METIS

Day 10: Advanced Model Evaluation

John Navarro
john.navarro@thisismetis.com
https://www.linkedin.com/in/johnnavarro/

Null Accuracy

METIS

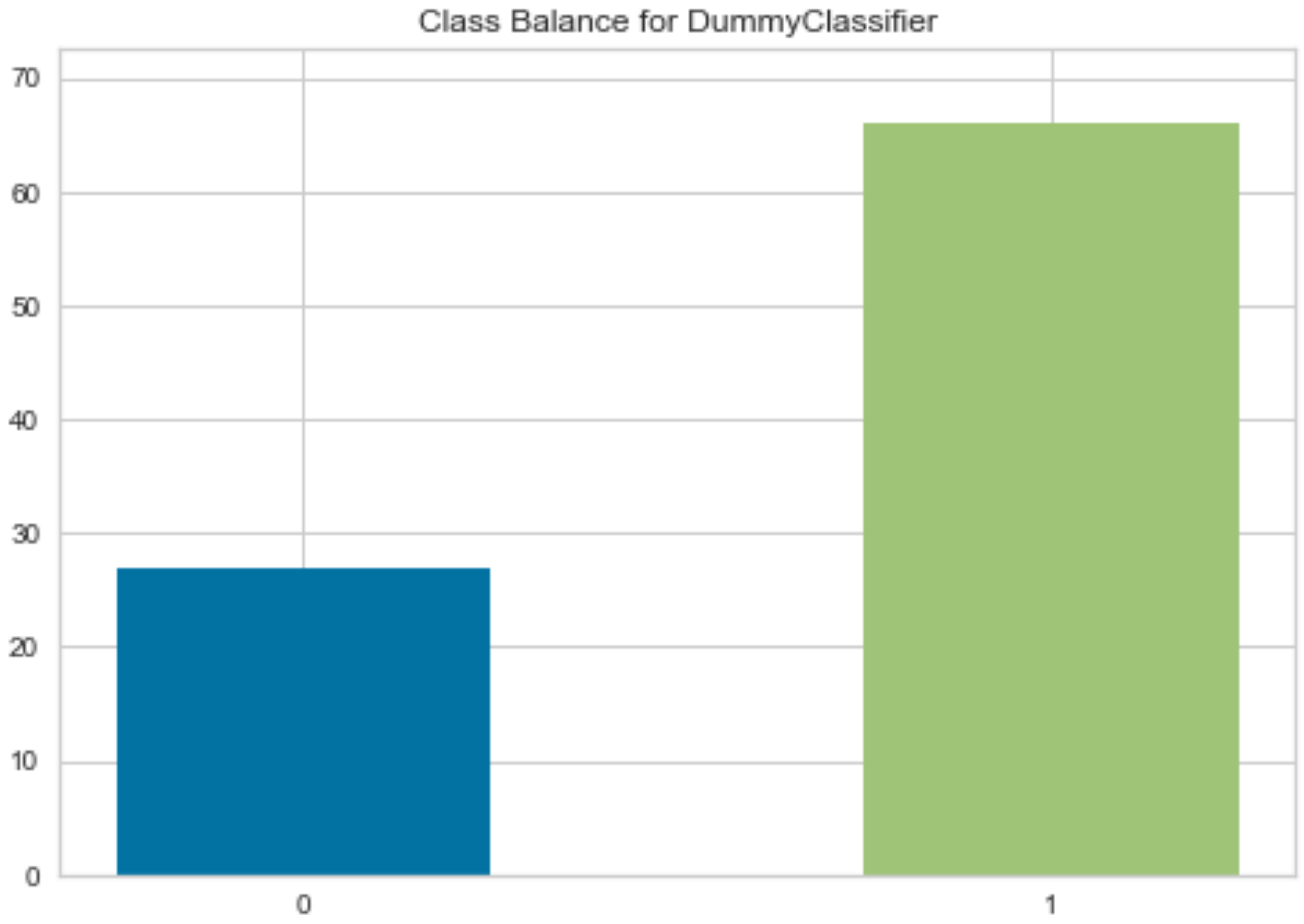
Null Accuracy

```
from sklearn.dummy import DummyClassifier
dumb_model = DummyClassifier(strategy='most_frequent')
dumb_model.fit(X_train, y_train)
y_dumb_class = dumb_model.predict(X_test)
metrics.accuracy_score(y_test, y_dumb_class)

>>> 0.709677419355
```

METIS

Null Accuracy

```
from yellowbrick import ClassBalance
visualizer = ClassBalance(dumb_model, classes=[0,1])
visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
g = visualizer.poof()
```



METIS

Performance Measures

- **Sensitivity/true positive rate(TPR)/recall:** What fraction of the "abnormal" samples in unseen data did we correctly predict?
- **Specificity/true negative rate(TNR):** What fraction of "normal" samples in unseen data did we correctly predict?
- **Precision/positive predictive value(PPV)** How frequently is our model correct when it predicts "abnormal" on new data?
- **Negative predictive value (NPV):** How frequently is our model correct when it predicts "normal" on new data?
- **Accuracy (ACC):** How frequently is our model correct on all new data, regardless of class?
- **F1 score (F1):** The harmonic mean of precision and recall:

METIS

Performance Measures

```
metrics.classification_report(y_test,y_test_pred)

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.78      0.78        27
           1       0.91      0.91      0.91        66

avg / total       0.87      0.87      0.87        93
```

METIS

Confusion Matrix

	Classified as: Positive	Classified as: Negative
Really is: Positive	True Positive	False Negative
Really is: Negative	False Positive	True Negative

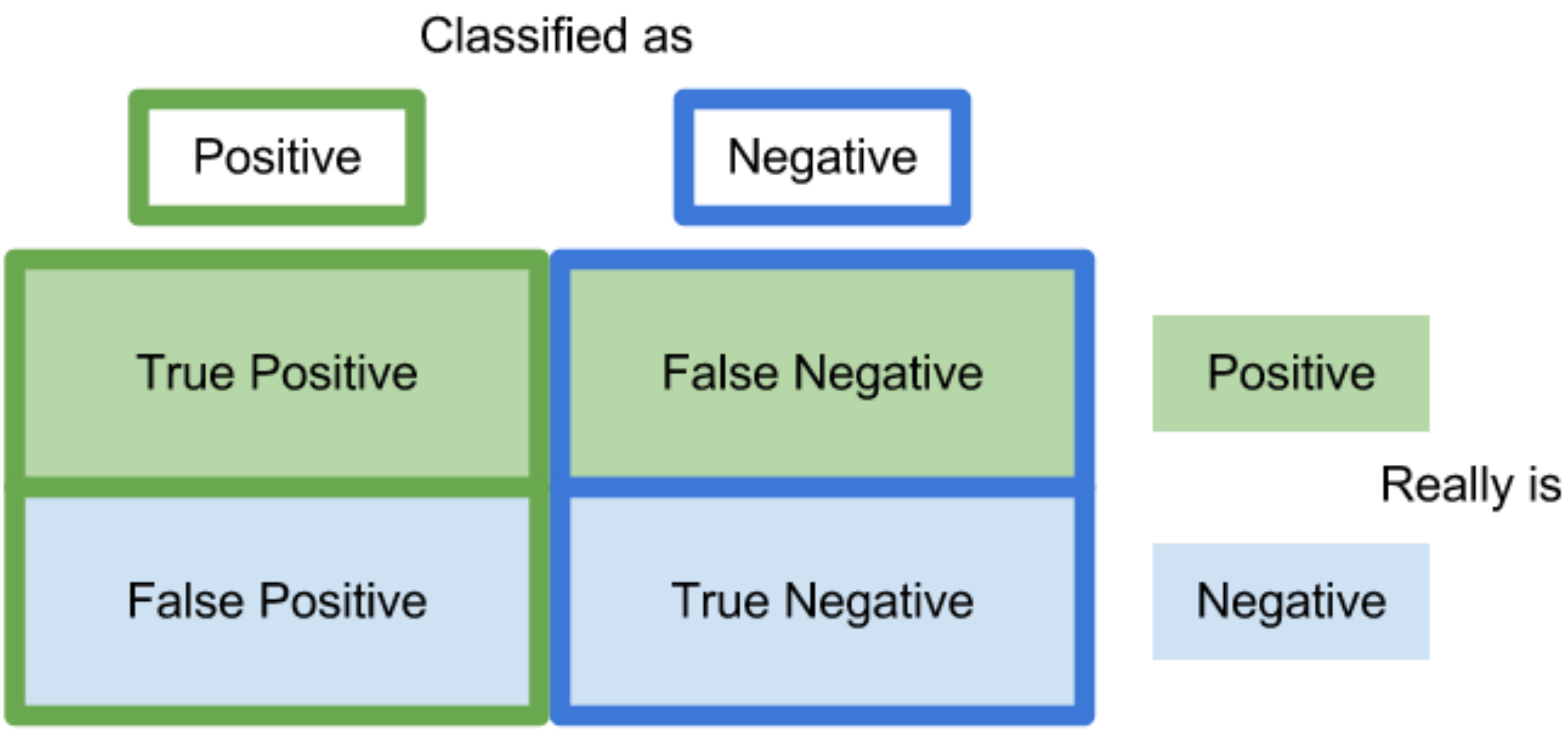
METIS

Confusion Matrix

- predict 0 (normal), actual 0 (normal) - called a **correct rejection/true negative**
- predict 0 (normal), actual 1 (abnormal) <-- this is an error called a **miss/false negative**
- predict 1 (abnormal), actual 0 (normal) <-- this is an error called a **false alarm/false positive**
- predict 1 (abnormal), actual 1 (abnormal) - called a **hit/true positive**

METIS

Confusion Matrix



METIS

Confusion Matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0]
print(metrics.confusion_matrix(target, target_pred))
```

$$\begin{array}{ccc} \text{True target} & 0 & [[7 \quad 2] \\ & 1 & \;\;[3 \quad 4]] \\ & & 0 \qquad 1 \\ & & \text{Predicted target} \end{array}$$

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

METIS

Accuracy

```
accuracy = metrics.accuracy_score(target, target_pred)
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

$$\text{True target}\quad \begin{array}{c|cc} 0 & [[7 & 2] \\ 1 & [3 & 4]] \\ \hline & 0 & 1 \end{array}$$

Predicted target

Accuracy = (TP + TN)/ N

Quiz: Calculate by hand!

METIS

Accuracy

```
accuracy = metrics.accuracy_score(target, target_pred)
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 7 & 2 \\ 1 & 3 & 4 \end{array} \qquad [[7\ \ 2]\ [3\ \ 4]]$$

Predicted target

$$\begin{aligned} \text{Accuracy} &= (TP + TN)/\ N \\ &= (7+4)/16 \\ &= 0.6875 \end{aligned}$$

METIS

Precision

Column-wise!

Precision

```
precision_class = metrics.precision_score(target, target_pred, average = None)
precision_avg = metrics.precision_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

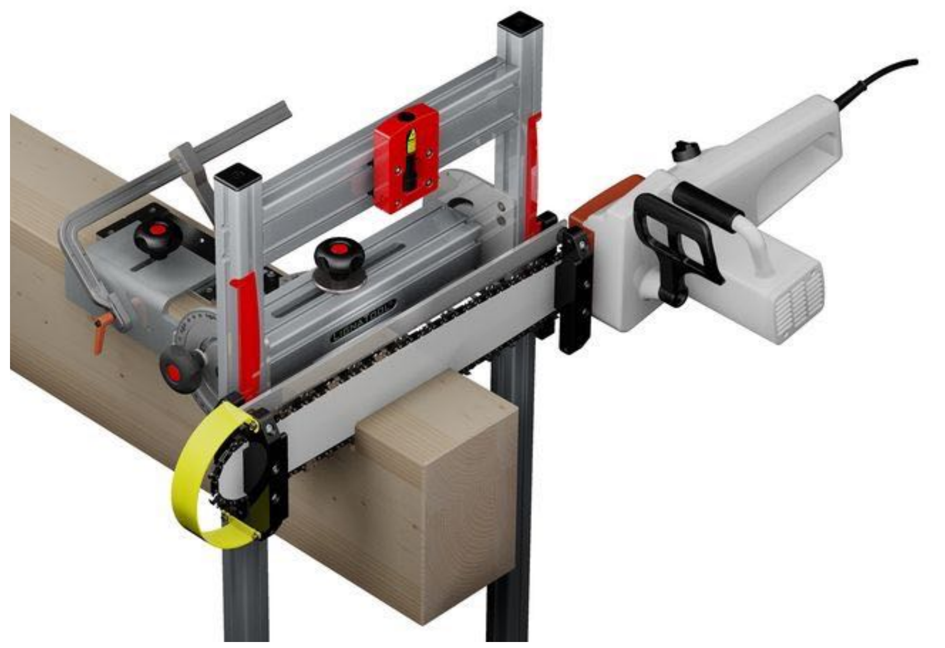
$$
\text{True target}\quad
\begin{array}{c c}
0 & \\
1 & \\
\end{array}
\begin{bmatrix} 7 & 2 \\ 3 & 4 \end{bmatrix}
$$

$$
\begin{array}{cc} 0 & 1 \end{array}
$$

Predicted target

Precision = TP/ (TP + FP)

Quiz: Calculate precision by hand for both classes!



Precision

```
precision_class = metrics.precision_score(target, target_pred, average = None)
precision_avg = metrics.precision_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

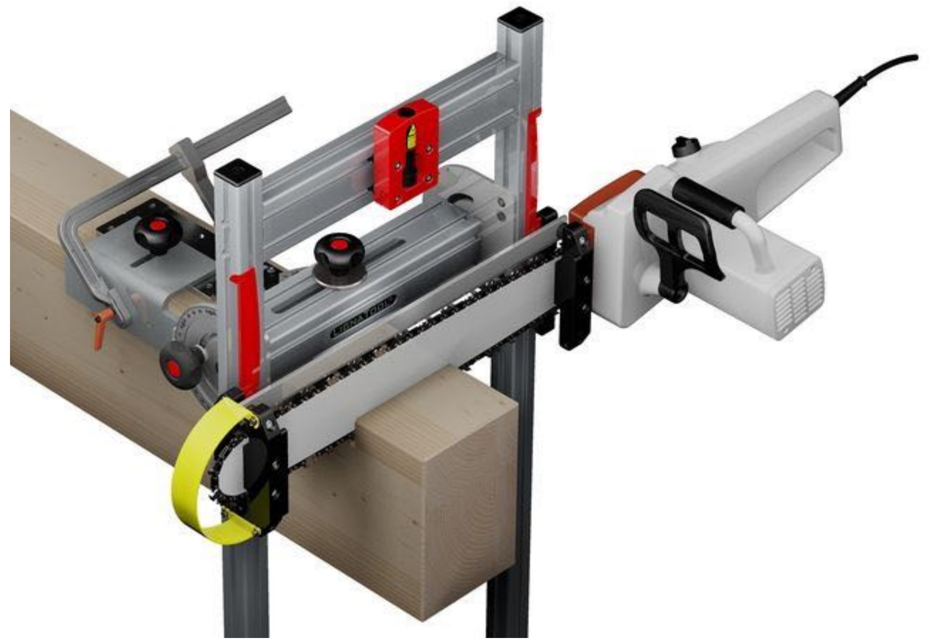
$$
\text{True target}\quad
\begin{array}{c|cc}
 & & \\
0 & 7 & 2 \\
1 & 3 & 4 \\
\hline
 & 0 & 1
\end{array}
\qquad
\begin{bmatrix} 7 & 2 \\ 3 & 4 \end{bmatrix}
$$

Predicted target

Precision = TP/ (TP + FP)

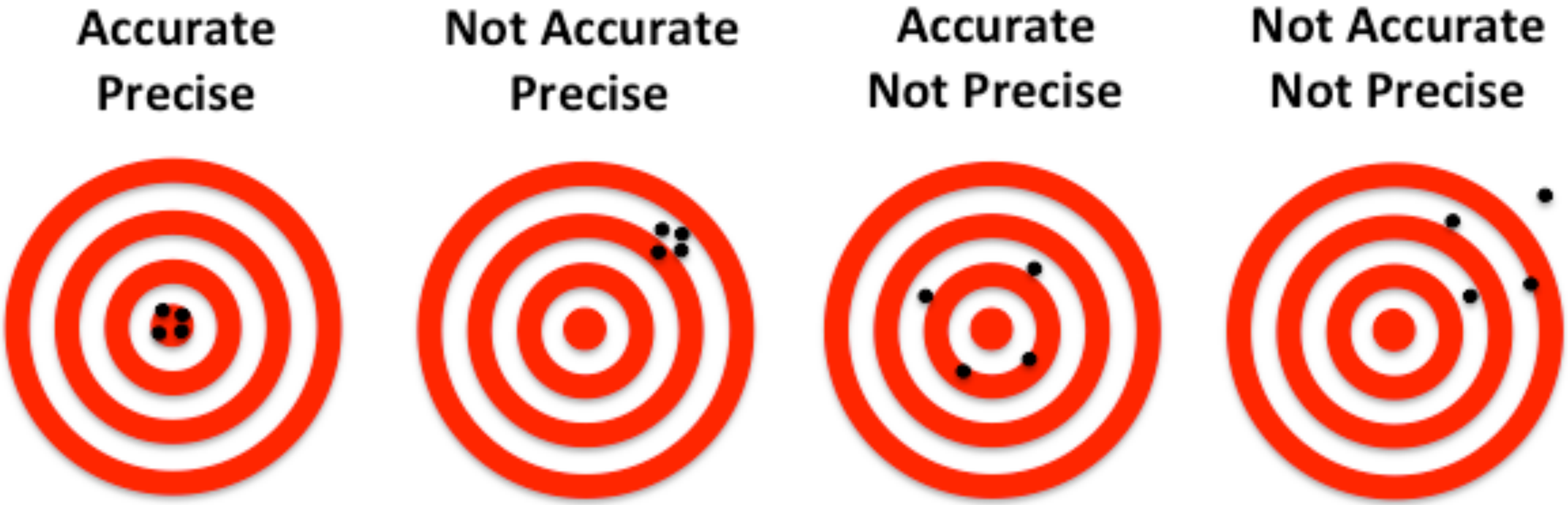
Prec_Class0 = 7/(7+3) = 0.7

Prec_Class1 = 4/(4+2) = 0.666



METIS

Precision



METIS

Recall

Row-wise!

Recall

```
recall_class = metrics.recall_score(target, target_pred, average = None)
recall_avg = metrics.recall_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

$$\text{True target} \quad \begin{array}{c|c} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 7 & 2 \\ 3 & 4 \end{bmatrix} \end{array}$$

Predicted target

Recall = TP/ (TP + FN)
Quiz: Calculate recall by hand for both classes!

METIS

Recall

```
recall_class = metrics.recall_score(target, target_pred, average = None)
recall_avg = metrics.recall_score(target, target_pred, average = 'binary')
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

$$\begin{array}{c c} 0 & \begin{matrix}[[7 & 2]\end{matrix} \\ 1 & \begin{matrix}[3 & 4]]\end{matrix} \\ & \begin{matrix}0 & 1\end{matrix} \end{array}$$

Predicted target

Recall = TP/ (TP + FN)

Rec_Class0 = 7/(7+2) = 0.777

Rec_Class1 = 4/(4+3) = 0.571

METIS

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(metrics.confusion_matrix(target, target_pred))
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

```
[[7 2 5]
 [3 4 2]
 [0 1 4]]
```

Predicted target

Accuracy = (Sum Diagonal)/ N

Precision = TP/ (TP + FP)

Recall = TP/ (TP + FN)

METIS

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(metrics.confusion_matrix(target, target_pred))
```

True positive	False Negative (Type II error)
False Positive (Type I error)	True negative

True target

```
[[7 2 5]
 [3 4 2]
 [0 1 4]]
```

Predicted target

Accuracy = (7+4+4)/28 = 0.5357

Prec_Class2 = 4/(4 + 5 + 2) = 0.3636

Rec_Class2 = 4/(4 + 1 + 0) = 0.8

…

```
Precision:  [ 0.7   0.57142857  0.36363636]
Recall:  [ 0.5  0.44444444  0.8]
```

METIS

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
[[10  0]
 [5  1]]
```

A

```
[[7  2]
 [3  4]]
```

B

```
[[0  4]
 [1  11]]
```

C

METIS

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
Precision:  [ 0.7   1]
Recall:  [ 1   0.2]

[[10  0]
 [5  1]]
```

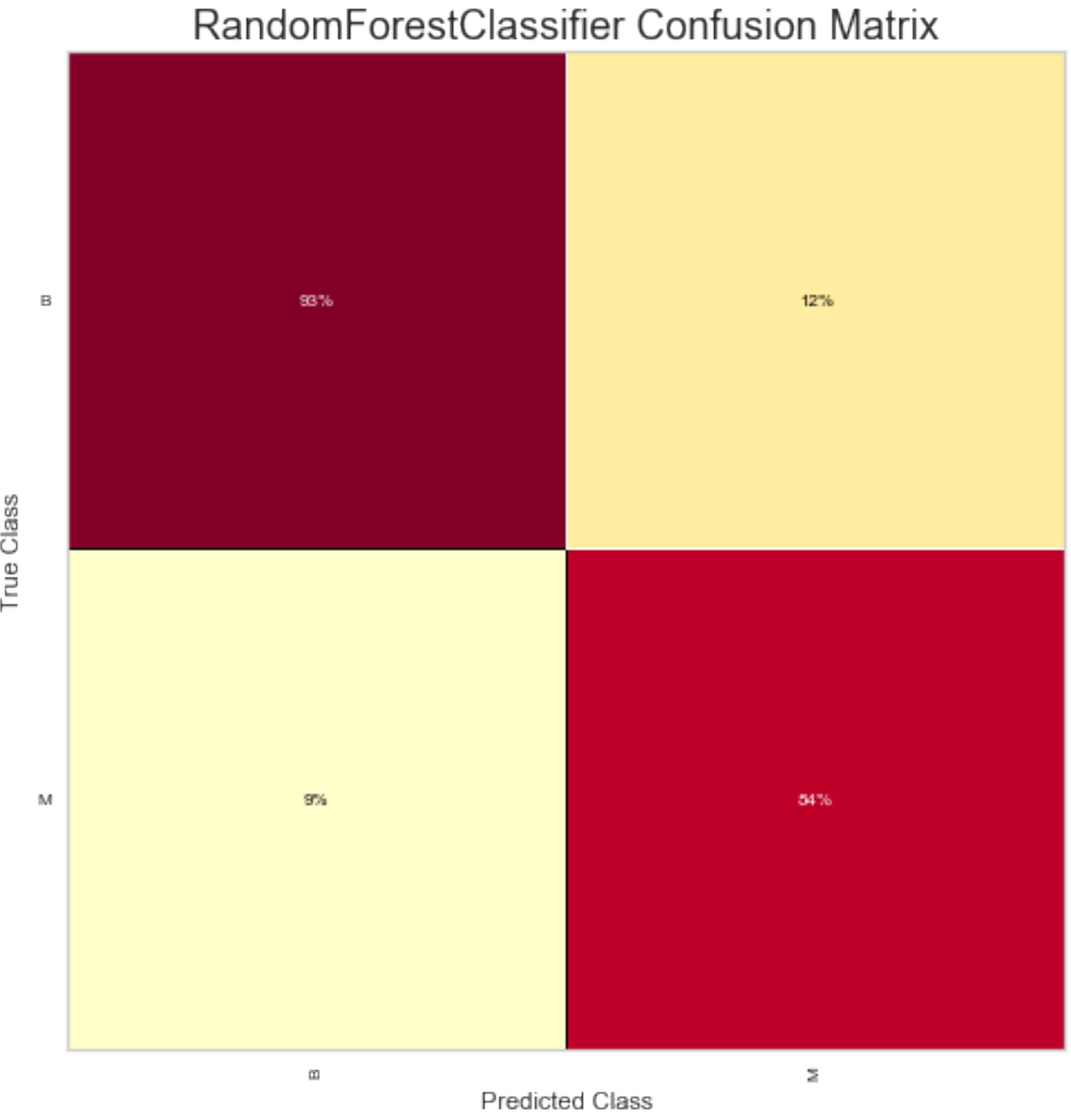
A

```
Precision:  [ 0.7   0.7]
Recall:  [ 0.8   0.6]

[[7  2]
 [3  4]]
```

B

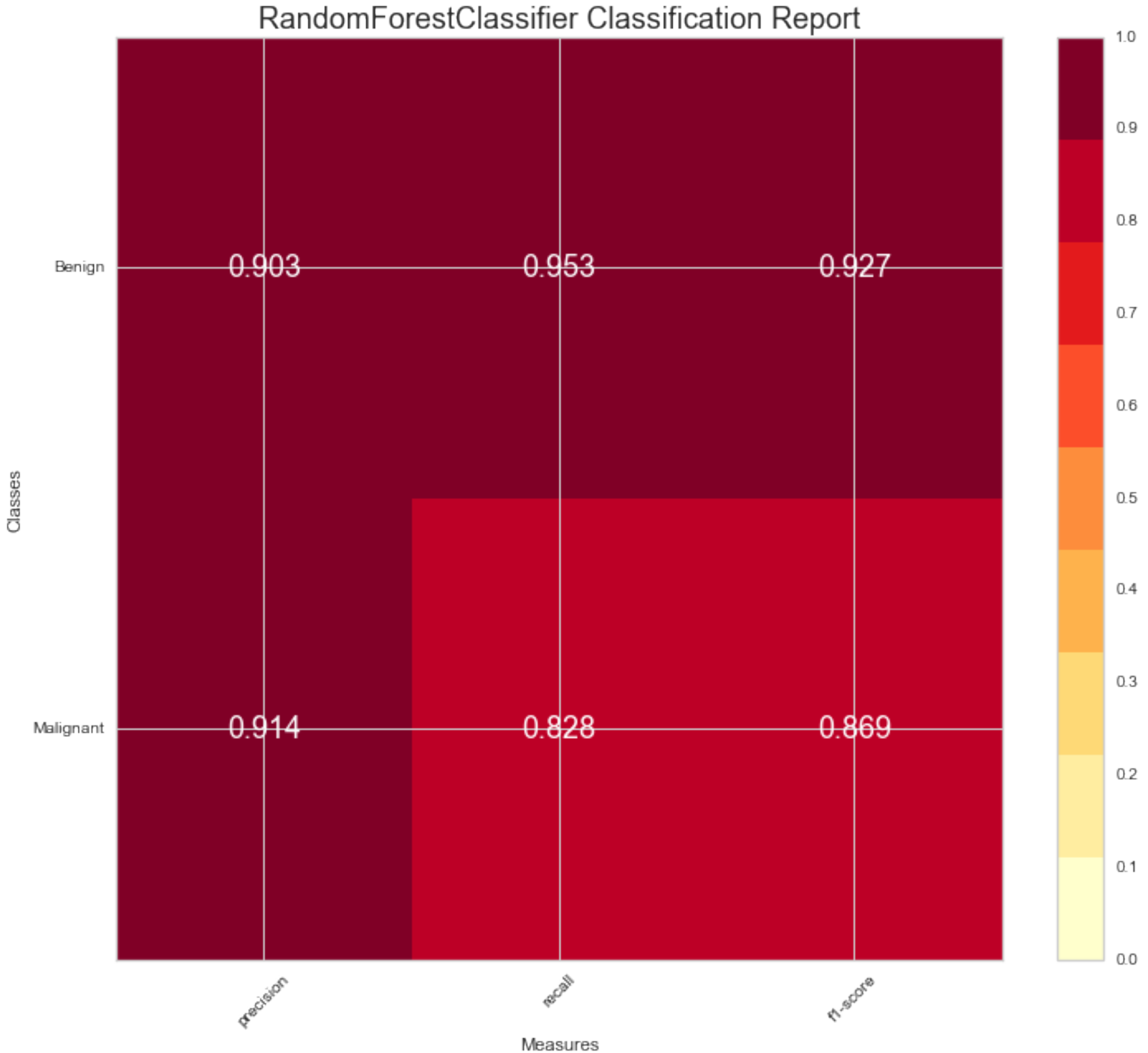
```
Precision:  [ 0   0.7]
Recall:  [ 0   0.9]

[[0  4]
 [1  11]]
```

Visualizing the Confusion Matrix

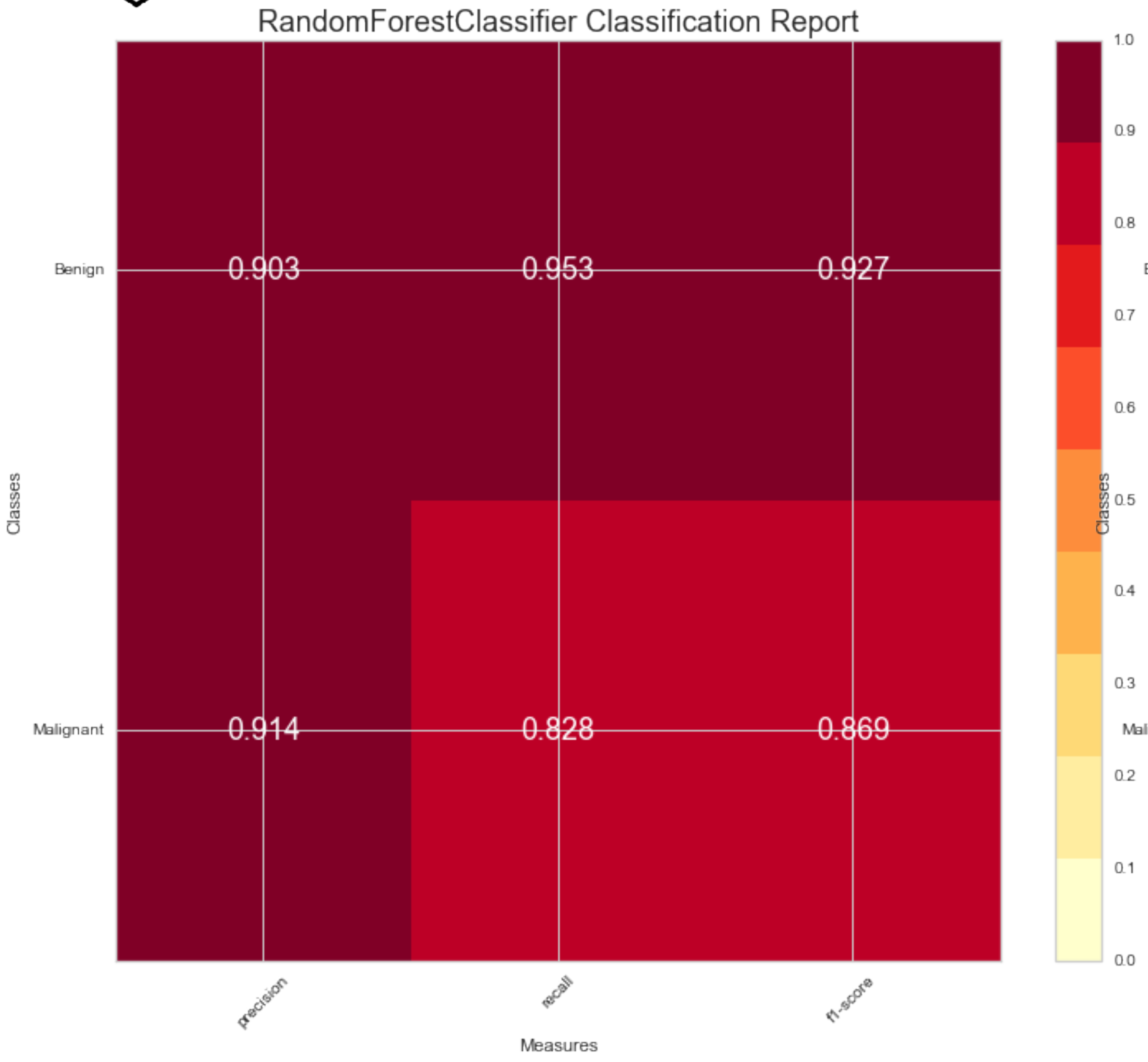


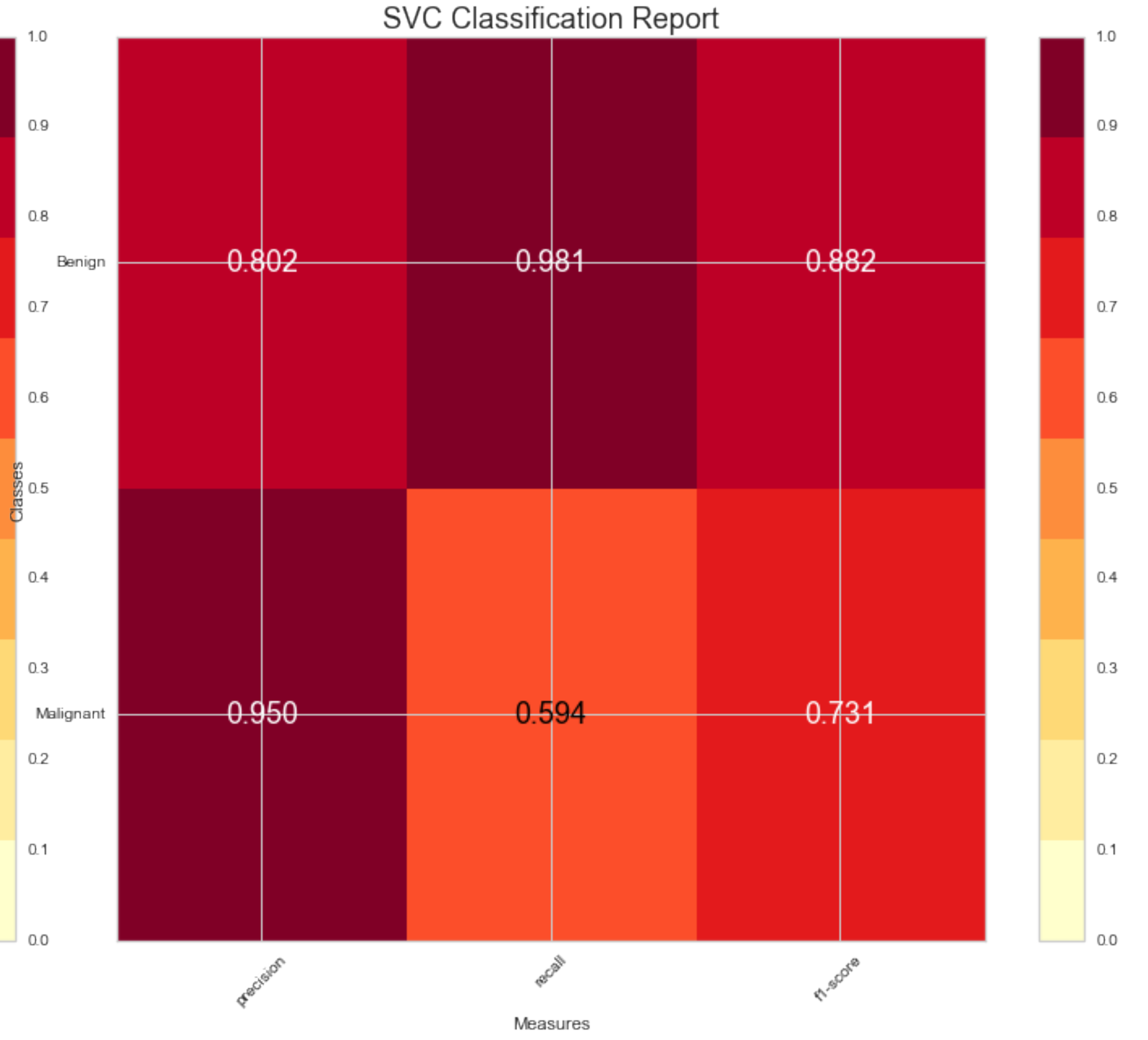

```
from yellowbrick.classifier import ConfusionMatrix

#Create the models for both the Random Forest
random_forest_model = RandomForestClassifier()

random_forest_conf_matrix = ConfusionMatrix(
                              random_forest_model )
random_forest_conf_matrix.fit( X_train, y_train )
random_forest_conf_matrix.score( X_test, y_test )
random_forest_conf_matrix.poof()
```

Visualizing the Classification Report

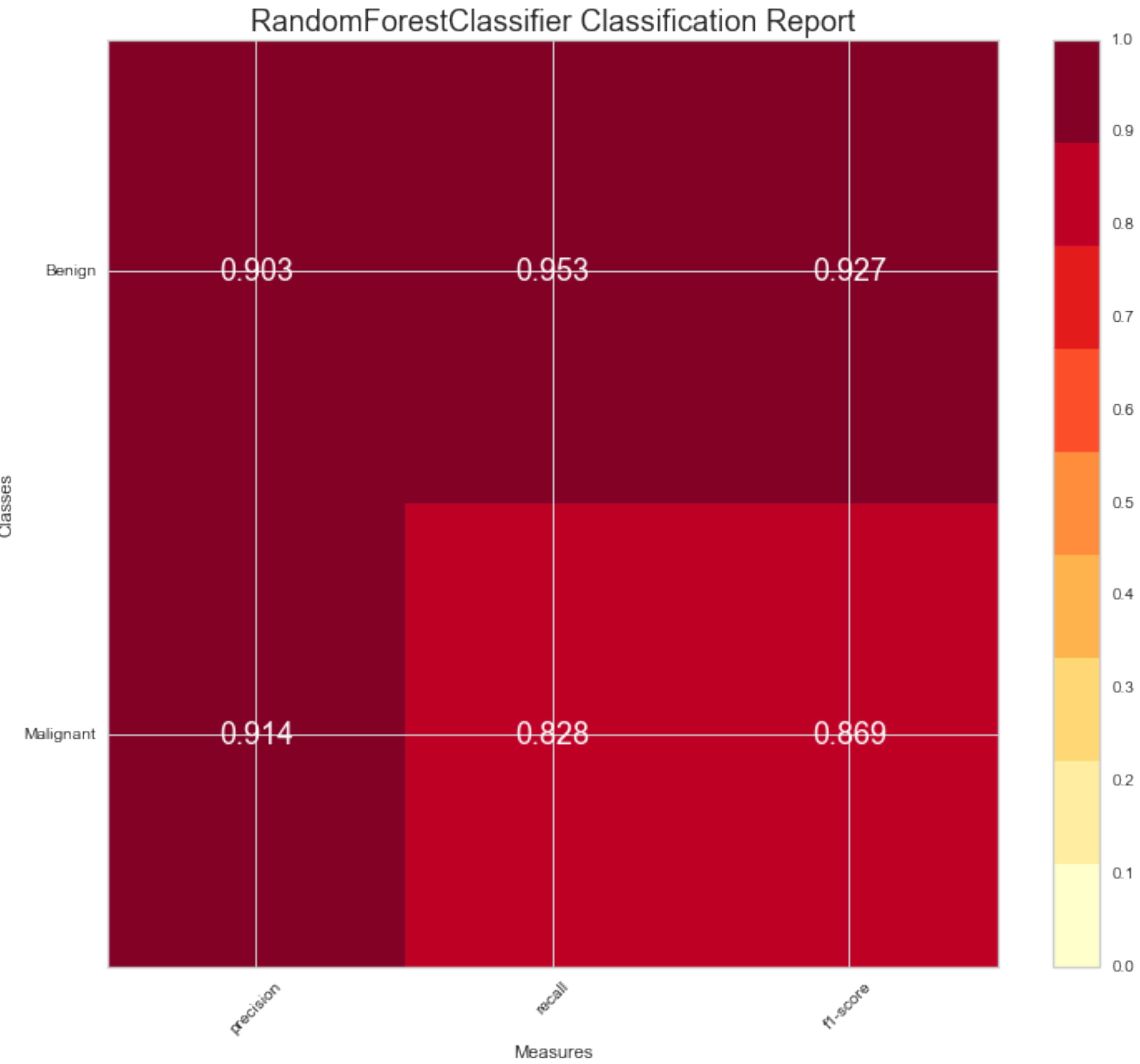


Visualizing the Classification Report





Visualizing the Classification Report




```python
from yellowbrick.classifier import ClassificationReport

random_forest_class_report = ClassificationReport( random_forest_model,
                                                    classes=['Benign', 'Malignant'])
random_forest_class_report.fit(X_train, y_train)
random_forest_class_report.score(X_test, y_test)
random_forest_class_report.poof()
```

METIS

An Receiver Operating Characteristic (ROC) Curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is systematically varied.

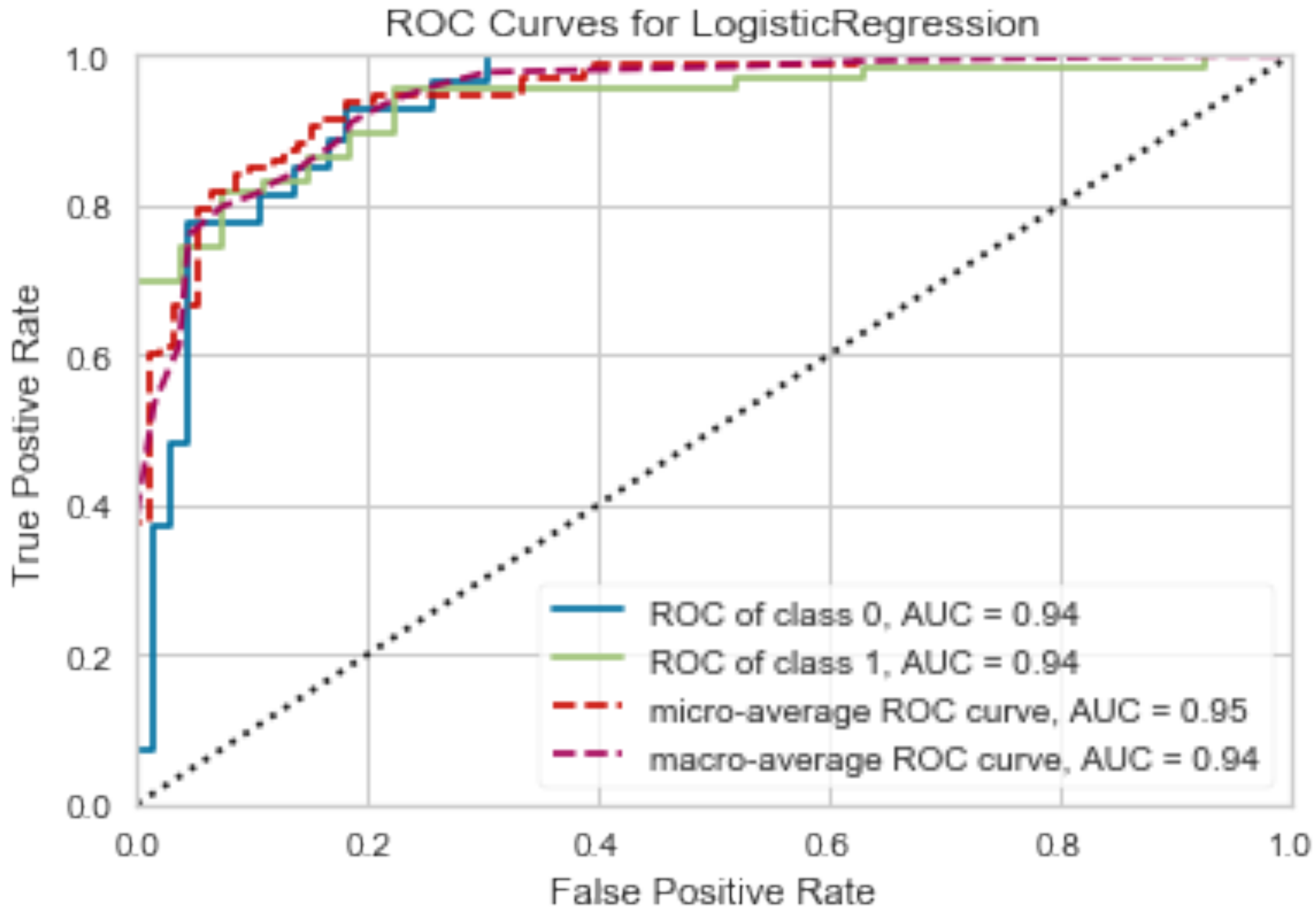
METIS

ROC Curves

	class_0	class_1	predicted	actual
0	0.06	0.94	1.0	1.0
1	0.38	0.62	1.0	0.0
2	0.08	0.92	1.0	1.0
3	0.08	0.92	1.0	1.0
4	0.22	0.78	1.0	1.0

ROC Curves

```
visualizer = ROCAUC(lr)

visualizer.fit(X_train, y_train)
visualizer.score(X_test, y_test)
g = visualizer.poof()
```



METIS

ROC Curves

```
metrics.roc_auc_score(y_test, y_preds)
```