driving_behavior_brf_v3

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      VelX      VelY
0     0.000000  0.000000  0.101938  NORMAL  0.000000  0.000000
1    -1.624864 -1.082492  0.135536  NORMAL -0.812432 -0.541246
2    -0.594660 -0.122410  0.087888  NORMAL -0.297330 -0.061205
3     0.738478 -0.228456  0.054902  NORMAL  0.369239 -0.114228
4     0.101741  0.777568  0.054902  NORMAL  0.050871  0.388784
…          …         …         …     …         …         …
3639  0.915688 -2.017489 -1.236468    SLOW  0.457844 -1.008745
3640 -1.934203  0.914925 -0.477162    SLOW -0.967102  0.457462
3641 -0.222845  0.747304  0.054291    SLOW -0.111422  0.373652
3642 -0.349423  0.067261 -0.004963    SLOW -0.174712  0.033630
3643 -0.402428  0.406218  0.001145    SLOW -0.201214  0.203109

[3644 rows x 6 columns]
```

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

[ ]:
```
AccX     0
AccY     0
GyroZ    0
Class    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      VelX      VelY
0     0.000000  0.000000  0.101938      0  0.000000  0.000000
1    -1.624864 -1.082492  0.135536      0 -0.812432 -0.541246
2    -0.594660 -0.122410  0.087888      0 -0.297330 -0.061205
3     0.738478 -0.228456  0.054902      0  0.369239 -0.114228
4     0.101741  0.777568  0.054902      0  0.050871  0.388784
…          …         …         …    …       …         …
3639  0.915688 -2.017489 -1.236468      1  0.457844 -1.008745
3640 -1.934203  0.914925 -0.477162      1 -0.967102  0.457462
3641 -0.222845  0.747304  0.054291      1 -0.111422  0.373652
3642 -0.349423  0.067261 -0.004963      1 -0.174712  0.033630
3643 -0.402428  0.406218  0.001145      1 -0.201214  0.203109

[3644 rows x 6 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      VelX      VelY
0     0.000000  0.000000  0.101938      0  0.000000  0.000000
1    -1.624864 -1.082492  0.135536      0 -0.812432 -0.541246
2    -0.594660 -0.122410  0.087888      0 -0.297330 -0.061205
3     0.738478 -0.228456  0.054902      0  0.369239 -0.114228
4     0.101741  0.777568  0.054902      0  0.050871  0.388784
…          …         …         …    …       …         …
2308  0.538870 -1.645984  0.662712      2  0.269435 -0.822992
2309  1.678918 -1.392127 -0.168675      2  0.839459 -0.696064
2310  0.323433  0.589311  0.639500      2  0.161716  0.294656
```

```
2311  2.497311 -0.606175 -0.240757      2  1.248655 -0.303088
2312  0.482297 -0.090277 -0.383700      2  0.241148 -0.045139

[2313 rows x 6 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        VelX        VelY
0      -3.509345    9.776257   74.896498   -3.509345    9.776257
1    -157.992905  -99.985349  102.351035 -157.992905  -99.985349
2     -60.046498   -2.635757   63.415515  -60.046498   -2.635757
3      66.701299  -13.388488   36.460154   66.701299  -13.388488
4       6.163664   88.619412   36.460154    6.163664   88.619412
…            …          …          …          …          …
2308   47.723614 -157.121833  533.137616   47.723614 -157.121833
2309  156.113434 -131.381523 -146.237282  156.113434 -131.381523
2310   27.240939   69.530762  514.169013   27.240939   69.530762
2311  233.921883  -51.688213 -205.139734  233.921883  -51.688213
2312   42.344893    0.622405 -321.946292   42.344893    0.622405

