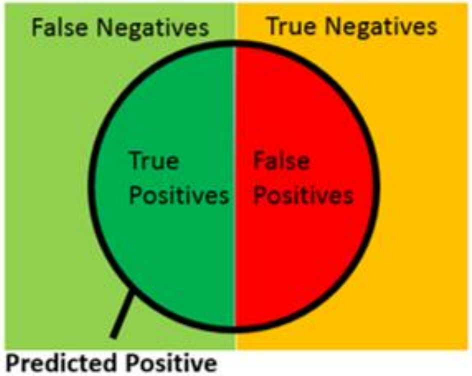
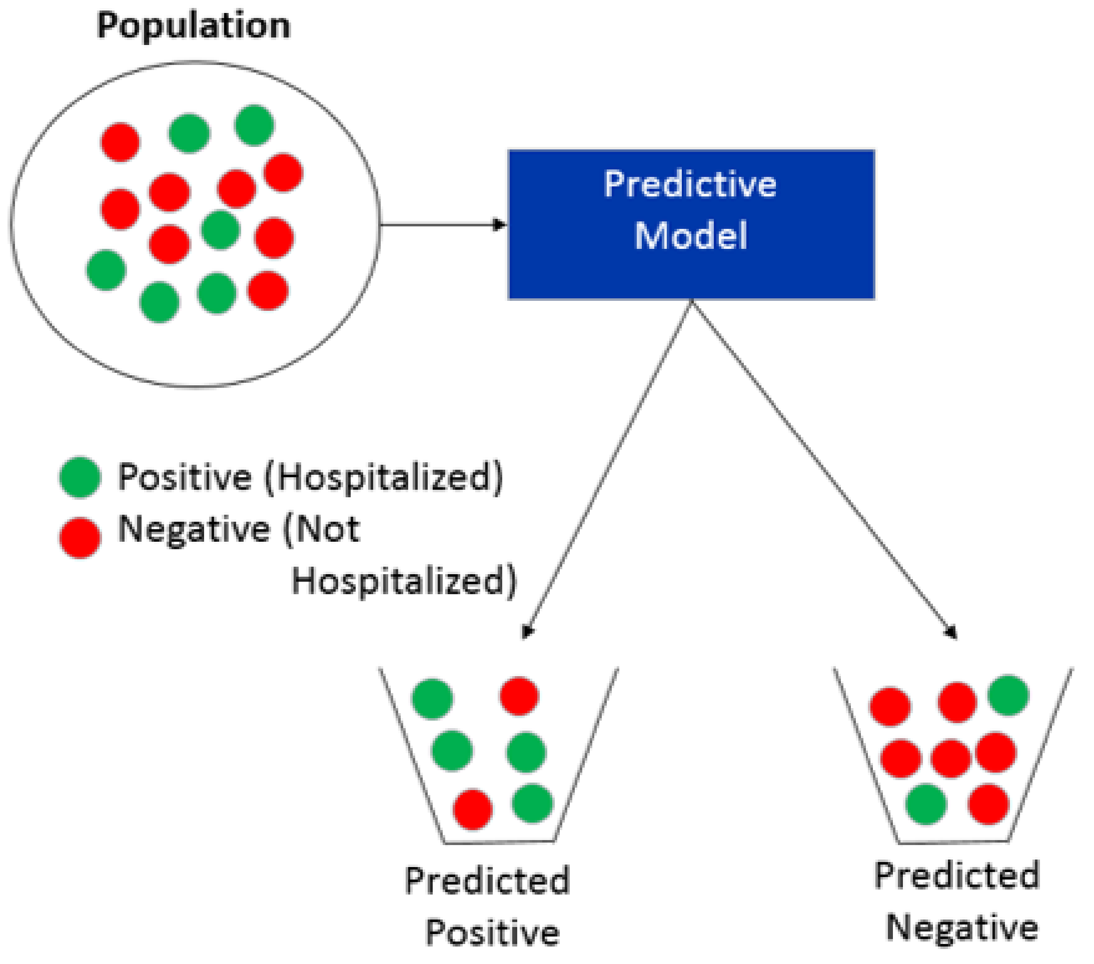
Classification Metrics in Scikit Learn

Metrics in Predictive Modelling

- One major area of predictive modeling in data science is classification. Classification consists of trying to predict which class a particular sample from a population comes from.
- For example, if we are trying to predict if a particular patient will be re-hospitalized, the two possible classes are hospital (positive) and not-hospitalized (negative).
- The classification model then tries to predict if each patient will be hospitalized or not hospitalized.
- In other words, classification is simply trying to predict which bucket (predicted positive vs predicted negative) a particular sample from the population should be placed as seen below.



Metrics in Predictive Modelling



Metrics in Predictive Modelling

- True Positives: people that are hospitalized that you predict will be hospitalized
- True Negatives: people that are NOT hospitalized that you predict will NOT be hospitalized
- False Positives: people that are NOT hospitalized that you predict will be hospitalized
- False Negatives: people that are hospitalized that you predict will NOT be hospitalized

Analysing Performance of models

- As you train your classification predictive model, you will want to assess how good it is. Interestingly, there are many different ways of evaluating the performance.
- Scikit-learn contains many built-in functions for analyzing the performance of models.

Classification Accuracy and its Limitations

- Classification accuracy is the ratio of correct predictions to total predictions made:

- It is often presented as a percentage by multiplying the result by 100.

  classification accuracy = correct predictions / total predictions * 100

- Classification accuracy can also easily be turned into a misclassification rate or error rate by inverting the value, such as:

  error rate = (1 - (correct predictions / total predictions)) * 100

Metrics for Evaluating Performance of the Models

1. Confusion Matrix

- confusion matrix is a technique for summarizing the performance of a classification algorithm.
- Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.
- Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.
- A confusion matrix is a summary of prediction results on a classification problem.
- The number of correct and incorrect predictions are summarized with count values and broken down by each class
- This is the key to the confusion matrix.

Metrics for Evaluating Performance of the Models

- **The confusion matrix shows the ways in which your classification model is confused when it makes predictions.**
- It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made.
- It is this breakdown that overcomes the limitation of using classification accuracy alone.

Metrics for Evaluating Performance of the Models

Process for calculating a confusion Matrix

- You need a test dataset or a validation dataset with expected outcome values.
- Make a prediction for each row in your test dataset.
- From the expected outcomes and predictions count:
  - The number of correct predictions for each class.
  - The number of incorrect predictions for each class, organized by the class that was predicted.

Metrics for Evaluating Performance of the Models

- These numbers are then organized into a table, or a matrix as follows:
- **Expected down the side**: Each row of the matrix corresponds to a predicted class.
- **Predicted across the top**: Each column of the matrix corresponds to an actual class.
- The counts of correct and incorrect classification are then filled into the table.

Confusion Matrix		Actual	
		Hospitalized	Not Hospitalized
Predicted	Hospitalized	33	10
	Not Hospitalized	17	40

2-Class Confusion Matrix Case Study

- Let’s pretend we have a two-class classification problem of predicting whether a photograph contains a man or a woman.
- We have a test dataset of 10 records with expected outcomes and a set of predictions from our classification algorithm.

Expected, Predicted
man, woman
man, man
woman, woman
man, man
woman, man
woman, woman
woman, woman
man, man
man, woman
woman, woman

2-Class Confusion Matrix Case Study

- Let’s start off and calculate the classification accuracy for this set of predictions.
- The algorithm made 7 of the 10 predictions correct with an accuracy of 70%.
- accuracy = total correct predictions / total predictions made * 100
- accuracy = 7 / 10 * 100
- But what type of errors were made?
- Let’s turn our results into a confusion matrix.
- First, we must calculate the number of correct predictions for each class.

2-Class Confusion Matrix Case Study

  - men classified as men: 3
  - women classified as women: 4
- We can now arrange these values into the 2-class confusion matrix:

	men	women
men	3	1
women	2	4

	Positive	**Negative**
Positive	True Positive	False Positive
Negative	False Negative	True Negative

- The total actual men in the dataset is the sum of the values on the men column (3 + 2)
- The total actual women in the dataset is the sum of values in the women column (1 +4).
- The correct values are organized in a diagonal line from top left to bottom-right of the matrix (3 + 4).
- More errors were made by predicting men as women than predicting women as men.

Two-Class Problems Are Special

- In a two-class problem, we are often looking to discriminate between observations with a specific outcome, from normal observations.
- Such as a disease state or event from no disease state or no event.
- In this way, we can assign the event row as "*positive*" and the no-event row as "*negative*". We can then assign the event column of predictions as "*true*" and the no-event as "*false*".
- This gives us:
- "**true positive**" for correctly predicted event values.
- "**false positive**" for incorrectly predicted event values.
- "**true negative**" for correctly predicted no-event values.
- "**false negative**" for incorrectly predicted no-event values.

Example of Confusion Matrix

- Given an array or list of expected values and a list of predictions from your machine learning model, the confusion_matrix() function will calculate a confusion matrix and return the result as an array. You can then print this array and interpret the results.
- from sklearn.metrics import confusion_matrix
- expected = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
- predicted = [1, 0, 0, 1, 0, 0, 1, 1, 1, 0]
- results = confusion_matrix(expected, predicted)
- print(results)

Shows:

- [[4 2]
- [1 3]]

Metrics for Evaluating Performance of the Models

2. Accuracy Score Metric

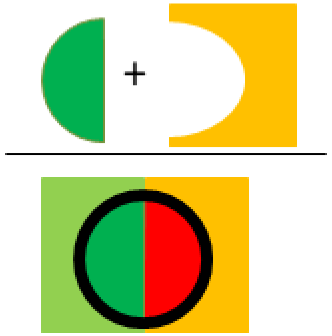
Accuracy_score which is imported as

- from sklearn.metrics import accuracy_score

returns "accuracy classification score". What it does is the calculation of "How accurate the classification is"

- It is the most common metric for classification which is the fraction of samples predicted correctly as shown below:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} =$$



Fraction predicted correctly

Metrics for Evaluating Performance of the Models

We can obtain the accuracy score from scikit-learn, which takes as inputs the actual labels and the predicted labels

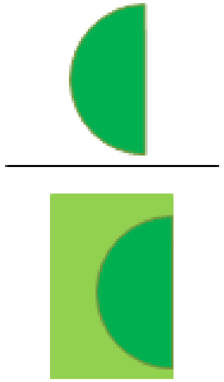
- from sklearn.metrics import accuracy_score
- accuracy_score(df.actual_label.values, df.predicted_RF.values)

- Shows answer like 0.6705165630156111

Metrics for Evaluating Performance of the Models

- 3. Recall Score Metric
- Recall (also known as sensitivity) is the fraction of positives events that you predicted correctly as shown below:
- from sklearn.metrics import recall_score
- recall_score(df.actual_label.values, df.predicted_RF.values)

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN} =$$



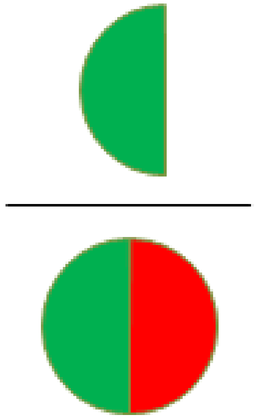
Fraction of positives predicted correctly

Metrics for Evaluating Performance of the Models

4. Precision Score Metric

- Precision is the fraction of predicted positives events that are actually positive as shown below:
- from sklearn.metrics import precision_score
- precision_score(df.actual_label.values, df.predicted_RF.values)

$$\text{Precision} = \frac{TP}{TP + FP} =$$



Fraction of predicted positives that are actually positive

Metrics for Evaluating Performance of the Models

5. F1 Score Metric

- The f1 score is the harmonic mean of recall and precision, with a higher score as a better model. The f1 score is calculated using the following formula:
- from sklearn.metrics import f1_score
- f1_score(df.actual_label.values, df.predicted_RF.values)

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

Conclusion

- In predictive analytics, when deciding between two models it is important to pick a single performance metric.
- As you can see here, there are many that you can choose from (accuracy, recall, precision, f1-score, AUC, etc).
- Ultimately, you should use the performance metric that is most suitable for the business problem at hand.

Titanic Project

- https://www.ritchieng.com/machine-learning-project-titanic-survival/