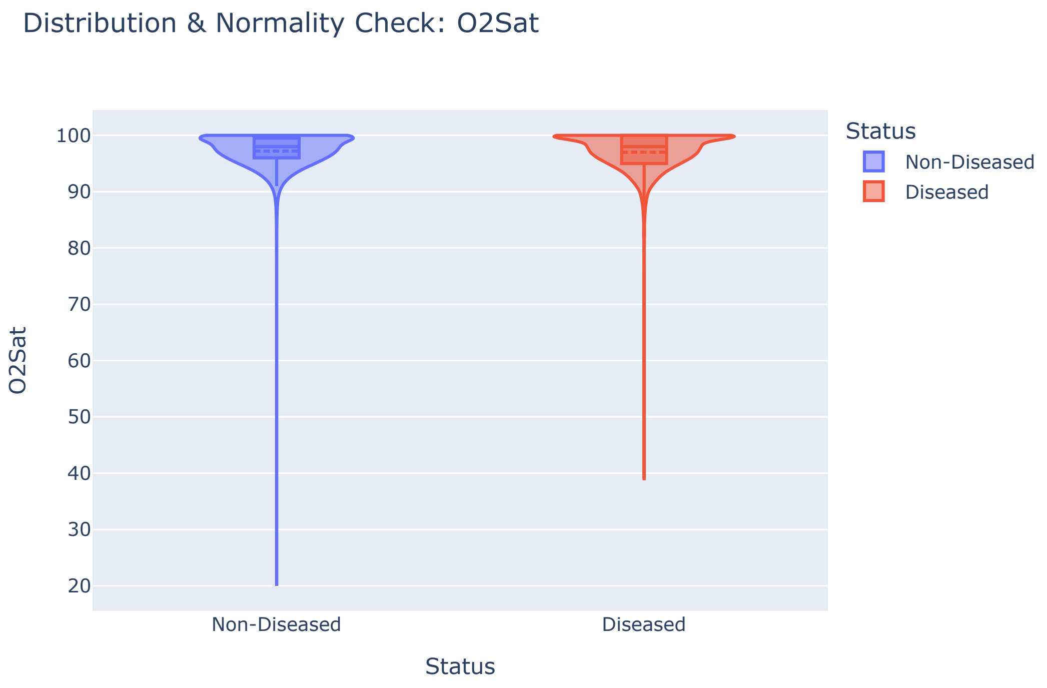
[8]:
```
import plotly.express as px

tmp = pd.concat([diseased, non_diseased], ignore_index=True, sort=False)
tmp_cols = ['age', 'HRsingle_pt', 'O2Satsingle_pt', 'SBPsingle_pt',␣
 ↪'DBPsingle_pt', 'MAPsingle_pt', 'Respsingle_pt', 'FiO2single_pt',␣
 ↪'sepsis_index']

# want to see what correlates with the 'sepsis_index' value the most.
fig = px.imshow(
    tmp[tmp_cols].corr(method='spearman'),
    text_auto=".2f",                # Shows the values in the cells
    color_continuous_scale='RdBu_r', # Red-Blue scale (Red = Positive, Blue =␣
 ↪Negative)
    aspect="auto",
    title="Spearman Correlation Heatmap",
    labels=dict(color="Correlation")
)

fig.update_layout(
    xaxis_title="Biomarkers",
    yaxis_title="Biomarkers",
    width=700,
    height=700
)

fig.show()
```

