import pandas as pd  
##%matplotlib inline  
import matplotlib  
import numpy as np  
import matplotlib.pyplot as plt  
from plotly import graph\_objects  
import streamlit as st

## My Info

st.title('Computer Science Capstone — C964')  
st.header('Student – Mark Nefzger')  
st.header('Student ID: 001411596')

DeltaGenerator()

## Version Info

pd\_ver = pd.\_\_version\_\_  
st.write("Pandas version: ", pd\_ver)

np\_ver = np.\_\_version\_\_  
st.write("Numpy version: ", np\_ver)

st\_ver = st.\_\_version\_\_  
st.write("Streamlit version: ", st\_ver)

plt\_ver = matplotlib.\_\_version\_\_  
st.write("Matplotlib version: ", plt\_ver)

## Import data

# Import Data  
health\_data = pd.read\_csv("New Data/oura\_2019-01-01\_2023-09-09\_trends\_Shifted.csv")

## View data

st.header('Imported Data')

DeltaGenerator()

health\_data

date Sleep Score ... HRV Balance Score Recovery Index Score  
0 NaN NaN ... NaN NaN  
1 1/13/2021 83.0 ... NaN 99.0  
2 1/14/2021 76.0 ... NaN 69.0  
3 1/15/2021 78.0 ... NaN 96.0  
4 1/16/2021 69.0 ... NaN 51.0  
.. ... ... ... ... ...  
966 7/22/2023 NaN ... NaN NaN  
967 7/29/2023 NaN ... NaN NaN  
968 7/31/2023 NaN ... NaN NaN  
969 8/7/2023 NaN ... NaN NaN  
970 9/5/2023 NaN ... NaN NaN  
  
[971 rows x 54 columns]

health\_data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 971 entries, 0 to 970  
Data columns (total 54 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 date 970 non-null object   
 1 Sleep Score 898 non-null float64  
 2 Total Sleep Score 898 non-null float64  
 3 REM Sleep Score 898 non-null float64  
 4 Deep Sleep Score 898 non-null float64  
 5 Sleep Efficiency Score 898 non-null float64  
 6 Restfulness Score 898 non-null float64  
 7 Sleep Latency Score 898 non-null float64  
 8 Sleep Timin Score 898 non-null float64  
 9 Total Sleep Duration 896 non-null float64  
 10 Total Bedtime 896 non-null float64  
 11 Awake Time 896 non-null float64  
 12 REM Sleep Duration 896 non-null float64  
 13 Light Sleep Duration 896 non-null float64  
 14 Deep Sleep Duration 896 non-null float64  
 15 Restless Sleep 896 non-null float64  
 16 Sleep Efficiency 896 non-null float64  
 17 Sleep Latency 896 non-null float64  
 18 Sleep Timing 896 non-null float64  
 19 Bedtime Start 896 non-null object   
 20 Bedtime End 896 non-null object   
 21 Average Resting Heart Rate 895 non-null float64  
 22 Lowest Resting Heart Rate 895 non-null float64  
 23 Average HRV 895 non-null float64  
 24 Temperature Deviation (°C) 883 non-null float64  
 25 Temperature Trend Deviation 866 non-null float64  
 26 Respiratory Rate 896 non-null float64  
 27 Activity Score 970 non-null float64  
 28 Stay Active Score 968 non-null float64  
 29 Move Every Hour Score 968 non-null float64  
 30 Meet Daily Targets Score 968 non-null float64  
 31 Training Frequency Score 968 non-null float64  
 32 Training Volume Score 968 non-null float64  
 33 Activity Burn 970 non-null float64  
 34 Total Burn 970 non-null float64  
 35 Steps 970 non-null float64  
 36 Equivalent Walking Distance 970 non-null float64  
 37 Inactive Time 970 non-null float64  
 38 Rest Time 970 non-null float64  
 39 Low Activity Time 970 non-null float64  
 40 Medium Activity Time 970 non-null float64  
 41 High Activity Time 970 non-null float64  
 42 Non-wear Time 970 non-null float64  
 43 Average MET 970 non-null float64  
 44 Long Periods of Inactivity 970 non-null float64  
 45 Readiness Score 898 non-null float64  
 46 Previous Night Score 888 non-null float64  
 47 Sleep Balance Score 892 non-null float64  
 48 Previous Day Activity Score 881 non-null float64  
 49 Activity Balance Score 887 non-null float64  
 50 Temperature Score 895 non-null float64  
 51 Resting Heart Rate Score 898 non-null float64  
 52 HRV Balance Score 887 non-null float64  
 53 Recovery Index Score 898 non-null float64  
dtypes: float64(51), object(3)  
memory usage: 409.8+ KB

