presentation_client

September 10, 2020

[1]:
```
# import packages
import os
import pandas as pd
#import pandas_profiling
import numpy as np

import warnings
warnings.filterwarnings('ignore')
import xgboost
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.ensemble import (GradientBoostingRegressor,␣
 ↪GradientBoostingClassifier)
pd.set_option('max.columns',100)
pd.set_option('max.rows',500)


import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

get_ipython().run_line_magic('matplotlib', 'inline')
matplotlib.rcParams['figure.figsize'] = (12,8)

pd.options.mode.chained_assignment = None

import seaborn as sns

from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪StratifiedKFold
from sklearn.linear_model import LogisticRegression
from IPython.display import Image
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.cluster import DBSCAN
from sklearn.neighbors import LocalOutlierFactor
sns.set(style="darkgrid", palette="pastel", color_codes=True)
sns.set_context('talk')
from sklearn.impute import KNNImputer

import missingno as msno
from datetime import datetime
```

[6]:
```
#reading data

os.chdir("D:\\DSP2\\Git\\monitoring-athletes-performance\\main")
data_path = '{}/data'.format(os.path.pardir)
athlete_csv_file = '{}/{}'.format(data_path, 'Eduardo Oliveira (Intermediate).
 ↪csv')
```

[7]:
```
#reading eddy data and print its shape and data type
eddy=pd.read_csv(athlete_csv_file)
print('eddy data shape: ', eddy.shape)#shape
print(eddy.dtypes)#data type
```

```
eddy data shape:  (1140, 43)
Activity Type                object
Date                         object
Favorite                       bool
Title                        object
Distance                     object
Calories                     object
Time                         object
Avg HR                       object
Max HR                       object
Aerobic TE                   object
Avg Run Cadence              object
Max Run Cadence              object
Avg Speed                    object
Max Speed                    object
Elev Gain                    object
Elev Loss                    object
Avg Stride Length           float64
Avg Vertical Ratio          float64
Avg Vertical Oscillation    float64
Avg Ground Contact Time      object
Avg GCT Balance              object
Avg Bike Cadence             object
Max Bike Cadence             object
Normalized Power® (NP®)      object
L/R Balance                  object
```

```
Training Stress Score®      float64
Max Avg Power (20 min)       object
Avg Power                    object
Max Power                    object
Grit                         object
Flow                        float64
Total Strokes                object
Avg. Swolf                   object
Avg Stroke Rate              object
Total Reps                   object
Total Sets                   object
Bottom Time                  object
Min Temp                    float64
Surface Interval             object
Decompression                object
Best Lap Time                object
Number of Laps               object
Max Temp                    float64
dtype: object
```

[8]:
```
eddy.drop(['Favorite','Aerobic TE','Avg Run Cadence','Max Run Cadence','Avg␣
 ↪Stride Length','Avg Vertical Ratio','Avg Vertical Oscillation','Avg Ground␣
 ↪Contact Time'
,'Avg GCT Balance','L/R Balance','Grit','Flow','Total Reps','Total␣
 ↪Sets','Bottom Time','Min Temp','Surface Interval','Decompression','Best Lap␣
 ↪Time','Max Temp'], axis =1, inplace=True)
```

[9]:
```
print(eddy.head())
```

```
            Activity Type                 Date  \
0         Virtual Cycling  2020-04-06 18:15:01
1          Indoor Cycling  2020-04-05 17:00:02
2         Virtual Cycling  2020-04-05 16:00:01
3         Virtual Cycling  2020-04-04 06:59:59
4         Virtual Cycling  2020-04-03 18:00:28
…                     …                   …
1135       Indoor Cycling  2017-03-16 18:44:33
1136              Running  2017-03-16 18:30:17
1137       Indoor Cycling  2017-03-16 18:08:25
1138           Multisport  2017-03-12 07:52:43
1139  Open Water Swimming  2017-03-11 12:56:24

                                          Title Distance Calories  \
0              Zwift - TBR Knights of Suburbia (D)    27.56      479
1                                   Indoor Cycling    14.08      398
2                             Zwift - AHDR BBQ (D)    23.22      431
3     Zwift - Scott D'Aucourt's Meetup - Tick Tock    50.56      838
```

4	Zwift - Haute Route Watopia Stage 1 (E)	10.32	218	
…	…	…	…	…
1135	Indoor Cycling	3.83	118	
1136	Elwood Running	1.84	153	
1137	Indoor Cycling	8.13	198	
1138	Portarlington Multi-Sport	35.83	1,725	
1139	St Leonards Open Water Swimming	411	98	

	Time	Avg HR	Max HR	Avg Speed	Max Speed	Elev Gain	Elev Loss	\
0	00:45:14	--	--	36.6	56.5	80	--	
1	00:36:17	--	--	23.3	30.2	--	--	
2	00:40:38	--	--	34.3	54.1	89	--	
3	01:36:19	--	--	31.5	59.0	158	--	
4	00:19:28	--	--	31.8	68.1	92	--	
…	…	…	…	…	…	…		
1135	00:09:19.0	144	153	24.6	31.4	2	--	
1136	00:10:27	168	183	5:40	4:04	1	3	
1137	00:21:31	125	147	22.7	35.0	2	2	
1138	02:03:46	--	186	17.4	--	180	170	
1139	00:10:34	--	--	2:34	0:40	--	10	

	Avg Bike Cadence	Max Bike Cadence	Normalized Power® (NP®)	\
0	87	111	191	
1	89	127	195	
2	85	111	192	
3	84	125	167	
4	92	116	189	
…	…	…	…	
1135	80	92	--	
1136	--	--	--	
1137	88	115	--	
1138	--	--	--	
1139	--	--	--	

	Training Stress Score®	Max Avg Power (20 min)	Avg Power	Max Power	\
0	0.0	197	181	445	
1	43.2	195	183	623	
2	0.0	198	180	620	
3	0.0	166	152	737	
4	0.0	--	183	647	
…	…	…	…	…	
1135	0.0	--	--	--	
1136	0.0	--	--	--	
1137	0.0	--	--	--	
1138	0.0	--	--	--	
1139	0.0	--	--	--	

	Total Strokes	Avg. Swolf	Avg Stroke Rate	Number of Laps

```
0              --         --              --              1
1            3179         --              --              2
2              --         --              --              1
3              --         --              --              1
4              --         --              --              1
…             …          …               …               …
1135           750        --              --             --
1136            --        --              --             --
1137          1867        --              --             --
1138            --        --              --             --
1139           284        56              26             --

[1140 rows x 23 columns]
```

[10]:
```
eddy.columns
```

[10]:
```
Index(['Activity Type', 'Date', 'Title', 'Distance', 'Calories', 'Time',
       'Avg HR', 'Max HR', 'Avg Speed', 'Max Speed', 'Elev Gain', 'Elev Loss',
       'Avg Bike Cadence', 'Max Bike Cadence', 'Normalized Power® (NP®)',
       'Training Stress Score®', 'Max Avg Power (20 min)', 'Avg Power',
       'Max Power', 'Total Strokes', 'Avg. Swolf', 'Avg Stroke Rate',
       'Number of Laps'],
      dtype='object')
```

[11]:
```
eddy.columns= eddy.columns.str.replace(',', '')
print(eddy.columns)
```

```
Index(['Activity Type', 'Date', 'Title', 'Distance', 'Calories', 'Time',
       'Avg HR', 'Max HR', 'Avg Speed', 'Max Speed', 'Elev Gain', 'Elev Loss',
       'Avg Bike Cadence', 'Max Bike Cadence', 'Normalized Power® (NP®)',
       'Training Stress Score®', 'Max Avg Power (20 min)', 'Avg Power',
       'Max Power', 'Total Strokes', 'Avg. Swolf', 'Avg Stroke Rate',
       'Number of Laps'],
      dtype='object')
```

[12]:
```
eddy.head()
```

[12]:
```
        Activity Type                 Date  \
0     Virtual Cycling  2020-04-06 18:15:01
1      Indoor Cycling  2020-04-05 17:00:02
2     Virtual Cycling  2020-04-05 16:00:01
3     Virtual Cycling  2020-04-04 06:59:59
4     Virtual Cycling  2020-04-03 18:00:28
5     Virtual Cycling  2020-04-03 17:42:41
6     Virtual Cycling  2020-04-03 17:08:26
7     Virtual Cycling  2020-04-02 17:05:54
8     Virtual Cycling  2020-04-01 18:10:01
```

```
9             Running  2020-03-31 18:03:03
10    Virtual Cycling  2020-03-30 18:15:01
11    Virtual Cycling  2020-03-29 17:00:01
12    Virtual Cycling  2020-03-28 06:59:58
13    Virtual Cycling  2020-03-26 18:05:01
14    Virtual Cycling  2020-03-24 18:20:17
15  Strength Training  2020-03-24 17:21:40
16    Virtual Cycling  2020-03-23 20:55:01
17            Running  2020-03-22 08:08:25
18    Virtual Cycling  2020-03-21 19:00:01
19            Running  2020-03-20 08:36:49
20            Running  2020-03-20 07:37:03
21    Virtual Cycling  2020-03-18 16:30:01
22            Running  2020-03-17 18:00:33
23       Road Cycling  2020-03-14 06:41:32
24    Virtual Cycling  2020-03-13 16:45:01
25    Virtual Cycling  2020-03-12 18:49:11
26            Running  2020-03-12 10:09:35
27  Strength Training  2020-03-12 09:19:15
28            Running  2020-03-09 19:19:16
29             Hiking  2020-03-08 09:32:27
30       Road Cycling  2020-03-07 06:32:23
31  Strength Training  2020-03-05 18:19:06
32  Strength Training  2020-03-05 09:20:12
33       Pool Swimming  2020-03-04 05:45:03
34       Pool Swimming  2020-03-02 18:15:17
35            Running  2020-03-01 09:13:49
36       Road Cycling  2020-02-29 06:36:32
37  Strength Training  2020-02-28 18:17:14
38  Strength Training  2020-02-27 09:17:04
39       Pool Swimming  2020-02-26 05:58:00
40            Running  2020-02-25 05:47:01
41       Pool Swimming  2020-02-24 18:15:33
42  Strength Training  2020-02-24 09:16:48

                                                 Title Distance Calories  \
0                 Zwift - TBR Knights of Suburbia (D)    27.56      479
1                                      Indoor Cycling    14.08      398
2                                 Zwift - AHDR BBQ (D)    23.22      431
3        Zwift - Scott D'Aucourt's Meetup - Tick Tock    50.56      838
4            Zwift -  Haute Route Watopia Stage 1 (E)    10.32      218
5                                    Zwift - Richmond     7.77      129
6     Zwift - Stage 1 Race (D) - Tour of Watopia 2020     8.12      191
7                       Zwift - SZR Sunrise Ride (C)    16.77      315
8   Zwift - TBR Crikey Down Under - Galahs vs Womb…    30.96      527
9                                  Melbourne Running     8.02      468
10                Zwift - TBR Knights of Suburbia (D)    27.58      498
```

11	Zwift - AHDR BBQ (D)	36.45	598	
12	Zwift - Scott D'Aucourt's Meetup - Greater Lon…	30.08	654	
13	Zwift - SZR Sunrise Ride (C)	17.09	334	
14	Zwift - NYC	20.96	374	
15	Strength	0.00	223	
16	Zwift - The Herd's Monday Morning Coffee Crew …	12.16	235	
17	Melbourne Running	15.03	982	
18	Zwift - SZR Morning Ride (D)	30.30	537	
19	Melbourne Running	2.35	161	
20	Melbourne Running	8.67	518	
21	Zwift - The Herd's Wednesday Social Down Under…	20.20	356	
22	Melbourne Running	8.05	469	
23	Melbourne Road Cycling	54.42	1,121	
24	Zwift - EVO CC Flux Ride [1.5 - 2.0w/kg avg] (D)	26.12	393	
25	Zwift - TBR Get Fried Fenced Sprint / Spin (D)	17.21	267	
26	Melbourne Running	2.01	117	
27	Strength	0.00	197	
28	Melbourne Running	5.17	352	
29	Pentland Hills Hiking	7.70	734	
30	Melbourne Road Cycling	58.74	647	
31	Strength	0.00	212	
32	Strength	0.00	174	
33	Pool Swimming	2,200	537	
34	Pool Swimming	3,800	873	
35	Melbourne Running	11.01	628	
36	Melbourne Road Cycling	70.53	950	
37	Strength	0.00	175	
38	Strength	0.00	142	
39	Lap Swimming	3,300	760	
40	Melbourne Running	8.01	439	
41	Lap Swimming	3,300	833	
42	Strength	0.00	259	

	Time	Avg HR	Max HR	Avg Speed	Max Speed	Elev Gain	Elev Loss	\
0	00:45:14	--	--	36.6	56.5	80	--	
1	00:36:17	--	--	23.3	30.2	--	--	
2	00:40:38	--	--	34.3	54.1	89	--	
3	01:36:19	--	--	31.5	59.0	158	--	
4	00:19:28	--	--	31.8	68.1	92	--	
5	00:14:24	--	--	32.4	52.3	24	--	
6	00:12:51	--	--	37.9	51.9	37	--	
7	00:27:13	--	--	37.0	63.1	136	--	
8	00:50:13	152	177	37.0	56.3	102	--	
9	00:42:04	146	167	5:15	4:39	40	36	
10	00:45:14	--	--	36.6	64.0	80	--	
11	01:00:19	--	--	36.3	55.3	106	--	
12	01:05:08	161	197	27.7	71.1	262	--	

13	00:27:48	158	183	36.9	57.2	134	--
14	00:42:13	--	--	29.8	57.3	219	--
15	00:34:56	115	152	--	--	--	--
16	00:20:33	151	172	35.5	49.1	31	--
17	01:20:25	159	181	5:21	4:21	148	164
18	00:50:28	158	174	36.0	55.0	102	--
19	00:12:58	148	159	5:31	4:41	20	4
20	00:47:38	145	165	5:30	4:22	50	62
21	00:35:43	--	--	33.9	53.8	121	--
22	00:42:38	143	171	5:18	4:32	31	29
23	01:53:54	150	187	28.7	55.1	220	239
24	00:45:15	142	168	34.6	52.9	82	--
25	00:30:20	150	193	34.0	59.1	36	--
26	00:10:41	138	155	5:19	4:10	21	9
27	00:45:30	97	147	--	--	--	--
28	00:28:25	149	163	5:30	4:18	24	34
29	02:10:23	111	166	16:55	7:14	312	293
30	02:02:04	142	176	28.9	48.9	212	232
31	00:41:43	102	149	--	--	--	--
32	00:41:39	95	139	--	--	--	--
33	00:39:42	--	--	1:28	0:25	--	--
34	01:07:38	--	--	1:39	0:34	--	--
35	00:58:02	146	164	5:16	3:45	82	64
36	02:30:25	137	175	28.1	45.3	209	225
37	00:46:06	92	141	--	--	--	--
38	00:38:18	90	143	--	--	--	--
39	00:55:24	--	--	1:36	0:29	--	--
40	00:40:48	144	158	5:06	3:37	11	12
41	00:54:59	--	--	1:29	0:27	--	--
42	00:47:51	107	142	--	--	--	--

	Avg Bike Cadence	Max Bike Cadence	Normalized Power® (NP®)	\
0	87	111	191	
1	89	127	195	
2	85	111	192	
3	84	125	167	
4	92	116	189	
5	84	123	179	
6	87	105	234	
7	84	98	205	
8	80	100	188	
9	--	--	--	
10	89	113	194	
11	89	127	181	
12	86	122	185	
13	88	106	222	
14	91	113	173	

15	--	--	--
16	81	111	215
17	--	--	--
18	85	130	190
19	--	--	--
20	--	--	--
21	81	121	201
22	--	--	--
23	82	115	189
24	84	118	156
25	84	135	214
26	--	--	--
27	--	--	--
28	--	--	--
29	--	--	--
30	82	115	145
31	--	--	--
32	--	--	--
33	--	--	--
34	--	--	--
35	--	--	--
36	80	121	148
37	--	--	--
38	--	--	--
39	--	--	--
40	--	--	--
41	--	--	--
42	--	--	--

	Training Stress Score®	Max Avg Power (20 min)	Avg Power	Max Power	\
0	0.0	197	181	445	
1	43.2	195	183	623	
2	0.0	198	180	620	
3	0.0	166	152	737	
4	0.0	--	183	647	
5	0.0	--	157	699	
6	0.0	--	228	580	
7	0.0	210	200	383	
8	0.0	189	178	420	
9	0.0	--	--	--	
10	0.0	208	189	672	
11	0.0	182	171	855	
12	0.0	199	175	744	
13	0.0	220	208	565	
14	0.0	170	154	566	
15	0.0	--	--	--	
16	0.0	200	196	740	

