Imports

In [1]:
```
import os
import sys
sys.path.append('/home/rcendre/classification')

import itertools
import pandas
import webbrowser
from pathlib import Path
import matplotlib.pyplot as plt
from misvm import SIL, MISVM
from numpy import array, logspace
from scipy.stats import randint,uniform
from sklearn.decomposition import PCA
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.manifold import TSNE
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, Rob
ustScaler, StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import davies_bouldin_score
from toolbox.classification.common import Data, Folds, IO, Tools
from toolbox.classification.parameters import Dermatology, Settin
gs
from toolbox.models.models import CustomMIL, MultimodalClassifier
from toolbox.models.builtin import Applications
from toolbox.IO import dermatology
from sklearn.metrics import f1_score
from toolbox.transforms.common import PredictorTransform, Flatten
Transform, LinearTransform
from toolbox.transforms.labels import OrderedEncoder
from toolbox.transforms.images import DistributionImageTransform,
DWTImageTransform, FourierImageTransform, HaralickImageTransform,
SpatialImageTransform
from toolbox.views.common import Views, ViewsTools
from toolbox.views.images import ImagesViews
import warnings
warnings.filterwarnings('ignore')
```

```
Using TensorFlow backend.
```

Parameters

In [2]:
```
# Advanced parameters
data_type='Full'
prefix = 'Cumulative'
validation = 10
settings = Settings.get_default_dermatology()
```

Photography

In [3]:
```
p_inputs = IO.load('Photography.pickle')
```

In [4]:
```
p_inputs['Extractor'] = p_inputs['ResNetAvg']

# Pipeline
model = Pipeline([('scale', MinMaxScaler()), ('clf', LinearSVC(cl
ass_weight='balanced'))])
model_params = {'clf__C': logspace(-2, 3, 6).tolist()}
```

In [5]:
```
low_folds = Tools.generate_folds(([1, 2, 3, 4, 5, 6, 7, 8], [9]),
validation)
Tools.evaluate(p_inputs, {'datum': 'Extractor', 'label_encode': '
LesionEncode'}, model, 'Prediction', folds=low_folds, distributio
n=model_params)
```

```
Evaluation achieved!
```

In [6]:
```
from IPython.display import HTML
from IPython.display import display

diagnosis_encoder = OrderedEncoder().fit(['Benign', 'Malignant'])

# ROC Curve
ViewsTools.plot_size((8,8))

name = f'Prediction'
# Label
display(HTML(ViewsTools.dataframe_renderer([Views.report(p_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Photography'])))
```

Photography

	precision	recall	f1-score	support
Benign	0.45±0.15	0.51±0.17	0.48±0.14	86.00±2.01
Malignant	0.67±0.11	0.61±0.17	0.64±0.13	137.00±2.00
accuracy	0.57±0.10	0.57±0.10	0.57±0.10	0.57±0.10
macro avg	0.56±0.10	0.56±0.10	0.56±0.10	223.00±0.46
weighted avg	0.58±0.10	0.57±0.10	0.58±0.11	223.00±0.46

Dermoscopy

In [7]:
```
d_inputs = IO.load('Dermoscopy.pickle')
d_inputs['Extractor'] = d_inputs['ResNetAvg']
```

In [8]:
```
dp_inputs = IO.load('Photography.pickle')
dp_inputs['Extractor'] = dp_inputs['ResNetAvg']
d_inputs = pandas.concat([dp_inputs, d_inputs], axis=0)
```

In [9]:
```
all_image = [True] * len(d_inputs.index)
single_image = d_inputs['Modality'] == 'Dermoscopy'
Data.build_bags(d_inputs, single_image, 'ID_Lesion', all_image, '
ID_Lesion', 'Extractor')
d_inputs = d_inputs[single_image].reset_index()
Tools.transform(d_inputs, {'datum': 'Extractor'}, FlattenTransfor
m(), 'Flat')
```

In [10]:
```
low_folds = Tools.generate_folds(([1, 2, 3, 4, 5, 6, 7, 8], [9]),
validation)

# Pipeline
model = Pipeline([('scale', MinMaxScaler()), ('clf', LinearSVC(cl
ass_weight='balanced'))])
model_params = {'clf__C': logspace(-2, 3, 6).tolist()}

Tools.evaluate(d_inputs, {'datum': 'Flat', 'label_encode': 'Lesio
nEncode'}, model, 'Prediction', folds=low_folds, distribution=mod
el_params)
```

```
Evaluation achieved!
```

```
In [11]: from IPython.display import HTML
         from IPython.display import display

         diagnosis_encoder = OrderedEncoder().fit(['Benign', 'Malignant'])

         # ROC Curve
         ViewsTools.plot_size((8,8))

         name = f'Prediction'
         # Label
         display(HTML(ViewsTools.dataframe_renderer([Views.report(d_input
         s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
         der)],

         title=[f'Dermoscopy'])))
```

Dermoscopy

	precision	recall	f1-score	support
Benign	0.49±0.16	0.52±0.24	0.51±0.19	86.00±2.01
Malignant	0.69±0.13	0.66±0.07	0.68±0.08	137.00±2.00
accuracy	0.61±0.10	0.61±0.10	0.61±0.10	0.61±0.10
macro avg	0.59±0.12	0.59±0.12	0.59±0.12	223.00±0.46
weighted avg	0.61±0.12	0.61±0.10	0.61±0.11	223.00±0.46

Microscopy

```
In [12]: m_inputs = IO.load('Microscopy.pickle')
```

Merge

```
In [13]: extractor = 'ResNetAvg'
         low_folds = Tools.generate_folds(([1, 2, 3, 4, 5, 6, 7, 8], [9]),
         validation)
```

```
In [14]: all_image = [True] * len(m_inputs.index)
         single_image = m_inputs['ID_Image'] == '0M'
         Data.build_bags(m_inputs, single_image, 'ID_Lesion', all_image, '
         ID_Lesion', extractor)
         m_inputs = m_inputs[single_image].reset_index()
```

In [15]:
```
Tools.transform(m_inputs, {'datum': extractor}, LinearTransform(m
ethod='average'), 'Avg')
Tools.transform(m_inputs, {'datum': extractor}, LinearTransform(m
ethod='max'), 'Max')
Tools.transform(m_inputs, {'datum': extractor}, LinearTransform(m
ethod=4), 'Norm4')
Tools.transform(m_inputs, {'datum': extractor}, LinearTransform(m
ethod=6), 'Norm6')
Tools.transform(m_inputs, {'datum': extractor}, LinearTransform(m
ethod=8), 'Norm8')
```

In [16]:
```
# Pipeline
model = Pipeline([('scale', MinMaxScaler()), ('clf', LinearSVC(cl
ass_weight='balanced'))])
model_params = {'clf__C': logspace(-2, 3, 6).tolist()}

Tools.fit_predict(m_inputs, {'datum': 'Avg', 'label_encode': 'Les
ionEncode'}, model, 'PredictionAvg', folds=low_folds, distributio
n=model_params)
Tools.fit_predict(m_inputs, {'datum': 'Max', 'label_encode': 'Les
ionEncode'}, model, 'PredictionMax', folds=low_folds, distributio
n=model_params)
Tools.fit_predict(m_inputs, {'datum': 'Norm4', 'label_encode': 'L
esionEncode'}, model, 'PredictionNorm4', folds=low_folds, distrib
ution=model_params)
Tools.fit_predict(m_inputs, {'datum': 'Norm6', 'label_encode': 'L
esionEncode'}, model, 'PredictionNorm6', folds=low_folds, distrib
ution=model_params)
Tools.fit_predict(m_inputs, {'datum': 'Norm8', 'label_encode': 'L
esionEncode'}, model, 'PredictionNorm8', folds=low_folds, distrib
ution=model_params)
```

In [17]:
```
from IPython.display import HTML
from IPython.display import display

diagnosis_encoder = OrderedEncoder().fit(['Benign', 'Malignant'])

# ROC Curve
ViewsTools.plot_size((8,8))

name = f'PredictionAvg'
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopy Avg'])))

name = f'PredictionMax'
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopy Max'])))

name = f'PredictionNorm4'
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopy Norm4'])))

name = f'PredictionNorm6'
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopy Norm6'])))

name = f'PredictionNorm8'
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopy Norm8'])))
```

Microscopy Avg

	precision	recall	f1-score	support
Benign	0.70±0.14	0.77±0.12	0.73±0.11	86.00±2.01
Malignant	0.84±0.08	0.80±0.10	0.82±0.06	137.00±2.00
accuracy	0.78±0.07	0.78±0.07	0.78±0.07	0.78±0.07
macro avg	0.77±0.07	0.78±0.07	0.78±0.08	223.00±0.46
weighted avg	0.79±0.06	0.78±0.07	0.79±0.06	223.00±0.46

Microscopy Max

	precision	recall	f1-score	support
Benign	0.73±0.16	0.72±0.14	0.73±0.12	86.00±2.01
Malignant	0.83±0.10	0.83±0.11	0.83±0.07	137.00±2.00
accuracy	0.79±0.08	0.79±0.08	0.79±0.08	0.79±0.08
macro avg	0.78±0.09	0.78±0.08	0.78±0.08	223.00±0.46
weighted avg	0.79±0.07	0.79±0.08	0.79±0.08	223.00±0.46

Microscopy Norm4

	precision	recall	f1-score	support
Benign	0.75±0.14	0.72±0.15	0.73±0.11	86.00±2.01
Malignant	0.83±0.11	0.85±0.07	0.84±0.05	137.00±2.00
accuracy	0.80±0.07	0.80±0.07	0.80±0.07	0.80±0.07
macro avg	0.79±0.08	0.78±0.08	0.79±0.08	223.00±0.46
weighted avg	0.80±0.06	0.80±0.07	0.80±0.07	223.00±0.46

Microscopy Norm6

	precision	recall	f1-score	support
Benign	0.76±0.14	0.76±0.14	0.76±0.12	86.00±2.01
Malignant	0.85±0.09	0.85±0.07	0.85±0.05	137.00±2.00
accuracy	0.81±0.07	0.81±0.07	0.81±0.07	0.81±0.07
macro avg	0.80±0.08	0.80±0.08	0.80±0.08	223.00±0.46
weighted avg	0.81±0.06	0.81±0.07	0.81±0.07	223.00±0.46

Microscopy Norm8

	precision	recall	f1-score	support
Benign	0.74±0.14	0.74±0.14	0.74±0.11	86.00±2.01
Malignant	0.84±0.10	0.84±0.08	0.84±0.05	137.00±2.00
accuracy	0.80±0.07	0.80±0.07	0.80±0.07	0.80±0.07
macro avg	0.79±0.08	0.79±0.07	0.79±0.08	223.00±0.46
weighted avg	0.80±0.06	0.80±0.07	0.80±0.06	223.00±0.46

Predicting on microscopy

In [18]:
```
mp_inputs = IO.load('Photography.pickle')
mp_inputs['Extractor'] = mp_inputs['ResNetAvg']
```

In [19]:
```
md_inputs = IO.load('Dermoscopy.pickle')
md_inputs['Extractor'] = md_inputs['ResNetAvg']
```

In [20]:
```
m_inputs['Extractor'] = m_inputs['Avg']
m_inputs = pandas.concat([mp_inputs, md_inputs, m_inputs], axis=
0)
```

In [21]:
```
all_image = [True] * len(m_inputs.index)
single_image = m_inputs['Modality'] == 'Microscopy'
Data.build_bags(m_inputs, single_image, 'ID_Lesion', all_image, '
ID_Lesion', 'Extractor')
m_inputs = m_inputs[single_image].reset_index()
Tools.transform(m_inputs, {'datum': 'Extractor'}, FlattenTransfor
m(), 'Flat')
```

In [22]:
```
# Pipeline
model = Pipeline([('scale', MinMaxScaler()), ('clf', LinearSVC(cl
ass_weight='balanced'))])
model_params = {'clf__C': logspace(-2, 3, 6).tolist()}
```

In [23]:
```
low_folds = Tools.generate_folds(([1, 2, 3, 4, 5, 6, 7, 8], [9]),
validation)
Tools.evaluate(m_inputs, {'datum': 'Flat', 'label_encode': 'Lesio
nEncode'}, model, 'Prediction', folds=low_folds, distribution=mod
el_params)
```

```
Evaluation achieved!
```

In [24]:
```
from IPython.display import HTML
from IPython.display import display

diagnosis_encoder = OrderedEncoder().fit(['Benign', 'Malignant'])

# ROC Curve
ViewsTools.plot_size((8,8))

name = f'Prediction'
# Label
display(HTML(ViewsTools.dataframe_renderer([Views.report(m_input
s, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Microscopie'])))
```