## Describe Data

health\_data.describe()

Sleep Score Total Sleep Score ... HRV Balance Score Recovery Index Score  
count 898.000000 898.000000 ... 887.000000 898.000000  
mean 81.312918 78.945434 ... 73.838782 63.732739  
std 9.733865 17.745517 ... 15.558814 27.610886  
min 21.000000 1.000000 ... 22.000000 0.000000  
25% 76.250000 68.000000 ... 64.000000 41.000000  
50% 83.000000 82.000000 ... 77.000000 62.000000  
75% 88.000000 95.000000 ... 87.000000 92.000000  
max 97.000000 100.000000 ... 100.000000 100.000000  
  
[8 rows x 51 columns]

## Set up dataframe

df = pd.DataFrame(health\_data)

## Convert Sleep Duration and Rest Time to hours

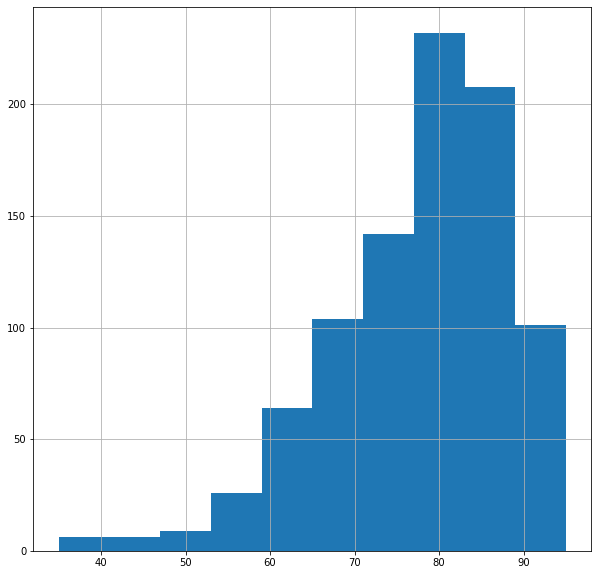
df["Total Sleep Duration"] = df["Total Sleep Duration"] / 3600  
df["Rest Time"] = df["Rest Time"] / 3600

pd.crosstab(df["Total Sleep Duration"] > 7, df["Readiness Score"] >85)

Readiness Score False True   
Total Sleep Duration   
False 494 47  
True 279 151

(df["Readiness Score"].hist(figsize=(10, 10)))

<Axes: >



## Manipulating Data

df.dropna(inplace=True)

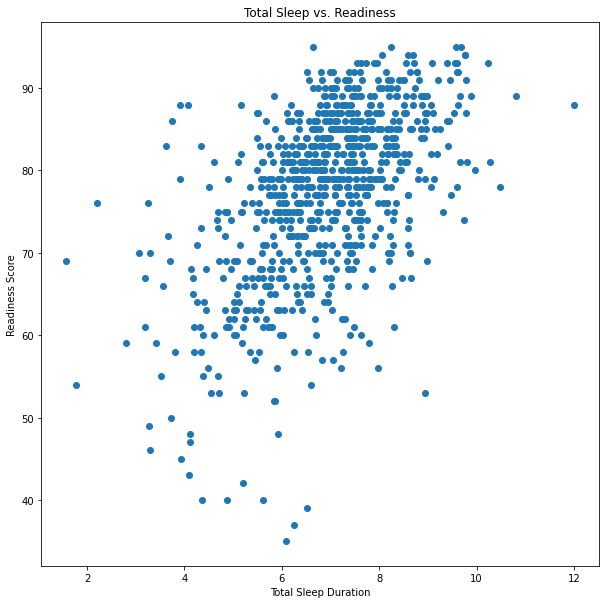
# Randomize data 1 = 100%  
df.sample(frac=1)

date Sleep Score ... HRV Balance Score Recovery Index Score  
584 9/15/2022 84.0 ... 41.0 41.0  
53 3/6/2021 74.0 ... 68.0 60.0  
151 6/16/2021 78.0 ... 86.0 100.0  
606 10/16/2022 66.0 ... 53.0 52.0  
567 8/27/2022 88.0 ... 87.0 100.0  
.. ... ... ... ... ...  
78 4/2/2021 79.0 ... 89.0 41.0  
445 4/19/2022 58.0 ... 58.0 23.0  
645 11/28/2022 81.0 ... 57.0 92.0  
33 2/14/2021 90.0 ... 67.0 100.0  
236 9/15/2021 80.0 ... 90.0 41.0  
  
[819 rows x 54 columns]

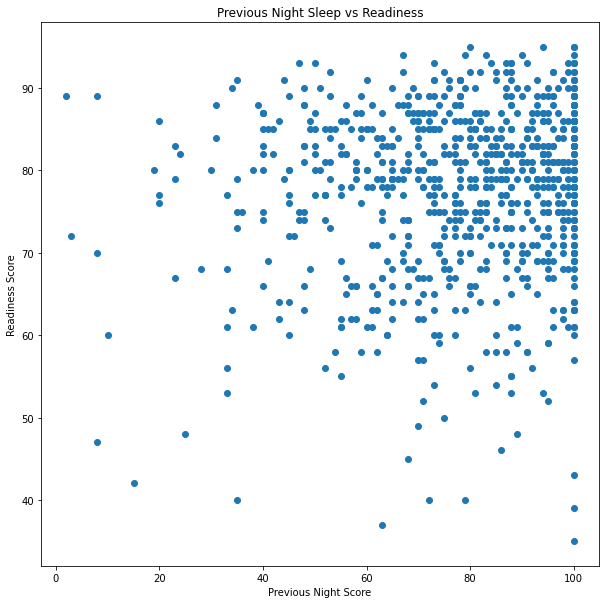
# Reset index if necessary  
# df.reset\_index(drop=True, inplace=True)

## Matplotlib

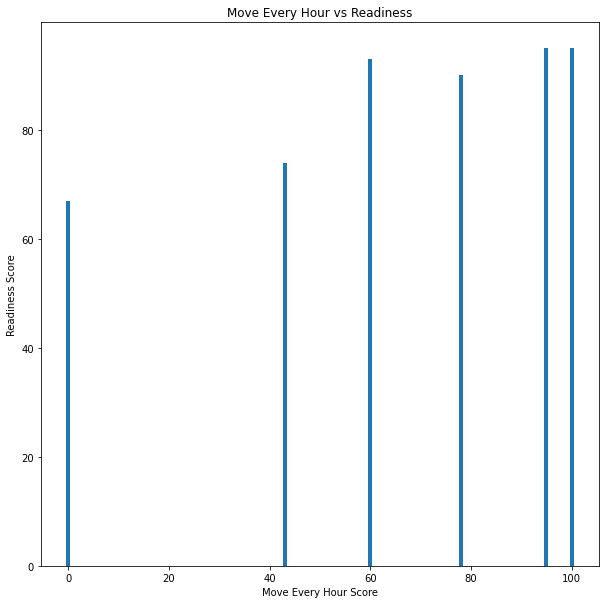
# 1. Prepare data  
x = df["Total Sleep Duration"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Total Sleep vs. Readiness",   
 xlabel="Total Sleep Duration",  
 ylabel="Readiness Score")  
  
# 5. Save and show  
fig.savefig("Figures/Figure\_1.png")



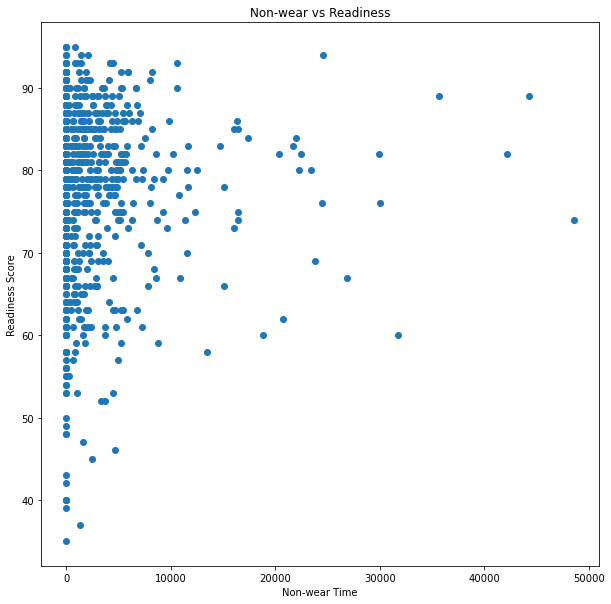
# 1. Prepare data  
x = df["Previous Night Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
  
# 4. Customize plot  
ax.set(title="Previous Night Sleep vs Readiness",   
 xlabel="Previous Night Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_2.png")



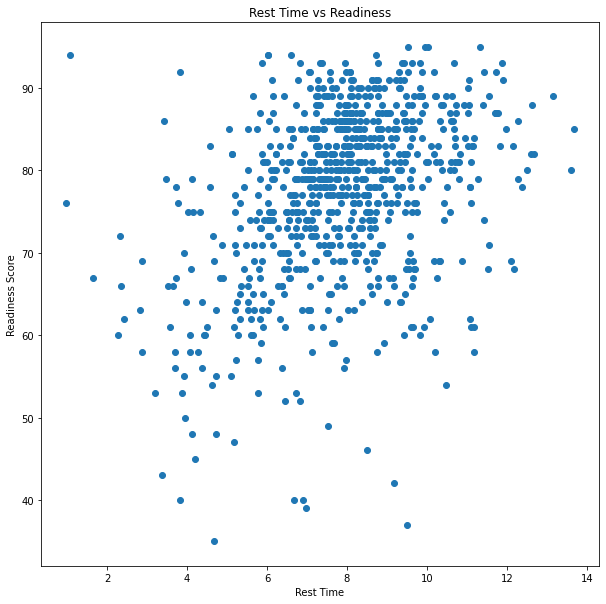
# 1. Prepare data  
x = df["Move Every Hour Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.bar(x,y)  
  
# 4. Customize plot  
ax.set(title="Move Every Hour vs Readiness",   
 xlabel="Move Every Hour Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_3.png")



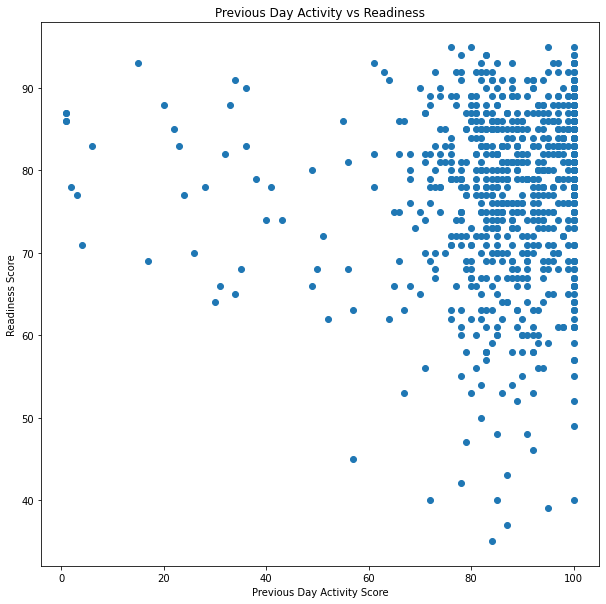
# 1. Prepare data  
x = df["Non-wear Time"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Non-wear vs Readiness",   
 xlabel="Non-wear Time",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_4.png")



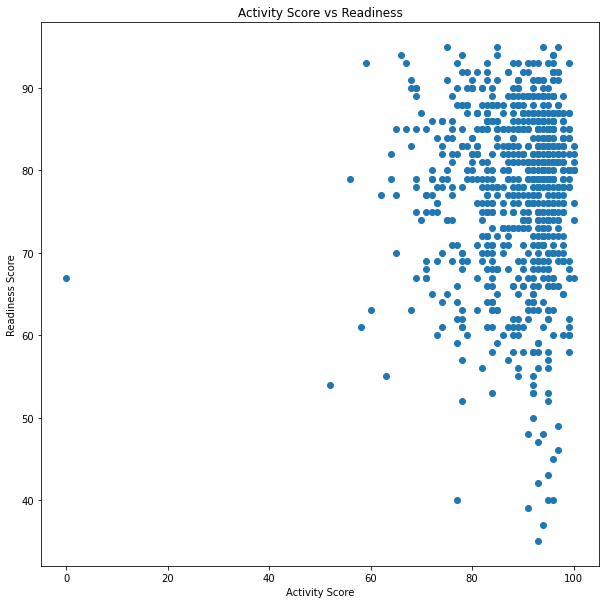
# 1. Prepare data  
x = df["Rest Time"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Rest Time vs Readiness",   
 xlabel="Rest Time",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_5.png")



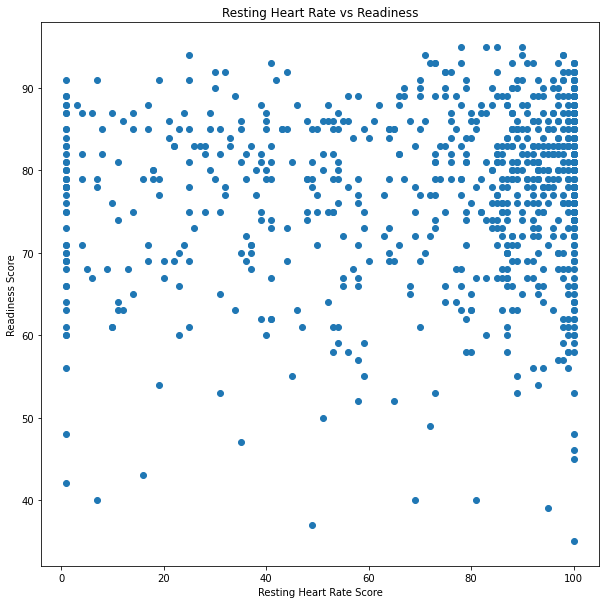
# 1. Prepare data  
x = df["Previous Day Activity Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Previous Day Activity vs Readiness",   
 xlabel="Previous Day Activity Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_6.png")



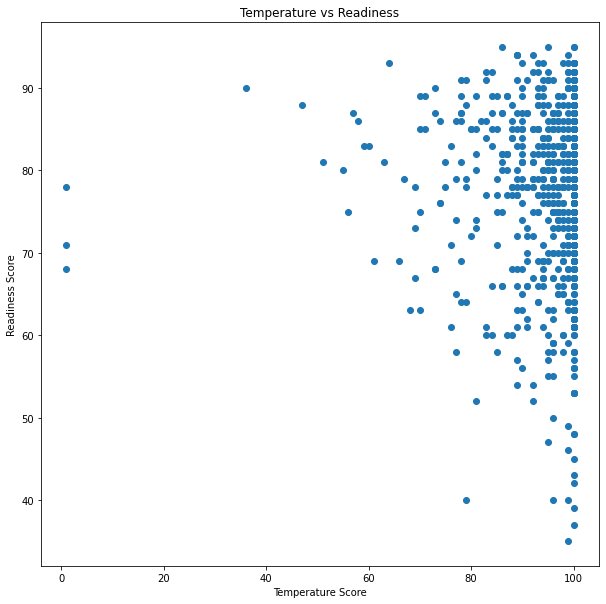
# 1. Prepare data  
x = df["Activity Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Activity Score vs Readiness",   
 xlabel="Activity Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_7.png")



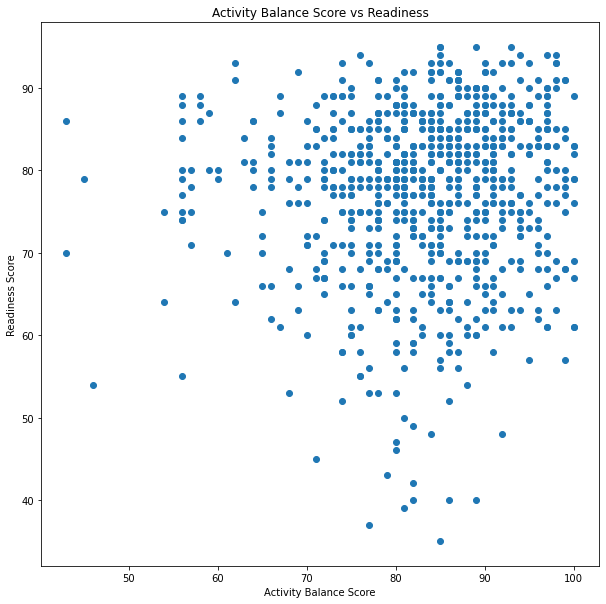
# 1. Prepare data  
x = df["Resting Heart Rate Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Resting Heart Rate vs Readiness",   
 xlabel="Resting Heart Rate Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_8.png")



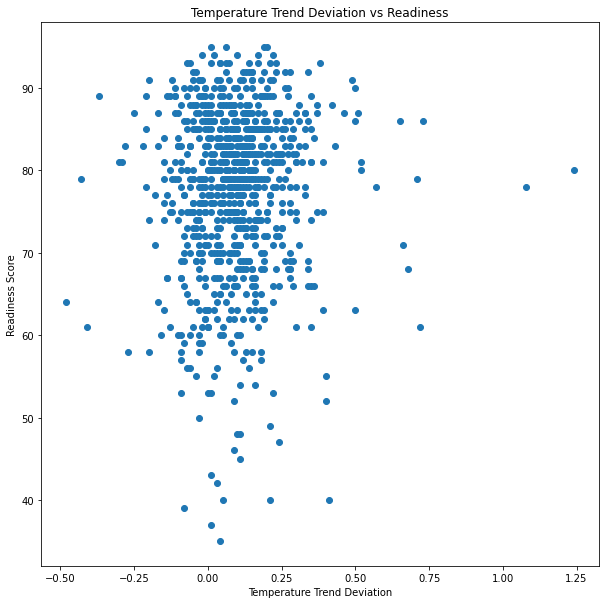
# 1. Prepare data  
x = df["Temperature Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Temperature vs Readiness",   
 xlabel="Temperature Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_9.png")



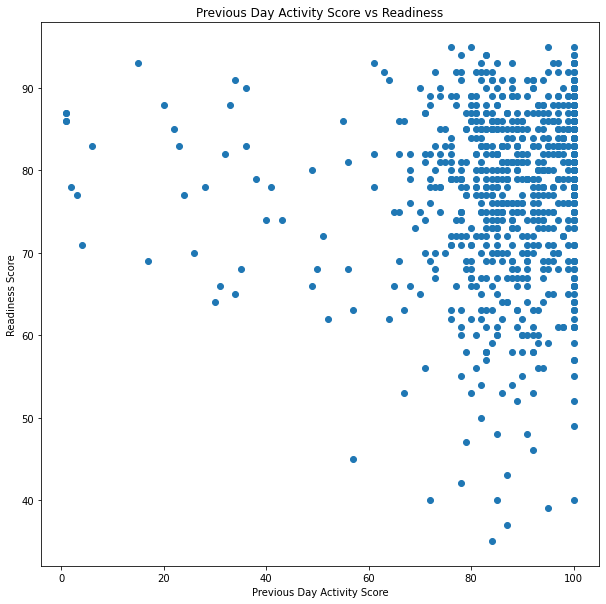
# 1. Prepare data  
x = df["Activity Balance Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Activity Balance Score vs Readiness",   
 xlabel="Activity Balance Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_10.png")