17	0.0	--	--	--
18	0.0	200	183	708
19	0.0	--	--	--
20	0.0	--	--	--
21	0.0	188	171	812
22	0.0	--	--	--
23	128.5	188	164	1,039
24	0.0	158	149	654
25	0.0	160	154	944
26	0.0	--	--	--
27	0.0	--	--	--
28	0.0	--	--	--
29	0.0	--	--	--
30	80.4	167	89	1,062
31	0.0	--	--	--
32	0.0	--	--	--
33	0.0	--	--	--
34	0.0	--	--	--
35	0.0	--	--	--
36	103.9	143	106	889
37	0.0	--	--	--
38	0.0	--	--	--
39	0.0	--	--	--
40	0.0	--	--	--
41	0.0	--	--	--
42	0.0	--	--	--

	Total Strokes	Avg. Swolf	Avg Stroke Rate	Number of Laps
0	--	--	--	1
1	3179	--	--	2
2	--	--	--	1
3	--	--	--	1
4	--	--	--	1
5	--	--	--	1
6	--	--	--	1
7	--	--	--	1
8	--	--	--	1
9	--	--	--	9
10	--	--	--	1
11	--	--	--	1
12	--	--	--	1
13	--	--	--	1
14	--	--	--	1
15	--	--	--	1
16	--	--	--	1
17	--	--	--	16
18	--	--	--	1

```
19           --         --              --              3
20           --         --              --              9
21           --         --              --              1
22           --         --              --              9
23         8772         --              --              6
24           --         --              --              1
25           --         --              --              1
26           --         --              --              3
27           --         --              --              1
28           --         --              --              6
29           --         --              --              1
30         9431         --              --              6
31           --         --              --              1
32           --         --              --              1
33          840         63              26              2
34         1762         73              28              4
35           --         --              --             12
36        10859         --              --              8
37           --         --              --              1
38           --         --              --              1
39         1422         70              27              6
40           --         --              --              9
41         1285         64              26              3
42           --         --              --              1
```

[13]:
```
eddy = eddy.replace({ "--": np.nan, "...": np.nan })#missing values replaced by␣
 ↪nan
eddy
```

[13]:
```
            Activity Type                 Date  \
0         Virtual Cycling  2020-04-06 18:15:01
1          Indoor Cycling  2020-04-05 17:00:02
2         Virtual Cycling  2020-04-05 16:00:01
3         Virtual Cycling  2020-04-04 06:59:59
4         Virtual Cycling  2020-04-03 18:00:28
…                     …                   …
1135       Indoor Cycling  2017-03-16 18:44:33
1136              Running  2017-03-16 18:30:17
1137       Indoor Cycling  2017-03-16 18:08:25
1138           Multisport  2017-03-12 07:52:43
1139  Open Water Swimming  2017-03-11 12:56:24

                                          Title Distance Calories  \
0              Zwift - TBR Knights of Suburbia (D)    27.56      479
1                                   Indoor Cycling    14.08      398
2                             Zwift - AHDR BBQ (D)    23.22      431
3     Zwift - Scott D'Aucourt's Meetup - Tick Tock    50.56      838
```

```
4          Zwift -  Haute Route Watopia Stage 1 (E)    10.32      218
…                                                  …      …       …
1135                                Indoor Cycling     3.83      118
1136                                Elwood Running     1.84      153
1137                                Indoor Cycling     8.13      198
1138                     Portarlington Multi-Sport    35.83    1,725
1139                St Leonards Open Water Swimming      411       98

             Time Avg HR Max HR Avg Speed Max Speed Elev Gain Elev Loss  \
0        00:45:14    NaN    NaN      36.6      56.5        80       NaN
1        00:36:17    NaN    NaN      23.3      30.2       NaN       NaN
2        00:40:38    NaN    NaN      34.3      54.1        89       NaN
3        01:36:19    NaN    NaN      31.5      59.0       158       NaN
4        00:19:28    NaN    NaN      31.8      68.1        92       NaN
…           …    …    …       …        …        …       …
1135  00:09:19.0    144    153      24.6      31.4         2       NaN
1136    00:10:27    168    183      5:40      4:04         1         3
1137    00:21:31    125    147      22.7      35.0         2         2
1138    02:03:46    NaN    186      17.4       NaN       180       170
1139    00:10:34    NaN    NaN      2:34      0:40       NaN        10

     Avg Bike Cadence Max Bike Cadence Normalized Power® (NP®)  \
0                  87              111                      191
1                  89              127                      195
2                  85              111                      192
3                  84              125                      167
4                  92              116                      189
…                 …                …                        …
1135               80               92                      NaN
1136              NaN              NaN                      NaN
1137               88              115                      NaN
1138              NaN              NaN                      NaN
1139              NaN              NaN                      NaN

      Training Stress Score® Max Avg Power (20 min) Avg Power Max Power  \
0                        0.0                    197       181       445
1                       43.2                    195       183       623
2                        0.0                    198       180       620
3                        0.0                    166       152       737
4                        0.0                    NaN       183       647
…                       …                       …        …         …
1135                     0.0                    NaN       NaN       NaN
1136                     0.0                    NaN       NaN       NaN
1137                     0.0                    NaN       NaN       NaN
1138                     0.0                    NaN       NaN       NaN
1139                     0.0                    NaN       NaN       NaN
```

```
     Total Strokes Avg. Swolf Avg Stroke Rate Number of Laps
0              NaN        NaN             NaN              1
1             3179        NaN             NaN              2
2              NaN        NaN             NaN              1
3              NaN        NaN             NaN              1
4              NaN        NaN             NaN              1
…              …          …               …                …
1135           750        NaN             NaN            NaN
1136           NaN        NaN             NaN            NaN
1137          1867        NaN             NaN            NaN
1138           NaN        NaN             NaN            NaN
1139           284         56              26            NaN

[1140 rows x 23 columns]
```

[14]:
```
#capitalization
eddy['Activity Type'] = eddy['Activity Type'].str.lower()
eddy['Title'] = eddy['Title'].str.lower()
print(eddy.head())
```

```
            Activity Type                 Date  \
0         virtual cycling  2020-04-06 18:15:01
1          indoor cycling  2020-04-05 17:00:02
2         virtual cycling  2020-04-05 16:00:01
3         virtual cycling  2020-04-04 06:59:59
4         virtual cycling  2020-04-03 18:00:28
…                     …                    …
1135       indoor cycling  2017-03-16 18:44:33
1136              running  2017-03-16 18:30:17
1137       indoor cycling  2017-03-16 18:08:25
1138           multisport  2017-03-12 07:52:43
1139  open water swimming  2017-03-11 12:56:24

                                          Title Distance Calories  \
0             zwift - tbr knights of suburbia (d)    27.56      479
1                                  indoor cycling    14.08      398
2                           zwift - ahdr bbq (d)    23.22      431
3     zwift - scott d'aucourt's meetup - tick tock    50.56      838
4         zwift -  haute route watopia stage 1 (e)    10.32      218
…                                               …       …        …
1135                               indoor cycling     3.83      118
1136                               elwood running     1.84      153
1137                               indoor cycling     8.13      198
1138                    portarlington multi-sport    35.83    1,725
1139                st leonards open water swimming      411       98

             Time Avg HR Max HR Avg Speed Max Speed Elev Gain Elev Loss  \
```

```
0       00:45:14    NaN    NaN      36.6      56.5        80       NaN
1       00:36:17    NaN    NaN      23.3      30.2       NaN       NaN
2       00:40:38    NaN    NaN      34.3      54.1        89       NaN
3       01:36:19    NaN    NaN      31.5      59.0       158       NaN
4       00:19:28    NaN    NaN      31.8      68.1        92       NaN
…          …    …    …       …         …         …       …
1135  00:09:19.0    144    153      24.6      31.4         2       NaN
1136    00:10:27    168    183      5:40      4:04         1         3
1137    00:21:31    125    147      22.7      35.0         2         2
1138    02:03:46    NaN    186      17.4       NaN       180       170
1139    00:10:34    NaN    NaN      2:34      0:40       NaN        10

     Avg Bike Cadence Max Bike Cadence Normalized Power® (NP®)  \
0                  87              111                       191
1                  89              127                       195
2                  85              111                       192
3                  84              125                       167
4                  92              116                       189
…                 …               …                         …
1135               80               92                       NaN
1136              NaN              NaN                       NaN
1137               88              115                       NaN
1138              NaN              NaN                       NaN
1139              NaN              NaN                       NaN

      Training Stress Score® Max Avg Power (20 min) Avg Power Max Power  \
0                        0.0                    197       181       445
1                       43.2                    195       183       623
2                        0.0                    198       180       620
3                        0.0                    166       152       737
4                        0.0                    NaN       183       647
…                       …                      …         …         …
1135                     0.0                    NaN       NaN       NaN
1136                     0.0                    NaN       NaN       NaN
1137                     0.0                    NaN       NaN       NaN
1138                     0.0                    NaN       NaN       NaN
1139                     0.0                    NaN       NaN       NaN

     Total Strokes Avg. Swolf Avg Stroke Rate Number of Laps
0              NaN        NaN             NaN              1
1             3179        NaN             NaN              2
2              NaN        NaN             NaN              1
3              NaN        NaN             NaN              1
4              NaN        NaN             NaN              1
…             …          …               …               …
1135           750        NaN             NaN             NaN
1136           NaN        NaN             NaN             NaN
1137          1867        NaN             NaN             NaN
```

```
1138           NaN         NaN              NaN             NaN
1139           284          56               26             NaN

[1140 rows x 23 columns]
```

[15]:
```
#formats
eddy['Elev Gain'] = eddy['Elev Gain'].str.replace(',', '')
eddy['Elev Gain'] = eddy['Elev Gain'].astype(float)
```

[16]:
```
eddy["Elev Gain"] = pd.to_numeric(eddy["Elev Gain"])
print(eddy.dtypes)
```

```
Activity Type               object
Date                        object
Title                       object
Distance                    object
Calories                    object
Time                        object
Avg HR                      object
Max HR                      object
Avg Speed                   object
Max Speed                   object
Elev Gain                  float64
Elev Loss                   object
Avg Bike Cadence            object
Max Bike Cadence            object
Normalized Power® (NP®)     object
Training Stress Score®     float64
Max Avg Power (20 min)      object
Avg Power                   object
Max Power                   object
Total Strokes               object
Avg. Swolf                  object
Avg Stroke Rate             object
Number of Laps              object
dtype: object
```

[17]:
```
eddy['Elev Loss'] = eddy['Elev Loss'].str.replace(',', '')
eddy['Elev Loss'] = eddy['Elev Loss'].astype(float)
```

[18]:
```
eddy['Elev Loss'] = pd.to_numeric(eddy['Elev Loss'])
print(eddy.dtypes)
```

```
Activity Type               object
Date                        object
Title                       object
Distance                    object
Calories                    object
```

```
Time                          object
Avg HR                        object
Max HR                        object
Avg Speed                     object
Max Speed                     object
Elev Gain                    float64
Elev Loss                    float64
Avg Bike Cadence              object
Max Bike Cadence              object
Normalized Power® (NP®)       object
Training Stress Score®       float64
Max Avg Power (20 min)        object
Avg Power                     object
Max Power                     object
Total Strokes                 object
Avg. Swolf                    object
Avg Stroke Rate               object
Number of Laps                object
dtype: object
```

[19]:
```
eddy['Distance'] = eddy['Distance'].str.replace(',', '')
eddy['Distance'] = eddy['Distance'].astype(float)
```

[20]:
```
eddy['Distance'] = pd.to_numeric(eddy['Distance'])
print(eddy.dtypes)
```

```
Activity Type                 object
Date                          object
Title                         object
Distance                     float64
Calories                      object
Time                          object
Avg HR                        object
Max HR                        object
Avg Speed                     object
Max Speed                     object
Elev Gain                    float64
Elev Loss                    float64
Avg Bike Cadence              object
Max Bike Cadence              object
Normalized Power® (NP®)       object
Training Stress Score®       float64
Max Avg Power (20 min)        object
Avg Power                     object
Max Power                     object
Total Strokes                 object
Avg. Swolf                    object
Avg Stroke Rate               object
```

```
Number of Laps              object
dtype: object
```

[21]:
```
eddy['Calories'] = eddy['Calories'].str.replace(',', '')
eddy['Calories'] = eddy['Calories'].astype(float)
```

[22]:
```
eddy['Calories'] = pd.to_numeric(eddy['Calories'])
print(eddy.dtypes)
```

```
Activity Type               object
Date                        object
Title                       object
Distance                   float64
Calories                   float64
Time                        object
Avg HR                      object
Max HR                      object
Avg Speed                   object
Max Speed                   object
Elev Gain                  float64
Elev Loss                  float64
Avg Bike Cadence            object
Max Bike Cadence            object
Normalized Power® (NP®)     object
Training Stress Score®     float64
Max Avg Power (20 min)      object
Avg Power                   object
Max Power                   object
Total Strokes               object
Avg. Swolf                  object
Avg Stroke Rate             object
Number of Laps              object
dtype: object
```

[23]:
```
eddy['Max Power'] = eddy['Max Power'].str.replace(',', '')
eddy['Max Power'] = eddy['Max Power'].astype(float)
eddy['Max Power'] = pd.to_numeric(eddy['Max Power'])
print(eddy.dtypes)
```

```
Activity Type               object
Date                        object
Title                       object
Distance                   float64
Calories                   float64
Time                        object
Avg HR                      object
Max HR                      object
Avg Speed                   object
```

```
Max Speed                   object
Elev Gain                  float64
Elev Loss                  float64
Avg Bike Cadence            object
Max Bike Cadence            object
Normalized Power® (NP®)     object
Training Stress Score®     float64
Max Avg Power (20 min)      object
Avg Power                   object
Max Power                  float64
Total Strokes               object
Avg. Swolf                  object
Avg Stroke Rate             object
Number of Laps              object
dtype: object
```

[24]:
```
eddy['Avg Power'] = eddy['Avg Power'].astype(str)
eddy['Avg Power'] = eddy['Avg Power'].str.replace(',', '')
eddy['Avg Power'] = eddy['Avg Power'].astype(float)
eddy['Avg Power'] = pd.to_numeric(eddy['Avg Power'])
print(eddy.dtypes)
```

```
Activity Type               object
Date                        object
Title                       object
Distance                   float64
Calories                   float64
Time                        object
Avg HR                      object
Max HR                      object
Avg Speed                   object
Max Speed                   object
Elev Gain                  float64
Elev Loss                  float64
Avg Bike Cadence            object
Max Bike Cadence            object
Normalized Power® (NP®)     object
Training Stress Score®     float64
Max Avg Power (20 min)      object
Avg Power                  float64
Max Power                  float64
Total Strokes               object
Avg. Swolf                  object
Avg Stroke Rate             object
Number of Laps              object
dtype: object
```

[25]:
```
eddy.loc[eddy['Avg Speed'].str.contains(":", na=False),'Avg Speed']=np.nan
eddy['Avg Speed'] = pd.to_numeric(eddy['Avg Speed'])
```

[26]:
```
eddy.loc[eddy['Max Speed'].str.contains(":", na=False),'Max Speed']=np.nan
eddy['Max Speed'] = pd.to_numeric(eddy['Max Speed'])
print(eddy.dtypes)
```

```
Activity Type              object
Date                       object
Title                      object
Distance                  float64
Calories                  float64
Time                       object
Avg HR                     object
Max HR                     object
Avg Speed                 float64
Max Speed                 float64
Elev Gain                 float64
Elev Loss                 float64
Avg Bike Cadence           object
Max Bike Cadence           object
Normalized Power® (NP®)    object
Training Stress Score®    float64
Max Avg Power (20 min)     object
Avg Power                 float64
Max Power                 float64
Total Strokes              object
Avg. Swolf                 object
Avg Stroke Rate            object
Number of Laps             object
dtype: object
```

[27]:
```
eddy[['Max Avg Power (20 min)','Avg Power','Avg Stroke Rate','Avg HR', 'Max␣
 ↪HR','Total Strokes','Avg. Swolf','Avg Bike Cadence','Max Bike␣
 ↪Cadence','Normalized Power® (NP®)', 'Number of Laps']] = eddy[['Max Avg␣
 ↪Power (20 min)','Avg Power','Avg Stroke Rate','Avg HR', 'Max HR','Total␣
 ↪Strokes','Avg. Swolf','Avg Bike Cadence','Max Bike Cadence','Normalized␣
 ↪Power® (NP®)' ,'Number of Laps']].apply(pd.to_numeric)
print(eddy.dtypes)
```

```
Activity Type              object
Date                       object
Title                      object
Distance                  float64
Calories                  float64
Time                       object
Avg HR                    float64
```

```
Max HR                        float64
Avg Speed                     float64
Max Speed                     float64
Elev Gain                     float64
Elev Loss                     float64
Avg Bike Cadence              float64
Max Bike Cadence              float64
Normalized Power® (NP®)       float64
Training Stress Score®        float64
Max Avg Power (20 min)        float64
Avg Power                     float64
Max Power                     float64
Total Strokes                 float64
Avg. Swolf                    float64
Avg Stroke Rate               float64
Number of Laps                float64
dtype: object
```