Microscopie

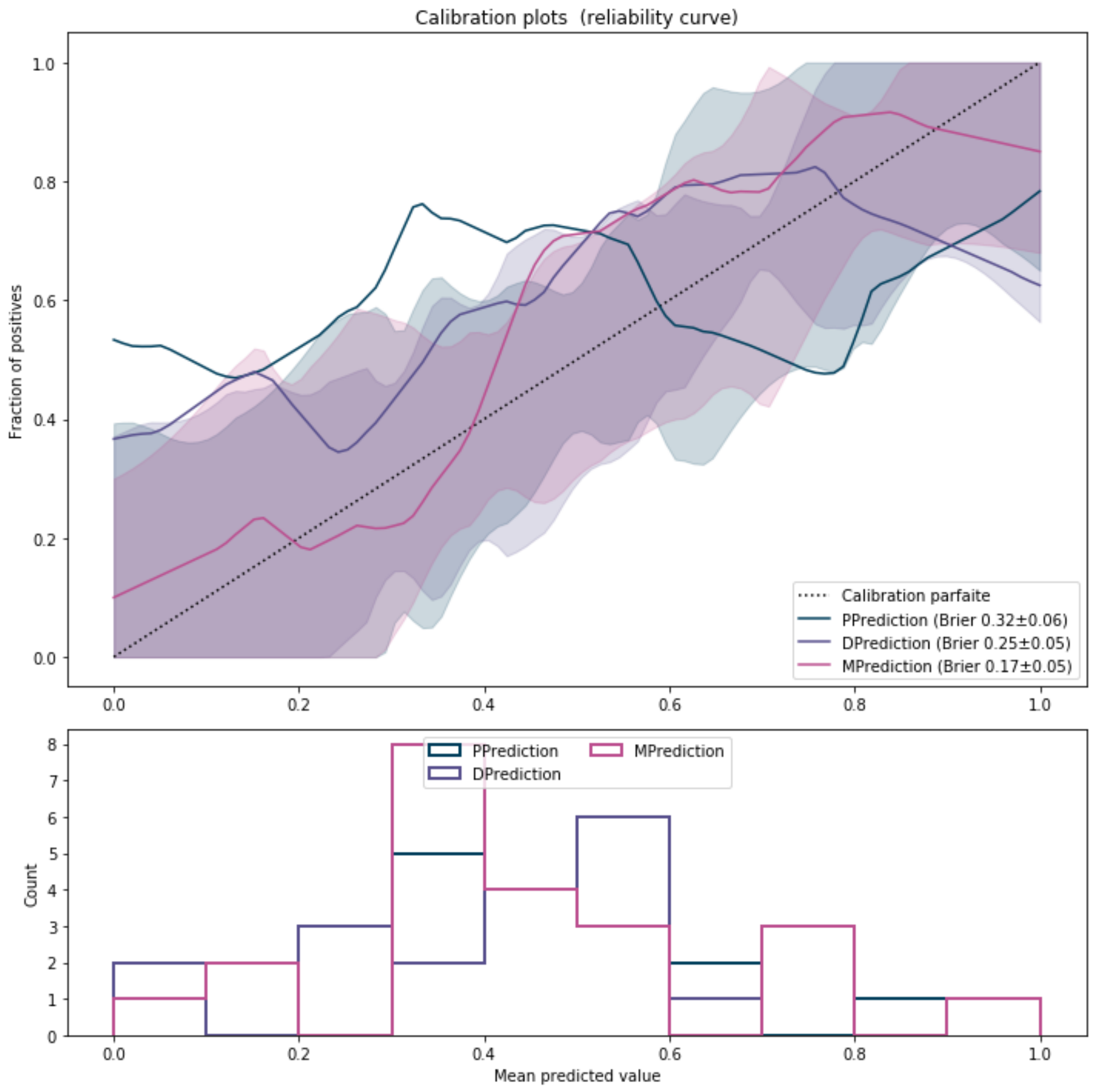
	precision	recall	f1-score	support
Benign	0.71±0.16	0.67±0.20	0.69±0.16	86.00±2.01
Malignant	0.80±0.11	0.82±0.08	0.81±0.07	137.00±2.00
accuracy	0.77±0.10	0.77±0.10	0.77±0.10	0.77±0.10
macro avg	0.75±0.10	0.75±0.11	0.75±0.11	223.00±0.46
weighted avg	0.77±0.09	0.77±0.10	0.77±0.09	223.00±0.46

Calibration

In [25]:
```
m_inputs['Prediction_Score']
```

Out[25]:
```
0       [0.40371908087811403, 0.596280919121886]
1       [0.19951326503065503, 0.800486734969345]
2      [0.5033853343244741, 0.49661466567552587]
3                                     [1.0, 0.0]
4      [0.6802568556876807, 0.31974314431231926]
                         ...
218     [0.7123633805427707, 0.2876366194572293]
219                                   [0.0, 1.0]
220     [0.4673873685860551, 0.5326126314139449]
221     [0.5389098685857334, 0.4610901314142665]
222    [0.6710724474446443, 0.32892755255535566]
Name: Prediction_Score, Length: 223, dtype: object
```

In [26]:
```
p_inputs['PPrediction_Prediction'] = p_inputs['Prediction_Predict
ion']
p_inputs['PPrediction_Probability'] = p_inputs['Prediction_Score
']
p_inputs['DPrediction_Prediction'] = d_inputs['Prediction_Predict
ion']
p_inputs['DPrediction_Probability'] = d_inputs['Prediction_Score
']
p_inputs['MPrediction_Prediction'] = m_inputs['Prediction_Predict
ion']
p_inputs['MPrediction_Probability'] = m_inputs['Prediction_Score
']

figure = Views.reliability_curve(p_inputs, 'LesionEncode', ['PPre
diction','DPrediction','MPrediction'])
figure.savefig(f'{prefix}.svg')
figure.show()
```



Fusion

In [27]:
```
inputs = pandas.concat([p_inputs, d_inputs, m_inputs], axis=0)
inputs = inputs.reset_index(drop=True)
 # Save
IO.save(inputs, f'Low_{prefix}.pickle')
```

Cumulative

In [28]:
```
inputs = IO.load(f'Low_{prefix}.pickle')
```

In [29]:
```
all_image = [True] * len(inputs.index)
single_image = inputs['Modality'] == 'Photography'
Data.build_bags(inputs, single_image, 'ID_Lesion', all_image, 'ID
_Lesion', f'Prediction_{Tools.SCORE}')
inputs = inputs[single_image].reset_index(drop=True)
```

In [30]:
```
high_folds = Tools.generate_folds(([1], [2]), validation)
modality = MultimodalClassifier(method='modality', metric=f1_scor
e)
modality_rev = MultimodalClassifier(method='modality', metric=f1_
score, ordered=False)
classe = MultimodalClassifier(method='modality_class', metric=f1_
score)
classe_rev = MultimodalClassifier(method='modality_class', metric
=f1_score, ordered=False)

modality_ones = MultimodalClassifier(method='modality', metric=f1
_score, from_zero=False)
modality_rev_ones = MultimodalClassifier(method='modality', metri
c=f1_score, ordered=False, from_zero=False)
classe_ones = MultimodalClassifier(method='modality_class', metri
c=f1_score, from_zero=False)
classe_rev_ones = MultimodalClassifier(method='modality_class', m
etric=f1_score, ordered=False, from_zero=False)
```

In [31]:
```
inputs[f'Prediction_{Tools.SCORE}']
```

Out[31]:
```
0      [[0.22616350895752202, 0.773836491042478], [0....
1      [[0.44718931553155583, 0.5528106844684442], [0...
2      [[0.8826321862195973, 0.11736781378040274], [0...
3      [[0.8231515952291575, 0.17684840477084257], [0...
4      [[0.7268276328472221, 0.273172367152778], [0.6...
                              ...
218    [[0.7870267096489789, 0.21297329035102108], [0...
219    [[0.41640379503068503, 0.583596204969315], [0....
220    [[1.0, 0.0], [0.8898923940151052, 0.1101076059...
221    [[0.8282150478354442, 0.17178495216455586], [0...
222    [[0.600715821563629, 0.39928417843637104], [0....
Name: Prediction_Score, Length: 223, dtype: object
```

In [32]:
```
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, modality, 'simple_increase_increase
')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, modality_rev, 'simple_increase_decr
ease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, classe, 'double_increase_increase')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, classe_rev, 'double_increase_decrea
se')

Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, modality_ones, 'simple_decrease_inc
rease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, modality_rev_ones, 'simple_decrease
_decrease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, classe_ones, 'double_decrease_incre
ase')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.SCORE}', 'la
bel_encode': 'LesionEncode'}, classe_rev_ones, 'double_decrease_d
ecrease')
IO.save(inputs, f'High_{prefix}.pickle')
```

```
Evaluation achieved!
```

Score

In [33]:
```
from IPython.display import HTML
from IPython.display import display

inputs = IO.load(f'High_{prefix}.pickle')
diagnosis_encoder = OrderedEncoder().fit(['Benign', 'Malignant'])

# ROC Curve
ViewsTools.plot_size((8,8))

name = f'simple_increase_increase'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'simple_increase_decrease'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'double_increase_increase'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'double_increase_decrease'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'simple_decrease_increase'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
```

```
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'simple_decrease_decrease'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'double_decrease_increase'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'double_decrease_decrease'
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs,
{'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encode
r)],

title=[f'Test - {name}'])))
figure = Views.steps_visualization(inputs, {'label_encode': 'Lesi
onEncode', 'eval': name})
figure.savefig(f'{prefix}_{name}.svg')
figure.show()

name = f'double_increase_decrease'
inputs_lm = inputs[(inputs['Diagnosis']=='LM/LMM')|(inputs['Binar
y_Diagnosis']=='Benign')]
display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs_l
m, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_enco
der)],

title=[f'Test - LM {name}'])))
```

Test - simple_increase_increase

	precision	recall	f1-score	support
Benign	0.61±0.13	0.79±0.11	0.69±0.11	86.00±2.01
Malignant	0.84±0.10	0.68±0.13	0.75±0.10	137.00±2.00
accuracy	0.72±0.10	0.72±0.10	0.72±0.10	0.72±0.10
macro avg	0.72±0.08	0.73±0.09	0.72±0.10	223.00±0.46
weighted avg	0.75±0.08	0.72±0.10	0.73±0.09	223.00±0.46

Test - simple_increase_decrease

	precision	recall	f1-score	support
Benign	0.57±0.13	0.79±0.11	0.66±0.11	86.00±2.01
Malignant	0.83±0.11	0.63±0.12	0.71±0.10	137.00±2.00
accuracy	0.69±0.09	0.69±0.09	0.69±0.09	0.69±0.09
macro avg	0.70±0.08	0.71±0.08	0.69±0.10	223.00±0.46
weighted avg	0.73±0.08	0.69±0.09	0.69±0.09	223.00±0.46

Test - double_increase_increase

	precision	recall	f1-score	support
Benign	0.41±0.40	0.15±0.18	0.22±0.15	86.00±2.01
Malignant	0.62±0.10	0.86±0.12	0.72±0.05	137.00±2.00
accuracy	0.59±0.07	0.59±0.07	0.59±0.07	0.59±0.07
macro avg	0.51±0.21	0.51±0.06	0.47±0.09	223.00±0.46
weighted avg	0.54±0.19	0.59±0.07	0.53±0.11	223.00±0.46

Test - double_increase_decrease

	precision	recall	f1-score	support
Benign	0.41±0.40	0.14±0.13	0.21±0.15	86.00±2.01
Malignant	0.62±0.10	0.88±0.09	0.73±0.05	137.00±2.00
accuracy	0.59±0.07	0.59±0.07	0.59±0.07	0.59±0.07
macro avg	0.52±0.21	0.51±0.06	0.47±0.09	223.00±0.46
weighted avg	0.54±0.19	0.59±0.07	0.53±0.11	223.00±0.46

Test - simple_decrease_increase

	precision	recall	f1-score	support
Benign	0.64±0.14	0.80±0.11	0.71±0.11	86.00±2.01
Malignant	0.85±0.10	0.72±0.14	0.78±0.11	137.00±2.00
accuracy	0.75±0.10	0.75±0.10	0.75±0.10	0.75±0.10
macro avg	0.75±0.09	0.76±0.09	0.74±0.10	223.00±0.46
weighted avg	0.77±0.08	0.75±0.10	0.75±0.10	223.00±0.46

Test - simple_decrease_decrease

	precision	recall	f1-score	support
Benign	0.63±0.13	0.80±0.11	0.70±0.11	86.00±2.01
Malignant	0.85±0.10	0.70±0.14	0.77±0.11	137.00±2.00
accuracy	0.74±0.10	0.74±0.10	0.74±0.10	0.74±0.10
macro avg	0.74±0.08	0.75±0.09	0.74±0.10	223.00±0.46
weighted avg	0.76±0.08	0.74±0.10	0.74±0.10	223.00±0.46

Test - double_decrease_increase

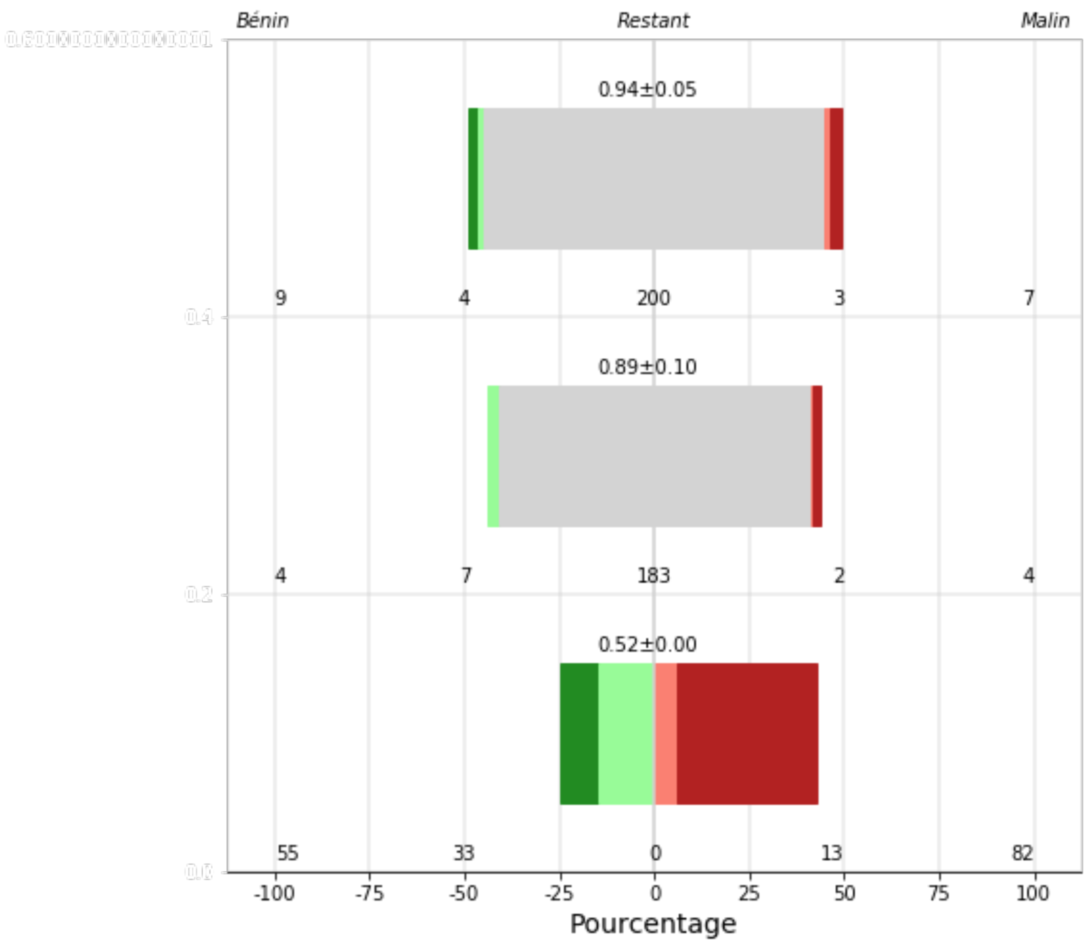
	precision	recall	f1-score	support
Benign	0.77±0.20	0.62±0.17	0.68±0.16	86.00±2.01
Malignant	0.79±0.11	0.88±0.10	0.83±0.09	137.00±2.00
accuracy	0.78±0.11	0.78±0.11	0.78±0.11	0.78±0.11
macro avg	0.78±0.12	0.75±0.11	0.76±0.12	223.00±0.46
weighted avg	0.78±0.11	0.78±0.11	0.77±0.11	223.00±0.46

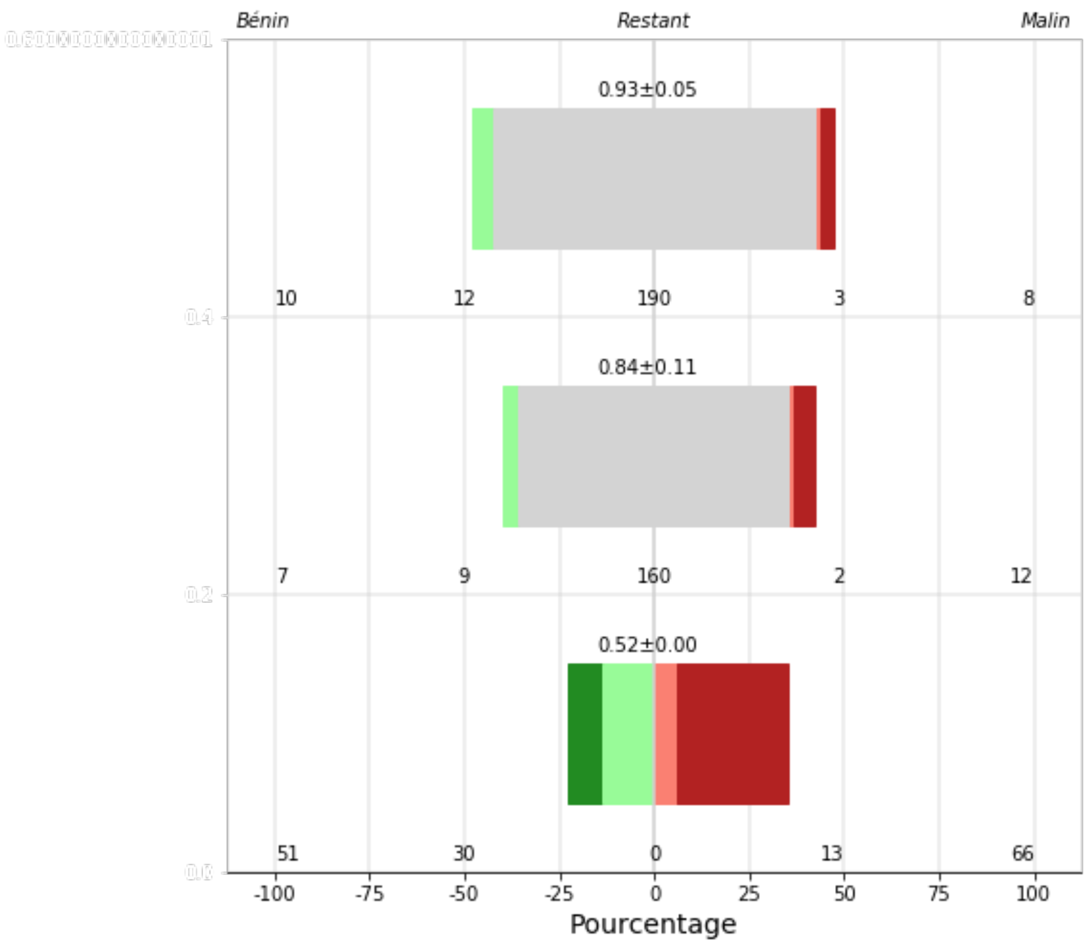
Test - double_decrease_decrease

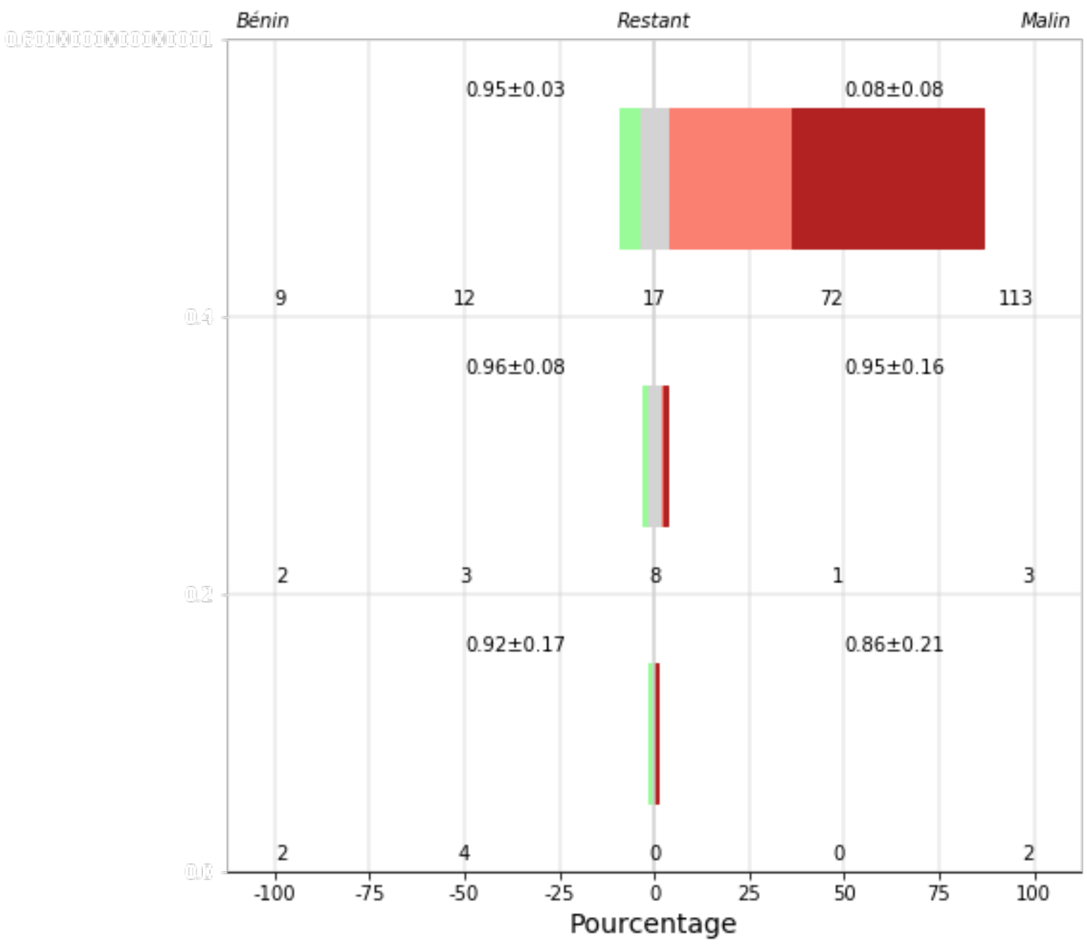
	precision	recall	f1-score	support
Benign	0.77±0.20	0.62±0.17	0.68±0.16	86.00±2.01
Malignant	0.79±0.11	0.88±0.10	0.83±0.09	137.00±2.00
accuracy	0.78±0.11	0.78±0.11	0.78±0.11	0.78±0.11
macro avg	0.78±0.12	0.75±0.11	0.76±0.12	223.00±0.46
weighted avg	0.78±0.11	0.78±0.11	0.77±0.11	223.00±0.46

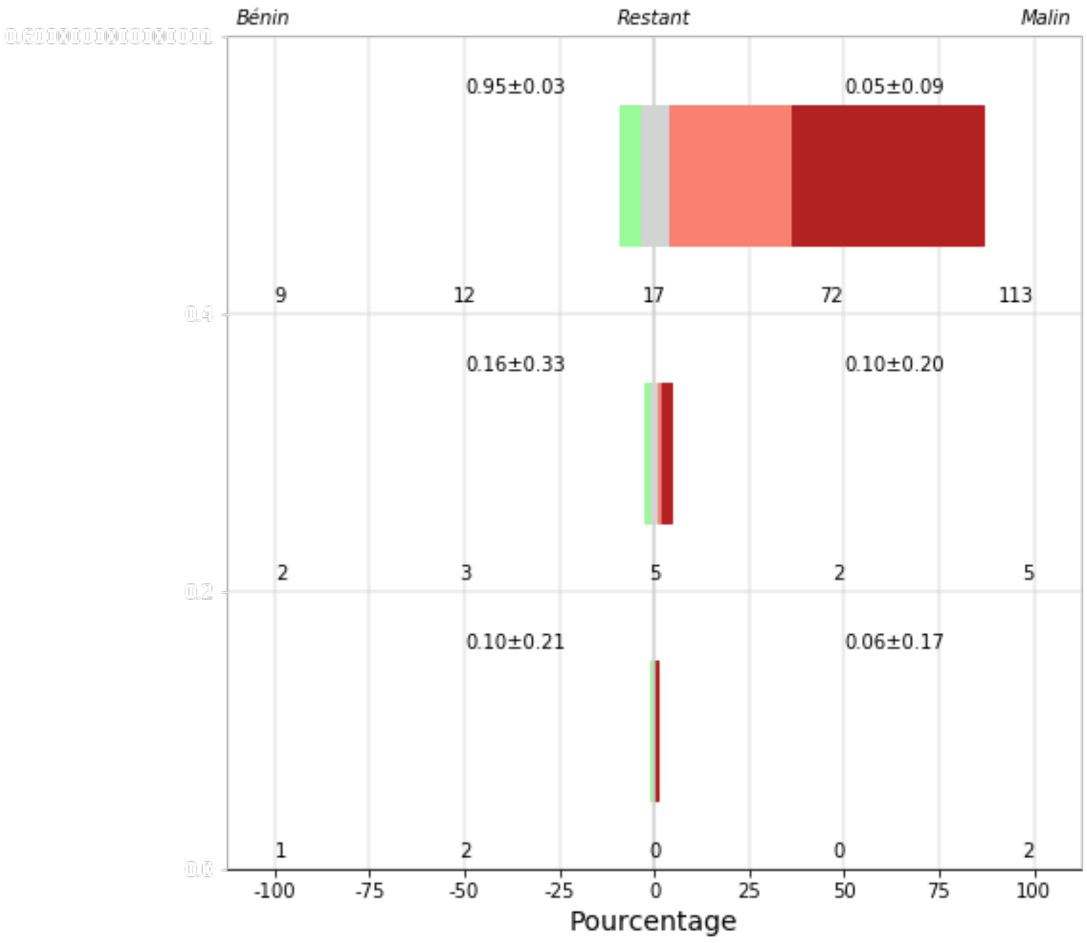
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/anaconda3/envs/PythonGPU/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Diagnosis'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-33-8bfd895f6529> in <module>
     65 
     66 name = f'double_increase_decrease'
---> 67 inputs_lm = inputs[(inputs['Diagnosis']=='LM/LMM')|(inputs['Binary_Diagnosis']=='Benign')]
     68 display(HTML(ViewsTools.dataframe_renderer([Views.report(inputs_lm, {'label_encode': 'LesionEncode', 'eval': name}, diagnosis_encoder)],
     69 title=[f'Test - LM {name}']))

~/anaconda3/envs/PythonGPU/lib/python3.7/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2993             if self.columns.nlevels > 1:
   2994                 return self._getitem_multilevel(key)
-> 2995             indexer = self.columns.get_loc(key)
   2996             if is_integer(indexer):
   2997                 indexer = [indexer]

~/anaconda3/envs/PythonGPU/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc
()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtabl
e.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtabl
e.PyObjectHashTable.get_item()

KeyError: 'Diagnosis'
```

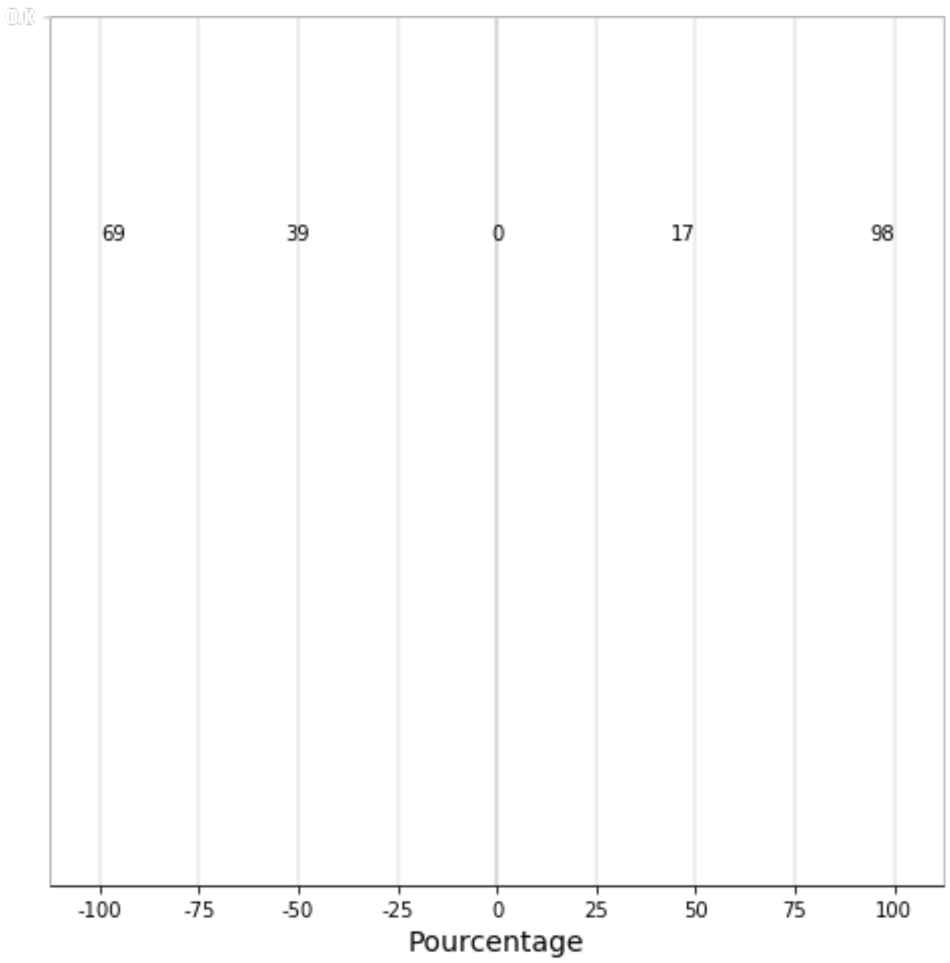


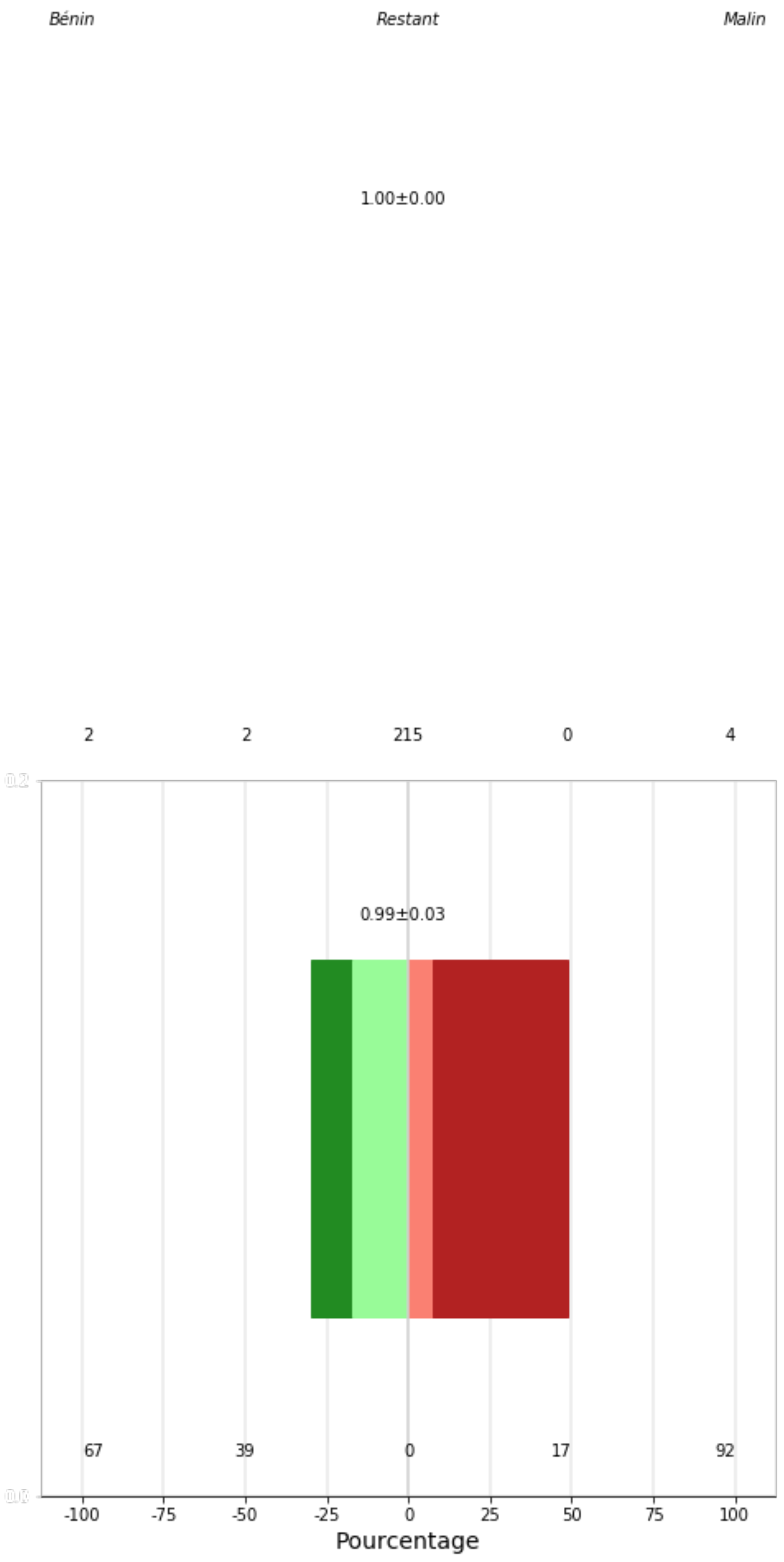

Bénin
Restant
Malin
0.93±0.05
10
12
190
3
8
0.84±0.11
7
9
160
2
12
0.52±0.00
51
30
0
13
66
-100
-75
-50
-25
0
25
50
75
100
Pourcentage

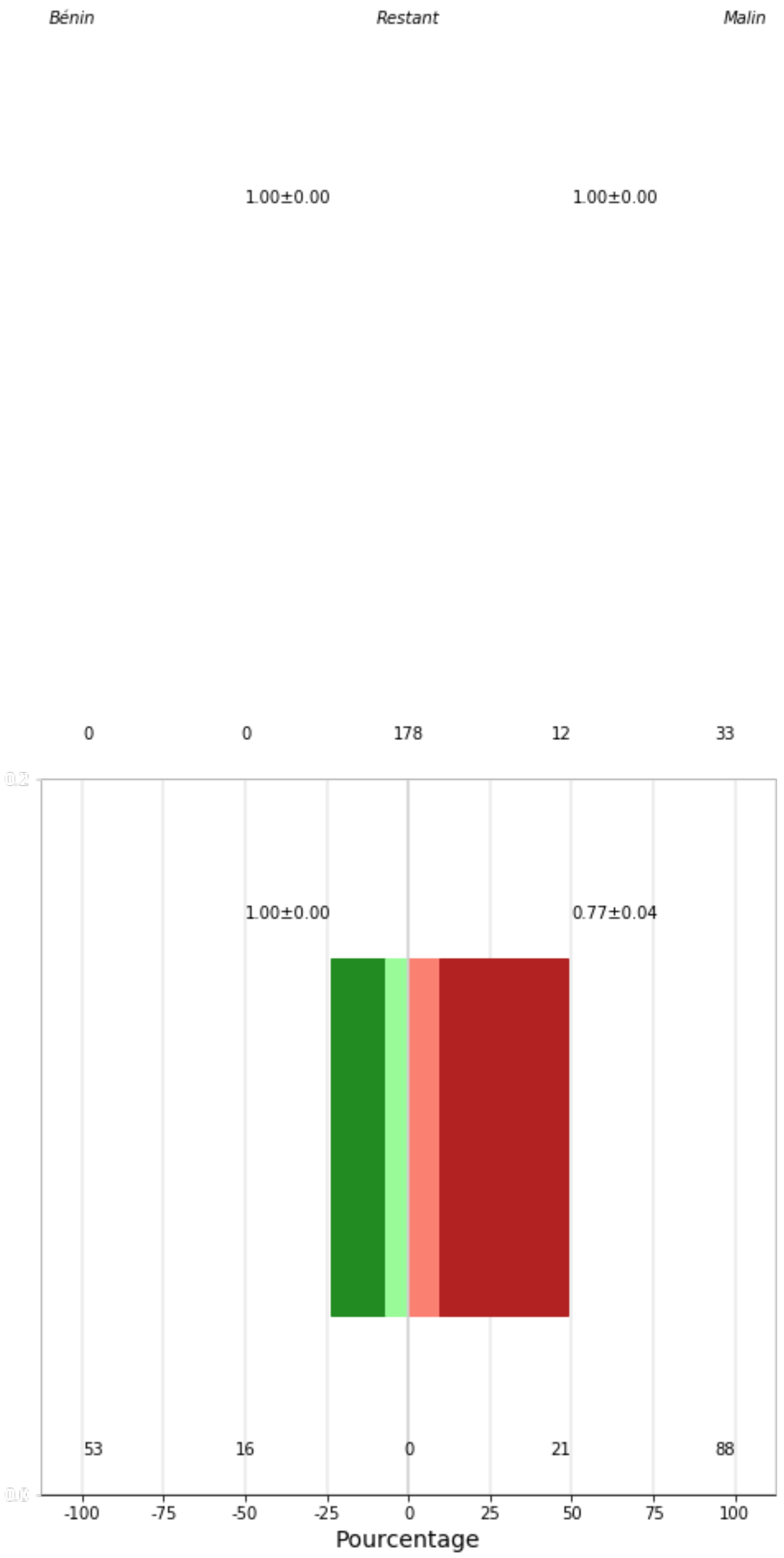

Bénin
Restant
Malin
0.95±0.03
0.08±0.08
9
12
17
72
113
0.96±0.08
0.95±0.16
2
3
8
1
3
0.92±0.17
0.86±0.21
2
4
0
0
2
-100
-75
-50
-25
0
25
50
75
100
Pourcentage

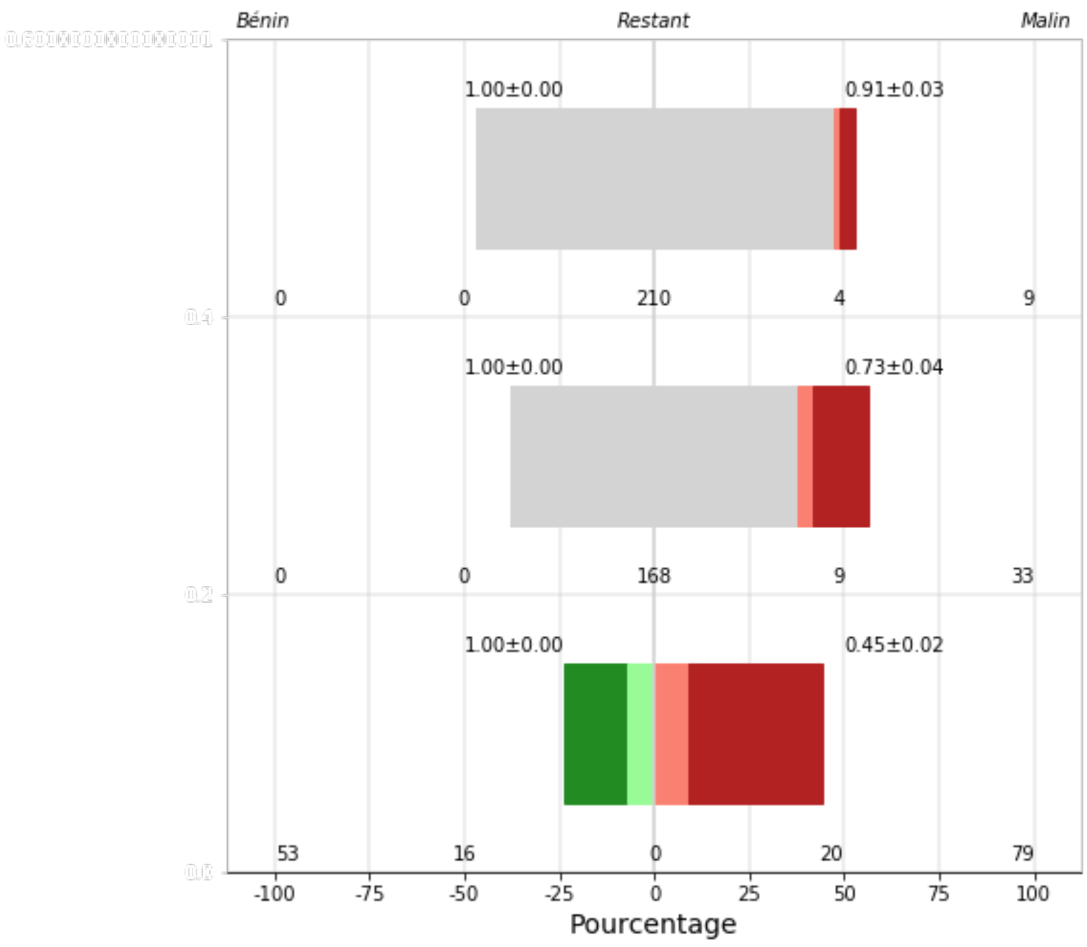


Bénin
Restant
Malin
0.95±0.03
0.05±0.09
9
12
17
72
113
0.16±0.33
0.10±0.20
2
3
5
2
5
0.10±0.21
0.06±0.17
1
2
0
0
2
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


0.0
69
39
0
17
98
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
1.00±0.00
2
2
215
0
4
0.2
0.99±0.03
67
39
0
17
92
0.0
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
1.00±0.00
1.00±0.00
0
0
178
12
33
0.2
1.00±0.00
0.77±0.04
53
16
0
21
88
0.0
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.6000000000000001
1.00±0.00
0.91±0.03
0
0
210
4
9
0.4
1.00±0.00
0.73±0.04
0
0
168
9
33
0.2
1.00±0.00
0.45±0.02
53
16
0
20
79
0.0
-100
-75
-50
-25
0
25
50
75
100
Pourcentage