# 1. Prepare data  
x = df["Temperature Trend Deviation"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Temperature Trend Deviation vs Readiness",   
 xlabel="Temperature Trend Deviation",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_11.png")



# 1. Prepare data  
x = df["Previous Day Activity Score"]  
y = df["Readiness Score"]  
  
# 2. Setup plot  
fig, ax = plt.subplots(figsize=(10, 10))  
  
# 3. Plot data  
ax.scatter(x,y)  
  
# 4. Customize plot  
ax.set(title="Previous Day Activity Score vs Readiness",   
 xlabel="Previous Day Activity Score",  
 ylabel="Readiness Score")  
  
# 5. Save and show   
fig.savefig("Figures/Figure\_12.png")



st.header('Relevent Data')

DeltaGenerator()

st.image('Figures/Figure\_1.png')  
st.image('Figures/Figure\_2.png')  
st.image('Figures/Figure\_5.png')  
st.image('Figures/Figure\_7.png')  
st.image('Figures/Figure\_10.png')  
st.image('Figures/Figure\_11.png')  
st.image('Figures/Figure\_12.png')

DeltaGenerator()

## Remove data columns that are lagging data fields or not necessary

df.drop(df.columns[[0,1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,28,29,30,31,32,33,34,35,36,37,39,40,41,42,43,44,50,51,52,53]], axis=1, inplace=True)

df.info()

<class 'pandas.core.frame.DataFrame'>  
Index: 819 entries, 8 to 895  
Data columns (total 9 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Total Sleep Duration 819 non-null float64  
 1 Temperature Trend Deviation 819 non-null float64  
 2 Activity Score 819 non-null float64  
 3 Rest Time 819 non-null float64  
 4 Readiness Score 819 non-null float64  
 5 Previous Night Score 819 non-null float64  
 6 Sleep Balance Score 819 non-null float64  
 7 Previous Day Activity Score 819 non-null float64  
 8 Activity Balance Score 819 non-null float64  
dtypes: float64(9)  
memory usage: 64.0 KB

## Describe Relevent Data

st.header('Describe Relevent Data')  
st.write(df.describe())  
df.describe()

Total Sleep Duration ... Activity Balance Score  
count 819.000000 ... 819.000000  
mean 6.887210 ... 83.106227  
std 1.367014 ... 9.735987  
min 1.575000 ... 43.000000  
25% 6.095833 ... 78.000000  
50% 6.950000 ... 84.000000  
75% 7.708333 ... 90.000000  
max 11.991667 ... 100.000000  
  
[8 rows x 9 columns]

## Algorithm/Estimator

# Import algorithm/estimator  
  
# Instantiate and fit the model (on the training set)  
# Try RandomForest estimator  
  
  
from sklearn.ensemble import RandomForestClassifier  
model = RandomForestClassifier()  
  
# Setup random seed  
np.random.seed(42)  
  
# Create the data  
df.dropna(inplace=True)  
X = df.drop("Readiness Score", axis=1)  
y = df["Readiness Score"] #target  
  
# Split into train and test sets  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

model.get\_params()

{'bootstrap': True,  
 'ccp\_alpha': 0.0,  
 'class\_weight': None,  
 'criterion': 'gini',  
 'max\_depth': None,  
 'max\_features': 'sqrt',  
 'max\_leaf\_nodes': None,  
 'max\_samples': None,  
 'min\_impurity\_decrease': 0.0,  
 'min\_samples\_leaf': 1,  
 'min\_samples\_split': 2,  
 'min\_weight\_fraction\_leaf': 0.0,  
 'n\_estimators': 100,  
 'n\_jobs': None,  
 'oob\_score': False,  
 'random\_state': None,  
 'verbose': 0,  
 'warm\_start': False}

model.fit(X\_train, y\_train);

y\_preds = model.predict(X\_test)

model.score(X\_test, y\_test)

0.054878048780487805

# Try Ridge Regression  
  
from sklearn.linear\_model import Ridge  
  
model = Ridge()  
model.fit(X\_train, y\_train)  
  
# Check the score of the model (on the test set)  
model.score(X\_test, y\_test)

0.3891282480608017

# Try LassoLars  
  
from sklearn import linear\_model  
model = linear\_model.LassoLars(alpha=0.1)  
model.fit(X\_train, y\_train)  
  
# Check the score of the model (on the test set)  
model.score(X\_test, y\_test) #Coefficient of determination of the prediction R^2

0.38925080239747933

model.get\_params()

{'alpha': 0.1,  
 'copy\_X': True,  
 'eps': 2.220446049250313e-16,  
 'fit\_intercept': True,  
 'fit\_path': True,  
 'jitter': None,  
 'max\_iter': 500,  
 'normalize': 'deprecated',  
 'positive': False,  
 'precompute': 'auto',  
 'random\_state': None,  
 'verbose': False}

## Make Predictions Using Machine Language Model

## Pick Typical Day (2/20/2023)

test\_data = pd.read\_csv("New Data/oura\_2023-02-20\_2023-02-20\_trends.csv")

test\_data.drop(test\_data.columns[[0,1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,28,29,30,31,32,33,34,35,36,37,39,40,41,42,43,44,50,51,52,53]], axis=1, inplace=True)

## Remove Readiness Score  
test\_data.drop(test\_data.columns[4], axis=1, inplace=True) #Drop Readiness Score  
test\_data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1 entries, 0 to 0  
Data columns (total 8 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Total Sleep Duration 1 non-null int64   
 1 Temperature Trend Deviation 1 non-null float64  
 2 Activity Score 1 non-null int64   
 3 Rest Time 1 non-null int64   
 4 Previous Night Score 1 non-null int64   
 5 Sleep Balance Score 1 non-null int64   
 6 Previous Day Activity Score 1 non-null int64   
 7 Activity Balance Score 1 non-null int64   
dtypes: float64(1), int64(7)  
memory usage: 196.0 bytes

## Convert Total Sleep Duration to hours  
test\_data["Total Sleep Duration"] = test\_data["Total Sleep Duration"] / 3600  
test\_data["Rest Time"] = test\_data["Rest Time"] / 3600

st.header('Data for a Typical Day (2/20/2023)')

DeltaGenerator()

value = st.slider(  
 'Select a estimated sleep',  
 4.0, 12.0, 8.0)  
st.write('Estimated Sleep:', value)  
test\_data["Total Sleep Duration"] = value  
st.write(test\_data)

## Prediction:

st.header('Based on your estimated sleep, your readiness score for 2/21/23 is prediced to be: ')  
Prediction = str(model.predict(test\_data))  
font\_size = 50  
  
html\_str = f"""  
<style>  
p.a {{  
 font: bold {font\_size}px Courier;  
}}  
</style>  
<p class="a">{Prediction}</p>  
"""  
  
st.markdown(html\_str, unsafe\_allow\_html=True)  
#value=model.predict(test\_data)

DeltaGenerator()

!jupyter nbconvert --to script Capstone.ipynb  
with open('Capstone.py', 'r') as f:  
 lines = f.readlines()  
with open('Capstone.py', 'w') as f:  
 for line in lines:  
 if 'nbconvert --to script' in line:  
 break  
 else:  
 f.write(line)

[NbConvertApp] Converting notebook Capstone.ipynb to script  
[NbConvertApp] Writing 9754 bytes to Capstone.py