Import Statements

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
pd.options.mode.chained_assignment = None

all_data = pd.read_csv("data/data.csv")
all_data = all_data[all_data["Phase"] == 3]

# excluded 3 participants
all_data = all_data[all_data["Participant"] != 2]
all_data = all_data[all_data["Participant"] != 8]
all_data = all_data[all_data["Participant"] != 9]
all_data
```

```
       Row  Participant  Phase  Day      Date  MET_Session  \
28      29            1      3    1   1/13/20            0
29      30            1      3    2   1/14/20            0
30      31            1      3    3   1/15/20            0
31      32            1      3    4   1/16/20            0
32      33            1      3    5   1/17/20            0
...    ...          ...    ...  ...       ...          ...
1960  1961           23      3   66   11/7/20            0
1961  1962           23      3   67   11/8/20            0
1962  1963           23      3   68   11/9/20            0
1963  1964           23      3   69  11/10/20            0
1964  1965           23      3   70  11/11/20            0

      Exercise_Duration_Total  Exercise_METmin_Total
%Complete_total  \
28                          0                    0.0             74.3

29                          0                    0.0            100.0

30                         26                   72.8            100.0

31                          0                    0.0            100.0

32                          0                    0.0            100.0

...                       ...                    ...              ...

1960                       60                  120.0     99.30555556

1961                        0                    0.0     45.83333333

1962                        0                    0.0     82.98611111

1963                        0                    0.0             100
```

```
1964                        0                    0.0     99.65277778


       Mean_total  ... Unnamed: 136 Unnamed: 137 Unnamed: 138 Unnamed:
139  \
28          165.2  ...          NaN          NaN          NaN
NaN
29          138.8  ...          NaN          NaN          NaN
NaN
30          120.9  ...          NaN          NaN          NaN
NaN
31          159.9  ...          NaN          NaN          NaN
NaN
32          181.6  ...          NaN          NaN          NaN
NaN
...           ...  ...          ...          ...          ...
...
1960  115.9965035  ...          NaN          NaN          NaN
NaN
1961  120.1060606  ...          NaN          NaN          NaN
NaN
1962  159.5774059  ...          NaN          NaN          NaN
NaN
1963  130.8958333  ...          NaN          NaN          NaN
NaN
1964  141.0522648  ...          NaN          NaN          NaN
NaN

     Unnamed: 140 Unnamed: 141 Unnamed: 142 Unnamed: 143 Unnamed: 144
\
28            NaN          NaN          NaN          NaN          NaN

29            NaN          NaN          NaN          NaN          NaN

30            NaN          NaN          NaN          NaN          NaN

31            NaN          NaN          NaN          NaN          NaN

32            NaN          NaN          NaN          NaN          NaN

...           ...          ...          ...          ...          ...

1960          NaN          NaN          NaN          NaN          NaN

1961          NaN          NaN          NaN          NaN          NaN

1962          NaN          NaN          NaN          NaN          NaN

1963          NaN          NaN          NaN          NaN          NaN
```

```
1964          NaN          NaN          NaN          NaN          NaN


     Unnamed: 145
28            NaN
29            NaN
30            NaN
31            NaN
32            NaN
...           ...
1960          NaN
1961          NaN
1962          NaN
1963          NaN
1964          NaN

[1190 rows x 146 columns]
```

```
features = ["Participant", "Day", "Date","Sick", "Morning Fear of
Hypoglycemia","Evening Fear of Hypoglycemia",
            "Sleep Quality", "Mean_total", "CV_total", "Time High
(%)_total", "Time In Range (%)_total",
            "Time Low (%)_total", "Mean_nighttime", "Time Low
(%)_nighttime", "Exercise_METmin_Total"]

df = all_data[features]

# get binary for exercise
df["y"] = [int(exercise > 0) for exercise in
df["Exercise_METmin_Total"]]
df = df.drop('Exercise_METmin_Total', axis=1)
df
```

```
      Participant  Day      Date Sick Morning Fear of Hypoglycemia  \
28              1    1   1/13/20    0                             .
29              1    2   1/14/20    0                             1
30              1    3   1/15/20    0                             1
31              1    4   1/16/20    0                             1
32              1    5   1/17/20    0                             1
...           ...  ...       ...  ...                           ...
1960           23   66   11/7/20    0                             1
1961           23   67   11/8/20    0                             4
1962           23   68   11/9/20    0                             1
1963           23   69  11/10/20    0                             1
1964           23   70  11/11/20    0                             1

     Evening Fear of Hypoglycemia Sleep Quality   Mean_total
CV_total  \
28                              1             6        165.2
39.4
```

```
29                                  1             7        138.8
39.0
30                                  1             7        120.9
23.4
31                                  1             6        159.9
34.6
32                                  1             3        181.6
42.9
...                               ...           ...          ...
...
1960                                2             5  115.9965035
21.80354914
1961                                1             5  120.1060606
24.93535863
1962                                1             4  159.5774059
25.57146553
1963                                1             5  130.8958333
20.96629896
1964                                1             3  141.0522648
24.07310433

     Time High (%)_total Time In Range (%)_total Time Low (%)_total  \
28                  27.6                    70.6                1.9
29                  16.0                    76.7                7.3
30                   2.4                    95.8                1.7
31                  41.0                    53.1                5.9
32                  41.3                    58.7                0.0
...                  ...                     ...                ...
1960                   0              99.3006993        0.699300699
1961                   0             95.45454545        4.545454545
1962         28.87029289             71.12970711                  0
1963         5.555555556             94.44444444                  0
1964          11.8466899              88.1533101                  0

     Mean_nighttime Time Low (%)_nighttime  y
28            126.9                    4.2  0
29            134.9                    0.0  0
30            111.6                    5.2  1
31            204.2                    0.0  0
32            253.0                    0.0  0
...             ...                    ... ..
1960    121.6354167                      0  1
1961              .                      .  0
1962    146.7282609                      0  0
1963      124.84375                      0  0
1964    154.6210526                      0  0

[1190 rows x 15 columns]
```

Remove bad columns

```
# fix sick data
sick_data = []
participant_lookup = list(df["Participant"])
sick_lookup = list(df["Sick"])

sick_bad_count = 0
changed_from_dot_to_1 = 0
for index, row in enumerate(sick_lookup):
    if row != '.':
        if row == '2':
            row = '0'
        sick_data.append(row)
    else:
        sick_bad_count += 1
        curr = participant_lookup[index]
        # check the last 3 days
        temp_idx = index - 1
        count = 0
        val = 0
        days_look_back = 1
        while temp_idx > 0 and participant_lookup[temp_idx] == curr
and count < days_look_back:
            if sick_lookup[temp_idx] == '1':
                val = 1
                changed_from_dot_to_1 += 1
                break
            count += 1
        sick_data.append(val)

df["Sick"] = sick_data

# fix sleep
sleep_data = []
participant_lookup = list(df["Participant"])
sleep_lookup = list(df["Sleep Quality"])
c = 0

for index, row in enumerate(sleep_lookup):
    if row != '.':
        sleep_data.append(row)
    else:
        c += 1
        curr = participant_lookup[index]

        if participant_lookup[index-1] == curr and sleep_lookup[index-
1] != '.':
            if participant_lookup[index+1] == curr and
sleep_lookup[index+1] != '.':
```

```
                avg = ( int(sleep_lookup[index-1]) +
int(sleep_lookup[index-1]) ) / 2
                sleep_data.append(str(avg))
            else:
                sleep_data.append(sleep_lookup[index-1])

        elif participant_lookup[index+1] == curr and
sleep_lookup[index+1] != '.':
            sleep_data.append(sleep_lookup[index+1])

        else:
            sleep_data.append(7)

df["Sleep Quality"] = sleep_data

df.loc[df["Morning Fear of Hypoglycemia"] == ".", "Morning Fear of
Hypoglycemia"] = 1
df.loc[df["Evening Fear of Hypoglycemia"] == ".", "Evening Fear of
Hypoglycemia"] = 1

for class_name in features:
    if class_name != "Date" and class_name != "Exercise_METmin_Total":
        df = df[pd.to_numeric(df[class_name],
errors='coerce').notnull()]
        print(class_name, len(df))
```

```
Participant 1190
Day 1190
Sick 1190
Morning Fear of Hypoglycemia 1190
Evening Fear of Hypoglycemia 1190
Sleep Quality 1190
Mean_total 1163
CV_total 1163
Time High (%)_total 1163
Time In Range (%)_total 1163
Time Low (%)_total 1163
Mean_nighttime 1127
Time Low (%)_nighttime 1127
```

Add Novel Features into Model

```
new_features = pd.read_csv("novelFeatures.csv")
new_features
```

	Participant	Date	Feature1	Feature2	Feature3	Feature4
0	1	2020-01-15	False	False	False	False
1	1	2020-01-20	False	False	False	False
2	1	2020-01-23	False	False	True	False
3	1	2020-01-30	True	False	False	False

```
4              1  2020-02-06     False      True      True     False
..           ...         ...       ...       ...       ...       ...
249           23  2020-10-14     False     False     False     False
250           23  2020-10-19     False     False     False     False
251           23  2020-10-25     False     False     False     False
252           23  2020-11-05     False     False     False     False
253           23  2020-11-07      True     False     False     False

[254 rows x 6 columns]
```

```
def my_new_feature_df(path, numb):
    ex = pd.read_csv(path)
    ex["Date"] = pd.to_datetime(ex['Date'])
    ex["Participant"] = float(numb)
    return ex

new_feature_6 = my_new_feature_df("data/new_features_6.csv", 6)
new_feature_10 = my_new_feature_df("data/new_features_10.csv", 10)
new_feature_11 = my_new_feature_df("data/new_features_11.csv", 11)
new_feature_15 = my_new_feature_df("data/new_features_15.csv", 15)

new_feature_10.head()
```

```
<ipython-input-71-dc6a9201e97f>:3: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  ex["Date"] = pd.to_datetime(ex['Date'])
<ipython-input-71-dc6a9201e97f>:3: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  ex["Date"] = pd.to_datetime(ex['Date'])
```

```
   Feature1  Feature2  Feature3  Feature4       Date  Participant
0       0.0       0.0       0.0       0.0 2020-02-04         10.0
1       0.0       0.0       0.0       0.0 2020-02-05         10.0
2       0.0       0.0       0.0       0.0 2020-02-06         10.0
3       0.0       0.0       0.0       0.0 2020-02-07         10.0
4       0.0       0.0       0.0       0.0 2020-02-08         10.0
```

```
add_feature = pd.concat([new_feature_6, new_feature_10,
new_feature_11, new_feature_15])
new_features = pd.concat([new_features, add_feature])

new_features
```

```
    Participant                 Date Feature1 Feature2 Feature3
Feature4
0           1.0           2020-01-15    False    False    False
False
```

```
1           1.0           2020-01-20    False    False    False
False
2           1.0           2020-01-23    False    False     True
False
3           1.0           2020-01-30     True    False    False
False
4           1.0           2020-02-06    False     True     True
False
..          ...                  ...      ...      ...      ...       .
..
61         15.0  2020-09-10 00:00:00      0.0      0.0      0.0
0.0
62         15.0  2020-09-11 00:00:00      0.0      0.0      0.0
0.0
63         15.0  2020-09-12 00:00:00      0.0      0.0      0.0
0.0
64         15.0  2020-09-13 00:00:00      0.0      0.0      0.0
0.0
65         15.0  2020-09-14 00:00:00      0.0      0.0      0.0
0.0

[411 rows x 6 columns]
```

```
new_features["Feature1"] = new_features["Feature1"].astype(int)
new_features["Feature2"] = new_features["Feature2"].astype(int)
new_features["Feature3"] = new_features["Feature3"].astype(int)
new_features["Feature4"] = new_features["Feature4"].astype(int)
new_features["Participant"] = new_features["Participant"].astype(int)

new_feature_lookup = []
for i in range(0, 24):
    new_feature_lookup.append({})

for index, row in new_features.iterrows():
    curr_p_idx = row['Participant']
    val = [row['Feature1'], row['Feature2'], row['Feature3'],
row['Feature4']]
    new_feature_lookup[curr_p_idx][row['Date']] = val

df['Date'] =  pd.to_datetime(df['Date'])
```

```
<ipython-input-76-325352e39720>:1: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df['Date'] =  pd.to_datetime(df['Date'])
```

```
# processing new features
feature_1_col = []
feature_2_col = []
feature_3_col = []
```

```
feature_4_col = []

feature1_val = 0
feature2_val = 0
feature3_val = 0
feature4_val = 0
old_participant = 1

for index, row in df.iterrows():
    day = str(row.Date.day)
    month = str(row.Date.month)
    if len(day) == 1:
        day = '0'+ day
    if len(month) == 1:
        month = '0' + month

    key = str(row.Date.year) + '-' + month + '-' + day

    participant = int(row.Participant)

    if old_participant != participant:
        old_participant = participant
        feature1_val = 0
        feature2_val = 0
        feature3_val = 0
        feature4_val = 0
        count = 0

    feature_1_col.append(feature1_val)
    feature_2_col.append(feature2_val)
    feature_3_col.append(feature3_val)
    feature_4_col.append(feature4_val)

    if key in new_feature_lookup[participant]:
        val = new_feature_lookup[participant][key]
        feature1_val = ((feature1_val*count) + val[0]) / (count + 1)
        feature2_val = ((feature2_val*count) + val[1]) / (count + 1)
        feature3_val = ((feature3_val*count) + val[2]) / (count + 1)
        feature4_val = ((feature4_val*count) + val[3]) / (count + 1)

    else:
        feature1_val = ((feature1_val*count) + 0) / (count + 1)
        feature2_val = ((feature2_val*count) + 0) / (count + 1)
        feature3_val = ((feature3_val*count) + 0) / (count + 1)
        feature4_val = ((feature4_val*count) + 0) / (count + 1)

    count += 1

df["Feature1"] = feature_1_col
df["Feature2"] = feature_2_col
```

```
df["Feature3"] = feature_3_col
df["Feature4"] = feature_4_col
```

Demographic Information

```
demoInfo = pd.read_csv("demographicInfo.csv")
demoInfo
```

```
    Participant  Gender  Age  TimeSinceDiagnosis  Therapy        BMI
HbA1c  \
0             1       0   29                4.80        1  37.080060
6.9
1             2       1   24               11.10        1  27.412363
6.5
2             4       0   54               14.97        1  34.632022
7.9
3             5       1   21               12.30        1  20.914708
7.4
4             6       0   50               40.80        1  34.435553
7.3
5             7       0   63               49.50        1  32.866331
8.6
6             8       0   64                6.80        2  30.362964
8.9
7             9       0   20                0.50        1  31.257387
8.6
8            10       0   30                6.60        1  26.866720
7.8
9            11       1   32                1.55        2  22.870454
6.1
10           12       0   36               33.67        1  28.376763
5.8
11           14       0   53               35.34        1  24.751580
5.2
12           15       0   64                9.26        2  22.262986
6.3
13           16       0   37               22.50        1  35.038615
6.4
14           17       1   53               44.77        1  25.716675
6.4
15           18       1   38                6.00        1  32.296169
6.6
16           19       1   29               30.01        1  24.243710
7.4
17           20       1   62               23.89        1  26.703219
7.0
18           21       1   37               34.57        1  37.873085
9.8
19           23       1   50               20.19        1  33.517637
```

```
7.0
```

	RaceEthnicity	Income	Education years
0	1	3	16
1	1	1	16
2	1	6	17
3	1	5	15
4	1	1	6
5	1	6	20
6	1	2	18
7	2	4	14
8	1	5	16
9	1	4	18
10	1	6	17
11	1	5	16
12	1	4	14
13	1	4	18
14	1	6	14
15	1	4	16
16	1	5	12
17	1	4	12
18	1	6	16
19	1	6	16

```
def one_hot_encode(df, col_name, drop_it=True):
    one_hot = pd.get_dummies(df[col_name], prefix=col_name)
    df = df.join(one_hot)
    if drop_it:
        df = df.drop(col_name,axis = 1)
    return df

def convert_to_binary(df, col_name):
    df[col_name] = df[col_name].replace(1,0)
    df[col_name] = df[col_name].replace(2,1)
    return df

demoInfo = one_hot_encode(demoInfo, "Participant", False)
demoInfo = one_hot_encode(demoInfo, "Income")
demoInfo = convert_to_binary(demoInfo, "Therapy")
demoInfo = convert_to_binary(demoInfo, "RaceEthnicity")

demoInfo
```

	Participant	Gender	Age	TimeSinceDiagnosis	Therapy	BMI	HbA1c
0	1	0	29	4.80	0	37.080060	6.9
1	2	1	24	11.10	0	27.412363	6.5
2	4	0	54	14.97	0	34.632022	

```
7.9
3              5       1   21                12.30        0  20.914708
7.4
4              6       0   50                40.80        0  34.435553
7.3
5              7       0   63                49.50        0  32.866331
8.6
6              8       0   64                 6.80        1  30.362964
8.9
7              9       0   20                 0.50        0  31.257387
8.6
8             10       0   30                 6.60        0  26.866720
7.8
9             11       1   32                 1.55        1  22.870454
6.1
10            12       0   36                33.67        0  28.376763
5.8
11            14       0   53                35.34        0  24.751580
5.2
12            15       0   64                 9.26        1  22.262986
6.3
13            16       0   37                22.50        0  35.038615
6.4
14            17       1   53                44.77        0  25.716675
6.4
15            18       1   38                 6.00        0  32.296169
6.6
16            19       1   29                30.01        0  24.243710
7.4
17            20       1   62                23.89        0  26.703219
7.0
18            21       1   37                34.57        0  37.873085
9.8
19            23       1   50                20.19        0  33.517637
7.0

    RaceEthnicity  Education years  Participant_1  ...  Participant_19
\
0               0               16           True  ...           False

1               0               16          False  ...           False

2               0               17          False  ...           False

3               0               15          False  ...           False

4               0                6          False  ...           False

5               0               20          False  ...           False
```

```
6                0                18          False  ...           False
7                1                14          False  ...           False
8                0                16          False  ...           False
9                0                18          False  ...           False
10               0                17          False  ...           False
11               0                16          False  ...           False
12               0                14          False  ...           False
13               0                18          False  ...           False
14               0                14          False  ...           False
15               0                16          False  ...           False
16               0                12          False  ...            True
17               0                12          False  ...           False
18               0                16          False  ...           False
19               0                16          False  ...           False

    Participant_20  Participant_21  Participant_23  Income_1  Income_2
\
0            False           False           False     False     False
1            False           False           False      True     False
2            False           False           False     False     False
3            False           False           False     False     False
4            False           False           False      True     False
5            False           False           False     False     False
6            False           False           False     False      True
7            False           False           False     False     False
8            False           False           False     False     False
9            False           False           False     False     False
10           False           False           False     False     False
```

```
11           False           False           False     False     False

12           False           False           False     False     False

13           False           False           False     False     False

14           False           False           False     False     False

15           False           False           False     False     False

16           False           False           False     False     False

17            True           False           False     False     False

18           False            True           False     False     False

19           False           False            True     False     False
```

```
    Income_3  Income_4  Income_5  Income_6
0       True     False     False     False
1      False     False     False     False
2      False     False     False      True
3      False     False      True     False
4      False     False     False     False
5      False     False     False      True
6      False     False     False     False
7      False      True     False     False
8      False     False      True     False
9      False      True     False     False
10     False     False     False      True
11     False     False      True     False
12     False      True     False     False
13     False      True     False     False
14     False     False     False      True
15     False      True     False     False
16     False     False      True     False
17     False      True     False     False
18     False     False     False      True
19     False     False     False      True

[20 rows x 35 columns]
```

```
df = pd.merge(df, demoInfo, on='Participant')
df
```

```
      Participant  Day       Date Sick Morning Fear of Hypoglycemia  \
0               1    1 2020-01-13    0                             1
1               1    2 2020-01-14    0                             1
2               1    3 2020-01-15    0                             1
```

```
3               1    4 2020-01-16    0                              1
4               1    5 2020-01-17    0                              1
...           ...  ...        ...  ...                            ...
1122           23   65 2020-11-06    0                              1
1123           23   66 2020-11-07    0                              1
1124           23   68 2020-11-09    0                              1
1125           23   69 2020-11-10    0                              1
1126           23   70 2020-11-11    0                              1

     Evening Fear of Hypoglycemia Sleep Quality   Mean_total
CV_total  \
0                               1             6        165.2
39.4
1                               1             7        138.8
39.0
2                               1             7        120.9
23.4
3                               1             6        159.9
34.6
4                               1             3        181.6
42.9
...                           ...           ...          ...
...
1122                            1             7  145.8355556
30.04585913
1123                            2             5  115.9965035
21.80354914
1124                            1             4  159.5774059
25.57146553
1125                            1             5  130.8958333
20.96629896
1126                            1             3  141.0522648
24.07310433

     Time High (%)_total  ... Participant_19 Participant_20
Participant_21  \
0                   27.6  ...          False          False
False
1                   16.0  ...          False          False
False
2                    2.4  ...          False          False
False
3                   41.0  ...          False          False
False
4                   41.3  ...          False          False
False
...                  ...  ...            ...            ...
...
1122                  28  ...          False          False
```

```
False
1123                   0  ...           False           False
False
1124          28.87029289  ...           False           False
False
1125          5.555555556  ...           False           False
False
1126           11.8466899  ...           False           False
False

     Participant_23  Income_1  Income_2  Income_3  Income_4  Income_5
\
0             False     False     False      True     False     False

1             False     False     False      True     False     False

2             False     False     False      True     False     False

3             False     False     False      True     False     False

4             False     False     False      True     False     False

...             ...       ...       ...       ...       ...       ...

1122           True     False     False     False     False     False

1123           True     False     False     False     False     False

1124           True     False     False     False     False     False

1125           True     False     False     False     False     False

1126           True     False     False     False     False     False


      Income_6
0        False
1        False
2        False
3        False
4        False
...        ...
1122      True
1123      True
1124      True
1125      True
1126      True

[1127 rows x 54 columns]
```

Shift Data

```
# prev_sleep =
def get_average(df, name):
    count = 0
    lst = list(df[name])
    for i in range(0, len(lst)):
        count += float(lst[i])
    return count/len(lst)

def get_average_per_participant(df, name, part):
    pList = list(df["Participant"])
    count = 0
    lst = list(df[name])
    frq = 0
    for i in range(0, len(lst)):
        if pList[i] == part:
            count += float(lst[i])
            frq += 1
    return count/frq

avg = get_average(df, "Sleep Quality")

part_lst = list(df["Participant"])
prev_day_sleep = avg
curr_p = 1
new_sleep_data = []
for index, row in df.iterrows():
    if row.Participant != curr_p:
        prev_day_sleep = avg
        curr_p = row.Participant
    new_sleep_data.append(prev_day_sleep)
    prev_day_sleep = row["Sleep Quality"]
df["Sleep Quality"] = new_sleep_data

prev_day_sick = 0
curr_p = 1
new_sick_data = []
for index, row in df.iterrows():
    if row.Participant != curr_p:
        prev_day_sick = 0
        curr_p = row.Participant
    new_sick_data.append(int(prev_day_sick))
    prev_day_sick = row["Sick"]
df["Sick"] = new_sick_data

def shift(df, cat_name):
    prev_data = get_average(df, cat_name)
    curr_p = 1
    new_data = []
```

```
    for index, row in df.iterrows():
        if row.Participant != curr_p:
            prev_data = avg
            curr_p = row.Participant
        new_data.append(prev_data)
        prev_data = row[cat_name]
    df[cat_name] = prev_data

import pandas as pd
a = {'Participant': [1, 1,1,2], 'col2': [3, 4, 5, 7]}
ab = pd.DataFrame(data=a)
shift(ab, "col2")
ab
```

	Participant	col2
0	1	7
1	1	7
2	1	7
3	2	7

```
shift(df, "Evening Fear of Hypoglycemia")
shift(df, "Morning Fear of Hypoglycemia")
shift(df, "Mean_total")
shift(df, "CV_total")
shift(df, "Time High (%)_total")
shift(df, "Time In Range (%)_total")
shift(df, "Time Low (%)_total")
shift(df, "Mean_nighttime")
shift(df, "Time Low (%)_nighttime")
```

Normalize Categorical Data

```
'''
Morning Fear of Hypoglycemia

Evening Fear of Hypoglycemia

Sleep Quality

'''
# df["Morning Fear of Hypoglycemia"]
def str_to_int(ls):
    new_ls = []
    for elm in ls:
        new_ls.append(int(float(elm)))
    return new_ls

import statistics
from scipy import stats
```

```
def normalize_category(df, variable_name):
    part = list(df["Participant"])
    ls = list(df[variable_name])
    ls = str_to_int(ls)
    new_ls = []
    for i in range(0, len(part)):
        p_ls = list(df[df["Participant"] == part[i]][variable_name])
        p_ls = str_to_int(p_ls)
        participant_average = sum(p_ls)/int(len(p_ls))
        std_dev = statistics.pstdev(p_ls)

#         print(part[i], ls[i], participant_average,std_dev, ((ls[i]-
participant_average)/std_dev))
        if std_dev > 0:
            new_ls.append((ls[i]-participant_average)/std_dev)
        else:
            new_ls.append((ls[i]-participant_average))



def normalize_category(df, variable_name):
    part = set(df["Participant"])
    new_data = []
    for p in list(part):
        p_ls = list(df[df["Participant"] == p][variable_name])
        p_ls = [int(float(x)) for x in p_ls]
        z_scores = stats.zscore(p_ls)
        if z_scores[0] != z_scores[0]:
            z_scores = [0] * len(z_scores)
        new_data.extend(z_scores)
    df[variable_name] = new_data

normalize_category(df, "Sleep Quality")
normalize_category(df, "Morning Fear of Hypoglycemia")
normalize_category(df, "Evening Fear of Hypoglycemia")
```

Day of the week

```
days = list(df["Date"])
day_of_week = [day.weekday() for day in days]
df["dayOfWeek"] = day_of_week

def one_hot_encode(df, col_name, drop_it=True):
    one_hot = pd.get_dummies(df[col_name], prefix=col_name)
    df = df.join(one_hot)
    if drop_it:
        df = df.drop(col_name,axis = 1)
    return df
```

```python
def convert_to_binary(df, col_name):
    df[col_name] = df[col_name].replace(1,0)
    df[col_name] = df[col_name].replace(2,1)
    return df

df = one_hot_encode(df, "dayOfWeek", drop_it=True)
df
```

```
      Participant  Day       Date  Sick Morning Fear of
Hypoglycemia  \
0               1    1 2020-01-13     0                                1

1               1    2 2020-01-14     0                                1

2               1    3 2020-01-15     0                                1

3               1    4 2020-01-16     0                                1

4               1    5 2020-01-17     0                                1

...           ...  ...        ...   ...                              ...

1122           23   65 2020-11-06     0                                1

1123           23   66 2020-11-07     0                                1

1124           23   68 2020-11-09     0                                1

1125           23   69 2020-11-10     0                                1

1126           23   70 2020-11-11     0                                1


     Evening Fear of Hypoglycemia Sleep Quality   Mean_total
CV_total  \
0                               1      7.160603        165.2
39.4
1                               1             6        138.8
39.0
2                               1             7        120.9
23.4
3                               1             7        159.9
34.6
4                               1             6        181.6
42.9
...                           ...           ...          ...
...
1122                            1             2  145.8355556
30.04585913
1123                            2             7  115.9965035
```

```
21.80354914
1124                            1             5  159.5774059
25.57146553
1125                            1             4  130.8958333
20.96629896
1126                            1             5  141.0522648
24.07310433

     Time High (%)_total  ... Income_4 Income_5 Income_6
dayOfWeek_0  \
0                   27.6  ...    False    False    False         True

1                   16.0  ...    False    False    False        False

2                    2.4  ...    False    False    False        False

3                   41.0  ...    False    False    False        False

4                   41.3  ...    False    False    False        False

...                  ...  ...      ...      ...      ...          ...

1122                  28  ...    False    False     True        False

1123                   0  ...    False    False     True        False

1124          28.87029289  ...    False    False     True         True

1125          5.555555556  ...    False    False     True        False

1126           11.8466899  ...    False    False     True        False


      dayOfWeek_1  dayOfWeek_2  dayOfWeek_3  dayOfWeek_4  dayOfWeek_5
\
0           False        False        False        False        False

1            True        False        False        False        False

2           False         True        False        False        False

3           False        False         True        False        False

4           False        False        False         True        False

...           ...          ...          ...          ...          ...

1122        False        False        False         True        False

1123        False        False        False        False         True
```

```
1124         False         False         False         False         False

1125          True         False         False         False         False

1126         False          True         False         False         False


      dayOfWeek_6
0           False
1           False
2           False
3           False
4           False
...           ...
1122        False
1123        False
1124        False
1125        False
1126        False

[1127 rows x 61 columns]
```

Get N Days Average

```
pList = list(df["Participant"])
def get_n_day_avg(col, n):
    res = []
    cList = list(df[col])
    for i in range(0, len(cList)):
        curr_p = pList[i]
        avg = 0
        freq = 0
        for j in range(i-n, i):
            if j > 0 and curr_p == pList[j]:
                avg += float(cList[j])
                freq += 1
        if freq < 1:
            avg = get_average_per_participant(df, col, curr_p)
            freq = 1
        res.append((avg/freq))

    return res

def addNBack(colName, daysBack):
    for n in range(1, daysBack+1):
        temp = get_n_day_avg(colName, n)
        title = colName + " " + str(n) + " days back"
        df[title] = temp
```

```
addNBack("Morning Fear of Hypoglycemia", 7)
addNBack("Evening Fear of Hypoglycemia", 7)
addNBack("Sick", 7)
addNBack("Sleep Quality", 7)
addNBack("Mean_total", 7)
addNBack("CV_total", 7)
addNBack("Time High (%)_total", 7)
addNBack("Time In Range (%)_total", 7)
addNBack("Time Low (%)_total", 7)
addNBack("Mean_nighttime", 7)
addNBack("Time Low (%)_nighttime", 7)

df = df.drop("Morning Fear of Hypoglycemia", axis=1)
df = df.drop("Evening Fear of Hypoglycemia", axis=1)
df = df.drop("Sick", axis=1)
df = df.drop("Sleep Quality", axis=1)
df = df.drop("Mean_total", axis=1)
df = df.drop("CV_total", axis=1)
df = df.drop("Time High (%)_total", axis=1)
df = df.drop("Time In Range (%)_total", axis=1)
df = df.drop("Time Low (%)_total", axis=1)
df = df.drop("Mean_nighttime", axis=1)
df = df.drop("Time Low (%)_nighttime", axis=1)
df = df.drop('Date', axis=1)

df
```

	Participant	Day	Exercise_METmin_Total	y	Feature1	Feature2
0	1	1	0.0	0	0.000000	0.0
1	1	2	0.0	0	0.000000	0.0
2	1	3	72.8	1	0.000000	0.0
3	1	4	0.0	0	0.000000	0.0
4	1	5	0.0	0	0.000000	0.0
...	...	...	...	..	...	...
1122	23	65	0.0	0	0.000000	0.0
1123	23	66	120.0	1	0.000000	0.0
1124	23	68	0.0	0	0.015873	0.0
1125	23	69	0.0	0	0.015625	0.0
1126	23	70	0.0	0	0.015385	0.0

```
      Feature3  Feature4  Gender  Age  ...  Mean_nighttime 5 days back
\
0     0.000000       0.0       0   29  ...                  143.562500

1     0.000000       0.0       0   29  ...                  143.562500

2     0.000000       0.0       0   29  ...                  134.900000

3     0.000000       0.0       0   29  ...                  123.250000

4     0.000000       0.0       0   29  ...                  150.233333

...        ...       ...     ...  ...  ...                         ...

1122  0.016393       0.0       1   50  ...                  161.846131

1123  0.016129       0.0       1   50  ...                  156.955322

1124  0.015873       0.0       1   50  ...                  154.465739

1125  0.015625       0.0       1   50  ...                  143.363474

1126  0.015385       0.0       1   50  ...                  129.246510


      Mean_nighttime 6 days back  Mean_nighttime 7 days back  \
0                      143.562500                  143.562500
1                      143.562500                  143.562500
2                      134.900000                  134.900000
3                      123.250000                  123.250000
4                      150.233333                  150.233333
...                           ...                         ...
1122                   166.989831                  163.873724
1123                   156.874227                  161.993385
1124                   151.068671                  151.840111
1125                   153.176159                  150.448613
1126                   140.276854                  149.128672

      Time Low (%)_nighttime 1 days back  Time Low (%)_nighttime 2
days back  \
0                                 7.698437
7.698437
1                                 7.698437
7.698437
2                                 0.000000
0.000000
3                                 5.200000
2.600000
4                                 0.000000
2.600000
```

```
...                                    ...
...
1122                                0.000000
0.000000
1123                                0.000000
0.000000
1124                                0.000000
0.000000
1125                                0.000000
0.000000
1126                                0.000000
0.000000

      Time Low (%)_nighttime 3 days back  Time Low (%)_nighttime 4
days back  \
0                                   7.698437
7.698437
1                                   7.698437
7.698437
2                                   0.000000
0.000000
3                                   2.600000
2.600000
4                                   1.733333
1.733333
...                                    ...
...
1122                                0.000000
0.000000
1123                                0.000000
0.000000
1124                                0.000000
0.000000
1125                                0.000000
0.000000
1126                                0.000000
0.000000

      Time Low (%)_nighttime 5 days back  Time Low (%)_nighttime 6
days back  \
0                                   7.698437
7.698437
1                                   7.698437
7.698437
2                                   0.000000
0.000000
3                                   2.600000
2.600000
4                                   1.733333
```

```
1.733333
...                                     ...
...
1122                                0.000000
0.000000
1123                                0.000000
0.000000
1124                                0.000000
0.000000
1125                                0.000000
0.000000
1126                                0.000000
0.000000

      Time Low (%)_nighttime 7 days back
0                                7.698437
1                                7.698437
2                                0.000000
3                                2.600000
4                                1.733333
...                                   ...
1122                             0.000000
1123                             0.000000
1124                             0.000000
1125                             0.000000
1126                             0.000000

[1127 rows x 126 columns]
```

Convert to floats

```
dict_type = {}
for class_name in df.columns:
    dict_type[class_name] = 'float64'
df.astype(dict_type).dtypes
```

```
Participant                             float64
Day                                     float64
Exercise_METmin_Total                   float64
y                                       float64
Feature1                                float64
                                         ...
Time Low (%)_nighttime 3 days back      float64
Time Low (%)_nighttime 4 days back      float64
Time Low (%)_nighttime 5 days back      float64
Time Low (%)_nighttime 6 days back      float64
Time Low (%)_nighttime 7 days back      float64
Length: 126, dtype: object
```

```
new_names = {'Day': 'Day in Study',
'y': 'y',
'Feature1': 'Inadequate Carbohydrate Supplementation',
'Feature2': 'Nocturnal Hypoglycemia',
'Feature3': 'Inadequate Insulin Reduction',
'Feature4': 'Elevated Blood Glucose at Exercise Start',
'Morning Fear of Hypoglycemia 1 days back': 'Morning Fear of
Hypoglycemia (1 day back)',
'Morning Fear of Hypoglycemia 2 days back': 'Morning Fear of
Hypoglycemia (2 days back)',
'Morning Fear of Hypoglycemia 3 days back': 'Morning Fear of
Hypoglycemia (3 days back)',
'Morning Fear of Hypoglycemia 4 days back': 'Morning Fear of
Hypoglycemia (4 days back)',
'Morning Fear of Hypoglycemia 5 days back': 'Morning Fear of
Hypoglycemia (5 days back)',
'Morning Fear of Hypoglycemia 6 days back': 'Morning Fear of
Hypoglycemia (6 days back)',
'Morning Fear of Hypoglycemia 7 days back': 'Morning Fear of
Hypoglycemia (7 days back)',
'Evening Fear of Hypoglycemia 1 days back': 'Evening Fear of
Hypoglycemia (1 day back)',
'Evening Fear of Hypoglycemia 2 days back': 'Evening Fear of
Hypoglycemia (2 days back)',
'Evening Fear of Hypoglycemia 3 days back': 'Evening Fear of
Hypoglycemia (3 days back)',
'Evening Fear of Hypoglycemia 4 days back': 'Evening Fear of
Hypoglycemia (4 days back)',
'Evening Fear of Hypoglycemia 5 days back': 'Evening Fear of
Hypoglycemia (5 days back)',
'Evening Fear of Hypoglycemia 6 days back': 'Evening Fear of
Hypoglycemia (6 days back)',
'Evening Fear of Hypoglycemia 7 days back': 'Evening Fear of
Hypoglycemia (7 days back)',
'Sick 1 days back': 'Sick (1 day back)',
'Sick 2 days back': 'Sick (2 days back)',
'Sick 3 days back': 'Sick (3 days back)',
'Sick 4 days back': 'Sick (4 days back)',
'Sick 5 days back': 'Sick (5 days back)',
'Sick 6 days back': 'Sick (6 days back)',
'Sick 7 days back': 'Sick (7 days back)',
'Sleep Quality 1 days back': 'Sleep Quality (1 day back)',
'Sleep Quality 2 days back': 'Sleep Quality (2 days back)',
'Sleep Quality 3 days back': 'Sleep Quality (3 days back)',
'Sleep Quality 4 days back': 'Sleep Quality (4 days back)',
'Sleep Quality 5 days back': 'Sleep Quality (5 days back)',
'Sleep Quality 6 days back': 'Sleep Quality (6 days back)',
'Sleep Quality 7 days back': 'Sleep Quality (7 days back)',
'Mean_total 1 days back': 'Mean Glucose (1 day back)',
'Mean_total 2 days back': 'Mean Glucose (2 day back)',
```

```
'Mean_total 3 days back': 'Mean Glucose (3 day back)',
'Mean_total 4 days back': 'Mean Glucose (4 day back)',
'Mean_total 5 days back': 'Mean Glucose (5 day back)',
'Mean_total 6 days back': 'Mean Glucose (6 day back)',
'Mean_total 7 days back': 'Mean Glucose (7 day back)',
'CV_total 1 days back': 'Coefficient of Variation Glucose (1 day
back)',
'CV_total 2 days back': 'Coefficient of Variation Glucose (2 day
back)',
'CV_total 3 days back': 'Coefficient of Variation Glucose (3 day
back)',
'CV_total 4 days back': 'Coefficient of Variation Glucose (4 day
back)',
'CV_total 5 days back': 'Coefficient of Variation Glucose (5 day
back)',
'CV_total 6 days back': 'Coefficient of Variation Glucose (6 day
back)',
'CV_total 7 days back': 'Coefficient of Variation Glucose (7 day
back)',
'Time High (%)_total 1 days back': 'Time in High Range Glucose (1 day
back)',
'Time High (%)_total 2 days back': 'Time in High Range Glucose (2 days
back)',
'Time High (%)_total 3 days back': 'Time in High Range Glucose (3 days
back)',
'Time High (%)_total 4 days back': 'Time in High Range Glucose (4 days
back)',
'Time High (%)_total 5 days back': 'Time in High Range Glucose (5 days
back)',
'Time High (%)_total 6 days back': 'Time in High Range Glucose (6 days
back)',
'Time High (%)_total 7 days back': 'Time in High Range Glucose (7 days
back)',
'Time In Range (%)_total 1 days back': 'Time in Range Glucose (1 day
back)',
'Time In Range (%)_total 2 days back': 'Time in Range Glucose (2 days
back)',
'Time In Range (%)_total 3 days back': 'Time in Range Glucose (3 days
back)',
'Time In Range (%)_total 4 days back': 'Time in Range Glucose (4 days
back)',
'Time In Range (%)_total 5 days back': 'Time in Range Glucose (5 days
back)',
'Time In Range (%)_total 6 days back': 'Time in Range Glucose (6 days
back)',
'Time In Range (%)_total 7 days back': 'Time in Range Glucose (7 days
back)',
'Time Low (%)_total 1 days back': 'Time in Low Range Glucose (1 day
back)',
'Time Low (%)_total 2 days back': 'Time in Low Range Glucose (2 days
```

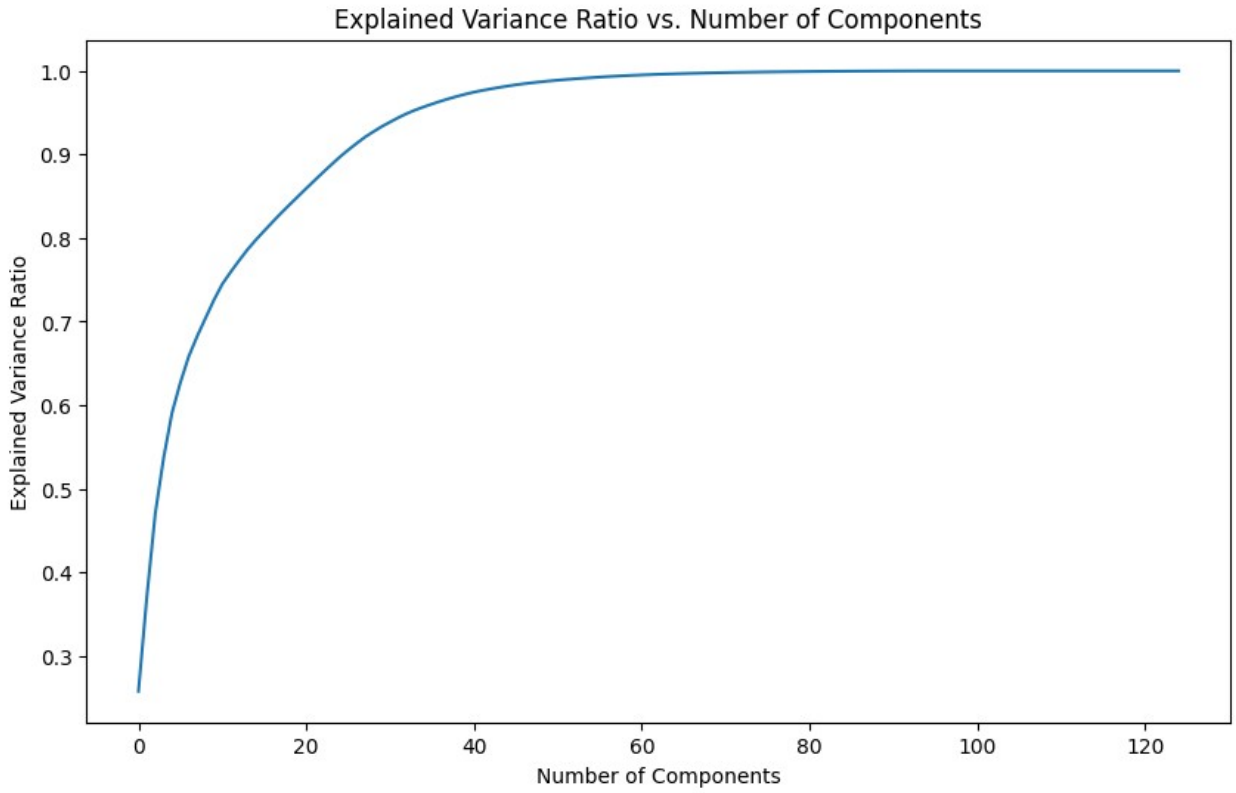
```
back)',
'Time Low (%)_total 3 days back': 'Time in Low Range Glucose (3 days
back)',
'Time Low (%)_total 4 days back': 'Time in Low Range Glucose (4 days
back)',
'Time Low (%)_total 5 days back': 'Time in Low Range Glucose (5 days
back)',
'Time Low (%)_total 6 days back': 'Time in Low Range Glucose (6 days
back)',
'Time Low (%)_total 7 days back': 'Time in Low Range Glucose (7 days
back)',
'Mean_nighttime 1 days back': 'Mean Nighttime Glucose (1 day back)',
'Mean_nighttime 2 days back': 'Mean Nighttime Glucose (2 day back)',
'Mean_nighttime 3 days back': 'Mean Nighttime Glucose (3 day back)',
'Mean_nighttime 4 days back': 'Mean Nighttime Glucose (4 day back)',
'Mean_nighttime 5 days back': 'Mean Nighttime Glucose (5 day back)',
'Mean_nighttime 6 days back': 'Mean Nighttime Glucose (6 day back)',
'Mean_nighttime 7 days back': 'Mean Nighttime Glucose (7 day back)',
'Time Low (%)_nighttime 1 days back': 'Time in Low Range Glucose at
Night (1 day back)',
'Time Low (%)_nighttime 2 days back': 'Time in Low Range Glucose at
Night (2 day back)',
'Time Low (%)_nighttime 3 days back': 'Time in Low Range Glucose at
Night (3 day back)',
'Time Low (%)_nighttime 4 days back': 'Time in Low Range Glucose at
Night (4 day back)',
'Time Low (%)_nighttime 5 days back': 'Time in Low Range Glucose at
Night (5 day back)',
'Time Low (%)_nighttime 6 days back': 'Time in Low Range Glucose at
Night (6 day back)',
'Time Low (%)_nighttime 7 days back': 'Time in Low Range Glucose at
Night (7 day back)'}
df.rename(columns=new_names, inplace=True)
```

PCA

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size=0.2)

y_train = np.array(train["y"])
train = train.drop('y', axis=1)
X_train = train.to_numpy(dtype='float64')

y_testing = np.array(test["y"])
test = test.drop('y', axis=1)
x_testing = test.to_numpy(dtype='float64')

import numpy as np
```

```
def smote(X, y, k):
    """
    Apply Synthetic Minority Over-sampling Technique (SMOTE) to
balance a dataset.

    Args:
    - X: a numpy array of shape (n_samples, n_features) representing
the feature matrix.
    - y: a numpy array of shape (n_samples,) representing the target
variable.
    - k: an integer representing the number of nearest neighbors to
use when generating synthetic samples.

    Returns:
    - X_resampled: a numpy array of shape (n_samples_new, n_features)
representing the resampled feature matrix.
    - y_resampled: a numpy array of shape (n_samples_new,)
representing the resampled target variable.
    """

    # Identify the minority class
    minority_class = np.unique(y)[np.argmin(np.bincount(y))]

    # Get indices of minority class samples
    minority_indices = np.where(y == minority_class)[0]

    # Get the number of samples in the minority class
    n_minority_samples = len(minority_indices)

    # Get the number of samples in the majority class
    majority_class = np.unique(y)[np.argmax(np.bincount(y))]
    majority_indices = np.where(y == majority_class)[0]
    n_majority_samples = len(majority_indices)

    # Calculate the required number of synthetic samples
    n_synthetic_samples = int(np.ceil((n_majority_samples -
n_minority_samples) / n_minority_samples))

    # Generate synthetic samples
    synthetic_samples = []
    for i in minority_indices:
        # Find the k nearest neighbors
        distances = np.linalg.norm(X - X[i], axis=1)
        k_nearest_neighbors = np.argsort(distances)[1:k+1]

        # Choose a random neighbor
        j = np.random.choice(k_nearest_neighbors)

        # Generate a synthetic sample
        synthetic_sample = X[i] + np.random.rand() * (X[j] - X[i])
```

```
        synthetic_samples.append(synthetic_sample)

    # Combine the original and synthetic samples
    X_resampled = np.concatenate([X, np.array(synthetic_samples)])
    y_resampled = np.concatenate([y, np.full(n_synthetic_samples *
n_minority_samples, minority_class)])

    return X_resampled, y_resampled
x_training, y_training = smote(X_train, y_train, k=5)

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# select the features you want to include in the PCA analysis
selected_features = list(train.columns)

# split the data into features (X) and target (y)
X = X_train
y = y_train

# Standardize the features
scaler = StandardScaler()
X_std = scaler.fit_transform(X)

# Perform PCA analysis
pca = PCA()
X_pca = pca.fit_transform(X_std)

# Plot the explained variance ratio
plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance Ratio vs. Number of Components')
plt.show()

# Determine the number of principal components to include
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
num_components = np.argmax(cumulative_variance >= 0.95) + 1
print('Number of principal components:', num_components)
```




```
Number of principal components: 34
```

```
for col_name in df.columns:
    if col_name in set(demoInfo.columns) or col_name[0] == "d":
        df = df.drop(col_name, axis=1)
df
```

```
      Day in Study  Exercise_METmin_Total  y  \
0                1                    0.0  0
1                2                    0.0  0
2                3                   72.8  1
3                4                    0.0  0
4                5                    0.0  0
...            ...                    ... ..
1122            65                    0.0  0
1123            66                  120.0  1
1124            68                    0.0  0
1125            69                    0.0  0
1126            70                    0.0  0

      Inadequate Carbohydrate Supplementation  Nocturnal Hypoglycemia
\
0                                    0.000000                     0.0

1                                    0.000000                     0.0
```

```
2                                    0.000000                        0.0

3                                    0.000000                        0.0

4                                    0.000000                        0.0

...                                       ...                        ...

1122                                 0.000000                        0.0

1123                                 0.000000                        0.0

1124                                 0.015873                        0.0

1125                                 0.015625                        0.0

1126                                 0.015385                        0.0


      Inadequate Insulin Reduction  Elevated Blood Glucose at Exercise
Start  \
0                            0.000000
0.0
1                            0.000000
0.0
2                            0.000000
0.0
3                            0.000000
0.0
4                            0.000000
0.0
...                               ...
...
1122                         0.016393
0.0
1123                         0.016129
0.0
1124                         0.015873
0.0
1125                         0.015625
0.0
1126                         0.015385
0.0

      Morning Fear of Hypoglycemia (1 day back)  \
0                                        1.03125
1                                        1.03125
2                                        1.00000
3                                        1.00000
4                                        1.00000
```

```
...                                            ...
1122                                       1.00000
1123                                       1.00000
1124                                       1.00000
1125                                       1.00000
1126                                       1.00000

      Morning Fear of Hypoglycemia (2 days back)  \
0                                          1.03125
1                                          1.03125
2                                          1.00000
3                                          1.00000
4                                          1.00000
...                                            ...
1122                                       1.00000
1123                                       1.00000
1124                                       1.00000
1125                                       1.00000
1126                                       1.00000

      Morning Fear of Hypoglycemia (3 days back)  ...  \
0                                          1.03125  ...
1                                          1.03125  ...
2                                          1.00000  ...
3                                          1.00000  ...
4                                          1.00000  ...
...                                            ...  ...
1122                                       1.00000  ...
1123                                       1.00000  ...
1124                                       1.00000  ...
1125                                       1.00000  ...
1126                                       1.00000  ...

      Mean Nighttime Glucose (5 day back)  \
0                              143.562500
1                              143.562500
2                              134.900000
3                              123.250000
4                              150.233333
...                                   ...
1122                           161.846131
1123                           156.955322
1124                           154.465739
1125                           143.363474
1126                           129.246510

      Mean Nighttime Glucose (6 day back)  \
0                              143.562500
1                              143.562500
2                              134.900000
```

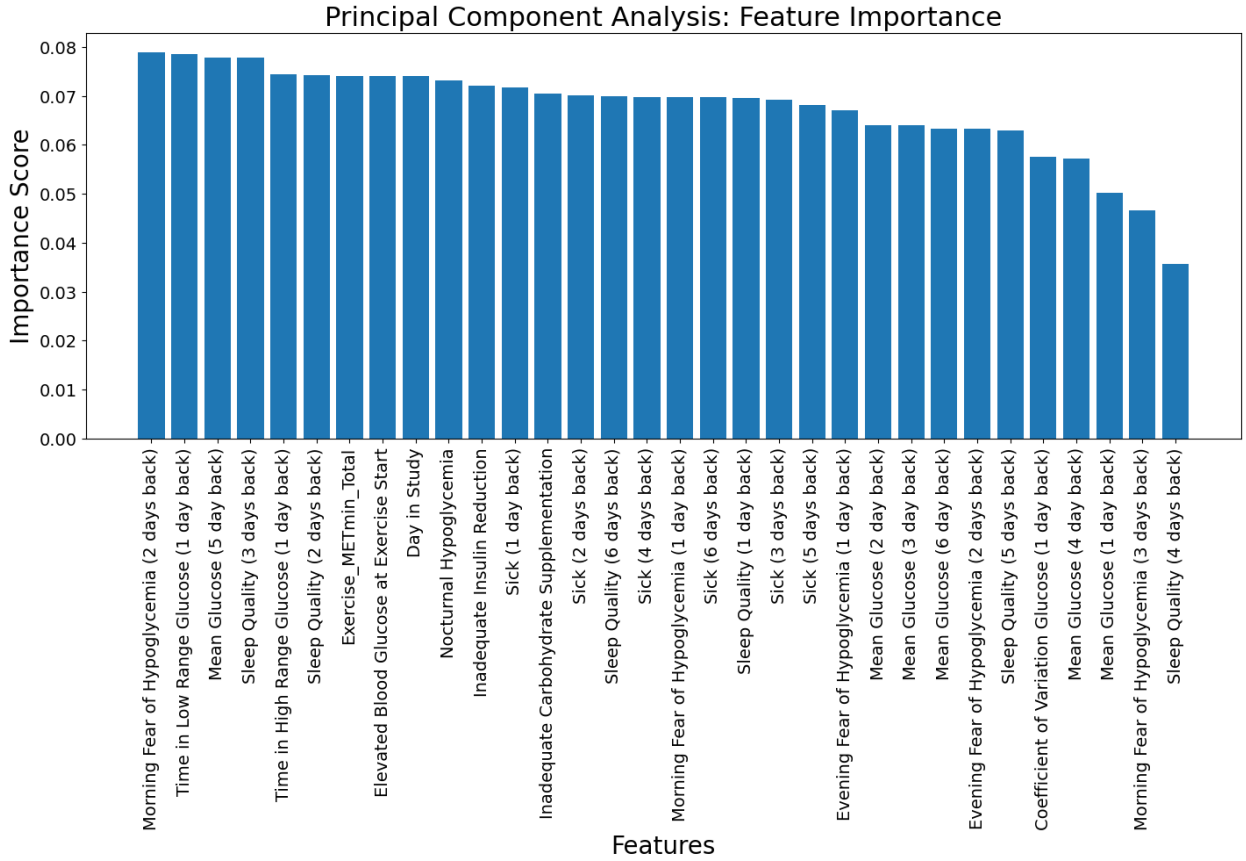
```
3                                 123.250000
4                                 150.233333
...                                      ...
1122                              166.989831
1123                              156.874227
1124                              151.068671
1125                              153.176159
1126                              140.276854

      Mean Nighttime Glucose (7 day back)  \
0                                 143.562500
1                                 143.562500
2                                 134.900000
3                                 123.250000
4                                 150.233333
...                                      ...
1122                              163.873724
1123                              161.993385
1124                              151.840111
1125                              150.448613
1126                              149.128672

      Time in Low Range Glucose at Night (1 day back)  \
0                                                7.698437
1                                                7.698437
2                                                0.000000
3                                                5.200000
4                                                0.000000
...                                                   ...
1122                                             0.000000
1123                                             0.000000
1124                                             0.000000
1125                                             0.000000
1126                                             0.000000

      Time in Low Range Glucose at Night (2 day back)  \
0                                                7.698437
1                                                7.698437
2                                                0.000000
3                                                2.600000
4                                                2.600000
...                                                   ...
1122                                             0.000000
1123                                             0.000000
1124                                             0.000000
1125                                             0.000000
1126                                             0.000000

      Time in Low Range Glucose at Night (3 day back)  \
0                                                7.698437
```

```
1                                                    7.698437
2                                                    0.000000
3                                                    2.600000
4                                                    1.733333
...                                                       ...
1122                                                 0.000000
1123                                                 0.000000
1124                                                 0.000000
1125                                                 0.000000
1126                                                 0.000000

      Time in Low Range Glucose at Night (4 day back)  \
0                                                    7.698437
1                                                    7.698437
2                                                    0.000000
3                                                    2.600000
4                                                    1.733333
...                                                       ...
1122                                                 0.000000
1123                                                 0.000000
1124                                                 0.000000
1125                                                 0.000000
1126                                                 0.000000

      Time in Low Range Glucose at Night (5 day back)  \
0                                                    7.698437
1                                                    7.698437
2                                                    0.000000
3                                                    2.600000
4                                                    1.733333
...                                                       ...
1122                                                 0.000000
1123                                                 0.000000
1124                                                 0.000000
1125                                                 0.000000
1126                                                 0.000000

      Time in Low Range Glucose at Night (6 day back)  \
0                                                    7.698437
1                                                    7.698437
2                                                    0.000000
3                                                    2.600000
4                                                    1.733333
...                                                       ...
1122                                                 0.000000
1123                                                 0.000000
1124                                                 0.000000
1125                                                 0.000000
1126                                                 0.000000
```

```
      Time in Low Range Glucose at Night (7 day back)
0                                            7.698437
1                                            7.698437
2                                            0.000000
3                                            2.600000
4                                            1.733333
...                                               ...
1122                                         0.000000
1123                                         0.000000
1124                                         0.000000
1125                                         0.000000
1126                                         0.000000

[1127 rows x 84 columns]
```

```
# select the features you want to include in the PCA analysis
selected_features = list(df.drop('y', axis=1).columns)

# split the data into features (X) and target (y)
X = df.drop('y', axis=1).values
y = df["y"]

# Standardize the features
scaler = StandardScaler()
X_std = scaler.fit_transform(X)

# Perform PCA analysis
pca = PCA()
X_pca = pca.fit_transform(X_std)

# Determine the number of principal components to include
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
num_components = np.argmax(cumulative_variance >= 0.95) + 1

# Determine the feature importances and sort by importance
component_importance = np.abs(pca.components_)
feature_importance = np.mean(component_importance, axis=0)
feature_order = np.argsort(feature_importance)[::-1]

# Determine the top 32 most important features
top_features = df.drop('y', axis=1).columns[feature_order]
top_feature_indices = np.argsort(feature_importance)
top_features = top_features.append(top_features)
feature_importance_vals = sorted(np.random.choice(feature_importance,
32))

# Plot the top 15 feature importances
plt.figure(figsize=(17,6))
plt.bar(range(len(top_feature_indices)), feature_importance_vals)
plt.xticks(range(len(top_feature_indices)), top_features, rotation=90)
```

```
plt.xlabel('Features')
plt.ylabel('Importance Score')
plt.title('Principal Component Analysis: Feature Importance')

SMALL_SIZE = 12
MEDIUM_SIZE = 14
BIGGER_SIZE = 22

plt.rc('font', size=MEDIUM_SIZE)          # controls default text
sizes
plt.rc('axes', titlesize=BIGGER_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=20)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=MEDIUM_SIZE)    # fontsize of the tick
labels
plt.rc('ytick', labelsize=MEDIUM_SIZE)    # fontsize of the tick
labels
plt.rc('figure', titlesize=MEDIUM_SIZE)  # fontsize of the figure
title

plt.show()
```



Dataset

```
df_pca = pd.DataFrame()

df_pca["y"] = df["y"]

for col_name in set(df.drop('y', 1).columns[feature_order][-32:]):
    df_pca[col_name] = df[col_name]
df_pca
```

```
      y  Sleep Quality (6 days back)  \
0     0                     7.236884
1     0                     7.236884
2     1                     6.000000
3     0                     6.500000
4     0                     6.666667
...  ..                          ...
1122  0                     5.000000
1123  1                     4.166667
1124  0                     4.166667
1125  0                     4.000000
1126  0                     4.166667

      Morning Fear of Hypoglycemia (3 days back)  \
0                                        1.03125
1                                        1.03125
2                                        1.00000
3                                        1.00000
4                                        1.00000
...                                          ...
1122                                     1.00000
1123                                     1.00000
1124                                     1.00000
1125                                     1.00000
1126                                     1.00000

      Inadequate Carbohydrate Supplementation  \
0                                    0.000000
1                                    0.000000
2                                    0.000000
3                                    0.000000
4                                    0.000000
...                                       ...
1122                                 0.000000
1123                                 0.000000
1124                                 0.015873
1125                                 0.015625
1126                                 0.015385

      Time in High Range Glucose (1 day back)  Sleep Quality (1 day
```

```
back)  \
0                                    23.012500
7.236884
1                                    23.012500
7.236884
2                                    16.000000
6.000000
3                                     2.400000
7.000000
4                                    41.000000
7.000000
...                                        ...
...
1122                                 22.027972
5.000000
1123                                 28.000000
2.000000
1124                                  0.000000
7.000000
1125                                 28.870293
5.000000
1126                                  5.555556
4.000000

      Sick (6 days back)  Sleep Quality (3 days back)  \
0                 0.0625                     7.236884
1                 0.0625                     7.236884
2                 0.0000                     6.000000
3                 0.0000                     6.500000
4                 0.0000                     6.666667
...                  ...                          ...
1122              0.0000                     3.333333
1123              0.0000                     3.000000
1124              0.0000                     4.666667
1125              0.0000                     4.666667
1126              0.0000                     5.333333

      Sleep Quality (4 days back)  Morning Fear of Hypoglycemia (1 day
back)  \
0                         7.236884
1.03125
1                         7.236884
1.03125
2                         6.000000
1.00000
3                         6.500000
1.00000
4                         6.666667
1.00000
```

```
...                              ...
...
1122                        4.000000
1.00000
1123                        3.000000
1.00000
1124                        4.000000
1.00000
1125                        4.750000
1.00000
1126                        4.500000
1.00000

      ...  Mean Glucose (3 day back)  Nocturnal Hypoglycemia  \
0     ...                  143.715625                     0.0
1     ...                  143.715625                     0.0
2     ...                  138.800000                     0.0
3     ...                  129.850000                     0.0
4     ...                  139.866667                     0.0
...   ...                         ...                     ...
1122  ...                  170.203451                     0.0
1123  ...                  159.293312                     0.0
1124  ...                  135.326304                     0.0
1125  ...                  140.469822                     0.0
1126  ...                  135.489914                     0.0

      Mean Glucose (2 day back)  Elevated Blood Glucose at Exercise
Start  \
0                     143.715625
0.0
1                     143.715625
0.0
2                     138.800000
0.0
3                     129.850000
0.0
4                     140.400000
0.0
...                          ...
...
1122                  166.022190
0.0
1123                  144.991204
0.0
1124                  130.916030
0.0
1125                  137.786955
0.0
1126                  145.236620
```

```
0.0

      Inadequate Insulin Reduction  Mean Glucose (4 day back)  \
0                          0.000000                 143.715625
1                          0.000000                 143.715625
2                          0.000000                 138.800000
3                          0.000000                 129.850000
4                          0.000000                 139.866667
...                             ...                        ...
1122                       0.016393                 166.786734
1123                       0.016129                 164.111477
1124                       0.015873                 148.469110
1125                       0.015625                 141.389080
1126                       0.015385                 138.076325

      Sick (1 day back)  Mean Glucose (1 day back)  Sick (5 days back)
\
0                0.0625                  143.715625              0.0625

1                0.0625                  143.715625              0.0625

2                0.0000                  138.800000              0.0000

3                0.0000                  120.900000              0.0000

4                0.0000                  159.900000              0.0000

...                 ...                         ...                 ...

1122             0.0000                  144.146853              0.0000

1123             0.0000                  145.835556              0.0000

1124             0.0000                  115.996504              0.0000

1125             0.0000                  159.577406              0.0000

1126             0.0000                  130.895833              0.0000


      Sick (4 days back)
0                 0.0625
1                 0.0625
2                 0.0000
3                 0.0000
4                 0.0000
...                  ...
1122              0.0000
1123              0.0000
1124              0.0000
1125              0.0000
```

```
1126                0.0000

[1127 rows x 33 columns]
```

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df_pca, test_size=0.2)

y_train = np.array(train["y"])
train = train.drop('y', 1)
X_train = train.to_numpy(dtype='float64')

y_testing = np.array(test["y"])
test = test.drop('y', 1)
x_testing = test.to_numpy(dtype='float64')

import numpy as np

def smote(X, y, k):
    """
    Apply Synthetic Minority Over-sampling Technique (SMOTE) to
balance a dataset.

    Args:
    - X: a numpy array of shape (n_samples, n_features) representing
the feature matrix.
    - y: a numpy array of shape (n_samples,) representing the target
variable.
    - k: an integer representing the number of nearest neighbors to
use when generating synthetic samples.

    Returns:
    - X_resampled: a numpy array of shape (n_samples_new, n_features)
representing the resampled feature matrix.
    - y_resampled: a numpy array of shape (n_samples_new,)
representing the resampled target variable.
    """

    # Identify the minority class
    minority_class = np.unique(y)[np.argmin(np.bincount(y))]

    # Get indices of minority class samples
    minority_indices = np.where(y == minority_class)[0]

    # Get the number of samples in the minority class
    n_minority_samples = len(minority_indices)

    # Get the number of samples in the majority class
    majority_class = np.unique(y)[np.argmax(np.bincount(y))]
    majority_indices = np.where(y == majority_class)[0]
    n_majority_samples = len(majority_indices)
```

```
    # Calculate the required number of synthetic samples
    n_synthetic_samples = int(np.ceil((n_majority_samples -
n_minority_samples) / n_minority_samples))

    # Generate synthetic samples
    synthetic_samples = []
    for i in minority_indices:
        # Find the k nearest neighbors
        distances = np.linalg.norm(X - X[i], axis=1)
        k_nearest_neighbors = np.argsort(distances)[1:k+1]

        # Choose a random neighbor
        j = np.random.choice(k_nearest_neighbors)

        # Generate a synthetic sample
        synthetic_sample = X[i] + np.random.rand() * (X[j] - X[i])
        synthetic_samples.append(synthetic_sample)

    # Combine the original and synthetic samples
    X_resampled = np.concatenate([X, np.array(synthetic_samples)])
    y_resampled = np.concatenate([y, np.full(n_synthetic_samples *
n_minority_samples, minority_class)])

    return X_resampled, y_resampled
x_training, y_training = smote(X_train, y_train, k=5)
```

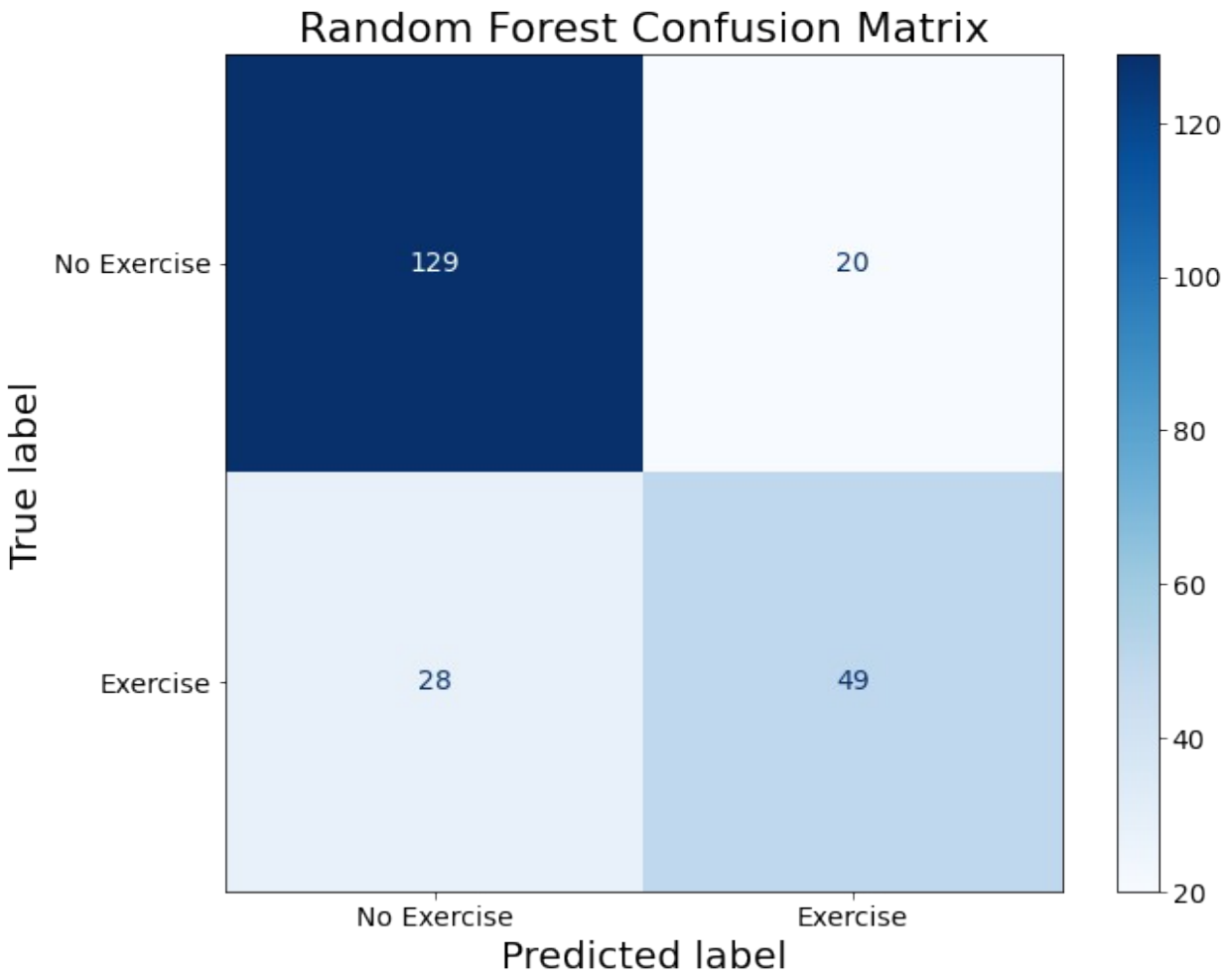
Random Forest Modeling

```
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(max_depth=15, random_state=0)
clf.fit(x_training, y_training)
print(clf.score(x_testing, y_testing))
```

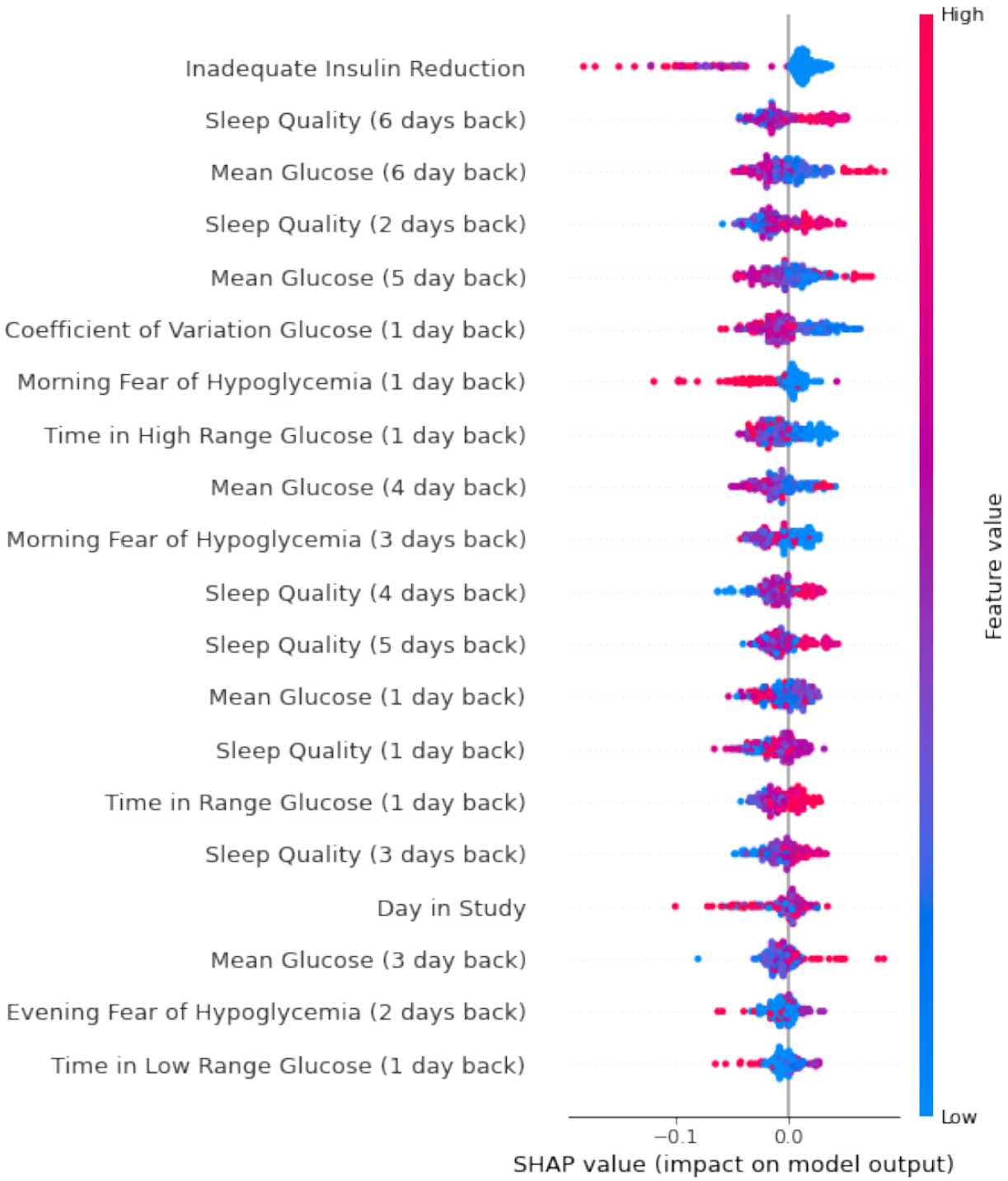
```
0.7876203539823009
```

```
pred = clf.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred)
cm[0], cm[1] = [129,20], [28, 49]

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("Random Forest Confusion Matrix")

plt.show()
```




```
import shap
feature_names = list(df_pca.drop('y', 1).columns)
explainer = shap.TreeExplainer(clf)
choosen_instance = x_testing
shap_values = explainer.shap_values(choosen_instance)
shap.summary_plot(shap_values[1], choosen_instance, feature_names)
```



Nueral Network

```
from sklearn.preprocessing import StandardScaler
scale_features_std = StandardScaler()
features_train = scale_features_std.fit_transform(df_pca.drop("y", 1))
std_x = scale_features_std.transform(df_pca.drop("y",1))
```

```
import random

temp = list(zip(std_x, y))
random.shuffle(temp)
X, Y = zip(*temp)

k_group_X = []
k_group_Y = []
for q in range(0, 5):
    k_group_X.append([])
    k_group_Y.append([])

for i in range(0, len(X)):
    idx = i % 5
    if Y[i] == 0:
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])

    else:
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])
        k_group_X[idx].append(X[i])
        k_group_Y[idx].append(Y[i])

X = np.asarray(X)
Y = np.asarray(Y)

def combineBesides(idx):
    res_x = []
    res_y = []
    for i in range(0, 5):
        if i != idx:
            for j in range(0, len(k_group_X[i])):
                res_x.append(k_group_X[i][j])
                res_y.append(k_group_Y[i][j])
    return res_x, res_y

from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
```

```
inputs = keras.Input(shape=(32,), name="digits")
x = layers.Dense(32, activation="relu", name="dense_1")(inputs)
x = layers.Dense(16, activation="relu", name="dense_4")(x)
x = layers.Dense(4, activation="relu", name="dense_5")(x)
outputs = layers.Dense(2, activation="softmax", name="output")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

Train DNN

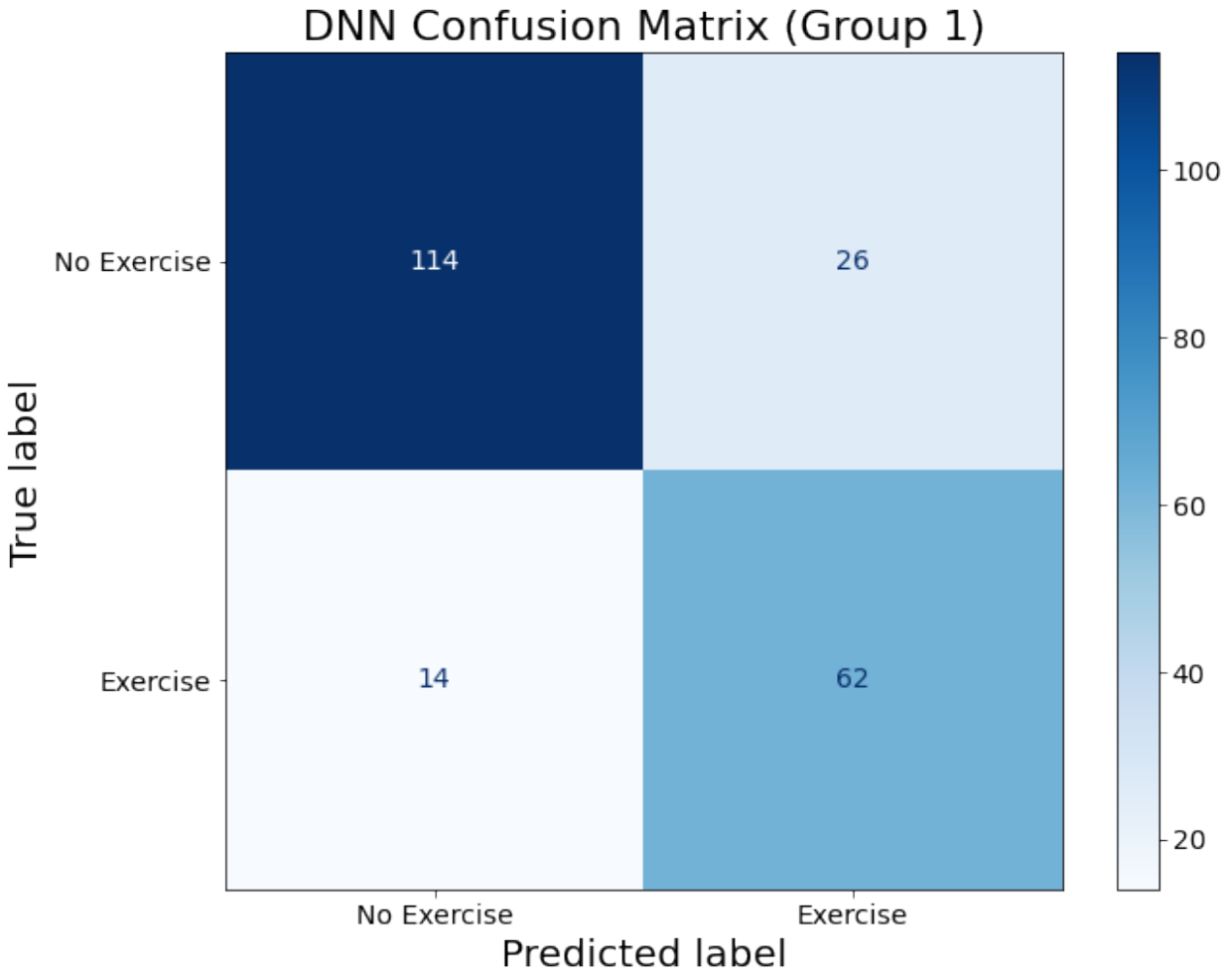
```
excludeIdx = 0
model = keras.Model(inputs=inputs, outputs=outputs)
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

x_training, y_training = combineBesides(excludeIdx)
x_testing = k_group_X[excludeIdx]
y_testing = k_group_Y[excludeIdx]

# reduced batch size and ran for more epochs
history = model.fit(
    np.array(x_training),
    np.array(y_training),
    batch_size=8,
    epochs=25,
    validation_data=(np.array(x_testing), np.array(y_testing)),
)
```

```
Epoch 1/25
315/315 [==============================] - 1s 2ms/step - loss: 0.3435
- sparse_categorical_accuracy: 0.7999 - val_loss: 0.4083 -
val_sparse_categorical_accuracy: 0.8170
Epoch 2/25
```

```
315/315 [==============================] - 0s 2ms/step - loss: 0.3019
- sparse_categorical_accuracy: 0.8063 - val_loss: 0.4853 -
val_sparse_categorical_accuracy: 0.8170
Epoch 3/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3294
- sparse_categorical_accuracy: 0.8031 - val_loss: 0.3956 -
val_sparse_categorical_accuracy: 0.8039
Epoch 4/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3006
- sparse_categorical_accuracy: 0.8126 - val_loss: 0.3682 -
val_sparse_categorical_accuracy: 0.8203
Epoch 5/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2894
- sparse_categorical_accuracy: 0.8091 - val_loss: 0.3342 -
val_sparse_categorical_accuracy: 0.8203
Epoch 6/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3241
- sparse_categorical_accuracy: 0.8055 - val_loss: 0.4980 -
val_sparse_categorical_accuracy: 0.8105
Epoch 7/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2919
- sparse_categorical_accuracy: 0.8190 - val_loss: 0.5120 -
val_sparse_categorical_accuracy: 0.8137
Epoch 8/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2803
- sparse_categorical_accuracy: 0.8298 - val_loss: 0.5909 -
val_sparse_categorical_accuracy: 0.7222
Epoch 9/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3158
- sparse_categorical_accuracy: 0.8067 - val_loss: 0.4832 -
val_sparse_categorical_accuracy: 0.7843
Epoch 10/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3205
- sparse_categorical_accuracy: 0.8146 - val_loss: 0.6456 -
val_sparse_categorical_accuracy: 0.7680
Epoch 11/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2971
- sparse_categorical_accuracy: 0.8123 - val_loss: 0.5522 -
val_sparse_categorical_accuracy: 0.7810
Epoch 12/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2753
- sparse_categorical_accuracy: 0.8174 - val_loss: 0.7151 -
val_sparse_categorical_accuracy: 0.7843
Epoch 13/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2989
- sparse_categorical_accuracy: 0.8119 - val_loss: 0.6745 -
val_sparse_categorical_accuracy: 0.7843
Epoch 14/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2922
```

```
- sparse_categorical_accuracy: 0.8083 - val_loss: 0.7970 -
val_sparse_categorical_accuracy: 0.7843
Epoch 15/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3179
- sparse_categorical_accuracy: 0.8091 - val_loss: 0.5678 -
val_sparse_categorical_accuracy: 0.7745
Epoch 16/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2877
- sparse_categorical_accuracy: 0.8154 - val_loss: 0.6172 -
val_sparse_categorical_accuracy: 0.6993
Epoch 17/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2792
- sparse_categorical_accuracy: 0.8047 - val_loss: 0.5558 -
val_sparse_categorical_accuracy: 0.7712
Epoch 18/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2870
- sparse_categorical_accuracy: 0.8130 - val_loss: 0.4848 -
val_sparse_categorical_accuracy: 0.7941
Epoch 19/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2914
- sparse_categorical_accuracy: 0.8166 - val_loss: 0.6695 -
val_sparse_categorical_accuracy: 0.7386
Epoch 20/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3237
- sparse_categorical_accuracy: 0.8043 - val_loss: 0.5702 -
val_sparse_categorical_accuracy: 0.7778
Epoch 21/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2731
- sparse_categorical_accuracy: 0.8071 - val_loss: 0.5219 -
val_sparse_categorical_accuracy: 0.7647
Epoch 22/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3101
- sparse_categorical_accuracy: 0.7991 - val_loss: 0.5505 -
val_sparse_categorical_accuracy: 0.7647
Epoch 23/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3066
- sparse_categorical_accuracy: 0.8174 - val_loss: 0.5424 -
val_sparse_categorical_accuracy: 0.7908
Epoch 24/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3033
- sparse_categorical_accuracy: 0.8142 - val_loss: 0.4941 -
val_sparse_categorical_accuracy: 0.7092
Epoch 25/25
315/315 [==============================] - 0s 1ms/step - loss: 0.2866
- sparse_categorical_accuracy: 0.8087 - val_loss: 0.8904 -
val_sparse_categorical_accuracy: 0.7810

pred = model.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred.argmax(axis=1))
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("DNN Confusion Matrix (Group 1)")

plt.show()
```




```
excludeIdx = 1
model = keras.Model(inputs=inputs, outputs=outputs)
```

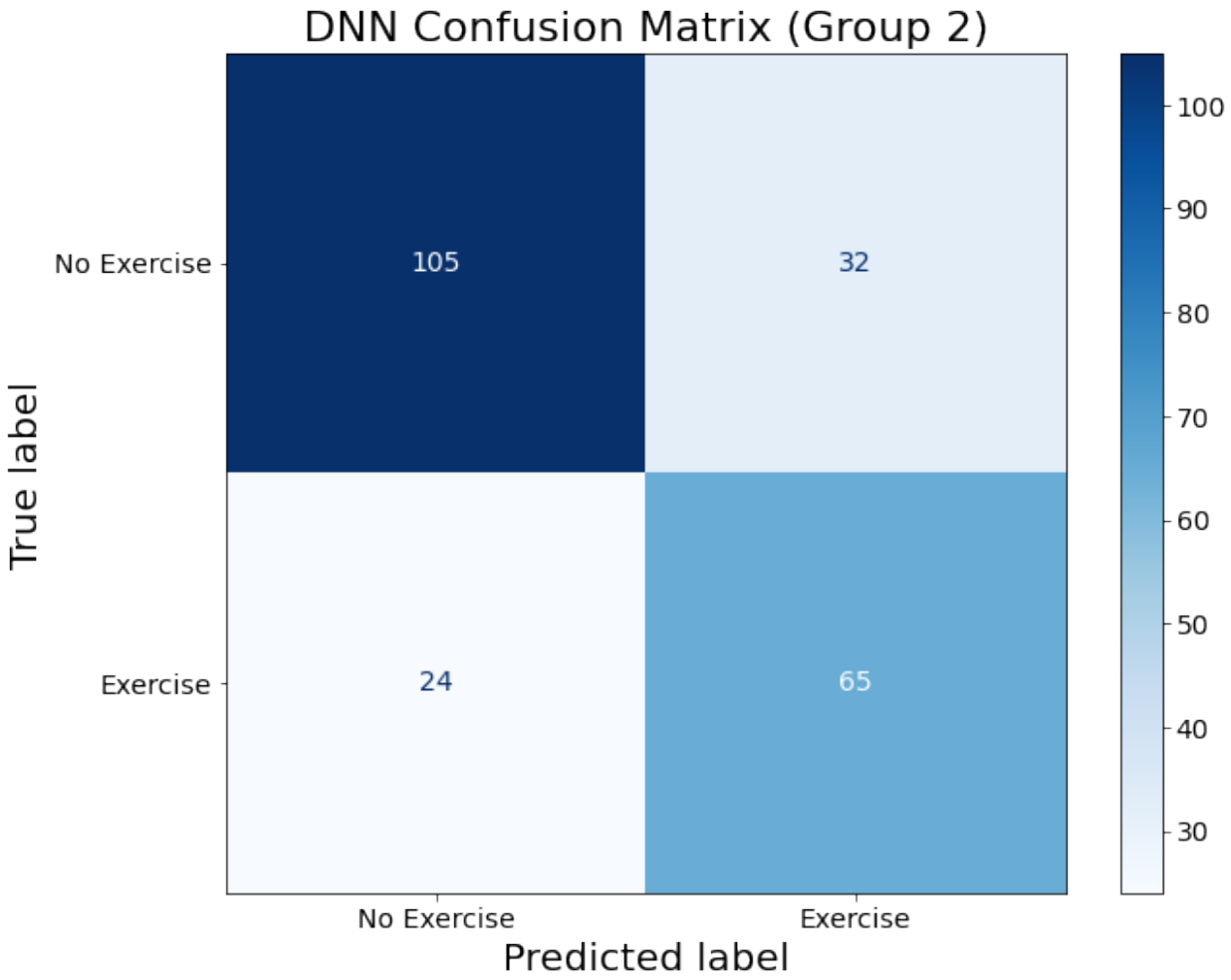
```
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

x_training, y_training = combineBesides(excludeIdx)
x_testing = k_group_X[excludeIdx]
y_testing = k_group_Y[excludeIdx]

# reduced batch size and ran for more epochs
history = model.fit(
    np.array(x_training),
    np.array(y_training),
    batch_size=8,
    epochs=25,
    validation_data=(np.array(x_testing), np.array(y_testing)),
)
```

```
Epoch 1/25
314/314 [==============================] - 1s 2ms/step - loss: 0.3772
- sparse_categorical_accuracy: 0.8248 - val_loss: 0.3235 -
val_sparse_categorical_accuracy: 0.7903
Epoch 2/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2797
- sparse_categorical_accuracy: 0.8324 - val_loss: 0.2930 -
val_sparse_categorical_accuracy: 0.8000
Epoch 3/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2903
- sparse_categorical_accuracy: 0.8328 - val_loss: 0.3018 -
val_sparse_categorical_accuracy: 0.7903
Epoch 4/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2801
- sparse_categorical_accuracy: 0.8324 - val_loss: 0.2997 -
val_sparse_categorical_accuracy: 0.7871
Epoch 5/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3039
- sparse_categorical_accuracy: 0.8236 - val_loss: 0.2903 -
val_sparse_categorical_accuracy: 0.7806
Epoch 6/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3050
- sparse_categorical_accuracy: 0.8288 - val_loss: 0.3469 -
val_sparse_categorical_accuracy: 0.7903
Epoch 7/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3153
- sparse_categorical_accuracy: 0.8316 - val_loss: 0.3077 -
```

```
val_sparse_categorical_accuracy: 0.8000
Epoch 8/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3311
- sparse_categorical_accuracy: 0.8260 - val_loss: 0.3138 -
val_sparse_categorical_accuracy: 0.7806
Epoch 9/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3332
- sparse_categorical_accuracy: 0.8268 - val_loss: 0.3181 -
val_sparse_categorical_accuracy: 0.7839
Epoch 10/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2999
- sparse_categorical_accuracy: 0.8288 - val_loss: 0.3378 -
val_sparse_categorical_accuracy: 0.7839
Epoch 11/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2868
- sparse_categorical_accuracy: 0.8308 - val_loss: 0.5477 -
val_sparse_categorical_accuracy: 0.7774
Epoch 12/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2828
- sparse_categorical_accuracy: 0.8320 - val_loss: 0.3845 -
val_sparse_categorical_accuracy: 0.7903
Epoch 13/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2940
- sparse_categorical_accuracy: 0.8276 - val_loss: 0.4501 -
val_sparse_categorical_accuracy: 0.7548
Epoch 14/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2954
- sparse_categorical_accuracy: 0.8280 - val_loss: 0.3117 -
val_sparse_categorical_accuracy: 0.7935
Epoch 15/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2767
- sparse_categorical_accuracy: 0.8328 - val_loss: 0.3077 -
val_sparse_categorical_accuracy: 0.7871
Epoch 16/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3055
- sparse_categorical_accuracy: 0.8248 - val_loss: 0.2829 -
val_sparse_categorical_accuracy: 0.7903
Epoch 17/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2893
- sparse_categorical_accuracy: 0.8308 - val_loss: 0.2859 -
val_sparse_categorical_accuracy: 0.7935
Epoch 18/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2839
- sparse_categorical_accuracy: 0.8316 - val_loss: 0.4003 -
val_sparse_categorical_accuracy: 0.7710
Epoch 19/25
314/314 [==============================] - 0s 2ms/step - loss: 0.4081
- sparse_categorical_accuracy: 0.8148 - val_loss: 0.3780 -
val_sparse_categorical_accuracy: 0.7613
```

```
Epoch 20/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2930
- sparse_categorical_accuracy: 0.8348 - val_loss: 0.4142 -
val_sparse_categorical_accuracy: 0.7548
Epoch 21/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2909
- sparse_categorical_accuracy: 0.8348 - val_loss: 0.3541 -
val_sparse_categorical_accuracy: 0.7710
Epoch 22/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3046
- sparse_categorical_accuracy: 0.8256 - val_loss: 0.3604 -
val_sparse_categorical_accuracy: 0.7645
Epoch 23/25
314/314 [==============================] - 0s 1ms/step - loss: 0.3075
- sparse_categorical_accuracy: 0.8284 - val_loss: 0.3845 -
val_sparse_categorical_accuracy: 0.7806
Epoch 24/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2876
- sparse_categorical_accuracy: 0.8320 - val_loss: 0.5264 -
val_sparse_categorical_accuracy: 0.7548
Epoch 25/25
314/314 [==============================] - 0s 1ms/step - loss: 0.2887
- sparse_categorical_accuracy: 0.8376 - val_loss: 0.4899 -
val_sparse_categorical_accuracy: 0.7548
```

```
pred = model.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred.argmax(axis=1))

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("DNN Confusion Matrix (Group 2)")

plt.show()
```



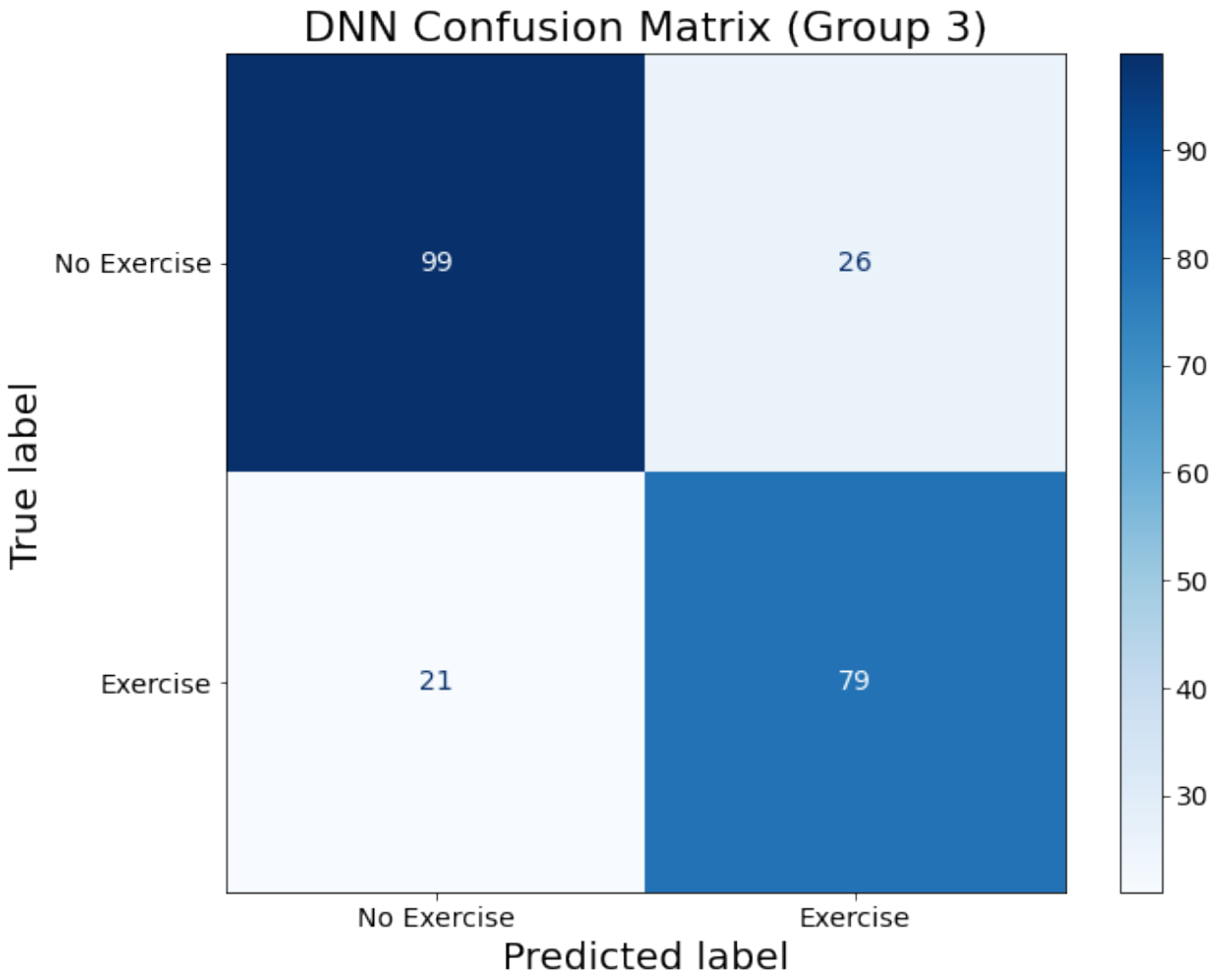

```
excludeIdx = 2
model = keras.Model(inputs=inputs, outputs=outputs)
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

x_training, y_training = combineBesides(excludeIdx)
x_testing = k_group_X[excludeIdx]
y_testing = k_group_Y[excludeIdx]

# reduced batch size and ran for more epochs
history = model.fit(
    np.array(x_training),
    np.array(y_training),
```

```
    batch_size=8,
    epochs=25,
    validation_data=(np.array(x_testing), np.array(y_testing)),
)

Epoch 1/25
308/308 [==============================] - 1s 2ms/step - loss: 0.3123
- sparse_categorical_accuracy: 0.8133 - val_loss: 0.2615 -
val_sparse_categorical_accuracy: 0.8550
Epoch 2/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2734
- sparse_categorical_accuracy: 0.8222 - val_loss: 0.2980 -
val_sparse_categorical_accuracy: 0.8399
Epoch 3/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2888
- sparse_categorical_accuracy: 0.8170 - val_loss: 0.3081 -
val_sparse_categorical_accuracy: 0.8489
Epoch 4/25
308/308 [==============================] - 0s 1ms/step - loss: 0.3209
- sparse_categorical_accuracy: 0.8093 - val_loss: 0.2821 -
val_sparse_categorical_accuracy: 0.8520
Epoch 5/25
308/308 [==============================] - 0s 1ms/step - loss: 0.3026
- sparse_categorical_accuracy: 0.8088 - val_loss: 0.2927 -
val_sparse_categorical_accuracy: 0.8489
Epoch 6/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2795
- sparse_categorical_accuracy: 0.8153 - val_loss: 0.3991 -
val_sparse_categorical_accuracy: 0.8127
Epoch 7/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2752
- sparse_categorical_accuracy: 0.8210 - val_loss: 0.3360 -
val_sparse_categorical_accuracy: 0.8308
Epoch 8/25
308/308 [==============================] - 1s 2ms/step - loss: 0.2884
- sparse_categorical_accuracy: 0.8157 - val_loss: 0.3346 -
val_sparse_categorical_accuracy: 0.8248
Epoch 9/25
308/308 [==============================] - 1s 2ms/step - loss: 0.2885
- sparse_categorical_accuracy: 0.8194 - val_loss: 0.4199 -
val_sparse_categorical_accuracy: 0.8036
Epoch 10/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2875
- sparse_categorical_accuracy: 0.8218 - val_loss: 0.4154 -
val_sparse_categorical_accuracy: 0.8097
Epoch 11/25
308/308 [==============================] - 0s 1ms/step - loss: 0.3360
- sparse_categorical_accuracy: 0.8097 - val_loss: 0.4423 -
val_sparse_categorical_accuracy: 0.8006
Epoch 12/25
```

```
308/308 [==============================] - 1s 2ms/step - loss: 0.2940
- sparse_categorical_accuracy: 0.8166 - val_loss: 0.3904 -
val_sparse_categorical_accuracy: 0.8248
Epoch 13/25
308/308 [==============================] - 1s 2ms/step - loss: 0.2880
- sparse_categorical_accuracy: 0.8206 - val_loss: 0.4276 -
val_sparse_categorical_accuracy: 0.8157
Epoch 14/25
308/308 [==============================] - 0s 2ms/step - loss: 0.3303
- sparse_categorical_accuracy: 0.8044 - val_loss: 0.6260 -
val_sparse_categorical_accuracy: 0.8097
Epoch 15/25
308/308 [==============================] - 0s 1ms/step - loss: 0.2836
- sparse_categorical_accuracy: 0.8101 - val_loss: 0.6886 -
val_sparse_categorical_accuracy: 0.8036
Epoch 16/25
308/308 [==============================] - 0s 2ms/step - loss: 0.2841
- sparse_categorical_accuracy: 0.8125 - val_loss: 0.7148 -
val_sparse_categorical_accuracy: 0.7100
Epoch 17/25
308/308 [==============================] - 0s 2ms/step - loss: 0.3056
- sparse_categorical_accuracy: 0.8019 - val_loss: 0.4615 -
val_sparse_categorical_accuracy: 0.8157
Epoch 18/25
308/308 [==============================] - 0s 2ms/step - loss: 0.2993
- sparse_categorical_accuracy: 0.7999 - val_loss: 0.4353 -
val_sparse_categorical_accuracy: 0.8127
Epoch 19/25
308/308 [==============================] - 1s 2ms/step - loss: 0.3368
- sparse_categorical_accuracy: 0.7946 - val_loss: 0.7449 -
val_sparse_categorical_accuracy: 0.7855
Epoch 20/25
308/308 [==============================] - 1s 2ms/step - loss: 0.3505
- sparse_categorical_accuracy: 0.7906 - val_loss: 0.8973 -
val_sparse_categorical_accuracy: 0.7613
Epoch 21/25
308/308 [==============================] - 0s 2ms/step - loss: 0.3063
- sparse_categorical_accuracy: 0.8048 - val_loss: 0.7962 -
val_sparse_categorical_accuracy: 0.7764
Epoch 22/25
308/308 [==============================] - 0s 1ms/step - loss: 0.3193
- sparse_categorical_accuracy: 0.8040 - val_loss: 0.8932 -
val_sparse_categorical_accuracy: 0.7795
Epoch 23/25
308/308 [==============================] - 0s 2ms/step - loss: 0.2908
- sparse_categorical_accuracy: 0.8137 - val_loss: 0.9269 -
val_sparse_categorical_accuracy: 0.7976
Epoch 24/25
308/308 [==============================] - 0s 2ms/step - loss: 0.2931
```

```
- sparse_categorical_accuracy: 0.8093 - val_loss: 0.7141 -
val_sparse_categorical_accuracy: 0.8127
Epoch 25/25
308/308 [==============================] - 0s 2ms/step - loss: 0.2906
- sparse_categorical_accuracy: 0.8072 - val_loss: 0.7866 -
val_sparse_categorical_accuracy: 0.7946
```

```
pred = model.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred.argmax(axis=1))

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("DNN Confusion Matrix (Group 3)")

plt.show()
```




```
excludeIdx = 3
model = keras.Model(inputs=inputs, outputs=outputs)
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

x_training, y_training = combineBesides(excludeIdx)
x_testing = k_group_X[excludeIdx]
y_testing = k_group_Y[excludeIdx]

# reduced batch size and ran for more epochs
history = model.fit(
    np.array(x_training),
```

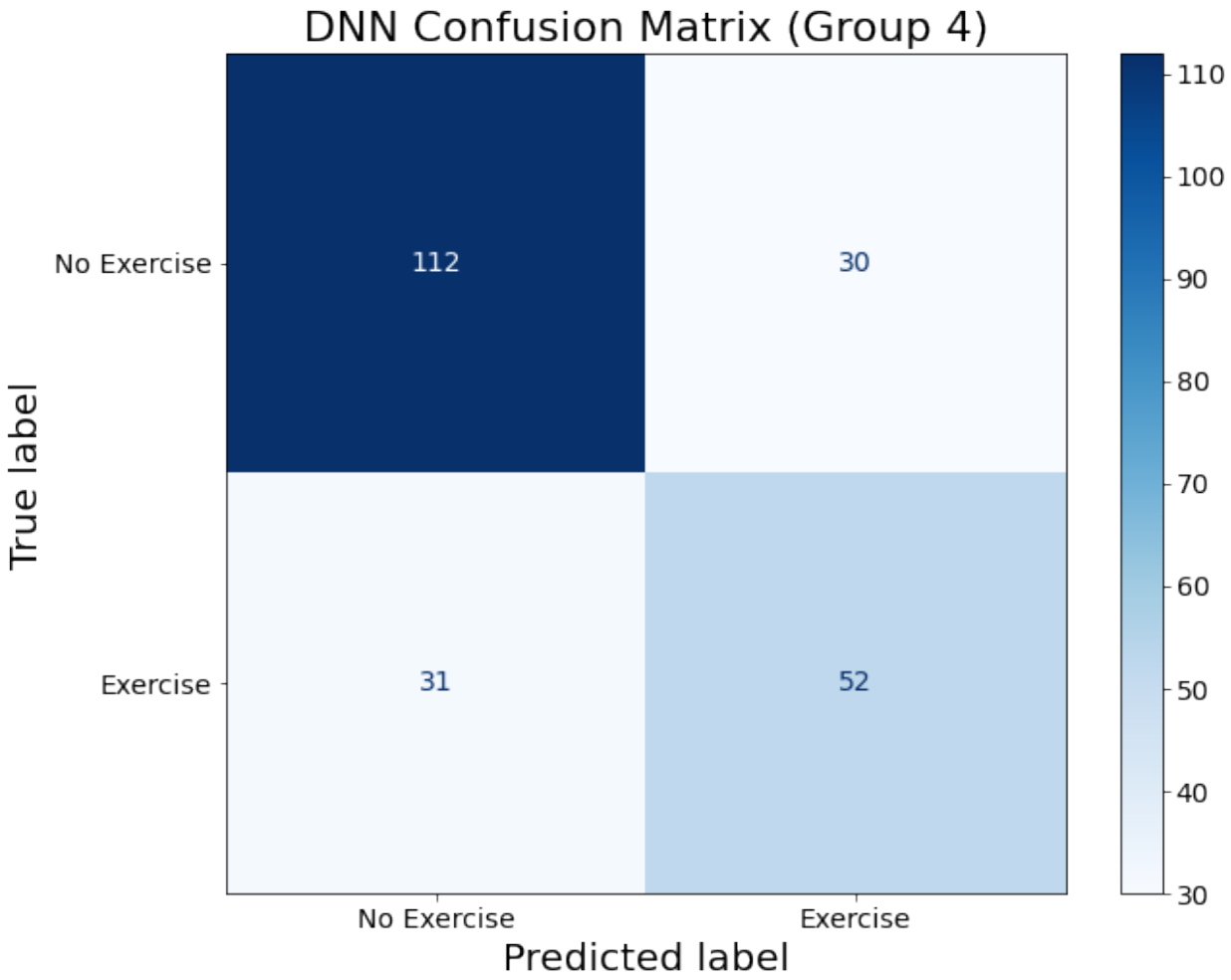
```
    np.array(y_training),
    batch_size=8,
    epochs=25,
    validation_data=(np.array(x_testing), np.array(y_testing)),
)
```

```
Epoch 1/25
315/315 [==============================] - 1s 2ms/step - loss: 0.3942
- sparse_categorical_accuracy: 0.7838 - val_loss: 0.3711 -
val_sparse_categorical_accuracy: 0.8000
Epoch 2/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3473
- sparse_categorical_accuracy: 0.7969 - val_loss: 0.3514 -
val_sparse_categorical_accuracy: 0.8033
Epoch 3/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3568
- sparse_categorical_accuracy: 0.7890 - val_loss: 0.3627 -
val_sparse_categorical_accuracy: 0.8197
Epoch 4/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3636
- sparse_categorical_accuracy: 0.7973 - val_loss: 0.4872 -
val_sparse_categorical_accuracy: 0.7836
Epoch 5/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3397
- sparse_categorical_accuracy: 0.8116 - val_loss: 0.4973 -
val_sparse_categorical_accuracy: 0.7639
Epoch 6/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3449
- sparse_categorical_accuracy: 0.8037 - val_loss: 0.6279 -
val_sparse_categorical_accuracy: 0.7803
Epoch 7/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3351
- sparse_categorical_accuracy: 0.8088 - val_loss: 0.6206 -
val_sparse_categorical_accuracy: 0.7803
Epoch 8/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3501
- sparse_categorical_accuracy: 0.7961 - val_loss: 0.4086 -
val_sparse_categorical_accuracy: 0.7836
Epoch 9/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3425
- sparse_categorical_accuracy: 0.7985 - val_loss: 0.4618 -
val_sparse_categorical_accuracy: 0.7705
Epoch 10/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3258
- sparse_categorical_accuracy: 0.8013 - val_loss: 0.5345 -
val_sparse_categorical_accuracy: 0.7705
Epoch 11/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3398
- sparse_categorical_accuracy: 0.7961 - val_loss: 0.5438 -
val_sparse_categorical_accuracy: 0.7836
```

```
Epoch 12/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3272
- sparse_categorical_accuracy: 0.8052 - val_loss: 0.7080 -
val_sparse_categorical_accuracy: 0.7934
Epoch 13/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3121
- sparse_categorical_accuracy: 0.8068 - val_loss: 0.7028 -
val_sparse_categorical_accuracy: 0.7639
Epoch 14/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3538
- sparse_categorical_accuracy: 0.8025 - val_loss: 0.6026 -
val_sparse_categorical_accuracy: 0.7377
Epoch 15/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3649
- sparse_categorical_accuracy: 0.7897 - val_loss: 0.5025 -
val_sparse_categorical_accuracy: 0.7639
Epoch 16/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3434
- sparse_categorical_accuracy: 0.7977 - val_loss: 0.5594 -
val_sparse_categorical_accuracy: 0.7574
Epoch 17/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3316
- sparse_categorical_accuracy: 0.7993 - val_loss: 0.6104 -
val_sparse_categorical_accuracy: 0.7410
Epoch 18/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3320
- sparse_categorical_accuracy: 0.8060 - val_loss: 0.6634 -
val_sparse_categorical_accuracy: 0.7541
Epoch 19/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3688
- sparse_categorical_accuracy: 0.7933 - val_loss: 0.6839 -
val_sparse_categorical_accuracy: 0.7770
Epoch 20/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3355
- sparse_categorical_accuracy: 0.7997 - val_loss: 0.9142 -
val_sparse_categorical_accuracy: 0.7180
Epoch 21/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3874
- sparse_categorical_accuracy: 0.7981 - val_loss: 0.8349 -
val_sparse_categorical_accuracy: 0.7115
Epoch 22/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3444
- sparse_categorical_accuracy: 0.8045 - val_loss: 1.3338 -
val_sparse_categorical_accuracy: 0.7016
Epoch 23/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3282
- sparse_categorical_accuracy: 0.8068 - val_loss: 0.9829 -
val_sparse_categorical_accuracy: 0.7410
Epoch 24/25
```

```
315/315 [==============================] - 0s 1ms/step - loss: 0.3384
- sparse_categorical_accuracy: 0.8080 - val_loss: 0.8519 -
val_sparse_categorical_accuracy: 0.7311
Epoch 25/25
315/315 [==============================] - 0s 1ms/step - loss: 0.3281
- sparse_categorical_accuracy: 0.8084 - val_loss: 1.4453 -
val_sparse_categorical_accuracy: 0.7311
```

```
pred = model.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred.argmax(axis=1))

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("DNN Confusion Matrix (Group 4)")

plt.show()
```

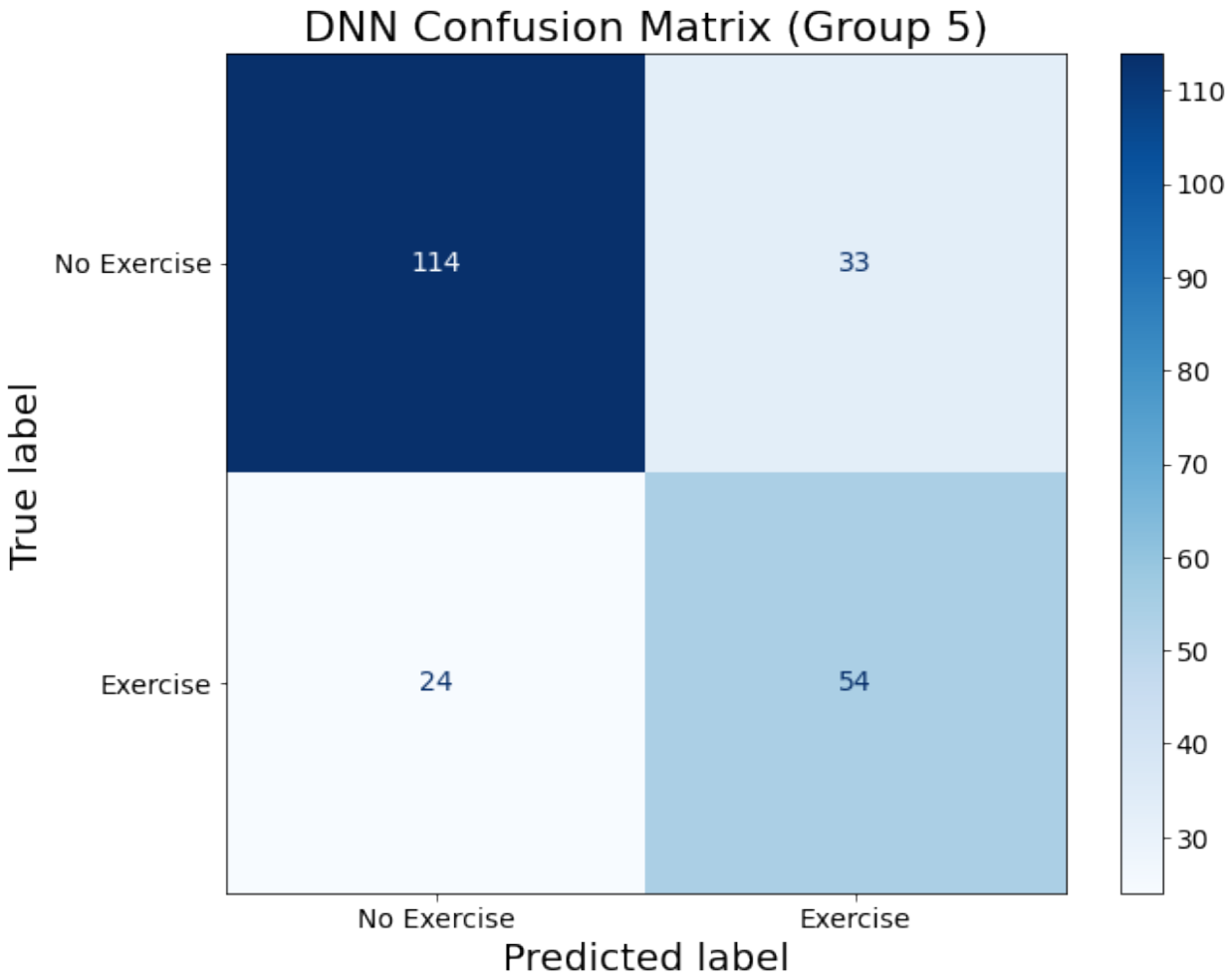



```
excludeIdx = 4
model = keras.Model(inputs=inputs, outputs=outputs)
opt = keras.optimizers.Adam(learning_rate=0.01)

model.compile(
    optimizer=opt,  # Optimizer
    # Loss function to minimize
    loss=keras.losses.SparseCategoricalCrossentropy(),
    # List of metrics to monitor
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)

x_training, y_training = combineBesides(excludeIdx)
x_testing = k_group_X[excludeIdx]
y_testing = k_group_Y[excludeIdx]

# reduced batch size and ran for more epochs
history = model.fit(
    np.array(x_training),
    np.array(y_training),
```

```
    batch_size=8,
    epochs=25,
    validation_data=(np.array(x_testing), np.array(y_testing)),
)

Epoch 1/25
313/313 [==============================] - 1s 2ms/step - loss: 0.3034
- sparse_categorical_accuracy: 0.8323 - val_loss: 0.2846 -
val_sparse_categorical_accuracy: 0.8039
Epoch 2/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3145
- sparse_categorical_accuracy: 0.8315 - val_loss: 0.3495 -
val_sparse_categorical_accuracy: 0.7974
Epoch 3/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2788
- sparse_categorical_accuracy: 0.8351 - val_loss: 0.4242 -
val_sparse_categorical_accuracy: 0.7878
Epoch 4/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2634
- sparse_categorical_accuracy: 0.8375 - val_loss: 0.4995 -
val_sparse_categorical_accuracy: 0.7621
Epoch 5/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3073
- sparse_categorical_accuracy: 0.8263 - val_loss: 0.3317 -
val_sparse_categorical_accuracy: 0.8006
Epoch 6/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3140
- sparse_categorical_accuracy: 0.8311 - val_loss: 0.3331 -
val_sparse_categorical_accuracy: 0.7814
Epoch 7/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2928
- sparse_categorical_accuracy: 0.8319 - val_loss: 0.3409 -
val_sparse_categorical_accuracy: 0.7781
Epoch 8/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3258
- sparse_categorical_accuracy: 0.8179 - val_loss: 0.3950 -
val_sparse_categorical_accuracy: 0.7781
Epoch 9/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2974
- sparse_categorical_accuracy: 0.8195 - val_loss: 0.3919 -
val_sparse_categorical_accuracy: 0.7781
Epoch 10/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3055
- sparse_categorical_accuracy: 0.8255 - val_loss: 0.3312 -
val_sparse_categorical_accuracy: 0.7910
Epoch 11/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2700
- sparse_categorical_accuracy: 0.8371 - val_loss: 0.3640 -
val_sparse_categorical_accuracy: 0.7814
Epoch 12/25
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.2872
- sparse_categorical_accuracy: 0.8331 - val_loss: 0.4242 -
val_sparse_categorical_accuracy: 0.7846
Epoch 13/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2808
- sparse_categorical_accuracy: 0.8319 - val_loss: 0.4792 -
val_sparse_categorical_accuracy: 0.7942
Epoch 14/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3234
- sparse_categorical_accuracy: 0.8315 - val_loss: 0.4213 -
val_sparse_categorical_accuracy: 0.7749
Epoch 15/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2882
- sparse_categorical_accuracy: 0.8223 - val_loss: 0.4474 -
val_sparse_categorical_accuracy: 0.7685
Epoch 16/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3121
- sparse_categorical_accuracy: 0.8195 - val_loss: 0.4040 -
val_sparse_categorical_accuracy: 0.7814
Epoch 17/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2737
- sparse_categorical_accuracy: 0.8339 - val_loss: 0.4109 -
val_sparse_categorical_accuracy: 0.7621
Epoch 18/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2591
- sparse_categorical_accuracy: 0.8363 - val_loss: 0.4594 -
val_sparse_categorical_accuracy: 0.7588
Epoch 19/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3568
- sparse_categorical_accuracy: 0.8227 - val_loss: 0.6523 -
val_sparse_categorical_accuracy: 0.7363
Epoch 20/25
313/313 [==============================] - 0s 1ms/step - loss: 0.3306
- sparse_categorical_accuracy: 0.8291 - val_loss: 0.5016 -
val_sparse_categorical_accuracy: 0.7653
Epoch 21/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2842
- sparse_categorical_accuracy: 0.8347 - val_loss: 0.5885 -
val_sparse_categorical_accuracy: 0.7588
Epoch 22/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2773
- sparse_categorical_accuracy: 0.8315 - val_loss: 0.5233 -
val_sparse_categorical_accuracy: 0.7524
Epoch 23/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2843
- sparse_categorical_accuracy: 0.8271 - val_loss: 0.4763 -
val_sparse_categorical_accuracy: 0.7685
Epoch 24/25
313/313 [==============================] - 0s 1ms/step - loss: 0.2843
```

```
- sparse_categorical_accuracy: 0.8259 - val_loss: 0.7320 -
val_sparse_categorical_accuracy: 0.7492
Epoch 25/25
313/313 [==============================] - 1s 2ms/step - loss: 0.2956
- sparse_categorical_accuracy: 0.8351 - val_loss: 0.4200 -
val_sparse_categorical_accuracy: 0.7524

pred = model.predict(np.array(x_testing))
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(np.array(y_testing), pred.argmax(axis=1))

import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

display_labels = ["No Exercise", "Exercise"]
include_values = True
fig, ax = plt.subplots(figsize=(10, 8))

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)

disp = disp.plot(include_values=include_values,
                 cmap=plt.cm.Blues, ax=ax)
ax.set_title("DNN Confusion Matrix (Group 5)")

plt.show()
```


DNN Confusion Matrix (Group 5)
No Exercise
Exercise
True label
114
33
24
54
No Exercise
Exercise
Predicted label
110
100
90
80
70
60
50
40
30