[2313 rows x 5 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)
```

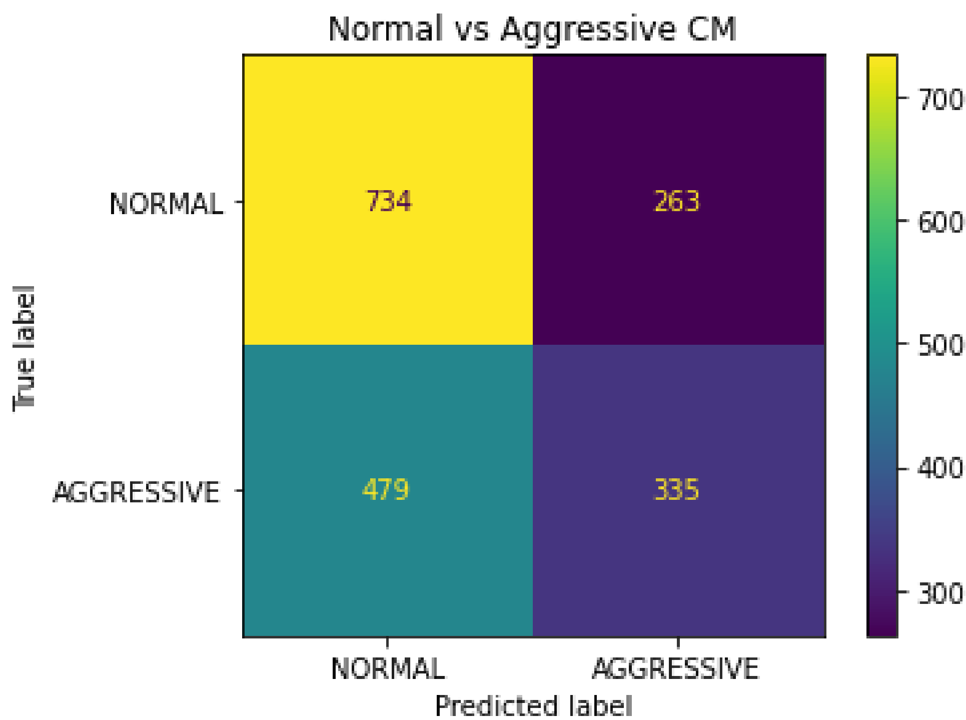
[ ]: 0.8698659749243407

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

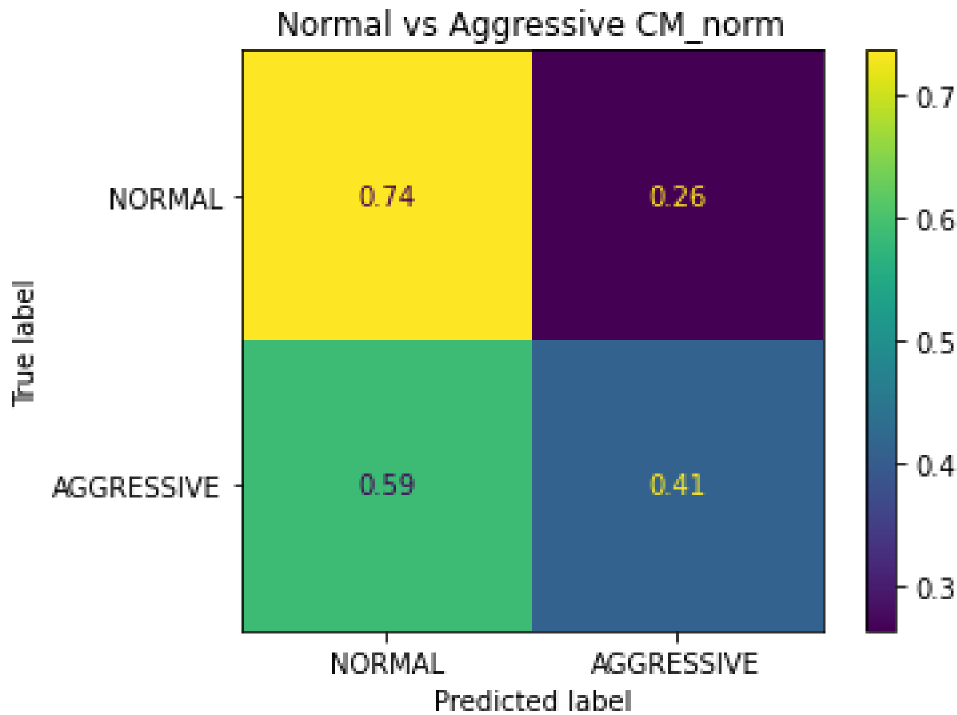
[ ]: 0.590281612368857

[ ]:
```
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)
```

```
[ ]: 0.590281612368857
```

```
[ ]: 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    20.707617
     AccY    18.978936
     GyroZ   19.055211
     VelX    19.951624
     VelY    21.306612
```

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

```
[ ]:        importance
     VelY    21.306612
     AccX    20.707617
     VelX    19.951624
```

```
GyroZ    19.055211
AccY     18.978936
```

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)]

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, 2:2.8}
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': 128,
 'min_samples_split': 4,
 'min_samples_leaf': 1,
 'max_features': None,
 'max_depth': 20,
 'bootstrap': True}
```

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

[ ]:
```
0.5602965788710929
```

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

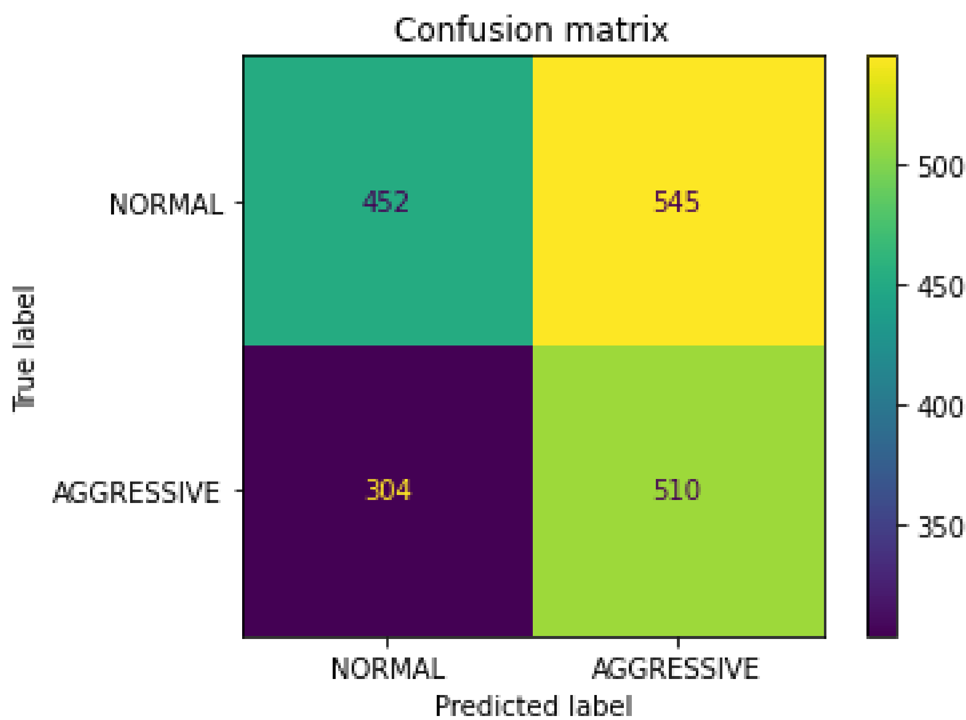
[ ]:
```
0.9818417639429312
```

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

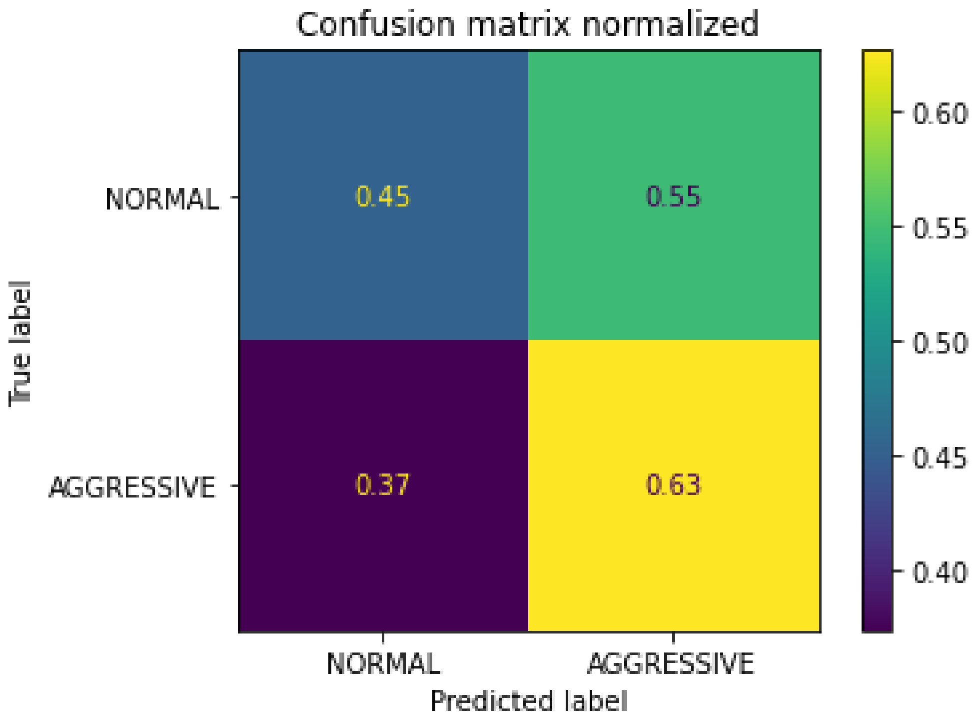
[ ]: 0.5311982330204307

[ ]:
```
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 = 0.568%.
Model Performance
Accuracy = 0.531%.
```

Improvement of -6.511%.

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': 86}

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

[ ]: 0.6143614484867184

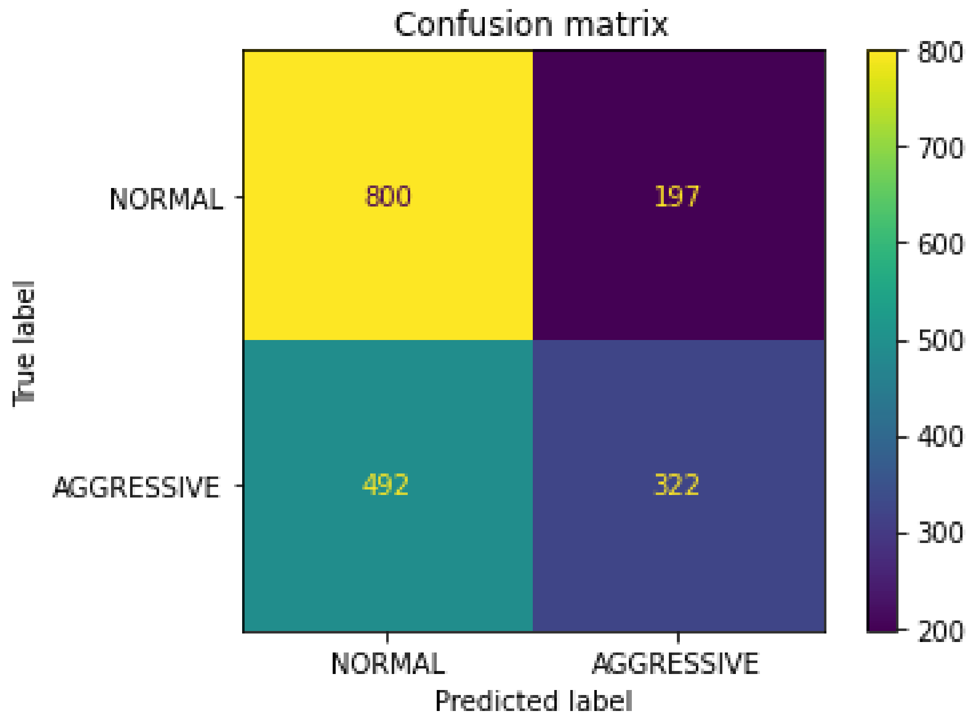
[ ]:
```
knn_gscv.score(X_train, y_train)
```

[ ]: 0.6191093817552962

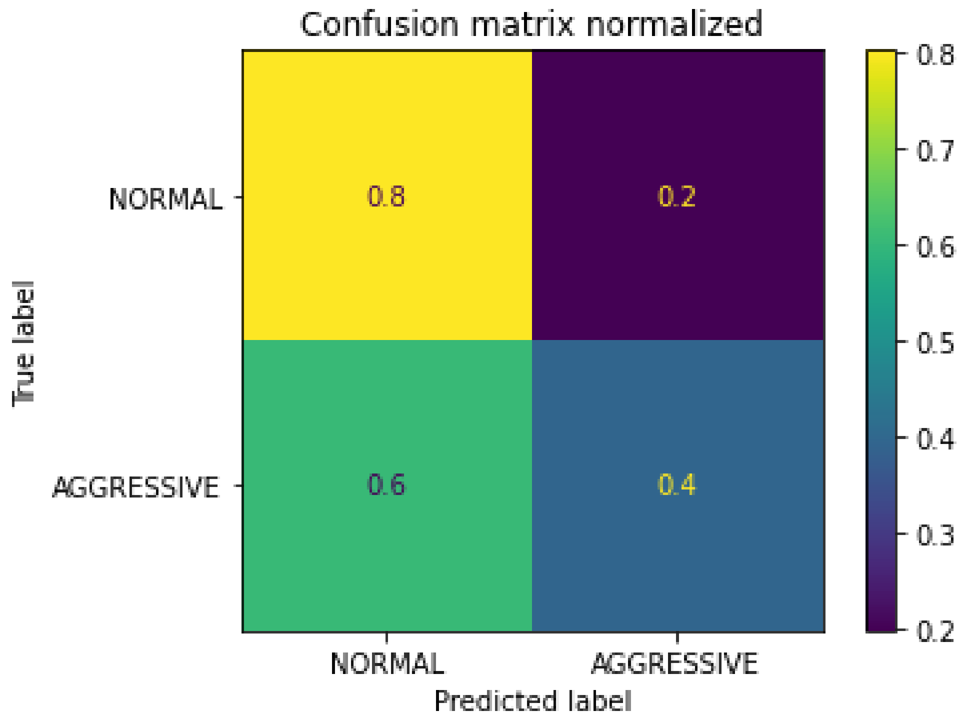
[ ]:
```
knn_gscv.score(X_test, y_test)
```

[ ]: 0.6195472114853672

[ ]:
```
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=0.9, max_features=0.8, random_state=0)
knn_bagging.fit(X_train, y_train)
```

[ ]:
```
BaggingClassifier(base_estimator=KNeighborsClassifier(leaf_size=20,
                                                      n_neighbors=86),
                  max_features=0.8, max_samples=0.9, random_state=0)
```

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

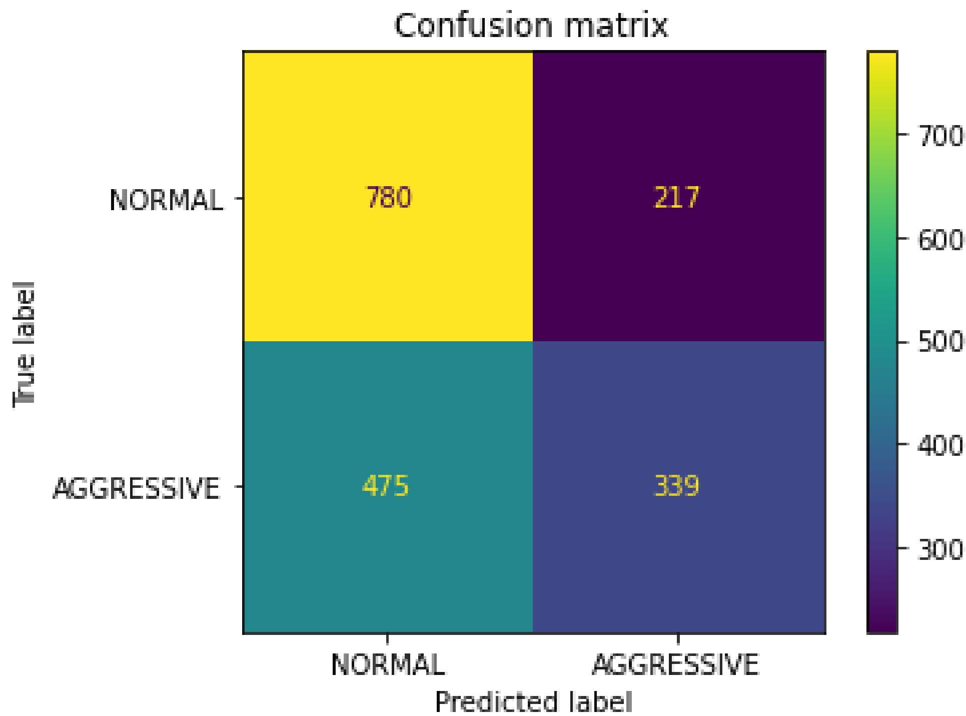
[ ]:
```
0.622568093385214
```

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

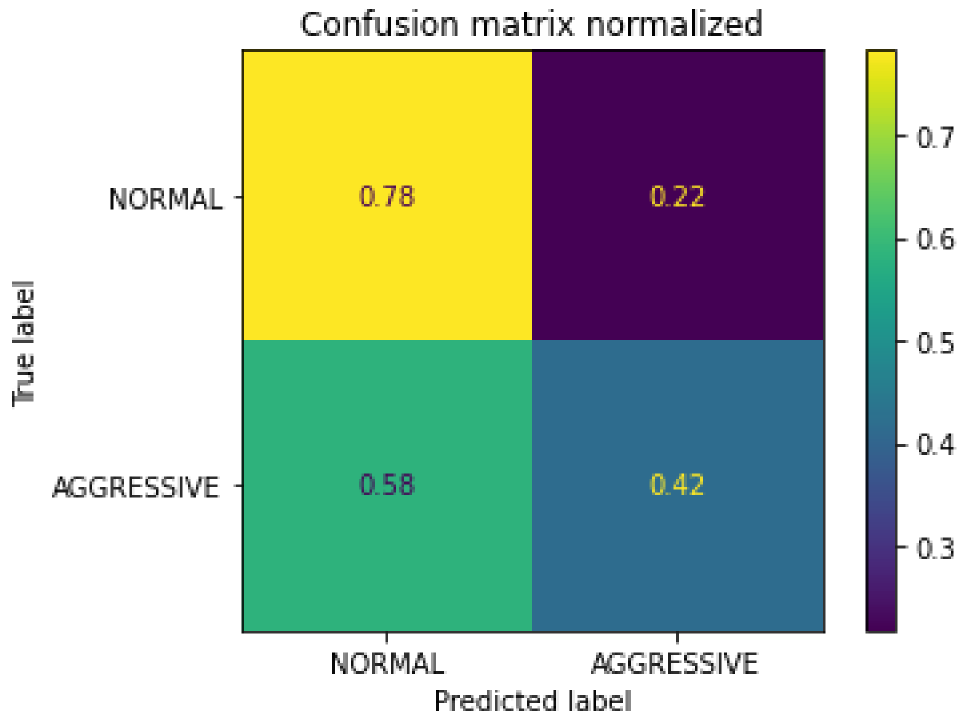
[ ]:
```
0.6178906681391496
```

[ ]:
```
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 = 0.618%.
Model Performance
Accuracy = 0.620%.
Improvement of -0.267%.
```

[ ]: