<u>**Comparing different models: SVM, KNN, Logistic Regression, XGBoost**</u>

How to best evaluate a classification model?

- Classification accuracy
- Confusion matrix
- Precision and recall
- F1 score
- Sensitivity and specificity
- ROC curve and AUC

1. Classification accuracy

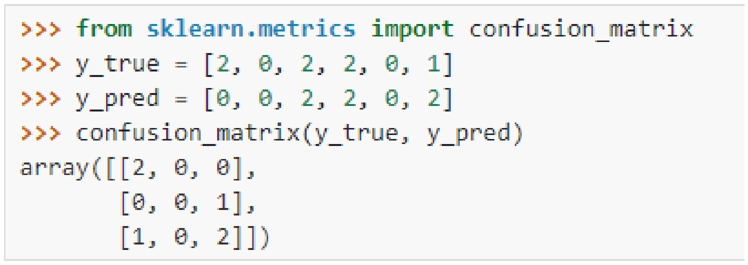
- Should be above 60%

The problem with only checking accuracy occurs when there is an unbalanced class distribution, this usually leads to a really high accuracy but isn't a good depiction of how well your model actually is at classifying. The following metrics are needed in order to truly gauge which classification model is best…

2. Confusion Matrix

- Identifies false negatives and false positives

How to implement: Confusion matrix can be implemented using scikitlearn, A basic code snippet looks like this:



```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

For more info, visit the following links:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

3. Precision and Recall

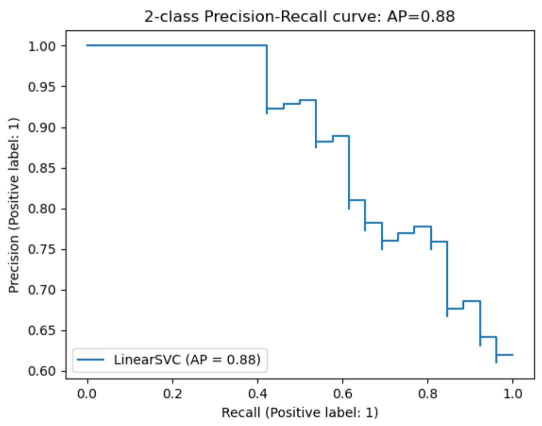
- <u>Precision</u> measures how good the model is when the prediction is positive. **(indicates how many positive predictions are true)**
- <u>Recall</u> is how good the model is when correctly predicting positive classes. (focus is on actual positive classes. **It indicates how many of the positive classes the model is able to predict correctly.**

There is a tradeoff between recall and precision and you cannot maximize both. For an email detection task for example, you want to maximize precision because you don't want false positives. In a case of tumor detection, you want to maximize recall because you want to accurately identify as many positive cases as you can.

A precision-recall curve shows the tradeoff. High scores for both means the model is returning accurate results (high precision) and a majority of all positive results (high recall).

In our dysarthria classifier, if diagnosing someone as healthy when they actually have Parkinson’s is a very bad outcome (false negative) , we might want to have a higher maximize recall.

Visual of precision-recall curve



How to implement:

You can also use scikitlearn to draw the curve and get the precision and recall:

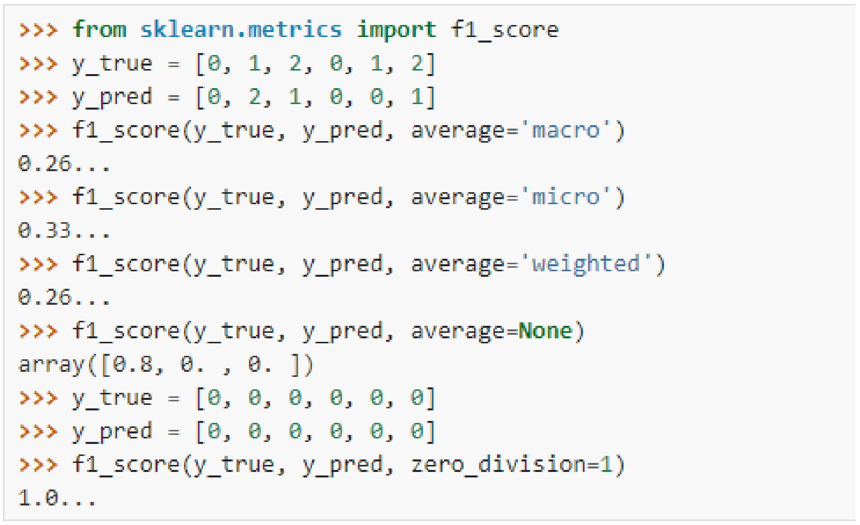
```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
```

More code snippets can be found here:

https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

4. F1 Score

- Weighted average of precision and recall
- F1 score is a more useful measure than accuracy for uneven distribution in classes
- Best value for F1 score is 1, and the worst value is 0



```
>>> from sklearn.metrics import f1_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> f1_score(y_true, y_pred, average='macro')
0.26...
>>> f1_score(y_true, y_pred, average='micro')
0.33...
>>> f1_score(y_true, y_pred, average='weighted')
0.26...
>>> f1_score(y_true, y_pred, average=None)
array([0.8, 0. , 0. ])
>>> y_true = [0, 0, 0, 0, 0, 0]
>>> y_pred = [0, 0, 0, 0, 0, 0]
>>> f1_score(y_true, y_pred, zero_division=1)
1.0...
```

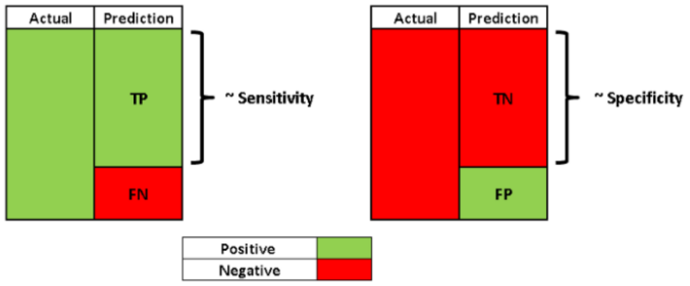
More info can be found at:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

5. Sensitivity and Specificity

- Sensitivity, also known as the true positive rate (TPR), is the same as recall (it measures the proportion of positive class that is correctly predicted as positive)
- Specificity is similar to sensitivity but focused on negative class. It measures the proportion of negative class that is correctly predicted as negative.



How to implement:

```
FP = confusion_matrix.sum(axis=0) - np.diag(confusion_matrix)
FN = confusion_matrix.sum(axis=1) - np.diag(confusion_matrix)
TP = np.diag(confusion_matrix)
TN = confusion_matrix.values.sum() - (FP + FN + TP)

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
```

More info at:

https://statinfer.com/204-4-2-calculating-sensitivity-and-specificity-in-python/

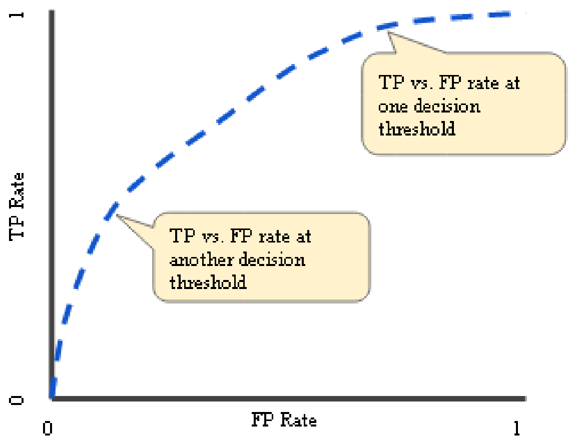
Note:

A higher value of sensitivity would mean higher value of true positive and lower value of false negative.

The higher value of specificity would mean higher value of true negative and lower false positive rate.

6. ROC or AUC

- ROC (receiver operating characteristics) summarizes the performance of the model at different threshold values by combining confusion matrices at all threshold values. (do not set the threshold to 0 or 1)



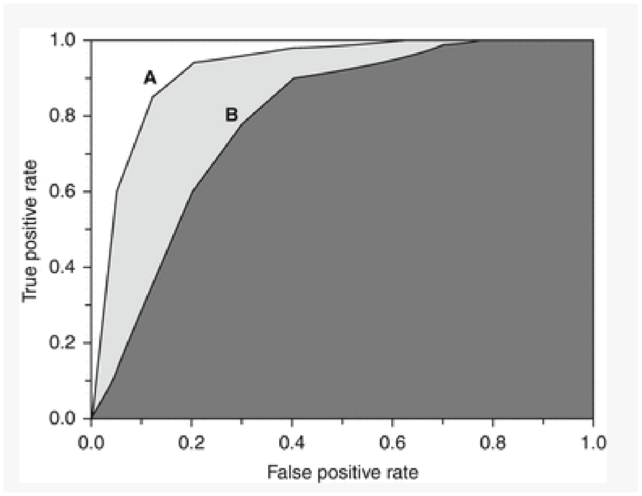
How to interpret

FP rate represents your False Positive Rate and TP rate is the True Positive rate. The aim is to keep the TP rate high when False positive rate stays low. As TPR increases, FPR also increases, so it comes down to how many false positives we can tolerate.

- AUC (area under the curve)

Instead of trying to find the best threshold for the AUC, we can instead try to calculate AUC. It is the area under ROC curve between (0,0) and (1,1) which can be calculated using integral calculus.

The best possible value of AUC is 1 which indicates a perfect classifier.



For example, Classifier A is better than Classifier B

How to implement:

We can plot a ROC curve for a model in Python using the roc_curve() scikit-learn function.
The AUC for the ROC can be calculated using the roc_auc_score() function.

More info can be found here: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

Implementation Summary for All Metrics:

https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/

Research paper that talks about these different measures on various classification models when classifying Parkinson’s:

https://www.ripublication.com/ijaer19/ijaerv14n2_17.pdf