driving_behavior_XGBoost_binary_v2

September 1, 2022

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

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

df_training
```

[ ]:

	AccX	AccY	GyroZ	Class	DiffAccX	DiffAccY	VelX	VelY
0	0.000000	0.000000	0.101938	NORMAL	0.000000	0.000000	0.000000	0.000000
1	-1.624864	-1.082492	0.135536	NORMAL	-1.624864	-1.082492	-0.812432	-0.541246
2	-0.594660	-0.122410	0.087888	NORMAL	1.030204	0.960082	-0.297330	-0.061205
3	0.738478	-0.228456	0.054902	NORMAL	1.333138	-0.106046	0.369239	-0.114228
4	0.101741	0.777568	0.054902	NORMAL	-0.636737	1.006023	0.050871	0.388784
…	…	…	…	…	…	…	…	…
3639	0.915688	-2.017489	-1.236468	SLOW	2.374675	-1.824629	0.457844	-1.008745
3640	-1.934203	0.914925	-0.477162	SLOW	-2.849891	2.932414	-0.967102	0.457462
3641	-0.222845	0.747304	0.054291	SLOW	1.711359	-0.167621	-0.111422	0.373652
3642	-0.349423	0.067261	-0.004963	SLOW	-0.126579	-0.680043	-0.174712	0.033630
3643	-0.402428	0.406218	0.001145	SLOW	-0.053005	0.338957	-0.201214	0.203109

[3644 rows x 8 columns]

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

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

0.1 Change categories to numbers

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

[ ]:
```
          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      2  2.374675 -1.824629  0.457844
3640 -1.934203  0.914925 -0.477162      2 -2.849891  2.932414 -0.967102
3641 -0.222845  0.747304  0.054291      2  1.711359 -0.167621 -0.111422
3642 -0.349423  0.067261 -0.004963      2 -0.126579 -0.680043 -0.174712
3643 -0.402428  0.406218  0.001145      2 -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]
```

0.1.1 Only select normal and aggressive values

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

df_training
```

[ ]:
```
          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      2  2.374675 -1.824629  0.457844
3640 -1.934203  0.914925 -0.477162      2 -2.849891  2.932414 -0.967102
3641 -0.222845  0.747304  0.054291      2  1.711359 -0.167621 -0.111422
3642 -0.349423  0.067261 -0.004963      2 -0.126579 -0.680043 -0.174712
3643 -0.402428  0.406218  0.001145      2 -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

[2531 rows x 8 columns]
```

0.2 Normalize the data

[ ]:
```
X_training = df_training.drop(columns=["Class"])
X_training = (X_training - X_training.mean()) / X_training.std() * 100

X_training["Class"] = df_training["Class"]
X_training
```

[ ]:
```
             AccX        AccY        GyroZ    DiffAccX    DiffAccY        VelX  \
0      -1.855230    3.971188    88.116927    0.012569   -0.067264   -1.855230
1    -190.162298 -135.853745   119.158011 -160.827145 -106.518393 -190.162298
2     -70.770948  -11.840434    75.136116  101.988935   94.346206  -70.770948
```

```
3      83.727731  -25.538301    44.659418  131.975345  -10.495686   83.727731
4       9.935643  104.409213    44.659418  -63.015859   98.864017    9.935643
…            …         …           …          …           …
3639  104.264697 -256.626925 -1148.446773  235.073473 -179.499382  104.264697
3640 -226.011955  122.151549  -446.918404 -282.088379  288.303311 -226.011955
3641  -27.680909  100.500014    44.095035  169.414117  -16.550950  -27.680909
3642  -42.350223   12.659225   -10.650142  -12.517010  -66.941969  -42.350223
3643  -48.492982   56.442174    -5.006309   -5.234178   33.265449  -48.492982

            VelY  Class
0       3.971188      0
1    -135.853745      0
2     -11.840434      0
3     -25.538301      0
4     104.409213      0
…            …    …
3639 -256.626925      2
3640  122.151549      2
3641  100.500014      2
3642   12.659225      2
3643   56.442174      2

[2531 rows x 8 columns]
```

[ ]:
```
X_testing = df_test.drop(columns="Class")
X_testing = (X_testing - X_testing.mean()) / X_testing.std() * 100

X_testing["Class"] = df_test["Class"]
X_testing
```

[ ]:
```
            AccX        AccY       GyroZ    DiffAccX    DiffAccY        VelX  \
814    79.340838   21.963793   38.859198    4.511762   78.994793   79.340838
815   132.192943  569.257650  -11.298662   43.314646  415.888708  132.192943
816   -33.998774   10.843662   -4.956864 -136.351589 -424.416667  -33.998774
817    38.437952   57.366915    1.961463   59.378533   35.317808   38.437952
818   -21.053767    3.156469  -25.711843  -48.833088  -41.236835  -21.053767
…            …         …          …           …           …
3079  -95.464081  -76.862781  479.325902  -79.996459   -1.185708  -95.464081
3080  180.478081   51.148641 -645.478582  226.299722   97.246415  180.478081
3081  151.492731 -217.785674 -450.612340  -23.810860 -204.420646  151.492731
3082  105.929590   84.125864  374.398012  -37.408456  229.405378  105.929590
3083  174.027088   32.946129   63.073341   55.819369  -38.933584  174.027088

            VelY  Class
814    21.963793      0
815   569.257650      0
816    10.843662      0
```

```
817    57.366915      0
818     3.156469      0
…           …     …
3079  -76.862781      2
3080   51.148641      2
3081 -217.785674      2
3082   84.125864      2
3083   32.946129      2

[2270 rows x 8 columns]
```

0.3 Train model

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

X_test = X_testing.drop(columns="Class")
y_test = X_testing.Class
```

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

[ ]:
```
xgb = GradientBoostingClassifier()

param_grid = {'n_estimators': np.arange(20, 80, 2), 'learning_rate': np.
 ↪linspace(0.2, 1, 20), 'max_depth': np.arange(1, 10), 'max_features':␣
 ↪['sqrt', None], 'max_leaf_nodes': np.arange(2, 30)}

xgb_gscv = RandomizedSearchCV(xgb, param_grid, n_iter=100, cv=5, verbose=10,␣
 ↪n_jobs=10, random_state=0)
xgb_gscv.fit(X_train, y_train)
```

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

[ ]:
```
{'n_estimators': 34,
 'max_leaf_nodes': 11,
 'max_features': None,
 'max_depth': 4,
 'learning_rate': 0.4105263157894737}
```

[ ]:
```
xgb_gscv.best_score_
```

[ ]:
```
0.5282448877766603
```

0.3.1 Check for overfitting

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

[ ]:
```
0.7890161991307784
```

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

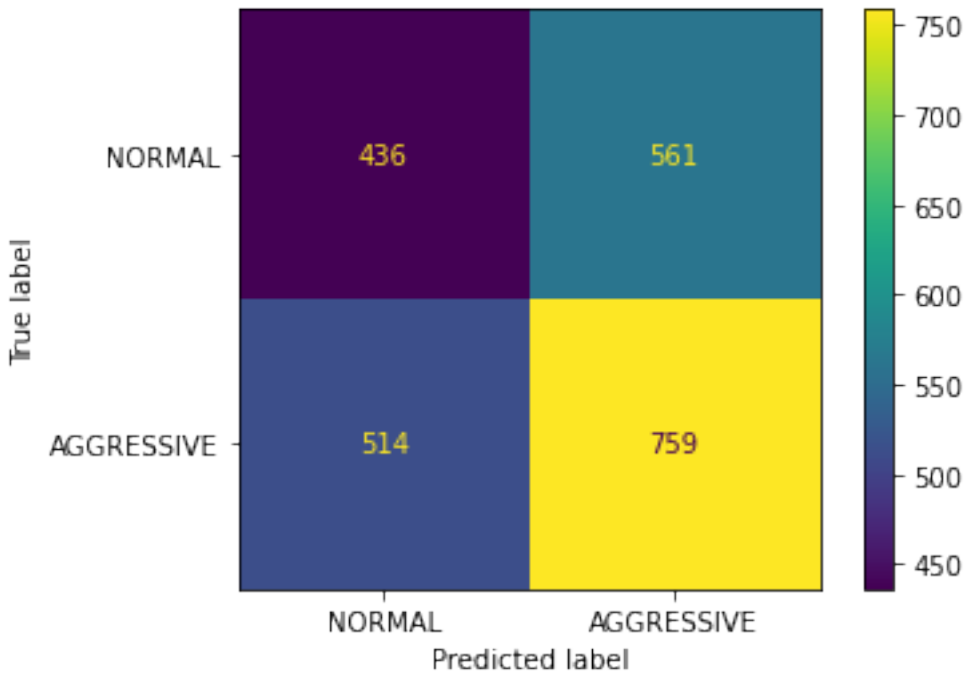
[ ]:
```
0.526431718061674
```

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

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

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

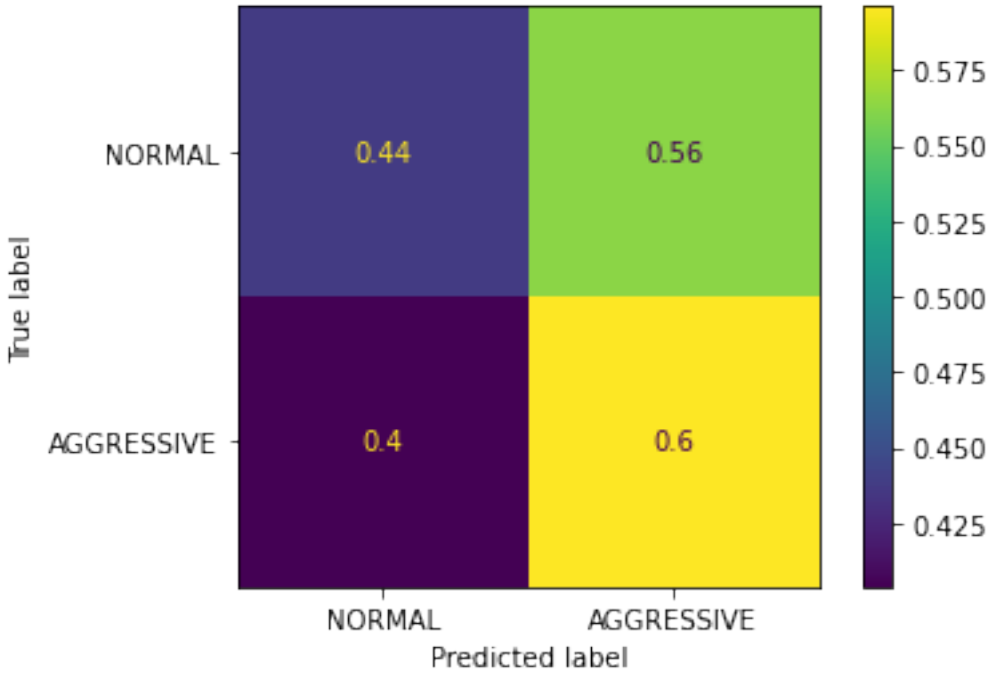
[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7ff074af2220>
```



[ ]:
```
CM_norm = confusion_matrix(y_test, y_pred, normalize="true")
display = ConfusionMatrixDisplay(confusion_matrix=CM_norm,
                         display_labels=classes)
```

```
display.plot()
```

[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7ff074c89ac0>



[ ]:
```
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 = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,␣
 ↪max_depth=1, random_state=0)
base_model.fit(X_train, y_train)
base_accuracy = evaluate(base_model, X_test, y_test)

best_random = xgb_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 = 0.522%.
```

```
Model Performance
Accuracy = 0.526%.
Improvement of 0.929%.
```

[ ]: