pearrr\_original

Downloading completed ...

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[LibLinear]

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[LibLinear]

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[LibLinear]

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[LibLinear]

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

[LibLinear]

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\pearrr\_original\feature\_extractor.py:33](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\pearrr_original\feature_extractor.py:33): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df[col] = df[col].map(f)

TP+FN 23 11

TP+FN 11 13

TP+FN 23 11

TP+FN 8 16

TP+FN 18 16

TP+FN 12 12

TP+FN 26 8

TP+FN 11 13

TP+FN 21 13

TP+FN 14 10

Results: {'ga': {1: {'overall': 0.6521739130434783, 'male': 0.6515151515151515, 'female': 0.6530612244897959}, 2: {'overall': 0.6608695652173913, 'male': 0.7272727272727273, 'female': 0.5714285714285714}, 3: {'overall': 0.6521739130434783, 'male': 0.6363636363636364, 'female': 0.673469387755102}, 4: {'overall': 0.6869565217391305, 'male': 0.7121212121212122, 'female': 0.6530612244897959}, 5: {'overall': 0.6956521739130435, 'male': 0.696969696969697, 'female': 0.6938775510204082}}, 'eo': {1: {'overall': 0.21813725490196084}, 2: {'overall': 0.34313725490196084}, 3: {'overall': 0.02941176470588236}, 4: {'overall': 0.3063725490196078}, 5: {'overall': 0.03431372549019607}}}

abethe\_original

Downloading completed ...

[c:\Users\kharu\miniconda3\envs\autism\lib\site-packages\sklearn\svm\base.py:929](file:///C:\Users\kharu\miniconda3\envs\autism\lib\site-packages\sklearn\svm\base.py:929): ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.

"the number of iterations.", ConvergenceWarning)

TP+FN 22 12

TP+FN 10 14

TP+FN 21 13

TP+FN 12 12

TP+FN 25 9

TP+FN 11 13

TP+FN 22 12

TP+FN 11 13

TP+FN 20 14

TP+FN 13 11

Results: {'ga': {1: {'overall': 0.6434782608695652, 'male': 0.6515151515151515, 'female': 0.6326530612244898}, 2: {'overall': 0.6086956521739131, 'male': 0.6060606060606061, 'female': 0.6122448979591837}, 3: {'overall': 0.6347826086956522, 'male': 0.6515151515151515, 'female': 0.6122448979591837}, 4: {'overall': 0.6086956521739131, 'male': 0.5909090909090909, 'female': 0.6326530612244898}, 5: {'overall': 0.6434782608695652, 'male': 0.6212121212121212, 'female': 0.673469387755102}}, 'eo': {1: {'overall': 0.2303921568627451}, 2: {'overall': 0.11764705882352944}, 3: {'overall': 0.27696078431372556}, 4: {'overall': 0.18872549019607848}, 5: {'overall': 0.04656862745098045}}}

amicie\_original

Downloading completed ...

fitted

fitted

fitted

fitted

fitted

TP+FN 23 11

TP+FN 8 16

TP+FN 23 11

TP+FN 12 12

TP+FN 21 13

TP+FN 11 13

TP+FN 23 11

TP+FN 12 12

TP+FN 19 15

TP+FN 8 16

Results: {'ga': {1: {'overall': 0.6173913043478261, 'male': 0.6363636363636364, 'female': 0.5918367346938775}, 2: {'overall': 0.6173913043478261, 'male': 0.6212121212121212, 'female': 0.6122448979591837}, 3: {'overall': 0.6521739130434783, 'male': 0.6363636363636364, 'female': 0.673469387755102}, 4: {'overall': 0.6086956521739131, 'male': 0.5909090909090909, 'female': 0.6326530612244898}, 5: {'overall': 0.6260869565217392, 'male': 0.6515151515151515, 'female': 0.5918367346938775}}, 'eo': {1: {'overall': 0.34313725490196084}, 2: {'overall': 0.17647058823529416}, 3: {'overall': 0.15931372549019612}, 4: {'overall': 0.17647058823529416}, 5: {'overall': 0.2254901960784314}}}

ayoub\_ghriss\_original

Downloading completed ...

Fitting fmri\_motions, Done in 0.033 min

Fitting fmri\_basc064, Done in 0.129 min

Fitting fmri\_basc122, Done in 0.325 min

Fitting fmri\_craddock\_scorr\_mean, Done in 1.348 min

Fitting fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.089 min

Fitting fmri\_msdl, Done in 0.073 min

Fitting fmri\_power\_2011, Done in 1.498 min

Transforming fmri\_basc064, Done in 0.087 min

Transforming fmri\_basc122, Done in 0.151 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.318 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.066 min

Transforming fmri\_motions, Done in 0.026 min

Transforming fmri\_msdl, Done in 0.056 min

Transforming fmri\_power\_2011, Done in 0.340 min

Transforming fmri\_basc064, Done in 0.086 min

Transforming fmri\_basc122, Done in 0.149 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.314 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.066 min

Transforming fmri\_motions, Done in 0.026 min

Transforming fmri\_msdl, Done in 0.056 min

Transforming fmri\_power\_2011, Done in 0.337 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.027 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.043 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.006 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.010 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.008 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.009 min

Transforming fmri\_basc064, Done in 0.012 min

Transforming fmri\_basc122, Done in 0.021 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.045 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.010 min

Transforming fmri\_motions, Done in 0.005 min

Transforming fmri\_msdl, Done in 0.008 min

Transforming fmri\_power\_2011, Done in 0.048 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting fmri\_motions, Done in 0.033 min

Fitting fmri\_basc064, Done in 0.129 min

Fitting fmri\_basc122, Done in 0.307 min

Fitting fmri\_craddock\_scorr\_mean, Done in 1.271 min

Fitting fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.094 min

Fitting fmri\_msdl, Done in 0.075 min

Fitting fmri\_power\_2011, Done in 1.502 min

Transforming fmri\_basc064, Done in 0.088 min

Transforming fmri\_basc122, Done in 0.149 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.320 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.068 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.058 min

Transforming fmri\_power\_2011, Done in 0.341 min

Transforming fmri\_basc064, Done in 0.087 min

Transforming fmri\_basc122, Done in 0.148 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.318 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.068 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.057 min

Transforming fmri\_power\_2011, Done in 0.342 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.027 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.045 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.006 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.011 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.008 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.009 min

Transforming fmri\_basc064, Done in 0.012 min

Transforming fmri\_basc122, Done in 0.021 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.044 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.010 min

Transforming fmri\_motions, Done in 0.004 min

Transforming fmri\_msdl, Done in 0.008 min

Transforming fmri\_power\_2011, Done in 0.048 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting fmri\_motions, Done in 0.033 min

Fitting fmri\_basc064, Done in 0.130 min

Fitting fmri\_basc122, Done in 0.304 min

Fitting fmri\_craddock\_scorr\_mean, Done in 1.264 min

Fitting fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.094 min

Fitting fmri\_msdl, Done in 0.078 min

Fitting fmri\_power\_2011, Done in 1.504 min

Transforming fmri\_basc064, Done in 0.088 min

Transforming fmri\_basc122, Done in 0.148 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.317 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.068 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.057 min

Transforming fmri\_power\_2011, Done in 0.341 min

Transforming fmri\_basc064, Done in 0.087 min

Transforming fmri\_basc122, Done in 0.148 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.316 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.067 min

Transforming fmri\_motions, Done in 0.026 min

Transforming fmri\_msdl, Done in 0.057 min

Transforming fmri\_power\_2011, Done in 0.338 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.027 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.044 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.006 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.010 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.008 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.009 min

Transforming fmri\_basc064, Done in 0.012 min

Transforming fmri\_basc122, Done in 0.021 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.045 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.010 min

Transforming fmri\_motions, Done in 0.004 min

Transforming fmri\_msdl, Done in 0.009 min

Transforming fmri\_power\_2011, Done in 0.048 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting fmri\_motions, Done in 0.033 min

Fitting fmri\_basc064, Done in 0.128 min

Fitting fmri\_basc122, Done in 0.307 min

Fitting fmri\_craddock\_scorr\_mean, Done in 1.269 min

Fitting fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.093 min

Fitting fmri\_msdl, Done in 0.077 min

Fitting fmri\_power\_2011, Done in 1.496 min

Transforming fmri\_basc064, Done in 0.088 min

Transforming fmri\_basc122, Done in 0.149 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.316 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.067 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.057 min

Transforming fmri\_power\_2011, Done in 0.340 min

Transforming fmri\_basc064, Done in 0.087 min

Transforming fmri\_basc122, Done in 0.147 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.315 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.067 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.057 min

Transforming fmri\_power\_2011, Done in 0.338 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.027 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.044 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.003 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.006 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.011 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.008 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.009 min

Transforming fmri\_basc064, Done in 0.012 min

Transforming fmri\_basc122, Done in 0.021 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.044 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.010 min

Transforming fmri\_motions, Done in 0.004 min

Transforming fmri\_msdl, Done in 0.008 min

Transforming fmri\_power\_2011, Done in 0.047 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting fmri\_motions, Done in 0.033 min

Fitting fmri\_basc064, Done in 0.127 min

Fitting fmri\_basc122, Done in 0.301 min

Fitting fmri\_craddock\_scorr\_mean, Done in 1.257 min

Fitting fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.091 min

Fitting fmri\_msdl, Done in 0.075 min

Fitting fmri\_power\_2011, Done in 1.488 min

Transforming fmri\_basc064, Done in 0.087 min

Transforming fmri\_basc122, Done in 0.146 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.314 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.066 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.056 min

Transforming fmri\_power\_2011, Done in 0.338 min

Transforming fmri\_basc064, Done in 0.085 min

Transforming fmri\_basc122, Done in 0.144 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.313 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.066 min

Transforming fmri\_motions, Done in 0.027 min

Transforming fmri\_msdl, Done in 0.056 min

Transforming fmri\_power\_2011, Done in 0.337 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.027 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.044 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.031 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.001 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=100.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l2', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('logisticregression',

LogisticRegression(C=1.0, class\_weight=None, dual=False,

fit\_intercept=True, intercept\_scaling=1,

l1\_ratio=None, max\_iter=100,

multi\_class='warn', n\_jobs=None,

penalty='l1', random\_state=None,

solver='warn', tol=0.0001, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('randomforestclassifier',

RandomForestClassifier(bootstrap=True, class\_weight=None,

criterion='gini', max\_depth=20,

max\_features='auto',

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators='warn', n\_jobs=4,

oob\_score=False, random\_state=None,

verbose=0, warm\_start=False))],

verbose=False), Done in 0.002 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance',

max\_depth=3, max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=50,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.006 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('gradientboostingclassifier',

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.01,

loss='exponential', max\_depth=3,

max\_features=None,

max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0,

min\_impurity\_split=None,

min\_samples\_leaf=1,

min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0,

n\_estimators=80,

n\_iter\_no\_change=None,

presort='auto', random\_state=None,

subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0,

warm\_start=False))],

verbose=False), Done in 0.010 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.008 min

Fitting Pipeline(memory=None,

steps=[('standardscaler',

StandardScaler(copy=True, with\_mean=True, with\_std=True)),

('svc',

SVC(C=100.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1,

probability=True, random\_state=None, shrinking=True,

tol=0.001, verbose=False))],

verbose=False), Done in 0.009 min

Transforming fmri\_basc064, Done in 0.012 min

Transforming fmri\_basc122, Done in 0.021 min

Transforming fmri\_craddock\_scorr\_mean, Done in 0.045 min

Transforming fmri\_harvard\_oxford\_cort\_prob\_2mm, Done in 0.010 min

Transforming fmri\_motions, Done in 0.004 min

Transforming fmri\_msdl, Done in 0.008 min

Transforming fmri\_power\_2011, Done in 0.047 min

[c:\Users\kharu\OneDrive\Documents\Thesis\Working\_File\autism-challenge\submissions\ayoub\_ghriss\_original\feature\_extractor.py:97](file:///C:\Users\kharu\OneDrive\Documents\Thesis\Working_File\autism-challenge\submissions\ayoub_ghriss_original\feature_extractor.py:97): SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

lambda x: 0 if x == "M" else 1)

TP+FN 22 12

TP+FN 14 10

TP+FN 22 12

TP+FN 10 14

TP+FN 23 11

TP+FN 11 13

TP+FN 22 12

TP+FN 13 11

TP+FN 24 10

TP+FN 12 12

Results: {'ga': {1: {'overall': 0.6695652173913044, 'male': 0.6060606060606061, 'female': 0.7551020408163265}, 2: {'overall': 0.6347826086956522, 'male': 0.6666666666666666, 'female': 0.5918367346938775}, 3: {'overall': 0.6695652173913044, 'male': 0.696969696969697, 'female': 0.6326530612244898}, 4: {'overall': 0.6521739130434783, 'male': 0.6363636363636364, 'female': 0.673469387755102}, 5: {'overall': 0.7130434782608696, 'male': 0.7424242424242424, 'female': 0.673469387755102}}, 'eo': {1: {'overall': 0.06372549019607843}, 2: {'overall': 0.2303921568627451}, 3: {'overall': 0.21813725490196084}, 4: {'overall': 0.10539215686274517}, 5: {'overall': 0.20588235294117652}}}

lbg\_original

Downloading completed ...

TP+FN 25 9

TP+FN 16 8

TP+FN 23 11

TP+FN 12 12

TP+FN 24 10

TP+FN 15 9

TP+FN 23 11

TP+FN 13 11

TP+FN 22 12

TP+FN 16 8

Results: {'ga': {1: {'overall': 0.6695652173913044, 'male': 0.6666666666666666, 'female': 0.673469387755102}, 2: {'overall': 0.6956521739130435, 'male': 0.7272727272727273, 'female': 0.6530612244897959}, 3: {'overall': 0.6695652173913044, 'male': 0.696969696969697, 'female': 0.6326530612244898}, 4: {'overall': 0.6695652173913044, 'male': 0.6818181818181818, 'female': 0.6530612244897959}, 5: {'overall': 0.6695652173913044, 'male': 0.6666666666666666, 'female': 0.673469387755102}}, 'eo': {1: {'overall': 0.06862745098039225}, 2: {'overall': 0.17647058823529416}, 3: {'overall': 0.08088235294117652}, 4: {'overall': 0.13480392156862753}, 5: {'overall': -0.019607843137254832}}}

mk\_original

Downloading completed ...

TP+FN 22 12

TP+FN 13 11

TP+FN 21 13

TP+FN 11 13

TP+FN 26 8

TP+FN 14 10

TP+FN 20 14

TP+FN 11 13

TP+FN 17 17

TP+FN 7 17

Results: {'ga': {1: {'overall': 0.6782608695652174, 'male': 0.6666666666666666, 'female': 0.6938775510204082}, 2: {'overall': 0.591304347826087, 'male': 0.5757575757575758, 'female': 0.6122448979591837}, 3: {'overall': 0.6608695652173913, 'male': 0.6666666666666666, 'female': 0.6530612244897959}, 4: {'overall': 0.6260869565217392, 'male': 0.5909090909090909, 'female': 0.673469387755102}, 5: {'overall': 0.6086956521739131, 'male': 0.6363636363636364, 'female': 0.5714285714285714}}, 'eo': {1: {'overall': 0.10539215686274517}, 2: {'overall': 0.15931372549019612}, 3: {'overall': 0.18137254901960775}, 4: {'overall': 0.12990196078431376}, 5: {'overall': 0.20833333333333331}}}

nguigui\_original

Downloading completed ...

TP+FN 23 11

TP+FN 12 12

TP+FN 24 10

TP+FN 13 11

TP+FN 24 10

TP+FN 15 9

TP+FN 24 10

TP+FN 12 12

TP+FN 21 13

TP+FN 11 13

Results: {'ga': {1: {'overall': 0.6, 'male': 0.6060606060606061, 'female': 0.5918367346938775}, 2: {'overall': 0.6347826086956522, 'male': 0.6363636363636364, 'female': 0.6326530612244898}, 3: {'overall': 0.6347826086956522, 'male': 0.6363636363636364, 'female': 0.6326530612244898}, 4: {'overall': 0.6260869565217392, 'male': 0.6060606060606061, 'female': 0.6530612244897959}, 5: {'overall': 0.6, 'male': 0.6212121212121212, 'female': 0.5714285714285714}}, 'eo': {1: {'overall': 0.17647058823529416}, 2: {'overall': 0.1642156862745099}, 3: {'overall': 0.08088235294117652}, 4: {'overall': 0.20588235294117652}, 5: {'overall': 0.15931372549019612}}}

Slasnista\_original

Downloading completed ...

TP+FN 23 11

TP+FN 10 14

TP+FN 23 11

TP+FN 11 13

TP+FN 24 10

TP+FN 13 11

TP+FN 24 10

TP+FN 12 12

TP+FN 22 12

TP+FN 11 13

Results: {'ga': {1: {'overall': 0.6608695652173913, 'male': 0.6666666666666666, 'female': 0.6530612244897959}, 2: {'overall': 0.6608695652173913, 'male': 0.6818181818181818, 'female': 0.6326530612244898}, 3: {'overall': 0.6695652173913044, 'male': 0.7121212121212122, 'female': 0.6122448979591837}, 4: {'overall': 0.6695652173913044, 'male': 0.6818181818181818, 'female': 0.6530612244897959}, 5: {'overall': 0.6782608695652174, 'male': 0.7121212121212122, 'female': 0.6326530612244898}}, 'eo': {1: {'overall': 0.25980392156862747}, 2: {'overall': 0.21813725490196084}, 3: {'overall': 0.1642156862745099}, 4: {'overall': 0.20588235294117652}, 5: {'overall': 0.18872549019607848}}}

vzantedeschi\_original

Downloading completed ...

fit feature extractor

time 68.81292963027954 s

transform data

time 63.84959173202515 s

transform data

time 9.204598665237427 s

fit feature extractor

time 69.95218539237976 s

transform data

time 65.0993218421936 s

transform data

time 9.176594257354736 s

fit feature extractor

time 69.89414143562317 s

transform data

time 64.73574709892273 s

transform data

time 9.23560357093811 s

fit feature extractor

time 69.54808187484741 s

transform data

time 64.73302698135376 s

transform data

time 9.17759370803833 s

fit feature extractor

time 69.74311590194702 s

transform data

time 64.60174298286438 s

transform data

time 9.151088953018188 s

TP+FN 15 19

TP+FN 10 14

TP+FN 17 17

TP+FN 9 15

TP+FN 16 18

TP+FN 9 15

TP+FN 16 18

TP+FN 11 13

TP+FN 17 17

TP+FN 10 14

Results: {'ga': {1: {'overall': 0.591304347826087, 'male': 0.5454545454545454, 'female': 0.6530612244897959}, 2: {'overall': 0.6086956521739131, 'male': 0.6060606060606061, 'female': 0.6122448979591837}, 3: {'overall': 0.6, 'male': 0.5757575757575758, 'female': 0.6326530612244898}, 4: {'overall': 0.6260869565217392, 'male': 0.5757575757575758, 'female': 0.6938775510204082}, 5: {'overall': 0.6086956521739131, 'male': 0.5909090909090909, 'female': 0.6326530612244898}}, 'eo': {1: {'overall': 0.024509803921568596}, 2: {'overall': 0.125}, 3: {'overall': 0.09558823529411764}, 4: {'overall': 0.012254901960784326}, 5: {'overall': 0.08333333333333331}}}

wwwwmmmm\_original

Downloading completed ...

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 37.9s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 20.3s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 13.0s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 2.2min finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 36.7s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 20.5s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 13.3s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 2.2min finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 36.6s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 20.5s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 13.2s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 2.2min finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 36.7s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 20.4s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 13.3s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 2.2min finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 36.8s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 20.6s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 13.2s finished

Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 45 out of 45 | elapsed: 2.2min finished

TP+FN 26 8

TP+FN 15 9

TP+FN 27 7

TP+FN 16 8

TP+FN 29 5

TP+FN 15 9

TP+FN 26 8

TP+FN 14 10

TP+FN 26 8

TP+FN 16 8

Results: {'ga': {1: {'overall': 0.591304347826087, 'male': 0.5909090909090909, 'female': 0.5918367346938775}, 2: {'overall': 0.6695652173913044, 'male': 0.6818181818181818, 'female': 0.6530612244897959}, 3: {'overall': 0.6347826086956522, 'male': 0.7121212121212122, 'female': 0.5306122448979592}, 4: {'overall': 0.6, 'male': 0.6060606060606061, 'female': 0.5918367346938775}, 5: {'overall': 0.6869565217391305, 'male': 0.6818181818181818, 'female': 0.6938775510204082}}, 'eo': {1: {'overall': 0.13970588235294112}, 2: {'overall': 0.12745098039215685}, 3: {'overall': 0.2279411764705882}, 4: {'overall': 0.18137254901960775}, 5: {'overall': 0.0980392156862745}}}