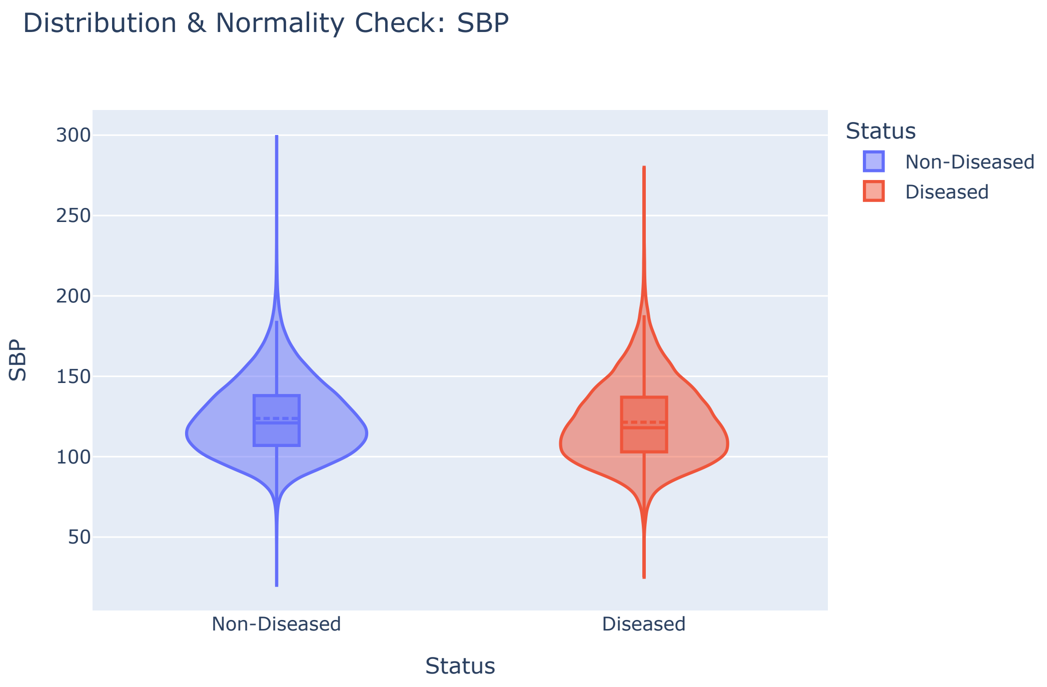


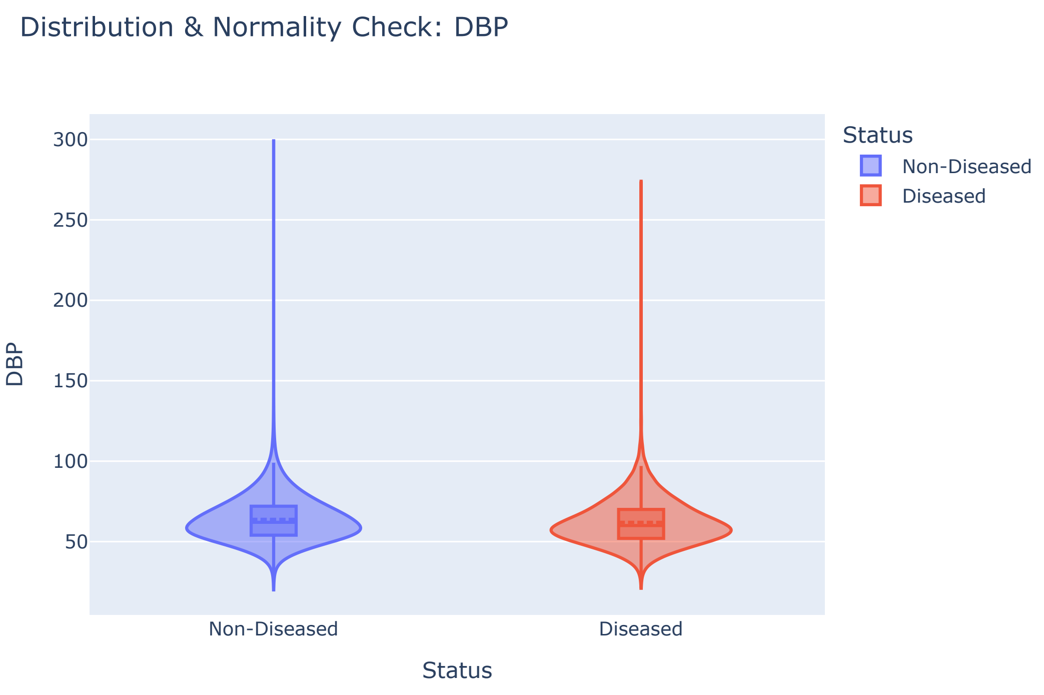
	age	HRsingle_pt	O2Satsingle_pt	SBPsingle_pt	DBPsingle_pt	MAPsingle_pt	Respsingle_pt	FiO2single_pt	sepsis_index
age	1.00	-0.17	-0.09	0.05	-0.34	-0.19	0.08	0.06	0.01
HRsingle_pt	-0.17	1.00	-0.03	-0.10	0.09	0.03	0.23	0.03	0.09
O2Satsingle_pt	-0.09	-0.03	1.00	0.00	-0.05	-0.01	-0.19	-0.17	0.05
SBPsingle_pt	0.05	-0.10	0.00	1.00	0.50	0.77	0.01	-0.12	-0.02
DBPsingle_pt	-0.34	0.09	-0.05	0.50	1.00	0.85	0.05	-0.14	-0.03
MAPsingle_pt	-0.19	0.03	-0.01	0.77	0.85	1.00	0.03	-0.16	-0.03
Respsingle_pt	0.08	0.23	-0.19	0.01	0.05	0.03	1.00	0.08	0.07
FiO2single_pt	0.06	0.03	-0.17	-0.12	-0.14	-0.16	0.08	1.00	0.12
sepsis_index	0.01	0.09	0.05	-0.02	-0.03	-0.03	0.07	0.12	1.00

[9]:
```python
import pandas as pd
import plotly.express as px

df = tmp

# 1. Define columns
# We only need to explode the specific metric and the SepsisLabel together
ts_metrics = ['HR', 'O2Sat', 'SBP', 'DBP', 'MAP', 'Resp', 'FiO2']

# 2. Loop through each metric to process and plot individually
for metric in ts_metrics:
    # Explode only the current metric and the label to save memory
    cols_to_use = [metric, 'SepsisLabel']

    # Create a temporary flat dataframe for just this metric
    temp_df = df.explode(cols_to_use)

    # Convert to numeric (only two columns, much faster)
    for col in cols_to_use:
        temp_df[col] = pd.to_numeric(temp_df[col], errors='coerce')
    # Drop NaNs for this specific metric
    temp_df = temp_df.dropna(subset=[metric])

    # Map Status
    temp_df['Status'] = temp_df['SepsisLabel'].map({1: 'Diseased', 0:␣
 ↪'Non-Diseased'})

    # 3. Generate Individual Plot
    fig = px.violin(
        temp_df,
        x="Status",
        y=metric,
        color="Status",
        box=True,
        points=False,
        title=f"Distribution & Normality Check: {metric}",
        category_orders={"Status": ["Non-Diseased", "Diseased"]}
    )

    # Performance and Normality visual aids
    fig.update_traces(meanline_visible=True, spanmode='hard')
    fig.show()

# 4. Handle 'age' separately (since it's not a list/time-series in your data)

# We create a column for status without duplicating the whole 'patients'␣
 ↪dataframe
patients['Status'] = 'Non-Diseased'
patients.loc[patients['sepsis_index'] != -1, 'Status'] = 'Diseased'

# 2. Create the side-by-side violin plot
fig_age = px.violin(
    patients,
    x="Status",
    y="age",
    color="Status",
    box=True,           # Adds the box plot inside to see quartiles
    points="outliers",  # Shows individual dots only for extreme values
    title="Age Distribution: Diseased vs. Non-Diseased",
    category_orders={"Status": ["Non-Diseased", "Diseased"]}, # Sets the order
    color_discrete_map={"Diseased": "#EF553B", "Non-Diseased": "#636EFA"} #␣
 ↪Explicit colors
)

# 3. Add visual aids for Normality
fig_age.update_traces(meanline_visible=True) # Shows the Mean line
fig_age.update_layout(showlegend=False)      # Legend is redundant with X-axis␣
↪labels

fig_age.show()
```

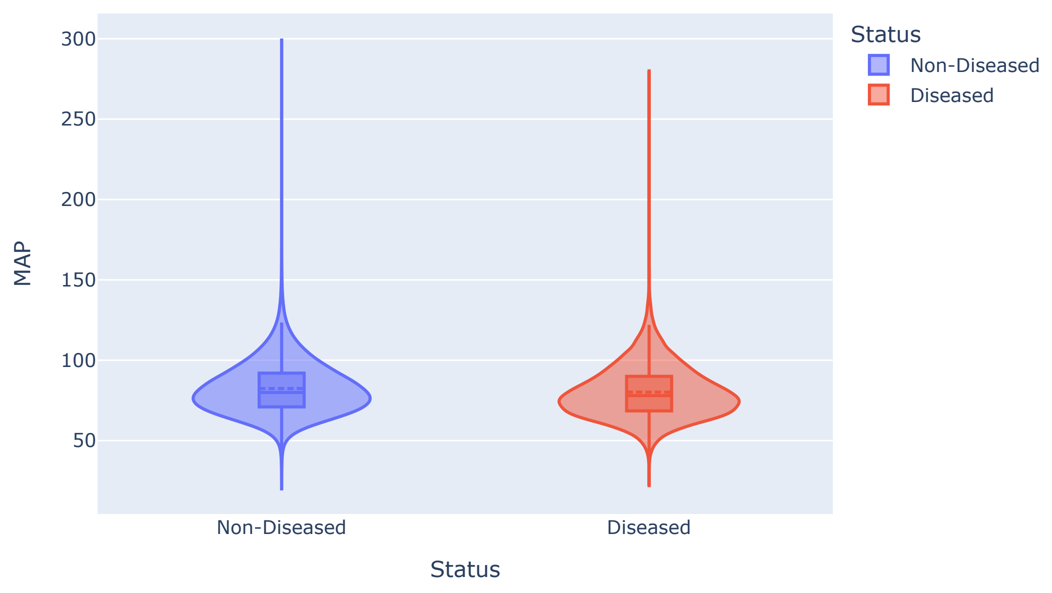



Distribution & Normality Check: O2Sat
O2Sat
100
90
80
70
60
50
40
30
20
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased

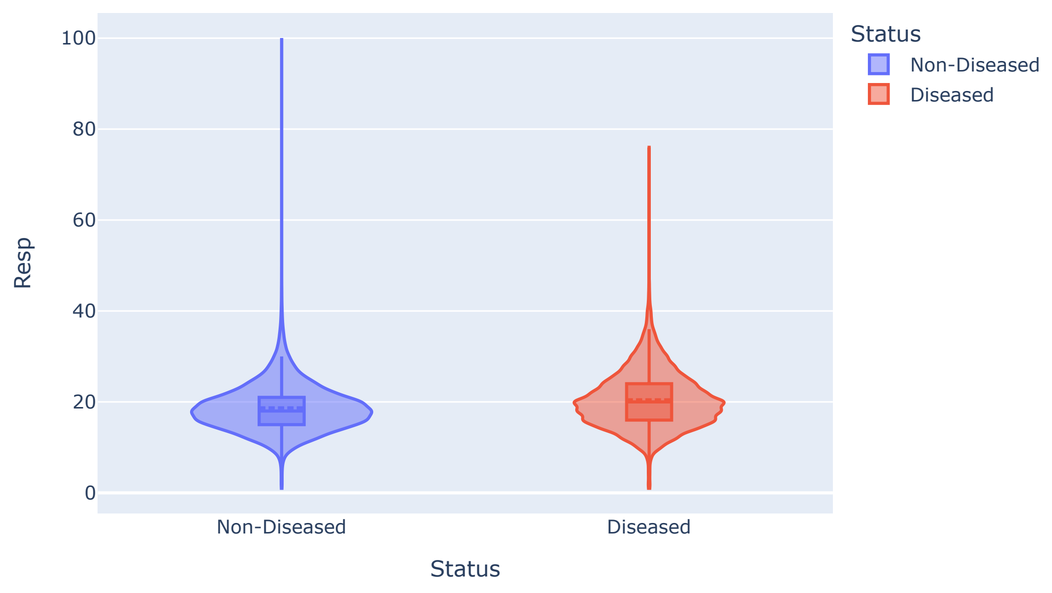

Distribution & Normality Check: SBP
SBP
300
250
200
150
100
50
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased

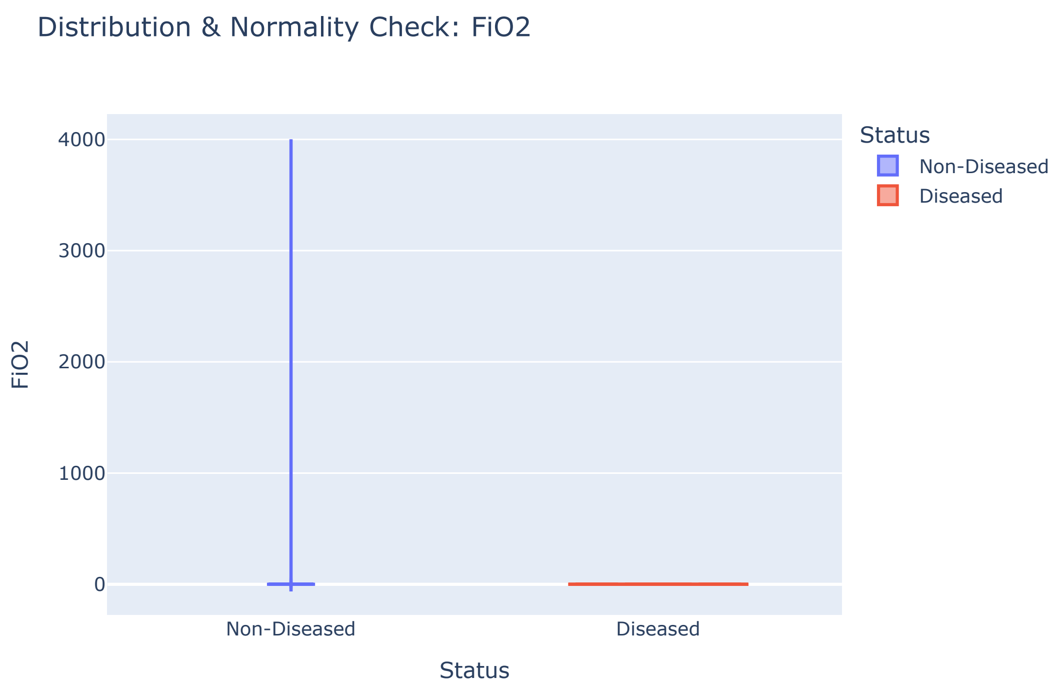

Distribution & Normality Check: DBP
DBP
300
250
200
150
100
50
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased

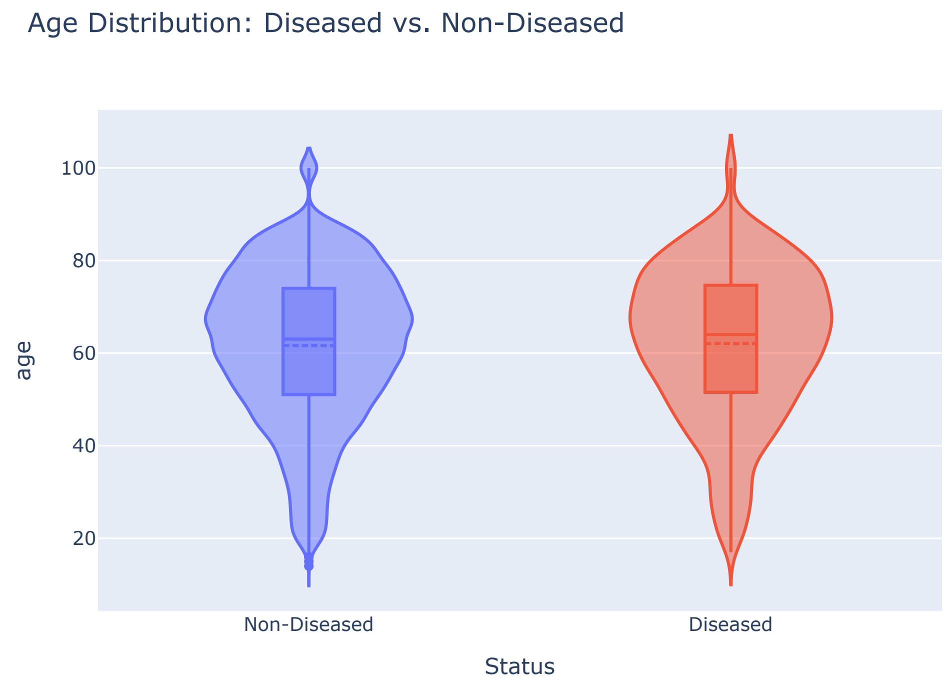
Distribution & Normality Check: MAP



Distribution & Normality Check: Resp




Distribution & Normality Check: FiO2
FiO2
4000
3000
2000
1000
0
Non-Diseased
Diseased
Status
Status
Non-Diseased
Diseased



The SBP, DBP, HR, MAP, Resp, and age look relatively normally distributed (box around the middle, symmetrical-ish tails).

[10]:
```
cols = ['age', 'HRsingle_pt', 'SBPsingle_pt', 'DBPsingle_pt', 'MAPsingle_pt',␣
  ↪'Respsingle_pt']

diseased[cols].head()
```

[10]:
```
      age  HRsingle_pt  SBPsingle_pt  DBPsingle_pt  MAPsingle_pt  \
8   27.92        119.0         140.0          85.0         106.0
10  65.79         98.0         147.0          59.0          86.0
14  58.54         86.0         107.0          69.0          83.0
17  39.28        105.0         166.0          66.5          75.0
21  77.26         73.0         117.0          44.0          65.0

    Respsingle_pt
8            26.5
10           12.5
14           10.0
17           20.0
21           14.0
```

[11]:
```
## Preprocessing done, moving to the fun stuff

diseased = diseased[cols].dropna()
non_diseased = non_diseased[cols].dropna()

print(f"Number of diseased after last drop, {len(diseased)}")
print(f"Number of non-diseased after last drop, {len(non_diseased)}")

mu_d = []
mu_n = []

diseased_summary = diseased.describe()
nondiseased_summary = non_diseased.describe()

for col in cols:
    mu_d.append(diseased_summary[col]['mean'])
    mu_n.append(nondiseased_summary[col]['mean'])

mu_d = np.array(mu_d)
mu_n = np.array(mu_n)
mu_d, mu_n
```

```
Number of diseased after last drop, 2425
Number of non-diseased after last drop, 30376
```

[11]:
```
(array([ 61.8262,  90.0252, 124.1587,  63.3364,  82.8291,  19.7478]),
 array([ 61.8538,  83.5392, 123.8734,  64.2153,  83.5295,  18.3454]))
```

[12]:
```
cov_n = np.cov(non_diseased.T)
cov_d = np.cov(diseased.T)

cov_n, cov_d
```

[12]:
```
(array([[260.7885, -39.1974,   9.223 , -59.1921, -39.1756,   1.7823],
        [-39.1974, 203.2584, -32.0765,  15.166 ,  -1.2471,  10.645 ],
        [  9.223 , -32.0765, 318.498 , 102.6381, 178.743 ,   0.0734],
        [-59.1921,  15.166 , 102.6381, 119.6596, 119.5803,   1.7023],
        [-39.1756,  -1.2471, 178.743 , 119.5803, 161.0662,   1.6176],
        [  1.7823,  10.645 ,   0.0734,   1.7023,   1.6176,  10.0297]]),
 array([[273.3461, -52.3647,   3.0679, -64.2258, -43.062 ,   4.9493],
        [-52.3647, 362.1953,  -8.0497,  48.8267,  33.8591,  24.4634],
        [  3.0679,  -8.0497, 635.1111, 193.0212, 314.953 ,   7.16  ],
        [-64.2258,  48.8267, 193.0212, 205.009 , 194.9137,   7.3786],
        [-43.062 ,  33.8591, 314.953 , 194.9137, 298.3187,  11.0093],
        [  4.9493,  24.4634,   7.16  ,   7.3786,  11.0093,  35.8621]]))
```

[13]:
```
cov = (cov_d + cov_n)/2
cov_inverse = np.linalg.inv(cov)
```

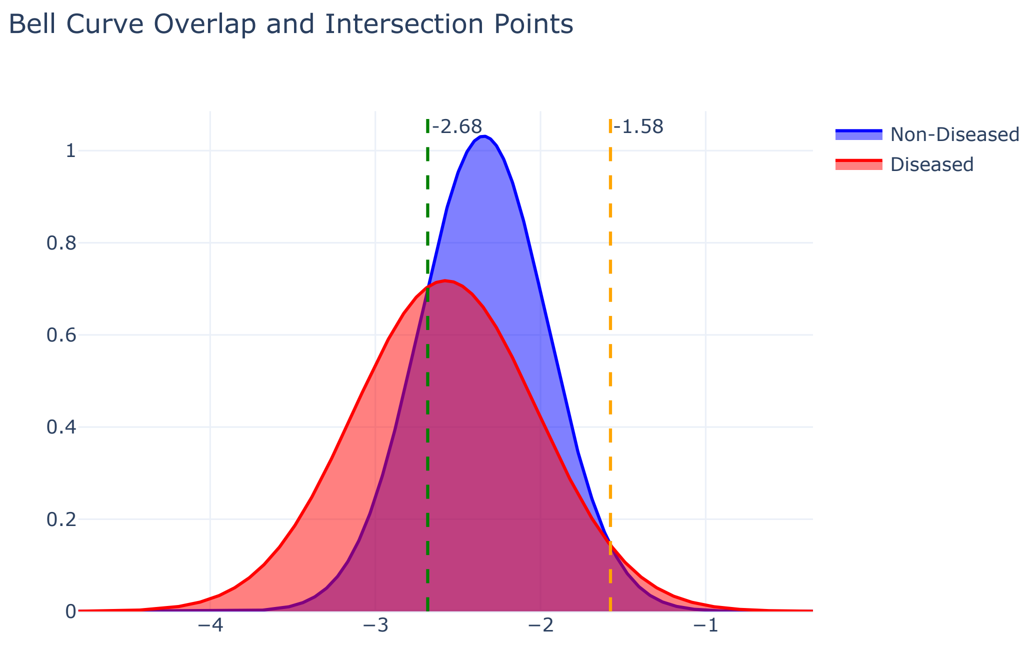
```
cov, cov_inverse
```

```
[13]: (array([[267.0673, -45.7811,   6.1454, -61.709 , -41.1188,   3.3658],
              [-45.7811, 282.7269, -20.0631,  31.9963,  16.306 ,  17.5542],
              [  6.1454, -20.0631, 476.8045, 147.8297, 246.848 ,   3.6167],
              [-61.709 ,  31.9963, 147.8297, 162.3343, 157.247 ,   4.5405],
              [-41.1188,  16.306 , 246.848 , 157.247 , 229.6925,   6.3135],
              [  3.3658,  17.5542,   3.6167,   4.5405,   6.3135,  22.9459]]),
        array([[ 0.0044,  0.0005, -0.0008,  0.0022,  0.0001, -0.0014],
              [ 0.0005,  0.0039,  0.0004, -0.001 ,  0.0001, -0.003 ],
              [-0.0008,  0.0004,  0.0051,  0.0014, -0.0066,  0.0005],
              [ 0.0022, -0.001 ,  0.0014,  0.0206, -0.0151,  0.0003],
              [ 0.0001,  0.0001, -0.0066, -0.0151,  0.0218, -0.0021],
              [-0.0014, -0.003 ,  0.0005,  0.0003, -0.0021,  0.0465]]))
```

```
[14]: a = (mu_n - mu_d).T @ cov_inverse
      a
```

```
[14]: array([ 0.0011, -0.0223, -0.0085,  0.0134,  0.0061, -0.0472])
```

```
[15]: import numpy as np
      import plotly.graph_objects as go
      from scipy.stats import norm

      # 1. Extract scalar parameters
      m_u = (a.T @ mu_n).item()
      s_u = np.sqrt((a.T @ cov_n @ a).item())

      m_v = (a.T @ mu_d).item()
      s_v = np.sqrt((a.T @ cov_d @ a).item())

      # 2. Solve for Intersection Points (Quadratic Equation: Ax^2 + Bx + C = 0)
      # Derived from setting the two PDF equations equal and taking the log
      A = 1/(2*s_u**2) - 1/(2*s_v**2)
      B = m_v/(s_v**2) - m_u/(s_u**2)
      C = m_u**2/(2*s_u**2) - m_v**2/(2*s_v**2) - np.log(s_v/s_u)

      intersections = np.roots([A, B, C])

      # 3. Create Plotly Figure
      x_start = min(m_u - 4*s_u, m_v - 4*s_v)
      x_end = max(m_u + 4*s_u, m_v + 4*s_v)
      x_range = np.linspace(x_start, x_end, 1000)

      fig = go.Figure()

      fig.add_trace(go.Scatter(x=x_range, y=norm.pdf(x_range, m_u, s_u),
                               name='Non-Diseased', fill='tozeroy',␣
 ↪line_color='blue'))
      fig.add_trace(go.Scatter(x=x_range, y=norm.pdf(x_range, m_v, s_v),
                               name='Diseased', fill='tozeroy', line_color='red'))

      # 4. Add the two intersection lines
      colors = ['green', 'orange']
      for i, pt in enumerate(intersections):
          # Only plot if the point is within a reasonable range of our data
          if x_start < pt < x_end:
              fig.add_vline(x=pt, line_dash="dash", line_color=colors[i],
                            annotation_text=f"{pt:.2f}",
                            )

      fig.update_layout(title="Bell Curve Overlap and Intersection Points",␣
      ↪template="plotly_white")
      fig.show()
```



There is significant overlap, which means we have a high chance of misclassifying. For arguments sake, let's say we want to increase sensitivity (to make sure we catch sepsis) at the cost of having more false alarms (i.e. decreasing specificity).

```
[16]: for cutoff in list(range(-5, 2, 1)):
          sensitivity = 1 - norm.cdf(cutoff, m_v, s_v)
          specificity = norm.cdf(cutoff, m_u, s_u)

          print(f"At cutoff {cutoff}:")
          print(f"\tSensitivity (Catch Rate): {sensitivity*100:.2f}%")
          print(f"\tSpecificity (Avoid False Alarms): {specificity*100:.2f}%")
          print("\tRule: If the value X is less than the cutoff, then it's classified␣
       ↪as Sepsis.")
```

```
At cutoff -5:
        Sensitivity (Catch Rate): 100.00%
        Specificity (Avoid False Alarms): 0.00%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff -4:
        Sensitivity (Catch Rate): 99.48%
        Specificity (Avoid False Alarms): 0.00%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff -3:
        Sensitivity (Catch Rate): 77.81%
        Specificity (Avoid False Alarms): 4.52%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff -2:
        Sensitivity (Catch Rate): 15.08%
        Specificity (Avoid False Alarms): 81.37%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff -1:
        Sensitivity (Catch Rate): 0.23%
        Specificity (Avoid False Alarms): 99.97%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff 0:
        Sensitivity (Catch Rate): 0.00%
        Specificity (Avoid False Alarms): 100.00%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
At cutoff 1:
        Sensitivity (Catch Rate): 0.00%
        Specificity (Avoid False Alarms): 100.00%
        Rule: If the value X is less than the cutoff, then it's classified as
Sepsis.
```

[17]:
```
cutoff = -3.25
sensitivity = 1 - norm.cdf(cutoff, m_v, s_v)
specificity = norm.cdf(cutoff, m_u, s_u)
sensitivity, specificity
```

[17]:
```
(np.float64(0.8879139716346451), np.float64(0.00964951316295226))
```

To apply these weights back, let's say Steve comes to the ER with the following biomarker values

$$\begin{bmatrix} age = 65 \\ HR = 100 \\ SBP = 122 \\ DBP = 61 \\ MAP = 82 \\ Resp = 15 \end{bmatrix}$$

[18]:
```
steve=np.array([
    [65],
    [100],
    [122],
    [61],
    [82],
    [15],
]).T

steve@a
```

[18]:
```
array([-2.587])
```

We'd say Steve does not have sepsis because his score is -2.6 to be classified as sepsis he'd need a score smaller than -3.25.