Performance Metrics in Machine Learning*

February 20, 2023

1 Introduction

In machine learning, it is key to be able to correctly evaluate the model being produced to guarantee that the predictions are accurately describing the intended phenomenon (disease prediction, future cost estimation, etc.). However, there are so many different performance metrics (accuracy, precision, recall, etc.) that it is often overwhelming to choose which one to use. Selecting the right metric for a specific model, however, is key to be able to measure the performance of the model objectively and in the right setting. In this notes, we will explore the different metrics, to be able to apply the most appropriate metrics for different tasks.

While classification and regression tasks form what's called supervised learning, clustering forms the majority of unsupervised learning tasks. The difference between these two macro-areas lies in the type of data used. While in supervised learning, samples are labeled with either a categorical label (for classification) or a numerical value (for regression), in unsupervised learning samples are not labeled, making it a relatively complex task to perform and evaluate.

2 Performance metrics for classification

2.1 Concepts

Before delving into the performance metrics themselves, it is important to make sure some concepts are clearly understood as they are consistently used across most performance metrics.

2.1.1 True Value vs Predicted Value

When evaluating the performance of a classification model, two concepts are key, the *real outcome* (usually called y) and the *predicted outcome* (usually called $\hat{y}$). For instance, a model can be trained to predict whether a person will develop a particular disease. In this case, it is trained with samples, e.g. a person's data, containing predictive information, such as age, gender, etc., and each person is labelled with a flag stating whether the disease will develop or not. In this case, the label can be whether the disease will happen ($y = 1$) or will not happen ($y = 0$).

*References

- Eugenio Zuccarelli. Performance Metrics in Machine Learning. 2020

A machine learning model aims at making sure that every time a sample is presented to it, the predicted outcome corresponds to the true outcome. The more the model's predictions are the same as the true values the higher is the performance of the model. There are many different ways of evaluating a model's performance, but in general, models make mistakes, lowering performance.

2.1.2 True Positive, True Negative, False Positive and False Negative

Each prediction from the model can be one of four types with regards to performance:

- **True Positive** (TP): A sample is **predicted to be positive** ($\hat{y} = 1$, e.g. the person is predicted to develop the disease) and its label is **actually positive** ($y = 1$, e.g., the person will actually develop the disease).
- **True Negative** (TN): A sample is **predicted to be negative** ($\hat{y} = 0$, e.g. the person is predicted to not develop the disease) and its label is **actually negative** ($y = 0$, e.g., the person will actually not develop the disease).
- **False Positive** (FP): A sample is **predicted to be positive** ($\hat{y} = 1$, e.g. the person is predicted to develop the disease) and its label is **actually negative** ($y = 0$, e.g., the person will actually not develop the disease). In this case, the sample is "falsely" predicted as positive.
- **False Negative** (FN): A sample is **predicted to be negative** ($\hat{y} = 0$, e.g. the person is predicted to not develop the disease) and its label is **actually positive** ($y = 1$, e.g., the person will actually develop the disease). In this case, the sample is "falsely" predicted as negative.

Even though the classes are usually labelled 1 and 0, these values are arbitrary and they can often be found labelled as 1 and -1, which is the reason "Positive" and "Negative" are used. Remembering False Positive and False Negative meanings is usually relatively tricky and it is common for data scientists to have to stop for a second to think about the meaning of each to remember which one represents what. An easy trick to remember the difference is focus first on the second part of the name ("Positive" or "Negative"). This relates to the prediction, basically saying "The sample is predicted to be Positive/Negative (belong to class 1/0)...". Then, we can look at the first part of the name to understand whether the prediction was correct or not ("True" or "False"). In this case, we are adding whether the prediction was correct or incorrect, and therefore if the sample was actually belonging to that class. For example, False Positive means that the sample is predicted to be Positive, but this is False/incorrect... as the sample is actually Negative.

2.1.3 Confusion Matrix

True Positive, True Negative, False Positive and False Negative are usually presented in a tabular format in the so-called **confusion matrix**, which is simply a table organizing the four values. Figure 1 shows such how the values are organized in a confusion matrix.

