Classification Metrics: Mathematical Formulations and Python Code

# Confusion Matrix

The confusion matrix is a summary of prediction results on a classification problem. It shows the counts of actual vs predicted labels.  
  
Mathematical Formulation:  
True Positive (TP): Correctly predicted positive samples  
True Negative (TN): Correctly predicted negative samples  
False Positive (FP): Incorrectly predicted as positive  
False Negative (FN): Incorrectly predicted as negative  
  
Confusion Matrix = [ TP FP ]  
 [ FN TN ]  
  
Python Code:  
```python  
from sklearn.metrics import confusion\_matrix  
y\_true = [0, 1, 0, 1, 0, 1, 0]  
y\_pred = [0, 1, 0, 0, 0, 1, 1]  
cm = confusion\_matrix(y\_true, y\_pred)  
print(cm)  
```

# Classification Accuracy

Accuracy measures the percentage of correct predictions out of all predictions.  
  
Mathematical Formulation:  
Accuracy = (TP + TN) / (TP + TN + FP + FN)  
  
Python Code:  
```python  
from sklearn.metrics import accuracy\_score  
accuracy = accuracy\_score(y\_true, y\_pred)  
print(accuracy)  
```

# Classification Report

A classification report provides detailed metrics like precision, recall, f1-score, and support for each class.  
  
Mathematical Formulation:  
- Precision = TP / (TP + FP)  
- Recall = TP / (TP + FN)  
- F1 Score = 2 \* (Precision \* Recall) / (Precision + Recall)  
- Support: Number of occurrences of each class in the dataset.  
  
Python Code:  
```python  
from sklearn.metrics import classification\_report  
report = classification\_report(y\_true, y\_pred)  
print(report)  
```

# Precision

Precision is the fraction of relevant instances among the retrieved instances.  
  
Mathematical Formulation:  
Precision = TP / (TP + FP)  
  
Python Code:  
```python  
from sklearn.metrics import precision\_score  
precision = precision\_score(y\_true, y\_pred)  
print(precision)  
```

# Recall or Sensitivity

Recall measures how many actual positive instances were correctly identified by the model.  
  
Mathematical Formulation:  
Recall = TP / (TP + FN)  
  
Python Code:  
```python  
from sklearn.metrics import recall\_score  
recall = recall\_score(y\_true, y\_pred)  
print(recall)  
```

# Specificity

Specificity measures the proportion of actual negatives that are correctly identified.  
  
Mathematical Formulation:  
Specificity = TN / (TN + FP)  
  
Python Code:  
```python  
specificity = cm[1,1] / (cm[1,1] + cm[0,1])  
print(specificity)  
```

# Support

Support refers to the number of actual occurrences of the class in the dataset.  
  
Support is directly output by classification\_report().  
  
Python Code:  
```python  
# Already obtained from classification\_report  
```

# F1 Score

The F1 Score is the harmonic mean of precision and recall, providing a balance between them.  
  
Mathematical Formulation:  
F1 = 2 \* (Precision \* Recall) / (Precision + Recall)  
  
Python Code:  
```python  
from sklearn.metrics import f1\_score  
f1 = f1\_score(y\_true, y\_pred)  
print(f1)  
```

# ROC AUC Score

The ROC (Receiver Operating Characteristic) AUC (Area Under Curve) score evaluates the model's ability to distinguish between classes. A value of 1 indicates a perfect classifier, and 0.5 indicates a random classifier.  
  
Mathematical Formulation:  
AUC measures the area under the ROC curve, which plots the True Positive Rate vs. False Positive Rate at various thresholds.  
  
Python Code:  
```python  
from sklearn.metrics import roc\_auc\_score  
roc\_auc = roc\_auc\_score(y\_true, y\_pred)  
print(roc\_auc)  
```

# Logarithmic Loss (LogLoss)

LogLoss quantifies the accuracy of a classifier by comparing the predicted probabilities to the actual labels. Lower values indicate better performance.  
  
Mathematical Formulation:  
LogLoss = - (1/N) \* sum [ y\_i \* log(p\_i) + (1 - y\_i) \* log(1 - p\_i) ]  
Where:  
y\_i is the true label.  
p\_i is the predicted probability of the positive class.  
  
Python Code:  
```python  
from sklearn.metrics import log\_loss  
y\_pred\_prob = [0.1, 0.9, 0.2, 0.8, 0.3, 0.7, 0.6]  
logloss = log\_loss(y\_true, y\_pred\_prob)  
print(logloss)  
```