driving_behavior_brf_v2

September 1, 2022

0.1 Binary Random Forest / KNN

[ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
df_train = pd.read_csv("../data_mod/train_motion_data.csv")
df_test = pd.read_csv("../data_mod/test_motion_data.csv")

df_train
```

[ ]:
```
          AccX      AccY     GyroZ   Class  DiffAccX  DiffAccY      VelX  \
0     0.000000  0.000000  0.101938  NORMAL  0.000000  0.000000  0.000000
1    -1.624864 -1.082492  0.135536  NORMAL -1.624864 -1.082492 -0.812432
2    -0.594660 -0.122410  0.087888  NORMAL  1.030204  0.960082 -0.297330
3     0.738478 -0.228456  0.054902  NORMAL  1.333138 -0.106046  0.369239
4     0.101741  0.777568  0.054902  NORMAL -0.636737  1.006023  0.050871
…          …         …         …       …         …         …         …
3639  0.915688 -2.017489 -1.236468    SLOW  2.374675 -1.824629  0.457844
3640 -1.934203  0.914925 -0.477162    SLOW -2.849891  2.932414 -0.967102
3641 -0.222845  0.747304  0.054291    SLOW  1.711359 -0.167621 -0.111422
3642 -0.349423  0.067261 -0.004963    SLOW -0.126579 -0.680043 -0.174712
3643 -0.402428  0.406218  0.001145    SLOW -0.053005  0.338957 -0.201214

          VelY
0     0.000000
1    -0.541246
2    -0.061205
3    -0.114228
4     0.388784
…          …
3639 -1.008745
3640  0.457462
3641  0.373652
3642  0.033630
3643  0.203109
```

```
[3644 rows x 8 columns]
```

```
[ ]: df_train.isna().sum()
```

```
[ ]: AccX        0
     AccY        0
     GyroZ       0
     Class       0
     DiffAccX    0
     DiffAccY    0
     VelX        0
     VelY        0
     dtype: int64
```

0.1.1 Change categories to numbers

```
[ ]: df_train = df_train.replace(
         {"Class": {"NORMAL": 0, "SLOW": 1, "AGGRESSIVE": 2}})
     df_test = df_test.replace(
         {"Class": {"NORMAL": 0, "SLOW": 1, "AGGRESSIVE": 2}})
     df_train
```

```
[ ]:          AccX      AccY     GyroZ  Class  DiffAccX  DiffAccY      VelX  \
     0     0.000000  0.000000  0.101938      0  0.000000  0.000000  0.000000
     1    -1.624864 -1.082492  0.135536      0 -1.624864 -1.082492 -0.812432
     2    -0.594660 -0.122410  0.087888      0  1.030204  0.960082 -0.297330
     3     0.738478 -0.228456  0.054902      0  1.333138 -0.106046  0.369239
     4     0.101741  0.777568  0.054902      0 -0.636737  1.006023  0.050871
     …          …         …         …      …        …         …
     3639  0.915688 -2.017489 -1.236468      1  2.374675 -1.824629  0.457844
     3640 -1.934203  0.914925 -0.477162      1 -2.849891  2.932414 -0.967102
     3641 -0.222845  0.747304  0.054291      1  1.711359 -0.167621 -0.111422
     3642 -0.349423  0.067261 -0.004963      1 -0.126579 -0.680043 -0.174712
     3643 -0.402428  0.406218  0.001145      1 -0.053005  0.338957 -0.201214

               VelY
     0     0.000000
     1    -0.541246
     2    -0.061205
     3    -0.114228
     4     0.388784
     …          …
     3639 -1.008745
     3640  0.457462
     3641  0.373652
     3642  0.033630
     3643  0.203109
```

```
[3644 rows x 8 columns]
```

0.1.2 Remove unnecessary columns

```
[ ]: # df_train.drop(['AccZ', 'GyroX', 'GyroY', 'Timestamp'], axis=1, inplace=True)
     # df_test.drop(['AccZ', 'GyroX', 'GyroY', 'Timestamp'], axis=1, inplace=True)

     # df_train
```

0.1.3 Only select normal and aggressive values

```
[ ]: df_train = df_train.loc[df_train['Class'] != 1]
     df_test = df_test.loc[df_test['Class'] != 1]

     df_train
```

```
[ ]:          AccX      AccY     GyroZ  Class  DiffAccX  DiffAccY      VelX  \
     0     0.000000  0.000000  0.101938      0  0.000000  0.000000  0.000000
     1    -1.624864 -1.082492  0.135536      0 -1.624864 -1.082492 -0.812432
     2    -0.594660 -0.122410  0.087888      0  1.030204  0.960082 -0.297330
     3     0.738478 -0.228456  0.054902      0  1.333138 -0.106046  0.369239
     4     0.101741  0.777568  0.054902      0 -0.636737  1.006023  0.050871
     …          …         …         …     …         …         …
     2308  0.538870 -1.645984  0.662712      2  0.200934 -0.962974  0.269435
     2309  1.678918 -1.392127 -0.168675      2  1.140048  0.253856  0.839459
     2310  0.323433  0.589311  0.639500      2 -1.355486  1.981439  0.161716
     2311  2.497311 -0.606175 -0.240757      2  2.173878 -1.195487  1.248655
     2312  0.482297 -0.090277 -0.383700      2 -2.015014  0.515898  0.241148

              VelY
     0     0.000000
     1    -0.541246
     2    -0.061205
     3    -0.114228
     4     0.388784
     …          …
     2308 -0.822992
     2309 -0.696064
     2310  0.294656
     2311 -0.303088
     2312 -0.045139

     [2313 rows x 8 columns]
```

```
[ ]: X_train = df_train.drop(columns=["Class"])
     y_train = df_train['Class']

     X_test = df_test.drop(columns=["Class"])
     y_test = df_test['Class']
```

0.1.4 Normalize data

```
[ ]: X_train = (X_train - X_train.mean()) / X_train.std() * 100
     X_test = (X_test - X_test.mean()) / X_test.std() * 100

     X_train
```

```
[ ]:            AccX        AccY       GyroZ    DiffAccX    DiffAccY        VelX  \
     0      -3.509345    9.776257   74.896498   -0.018756    0.003491   -3.509345
     1    -157.992905  -99.985349  102.351035 -146.171580  -96.829548 -157.992905
     2     -60.046498   -2.635757   63.415515   92.645759   85.886499  -60.046498
     3      66.701299  -13.388488   36.460154  119.893994   -9.482701   66.701299
     4       6.163664   88.619412   36.460154  -57.291816   89.996116    6.163664
     …              …           …           …           …           …           …
     2308   47.723614 -157.121833  533.137616   18.054827  -86.138250   47.723614
     2309  156.113434 -131.381523 -146.237282  102.525996   22.711910  156.113434
     2310   27.240939   69.530762  514.169013 -121.941626  177.250781   27.240939
     2311  233.921883  -51.688213 -205.139734  195.516638 -106.937380  233.921883
     2312   42.344893    0.622405 -321.946292 -181.264697   46.152563   42.344893

                 VelY
     0       9.776257
     1     -99.985349
     2      -2.635757
     3     -13.388488
     4      88.619412
     …              …
     2308 -157.121833
     2309 -131.381523
     2310   69.530762
     2311  -51.688213
     2312    0.622405