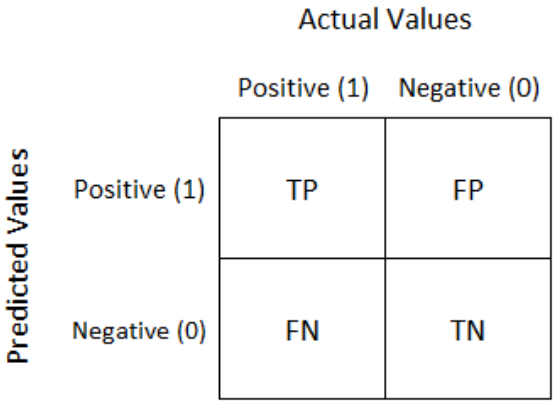

Figure 1: Confusion matrix

2.2 Performance Metrics

2.2.1 Accuracy

Accuracy is the fraction of predictions our model got right out of all the predictions. This means that we sum the number of predictions correctly predicted as Positive (TP) or correctly predicted as Negative (TN) and divide it by all types of predictions, both correct (TP, TN) and incorrect (FP, FN).

$$\text{Accuracy } = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

Accuracy ranges between 0 and 1. These extreme cases correspond to completely missing the predictions or having always correct predictions. For instance, if our model is able to perfectly predict, the model will have no False Positives or False Negatives, making the numerator be equal to the denominator, bringing the accuracy to 1. Conversely, if our system is always off, incorrectly predicting each time, the number of True Positives and True Negatives will be zero, making the equation be zero divided by something positive, leading to an accuracy equal to 0.

Accuracy, however, is not a great metric, especially when the data is imbalanced. When there is a significant disparity between the number of positive and negative labels, Accuracy does not tell the full story. For instance, let's consider an example where we have 100 samples, 95 of which labelled as belonging to class 0, and 5 labelled as class 1. In this case, a poorly built "dummy" model which always predicts class 0, achieves a 95% accuracy, which indicates a very strong model. However, this model is not really predictive and accuracy is not the right performance metric to evaluate the power of this model. If we used only accuracy to evaluate this model, we would end up providing stakeholders, and clients eventually, with a model that is not performant or predictive.

2.2.2 Precision

To overcome the limitations of Accuracy, we usually use Precision, Recall and Specificity. Precision tells what **proportion of positive predictions** was actually correct. It achieves this by counting the samples correctly predicted as positive (TP) and dividing

it by the total positive predictions, correct or incorrect (TP, FP).

$$\text{Precision } = \frac{TP}{(TP + FP)} \tag{2}$$

2.2.3 Recall (Sensitivity, True Positive Rate, Hit Rate)

Similarly to Precision, Recall aims at measuring what **proportion of actual positives** was identified correctly. It does so by dividing the correctly predicted positive samples (TP) by the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (TP, FN).

$$\text{Recall } = \frac{TP}{(TP + FN)} \tag{3}$$

2.2.4 Specificity (True Negative Rate, Selectivity)

Symmetrically to Recall, Specificity aims at measuring what **proportion of actual negatives** was identified correctly. It does so by dividing the correctly predicted negative samples by the total number of negatives, either correctly predicted as negative or incorrectly predicted as positive (TN, FP).

$$\text{Specificity } = \frac{TN}{(FP + TN)} \tag{4}$$

Considering the example to show the shortcomings of Accuracy, if we use Precision, Recall and Specificity, we get: Accuracy = 0.95 and Recall = 0. By using additional performance metrics instead of Accuracy, we can better understand that a model predicting the majority class all the time is actually a low-performance model (Recall = 0) even though Accuracy is high (Accuracy = 0.95).

2.2.5 Area Under the ROC Curve (AUC)

As we've seen, one of the issues of Accuracy is that it can lead to overly inflated performance if the distribution of the classes is not very well balanced. AUC, which stands for "Area Under the ROC Curve" (see Section 2.3.1), measures the entire two-dimensional area underneath the entire ROC curve. It is an aggregate measure of performance across all possible classification thresholds. Another way of interpreting AUC is as the probability that the model ranks a random positive sample higher than a random negative sample. AUC is a great metric, especially when dealing with imbalanced classes, and is one of the most frequently used performance measures in classification, even though it can be used only in binary classification settings (i.e. not with more than 2 classes as target). Some of the properties that make it a preferred metric are:

- **Scale-Invariance**. AUC measures how well predictions are ranked, instead of their absolute values.
- **Classification-Threshold-Invariance**. AUC measures the quality of the model's predictions regardless of what classification threshold is chosen.

2.2.6 F1 Score

The F1 score is a less known performance metric, indicating the harmonic mean of Precision and Recall. The highest value of an F1 Score is 1, indicating perfect Precision and Recall, and the lowest possible value is 0 if either the Precision or the Recall is zero.

$$F1 \text{ Score } = \frac{2TP}{(2TP + FP + FN)} \tag{5}$$

2.3 Performance Charts

Additionally, performance measures can be not only communicated as single numbers but also as charts. Some common charts showing a machine learning model's performance are the ROC Curve and the Precision/Recall Curve.

2.3.1 ROC Curve (Receiver Operating Characteristic Curve)

A ROC curve is a graph showing the performance of a classification model at all classification thresholds (see Figure 2). The chart's y-axis is the True Positive Rate, while the x-axis is the False Positive Rate and the plot consists of the TPR and FPR values varying the threshold. The worst-case scenario (random chance) consists of a 45 degrees diagonal line. The best-case scenario consists of an angled line, going vertically first and horizontally after. Lowering the classification threshold, the model classifies more items as positive, increasing both False Positives and True Positives.

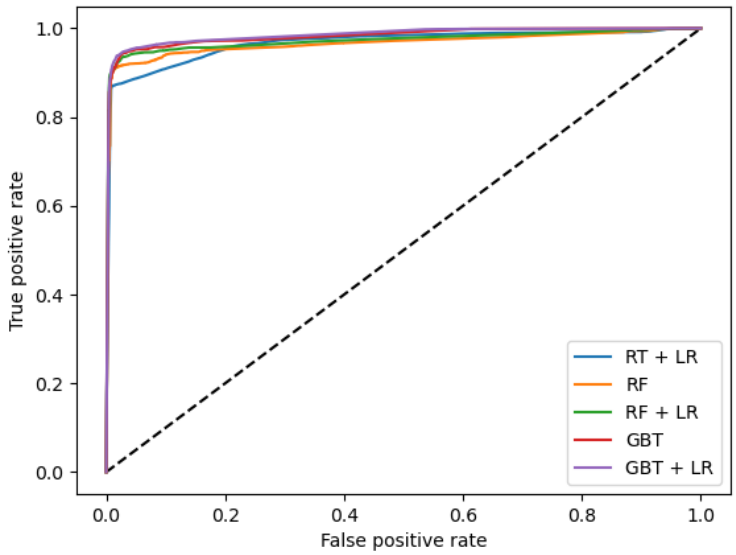

Figure 2: ROC curve

2.3.2 Precision/Recall Curve

Similarly to the ROC curve, a Precision/Recall curve plots performance over a y-axis showing Precision and an x-axis which is Recall (see Figure 3). Each point is evaluated at different threshold values. The best-case scenario is a flipped version of the ROC curve's best-case scenario, basically consisting of a horizontal line then becoming vertical. Differently, the worst-case scenario, random chance, is seen as a horizontal line at Precision = 0.5.

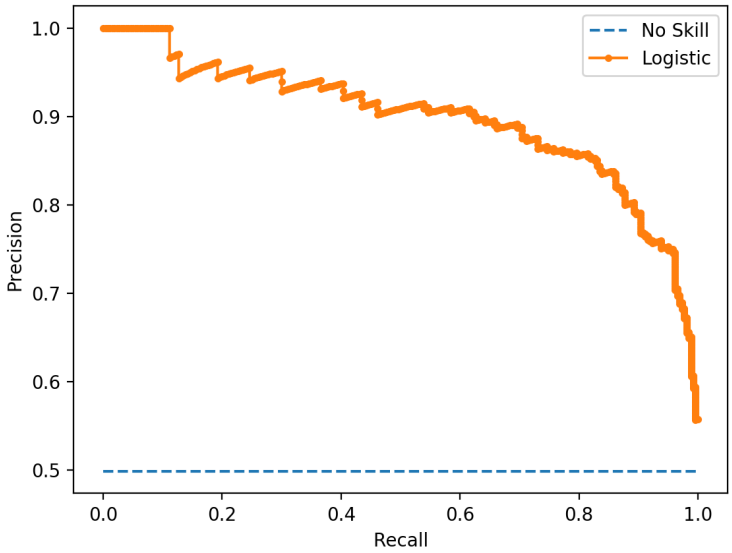

Figure 3: Precision/Recall curve

2.4 Impact of Choosing the Right Performance Metric

Choosing the right metric is key, especially in cases where False Positives and False Negatives do not have the same impact. Ideally, we would want to have a perfect prediction both in terms of False Positive and False Negative (both zero), but with machine learning models there is usually a tradeoff between detecting False Positives or False Negatives well. For instance, if our model predicts whether a person has got a deadly disease, like cancer, it could be said that False Positives are more important. We want to make sure that if that person has the disease, we correctly flag them. We are less concerned if we accidentally misclassify a person as having the disease even though they didn't have it. Conversely, if our model predicts whether a person is innocent or not, it might be argued that False Negatives are more important. We want to make sure that no innocent person is incorrectly jailed.

3 Performance metrics for linear regression

3.1 Mean Squared Error / Mean Squared Deviation

The Mean Squared Error (MSE) measures the average of the errors squared. It basically calculates the difference between the estimated and the actual value, squares these results and then computes their average. Because the errors are squared, MSE can only assume non-negative values. Due to the intrinsic randomness and noise associated with most processes, MSE is usually positive and not zero.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\,, \tag{6}$$

where y_i denotes the actual value, and $\hat{y}$ denotes predicted value. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated.

Similarly to the Variance, one major disadvantage of Mean Squared Error is that it is not robust to outliers. In case a sample has a "y" and associated error which is way larger than the other samples, the square of the error will be even larger. This, paired to the fact that MSE calculates the average of errors, makes MSE prone to outliers.

3.2 Root Mean Squared Error / Root Mean Squared Deviation

Similarly to the Mean Squared Error (RMSE), RMSE calculates the average of the squared errors across all samples but, in addition, takes the square root of the result, effectively taking the square root of MSE. By doing so, RMSE provides an error measure in the same unit as the target variable. For instance, if our target y is next year's sales in dollars, RMSE will give the error in dollars, while MSE would be in dollars squared, which is much less interpretable.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{7}$$

3.3 Mean Absolute Error

The Mean Absolute Error (MAE) does not take the square of the errors. Instead, it simply calculates the absolute value of the errors and then takes the average of these values. The MAE takes the absolute value as we are not interested in the direction in which the estimated and actual target values differ (estimated > actual or vice-versa) but on the absolute distance. This also avoids errors to cancel each other out when calculating the MAE.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{8}$$

Differently from MSE, MAE does not penalize larger errors more than smaller ones, because the formula for MAE does not apply the square to errors. Another advantage is that MAE does not square the units, similarly to RMSE, making the results more interpretable.

3.4 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) measures the error between actual and forecasted values as a percentage. It achieves so by calculating it similarly to MAE, but also dividing it by the actual value, expressing the result as a percentage, i.e.,

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}, \tag{9}$$

where y_i denotes the actual value, and $\hat{y}$ denotes predicted value.

By expressing the error as a percentage, we can have a better understanding of how off our predictions are in relative terms. For instance, if we were to predict next year's spending, an MAE error of \$50 could be both a relatively good or bad approximation. For instance, if the \$50 error was made with respect to an actual spending of \$1 million, we could safely say that the prediction is pretty good. Instead, if the error was on a \$60 cost prediction, it would be pretty far off from the actual value. In relative terms,

an error of \$50 against a \$1 million prediction is a 0.005% error. If this error was made on a \$60 prediction it would mean that the error is 83% of the predicted value (leading to basically a range of \$10 to \$110, almost reaching double the actual value). Using MAPE, in this case, shows a more accurate representation of the error with respect to the absolute values.

3.5 R Squared / Coefficient of Determination

R Squared (R^2) represents the proportion of the variance for the dependent variable y that's explained by the independent variables X. R^2 explains to what extent the variance of one variable explains the variance of the second variable. So, if the R^2 of a model is 0.75, then approximately 75% of the observed variation can be explained by the model's features.

R^2 is calculated by taking one minus the sum of squares of residuals divided by the total sum of squares,

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{10}$$

where SS_{res} denotes the *sum of squares of residuals* (or *residual sum of squares*), SS_{tot} denotes the *total sum of squares*, and $\bar{y}$ is the mean of the actual values.

R^2 compares the fit of the chosen model with that of a horizontal line, which acts as a baseline. If the chosen model fits worse than a horizontal line, the R^2 is negative. Because of the formula of R^2, even though the "square" is involved, it can have a negative value without violating any rules of math. R^2 is negative only when the model does not follow the trend of the data and fits worse than a horizontal line.

One of the drawbacks of R^2 is that the more features are added to a model, the more the R^2 increases. This happens even though the features added to the model are not intrinsically predictive.

3.6 Adjusted R Squared

For this reason, the Adjusted R^2 was introduced. It takes into account the features used in the predictive model. Doing so, the more predictive features are added to the model, the higher the Adjusted R^2. However, the more "useless" features are added to the model, the lower the Adjusted R^2 value, differently from what would happen with R^2. For this reason, the Adjusted R^2 is always less or equal the R^2 value.

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1} \tag{11}$$

where n is the number of data points and k is the number of features in the model.

3.7 Which metrics to yse?

Overall, it is usually important to report both an error measure, e.g. RMSE and an R^2 measure. This is because R^2 expresses the relation between the features X in the model and the target variable y. Error measures, instead, express how much spread out the data points are with respect to the regression fit. Reporting both Adjusted R^2 and RMSE, for instance, allows for a better comparison of the model against other benchmarks.

4 Performance metrics for clustering

For clustering, it often happens that clusters are manually and qualitatively inspected to determine whether the results are meaningful. In this section, we will go through the main metrics used to evaluate the performance of clustering algorithms, to rigorously have a set of measures.

4.1 Silhouette Score

The Silhouette Score and Silhouette Plot are used to measure the separation distance between clusters. It displays a measure of how close each point in a cluster is to points in the neighbouring clusters. This measure has a range of $[-1, 1]$ and is a great tool to visually inspect the similarities within clusters and differences across clusters.

The Silhouette Score is calculated using the mean intra-cluster distance (i) and the mean nearest-cluster distance (n) for each sample. The Silhouette Coefficient for a sample is

$$(n - i)/\max(i, n), \tag{12}$$

where n is the distance between each sample and the nearest cluster that the sample is not a part of while i is the mean distance within each cluster. The typical Silhouette Plots represent the cluster label on the y-axis, while the actual Silhouette Score on the x-axis. The size/thickness of the silhouettes is also proportional to the number of samples inside that cluster.

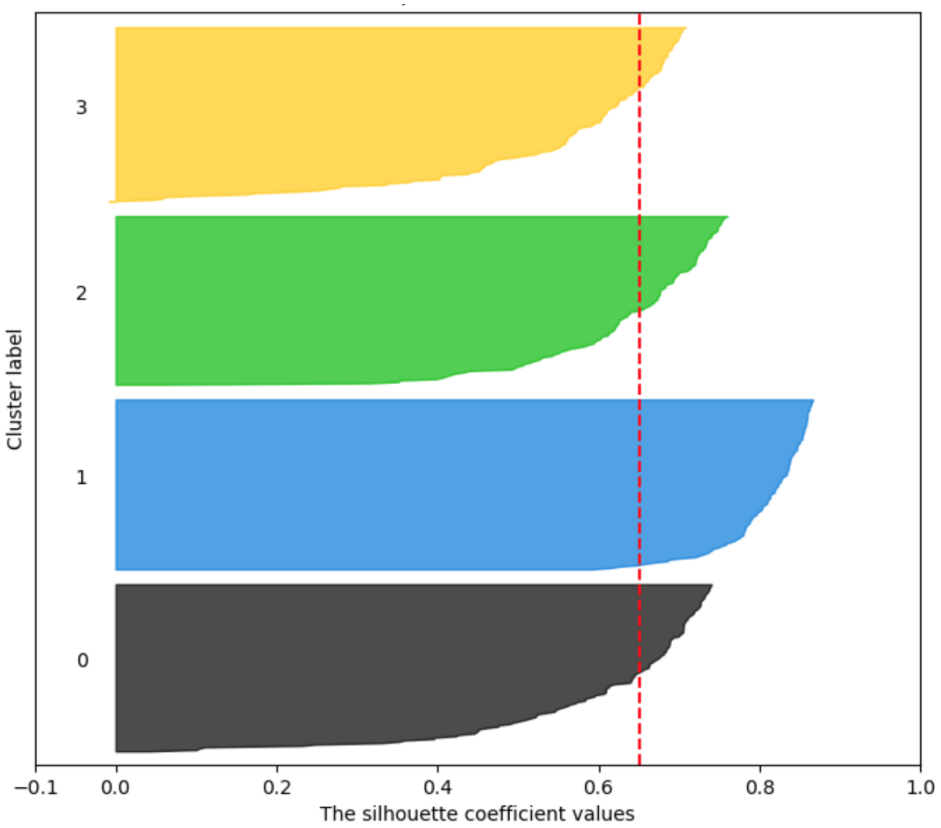

Figure 4: An example Silhouette Plot. On the y-axis, each value represents a cluster while the x-axis represents the Silhouette Coefficient/Score.

The higher the Silhouette Coefficients (the closer to +1), the further away the cluster's samples are from the neighboring clusters samples. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters. Negative values, instead, indicate that those samples might have been assigned to the wrong cluster. Averaging the Silhouette Coefficients, we can get to a global Silhouette Score which can be used to describe the entire population's performance with a single value.

Let's try to understand how the Silhouette Plot can help us find the best number of clusters by looking at the performance of each configuration:

Using "k=2", meaning two clusters to separate the population, we achieve an average Silhouette Score of 0.70 (see Figure 5).

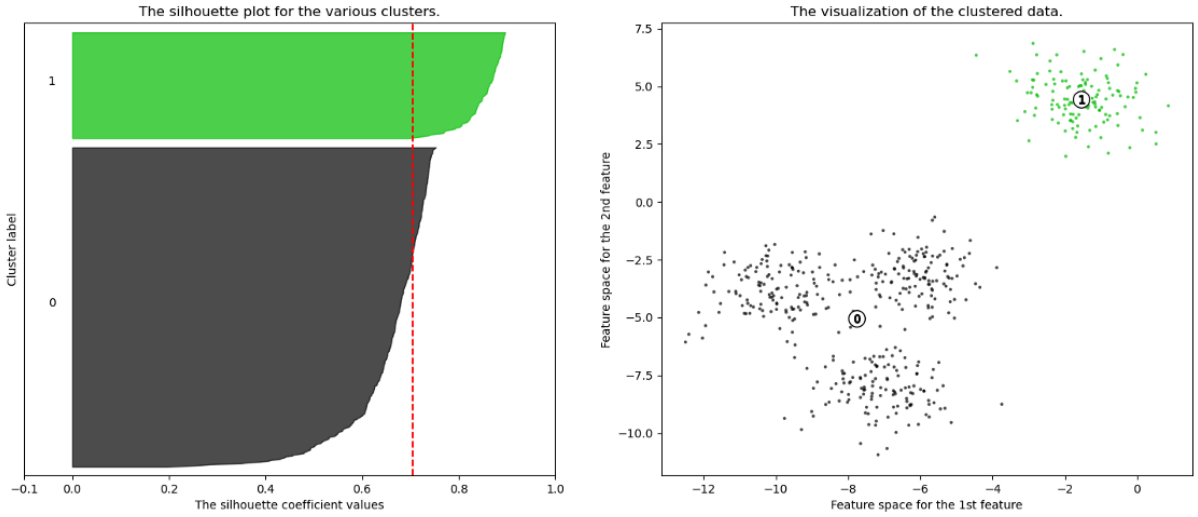

Figure 5: Silhouette analysis for k-means clustering on sample data with $k = 2$.

Increasing the number of clusters to three, the average Silhouette Score drops a bit.

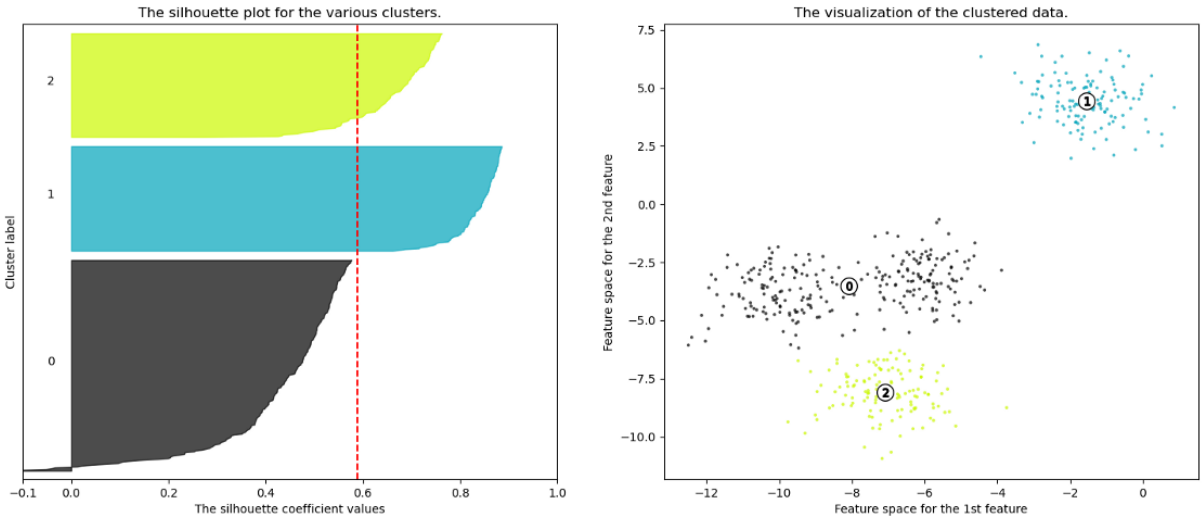

Figure 6: Silhouette analysis for k-means clustering on sample data with $k = 3$.

The same happens as the number of clusters increases (see Figures 7, 8, and 9). It can also be noticed that the thickness of the silhouettes keeps decreasing as the number of clusters increases, because there are less samples in each cluster.

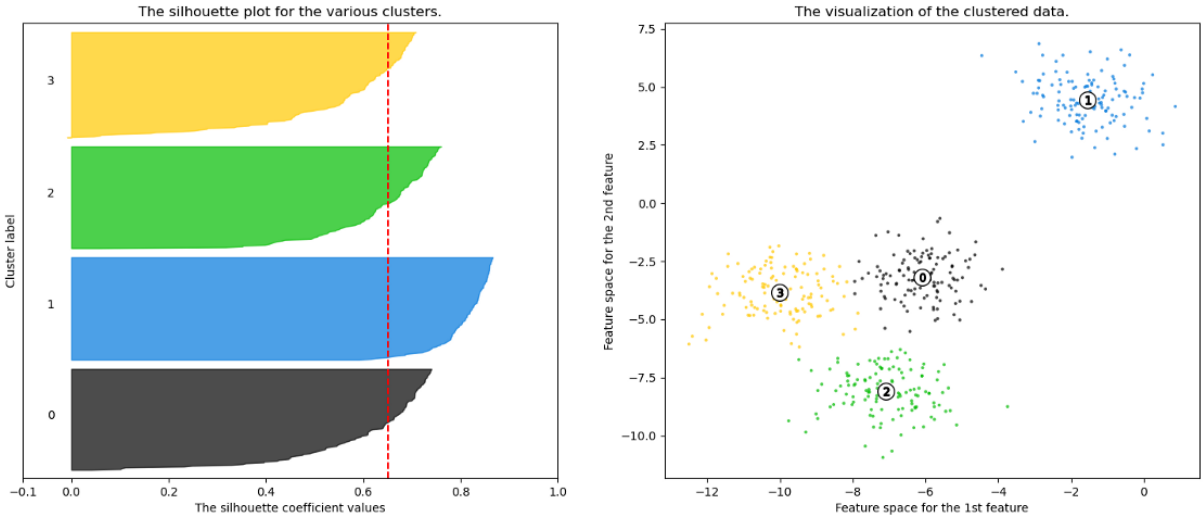

Figure 7: Silhouette analysis for k-means clustering on sample data with $k = 4$.

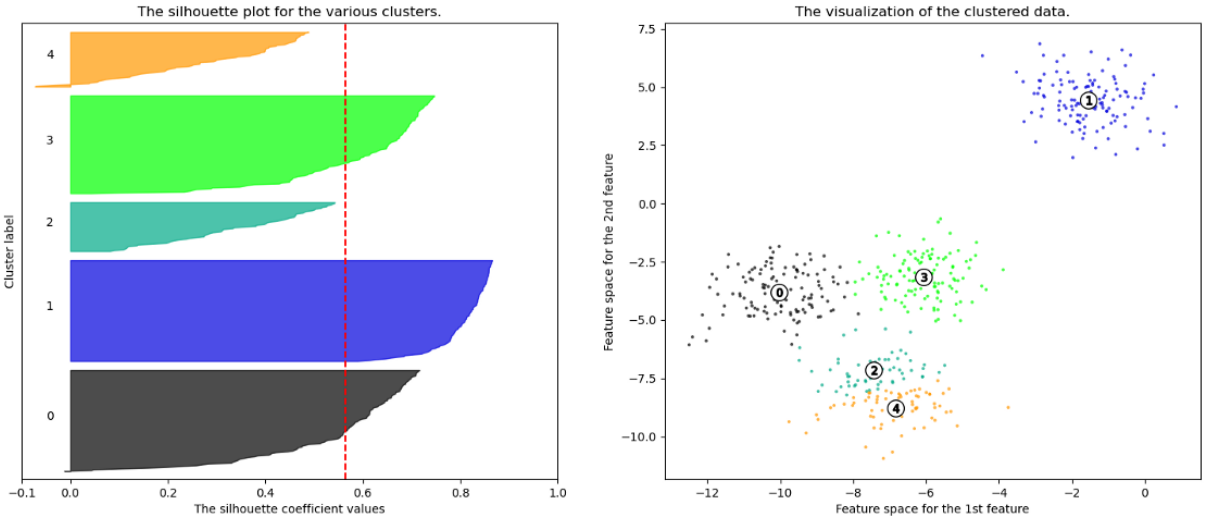

Figure 8: Silhouette analysis for k-means clustering on sample data with $k = 5$.

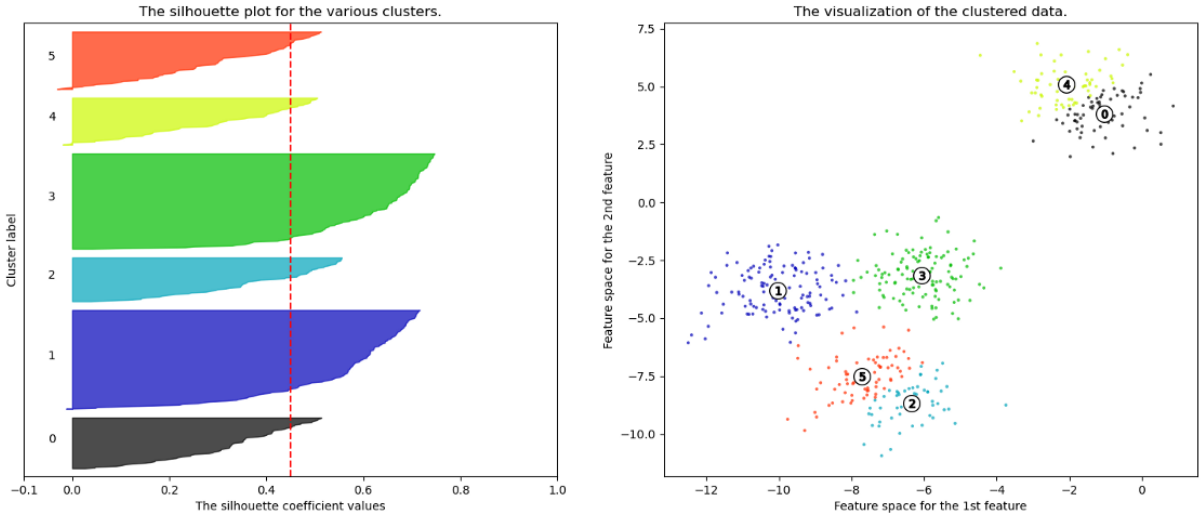

Figure 9: Silhouette analysis for k-means clustering on sample data with $k = 6$.

Overall, the average Silhouette Scores are:

```
For n_clusters = 2, the average silhouette_score is: 0.70
For n_clusters = 3, the average silhouette_score is: 0.59
For n_clusters = 4, the average silhouette_score is: 0.65
For n_clusters = 5, the average silhouette_score is: 0.56
For n_clusters = 6, the average silhouette_score is: 0.45
```

Having calculated the Silhouette Score for each possible configuration up to $k = 6$, we can see that the best number of clusters is 2, according to this metric and the higher the number of clusters the worse the performance becomes. To calculate the Silhouette Score in Python, you can simply use *scikit-learn* and do:

```
sklearn.metrics.silhouette_score(X, labels, *, metric='euclidean',
 sample_size=None, random_state=None, **kwds)
```

The function takes as input:

- X: An array of pairwise distances between samples, or a feature array, if the parameter "precomputed" is set to False.
- labels: A set of labels representing the label that each sample is assigned to.

4.2 Rand Index (RI)

Another commonly used metric is the Rand Index. It computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The formula of the Rand Index is:

$$RI = \frac{\text{Number of Agreeing Pairs}}{\text{Number of Pairs}} \tag{13}$$

The RI can range from 0 (the worst case) to 1 (a perfect match). The only drawback of Rand Index is that it assumes that we can find the ground-truth clusters labels and use them to compare the performance of our model, so it is much less useful than the Silhouette Score for pure unsupervised learning tasks.
To calculate the Rand Index, you canuse *scikit-learn* and do:

```
sklearn.metrics.rand_score(labels_true, labels_pred)
```

4.3 Adjusted Rand Index (ARI)

The Rand Index computes a similarity measure between two clusters by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. The raw RI score is then "adjusted for chance" into the ARI score using the following scheme:

$$ARI = \frac{\text{RI - Expected RI}}{\text{Max(RI)} - \text{ Expected RI}} \tag{14}$$

The Adjusted Rand Index, similarly to RI, ranges from 0 to 1, with 0 equating to random labelling and 1 when the clusters are identical. Similarly to RI, to calculate the ARI:

```
sklearn.metrics.adjusted_mutual_info_score(labels_true, labels_pred, *,
  average_method='arithmetic')
```

4.4 Mutual Information

The Mutual Information is another metric often used in evaluating the performance of clustering algorithms. It is a measure of the similarity between two labels of the same data. Where $|U_i|$ is the number of the samples in cluster U_i and $|V_j|$ is the number of the samples in cluster V_j, the Mutual Information between clusters U and V is given as:

$$MI(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N\,|U_i \cap V_j|}{|U_i|\,|V_j|} \tag{15}$$

Similarly to Rand Index, one of the major drawbacks of this metric is requiring to know the ground truth labels a priori for the distribution. Something which is almost never true in real-life scenarios with clustering.
Using *scikit-learn*:

```
sklearn.metrics.mutual_info_score(labels_true, labels_pred, *,
  contingency=None)
```

4.5 Calinski-Harabasz Index

Calinski-Harabasz Index is also known as the Variance Ratio Criterion. The score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. The C-H Index is a great way to evaluate the performance of a clustering algorithm as it does not require information on the ground truth labels. The higher the Index, the better the performance. The formula is:

$$s = \frac{\operatorname{tr}(B_k)}{\operatorname{tr}(W_k)} \times \frac{n_E - k}{k-1}, \tag{16}$$

where $\operatorname{tr}(B_k)$ is trace of the between group dispersion matrix and $\operatorname{tr}(W_k)$ is the trace of the within-cluster dispersion matrix defined by:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)\,(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q\,(c_q - c_E)\,(c_q - c_E)^T$$

To calculate it with *scikit-learn*:

```
sklearn.metrics.calinski_harabasz_score(X, labels)
```

4.6 Davies-Bouldin Index

The Davies-Bouldin Index is defined as the average similarity measure of each cluster with its most similar cluster. Similarity is the ratio of within-cluster distances to between-cluster distances. In this way, clusters which are farther apart and less dispersed will lead to a better score. The minimum score is zero, and differently from most performance metrics, the lower values the better clustering performance. Similarly to the Silhouette Score, the D-B Index does not require the a-priori knowledge of the ground-truth labels, but has a simpler implementation in terms of fomulation than Silhouette Score.

```
sklearn.metrics.davies_bouldin_score(X, labels)
```