3 - Evaluate Submission

June 20, 2023

1 Evaluate Submissions

2 Load libraries

[ ]:
```
import pandas as pd
import numpy as np
import math
import os
import sys

# For evaluation
import sklearn.metrics as metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

# Plotting
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
from sklearn.exceptions import UndefinedMetricWarning
warnings.filterwarnings('ignore', category=UndefinedMetricWarning)
```

3 Load configuration

[ ]:
```
ANSWER_PATH = '/workspaces/test/DO NOT PUBLISH/answer.csv'
SUBMISSIONS_PATH = '/workspaces/submissions'

FIGSIZE = (20, 7)
FIGSIZE_CM = (13, 7)
```

4 Load answer & submissions

[ ]:
```
# Specify team name
#
TEAMNAME = 'teamname'

# Load answers
#
y_true = pd.read_csv(ANSWER_PATH,
                     index_col=0
                     ).set_index('Timestamp')

# Load team's submission
#
y_pred = pd.read_csv(f'{SUBMISSIONS_PATH}/submission_{TEAMNAME}.csv',
                     index_col=0
                     ).set_index('Timestamp').rename(columns={
                      'final_wearing_off': f'final_wearing_off_{TEAMNAME}'})

# Force set index i.e., timestmap as datetime
#
y_true.index = pd.to_datetime(y_true.index)
y_pred.index = pd.to_datetime(y_pred.index)

# print(len(y_true), len(y_pred))
```

5 Combine answers & submissions

[ ]:
```
# Combine answers and submission
#   1. Do a left join between answers and submission on participant & timestamp
#   2. Select only the columns we need
#   3. Rename columns

#
#   1.
#   2.
#   3.
base_forecasts_output = y_true.reset_index().merge(
  y_pred.reset_index(),
  how='left',
  on=['participant', 'Timestamp']
)[[
  'Timestamp', 'participant',
  'final_wearing_off', 'final_wearing_off_teamname'
  ]]
base_forecasts_output.columns = ['Timestamp', 'patient_id',
```

```
                                  'ground_truth', 'forecasted_wearing_off']
base_forecasts_output.set_index('Timestamp', inplace=True)
base_forecasts_output
```

[ ]:

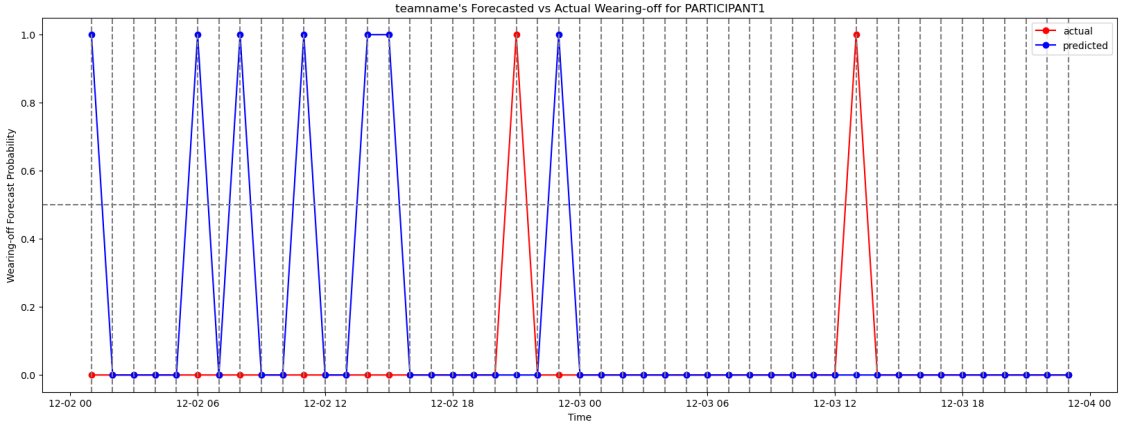
Timestamp	patient_id	ground_truth	forecasted_wearing_off
2021-12-02 01:00:00	participant1	0	1
2021-12-02 02:00:00	participant1	0	0
2021-12-02 03:00:00	participant1	0	0
2021-12-02 04:00:00	participant1	0	0
2021-12-02 05:00:00	participant1	0	0
…	…	…	…
2021-12-24 19:00:00	participant10	0	0
2021-12-24 20:00:00	participant10	0	0
2021-12-24 21:00:00	participant10	0	0
2021-12-24 22:00:00	participant10	0	0
2021-12-24 23:00:00	participant10	0	0

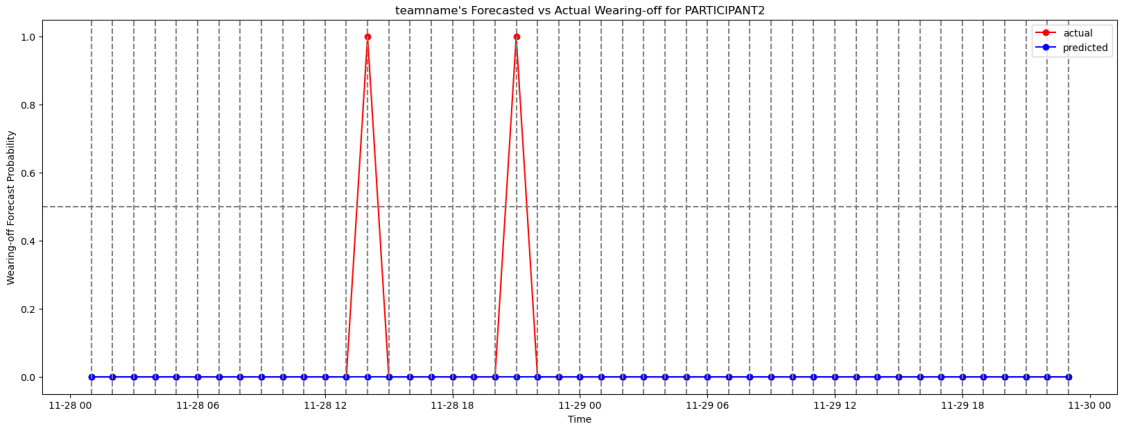
[440 rows x 3 columns]

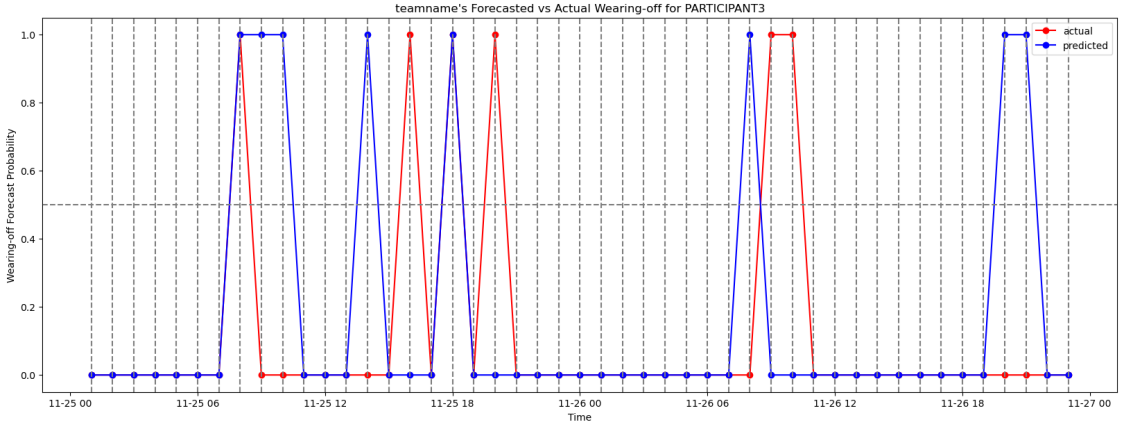
5.1 Plot Actual vs Forecast

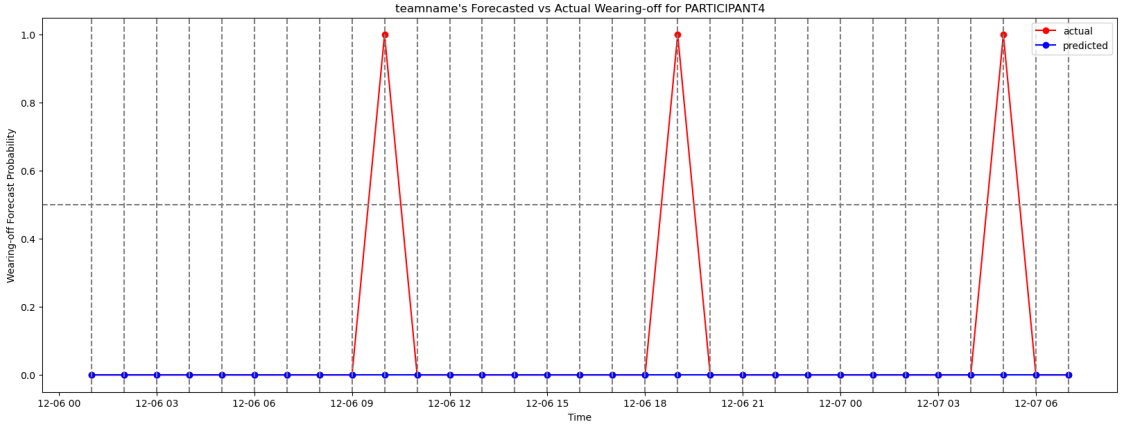
[ ]:
```
# Generate actual vs forecast for each participant
#
for i in range(1, 11):
  user = f'participant{i}'
  forecasts_output = base_forecasts_output.query(
    f'patient_id == "{user}"')

  # Plot answer, & submission on the same plot to show the difference
  #
  plt.figure(figsize=FIGSIZE)
  plt.plot(forecasts_output.ground_truth,
           label='actual', color='red', marker='o',)
  plt.plot(forecasts_output.forecasted_wearing_off,
           label='predicted', color='blue', marker='o')
  plt.legend()

  # Dashed horizontal line at 0.5
  # 0.5
  plt.axhline(0.5, linestyle='--', color='gray')

  # Dashed vertical lines on each hour
  #
  for i in forecasts_output.index:
    if pd.Timestamp(i).minute == 0:
      plt.axvline(i, linestyle='--', color='gray')
```

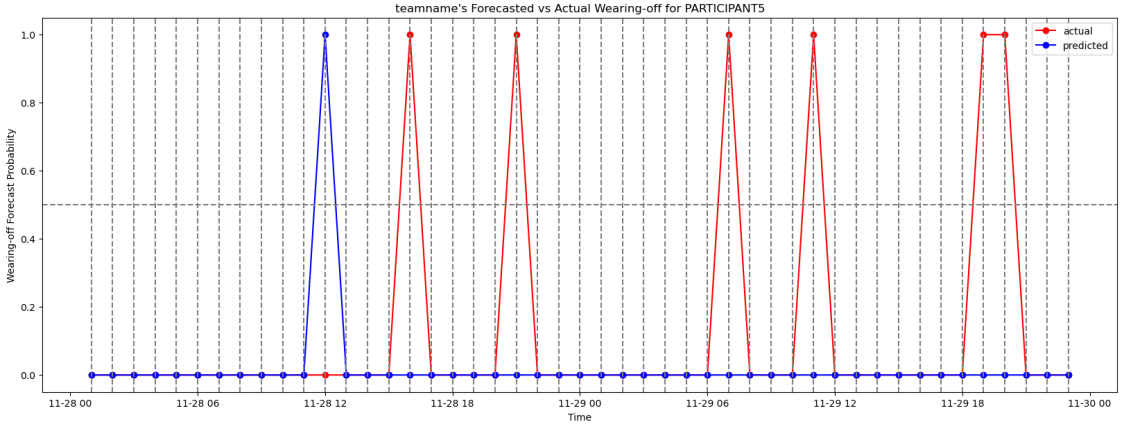
```
# Set y-axis label
# y
plt.ylabel('Wearing-off Forecast Probability')

# Set x-axis label
# x
plt.xlabel('Time')

# Set title
#
plt.title(f"{TEAMNAME}'s Forecasted vs Actual Wearing-off for {user.upper()}")

plt.show()
```

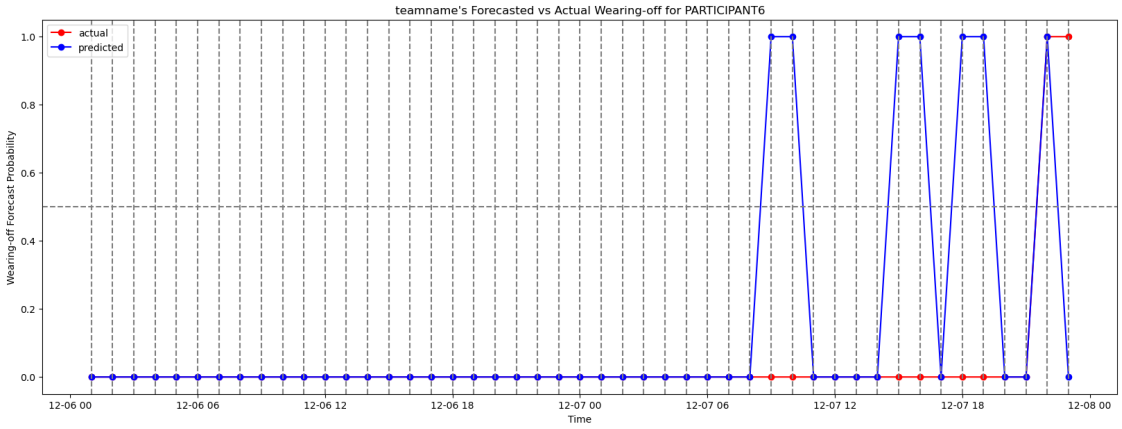


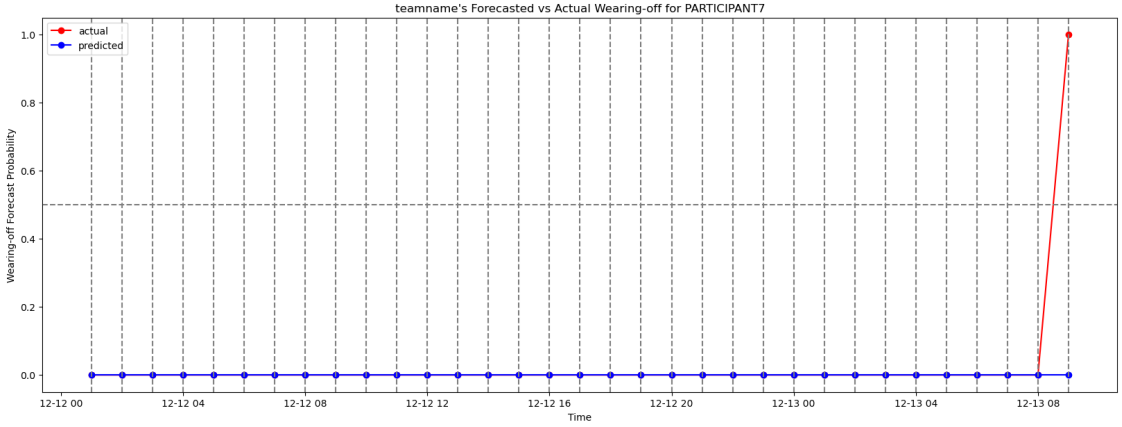


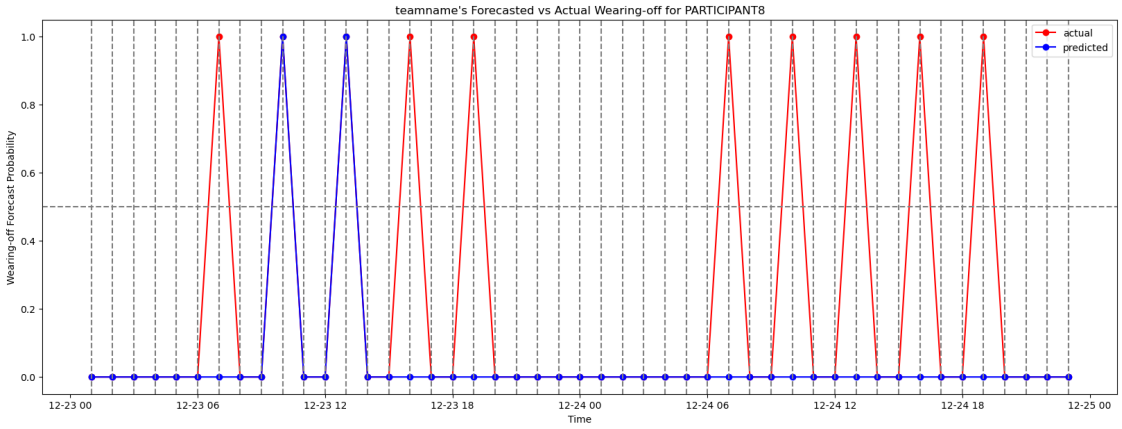

teamname's Forecasted vs Actual Wearing-off for PARTICIPANT3
actual
predicted
Wearing-off Forecast Probability
0.0
0.2
0.4
0.6
0.8
1.0
11-25 00
11-25 06
11-25 12
11-25 18
11-26 00
11-26 06
11-26 12
11-26 18
11-27 00
Time

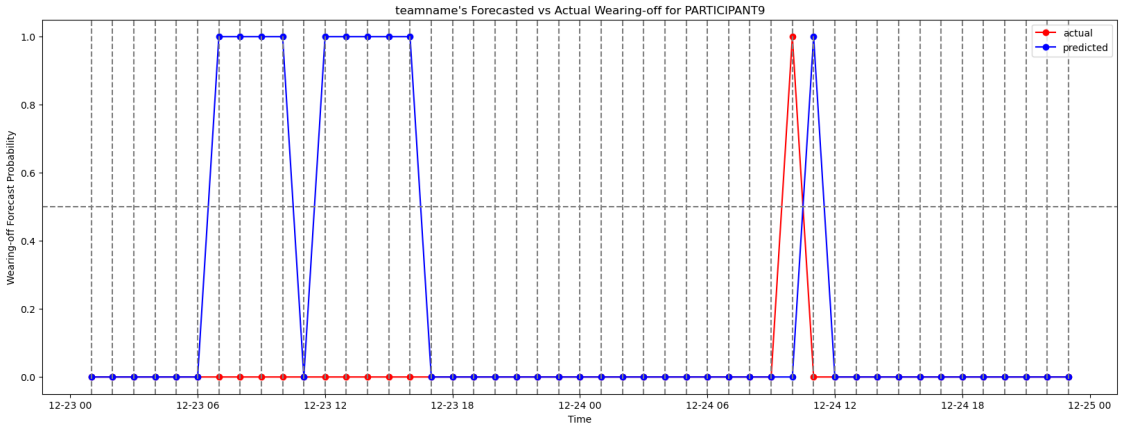

teamname's Forecasted vs Actual Wearing-off for PARTICIPANT4
actual
predicted
Wearing-off Forecast Probability
0.0
0.2
0.4
0.6
0.8
1.0
12-06 00
12-06 03
12-06 06
12-06 09
12-06 12
12-06 15
12-06 18
12-06 21
12-07 00
12-07 03
12-07 06
Time

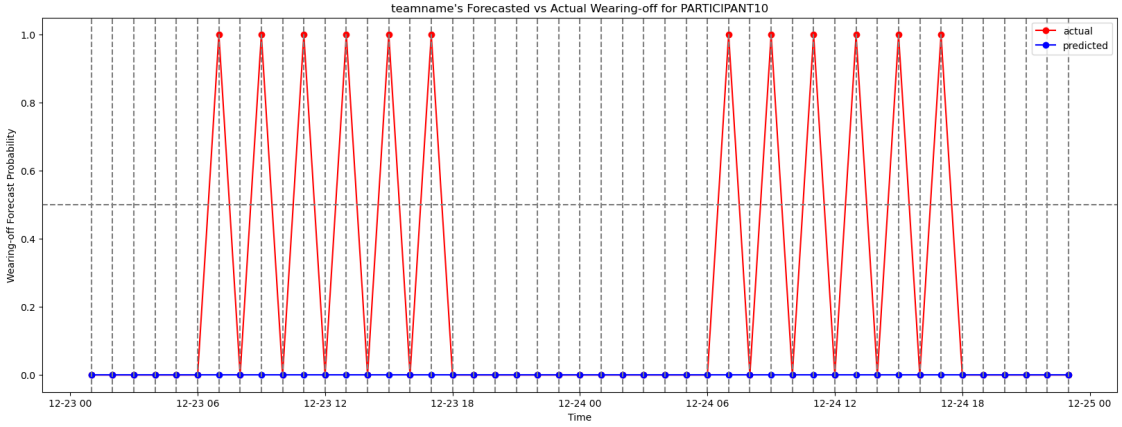

teamname's Forecasted vs Actual Wearing-off for PARTICIPANT5
actual
predicted
Wearing-off Forecast Probability
0.0
0.2
0.4
0.6
0.8
1.0
11-28 00
11-28 06
11-28 12
11-28 18
11-29 00
11-29 06
11-29 12
11-29 18
11-30 00
Time


teamname's Forecasted vs Actual Wearing-off for PARTICIPANT6
actual
predicted
Wearing-off Forecast Probability
1.0
0.8
0.6
0.4
0.2
0.0
12-06 00
12-06 06
12-06 12
12-06 18
12-07 00
12-07 06
12-07 12
12-07 18
12-08 00
Time


teamname's Forecasted vs Actual Wearing-off for PARTICIPANT7
actual
predicted
Wearing-off Forecast Probability
1.0
0.8
0.6
0.4
0.2
0.0
12-12 00
12-12 04
12-12 08
12-12 12
12-12 16
12-12 20
12-13 00
12-13 04
12-13 08
Time


teamname's Forecasted vs Actual Wearing-off for PARTICIPANT8
actual
predicted
Wearing-off Forecast Probability
1.0
0.8
0.6
0.4
0.2
0.0
12-23 00
12-23 06
12-23 12
12-23 18
12-24 00
12-24 06
12-24 12
12-24 18
12-25 00
Time

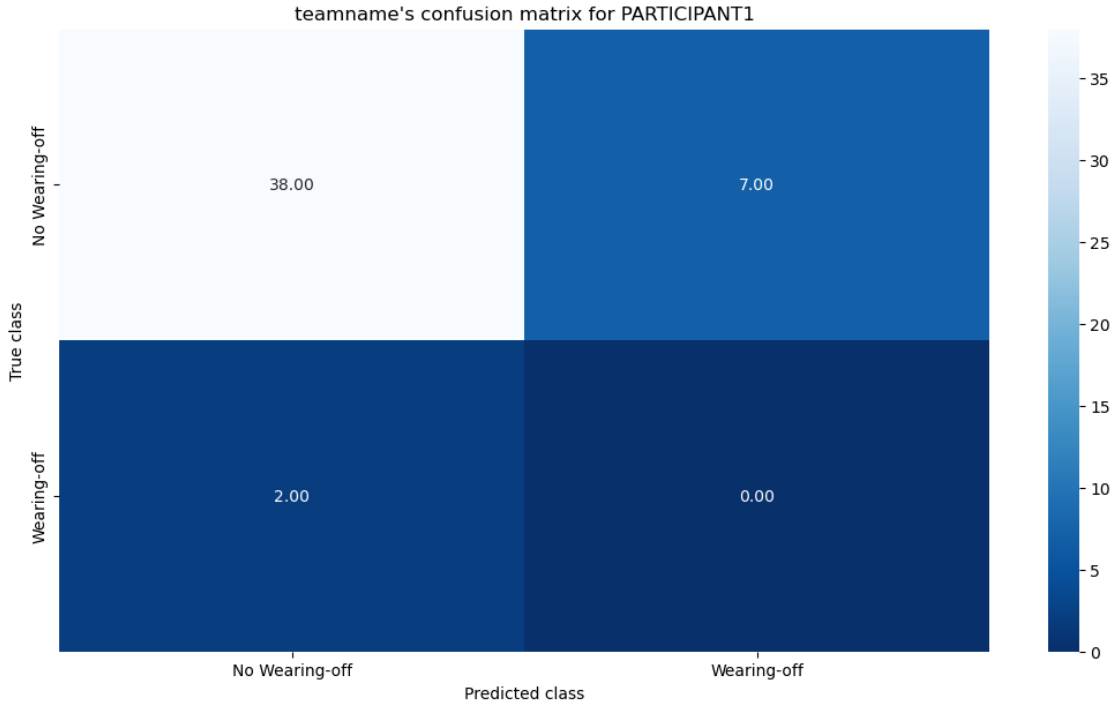


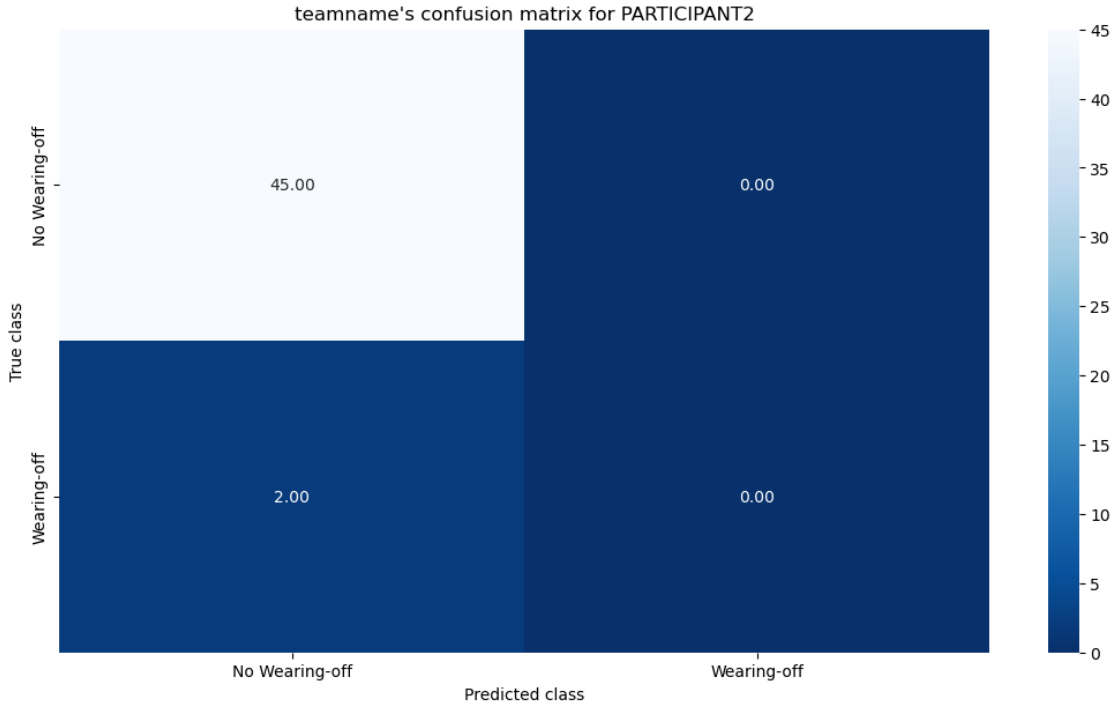


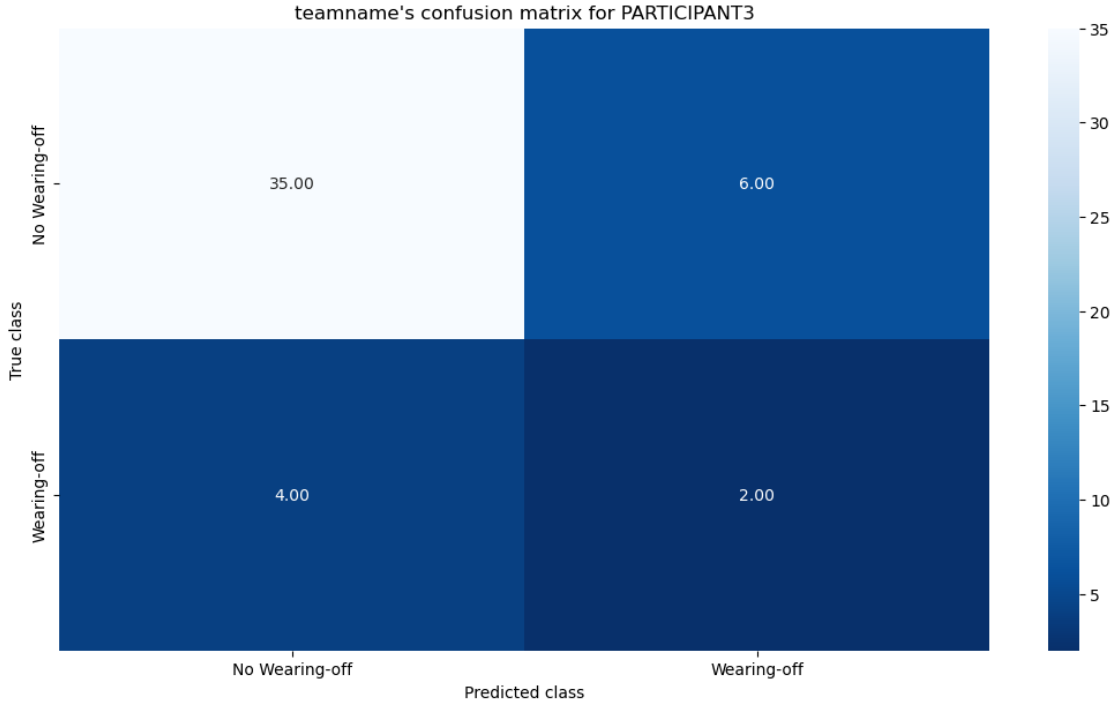
5.2 Plot Confusion Matrix

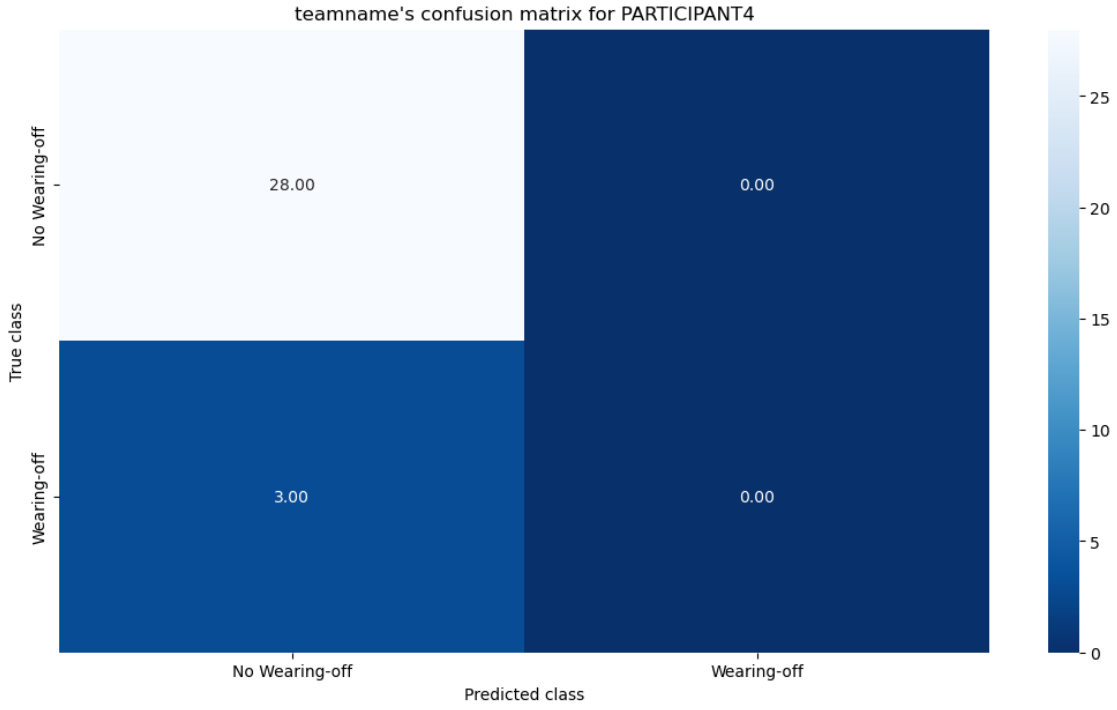
[ ]:
```
# Plot confusion matrix for each participant
#

# Set labels for confusion matrix
#
labels = ['No Wearing-off', 'Wearing-off']

for i in range(1, 11):
  user = f'participant{i}'
  forecasts_output = base_forecasts_output.query(
    f'patient_id == "{user}"')

  # Calculate confusion matrix
  #
  conf_matrix = confusion_matrix(forecasts_output.ground_truth,
```

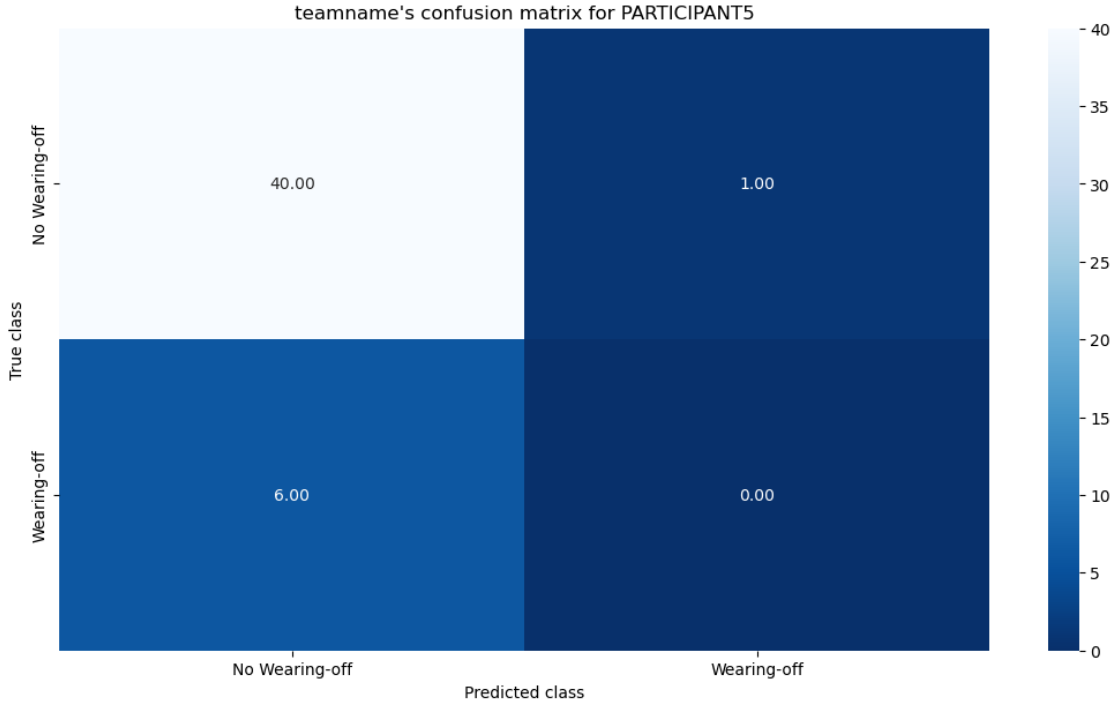
```
                                    forecasts_output.forecasted_wearing_off)
# Plot confusion matrix
#
plt.figure(figsize=FIGSIZE_CM)
sns.heatmap(conf_matrix,
            xticklabels=labels, yticklabels=labels,
            annot=True, fmt=".2f", cmap='Blues_r')

# Set y-axis label
# y
plt.ylabel('True class')

# Set x-axis label
# x
plt.xlabel('Predicted class')

# Set title
#
plt.title(f"{TEAMNAME}'s confusion matrix for {user.upper()}")

plt.show()
```

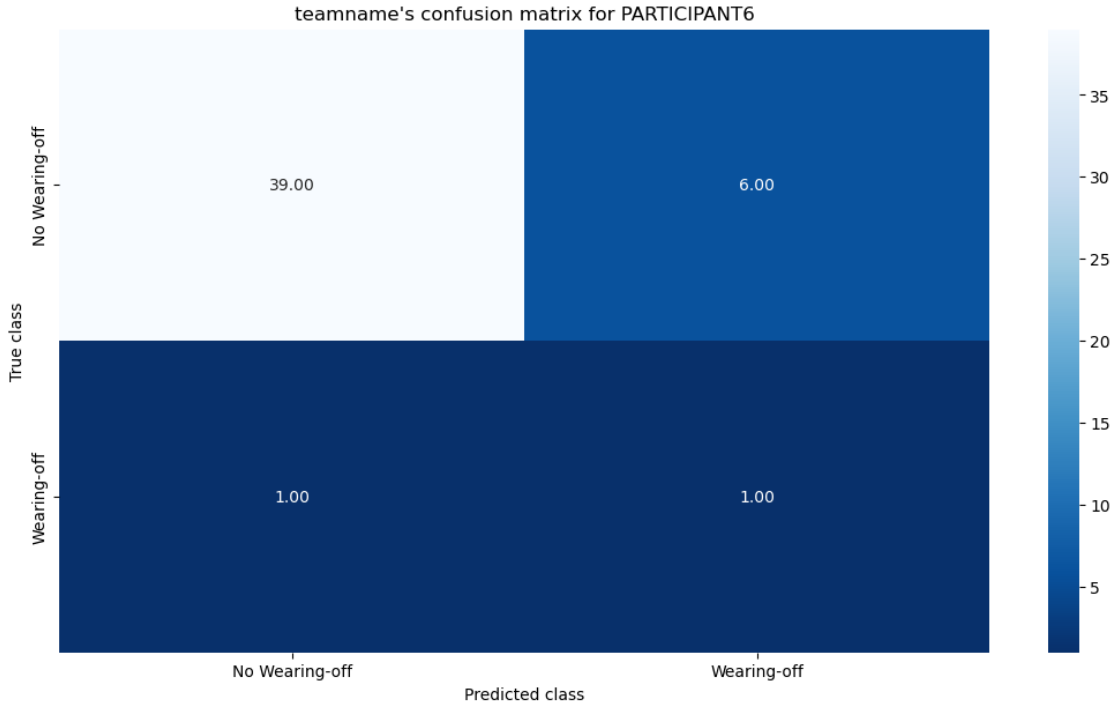


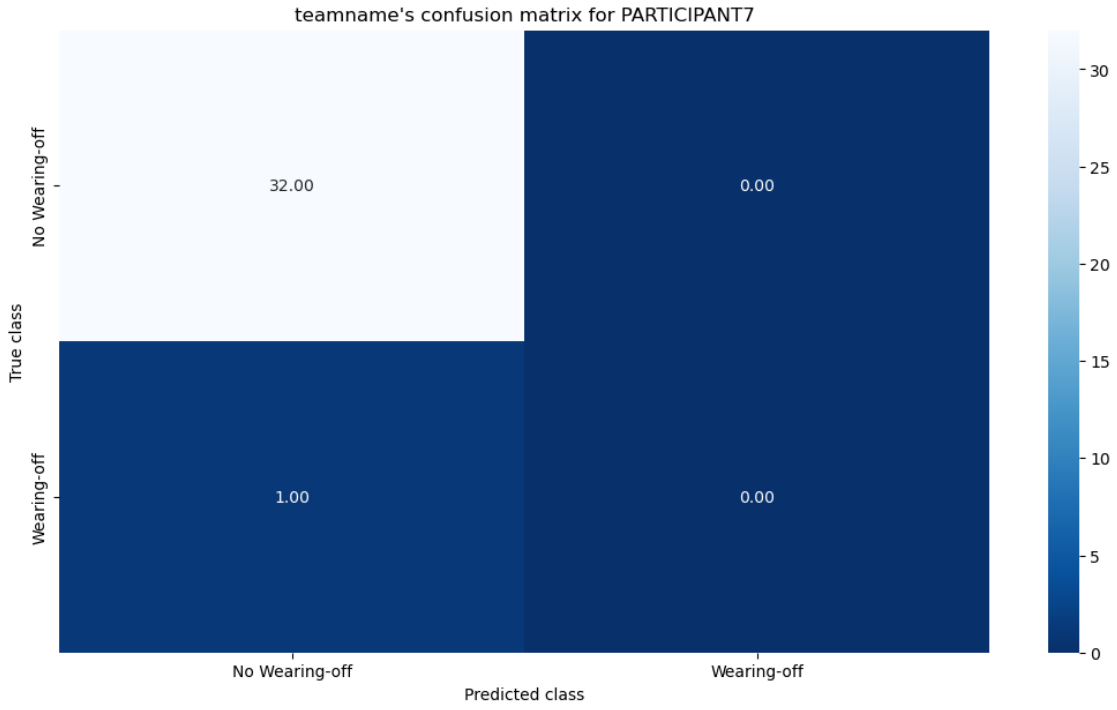

teamname's confusion matrix for PARTICIPANT2
45.00
0.00
2.00
0.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
45
40
35
30
25
20
15
10
5
0

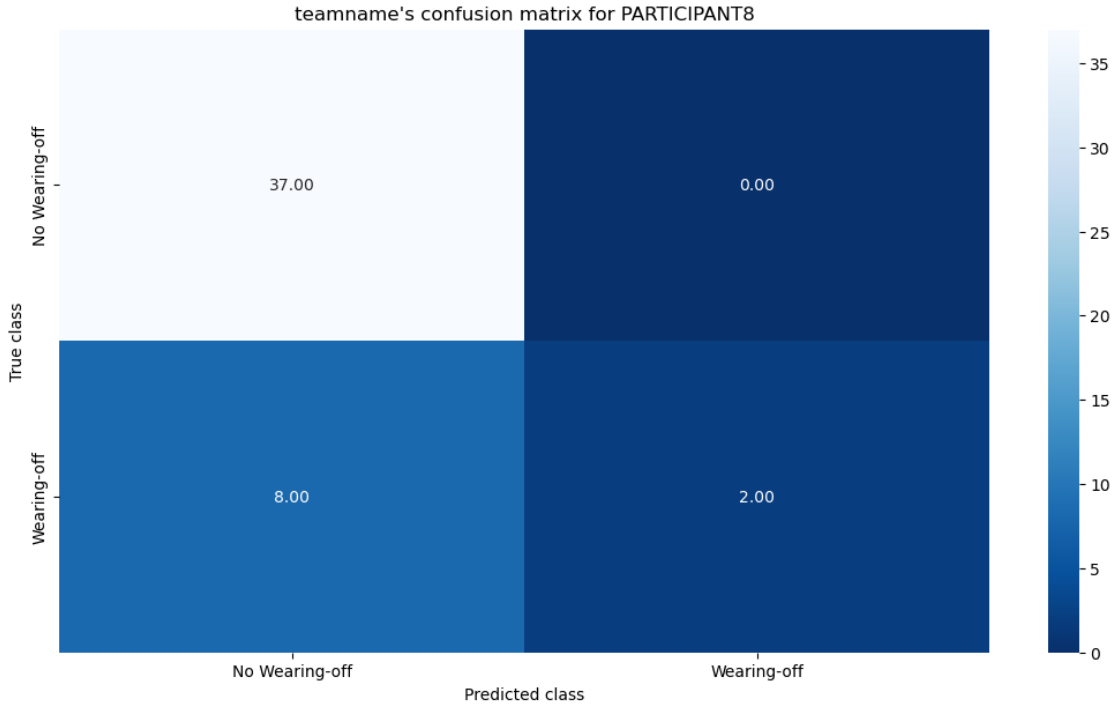

teamname's confusion matrix for PARTICIPANT3
35.00
6.00
4.00
2.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
35
30
25
20
15
10
5

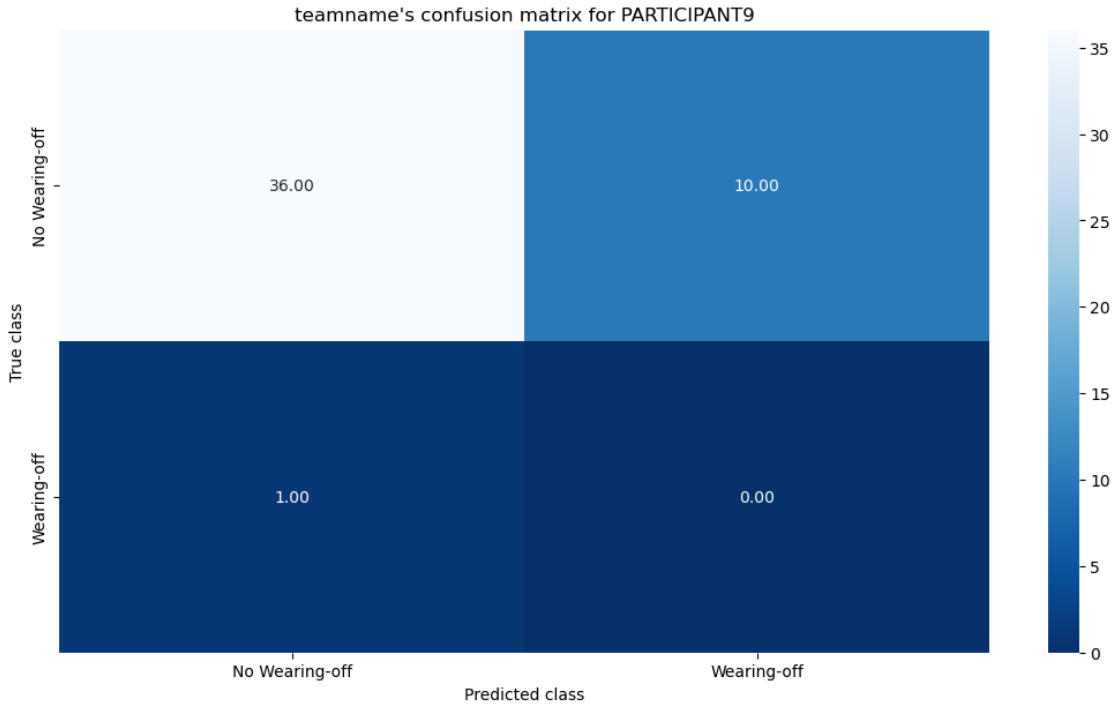

teamname's confusion matrix for PARTICIPANT4
28.00
0.00
3.00
0.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
25
20
15
10
5
0

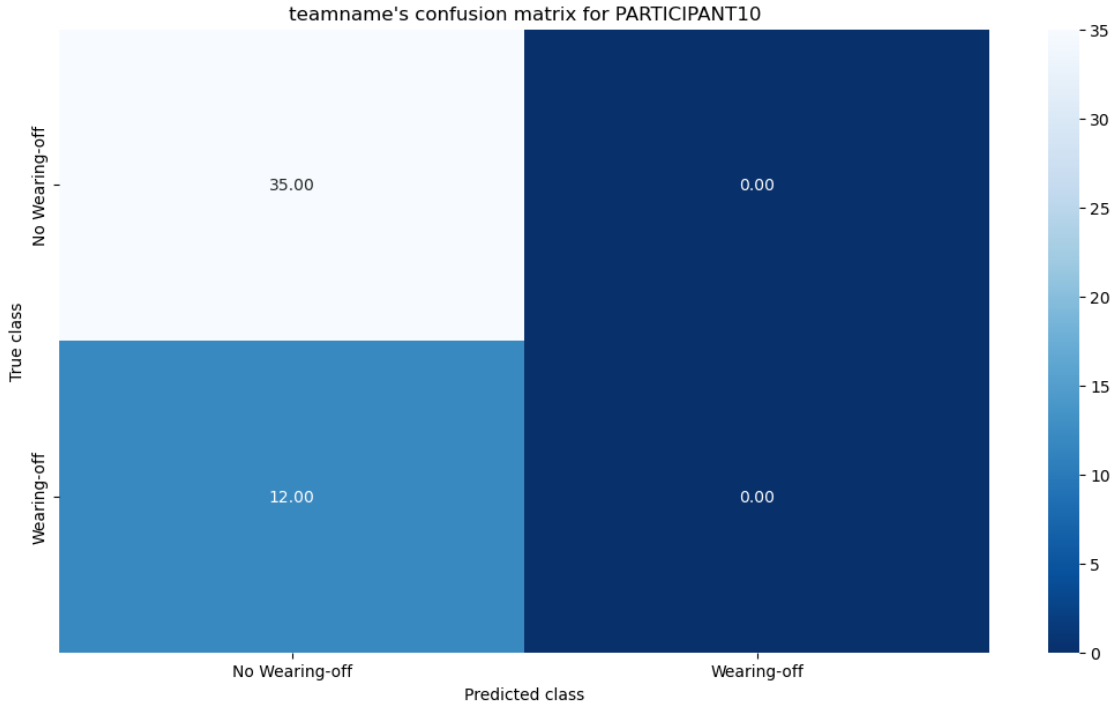

teamname's confusion matrix for PARTICIPANT5
40.00
1.00
6.00
0.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
40
35
30
25
20
15
10
5
0


teamname's confusion matrix for PARTICIPANT6
39.00
6.00
1.00
1.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
35
30
25
20
15
10
5


teamname's confusion matrix for PARTICIPANT7
32.00
0.00
1.00
0.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
30
25
20
15
10
5
0


teamname's confusion matrix for PARTICIPANT8
37.00
0.00
8.00
2.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
35
30
25
20
15
10
5
0


teamname's confusion matrix for PARTICIPANT9
36.00
10.00
1.00
0.00
No Wearing-off
Wearing-off
True class
No Wearing-off
Wearing-off
Predicted class
35
30
25
20
15
10
5
0



5.3 Calculate Metric Scores

[ ]:
```
# Generate metrics for each participant
#   Then compile all scores into 2 dataframes
#
#         2
final_model_metric_scores = pd.DataFrame()
final_model_classification_report = pd.DataFrame()

for i in range(1, 11):
  user = f'participant{i}'
  forecasts_output = base_forecasts_output.query(
    f'patient_id == "{user}"')

  # Calculate fpr, tpr, thresholds
  # fpr tpr
  fpr, tpr, thresholds = metrics.roc_curve(forecasts_output.sort_index().
 ↪ground_truth,
                                           forecasts_output.sort_index().
 ↪forecasted_wearing_off)

  ######################
  # Evaluate predictions with f1 score, recall, precision, accuracy, auc-roc,␣
 ↪auc-prc
```

```
# f1         auc-roc auc-prc
model_metric_scores = pd.DataFrame(
  [
    metrics.f1_score(
      forecasts_output.ground_truth,
      forecasts_output.forecasted_wearing_off),
    metrics.recall_score(
      forecasts_output.ground_truth,
      forecasts_output.forecasted_wearing_off),
    metrics.precision_score(
      forecasts_output.ground_truth,
      forecasts_output.forecasted_wearing_off),
    metrics.accuracy_score(
      forecasts_output.ground_truth,
      forecasts_output.forecasted_wearing_off),
    metrics.auc(fpr, tpr),
    metrics.average_precision_score(
      forecasts_output.sort_index().ground_truth,
      forecasts_output.sort_index().forecasted_wearing_off)
  ],
  index=['f1 score', 'recall', 'precision', 'accuracy', 'auc-roc', 'auc-prc'],
  columns=['metrics']
).T.round(3).assign(teamname=TEAMNAME, participant=user)
# Set index to teammame
#
model_metric_scores.set_index(['teamname'], inplace=True)

######################
# Generate classification report
#
model_classification_report = pd.DataFrame(
  classification_report(
    forecasts_output.ground_truth,
    forecasts_output.forecasted_wearing_off,
    output_dict=True
  )
).T.round(3).assign(teamname=TEAMNAME, participant=user)
# Set index's name to 'classification report'
#
model_classification_report.index.name = 'classification report'

# Remove row that has 'accuracy' as index (not needed)
#     accuracy       ( )
model_classification_report = model_classification_report.drop(
  ['accuracy'], axis=0)

# Set index to teammame, participant, classification report
```

```
  #
  model_classification_report = model_classification_report.reset_index()
  model_classification_report.set_index(
      ['teamname', 'participant', 'classification report'], inplace=True)

  ######################
  # Concatenate scores into final dataframes
  #
  model_metric_scores.reset_index(inplace=True)
  model_classification_report.reset_index(inplace=True)

  final_model_metric_scores = pd.concat([
    final_model_metric_scores,
    model_metric_scores
  ])

  final_model_classification_report = pd.concat([
    final_model_classification_report,
    model_classification_report
  ])

  # print(user)
  # display(model_metric_scores)
  # display(model_classification_report)
```

[ ]:
```
# Metric scores for each participant for positive class
#
final_model_metric_scores
```

[ ]:

	teamname	f1 score	recall	precision	accuracy	auc-roc	auc-prc	\
0	teamname	0.000	0.000	0.000	0.809	0.422	0.043	\
0	teamname	0.000	0.000	0.000	0.957	0.500	0.043	
0	teamname	0.286	0.333	0.250	0.787	0.593	0.168	
0	teamname	0.000	0.000	0.000	0.903	0.500	0.097	
0	teamname	0.000	0.000	0.000	0.851	0.488	0.128	
0	teamname	0.222	0.500	0.143	0.851	0.683	0.093	
0	teamname	0.000	0.000	0.000	0.970	0.500	0.030	
0	teamname	0.333	0.200	1.000	0.830	0.600	0.370	
0	teamname	0.000	0.000	0.000	0.766	0.391	0.021	
0	teamname	0.000	0.000	0.000	0.745	0.500	0.255	

	participant
0	participant1
0	participant2
0	participant3
0	participant4
0	participant5

```
0   participant6
0   participant7
0   participant8
0   participant9
0  participant10
```

[ ]:
```
# Classification report for each participant for weighted avg
#
final_model_classification_report.loc[
  final_model_classification_report['classification report'] == 'macro avg'
]
```

[ ]:
```
   teamname    participant classification report  precision  recall  f1-score  \
2  teamname   participant1              macro avg      0.475   0.422     0.447
2  teamname   participant2              macro avg      0.479   0.500     0.489
2  teamname   participant3              macro avg      0.574   0.593     0.580
2  teamname   participant4              macro avg      0.452   0.500     0.475
2  teamname   participant5              macro avg      0.435   0.488     0.460
2  teamname   participant6              macro avg      0.559   0.683     0.570
2  teamname   participant7              macro avg      0.485   0.500     0.492
2  teamname   participant8              macro avg      0.911   0.600     0.618
2  teamname   participant9              macro avg      0.486   0.391     0.434
2  teamname  participant10              macro avg      0.372   0.500     0.427

   support
2     47.0
2     47.0
2     47.0
2     31.0
2     47.0
2     47.0
2     33.0
2     47.0
2     47.0
2     47.0
```