result-20-05-2024

May 20, 2024

[ ]:
```python
import sys
import os

# Aggiungi il percorso del livello superiore al sys.path
sys.path.append(os.path.abspath(os.path.join('..')))
```

[ ]:
```python
import src.data.make_dataset as mk

features_list = ['t0', 't1', 't2', 'a0', 'a1', 'a2', 'b0', 'b1', 'b2', 'c0',␣
↪'c1', 'c2', 'dicnotch', 'winSys', 'maxAmpl', 'sysTime', 'duration',  'Es',␣
↪'As', 'Ed', 'Ad', 'td', 'fd', 'R2_of_fit']
feature_data_path  = r"C:
↪\Users\cical\Documents\GitHub\Repositories\tesina\data\interim\feature_extracted"

feature_df = mk.organize_data2(feature_data_path, features_list)
```

```
c:\Users\cical\Documents\GitHub\Repositories\tesina\src\data\make_dataset.py:49:
FutureWarning: The behavior of DataFrame concatenation with empty or all-NA
entries is deprecated. In a future version, this will no longer exclude empty or
all-NA columns when determining the result dtypes. To retain the old behavior,
exclude the relevant entries before the concat operation.
  df = pd.concat([df, data_df], ignore_index=True)
```

1 Data Pre-Processing

1.1 Split Dataset

[ ]:
```python
X_train, X_test, y_train, y_test = mk.split_train_test(feature_df, 'Group', 0.2)
```

1.2 Data selection (outliers and p-value)

[ ]:
```python
import src.data.data_selection as ds

X_train, y_train = ds.filter_outliers_by_group(X_train, y_train,␣
↪features_to_ignore=['PatientID', 'SignalID', 'R2_of_fit', 'pulse_index'])

X_train, y_train = ds.filter_fit_value(X_train, y_train, 0.9, 'R2_of_fit')
```

1.3 Data Trasformation

1.3.1 Patient median

[ ]:
```
X_train, y_train = ds.calculate_patient_median(X_train, y_train, 'PatientID',␣
↪features_list)
X_test, y_test = ds.calculate_patient_median(X_test, y_test, 'PatientID',␣
↪features_list)

X_train.drop(columns=['PatientID',  'R2_of_fit'], inplace=True)
X_test.drop(columns=['PatientID',   'R2_of_fit'], inplace=True)
```

1.3.2 SMOTE+EEN

[ ]:
```
import src.data.make_dataset as mk
import src.visualization.visualize as vis

#vis.plot_class_distribution(y_train, title="Distribution before SMOTEEN")

#X_train, y_train = mk.balance_dataset(X_train, y_train)

#vis.plot_class_distribution(y_train, title="Distribution after SMOTEEN")
```

1.3.3 Scaling

[ ]:
```
import src.data.make_dataset as mk

# si scalano i dati di training e test dividendoli in due gruppi malati e non␣
↪malati
#y_train_mapped = y_train.map({'covid_Empoli_60': 0, 'sepsis_MIMIC_125': 0,␣
↪'healthyControl_Empoli_60': 1, 'mentalDisorders_MIMIC_125': 1})
#y_test_mapped = y_test.map({'covid_Empoli_60': 0, 'sepsis_MIMIC_125': 0,␣
↪'healthyControl_Empoli_60': 1, 'mentalDisorders_MIMIC_125': 1})
#X_train = mk.scale_numeric_features(X_train, y_train_mapped)
#X_test = mk.scale_numeric_features(X_test, y_test_mapped)
```

2 Statistical tests

2.1 Shapiro-Wilk test

[ ]:
```
import src.statistics.tests as tests
import src.visualization.visualize as vis

normality_test = tests.shapiro_test(X_train, y_train)

for group in normality_test['Group'].unique():
    sub_df = normality_test[normality_test['Group'] == group]
```

```
    vis.plot_pvalues(sub_df, group)
```

2.2 Friedman test

[ ]:
```
import src.statistics.tests as tests
import src.visualization.visualize as vis

friedman_test = tests.friedman_test(X_train, y_train)

vis.plot_pvalues(friedman_test, 'Friedman Test')
```

2.3 Post-hoc Mann-Whitney U

[ ]:
```
import src.statistics.tests as tests

# valuto differenze tra gruppi per le features significative del test di␣
 ↪Friedman
significant_features = friedman_test[friedman_test['P-value'] < 0.
 ↪05]['Feature'].tolist()
significant_features_dict = tests.pairwise_mann_whitney_test(X_train, y_train,␣
 ↪significant_features=significant_features)
```

[ ]:
```
import src.visualization.visualize as vis

for group_pair, result in significant_features_dict.items():
    vis.plot_pvalues(result, group_pair)
```

2.4 Unified dataset

[ ]:
```
import src.statistics.tests as tests
import src.visualization.visualize as vis

y_train_unified = y_train.copy()

# Mappa i valori della colonna 'Group' come richiesto
mapping = {'covid_Empoli_60': "sick", 'sepsis_MIMIC_125': "sick",
           'healthyControl_Empoli_60': "healthy", 'mentalDisorders_MIMIC_125':␣
 ↪"healthy"}
y_train_unified = y_train_unified.replace(mapping)

unified_result = tests.mann_whitney_test(X_train, y_train_unified)

vis.plot_pvalues(unified_result, 'Mann-Whitney Test')
```

3 Training Model

3.1 Primo test

Nel primo test si cerca di addestrare un modello che permetta di identificare tra gruppo di sani (heamth and mental disorders) e patologici (covid o sepsi). A questo scopo vengono utilizzate le features con un valore di p-value al di sotto della soglia impostata del test di mann-Whitney per il dataset unificato.

[ ]:
```
import src.data.make_dataset as mk

features_test_1 = unified_result[unified_result['P-value'] < 0.05]['Feature'].
 ↪tolist()

X_train_t1 = X_train[features_test_1]
X_test_t1 = X_test[features_test_1]

y_train_t1 = y_train.map({'covid_Empoli_60': 0, 'sepsis_MIMIC_125': 0,
                              'healthyControl_Empoli_60': 1,␣
 ↪'mentalDisorders_MIMIC_125': 1})

y_test_t1 = y_test.map({'covid_Empoli_60': 0, 'sepsis_MIMIC_125': 0,
                              'healthyControl_Empoli_60': 1,␣
 ↪'mentalDisorders_MIMIC_125': 1})
```

3.1.1 Cross validation

Si effettua una cross validazione con StratifiedKFold (cv=5) e si valutano le performance dei modelli per f1_macro e il coeff. di correlazioen di Matthews. I tre modelli che presentano le prestazioni migliori veranno poi migliorati con un ottizzazione degli iperparametri.

[ ]:
```
import src.models.cross_validation as cv

models = cv.define_models()
metric_results = cv.evaluate_models(X_train_t1, y_train_t1, models)

cv.summarize_results(metric_results)
```

```
Models Evaluation with f1_macro: 100%|      | 10/10 [00:11<00:00,  1.19s/it]
Models Evaluation with make_scorer(matthews_corrcoef,
response_method='predict'): 100%|      | 10/10 [00:04<00:00,  2.06it/s]


Metric: f1_macro
Rank=1, Name=catboost, Score=0.664 (+/- 0.088)
Rank=2, Name=adaboost, Score=0.652 (+/- 0.029)
Rank=3, Name=nb, Score=0.652 (+/- 0.040)
Rank=4, Name=gbm, Score=0.635 (+/- 0.065)
Rank=5, Name=dt, Score=0.634 (+/- 0.022)
Rank=6, Name=rf, Score=0.631 (+/- 0.074)
```

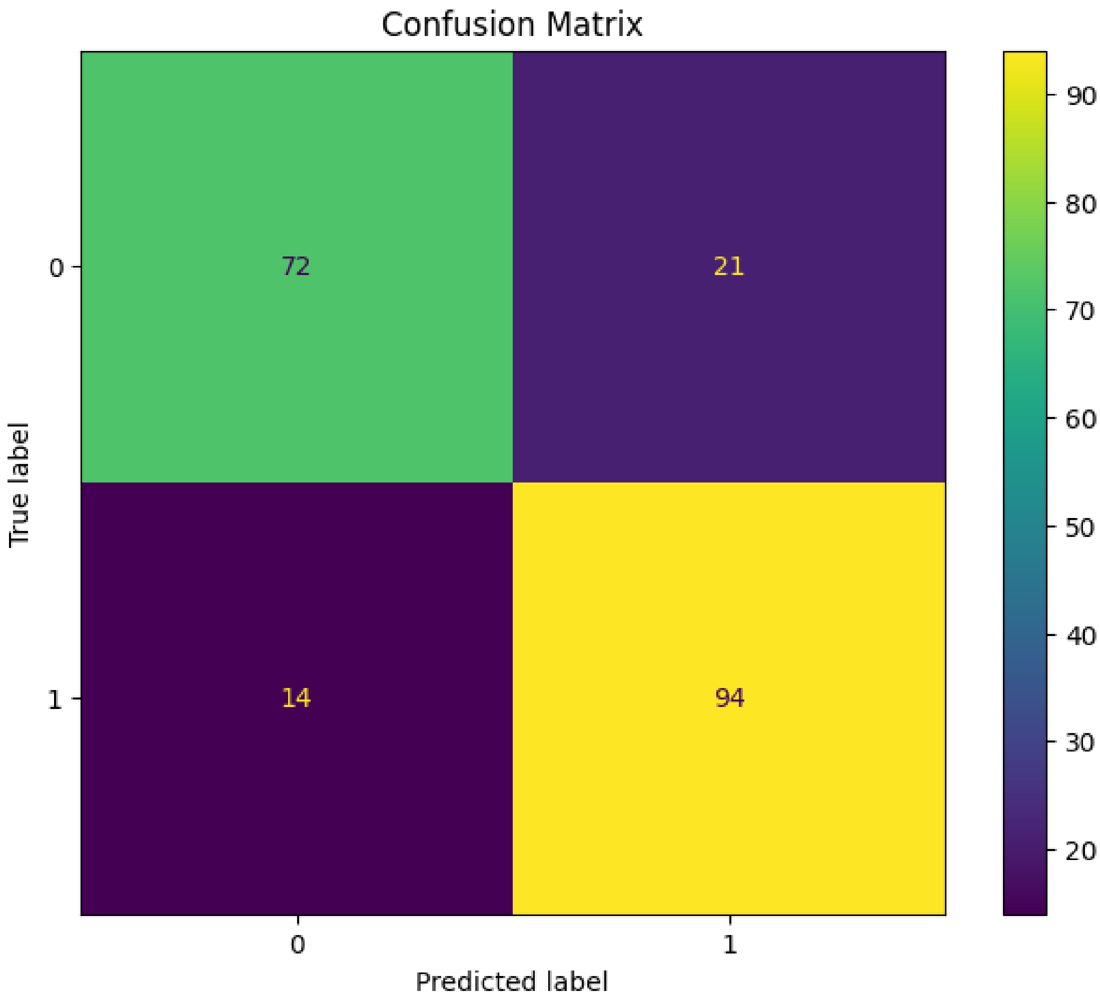
```
Rank=7, Name=gpc, Score=0.628 (+/- 0.079)
Rank=8, Name=nc, Score=0.596 (+/- 0.111)
Rank=9, Name=svm, Score=0.577 (+/- 0.128)
Rank=10, Name=mlp, Score=0.389 (+/- 0.144)

Metric: make_scorer(matthews_corrcoef, response_method='predict')
Rank=1, Name=rf, Score=0.357 (+/- 0.138)
Rank=2, Name=catboost, Score=0.351 (+/- 0.181)
Rank=3, Name=nb, Score=0.333 (+/- 0.067)
Rank=4, Name=adaboost, Score=0.320 (+/- 0.051)
Rank=5, Name=gbm, Score=0.283 (+/- 0.185)
Rank=6, Name=gpc, Score=0.281 (+/- 0.164)
Rank=7, Name=svm, Score=0.254 (+/- 0.247)
Rank=8, Name=dt, Score=0.245 (+/- 0.136)
Rank=9, Name=nc, Score=0.201 (+/- 0.216)
Rank=10, Name=mlp, Score=0.054 (+/- 0.169)
```

[ ]:
```
import src.models.cross_validation as cv

cv.evaluate_optimized_models(X_train_t1, y_train_t1, model_names=['nb',␣
↪'catboost', 'rf'], metric='f1_macro', cv=5)
```

```
Model nb does not have hyperparameters for fine tuning
Model nb - Score=0.652 (+/- 0.040)
---------------------------------------------------------------
Best hyperparameters for model catboost: {'iterations': 300, 'learning_rate':
0.01}
Model catboost - Score=0.673 (+/- 0.093)
---------------------------------------------------------------
Best hyperparameters for model rf: {'max_depth': 20, 'min_samples_leaf': 4,
'min_samples_split': 5, 'n_estimators': 100}
Model rf - Score=0.647 (+/- 0.056)
---------------------------------------------------------------
```

3.1.2 Training and test first model

[ ]:
```
from catboost import CatBoostClassifier

import src.visualization.visualize as vis

model = CatBoostClassifier(iterations=300, learning_rate=0.01,␣
↪logging_level='Silent')
model.fit(X_train_t1, y_train_t1)

vis.plot_model_performance(model, X_test_t1, y_test_t1, 'accuracy')
```



Accuracy: 0.83

3.2 Secondo test (ill)

[ ]:
```
import src.data.make_dataset as mk

result_statistical_ill = significant_features_dict[('covid_Empoli_60',␣
↪'sepsis_MIMIC_125')]
features_test_2 = result_statistical_ill[result_statistical_ill['P-value'] < 0.
↪05]['Feature'].tolist()

#features_test_2 = list(set(unified_result[unified_result['P-value'] > 0.
↪05]['Feature'].tolist()).intersection(features_test_2))

X_train_t2 = X_train[features_test_2]
```

```
X_test_t2 = X_test[features_test_2]

# si rimuovono le righe relative ai gruppi 'healthyControl_Empoli_60' e␣
↪'mentalDisorders_MIMIC_125'
target_values = ['covid_Empoli_60', 'sepsis_MIMIC_125']
X_train_t2, y_train_t2 = mk.filter_rows_by_values(X_train_t2, y_train,␣
↪target_values)
X_test_t2, y_test_t2 = mk.filter_rows_by_values(X_test_t2, y_test,␣
↪target_values)
```

3.2.1 Cross-validation

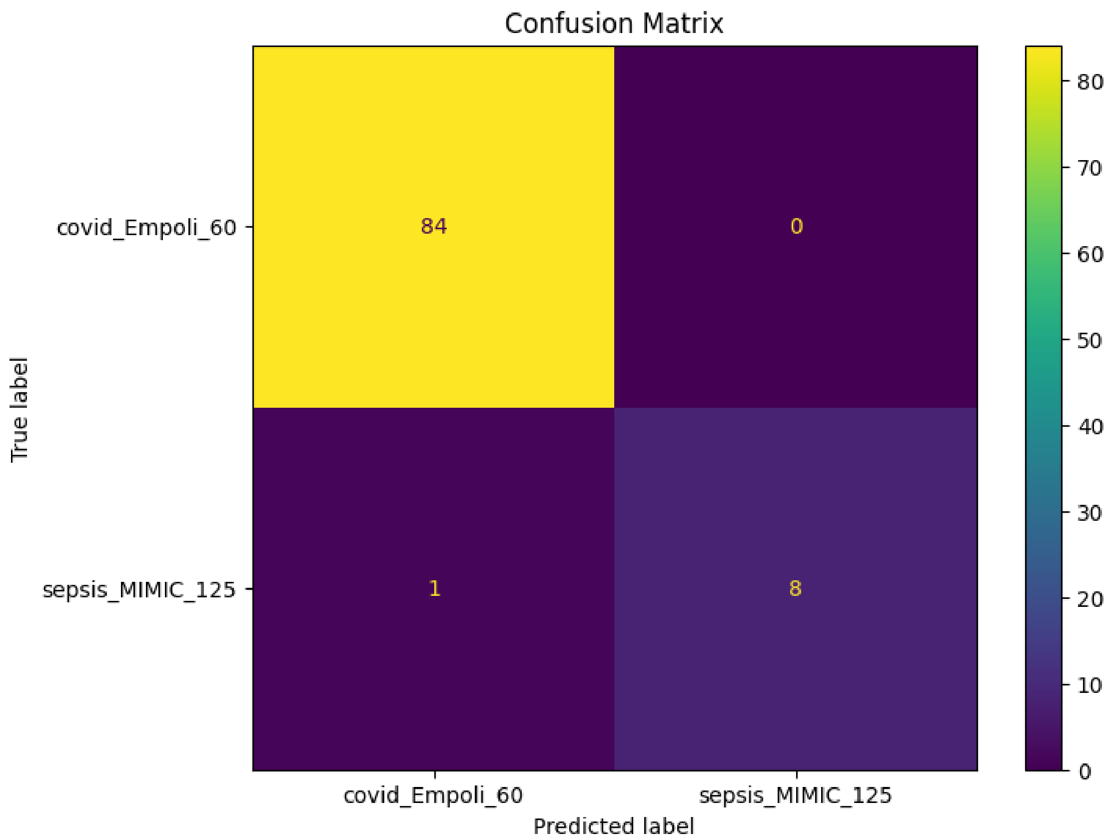
[ ]:
```
import src.models.cross_validation as cv

models = cv.define_models()
metric_results = cv.evaluate_models(X_train_t2, y_train_t2, models)

cv.summarize_results(metric_results)
```

```
Models Evaluation with f1_macro: 100%|      | 10/10 [00:03<00:00,  2.51it/s]
Models Evaluation with make_scorer(matthews_corrcoef,
response_method='predict'): 100%|      | 10/10 [00:03<00:00,  2.57it/s]


Metric: f1_macro
Rank=1, Name=dt, Score=0.944 (+/- 0.125)
Rank=2, Name=gbm, Score=0.944 (+/- 0.125)
Rank=3, Name=catboost, Score=0.944 (+/- 0.125)
Rank=4, Name=adaboost, Score=0.905 (+/- 0.134)
Rank=5, Name=rf, Score=0.854 (+/- 0.106)
Rank=6, Name=nb, Score=0.838 (+/- 0.212)
Rank=7, Name=gpc, Score=0.769 (+/- 0.230)
Rank=8, Name=nc, Score=0.640 (+/- 0.142)
Rank=9, Name=svm, Score=0.574 (+/- 0.239)
Rank=10, Name=mlp, Score=0.468 (+/- 0.019)

Metric: make_scorer(matthews_corrcoef, response_method='predict')
Rank=1, Name=dt, Score=0.709 (+/- 0.443)
Rank=2, Name=gbm, Score=0.709 (+/- 0.443)
Rank=3, Name=nb, Score=0.696 (+/- 0.412)
Rank=4, Name=catboost, Score=0.688 (+/- 0.455)
Rank=5, Name=adaboost, Score=0.642 (+/- 0.412)
Rank=6, Name=rf, Score=0.531 (+/- 0.329)
Rank=7, Name=nc, Score=0.390 (+/- 0.246)
Rank=8, Name=gpc, Score=0.380 (+/- 0.413)
Rank=9, Name=mlp, Score=0.184 (+/- 0.458)
Rank=10, Name=svm, Score=0.000 (+/- 0.000)
```

[ ]:
```
import src.models.cross_validation as cv

cv.evaluate_optimized_models(X_train_t2, y_train_t2, model_names=['dt', 'gbm',␣
↪'catboost', 'nb'], metric='f1_macro', cv=5)
```

```
Best hyperparameters for model dt: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 10}
Model dt - Score=0.944 (+/- 0.125)
---------------------------------------------------------------
Best hyperparameters for model gbm: {'learning_rate': 0.1, 'max_depth': 3,
'n_estimators': 50}
Model gbm - Score=0.944 (+/- 0.125)
---------------------------------------------------------------
Best hyperparameters for model catboost: {'iterations': 100, 'learning_rate':
0.1}
Model catboost - Score=0.928 (+/- 0.099)
---------------------------------------------------------------
Model nb does not have hyperparameters for fine tuning
Model nb - Score=0.838 (+/- 0.212)
---------------------------------------------------------------
```

[ ]:
```
from sklearn.ensemble import GradientBoostingClassifier
import src.visualization.visualize as vis

model = GradientBoostingClassifier(learning_rate=0.1, max_depth= 3,␣
↪n_estimators=50)
model.fit(X_train_t2, y_train_t2)

vis.plot_model_performance(model, X_test_t2, y_test_t2, 'f1-score')
```



F1-score: 0.97

3.3 Test 3 (Health)

[ ]:
```
import src.data.make_dataset as mk

result_statistical_health =␣
  ↪significant_features_dict[('mentalDisorders_MIMIC_125',␣
  ↪'healthyControl_Empoli_60')]
features_test_3 =␣
  ↪result_statistical_health[result_statistical_health['P-value'] < 0.
  ↪05]['Feature'].tolist()

#features_test_3 = list(set(unified_result[unified_result['P-value'] > 0.
  ↪05]['Feature'].tolist()).intersection(features_test_3))

X_train_t3 = X_train[features_test_3]
X_test_t3 = X_test[features_test_3]
```

```
# si rimuovono le righe relative ai gruppi 'covid_Empoli_60' e␣
↪'sepsis_MIMIC_125'
target_values = ['mentalDisorders_MIMIC_125', 'healthyControl_Empoli_60']
X_train_t3, y_train_t3 = mk.filter_rows_by_values(X_train_t3, y_train,␣
↪target_values)
X_test_t3, y_test_t3 = mk.filter_rows_by_values(X_test_t3, y_test,␣
↪target_values)
```

3.3.1 Cross-validation

```
[ ]: import src.models.cross_validation as cv

models = cv.define_models()
metric_results = cv.evaluate_models(X_train_t3, y_test_t3, models)

cv.summarize_results(metric_results)
```

```
Models Evaluation with f1_macro: 100%|      | 10/10 [00:04<00:00,  2.48it/s]
Models Evaluation with make_scorer(matthews_corrcoef,
response_method='predict'): 100%|      | 10/10 [00:03<00:00,  2.61it/s]


Metric: f1_macro
Rank=1, Name=nb, Score=1.000 (+/- 0.000)
Rank=2, Name=rf, Score=0.983 (+/- 0.038)
Rank=3, Name=catboost, Score=0.983 (+/- 0.038)
Rank=4, Name=gbm, Score=0.969 (+/- 0.068)
Rank=5, Name=adaboost, Score=0.935 (+/- 0.068)
Rank=6, Name=svm, Score=0.920 (+/- 0.050)
Rank=7, Name=mlp, Score=0.920 (+/- 0.050)
Rank=8, Name=gpc, Score=0.920 (+/- 0.050)
Rank=9, Name=dt, Score=0.867 (+/- 0.222)
Rank=10, Name=nc, Score=0.757 (+/- 0.156)

Metric: make_scorer(matthews_corrcoef, response_method='predict')
Rank=1, Name=nb, Score=0.800 (+/- 0.447)
Rank=2, Name=rf, Score=0.769 (+/- 0.435)
Rank=3, Name=catboost, Score=0.769 (+/- 0.435)
Rank=4, Name=dt, Score=0.739 (+/- 0.434)
Rank=5, Name=gbm, Score=0.739 (+/- 0.434)
Rank=6, Name=adaboost, Score=0.675 (+/- 0.394)
Rank=7, Name=svm, Score=0.655 (+/- 0.368)
Rank=8, Name=mlp, Score=0.655 (+/- 0.368)
Rank=9, Name=gpc, Score=0.655 (+/- 0.368)
Rank=10, Name=nc, Score=0.541 (+/- 0.311)
```

[ ]:
```python
import src.models.cross_validation as cv

cv.evaluate_optimized_models(X_train_t3, y_train_t3, model_names=['nb',␣
↪'catboost', 'rf'], metric='f1_macro', cv=5)
```

```
Model nb does not have hyperparameters for fine tuning
Model nb - Score=1.000 (+/- 0.000)
---------------------------------------------------------------
Best hyperparameters for model catboost: {'iterations': 100, 'learning_rate':
0.2}
Model catboost - Score=0.983 (+/- 0.038)
---------------------------------------------------------------
Best hyperparameters for model rf: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}
Model rf - Score=0.983 (+/- 0.038)
---------------------------------------------------------------
```

[ ]:
```python
from sklearn.naive_bayes import GaussianNB
import src.visualization.visualize as vis

model = GaussianNB()
model.fit(X_train_t3, y_train_t3)

vis.plot_model_performance(model, X_test_t3, y_test_t3, 'f1-score')
```