     [2313 rows x 7 columns]
```

0.2 Train model

0.2.1 Random Forest

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

[ ]:
```
rfc = RandomForestClassifier(n_estimators=30, max_depth=15, random_state=5,␣
 ↪criterion="entropy")
rfc.fit(X_train, y_train)
```

[ ]:
```
RandomForestClassifier(criterion='entropy', max_depth=15, n_estimators=30,
                       random_state=5)
```

[ ]:
```
rfc.score(X_train, y_train)
```

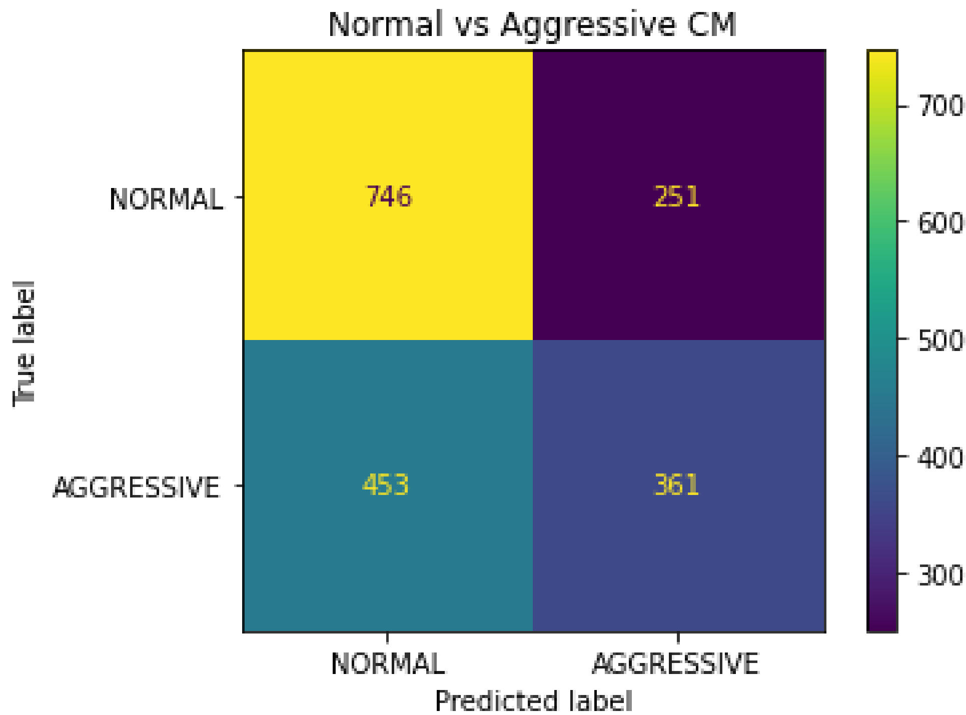
[ ]:
```
0.8832684824902723
```

[ ]:
```
rfc.score(X_test, y_test)
```

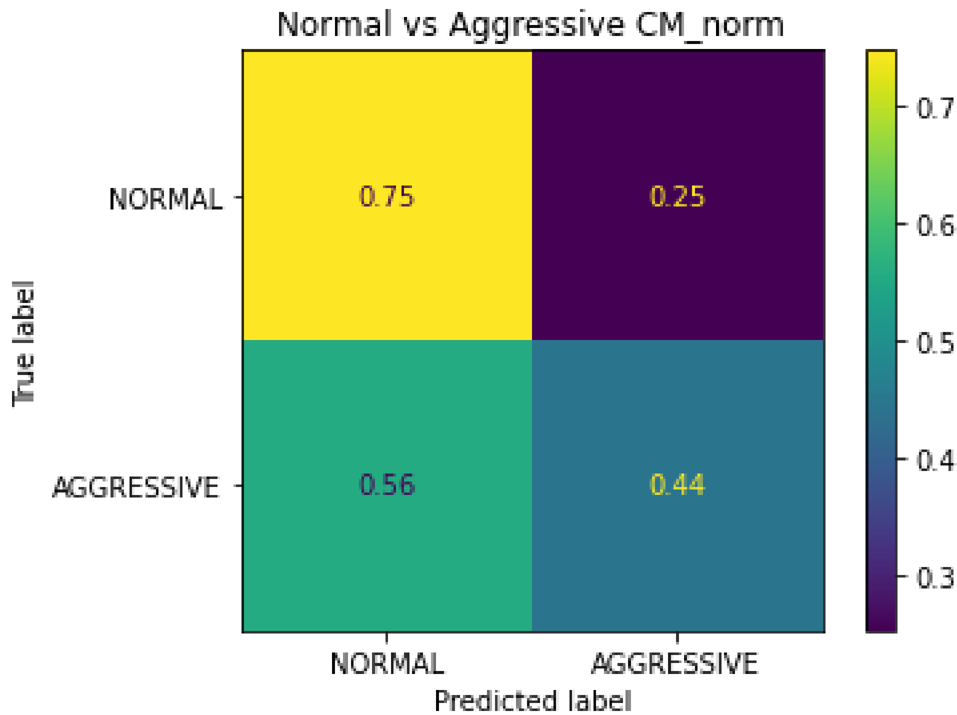
[ ]:
```
0.6112644947542794
```

[ ]:
```
classes=['NORMAL', 'AGGRESSIVE']
```

[ ]:
```
y_pred = rfc.predict(X_test)

CM = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=classes).plot()
plt.title('Normal vs Aggressive CM')
plt.show()
```



[ ]:
```
CM_norm = confusion_matrix(y_test, y_pred, normalize="true")

ConfusionMatrixDisplay(confusion_matrix=CM_norm, display_labels=classes).plot()
plt.title('Normal vs Aggressive CM_norm')
plt.show()
```




```
[ ]: rfc.score(X_test, y_test)
```

```
[ ]: 0.6112644947542794
```

```
[ ]: rfc_imp = pd.DataFrame(rfc.feature_importances_, columns=['importance'])
```

```
[ ]: rfc_imp['importance'] = rfc_imp['importance'] * 100
     rfc_imp = rfc_imp.set_index(X_train.columns)
     rfc_imp
```

```
[ ]:           importance
     AccX       14.332129
     AccY       13.421542
     GyroZ      13.370358
     DiffAccX   13.929883
     DiffAccY   15.421162
     VelX       14.128181
     VelY       15.396745
```

```
[ ]: rfc_imp.sort_values(by='importance', ascending=False)
```

```
[ ]:           importance
     DiffAccY   15.421162
```

```
VelY        15.396745
AccX        14.332129
VelX        14.128181
DiffAccX    13.929883
AccY        13.421542
GyroZ       13.370358
```

0.2.2 Train model with RandomSearchCV

[ ]:
```
n_estimators = np.arange(2, 200, 2)

max_features = ['sqrt', None]

max_depth = [int(x) for x in np.linspace(5, 20, num = 20)]
max_depth.append(None)

min_samples_split = np.arange(2, 10)

min_samples_leaf = np.arange(1, 4)

bootstrap = [True, False]
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

[ ]:
```
weights = {0:1, 2:1.4}
random_forest = RandomForestClassifier(random_state=0, criterion="entropy",␣
 ↪min_impurity_decrease=0, class_weight=weights)

random_gscv = RandomizedSearchCV(random_forest, random_grid, n_iter=1000, cv=5,␣
 ↪verbose=10, n_jobs=10, random_state=0)
random_gscv.fit(X_train, y_train)
```

[ ]:
```
random_gscv.best_params_
```

[ ]:
```
{'n_estimators': 48,
 'min_samples_split': 6,
 'min_samples_leaf': 3,
 'max_features': None,
 'max_depth': 9,
 'bootstrap': True}
```

[ ]:
```
random_gscv.best_score_
```

[ ]: 0.6212541957682346

[ ]:
```
random_gscv.score(X_train, y_train)
```

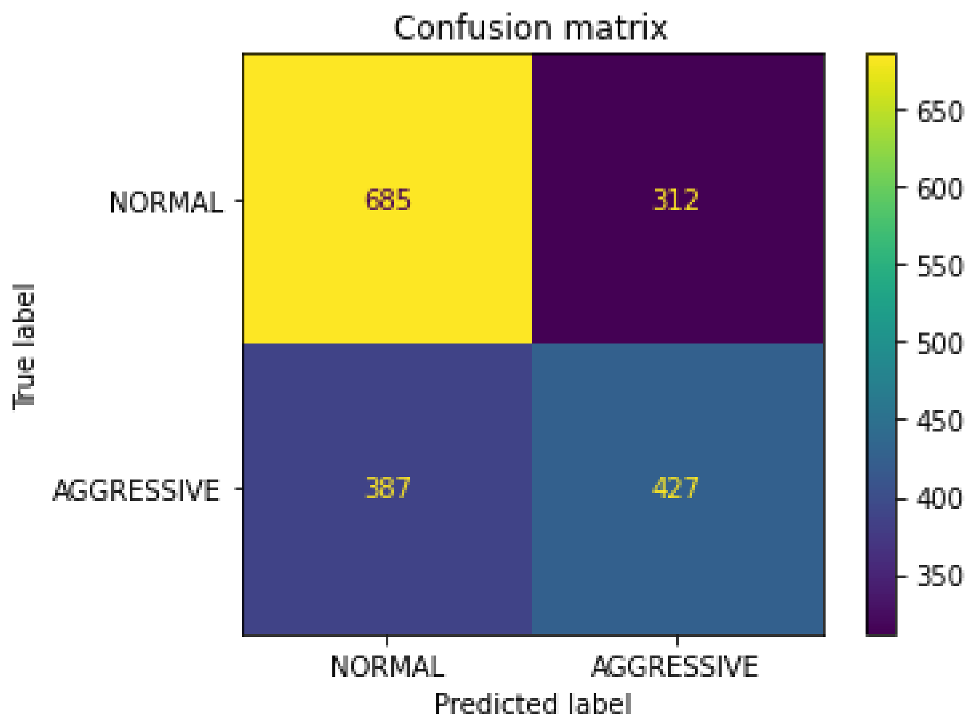
[ ]: 0.7842628620838737

[ ]:
```
random_gscv.score(X_test, y_test)
```

[ ]: 0.6140254003313087

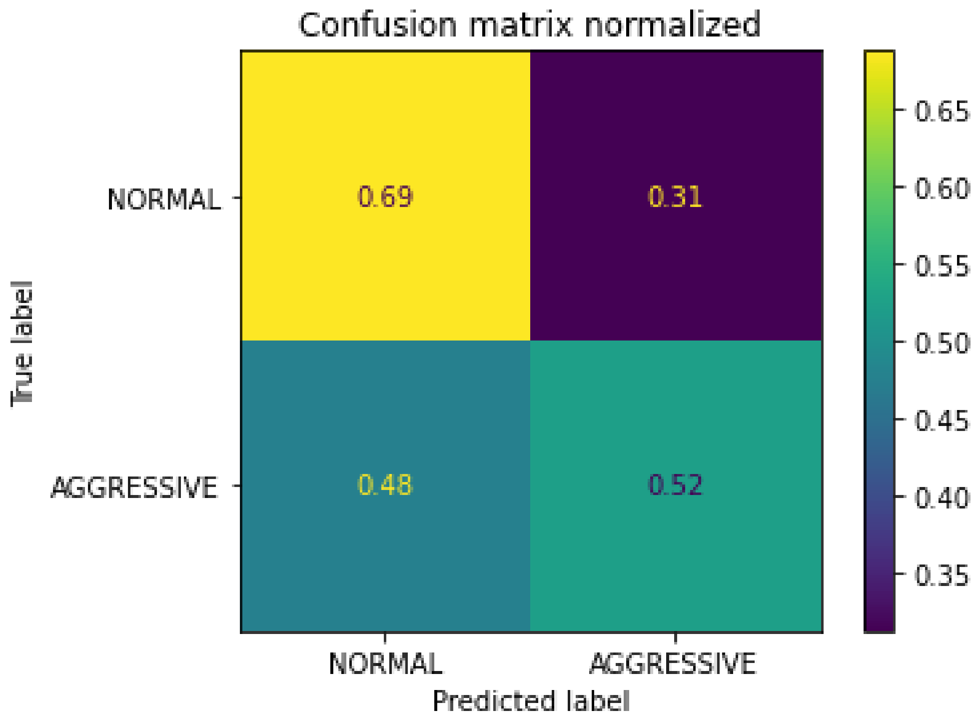
[ ]:
```
classes = ["NORMAL", "AGGRESSIVE"]
```

[ ]:
```
y_pred = random_gscv.predict(X_test)

CM = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=classes).plot()
plt.title('Confusion matrix')
plt.show()
```



[ ]:
```
CM_norm = confusion_matrix(y_test, y_pred, normalize="true")

ConfusionMatrixDisplay(confusion_matrix=CM_norm, display_labels=classes).plot()
plt.title('Confusion matrix normalized')
```

```
plt.show()
```



Evaluate improvment

```
[ ]: def evaluate(model, test_features, test_labels):
         accuracy = model.score(test_features, test_labels)
         print('Model Performance')
         print('Accuracy = {:0.3f}%.'.format(accuracy))

         return accuracy

     base_model = RandomForestClassifier(n_estimators = 10, random_state=0,␣
      ↪criterion="entropy", min_impurity_decrease=0, class_weight=weights)
     base_model.fit(X_train, y_train)
     base_accuracy = evaluate(base_model, X_test, y_test)

     best_random = random_gscv.best_estimator_
     random_accuracy = evaluate(best_random, X_test, y_test)

     print(f'Improvement of {100 * (random_accuracy - base_accuracy) / base_accuracy:
      ↪.3f}%.')
```

```
Model Performance
Accuracy = 0.592%.
```

```
Model Performance
Accuracy = 0.614%.
Improvement of 3.731%.
```

0.2.3 KNN

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

[ ]:
```
Kneigh = KNeighborsClassifier(weights="uniform")

param_grid = {'n_neighbors': np.arange(1, 100), 'leaf_size': np.arange(20, 40)}

knn_gscv = GridSearchCV(Kneigh, param_grid, cv=5, verbose=10, n_jobs=10)
knn_gscv.fit(X_train, y_train)
```

[ ]:
```
best_params = knn_gscv.best_params_
best_params
```

[ ]:
```
{'leaf_size': 20, 'n_neighbors': 51}
```

[ ]:
```
knn_gscv.best_score_
```

[ ]:
```
0.6173749216945761
```

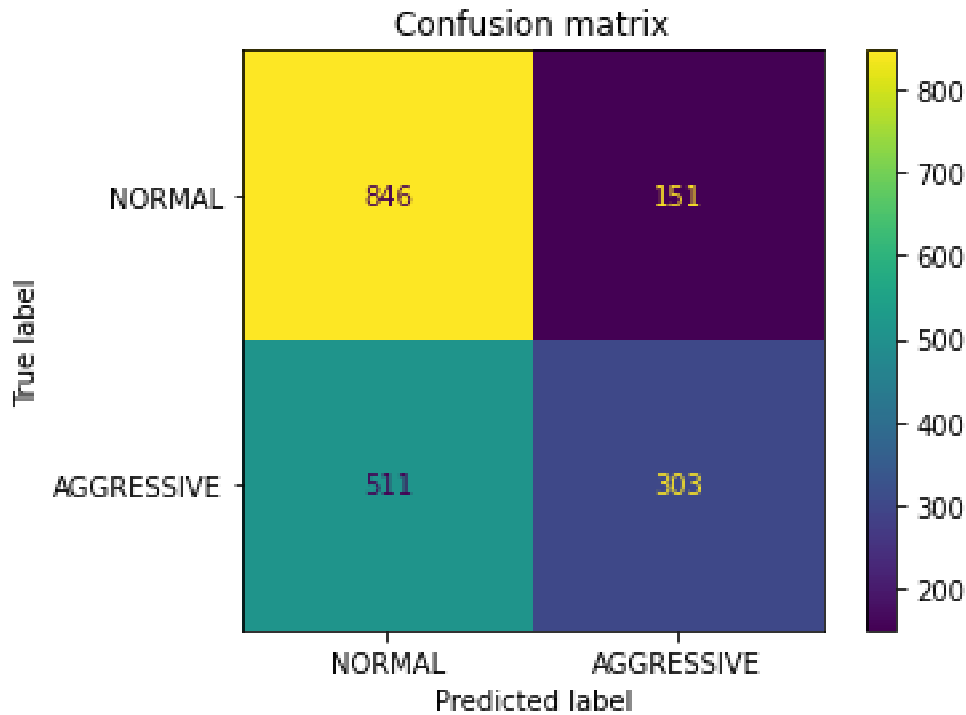
[ ]:
```
knn_gscv.score(X_train, y_train)
```

[ ]:
```
0.6372676178123649
```

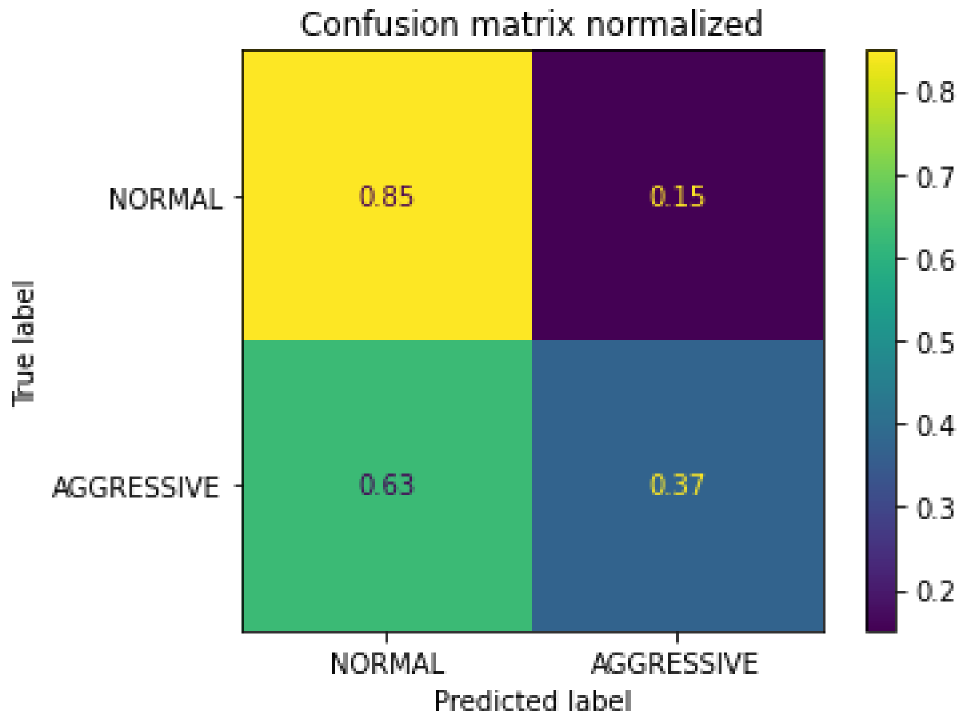
[ ]:
```
knn_gscv.score(X_test, y_test)
```

[ ]:
```
0.6344561016013253
```

[ ]:
```
y_pred = knn_gscv.predict(X_test)

CM = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=classes).plot()
plt.title('Confusion matrix')
plt.show()
```



[ ]:
```
CM_norm = confusion_matrix(y_test, y_pred, normalize="true")

ConfusionMatrixDisplay(confusion_matrix=CM_norm, display_labels=classes).plot()
plt.title('Confusion matrix normalized')
plt.show()
```



Knn with Bagging classifier

[ ]:
```
from sklearn.ensemble import BaggingClassifier

knn_bagging = BaggingClassifier(KNeighborsClassifier(**knn_gscv.best_params_),␣
↪max_samples=0.4, max_features=0.5, random_state=0)
knn_bagging.fit(X_train, y_train)
```

[ ]:
```
BaggingClassifier(base_estimator=KNeighborsClassifier(leaf_size=20,
                                                      n_neighbors=51),
                  max_features=0.5, max_samples=0.4, random_state=0)
```

[ ]:
```
knn_bagging.score(X_train, y_train)
```

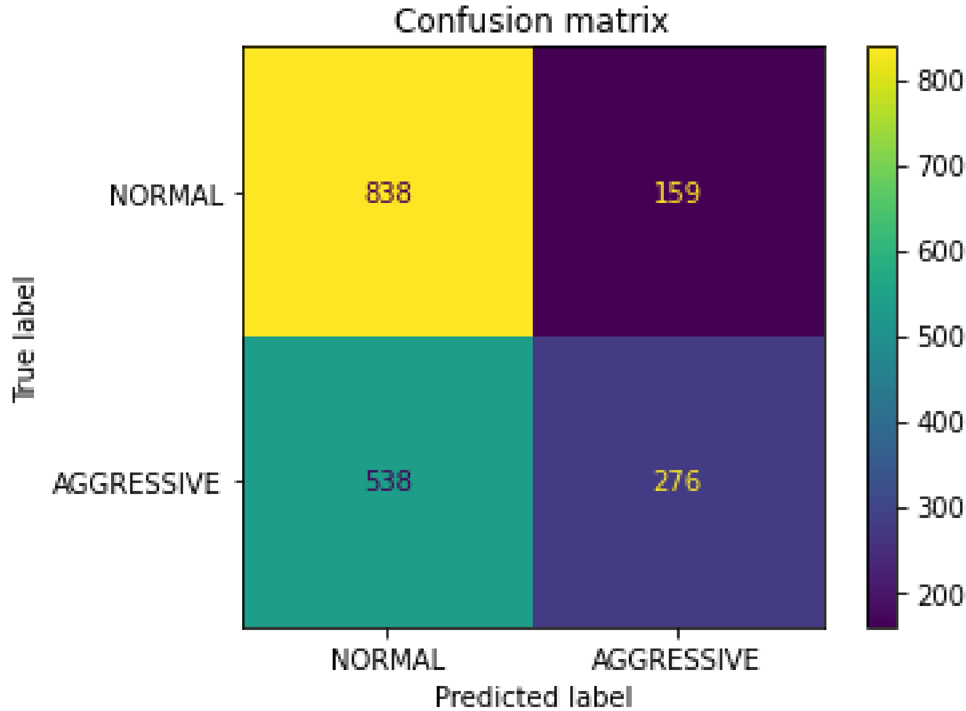
[ ]:
```
0.6160830090791181
```

[ ]:
```
knn_bagging.score(X_test, y_test)
```

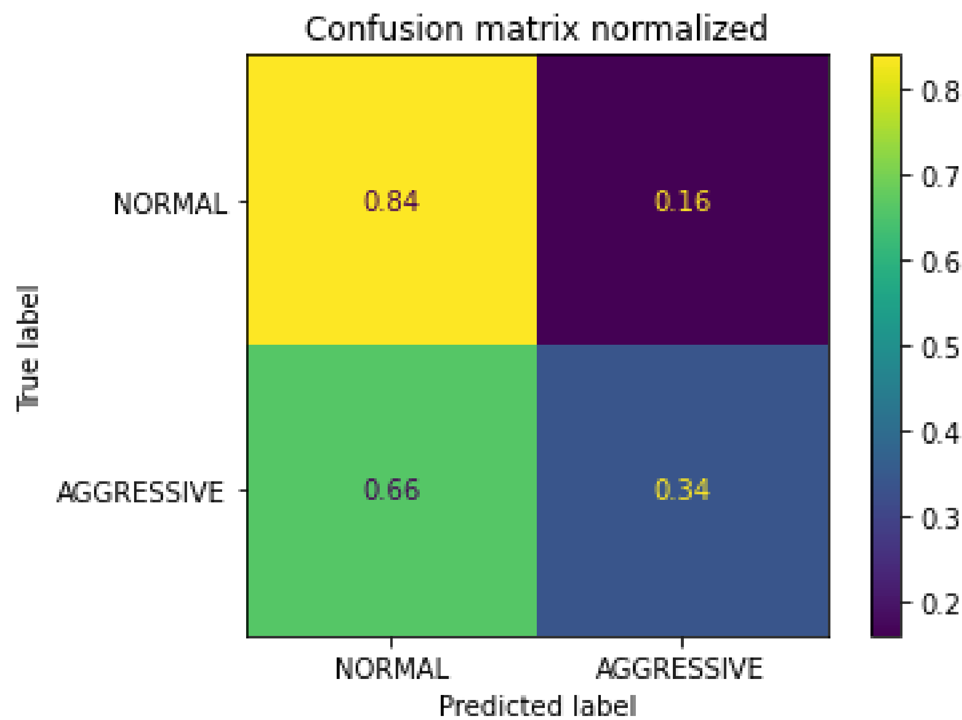
[ ]:
```
0.6151297625621204
```

[ ]:
```
y_pred = knn_bagging.predict(X_test)

CM = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=CM, display_labels=classes).plot()
```

```
plt.title('Confusion matrix')
plt.show()
```



[ ]:
```
CM_norm = confusion_matrix(y_test, y_pred, normalize="true")

ConfusionMatrixDisplay(confusion_matrix=CM_norm, display_labels=classes).plot()
plt.title('Confusion matrix normalized')
plt.show()
```



[ ]:
```
def evaluate(model, test_features, test_labels):
    accuracy = model.score(test_features, test_labels)
    print('Model Performance')
    print('Accuracy = {:0.3f}%.'.format(accuracy))

    return accuracy

bagging_accuracy = evaluate(knn_bagging, X_test, y_test)

best_random = knn_gscv.best_estimator_
random_accuracy = evaluate(best_random, X_test, y_test)

print(f'Improvement of {100 * (bagging_accuracy - random_accuracy) /␣
 ↪random_accuracy:.3f}%.')
```

```
Model Performance
Accuracy = 0.615%.
Model Performance
Accuracy = 0.634%.
Improvement of -3.046%.
```

[ ]: