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_Isotonic'
validation = 10
settings = Settings.get_default_dermatology()
```

Photography

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

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

# SVM Linear
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, calibrate='isotonic')
```

```
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	0.44±0.26	0.21±0.26	0.28±0.24	86.00±2.01
Malignant	0.63±0.14	0.83±0.10	0.71±0.08	137.00±2.00
accuracy	0.59±0.11	0.59±0.11	0.59±0.11	0.59±0.11
macro avg	0.53±0.18	0.52±0.11	0.50±0.14	223.00±0.46
weighted avg	0.55±0.18	0.59±0.11	0.55±0.15	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)

# SVM Linear
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, calibrate='isotonic')
```

```
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	0.56±0.19	0.44±0.18	0.49±0.16	86.00±2.01
Malignant	0.69±0.10	0.78±0.10	0.73±0.07	137.00±2.00
accuracy	0.65±0.08	0.65±0.08	0.65±0.08	0.65±0.08
macro avg	0.62±0.10	0.61±0.09	0.61±0.10	223.00±0.46
weighted avg	0.64±0.09	0.65±0.08	0.64±0.09	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')
```

Predicting on microscopy

In [16]:
```
mp_inputs = IO.load('Photography.pickle')
mp_inputs['Extractor'] = mp_inputs['ResNetAvg']
```

In [17]:
```
md_inputs = IO.load('Dermoscopy.pickle')
md_inputs['Extractor'] = md_inputs['ResNetAvg']
```

In [18]:
```
m_inputs['Extractor'] = m_inputs['Avg']
m_inputs = pandas.concat([mp_inputs, md_inputs, m_inputs], axis=
0)
```

In [19]:
```
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 [20]:
```
# SVM Linear
model = Pipeline([('scale', MinMaxScaler()), ('clf', LinearSVC(cl
ass_weight='balanced'))])
model_params = {'clf__C': logspace(-2, 3, 6).tolist()}
```

In [21]:
```
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, calibrate='isotonic')
```

```
Evaluation achieved!
```

In [22]:
```
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'])))
```

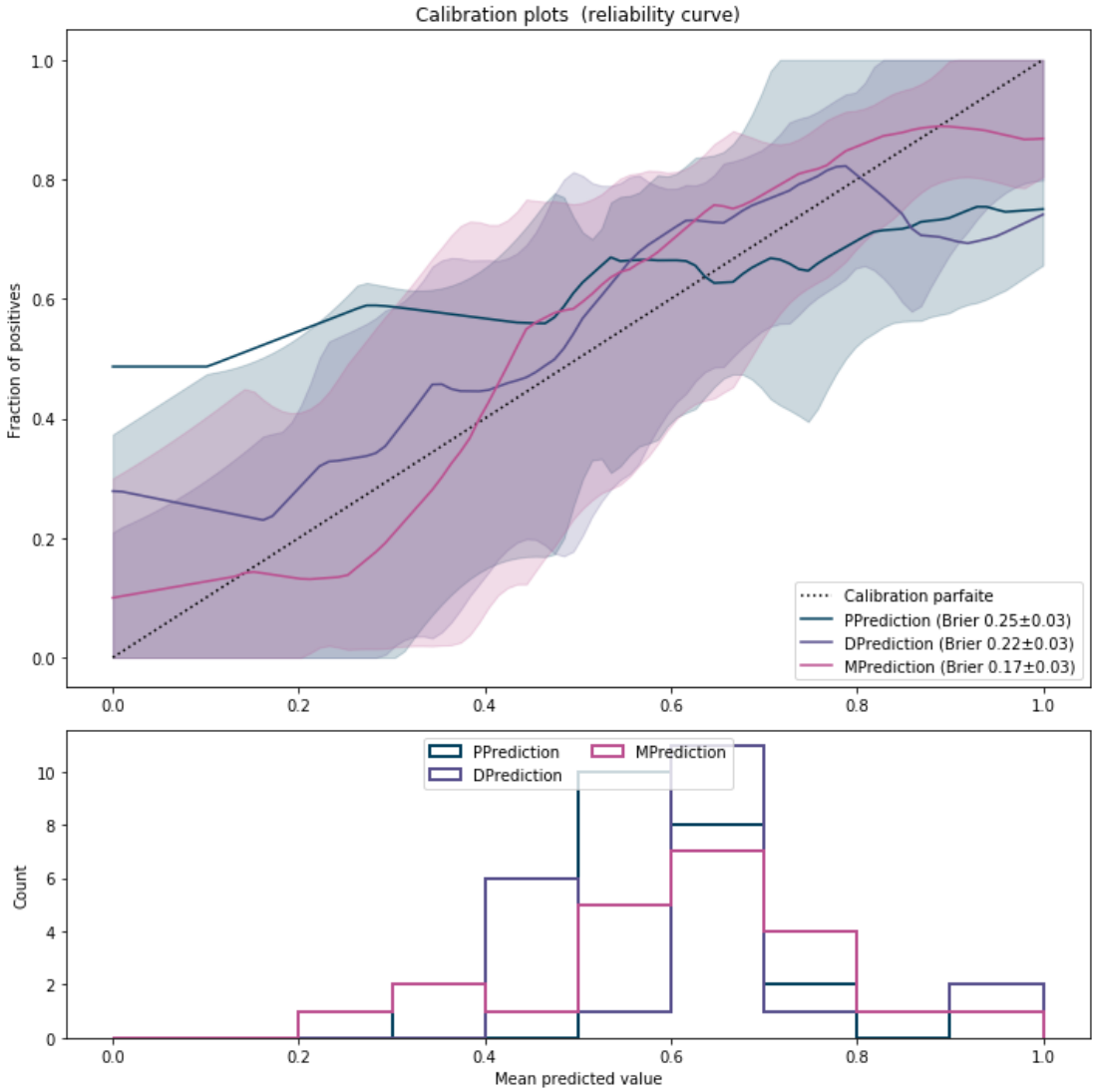
Microscopie

	precision	recall	f1-score	support
Benign	0.72±0.22	0.55±0.23	0.62±0.16	86.00±2.01
Malignant	0.75±0.10	0.87±0.12	0.81±0.04	137.00±2.00
accuracy	0.74±0.05	0.74±0.05	0.74±0.05	0.74±0.05
macro avg	0.74±0.09	0.71±0.08	0.71±0.09	223.00±0.46
weighted avg	0.74±0.07	0.74±0.05	0.74±0.07	223.00±0.46

In [23]:
```
# Calibration
```

In [24]:
```
p_inputs['PPrediction_Prediction'] = p_inputs['Prediction_Predict
ion']
p_inputs['PPrediction_Probability'] = p_inputs['Prediction_Probab
ility']
p_inputs['DPrediction_Prediction'] = d_inputs['Prediction_Predict
ion']
p_inputs['DPrediction_Probability'] = d_inputs['Prediction_Probab
ility']
p_inputs['MPrediction_Prediction'] = m_inputs['Prediction_Predict
ion']
p_inputs['MPrediction_Probability'] = m_inputs['Prediction_Probab
ility']

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



Fusion

In [25]:
```
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 [26]:
```
inputs = IO.load(f'Low_{prefix}.pickle')
```

In [27]:
```
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.PROBABILITY}')
inputs = inputs[single_image].reset_index(drop=True)
```

In [28]:
```
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 [29]:
```
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, modality, 'simple_increase_
increase')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, modality_rev, 'simple_incre
ase_decrease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, classe, 'double_increase_in
crease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, classe_rev, 'double_increas
e_decrease')

Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, modality_ones, 'simple_decr
ease_increase')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, modality_rev_ones, 'simple_
decrease_decrease')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, classe_ones, 'double_decrea
se_increase')
Tools.evaluate(inputs, {'datum': f'Prediction_{Tools.PROBABILIT
Y}', 'label_encode': 'LesionEncode'}, classe_rev_ones, 'double_de
crease_decrease')
IO.save(inputs, f'High_{prefix}.pickle')
```

```
Evaluation achieved!
```

Score

In [30]:
```
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()
```

Test - simple_increase_increase

	precision	recall	f1-score	support
Benign	0.62±0.38	0.34±0.31	0.44±0.29	86.00±2.01
Malignant	0.68±0.14	0.87±0.09	0.76±0.06	137.00±2.00
accuracy	0.66±0.11	0.66±0.11	0.66±0.11	0.66±0.11
macro avg	0.65±0.22	0.60±0.14	0.60±0.17	223.00±0.46
weighted avg	0.65±0.21	0.66±0.11	0.64±0.16	223.00±0.46

Test - simple_increase_decrease

	precision	recall	f1-score	support
Benign	0.56±0.32	0.29±0.33	0.38±0.30	86.00±2.01
Malignant	0.66±0.15	0.85±0.09	0.74±0.07	137.00±2.00
accuracy	0.64±0.11	0.64±0.11	0.64±0.11	0.64±0.11
macro avg	0.61±0.21	0.57±0.14	0.56±0.18	223.00±0.46
weighted avg	0.62±0.21	0.64±0.11	0.60±0.17	223.00±0.46

Test - double_increase_increase

	precision	recall	f1-score	support
Benign	0.50±0.20	0.05±0.18	0.09±0.17	86.00±2.01
Malignant	0.62±0.11	0.97±0.05	0.76±0.06	137.00±2.00
accuracy	0.61±0.09	0.61±0.09	0.61±0.09	0.61±0.09
macro avg	0.56±0.14	0.51±0.06	0.42±0.10	223.00±0.46
weighted avg	0.57±0.17	0.61±0.09	0.50±0.14	223.00±0.46

Test - double_increase_decrease

	precision	recall	f1-score	support
Benign	0.57±0.22	0.05±0.18	0.09±0.18	86.00±2.01
Malignant	0.62±0.11	0.98±0.04	0.76±0.07	137.00±2.00
accuracy	0.62±0.10	0.62±0.10	0.62±0.10	0.62±0.10
macro avg	0.60±0.15	0.51±0.07	0.42±0.12	223.00±0.46
weighted avg	0.60±0.17	0.62±0.10	0.50±0.14	223.00±0.46

Test - simple_decrease_increase

	precision	recall	f1-score	support
Benign	0.71±0.23	0.51±0.23	0.59±0.16	86.00±2.01
Malignant	0.74±0.10	0.87±0.13	0.80±0.05	137.00±2.00
accuracy	0.73±0.05	0.73±0.05	0.73±0.05	0.73±0.05
macro avg	0.72±0.09	0.69±0.08	0.70±0.09	223.00±0.46
weighted avg	0.73±0.07	0.73±0.05	0.72±0.07	223.00±0.46

Test - simple_decrease_decrease

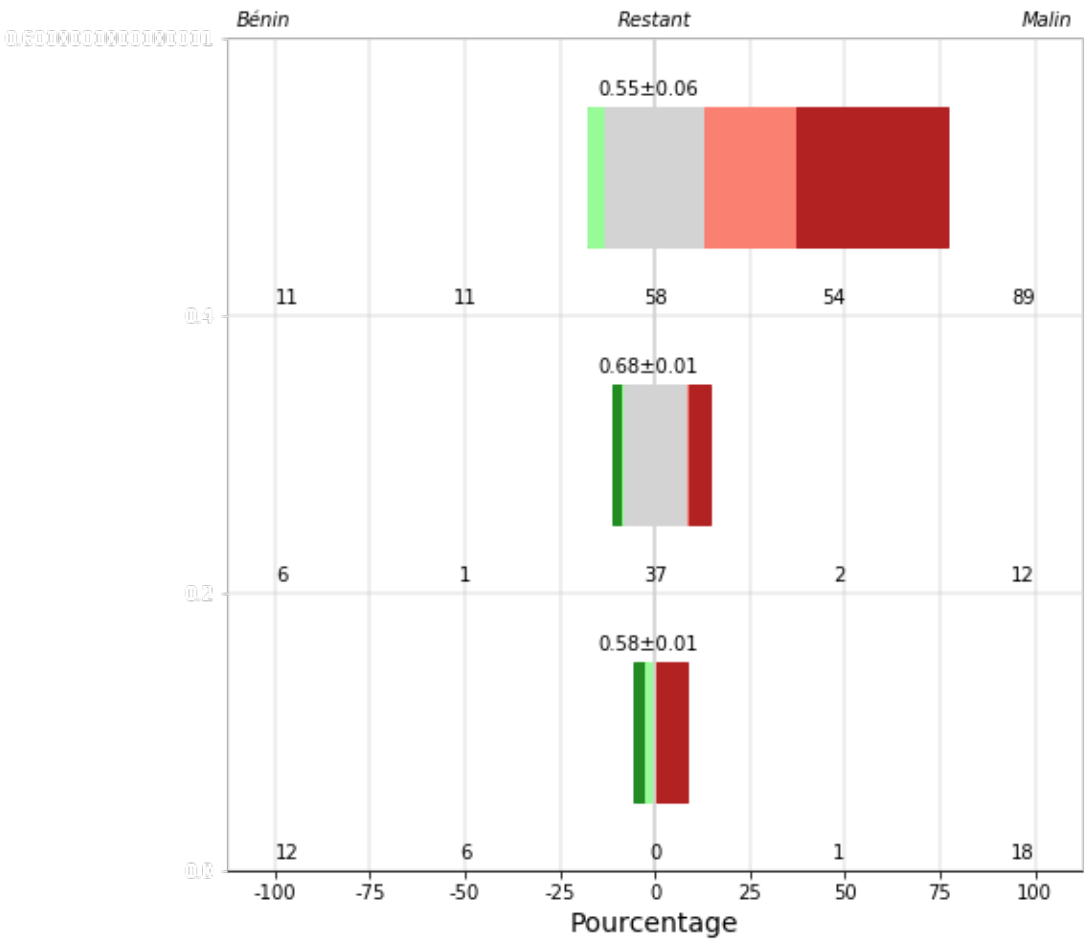
	precision	recall	f1-score	support
Benign	0.71±0.22	0.49±0.23	0.58±0.16	86.00±2.01
Malignant	0.73±0.11	0.88±0.11	0.80±0.04	137.00±2.00
accuracy	0.73±0.06	0.73±0.06	0.73±0.06	0.73±0.06
macro avg	0.72±0.09	0.68±0.09	0.69±0.09	223.00±0.46
weighted avg	0.72±0.08	0.73±0.06	0.71±0.07	223.00±0.46

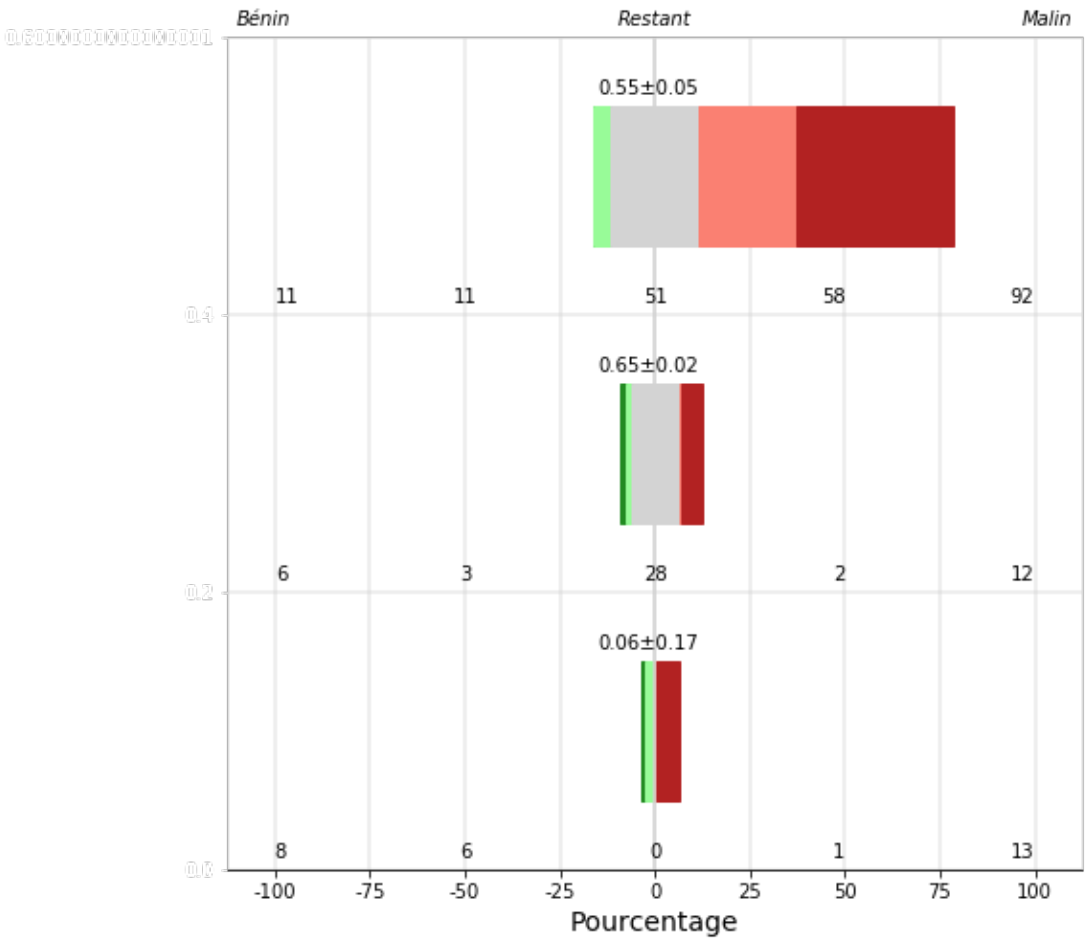
Test - double_decrease_increase

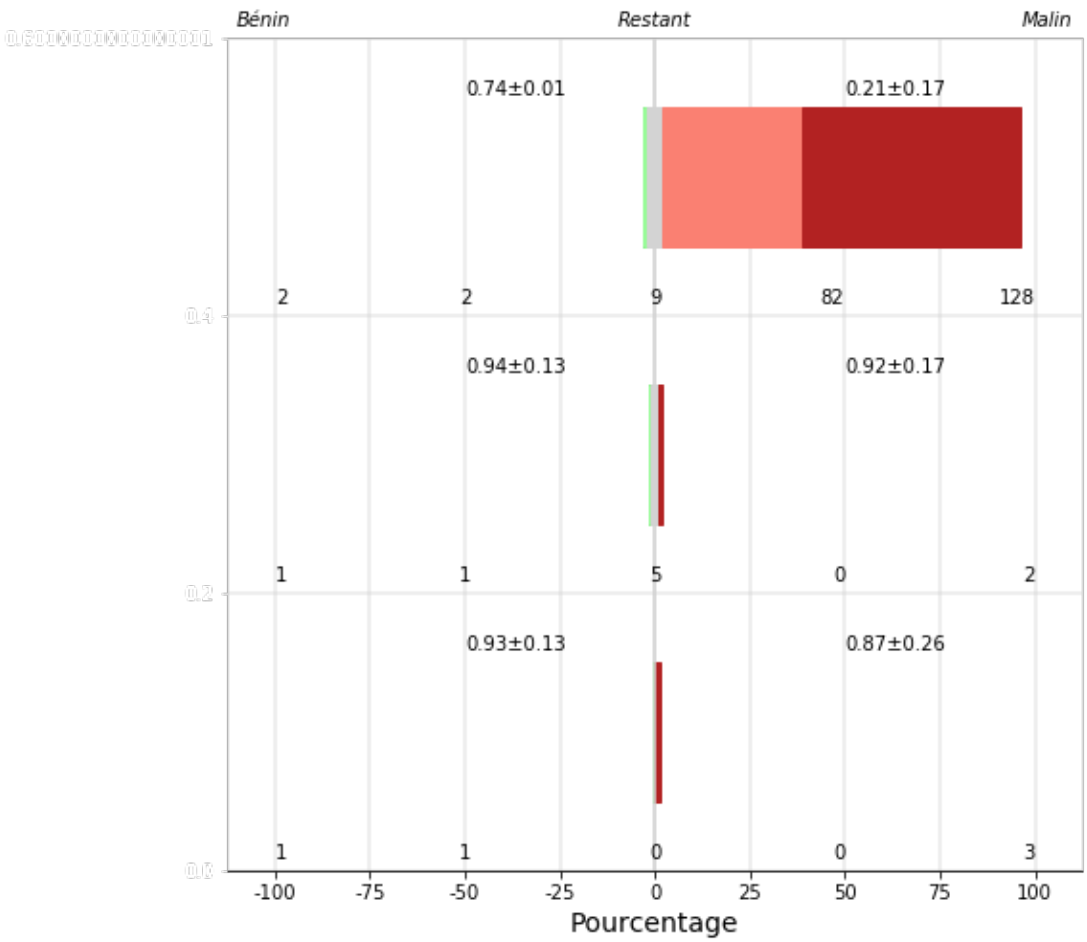
	precision	recall	f1-score	support
Benign	0.81±0.21	0.41±0.23	0.54±0.23	86.00±2.01
Malignant	0.72±0.11	0.94±0.08	0.81±0.07	137.00±2.00
accuracy	0.74±0.10	0.74±0.10	0.74±0.10	0.74±0.10
macro avg	0.77±0.10	0.67±0.11	0.68±0.14	223.00±0.46
weighted avg	0.75±0.07	0.74±0.10	0.71±0.13	223.00±0.46

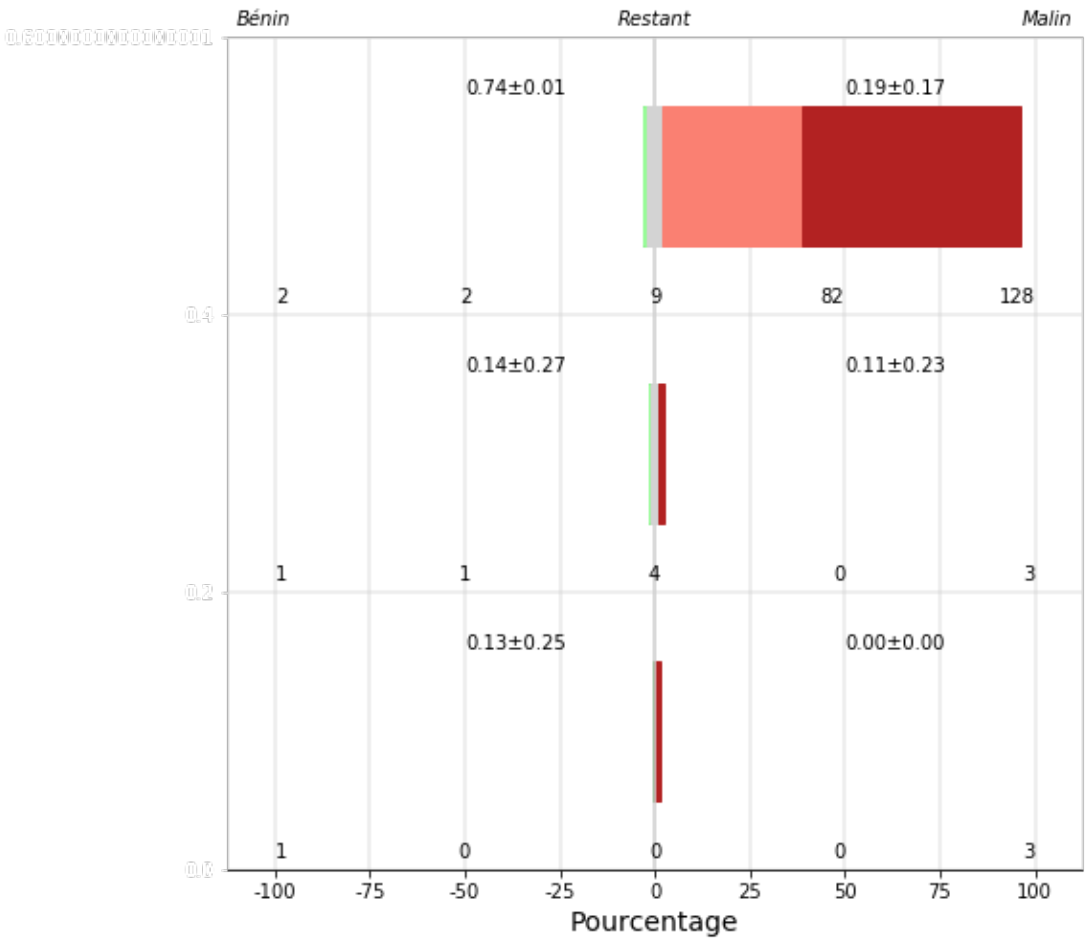
Test - double_decrease_decrease

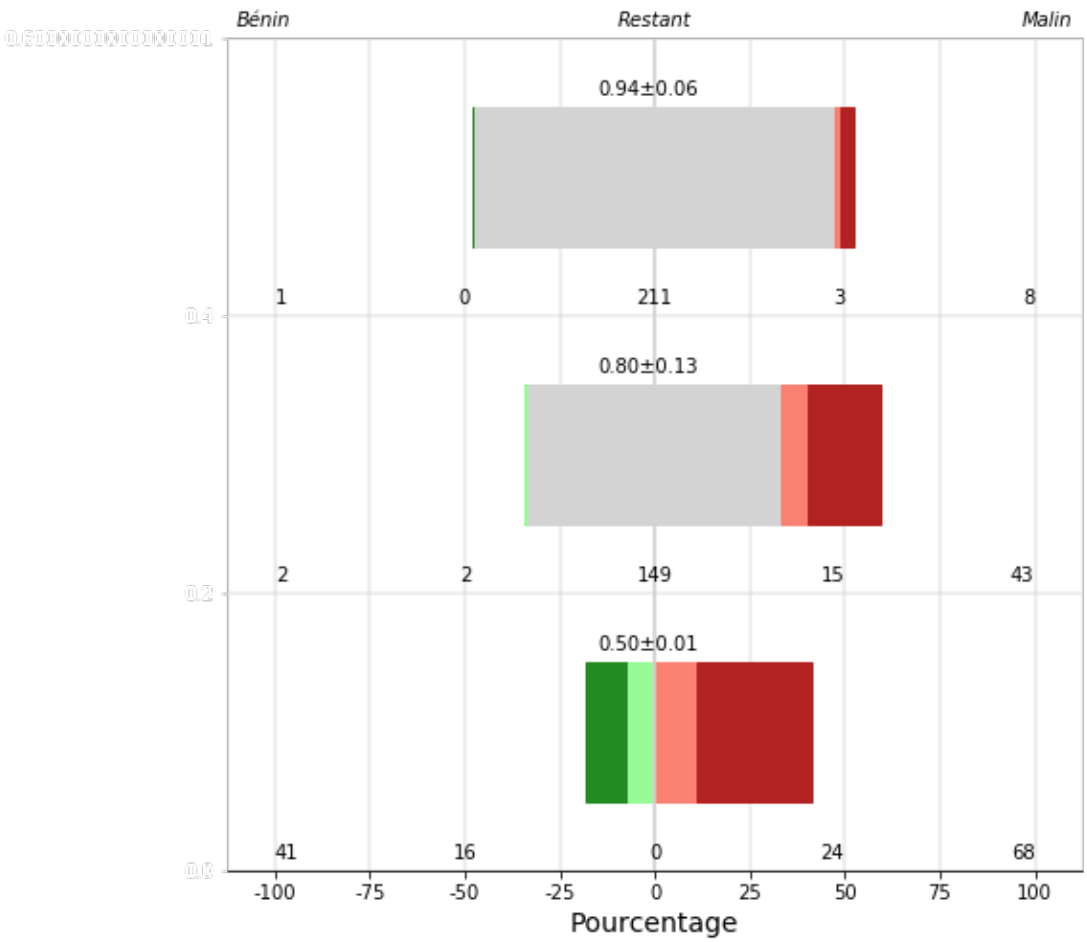
	precision	recall	f1-score	support
Benign	0.79±0.21	0.40±0.24	0.53±0.23	86.00±2.01
Malignant	0.71±0.12	0.93±0.08	0.81±0.07	137.00±2.00
accuracy	0.73±0.10	0.73±0.10	0.73±0.10	0.73±0.10
macro avg	0.75±0.09	0.66±0.11	0.67±0.14	223.00±0.46
weighted avg	0.74±0.07	0.73±0.10	0.70±0.14	223.00±0.46

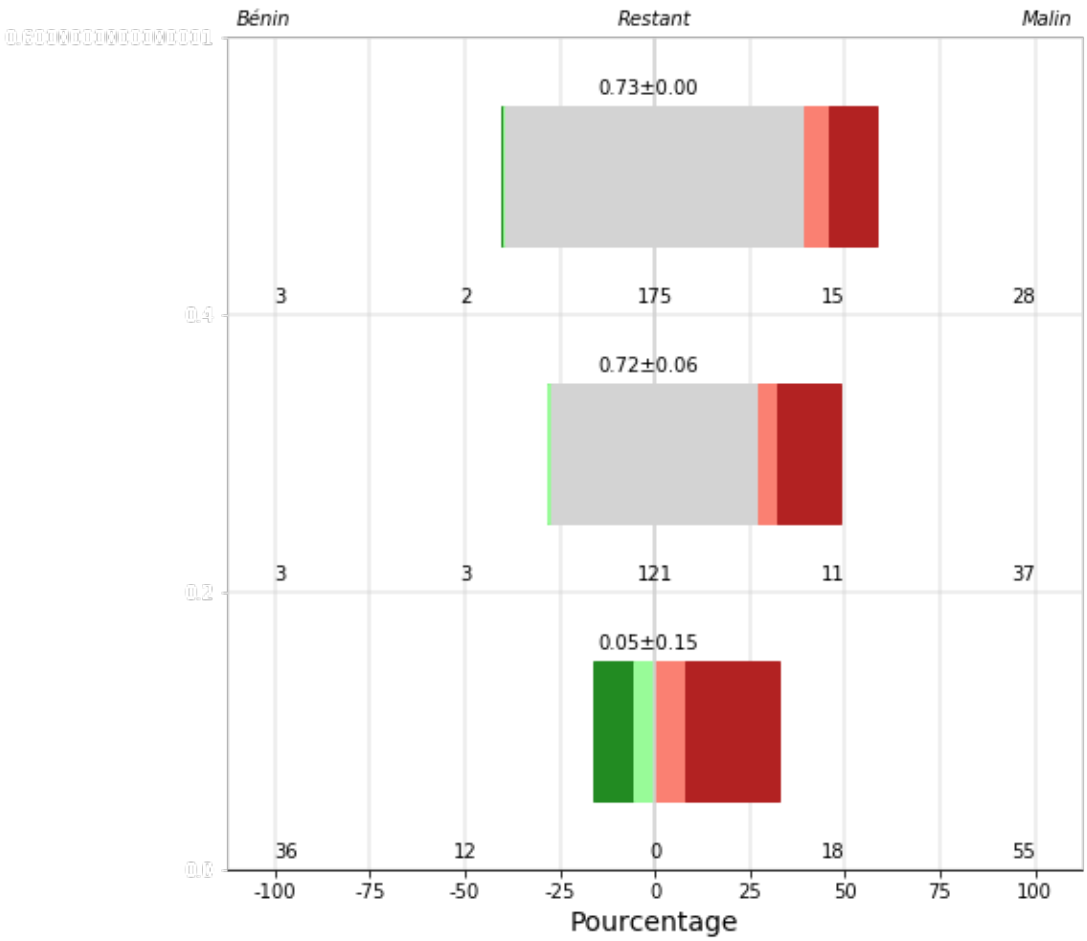

Bénin
Restant
Malin
0.55±0.06
11
11
58
54
89
0.68±0.01
6
1
37
2
12
0.58±0.01
12
6
0
1
18
0.4
0.2
-100
-75
-50
-25
0
25
50
75
100
Pourcentage

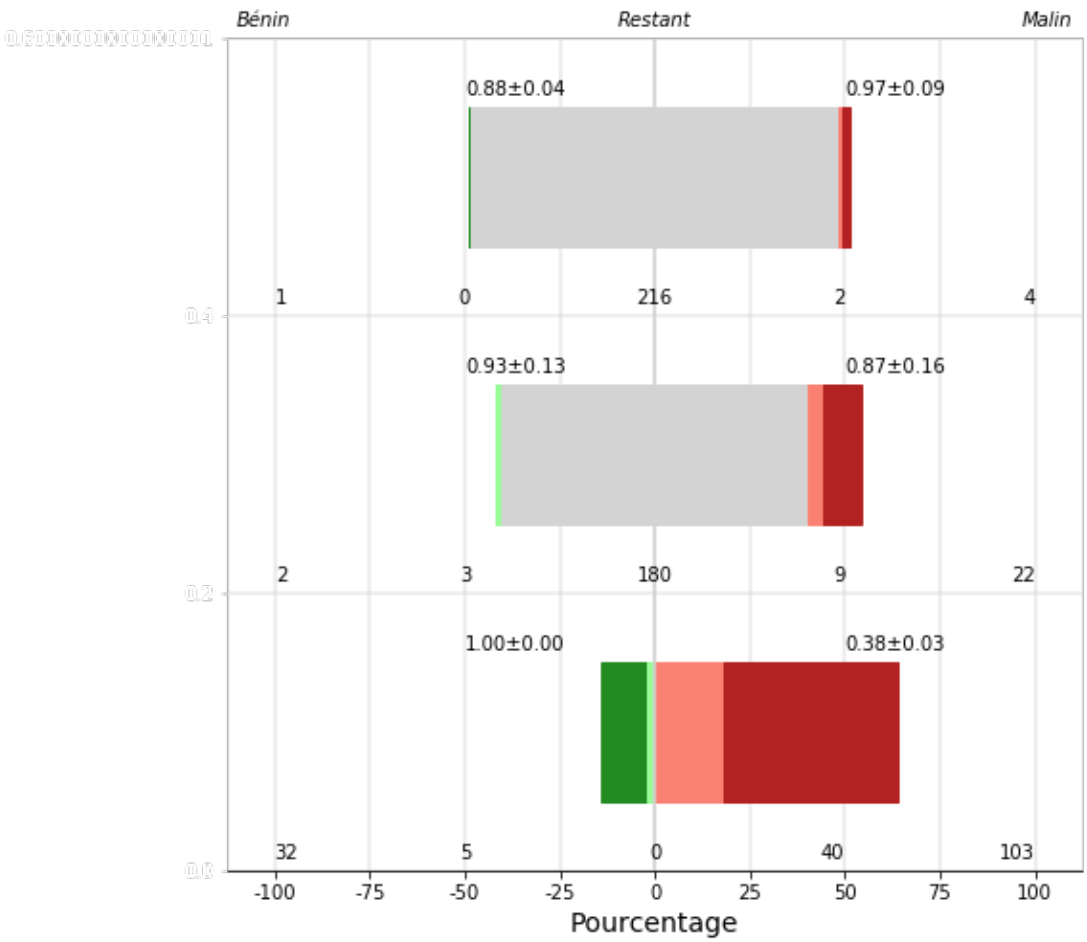

Bénin
Restant
Malin
0.55±0.05
11
11
51
58
92
0.65±0.02
6
3
28
2
12
0.06±0.17
8
6
0
1
13
0.4
0.2
-100
-75
-50
-25
0
25
50
75
100
Pourcentage

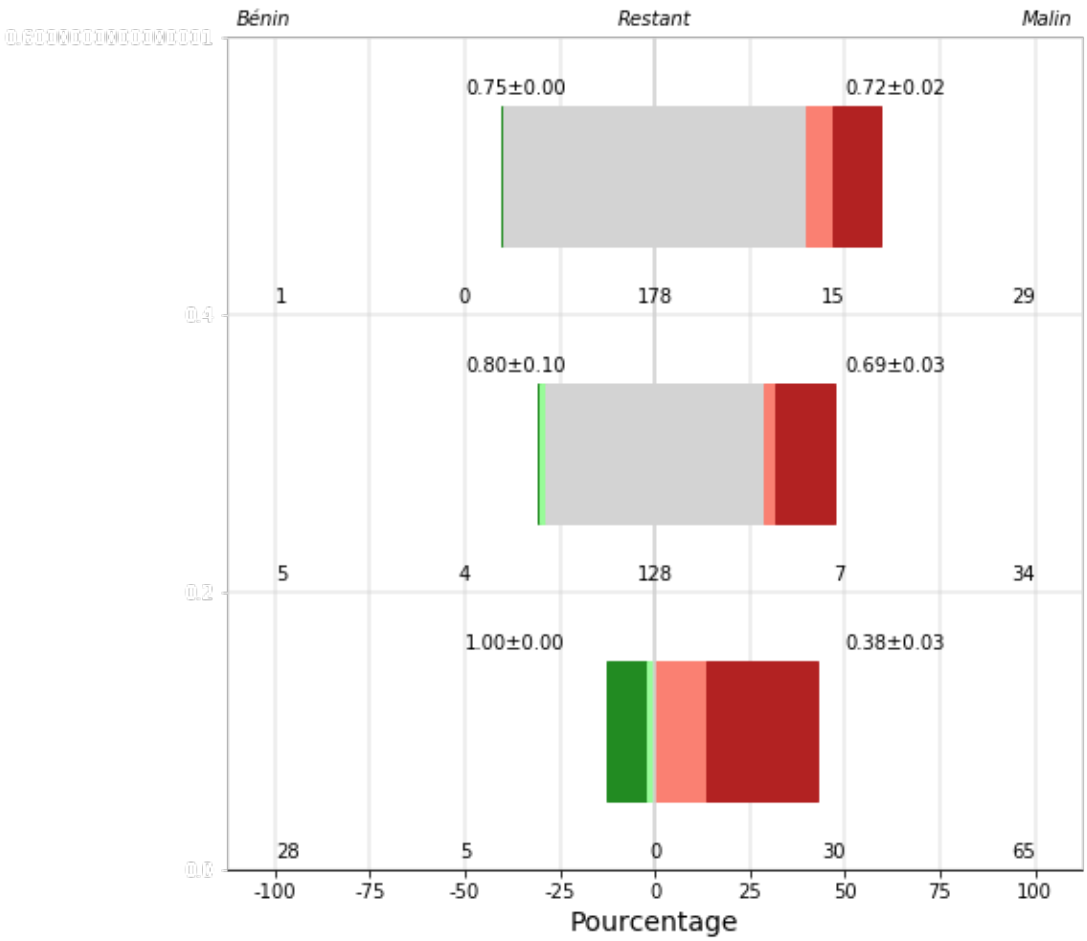

Bénin
Restant
Malin
0.74±0.01
0.21±0.17
2
2
9
82
128
0.94±0.13
0.92±0.17
1
1
5
0
2
0.93±0.13
0.87±0.26
1
1
0
0
3
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.74±0.01
0.19±0.17
2
2
9
82
128
0.14±0.27
0.11±0.23
1
1
4
0
3
0.13±0.25
0.00±0.00
1
0
0
0
3
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.94±0.06
1
0
211
3
8
0.80±0.13
2
2
149
15
43
0.50±0.01
41
16
0
24
68
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.73±0.00
3
2
175
15
28
0.72±0.06
3
3
121
11
37
0.05±0.15
36
12
0
18
55
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.88±0.04
0.97±0.09
1
0
216
2
4
0.93±0.13
0.87±0.16
2
3
180
9
22
1.00±0.00
0.38±0.03
32
5
0
40
103
-100
-75
-50
-25
0
25
50
75
100
Pourcentage


Bénin
Restant
Malin
0.75±0.00
0.72±0.02
1
0
178
15
29
0.80±0.10
0.69±0.03
5
4
128
7
34
1.00±0.00
0.38±0.03
28
5
0
30
65
-100
-75
-50
-25
0
25
50
75
100
Pourcentage