[28]:
```
#Translating DateTime into Date and Time
eddy['Date_extracted']=pd.to_datetime(eddy["Date"]).dt.normalize()
eddy['Time_extracted']=pd.to_datetime(eddy["Date"]).dt.time
eddy['Date']=pd.to_datetime(eddy['Date'])
```

[29]:
```
#Converting Time into sec for future analysis
eddy['Time_sec']=pd.to_timedelta(pd.to_datetime(eddy["Time"]).dt.strftime('%H:
 ↪%M:%S')).dt.total_seconds()
```

[30]:
```
print(eddy.dtypes)
```

```
Activity Type                      object
Date                       datetime64[ns]
Title                              object
Distance                          float64
Calories                          float64
Time                               object
Avg HR                            float64
Max HR                            float64
Avg Speed                         float64
Max Speed                         float64
Elev Gain                         float64
Elev Loss                         float64
Avg Bike Cadence                  float64
Max Bike Cadence                  float64
Normalized Power® (NP®)           float64
Training Stress Score®            float64
Max Avg Power (20 min)            float64
Avg Power                         float64
Max Power                         float64
```

```
Total Strokes                      float64
Avg. Swolf                         float64
Avg Stroke Rate                    float64
Number of Laps                     float64
Date_extracted             datetime64[ns]
Time_extracted                      object
Time_sec                           float64
dtype: object
```

[31]:
```
#handling irregular data
# select numeric columns
def func_numeric():
    eddy_numeric = eddy.select_dtypes(include=[np.number])
    numeric_cols = eddy_numeric.columns.values
    return numeric_cols,eddy_numeric
numeric_cols,eddy_numeric = func_numeric()
print(numeric_cols)
```

```
['Distance' 'Calories' 'Avg HR' 'Max HR' 'Avg Speed' 'Max Speed'
 'Elev Gain' 'Elev Loss' 'Avg Bike Cadence' 'Max Bike Cadence'
 'Normalized Power® (NP®)' 'Training Stress Score®'
 'Max Avg Power (20 min)' 'Avg Power' 'Max Power' 'Total Strokes'
 'Avg. Swolf' 'Avg Stroke Rate' 'Number of Laps' 'Time_sec']
```

[32]:
```
def func_categoric():
    eddy_categoric= eddy.select_dtypes(exclude=[np.number])
    categoric_cols = eddy_categoric.columns.values
    return  eddy_categoric,categoric_cols
eddy_categoric,categoric_cols = func_categoric()
print(categoric_cols)
```

```
['Activity Type' 'Date' 'Title' 'Time' 'Date_extracted' 'Time_extracted']
```

[33]:
```
def find_missing_percent(data):
    """
    Returns dataframe containing the total missing values and percentage of␣
 ↪total
    missing values of a column.
    """
    miss_eddy = pd.DataFrame({'ColumnName':[],'TotalMissingVals':
 ↪[],'PercentMissing':[]})
    for col in data.columns:
        sum_miss_val = data[col].isnull().sum()
        percent_miss_val = round((sum_miss_val/data.shape[0])*100,2)
        miss_eddy = miss_eddy.append(dict(zip(miss_eddy.
 ↪columns,[col,sum_miss_val,percent_miss_val])),ignore_index=True)
    return miss_eddy
```

```
miss_eddy = find_missing_percent(eddy)
'''Columns with missing values'''
print(f"Number of columns with missing values:␣
 ↪{str(miss_eddy[miss_eddy['PercentMissing']>0.0].shape[0])}")
display(miss_eddy[miss_eddy['PercentMissing']>0.0])
#'''Drop the columns with more than 90% of missing values'''
#drop_cols = miss_df[miss_df['PercentMissing'] >90.0].ColumnName.tolist()
#eddy = eddy.drop(drop_cols,axis=1)
```

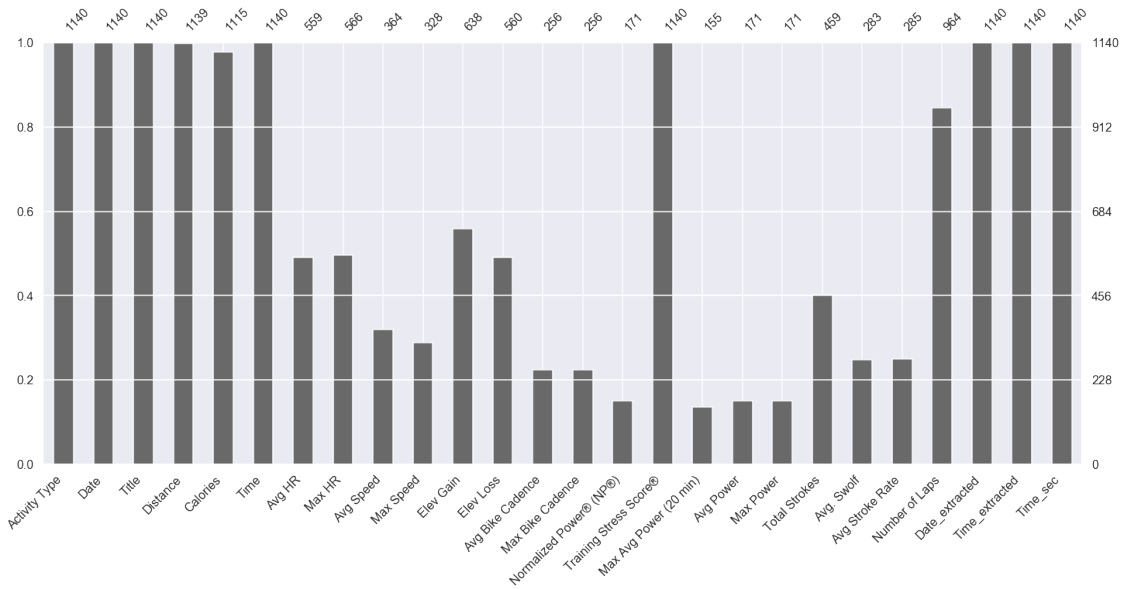
```
Number of columns with missing values: 18
```

	ColumnName	TotalMissingVals	PercentMissing
3	Distance	1.0	0.09
4	Calories	25.0	2.19
6	Avg HR	581.0	50.96
7	Max HR	574.0	50.35
8	Avg Speed	776.0	68.07
9	Max Speed	812.0	71.23
10	Elev Gain	502.0	44.04
11	Elev Loss	580.0	50.88
12	Avg Bike Cadence	884.0	77.54
13	Max Bike Cadence	884.0	77.54
14	Normalized Power® (NP®)	969.0	85.00
16	Max Avg Power (20 min)	985.0	86.40
17	Avg Power	969.0	85.00
18	Max Power	969.0	85.00
19	Total Strokes	681.0	59.74
20	Avg. Swolf	857.0	75.18
21	Avg Stroke Rate	855.0	75.00
22	Number of Laps	176.0	15.44

[34]:
```
# In[51]:

def missingno_bar():
    graph = msno.bar(eddy)
    return graph
print(missingno_bar())
#msno.bar(eddy)
```

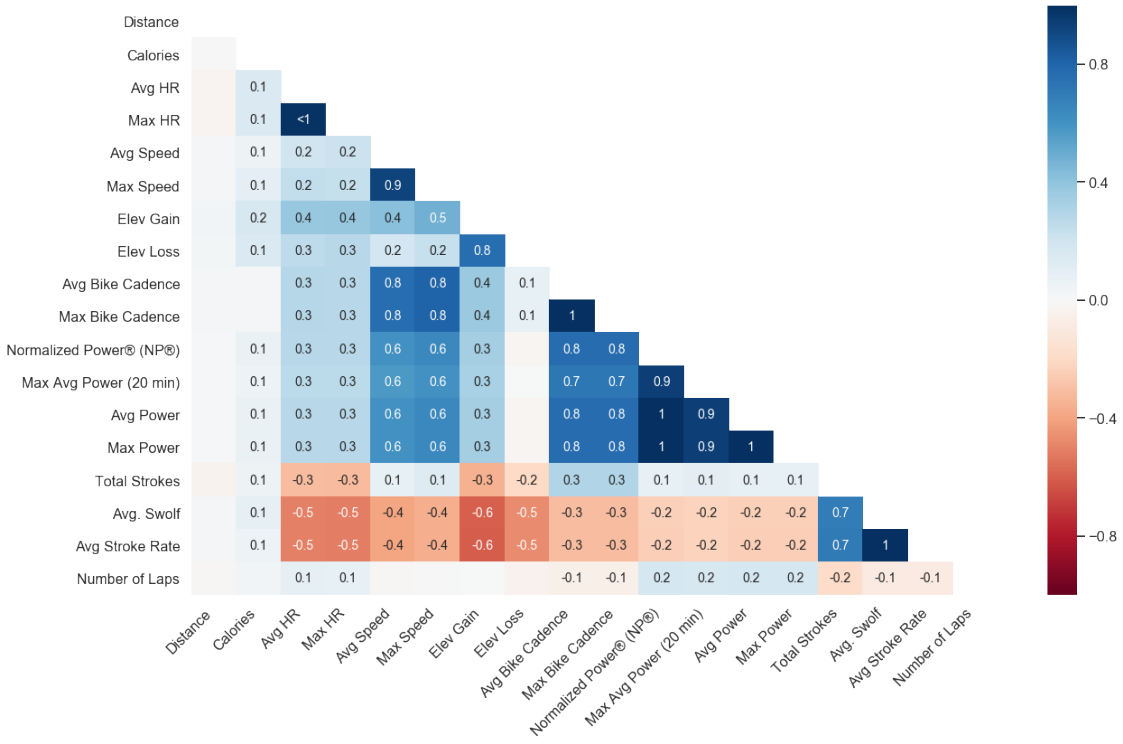
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



[36]:
```
#def missingno_matrix():
   # matrix = msno.matrix(eddy)
    #return matrix
#print(missingno_matrix())
#msno.matrix(eddy)#for visulaising the locations of the missing data
```

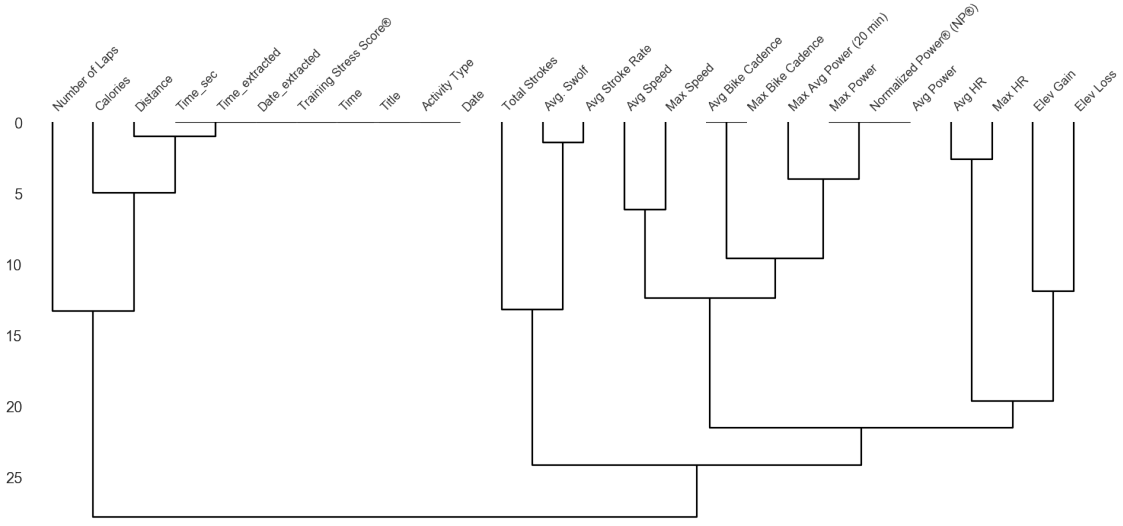
[37]:
```
def heat_map():
    heatmap=msno.heatmap(eddy)
    return(heatmap)
print(heat_map())
#msno.heatmap(eddy)
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



[32]:
```
def deno_gram():
    dendogram = msno.dendrogram(eddy)
    return(dendogram)
print(deno_gram())
#msno.dendrogram(eddy)# for grouping highly corelated   variable
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



[ ]:
```
pandas_profiling.ProfileReport(eddy)#not working with function
```

[38]:
```
#mean imputation

def mean_imputation(eddy_numeric2):
    for col in eddy_numeric2.columns:
        mean = eddy_numeric2[col].mean()
        eddy_numeric2[col] = eddy_numeric2[col].fillna(mean)
    return eddy_numeric2

#eddy_numeric,numeric_cols=func_numeric()
eddy_numeric2=eddy[numeric_cols]
eddy_mean_imp = mean_imputation(eddy_numeric2)
eddy_mean_imp.head(10)
```

[38]:

	Distance	Calories	Avg HR	Max HR	Avg Speed	Max Speed	\
0	27.56	479.0	136.146691	165.222615	36.600000	56.500000	
1	14.08	398.0	136.146691	165.222615	23.300000	30.200000	
2	23.22	431.0	136.146691	165.222615	34.300000	54.100000	
3	50.56	838.0	136.146691	165.222615	31.500000	59.000000	
4	10.32	218.0	136.146691	165.222615	31.800000	68.100000	
5	7.77	129.0	136.146691	165.222615	32.400000	52.300000	
6	8.12	191.0	136.146691	165.222615	37.900000	51.900000	
7	16.77	315.0	136.146691	165.222615	37.000000	63.100000	
8	30.96	527.0	152.000000	177.000000	37.000000	56.300000	
9	8.02	468.0	146.000000	167.000000	27.778022	49.810366	
10	27.58	498.0	136.146691	165.222615	36.600000	64.000000	
11	36.45	598.0	136.146691	165.222615	36.300000	55.300000	
12	30.08	654.0	161.000000	197.000000	27.700000	71.100000	
13	17.09	334.0	158.000000	183.000000	36.900000	57.200000	
14	20.96	374.0	136.146691	165.222615	29.800000	57.300000	
15	0.00	223.0	115.000000	152.000000	27.778022	49.810366	
16	12.16	235.0	151.000000	172.000000	35.500000	49.100000	
17	15.03	982.0	159.000000	181.000000	27.778022	49.810366	
18	30.30	537.0	158.000000	174.000000	36.000000	55.000000	
19	2.35	161.0	148.000000	159.000000	27.778022	49.810366	
20	8.67	518.0	145.000000	165.000000	27.778022	49.810366	
21	20.20	356.0	136.146691	165.222615	33.900000	53.800000	
22	8.05	469.0	143.000000	171.000000	27.778022	49.810366	
23	54.42	1121.0	150.000000	187.000000	28.700000	55.100000	
24	26.12	393.0	142.000000	168.000000	34.600000	52.900000	
25	17.21	267.0	150.000000	193.000000	34.000000	59.100000	
26	2.01	117.0	138.000000	155.000000	27.778022	49.810366	
27	0.00	197.0	97.000000	147.000000	27.778022	49.810366	
28	5.17	352.0	149.000000	163.000000	27.778022	49.810366	

29	7.70	734.0	111.000000	166.000000	27.778022	49.810366
30	58.74	647.0	142.000000	176.000000	28.900000	48.900000
31	0.00	212.0	102.000000	149.000000	27.778022	49.810366
32	0.00	174.0	95.000000	139.000000	27.778022	49.810366
33	2200.00	537.0	136.146691	165.222615	27.778022	49.810366
34	3800.00	873.0	136.146691	165.222615	27.778022	49.810366
35	11.01	628.0	146.000000	164.000000	27.778022	49.810366
36	70.53	950.0	137.000000	175.000000	28.100000	45.300000
37	0.00	175.0	92.000000	141.000000	27.778022	49.810366
38	0.00	142.0	90.000000	143.000000	27.778022	49.810366
39	3300.00	760.0	136.146691	165.222615	27.778022	49.810366
40	8.01	439.0	144.000000	158.000000	27.778022	49.810366
41	3300.00	833.0	136.146691	165.222615	27.778022	49.810366
42	0.00	259.0	107.000000	142.000000	27.778022	49.810366
43	6.58	398.0	145.000000	162.000000	27.778022	49.810366
44	3.60	226.0	143.000000	157.000000	27.778022	49.810366
45	15.03	26.0	127.000000	150.000000	26.800000	34.700000
46	47.21	1749.0	136.146691	165.222615	25.200000	44.600000
47	5.50	407.0	153.000000	165.000000	27.778022	49.810366
48	5.01	283.0	148.000000	167.000000	27.778022	49.810366
49	4850.00	1186.0	136.146691	165.222615	27.778022	49.810366

	Elev Gain	Elev Loss	Avg Bike Cadence	Max Bike Cadence	\
0	80.000000	164.817857	87.000000	111.000000	
1	158.619122	164.817857	89.000000	127.000000	
2	89.000000	164.817857	85.000000	111.000000	
3	158.000000	164.817857	84.000000	125.000000	
4	92.000000	164.817857	92.000000	116.000000	
5	24.000000	164.817857	84.000000	123.000000	
6	37.000000	164.817857	87.000000	105.000000	
7	136.000000	164.817857	84.000000	98.000000	
8	102.000000	164.817857	80.000000	100.000000	
9	40.000000	36.000000	80.796875	120.796875	
10	80.000000	164.817857	89.000000	113.000000	
11	106.000000	164.817857	89.000000	127.000000	
12	262.000000	164.817857	86.000000	122.000000	
13	134.000000	164.817857	88.000000	106.000000	
14	219.000000	164.817857	91.000000	113.000000	
15	158.619122	164.817857	80.796875	120.796875	
16	31.000000	164.817857	81.000000	111.000000	
17	148.000000	164.000000	80.796875	120.796875	
18	102.000000	164.817857	85.000000	130.000000	
19	20.000000	4.000000	80.796875	120.796875	
20	50.000000	62.000000	80.796875	120.796875	
21	121.000000	164.817857	81.000000	121.000000	
22	31.000000	29.000000	80.796875	120.796875	
23	220.000000	239.000000	82.000000	115.000000	

24	82.000000	164.817857	84.000000	118.000000
25	36.000000	164.817857	84.000000	135.000000
26	21.000000	9.000000	80.796875	120.796875
27	158.619122	164.817857	80.796875	120.796875
28	24.000000	34.000000	80.796875	120.796875
29	312.000000	293.000000	80.796875	120.796875
30	212.000000	232.000000	82.000000	115.000000
31	158.619122	164.817857	80.796875	120.796875
32	158.619122	164.817857	80.796875	120.796875
33	158.619122	164.817857	80.796875	120.796875
34	158.619122	164.817857	80.796875	120.796875
35	82.000000	64.000000	80.796875	120.796875
36	209.000000	225.000000	80.000000	121.000000
37	158.619122	164.817857	80.796875	120.796875
38	158.619122	164.817857	80.796875	120.796875
39	158.619122	164.817857	80.796875	120.796875
40	11.000000	12.000000	80.796875	120.796875
41	158.619122	164.817857	80.796875	120.796875
42	158.619122	164.817857	80.796875	120.796875
43	40.000000	41.000000	80.796875	120.796875
44	11.000000	19.000000	80.796875	120.796875
45	36.000000	28.000000	80.000000	101.000000
46	184.000000	197.000000	80.796875	120.796875
47	52.000000	74.000000	80.796875	120.796875
48	11.000000	19.000000	80.796875	120.796875
49	158.619122	164.817857	80.796875	120.796875

	Normalized Power® (NP®)	Training Stress Score®	Max Avg Power (20 min)	\
0	191.000000	0.0	197.000000	
1	195.000000	43.2	195.000000	
2	192.000000	0.0	198.000000	
3	167.000000	0.0	166.000000	
4	189.000000	0.0	174.567742	
5	179.000000	0.0	174.567742	
6	234.000000	0.0	174.567742	
7	205.000000	0.0	210.000000	
8	188.000000	0.0	189.000000	
9	173.923977	0.0	174.567742	
10	194.000000	0.0	208.000000	
11	181.000000	0.0	182.000000	
12	185.000000	0.0	199.000000	
13	222.000000	0.0	220.000000	
14	173.000000	0.0	170.000000	
15	173.923977	0.0	174.567742	
16	215.000000	0.0	200.000000	
17	173.923977	0.0	174.567742	
18	190.000000	0.0	200.000000	

19	173.923977	0.0	174.567742
20	173.923977	0.0	174.567742
21	201.000000	0.0	188.000000
22	173.923977	0.0	174.567742
23	189.000000	128.5	188.000000
24	156.000000	0.0	158.000000
25	214.000000	0.0	160.000000
26	173.923977	0.0	174.567742
27	173.923977	0.0	174.567742
28	173.923977	0.0	174.567742
29	173.923977	0.0	174.567742
30	145.000000	80.4	167.000000
31	173.923977	0.0	174.567742
32	173.923977	0.0	174.567742
33	173.923977	0.0	174.567742
34	173.923977	0.0	174.567742
35	173.923977	0.0	174.567742
36	148.000000	103.9	143.000000
37	173.923977	0.0	174.567742
38	173.923977	0.0	174.567742
39	173.923977	0.0	174.567742
40	173.923977	0.0	174.567742
41	173.923977	0.0	174.567742
42	173.923977	0.0	174.567742
43	173.923977	0.0	174.567742
44	173.923977	0.0	174.567742
45	39.000000	1.6	12.000000
46	173.923977	0.0	174.567742
47	173.923977	0.0	174.567742
48	173.923977	0.0	174.567742
49	173.923977	0.0	174.567742

	Avg Power	Max Power	Total Strokes	Avg. Swolf	Avg Stroke Rate	\
0	181.000000	445.000000	4174.936819	62.190813	27.308772	
1	183.000000	623.000000	3179.000000	62.190813	27.308772	
2	180.000000	620.000000	4174.936819	62.190813	27.308772	
3	152.000000	737.000000	4174.936819	62.190813	27.308772	
4	183.000000	647.000000	4174.936819	62.190813	27.308772	
5	157.000000	699.000000	4174.936819	62.190813	27.308772	
6	228.000000	580.000000	4174.936819	62.190813	27.308772	
7	200.000000	383.000000	4174.936819	62.190813	27.308772	
8	178.000000	420.000000	4174.936819	62.190813	27.308772	
9	148.894737	681.853801	4174.936819	62.190813	27.308772	
10	189.000000	672.000000	4174.936819	62.190813	27.308772	
11	171.000000	855.000000	4174.936819	62.190813	27.308772	
12	175.000000	744.000000	4174.936819	62.190813	27.308772	
13	208.000000	565.000000	4174.936819	62.190813	27.308772	

14	154.000000	566.000000	4174.936819	62.190813	27.308772
15	148.894737	681.853801	4174.936819	62.190813	27.308772
16	196.000000	740.000000	4174.936819	62.190813	27.308772
17	148.894737	681.853801	4174.936819	62.190813	27.308772
18	183.000000	708.000000	4174.936819	62.190813	27.308772
19	148.894737	681.853801	4174.936819	62.190813	27.308772
20	148.894737	681.853801	4174.936819	62.190813	27.308772
21	171.000000	812.000000	4174.936819	62.190813	27.308772
22	148.894737	681.853801	4174.936819	62.190813	27.308772
23	164.000000	1039.000000	8772.000000	62.190813	27.308772
24	149.000000	654.000000	4174.936819	62.190813	27.308772
25	154.000000	944.000000	4174.936819	62.190813	27.308772
26	148.894737	681.853801	4174.936819	62.190813	27.308772
27	148.894737	681.853801	4174.936819	62.190813	27.308772
28	148.894737	681.853801	4174.936819	62.190813	27.308772
29	148.894737	681.853801	4174.936819	62.190813	27.308772
30	89.000000	1062.000000	9431.000000	62.190813	27.308772
31	148.894737	681.853801	4174.936819	62.190813	27.308772
32	148.894737	681.853801	4174.936819	62.190813	27.308772
33	148.894737	681.853801	840.000000	63.000000	26.000000
34	148.894737	681.853801	1762.000000	73.000000	28.000000
35	148.894737	681.853801	4174.936819	62.190813	27.308772
36	106.000000	889.000000	10859.000000	62.190813	27.308772
37	148.894737	681.853801	4174.936819	62.190813	27.308772
38	148.894737	681.853801	4174.936819	62.190813	27.308772
39	148.894737	681.853801	1422.000000	70.000000	27.000000
40	148.894737	681.853801	4174.936819	62.190813	27.308772
41	148.894737	681.853801	1285.000000	64.000000	26.000000
42	148.894737	681.853801	4174.936819	62.190813	27.308772
43	148.894737	681.853801	4174.936819	62.190813	27.308772
44	148.894737	681.853801	4174.936819	62.190813	27.308772
45	14.000000	477.000000	2487.000000	62.190813	27.308772
46	148.894737	681.853801	4174.936819	62.190813	27.308772
47	148.894737	681.853801	4174.936819	62.190813	27.308772
48	148.894737	681.853801	4174.936819	62.190813	27.308772
49	148.894737	681.853801	1044.000000	35.000000	27.000000

	Number of Laps	Time_sec
0	1.0	2714.0
1	2.0	2177.0
2	1.0	2438.0
3	1.0	5779.0
4	1.0	1168.0
5	1.0	864.0
6	1.0	771.0
7	1.0	1633.0
8	1.0	3013.0

```
9              9.0    2524.0
10             1.0    2714.0
11             1.0    3619.0
12             1.0    3908.0
13             1.0    1668.0
14             1.0    2533.0
15             1.0    2096.0
16             1.0    1233.0
17            16.0    4825.0
18             1.0    3028.0
19             3.0     778.0
20             9.0    2858.0
21             1.0    2143.0
22             9.0    2558.0
23             6.0    6834.0
24             1.0    2715.0
25             1.0    1820.0
26             3.0     641.0
27             1.0    2730.0
28             6.0    1705.0
29             1.0    7823.0
30             6.0    7324.0
31             1.0    2503.0
32             1.0    2499.0
33             2.0    2382.0
34             4.0    4058.0
35            12.0    3482.0
36             8.0    9025.0
37             1.0    2766.0
38             1.0    2298.0
39             6.0    3324.0
40             9.0    2448.0
41             3.0    3299.0
42             1.0    2871.0
43             7.0    2099.0
44             4.0    1177.0
45             4.0    2015.0
46             5.0    6749.0
47             6.0    1954.0
48             6.0    1394.0
49             3.0    2780.0
```

[ ]:
```
#from sklearn.impute import SimpleImputer
#mean_imputation = eddy_numeric
#setting strategy to 'mean' to impute by the mean
#mean_imputer = SimpleImputer(strategy='mean')# strategy can also be mean or␣
 ↪median
```

```
#mean_imputation.iloc[:,:] = mean_imputer.fit_transform(mean_imputation)
```

[39]:
```
#regression imputaion
'''Select all the numeric columns for regression imputation'''
eddy_numeric_regr = eddy[numeric_cols]
'''Numeric columns with missing values which acts as target in training'''
target_cols = ['Distance','Calories','Avg HR','Max HR','Elev Gain','Elev␣
↪Loss','Avg Bike Cadence']
'''Predictors for regression imputation'''
predictors = eddy_numeric_regr.drop(target_cols, axis =1)

def find_missing_index(eddy_numeric_regr, target_cols):

    miss_index_dict = {}
    for tcol in target_cols:
        index = eddy_numeric_regr[tcol][eddy_numeric_regr[tcol].isnull()].index
        miss_index_dict[tcol] = index
    return miss_index_dict


def regression_imputation(eddy_numeric_regr, target_cols, miss_index_dict):

    for tcol in target_cols:
        y = eddy_numeric_regr[tcol]
        '''Initially impute the column with mean'''
        y = y.fillna(y.mean())
        xgb = xgboost.XGBRegressor(objective="reg:squarederror",␣
↪random_state=42)
        '''Fit the model where y is the target column which is to be imputed'''
        xgb.fit(predictors, y)
        predictions = pd.Series(xgb.predict(predictors),index= y.index)
        index = miss_index_dict[tcol]
        '''Replace the missing values with the predictions'''
        eddy_numeric_regr[tcol].loc[index] = predictions.loc[index]
    return eddy_numeric_regr

miss_index_dict = find_missing_index(eddy_numeric_regr, target_cols)
eddy_numeric_regr = regression_imputation(eddy_numeric_regr, target_cols,␣
↪miss_index_dict)
eddy_numeric_regr.head(10)
```

[39]:

	Distance	Calories	Avg HR	Max HR	Avg Speed	Max Speed	\
0	27.56	479.0	138.027252	166.677734	36.6	56.5	
1	14.08	398.0	142.026871	168.428284	23.3	30.2	
2	23.22	431.0	142.530960	173.605377	34.3	54.1	
3	50.56	838.0	135.568985	174.787567	31.5	59.0	
4	10.32	218.0	139.532608	171.437622	31.8	68.1	

5	7.77	129.0	137.691971	167.769913	32.4	52.3
6	8.12	191.0	139.438995	165.169220	37.9	51.9
7	16.77	315.0	140.636642	165.199066	37.0	63.1
8	30.96	527.0	152.000000	177.000000	37.0	56.3
9	8.02	468.0	146.000000	167.000000	NaN	NaN
10	27.58	498.0	140.914062	169.770004	36.6	64.0
11	36.45	598.0	137.537186	173.755692	36.3	55.3
12	30.08	654.0	161.000000	197.000000	27.7	71.1
13	17.09	334.0	158.000000	183.000000	36.9	57.2
14	20.96	374.0	134.153580	165.311554	29.8	57.3
15	0.00	223.0	115.000000	152.000000	NaN	NaN
16	12.16	235.0	151.000000	172.000000	35.5	49.1
17	15.03	982.0	159.000000	181.000000	NaN	NaN
18	30.30	537.0	158.000000	174.000000	36.0	55.0
19	2.35	161.0	148.000000	159.000000	NaN	NaN
20	8.67	518.0	145.000000	165.000000	NaN	NaN
21	20.20	356.0	136.063156	171.602783	33.9	53.8
22	8.05	469.0	143.000000	171.000000	NaN	NaN
23	54.42	1121.0	150.000000	187.000000	28.7	55.1
24	26.12	393.0	142.000000	168.000000	34.6	52.9
25	17.21	267.0	150.000000	193.000000	34.0	59.1
26	2.01	117.0	138.000000	155.000000	NaN	NaN
27	0.00	197.0	97.000000	147.000000	NaN	NaN
28	5.17	352.0	149.000000	163.000000	NaN	NaN
29	7.70	734.0	111.000000	166.000000	NaN	NaN
30	58.74	647.0	142.000000	176.000000	28.9	48.9
31	0.00	212.0	102.000000	149.000000	NaN	NaN
32	0.00	174.0	95.000000	139.000000	NaN	NaN
33	2200.00	537.0	135.897842	163.389801	NaN	NaN
34	3800.00	873.0	137.205811	167.522812	NaN	NaN
35	11.01	628.0	146.000000	164.000000	NaN	NaN
36	70.53	950.0	137.000000	175.000000	28.1	45.3
37	0.00	175.0	92.000000	141.000000	NaN	NaN
38	0.00	142.0	90.000000	143.000000	NaN	NaN
39	3300.00	760.0	138.311249	166.990326	NaN	NaN
40	8.01	439.0	144.000000	158.000000	NaN	NaN
41	3300.00	833.0	136.122757	164.251465	NaN	NaN
42	0.00	259.0	107.000000	142.000000	NaN	NaN
43	6.58	398.0	145.000000	162.000000	NaN	NaN
44	3.60	226.0	143.000000	157.000000	NaN	NaN
45	15.03	26.0	127.000000	150.000000	26.8	34.7
46	47.21	1749.0	137.213226	168.578659	25.2	44.6
47	5.50	407.0	153.000000	165.000000	NaN	NaN
48	5.01	283.0	148.000000	167.000000	NaN	NaN
49	4850.00	1186.0	134.679672	164.569092	NaN	NaN

	Elev Gain	Elev Loss	Avg Bike Cadence	Max Bike Cadence	\

0	80.000000	174.789108	87.000000	111.0
1	103.603691	129.713486	89.000000	127.0
2	89.000000	171.989853	85.000000	111.0
3	158.000000	244.408737	84.000000	125.0
4	92.000000	149.043137	92.000000	116.0
5	24.000000	127.447136	84.000000	123.0
6	37.000000	138.524612	87.000000	105.0
7	136.000000	164.594147	84.000000	98.0
8	102.000000	174.789108	80.000000	100.0
9	40.000000	36.000000	80.810928	NaN
10	80.000000	176.479218	89.000000	113.0
11	106.000000	148.219025	89.000000	127.0
12	262.000000	189.500229	86.000000	122.0
13	134.000000	183.228882	88.000000	106.0
14	219.000000	171.715576	91.000000	113.0
15	132.450211	138.289139	80.810928	NaN
16	31.000000	107.926537	81.000000	111.0
17	148.000000	164.000000	80.836647	NaN
18	102.000000	171.989853	85.000000	130.0
19	20.000000	4.000000	80.810928	NaN
20	50.000000	62.000000	80.810928	NaN
21	121.000000	142.318970	81.000000	121.0
22	31.000000	29.000000	80.810928	NaN
23	220.000000	239.000000	82.000000	115.0
24	82.000000	160.595642	84.000000	118.0
25	36.000000	149.171524	84.000000	135.0
26	21.000000	9.000000	80.810928	NaN
27	146.762177	150.195847	80.810928	NaN
28	24.000000	34.000000	80.810928	NaN
29	312.000000	293.000000	80.836647	NaN
30	212.000000	232.000000	82.000000	115.0
31	142.465820	150.195847	80.810928	NaN
32	142.465820	150.195847	80.810928	NaN
33	159.290817	162.339600	80.810928	NaN
34	140.067200	144.075729	80.810928	NaN
35	82.000000	64.000000	80.810928	NaN
36	209.000000	225.000000	80.000000	121.0
37	146.762177	150.195847	80.810928	NaN
38	132.450211	139.435303	80.810928	NaN
39	119.778580	114.671257	80.810928	NaN
40	11.000000	12.000000	80.810928	NaN
41	152.952972	152.972122	80.810928	NaN
42	146.762177	150.195847	80.810928	NaN
43	40.000000	41.000000	80.810928	NaN
44	11.000000	19.000000	80.810928	NaN
45	36.000000	28.000000	80.000000	101.0
46	184.000000	197.000000	80.531319	NaN

```
47   52.000000   74.000000         80.810928              NaN
48   11.000000   19.000000         80.810928              NaN
49  131.932022  120.576080         80.810928              NaN

    Normalized Power® (NP®)  Training Stress Score®  Max Avg Power (20 min)  \
0                     191.0                     0.0                   197.0
1                     195.0                    43.2                   195.0
2                     192.0                     0.0                   198.0
3                     167.0                     0.0                   166.0
4                     189.0                     0.0                     NaN
5                     179.0                     0.0                     NaN
6                     234.0                     0.0                     NaN
7                     205.0                     0.0                   210.0
8                     188.0                     0.0                   189.0
9                       NaN                     0.0                     NaN
10                    194.0                     0.0                   208.0
11                    181.0                     0.0                   182.0
12                    185.0                     0.0                   199.0
13                    222.0                     0.0                   220.0
14                    173.0                     0.0                   170.0
15                      NaN                     0.0                     NaN
16                    215.0                     0.0                   200.0
17                      NaN                     0.0                     NaN
18                    190.0                     0.0                   200.0
19                      NaN                     0.0                     NaN
20                      NaN                     0.0                     NaN
21                    201.0                     0.0                   188.0
22                      NaN                     0.0                     NaN
23                    189.0                   128.5                   188.0
24                    156.0                     0.0                   158.0
25                    214.0                     0.0                   160.0
26                      NaN                     0.0                     NaN
27                      NaN                     0.0                     NaN
28                      NaN                     0.0                     NaN
29                      NaN                     0.0                     NaN
30                    145.0                    80.4                   167.0
31                      NaN                     0.0                     NaN
32                      NaN                     0.0                     NaN
33                      NaN                     0.0                     NaN
34                      NaN                     0.0                     NaN
35                      NaN                     0.0                     NaN
36                    148.0                   103.9                   143.0
37                      NaN                     0.0                     NaN
38                      NaN                     0.0                     NaN
39                      NaN                     0.0                     NaN
40                      NaN                     0.0                     NaN
41                      NaN                     0.0                     NaN
```

42	NaN	0.0	NaN
43	NaN	0.0	NaN
44	NaN	0.0	NaN
45	39.0	1.6	12.0
46	NaN	0.0	NaN
47	NaN	0.0	NaN
48	NaN	0.0	NaN
49	NaN	0.0	NaN

	Avg Power	Max Power	Total Strokes	Avg. Swolf	Avg Stroke Rate	\
0	181.0	445.0	NaN	NaN	NaN	
1	183.0	623.0	3179.0	NaN	NaN	
2	180.0	620.0	NaN	NaN	NaN	
3	152.0	737.0	NaN	NaN	NaN	
4	183.0	647.0	NaN	NaN	NaN	
5	157.0	699.0	NaN	NaN	NaN	
6	228.0	580.0	NaN	NaN	NaN	
7	200.0	383.0	NaN	NaN	NaN	
8	178.0	420.0	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	
10	189.0	672.0	NaN	NaN	NaN	
11	171.0	855.0	NaN	NaN	NaN	
12	175.0	744.0	NaN	NaN	NaN	
13	208.0	565.0	NaN	NaN	NaN	
14	154.0	566.0	NaN	NaN	NaN	
15	NaN	NaN	NaN	NaN	NaN	
16	196.0	740.0	NaN	NaN	NaN	
17	NaN	NaN	NaN	NaN	NaN	
18	183.0	708.0	NaN	NaN	NaN	
19	NaN	NaN	NaN	NaN	NaN	
20	NaN	NaN	NaN	NaN	NaN	
21	171.0	812.0	NaN	NaN	NaN	
22	NaN	NaN	NaN	NaN	NaN	
23	164.0	1039.0	8772.0	NaN	NaN	
24	149.0	654.0	NaN	NaN	NaN	
25	154.0	944.0	NaN	NaN	NaN	
26	NaN	NaN	NaN	NaN	NaN	
27	NaN	NaN	NaN	NaN	NaN	
28	NaN	NaN	NaN	NaN	NaN	
29	NaN	NaN	NaN	NaN	NaN	
30	89.0	1062.0	9431.0	NaN	NaN	
31	NaN	NaN	NaN	NaN	NaN	
32	NaN	NaN	NaN	NaN	NaN	
33	NaN	NaN	840.0	63.0	26.0	
34	NaN	NaN	1762.0	73.0	28.0	
35	NaN	NaN	NaN	NaN	NaN	
36	106.0	889.0	10859.0	NaN	NaN	

37	NaN	NaN	NaN	NaN	NaN
38	NaN	NaN	NaN	NaN	NaN
39	NaN	NaN	1422.0	70.0	27.0
40	NaN	NaN	NaN	NaN	NaN
41	NaN	NaN	1285.0	64.0	26.0
42	NaN	NaN	NaN	NaN	NaN
43	NaN	NaN	NaN	NaN	NaN
44	NaN	NaN	NaN	NaN	NaN
45	14.0	477.0	2487.0	NaN	NaN
46	NaN	NaN	NaN	NaN	NaN
47	NaN	NaN	NaN	NaN	NaN
48	NaN	NaN	NaN	NaN	NaN
49	NaN	NaN	1044.0	35.0	27.0

	Number of Laps	Time_sec
0	1.0	2714.0
1	2.0	2177.0
2	1.0	2438.0
3	1.0	5779.0
4	1.0	1168.0
5	1.0	864.0
6	1.0	771.0
7	1.0	1633.0
8	1.0	3013.0
9	9.0	2524.0
10	1.0	2714.0
11	1.0	3619.0
12	1.0	3908.0
13	1.0	1668.0
14	1.0	2533.0
15	1.0	2096.0
16	1.0	1233.0
17	16.0	4825.0
18	1.0	3028.0
19	3.0	778.0
20	9.0	2858.0
21	1.0	2143.0
22	9.0	2558.0
23	6.0	6834.0
24	1.0	2715.0
25	1.0	1820.0
26	3.0	641.0
27	1.0	2730.0
28	6.0	1705.0
29	1.0	7823.0
30	6.0	7324.0
31	1.0	2503.0

```
32             1.0    2499.0
33             2.0    2382.0
34             4.0    4058.0
35            12.0    3482.0
36             8.0    9025.0
37             1.0    2766.0
38             1.0    2298.0
39             6.0    3324.0
40             9.0    2448.0
41             3.0    3299.0
42             1.0    2871.0
43             7.0    2099.0
44             4.0    1177.0
45             4.0    2015.0
46             5.0    6749.0
47             6.0    1954.0
48             6.0    1394.0
49             3.0    2780.0
```

[40]:
```
def mode_imputation(eddy_categoric):
    """
    Mode Imputation
    """
    for col in eddy_categoric.columns:
        mode = eddy_categoric[col].mode().iloc[0]
        eddy_categoric[col] = eddy_categoric[col].fillna(mode)
    return eddy_categoric

eddy_mode_imp = mode_imputation(eddy_categoric)
'''Concatenate the mean and mode imputed columns'''
#eddy_imputed = pd.concat([eddy_mean_imp, eddy_mode_imp], axis = 1)
#eddy_imputed.head()
eddy_categoric.head()
```

[40]:
```
     Activity Type                Date  \
0  virtual cycling 2020-04-06 18:15:01
1   indoor cycling 2020-04-05 17:00:02
2  virtual cycling 2020-04-05 16:00:01
3  virtual cycling 2020-04-04 06:59:59
4  virtual cycling 2020-04-03 18:00:28

                                        Title      Time Date_extracted  \
0           zwift - tbr knights of suburbia (d)  00:45:14     2020-04-06
1                                indoor cycling  00:36:17     2020-04-05
2                          zwift - ahdr bbq (d)  00:40:38     2020-04-05
3  zwift - scott d'aucourt's meetup - tick tock  01:36:19     2020-04-04
4      zwift -  haute route watopia stage 1 (e)  00:19:28     2020-04-03
```

```
  Time_extracted
0       18:15:01
1       17:00:02
2       16:00:01
3       06:59:59
4       18:00:28
```

[43]:
```
def mice_imputation_numeric(eddy_numeric):
    iter_imp_numeric = IterativeImputer(GradientBoostingRegressor())
    imputed_eddy = iter_imp_numeric.fit_transform(eddy_numeric)
    eddy_numeric_imp = pd.DataFrame(imputed_eddy, columns = eddy_numeric.
 ↪columns, index= eddy_numeric.index)
    return eddy_numeric_imp
eddy_numeric_imp  = mice_imputation_numeric(eddy_numeric)
```

```
C:\Users\Spoorthi\AppData\Roaming\Python\Python37\site-
packages\sklearn\impute\_iterative.py:638: ConvergenceWarning:
[IterativeImputer] Early stopping criterion not reached.
  " reached.", ConvergenceWarning)
```

[44]:
```
eddy_numeric_imp.head(10)
```

[44]:

	Distance	Calories	Avg HR	Max HR	Avg Speed	Max Speed	\
0	27.56	479.0	149.963530	174.480240	36.600000	56.500000	
1	14.08	398.0	149.750049	170.838755	23.300000	30.200000	
2	23.22	431.0	150.184801	173.300790	34.300000	54.100000	
3	50.56	838.0	137.147297	177.526504	31.500000	59.000000	
4	10.32	218.0	151.429885	172.178798	31.800000	68.100000	
5	7.77	129.0	139.890536	160.076423	32.400000	52.300000	
6	8.12	191.0	155.464285	175.273392	37.900000	51.900000	
7	16.77	315.0	156.214415	179.536288	37.000000	63.100000	
8	30.96	527.0	152.000000	177.000000	37.000000	56.300000	
9	8.02	468.0	146.000000	167.000000	25.009947	37.470998	
10	27.58	498.0	152.421547	174.613079	36.600000	64.000000	
11	36.45	598.0	147.869685	178.937452	36.300000	55.300000	
12	30.08	654.0	161.000000	197.000000	27.700000	71.100000	
13	17.09	334.0	158.000000	183.000000	36.900000	57.200000	
14	20.96	374.0	154.475216	176.744183	29.800000	57.300000	
15	0.00	223.0	115.000000	152.000000	22.242752	33.520983	
16	12.16	235.0	151.000000	172.000000	35.500000	49.100000	
17	15.03	982.0	159.000000	181.000000	21.343731	41.942773	
18	30.30	537.0	158.000000	174.000000	36.000000	55.000000	
19	2.35	161.0	148.000000	159.000000	25.745919	40.389172	
20	8.67	518.0	145.000000	165.000000	21.787279	36.826110	
21	20.20	356.0	145.880431	177.412873	33.900000	53.800000	
22	8.05	469.0	143.000000	171.000000	23.038771	36.135163	

23	54.42	1121.0	150.000000	187.000000	28.700000	55.100000
24	26.12	393.0	142.000000	168.000000	34.600000	52.900000
25	17.21	267.0	150.000000	193.000000	34.000000	59.100000
26	2.01	117.0	138.000000	155.000000	25.089648	41.031486
27	0.00	197.0	97.000000	147.000000	15.061626	32.910051
28	5.17	352.0	149.000000	163.000000	27.699229	46.446617
29	7.70	734.0	111.000000	166.000000	22.414017	43.863097
30	58.74	647.0	142.000000	176.000000	28.900000	48.900000
31	0.00	212.0	102.000000	149.000000	22.965102	39.574991
32	0.00	174.0	95.000000	139.000000	21.995880	37.090371
33	2200.00	537.0	107.849410	115.790662	33.418761	53.270118
34	3800.00	873.0	140.575741	165.901463	34.866358	60.694463
35	11.01	628.0	146.000000	164.000000	25.587312	44.193480
36	70.53	950.0	137.000000	175.000000	28.100000	45.300000
37	0.00	175.0	92.000000	141.000000	15.172467	34.438280
38	0.00	142.0	90.000000	143.000000	20.454754	33.520983
39	3300.00	760.0	140.701562	164.514743	34.439509	60.861283
40	8.01	439.0	144.000000	158.000000	24.613159	36.053534
41	3300.00	833.0	140.280621	164.520588	33.087994	60.861283
42	0.00	259.0	107.000000	142.000000	19.541108	34.249690
43	6.58	398.0	145.000000	162.000000	27.142091	46.906383
44	3.60	226.0	143.000000	157.000000	23.412964	37.498884
45	15.03	26.0	127.000000	150.000000	26.800000	34.700000
46	47.21	1749.0	141.719015	172.539517	25.200000	44.600000
47	5.50	407.0	153.000000	165.000000	29.338876	47.833844
48	5.01	283.0	148.000000	167.000000	28.445677	49.635744
49	4850.00	1186.0	141.863975	160.478879	34.306358	60.778102

	Elev Gain	Elev Loss	Avg Bike Cadence	Max Bike Cadence	\
0	80.000000	63.818326	87.000000	111.000000	
1	9.962020	11.638373	89.000000	127.000000	
2	89.000000	77.714031	85.000000	111.000000	
3	158.000000	157.293600	84.000000	125.000000	
4	92.000000	62.952900	92.000000	116.000000	
5	24.000000	20.534139	84.000000	123.000000	
6	37.000000	27.484898	87.000000	105.000000	
7	136.000000	97.751029	84.000000	98.000000	
8	102.000000	94.332248	80.000000	100.000000	
9	40.000000	36.000000	87.638286	118.855865	
10	80.000000	41.660189	89.000000	113.000000	
11	106.000000	105.674302	89.000000	127.000000	
12	262.000000	222.471066	86.000000	122.000000	
13	134.000000	111.107792	88.000000	106.000000	
14	219.000000	201.566641	91.000000	113.000000	
15	59.459405	56.907699	88.110365	116.845307	
16	31.000000	33.743905	81.000000	111.000000	
17	148.000000	164.000000	59.382200	99.038893	

18	102.000000	99.443836	85.000000	130.000000
19	20.000000	4.000000	88.086236	112.028322
20	50.000000	62.000000	87.898882	119.619542
21	121.000000	113.161483	81.000000	121.000000
22	31.000000	29.000000	87.782349	112.342069
23	220.000000	239.000000	82.000000	115.000000
24	82.000000	65.304658	84.000000	118.000000
25	36.000000	35.714088	84.000000	135.000000
26	21.000000	9.000000	77.323982	97.398333
27	54.606655	54.027057	72.603103	99.695205
28	24.000000	34.000000	91.433849	118.626720
29	312.000000	293.000000	81.148026	129.638151
30	212.000000	232.000000	82.000000	115.000000
31	62.514129	53.557044	84.711265	113.440500
32	59.789722	53.557044	89.220055	114.548611
33	17.842256	19.520512	83.815722	102.596965
34	53.794636	58.316968	79.953787	95.993992
35	82.000000	64.000000	88.048948	119.957428
36	209.000000	225.000000	80.000000	121.000000
37	54.606655	54.027057	76.287244	103.363495
38	59.304059	53.557044	89.973125	114.548611
39	53.794636	56.792723	80.172921	95.993992
40	11.000000	12.000000	87.578830	121.059698
41	54.091540	58.382969	80.995568	102.712064
42	57.750774	54.646943	73.727129	102.901459
43	40.000000	41.000000	86.882446	121.055062
44	11.000000	19.000000	78.369187	100.036020
45	36.000000	28.000000	80.000000	101.000000
46	184.000000	197.000000	77.581219	102.352431
47	52.000000	74.000000	88.025064	116.390429
48	11.000000	19.000000	86.119828	118.179312
49	52.874905	59.408324	77.232948	90.521635

	Normalized Power® (NP®)	Training Stress Score®	Max Avg Power (20 min)	\
0	191.000000	0.0	197.000000	
1	195.000000	43.2	195.000000	
2	192.000000	0.0	198.000000	
3	167.000000	0.0	166.000000	
4	189.000000	0.0	187.722481	
5	179.000000	0.0	161.168372	
6	234.000000	0.0	225.720606	
7	205.000000	0.0	210.000000	
8	188.000000	0.0	189.000000	
9	190.325759	0.0	198.345664	
10	194.000000	0.0	208.000000	
11	181.000000	0.0	182.000000	
12	185.000000	0.0	199.000000	

13	222.000000	0.0	220.000000
14	173.000000	0.0	170.000000
15	166.711720	0.0	157.935477
16	215.000000	0.0	200.000000
17	151.260559	0.0	145.216320
18	190.000000	0.0	200.000000
19	176.171439	0.0	173.460274
20	186.001730	0.0	195.603162
21	201.000000	0.0	188.000000
22	191.749302	0.0	197.552215
23	189.000000	128.5	188.000000
24	156.000000	0.0	158.000000
25	214.000000	0.0	160.000000
26	159.191468	0.0	138.770464
27	98.955132	0.0	60.487706
28	209.691909	0.0	207.878251
29	156.079762	0.0	168.644292
30	145.000000	80.4	167.000000
31	154.698627	0.0	157.981895
32	167.484027	0.0	164.062682
33	173.263332	0.0	178.555014
34	174.066312	0.0	181.726086
35	169.749275	0.0	184.329310
36	148.000000	103.9	143.000000
37	98.622069	0.0	57.796188
38	167.204888	0.0	158.193496
39	174.407816	0.0	181.896975
40	197.701055	0.0	196.216988
41	173.446961	0.0	181.896975
42	105.752144	0.0	72.206697
43	181.865359	0.0	181.217754
44	175.389803	0.0	164.582384
45	39.000000	1.6	12.000000
46	155.635289	0.0	177.725773
47	204.498035	0.0	207.410892
48	195.253527	0.0	187.700323
49	176.218447	0.0	183.894001

	Avg Power	Max Power	Total Strokes	Avg. Swolf	Avg Stroke Rate	\
0	181.000000	445.000000	2960.508240	132.138680	31.962284	
1	183.000000	623.000000	3179.000000	119.036794	20.539323	
2	180.000000	620.000000	2759.521378	128.342000	28.172992	
3	152.000000	737.000000	6156.207760	130.895218	23.695961	
4	183.000000	647.000000	2049.832566	118.979804	24.735058	
5	157.000000	699.000000	1853.213449	124.410299	24.998335	
6	228.000000	580.000000	1969.512511	165.408232	32.125085	
7	200.000000	383.000000	2386.742740	122.913202	31.716219	

8	178.000000	420.000000	3083.962766	134.864268	31.870291
9	178.764361	498.284441	3766.190849	137.396491	19.977726
10	189.000000	672.000000	2870.696213	127.938412	31.893968
11	171.000000	855.000000	4327.322306	131.285382	31.681410
12	175.000000	744.000000	4226.470815	127.841069	22.426197
13	208.000000	565.000000	2632.426715	125.169338	31.864290
14	154.000000	566.000000	2882.969428	130.188157	22.514978
15	157.105361	401.354164	2797.912428	142.114972	19.652490
16	196.000000	740.000000	1988.389519	121.014855	30.806082
17	103.785243	291.825695	3573.831507	120.448374	12.864848
18	183.000000	708.000000	3172.260825	125.801479	30.991951
19	167.992889	467.570631	2138.075924	139.590650	20.159837
20	174.211711	494.923662	3992.263527	142.795628	19.977295
21	171.000000	812.000000	2776.320954	119.548307	27.363326
22	178.278473	481.474337	3929.558870	138.711188	20.051097
23	164.000000	1039.000000	8772.000000	132.094949	22.080014
24	149.000000	654.000000	3198.762129	133.107473	28.824735
25	154.000000	944.000000	2364.086894	119.053831	27.425183
26	151.971215	232.534682	1914.328260	137.313639	18.137916
27	57.112476	418.397324	1972.948730	117.961021	15.582776
28	199.895527	449.089320	2969.353675	142.871876	20.954748
29	142.255442	537.059595	6937.533508	136.763368	19.684193
30	89.000000	1062.000000	9431.000000	127.674651	21.208859
31	150.473688	494.223859	2711.989075	139.867880	19.663745
32	157.446041	421.922521	2725.876389	137.575201	19.999539
33	161.342410	429.658490	840.000000	63.000000	26.000000
34	161.996149	406.130507	1762.000000	73.000000	28.000000
35	166.099859	489.347507	4855.564371	145.432288	19.980765
36	106.000000	889.000000	10859.000000	137.319118	20.875830
37	57.112476	499.946246	1967.650383	116.238170	16.698779
38	145.373623	384.929688	2760.236641	137.575201	20.024779
39	160.868231	406.130507	1422.000000	70.000000	27.000000
40	178.896611	551.904728	3868.032119	120.669122	20.395557
41	161.280344	482.717784	1285.000000	64.000000	26.000000
42	85.539082	499.946246	2306.329271	121.578501	15.524898
43	170.120722	542.229861	3122.559769	147.846042	20.633188
44	156.714231	336.728090	2242.244391	136.339090	18.632293
45	14.000000	477.000000	2487.000000	162.447764	17.949789
46	145.931017	445.353609	6255.108941	139.126662	19.684589
47	197.384911	434.822191	2843.272288	144.582933	20.884792
48	183.012287	660.486014	2407.016828	112.517329	21.346275
49	162.292287	391.753763	1044.000000	35.000000	27.000000

	Number of Laps	Time_sec
0	1.0	2714.0
1	2.0	2177.0
2	1.0	2438.0

3	1.0	5779.0
4	1.0	1168.0
5	1.0	864.0
6	1.0	771.0
7	1.0	1633.0
8	1.0	3013.0
9	9.0	2524.0
10	1.0	2714.0
11	1.0	3619.0
12	1.0	3908.0
13	1.0	1668.0
14	1.0	2533.0
15	1.0	2096.0
16	1.0	1233.0
17	16.0	4825.0
18	1.0	3028.0
19	3.0	778.0
20	9.0	2858.0
21	1.0	2143.0
22	9.0	2558.0
23	6.0	6834.0
24	1.0	2715.0
25	1.0	1820.0
26	3.0	641.0
27	1.0	2730.0
28	6.0	1705.0
29	1.0	7823.0
30	6.0	7324.0
31	1.0	2503.0
32	1.0	2499.0
33	2.0	2382.0
34	4.0	4058.0
35	12.0	3482.0
36	8.0	9025.0
37	1.0	2766.0
38	1.0	2298.0
39	6.0	3324.0
40	9.0	2448.0
41	3.0	3299.0
42	1.0	2871.0
43	7.0	2099.0
44	4.0	1177.0
45	4.0	2015.0
46	5.0	6749.0
47	6.0	1954.0
48	6.0	1394.0
49	3.0	2780.0

[45]:
```
def mice_imputation_categoric(eddy_categoric):
    ordinal_dict={}
    for col in eddy_categoric:
        ordinal_dict[col] = OrdinalEncoder()
        nn_vals = np.array(eddy_categoric[col][eddy_categoric[col].notnull()]).
 ↪reshape(-1,1)
        nn_vals_arr = np.array(ordinal_dict[col].fit_transform(nn_vals)).
 ↪reshape(-1,)
        eddy_categoric[col].loc[eddy_categoric[col].notnull()] = nn_vals_arr
    '''Impute the data using MICE with Gradient Boosting Classifier'''
    iter_imp_categoric = IterativeImputer(GradientBoostingClassifier(),␣
 ↪max_iter =5, initial_strategy='most_frequent')
    imputed_eddy = iter_imp_categoric.fit_transform(eddy_categoric)
    eddy_categoric_imp = pd.DataFrame(imputed_eddy, columns =eddy_categoric.
 ↪columns,index = eddy_categoric.index).astype(int)
    '''Inverse Transform'''
    for col in eddy_categoric_imp.columns:
        oe = ordinal_dict[col]
        eddy_arr= np.array(eddy_categoric_imp[col]).reshape(-1,1)
        eddy_categoric_imp[col] = oe.inverse_transform(eddy_arr)
    return eddy_categoric_imp
#eddy_categoric_imp = mice_imputation_categoric(eddy_categoric)

#'''Concatenate Numeric and Categoric Training and Test set data '''
#eddy_mice_imp = pd.join([eddy_numeric_imp, eddy_categoric_imp], axis = 1)
#eddy_mice_imp.head()
```

[46]:
```
def Linear_interpolation(eddy_numeric):
    for col in eddy_numeric.columns:
        numeric = eddy_numeric.interpolate(method='linear',␣
 ↪limit_direction='forward', axis=0).ffill().bfill()
    return(numeric)
eddy_Linearinterpolation = Linear_interpolation(eddy_numeric)
```

[47]:
```
print(eddy_Linearinterpolation.head())
```

[47]:

	Distance	Calories	Avg HR	Max HR	Avg Speed	Max Speed	Elev Gain	\
0	27.56	479.0	152.0	177.0	36.60	56.5	80.0	
1	14.08	398.0	152.0	177.0	23.30	30.2	84.5	
2	23.22	431.0	152.0	177.0	34.30	54.1	89.0	
3	50.56	838.0	152.0	177.0	31.50	59.0	158.0	
4	10.32	218.0	152.0	177.0	31.80	68.1	92.0	
…	…	…	…	…	…	…	…	
1135	3.83	118.0	144.0	153.0	24.60	31.4	2.0	
1136	1.84	153.0	168.0	183.0	23.65	33.2	1.0	
1137	8.13	198.0	125.0	147.0	22.70	35.0	2.0	
1138	35.83	1725.0	125.0	186.0	17.40	35.0	180.0	

```
1139    411.00      98.0   125.0   186.0      17.40       35.0      180.0

      Elev Loss  Avg Bike Cadence  Max Bike Cadence  Normalized Power® (NP®)  \
0          36.0              87.0             111.0                    191.0
1          36.0              89.0             127.0                    195.0
2          36.0              85.0             111.0                    192.0
3          36.0              84.0             125.0                    167.0
4          36.0              92.0             116.0                    189.0
…           …                 …                …                        …
1135        2.5              80.0              92.0                    165.0
1136        3.0              84.0             103.5                    165.0
1137        2.0              88.0             115.0                    165.0
1138      170.0              88.0             115.0                    165.0
1139       10.0              88.0             115.0                    165.0

      Training Stress Score®  Max Avg Power (20 min)  Avg Power  Max Power  \
0                        0.0                   197.0      181.0      445.0
1                       43.2                   195.0      183.0      623.0
2                        0.0                   198.0      180.0      620.0
3                        0.0                   166.0      152.0      737.0
4                        0.0                   177.0      183.0      647.0
…                         …                       …          …          …
1135                     0.0                   136.0      121.0      526.0
1136                     0.0                   136.0      121.0      526.0
1137                     0.0                   136.0      121.0      526.0
1138                     0.0                   136.0      121.0      526.0
1139                     0.0                   136.0      121.0      526.0

      Total Strokes  Avg. Swolf  Avg Stroke Rate  Number of Laps  Time_sec
0       3179.000000        63.0             26.0             1.0    2714.0
1       3179.000000        63.0             26.0             2.0    2177.0
2       3433.227273        63.0             26.0             1.0    2438.0
3       3687.454545        63.0             26.0             1.0    5779.0
4       3941.681818        63.0             26.0             1.0    1168.0
…               …           …                …               …         …
1135     750.000000        51.2             23.6             1.0     559.0
1136    1308.500000        52.4             24.2             1.0     627.0
1137    1867.000000        53.6             24.8             1.0    1291.0
1138    1075.500000        54.8             25.4             1.0    7426.0
1139     284.000000        56.0             26.0             1.0     634.0

[1140 rows x 20 columns]
```

[43]:
```
print(eddy_numeric.head())
```

[43]:
```
      Distance  Calories  Avg HR  Max HR  Avg Speed  Max Speed  Elev Gain  \
0        27.56     479.0     NaN     NaN       36.6       56.5       80.0
```

```
1        14.08     398.0     NaN     NaN       23.3       30.2        NaN
2        23.22     431.0     NaN     NaN       34.3       54.1       89.0
3        50.56     838.0     NaN     NaN       31.5       59.0      158.0
4        10.32     218.0     NaN     NaN       31.8       68.1       92.0
…        …       …     …     …       …         …       …
1135      3.83     118.0   144.0   153.0       24.6       31.4        2.0
1136      1.84     153.0   168.0   183.0        NaN        NaN        1.0
1137      8.13     198.0   125.0   147.0       22.7       35.0        2.0
1138     35.83    1725.0     NaN   186.0       17.4        NaN      180.0
1139    411.00      98.0     NaN     NaN        NaN        NaN        NaN

      Elev Loss  Avg Bike Cadence  Max Bike Cadence  Normalized Power® (NP®)  \
0           NaN              87.0             111.0                    191.0
1           NaN              89.0             127.0                    195.0
2           NaN              85.0             111.0                    192.0
3           NaN              84.0             125.0                    167.0
4           NaN              92.0             116.0                    189.0
…         …                …                …                      …
1135        NaN              80.0              92.0                      NaN
1136        3.0               NaN               NaN                      NaN
1137        2.0              88.0             115.0                      NaN
1138      170.0               NaN               NaN                      NaN
1139       10.0               NaN               NaN                      NaN

      Training Stress Score®  Max Avg Power (20 min)  Avg Power  Max Power  \
0                        0.0                   197.0      181.0      445.0
1                       43.2                   195.0      183.0      623.0
2                        0.0                   198.0      180.0      620.0
3                        0.0                   166.0      152.0      737.0
4                        0.0                     NaN      183.0      647.0
…                      …                       …        …        …
1135                     0.0                     NaN        NaN        NaN
1136                     0.0                     NaN        NaN        NaN
1137                     0.0                     NaN        NaN        NaN
1138                     0.0                     NaN        NaN        NaN
1139                     0.0                     NaN        NaN        NaN

      Total Strokes  Avg. Swolf  Avg Stroke Rate  Number of Laps  Time_sec
0               NaN         NaN              NaN             1.0    2714.0
1            3179.0         NaN              NaN             2.0    2177.0
2               NaN         NaN              NaN             1.0    2438.0
3               NaN         NaN              NaN             1.0    5779.0
4               NaN         NaN              NaN             1.0    1168.0
…             …         …              …             …       …
1135          750.0         NaN              NaN             NaN     559.0
1136            NaN         NaN              NaN             NaN     627.0
1137         1867.0         NaN              NaN             NaN    1291.0
```

```
1138            NaN         NaN              NaN             NaN    7426.0
1139          284.0        56.0             26.0             NaN     634.0

[1140 rows x 20 columns]
```

[46]:
```
from sklearn.preprocessing import MinMaxScaler#when imputing a knn data must be␣
↪normalised to reduce the bias in the imputation
scaler = MinMaxScaler()
scaling = pd.DataFrame(scaler.fit_transform(eddy_numeric), columns =␣
↪numeric_cols)
eddy.head()
```

[46]:
```
     Activity Type                Date  \
0  virtual cycling 2020-04-06 18:15:01
1   indoor cycling 2020-04-05 17:00:02
2  virtual cycling 2020-04-05 16:00:01
3  virtual cycling 2020-04-04 06:59:59
4  virtual cycling 2020-04-03 18:00:28

                                      Title  Distance  Calories      Time  \
0          zwift - tbr knights of suburbia (d)     27.56     479.0  00:45:14
1                              indoor cycling     14.08     398.0  00:36:17
2                        zwift - ahdr bbq (d)     23.22     431.0  00:40:38
3  zwift - scott d'aucourt's meetup - tick tock     50.56     838.0  01:36:19
4      zwift -  haute route watopia stage 1 (e)     10.32     218.0  00:19:28

   Avg HR  Max HR  Avg Speed  Max Speed  Elev Gain  Elev Loss  \
0     NaN     NaN       36.6       56.5       80.0        NaN
1     NaN     NaN       23.3       30.2        NaN        NaN
2     NaN     NaN       34.3       54.1       89.0        NaN
3     NaN     NaN       31.5       59.0      158.0        NaN
4     NaN     NaN       31.8       68.1       92.0        NaN

   Avg Bike Cadence  Max Bike Cadence  Normalized Power® (NP®)  \
0              87.0             111.0                    191.0
1              89.0             127.0                    195.0
2              85.0             111.0                    192.0
3              84.0             125.0                    167.0
4              92.0             116.0                    189.0

   Training Stress Score®  Max Avg Power (20 min)  Avg Power  Max Power  \
0                     0.0                   197.0      181.0      445.0
1                    43.2                   195.0      183.0      623.0
2                     0.0                   198.0      180.0      620.0
3                     0.0                   166.0      152.0      737.0
4                     0.0                     NaN      183.0      647.0
```

```
   Total Strokes  Avg. Swolf  Avg Stroke Rate  Number of Laps Date_extracted  \
0            NaN         NaN              NaN             1.0     2020-04-06
1         3179.0         NaN              NaN             2.0     2020-04-05
2            NaN         NaN              NaN             1.0     2020-04-05
3            NaN         NaN              NaN             1.0     2020-04-04
4            NaN         NaN              NaN             1.0     2020-04-03

  Time_extracted  Time_sec
0       18:15:01    2714.0
1       17:00:02    2177.0
2       16:00:01    2438.0
3       06:59:59    5779.0
4       18:00:28    1168.0
```

[48]:
```
def knn_imputation():
    imputer = KNNImputer(n_neighbors = 23)
    imputed_KNN = pd.DataFrame(imputer.fit_transform(eddy_numeric),columns =␣
 ↪numeric_cols)
    return imputed_KNN
knn_imputation().head()
```

[48]:
```
      Distance  Calories      Avg HR      Max HR  Avg Speed  Max Speed  \
0        27.56     479.0  141.478261  165.956522  36.600000  56.500000
1        14.08     398.0  144.391304  169.260870  23.300000  30.200000
2        23.22     431.0  145.652174  168.739130  34.300000  54.100000
3        50.56     838.0  144.217391  169.173913  31.500000  59.000000
4        10.32     218.0  146.043478  167.086957  31.800000  68.100000
…          …         …           …           …          …          …
1135      3.83     118.0  144.000000  153.000000  24.600000  31.400000
1136      1.84     153.0  168.000000  183.000000  25.752174  42.965217
1137      8.13     198.0  125.000000  147.000000  22.700000  35.000000
1138     35.83    1725.0  144.565217  186.000000  17.400000  47.004348
1139    411.00      98.0  142.304348  160.782609  26.265217  44.665217

       Elev Gain   Elev Loss  Avg Bike Cadence  Max Bike Cadence  \
0      80.000000   51.391304         87.000000        111.000000
1      71.086957   32.826087         89.000000        127.000000
2      89.000000   28.304348         85.000000        111.000000
3     158.000000  156.391304         84.000000        125.000000
4      92.000000   13.478261         92.000000        116.000000
…            …           …                 …                …
1135    2.000000    4.739130         80.000000         92.000000
1136    1.000000    3.000000         82.391304        108.347826
1137    2.000000    2.000000         88.000000        115.000000
1138  180.000000  170.000000         77.000000        118.043478
1139   12.478261   10.000000         81.565217        108.434783
```

```
      Normalized Power® (NP®)  Training Stress Score®  Max Avg Power (20 min)  \
0                  191.000000                     0.0              197.000000
1                  195.000000                    43.2              195.000000
2                  192.000000                     0.0              198.000000
3                  167.000000                     0.0              166.000000
4                  189.000000                     0.0              174.782609
…                         …                       …                       …
1135               184.478261                     0.0              177.391304
1136               179.956522                     0.0              169.347826
1137               192.739130                     0.0              167.173913
1138               157.000000                     0.0              162.826087
1139               183.739130                     0.0              180.782609

       Avg Power   Max Power  Total Strokes  Avg. Swolf  Avg Stroke Rate  \
0     181.000000  445.000000    3235.521739   55.173913        25.217391
1     183.000000  623.000000    3179.000000   45.260870        27.173913
2     180.000000  620.000000    2912.086957   61.869565        25.521739
3     152.000000  737.000000    6595.869565   65.391304        25.173913
4     183.000000  647.000000    1620.217391   60.608696        25.217391
…           …         …             …          …                …
1135  167.304348  573.391304     750.000000   60.608696        25.130435
1136  163.652174  554.608696     904.913043   60.608696        25.217391
1137  174.130435  622.347826    1867.000000   52.956522        26.913043
1138  125.608696  772.695652    8286.652174   63.565217        25.434783
1139  166.956522  565.478261     284.000000   56.000000        26.000000

      Number of Laps  Time_sec
0           1.000000    2714.0
1           2.000000    2177.0
2           1.000000    2438.0
3           1.000000    5779.0
4           1.000000    1168.0
…              …        …
1135        2.260870     559.0
1136        2.304348     627.0
1137        3.956522    1291.0
1138       12.173913    7426.0
1139        2.130435     634.0

[1140 rows x 20 columns]
```

[49]:
```
eddy = knn_imputation()
eddy.isna().any()
```

[49]:
```
Distance                 False
Calories                 False
Avg HR                   False
```

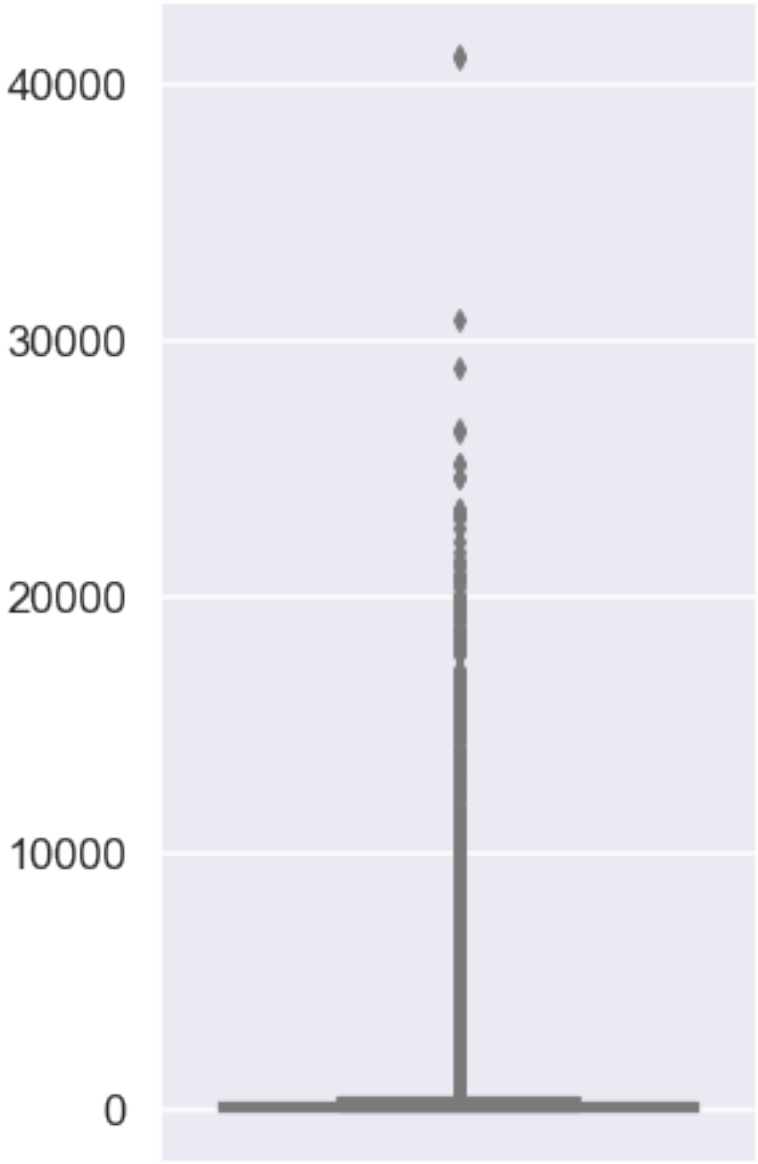
```
Max HR                      False
Avg Speed                   False
Max Speed                   False
Elev Gain                   False
Elev Loss                   False
Avg Bike Cadence            False
Max Bike Cadence            False
Normalized Power® (NP®)     False
Training Stress Score®      False
Max Avg Power (20 min)      False
Avg Power                   False
Max Power                   False
Total Strokes               False
Avg. Swolf                  False
Avg Stroke Rate             False
Number of Laps              False
Time_sec                    False
dtype: bool
```

[50]:
```
eddy.isna().sum()
```

[50]:
```
Distance                    0
Calories                    0
Avg HR                      0
Max HR                      0
Avg Speed                   0
Max Speed                   0
Elev Gain                   0
Elev Loss                   0
Avg Bike Cadence            0
Max Bike Cadence            0
Normalized Power® (NP®)     0
Training Stress Score®      0
Max Avg Power (20 min)      0
Avg Power                   0
Max Power                   0
Total Strokes               0
Avg. Swolf                  0
Avg Stroke Rate             0
Number of Laps              0
Time_sec                    0
dtype: int64
```

[64]:
```
plt.figure(figsize = (4,8))
sns.boxplot(y = eddy)
```

[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21c9e01bba8>
```



[51]:
```
def out_iqr(eddy , column):
    global lower,upper
    q25, q75 = np.quantile(eddy[column], 0.25), np.quantile(eddy[column], 0.75)
    # calculate the IQR
    iqr = q75 - q25
    # calculate the outlier cutoff
    cut_off = iqr * 1.5
    # calculate the lower and upper bound value
    lower, upper = q25 - cut_off, q75 + cut_off
```

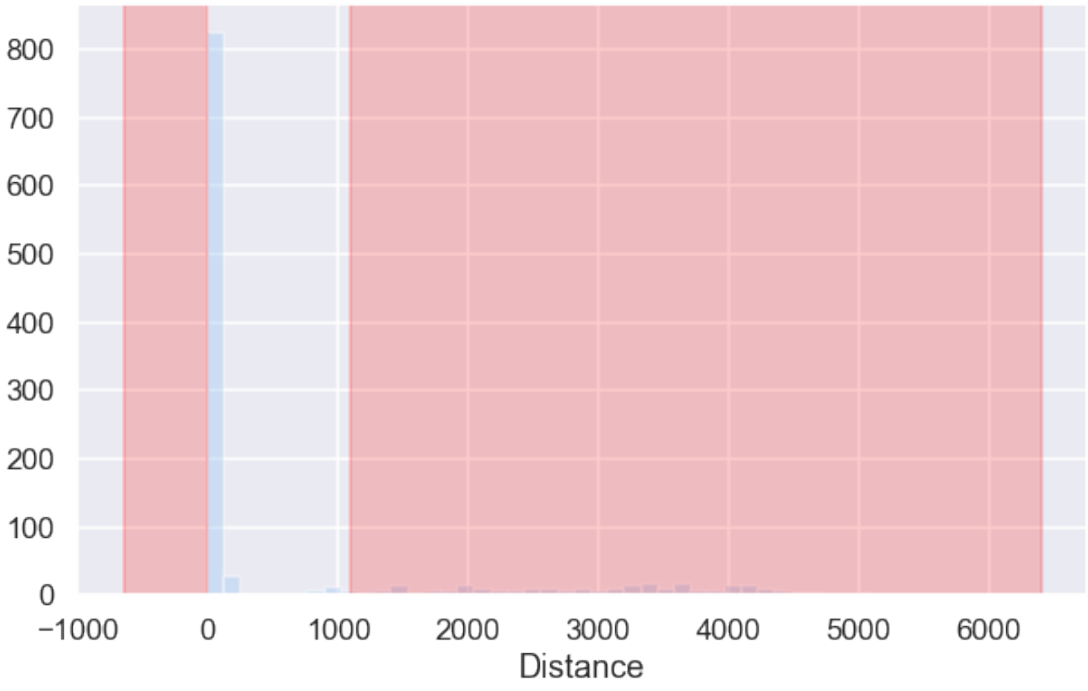
```
    print('The IQR is',iqr)
    print('The lower bound value is', lower)
    print('The upper bound value is', upper)
    # Calculate the number of records below and above lower and above bound␣
↪value respectively
    df1 = eddy[eddy[column] > upper]
    df2 = eddy[eddy[column] < lower]
    return print('Total number of outliers are', df1.shape[0]+ df2.shape[0])
```

[52]:
```
out_iqr(eddy,'Distance')
```

```
The IQR is 434.7225
The lower bound value is -644.80625
The upper bound value is 1094.08375
Total number of outliers are 261
```

[53]:
```
plt.figure(figsize = (10,6))
sns.distplot(eddy.Distance, kde=False)
plt.axvspan(xmin = lower,xmax= eddy.Distance.min(),alpha=0.2, color='red')
plt.axvspan(xmin = upper,xmax= eddy.Distance.max(),alpha=0.2, color='red')
```

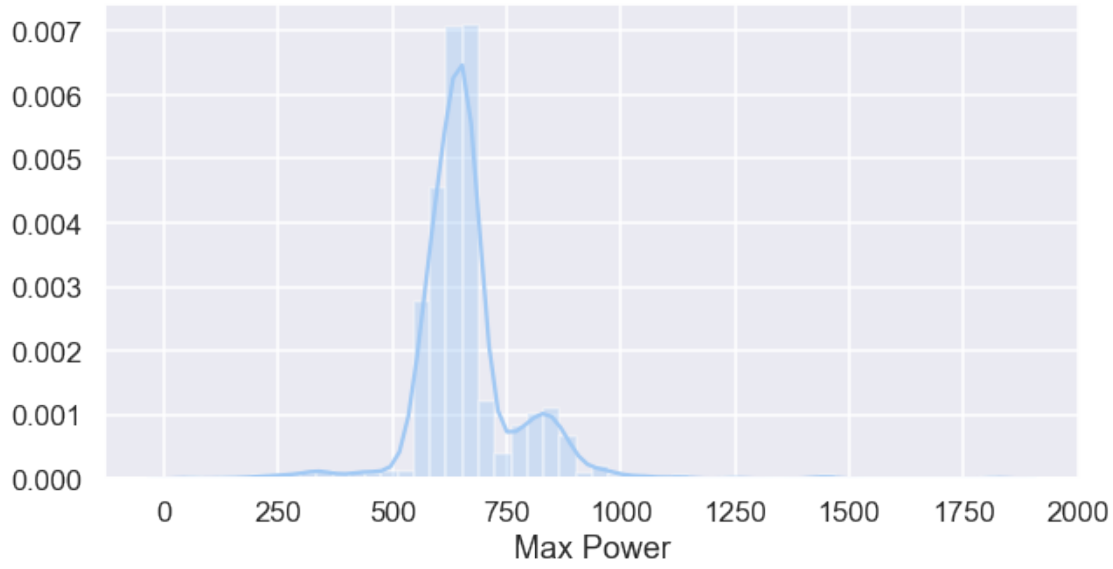
[53]: <matplotlib.patches.Polygon at 0x165c256cd30>



[ ]:
```
#Data Frame without outliers
#df_new = eddy[(eddy['Distance'] < upper) | (eddy['Distance'] > lower)]
```

[54]:
```
plt.figure(figsize = (10,5))
sns.distplot(eddy['Max Power'])
```

[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x165c46e8c88>
```



[56]:
```
def out_std(eddy, column):
    global lower,upper
    # calculate the mean and standard deviation of the data frame
    data_mean, data_std = eddy[column].mean(), eddy[column].std()
    # calculate the cutoff value
    cut_off = data_std * 3
    # calculate the lower and upper bound value
    lower, upper = data_mean - cut_off, data_mean + cut_off
    print('The lower bound value is', lower)
    print('The upper bound value is', upper)
    # Calculate the number of records below and above lower and above bound␣
 ↪value respectively
    df1 = eddy[eddy[column] > upper]
    df2 = eddy[eddy[column] < lower]
    return print('Total number of outliers are', df1.shape[0]+ df2.shape[0])
```

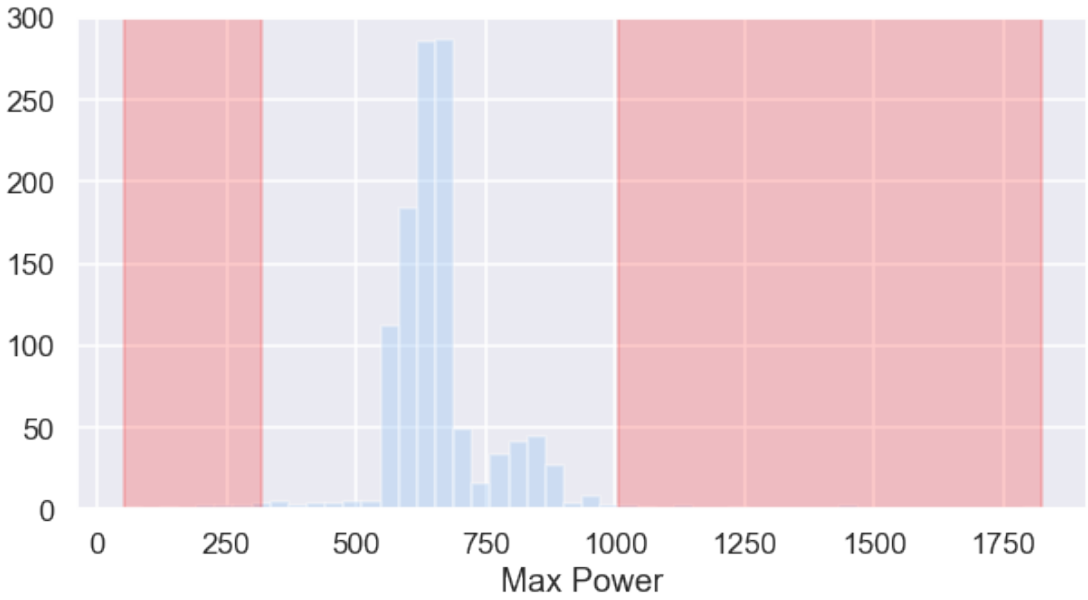
[57]:
```
out_std(eddy,'Max Power')
```

```
The lower bound value is 317.23750802226687
The upper bound value is 1006.5489145559156
```

```
Total number of outliers are 18
```

[58]:
```
plt.figure(figsize = (10,5))
sns.distplot(eddy['Max Power'], kde=False)
plt.axvspan(xmin = lower,xmax= eddy['Max Power'].min(),alpha=0.2, color='red')
plt.axvspan(xmin = upper,xmax= eddy['Max Power'].max(),alpha=0.2, color='red')
```

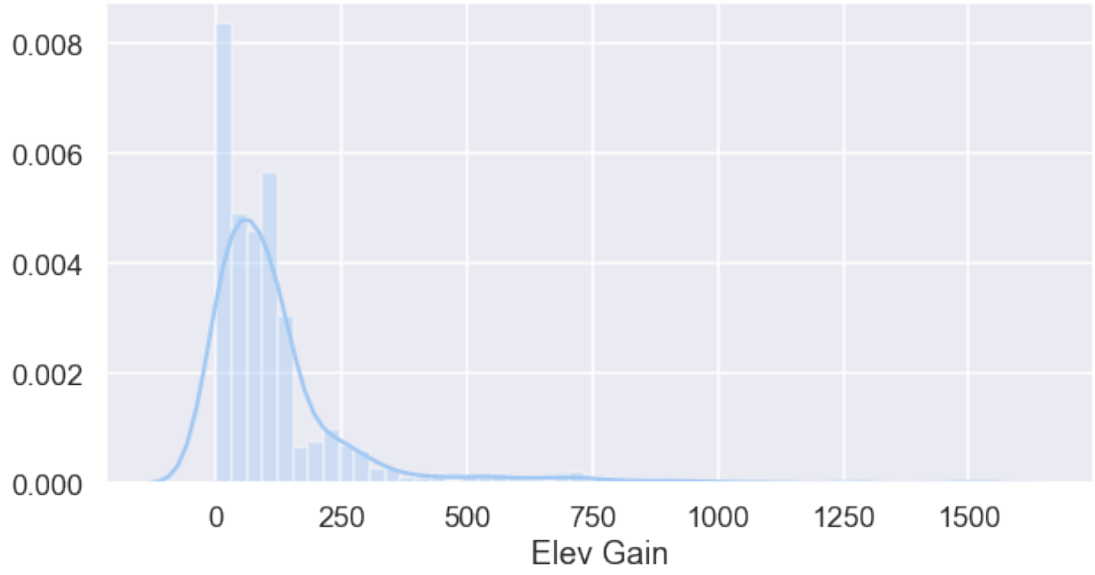
[58]:
```
<matplotlib.patches.Polygon at 0x165c2543a90>
```



[ ]:
```
#Data Frame without outliers
#df_new = eddy[(eddy['Max Power'] < upper) | (eddy['Max Power'] > lower)]
```

[59]:
```
#Zscore
plt.figure(figsize = (10,5))
sns.distplot(eddy['Elev Gain'])
```

[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x165c4d4c828>
```



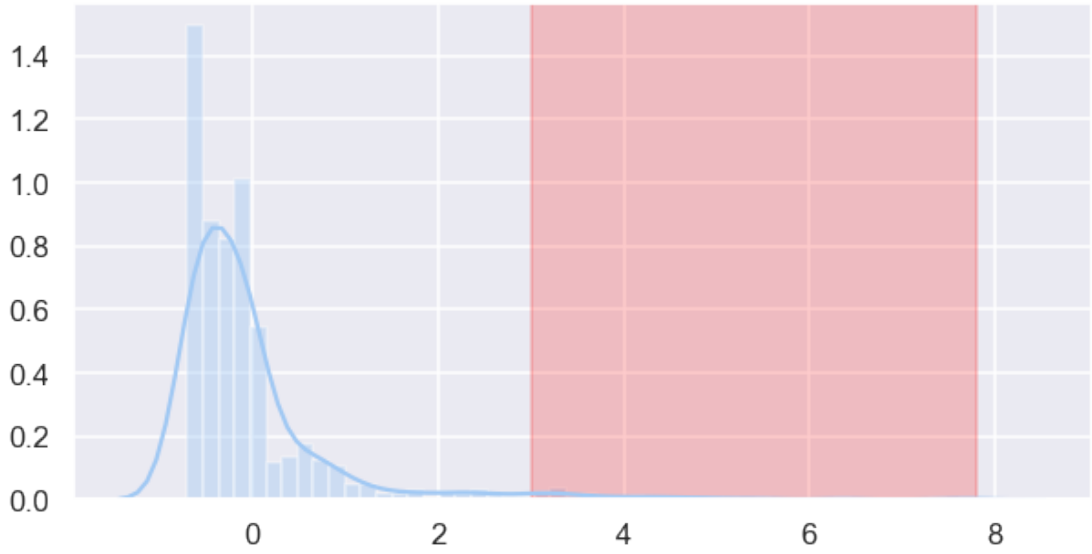
[60]:
```
def out_zscore(eddy):
    global outliers,zscore
    outliers = []
    zscore = []
    threshold = 3
    mean = np.mean(eddy)
    std = np.std(eddy)
    for i in eddy:
        z_score= (i - mean)/std
        zscore.append(z_score)
        if np.abs(z_score) > threshold:
            outliers.append(i)
    return print("Total number of outliers are",len(outliers))
```

[62]:
```
out_zscore(eddy['Elev Gain'])
```

```
Total number of outliers are 35
```

[63]:
```
plt.figure(figsize = (10,5))
sns.distplot(zscore)
plt.axvspan(xmin = 3 ,xmax= max(zscore),alpha=0.2, color='red')
```

[63]:
```
<matplotlib.patches.Polygon at 0x165c4d15128>
```



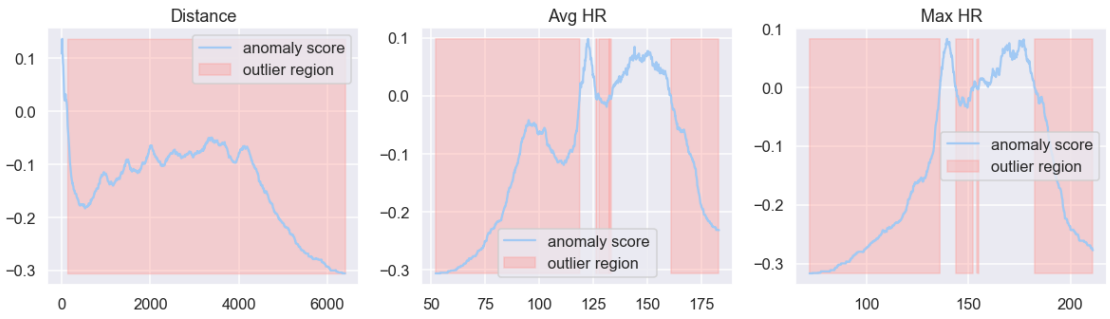
[ ]:
```
#df_new = eddy[(eddy['Calories'] < 3) | (eddy['Calories'] > -3)]
```

[64]:
```
#if
#Import necessary libraries
from sklearn.ensemble import IsolationForest
#The required columns
cols = ['Distance','Avg HR', 'Max HR']
#Plotting the sub plot
fig, axs = plt.subplots(1, 3, figsize=(20, 5), facecolor='w', edgecolor='k')
axs = axs.ravel()

for i, column in enumerate(cols):
    isolation_forest = IsolationForest(contamination='auto')
    isolation_forest.fit(eddy[column].values.reshape(-1,1))

    xx = np.linspace(eddy[column].min(), eddy[column].max(), len(eddy)).
 ↪reshape(-1,1)
    anomaly_score = isolation_forest.decision_function(xx)
    outlier = isolation_forest.predict(xx)

    axs[i].plot(xx, anomaly_score, label='anomaly score')
    axs[i].fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
                        where=outlier==-1, color='r',
                        alpha=.4, label='outlier region')
    axs[i].legend()
    axs[i].set_title(column)
```



[65]:
```
#DB scan
X = eddy[['Distance','Max HR']].values

db = DBSCAN(eps=3.0, min_samples=10).fit(X)
labels = db.labels_
```

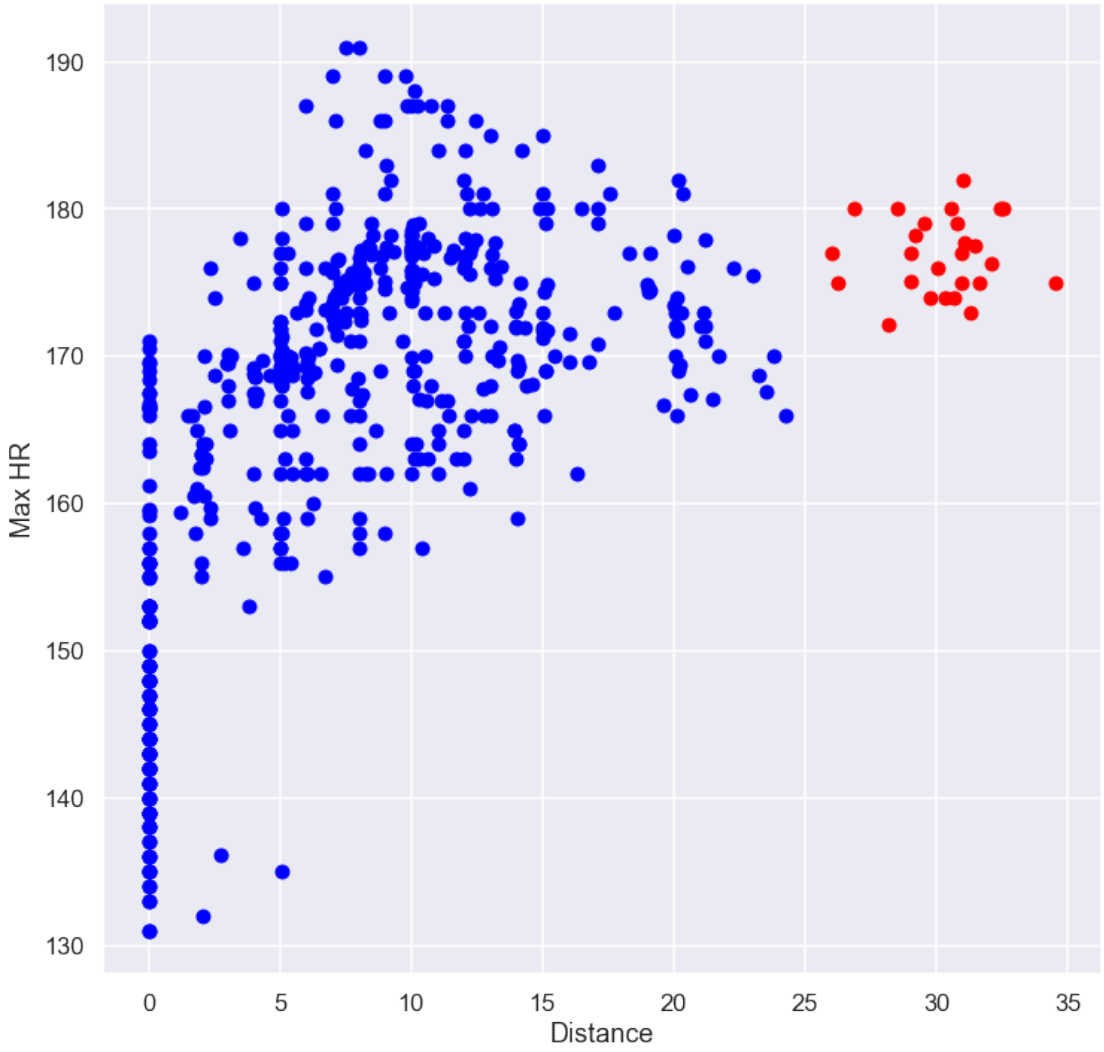
[66]:
```
pd.Series(labels).value_counts()#-1 represents outliers
```

[66]:
```
-1    546
 0    500
 1     26
 3     24
 5     18
 2     14
 6      6
 4      6
dtype: int64
```

[67]:
```
plt.figure(figsize=(12,12))#red outliers

unique_labels = set(labels)
colors = ['blue', 'red']

for color,label in zip(colors, unique_labels):
    sample_mask = [True if l == label else False for l in labels]
    plt.plot(X[:,0][sample_mask], X[:, 1][sample_mask], 'o', color=color);
plt.xlabel('Distance');
plt.ylabel('Max HR');
```



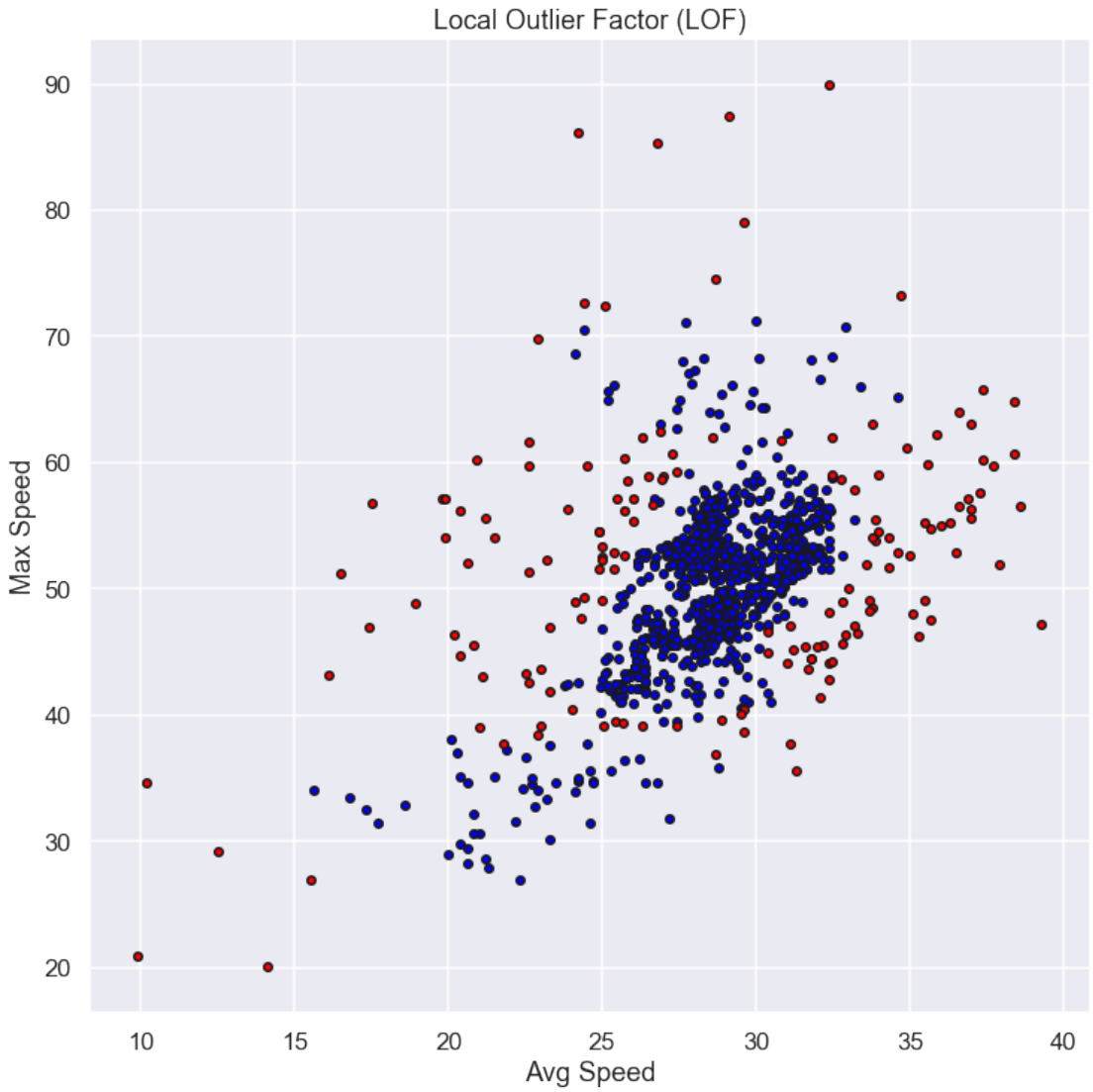
[69]:
```
#lofLocal Outlier Factor Method
clf = LocalOutlierFactor(n_neighbors=50, contamination='auto')
X = eddy[['Avg Speed','Max Speed']].values
y_pred = clf.fit_predict(X)
```

[70]:
```
plt.figure(figsize=(12,12))#red outliers ,blue nrml records
# plot the level sets of the decision function

in_mask = [True if l == 1 else False for l in y_pred]
out_mask = [True if l == -1 else False for l in y_pred]

plt.title("Local Outlier Factor (LOF)")
# inliers
a = plt.scatter(X[in_mask, 0], X[in_mask, 1], c = 'blue',
```

```
                edgecolor = 'k', s = 30)
# outliers
b = plt.scatter(X[out_mask, 0], X[out_mask, 1], c = 'red',
                edgecolor = 'k', s = 30)
plt.axis('tight')
plt.xlabel('Avg Speed');
plt.ylabel('Max Speed');
plt.show()
```



[ ]:

[ ]:

[ ]: