Machine Learning—Evaluation (Cross-validation, Metrics, Importance Scores...)

Abdulhakim Qahtan

Abstract

The high performance of machine learning (ML) techniques when handling different data analytics tasks resulted in developing a large number of models. Although these models can provide multiple options for performing the task at hand, selecting the right model becomes more challenging. As the ML models perform differently based on the nature of the data and the application, designing a good evaluation process would help in selecting the appropriate ML model. Considering the nature of the ML model and the user's interest, different evaluation experiments can be designed to get better insights about the performance of the model. In this chapter, we discuss different evaluation techniques that suit both supervised and unsupervised models including cross-validation and bootstrap. Moreover, we present a set of performance measures that can be used as an indication on how the model would perform in real applications. For each of the performance measures, we discuss the optimal values that can be achieved by a given model and what should be considered as acceptable. We also show the relationship between the different measures, which can give more insights when interpreting the results of a given ML model.

A. Qahtan (✉)
Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
e-mail: a.a.a.qahtan@uu.nl

Keywords

Machine learning · Training · Testing · Regression · Classification · Clustering

Before discussing how to evaluate the Machine Learning (ML) models, we give a brief summary about the different models and how they work. Depending on the nature of the data and the task at hand, different machine learning models can be selected. These models are usually parameterized to automatically adjust their performance according to the data and the performance criteria through a set of tunable parameters. The values of the different parameters are learned and automatically adjusted during a training (fitting) stage of the model development. Learning the models' parameters can be achieved using one of three main approaches.

- **Supervised learning**: When the training set consists of labeled examples (exemplars), the algorithms use the labeled examples to learn how to generalize to the set of all possible inputs. Examples of techniques that belong to supervised learning category include logistic regression [3], support vector machines [6], neural networks [11] decision trees [22], random forest [6], etc.

- **Unsupervised learning**: Refers to the set of algorithms that learn from a set of unlabeled examples. These algorithms learn the patterns that exist in the data according to a specific criterion that could be statistical, geometric or similarity criterion. Examples of unsupervised learning include k-means clustering [5] and kernel density estimation [17].
- **Reinforcement learning**: In this set of algorithms, learning is achieved by iterative exploring the solution space and receiving a feedback on the quality of the solution. The exploration is repeated until a satisfactory performance measure value is reached.

The decision on using supervised/unsupervised learning technique will depend mainly on the availability of the labeled examples in the training set. In this chapter, we focus on the evaluation of the different machine learning techniques.

1 Background

Evaluation is a key and challenging task when selecting a Machine Learning (ML) model for a specific problem. There are lots of models that can be used, but which one will perform better than the others. This requires a systematic way for evaluating the different models. In this chapter, we will discuss the different measures for evaluating the ML models. We will restrict our discussion on the predictive models that include the regression models, the classifiers and the clustering algorithms.

Selecting the performance measure to evaluate a ML model should consider the problem at hand. Evaluating a supervised model should be based on comparing the value of the target variable that has been predicted by the model with the actual value. However, evaluating the unsupervised learning techniques is more challenging and is based on computing a set of statistical measures such as Silhouette score [13] in measuring the quality of a clustering algorithm. Moreover, in case of supervised learning, the evaluation measures that are used to evaluate the regression models are different from those that are used to evaluate the classifiers. Deciding whether to use a regression model or a classifier depends on the target variable that should be predicted. If the variable contains continuous values, a regression model should be used. Otherwise (when the variable contains a few distinct values that represent class labels of the data records), a classifier is trained for this purpose. Evaluating the regression models is carried out by measuring the difference between the actual and the predicted values. This difference is used as an indicator of the performance of the regression model. For classifiers, matching the predicted class label with the actual label of the record is used as an indicator of the performance of the classifier.

Example 1 Considering the diabetes dataset,[1] a classifier should be trained and used to predict if a person is diabetic or not based on the existing information. However, predicting the person's *weight* based on the *waist circumference*, which could be useful for validating the recorded data, requires building a regression model.

However, the main idea of building supervised ML models is to train the models on a training set that contains the data records and their corresponding target variable (the variable that should be predicted for new unseen records). When using the ML model to predict the value of the target variable for an unseen record, we should have a certain level of confidence on the correctness of the predicted value. Using the proper evaluation method helps in building such confidence. Moreover, when comparing the performance of different ML models, it is important to ensure that the apparent differences in the performance is not caused by chance.

To build a supervised ML model, a labeled set that contains records with values for the independent variables and their corresponding responses (values of the target variable) is required. A typical question that could be asked is: why we do not select the model that best fits the labeled data? To answer this question, we extract a set of ten values from the feature *waist circumference* and their corresponding values from the *weight* feature for training regression models. After that, we train three different regression models to fit the

[1] https://github.com/semerj/NHANES-diabetes.

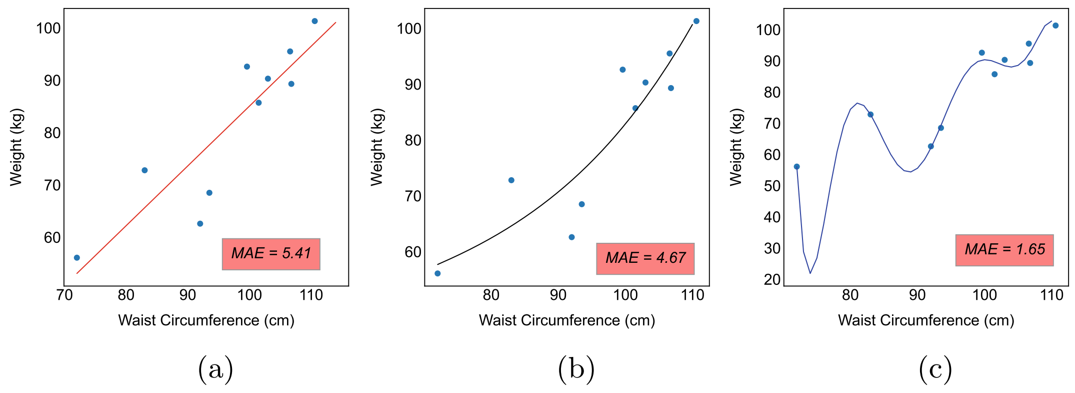

Fig. 1 Training three regression models **a** linear regression model, **b** polynomial regression model with degree 3 and **c** polynomial regression model with degree 7. In each subfigure, the mean absolute error between the actual values and the predicted ones is calculated and displayed in the red box inside the subfigure. Polynomial regression with degree 7 shows the smallest error

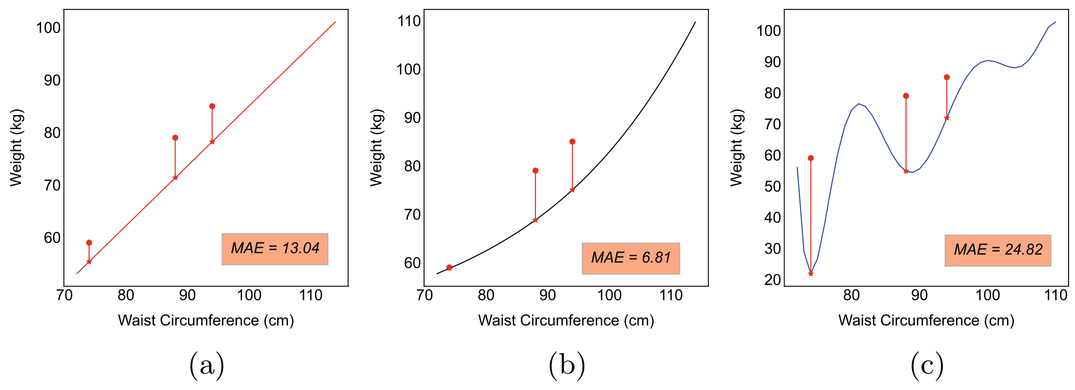

Fig. 2 Testing the three regression models on unseen data samples. In each subfigure, the mean absolute error between the actual values and the predicted ones is calculated and displayed in the red box in the subfigure. Polynomial regression with degree 3 shows the smallest error. The red vertical lines represent the difference between the actual test value (red circle) and the predicted values (red star on the regression curve)

training data as shown in Fig. 1. The Mean Absolute Error (MAE) [16] between the actual readings and the predicted ones is used as an indicator of the accuracy of the different models (MAE will be discussed later in the chapter). Comparing the MAE values in Fig. 1 (on the training data) with Fig. 2 (on the test data), it can be concluded that the model, which fits the training data the best is not necessarily the best model to be used for predicting new unseen values. This problem is a well-known problem and is called model over-fitting.

Based on the earlier discussion, the labeled data should be split into two parts (sometimes three parts) when building a supervised ML model. The first part is used for training the model and the second part is used to evaluate the model on data samples that have not been used during the training step. In some cases, a third subset of the data is used for parameter tuning of the model and is called the validation set. Evaluating the different ML models on values that have not been used during the training step is very important for comparing the different models and deciding which model to use. There are a lot of techniques that can be used to split the data into training and testing.

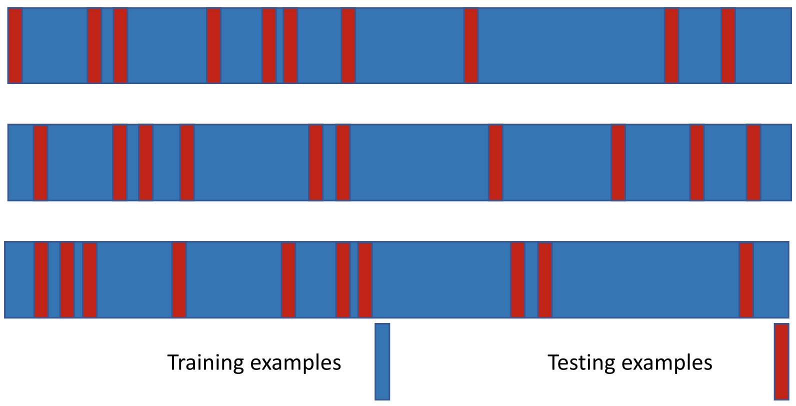

Fig. 3 Selecting random samples for training and testing from the labeled data

2 Train-Test Split

In this section, we discuss the different techniques that can be used to split the labeled data into training and testing in order to accurately estimate the performance of the ML models. The main idea is to split the labeled data into $x\%$ for training and $(100 - x)\%$ for testing (usually x is taken from the set $\{70, 75, 80\}$). The training subset is used to train (build) the ML model and the test subset is used to evaluate the performance of the model. In order to have a good estimation of the model's performance, this process is repeated multiple times and the average of the performance measure estimates is used as an indicator of the model's performance.

2.1 Random Split

For random sampling with $x\%$ for training and $(100 - x)\%$ for testing, a data example (record) from the labeled data is selected to be in the training set with probability $p = x/100$. Practically, this can be achieved by generating a random permutation for the index (sequence numbers of the records) and selecting the records with the first $x\%$ index values in the permutation. The rest of the records are assigned to the test set.

Alternatively, a random number generator can be used to generate random numbers between 0 and 100. For each record, the number r that is generated by the random number generator is compared to x and the record is selected to be in the training set if $r < x$; otherwise, the record is added to the test set. Figure 3, shows examples of splitting the labeled dataset into train and test substes randomly.

2.2 Split with Stratification

In a set of classification problems with imbalanced classes, splitting the labeled dataset into training and testing randomly may result with a training/testing dataset that contains records from only one class. For example, consider a data set of X-ray images, where the records are labeled as 0 if the person does not have cancer and 1 if the person has cancer. In such data set, the number of records that are labeled 1 is significantly smaller than those with label 0. If the training set contains only records with label 0, the trained model will not be able to recognize the records with label 1. Moreover, if test set contains records with label 0 only, the values of performance measures will be misleading.

To overcome such problem, train-test split with stratification [2] is introduced. This technique recognizes the different categories in the labeled data and generates the required ratio from each category. For example, to split the labeled data with $x\%$ for training, the labeled data is divided into a number of subsets equal to the number of classes and the records that belong to the same class fall in the same subset. After that, for each subset, an $x\%$ of the records in that subset are selected for training and the rest are held out to be used for testing.

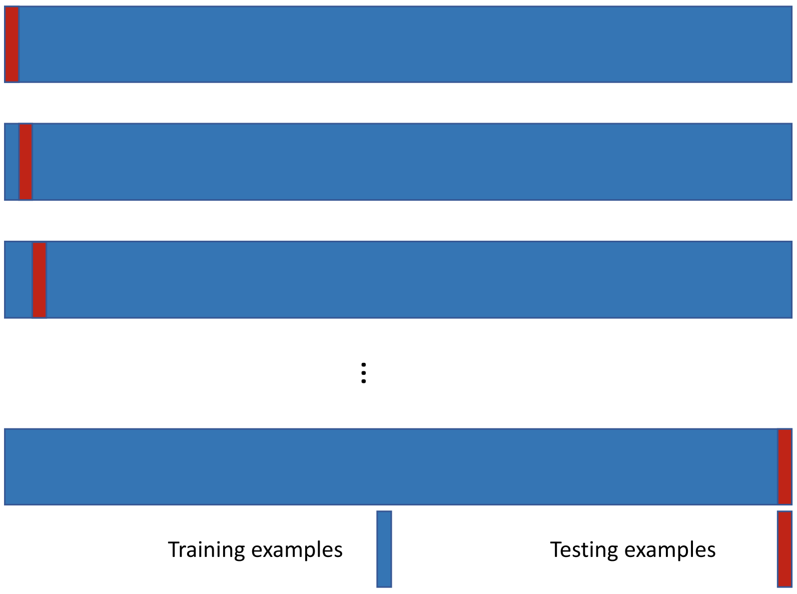

Fig. 4 Leave-one-out cross-validation for splitting the labeled dataset into training and testing

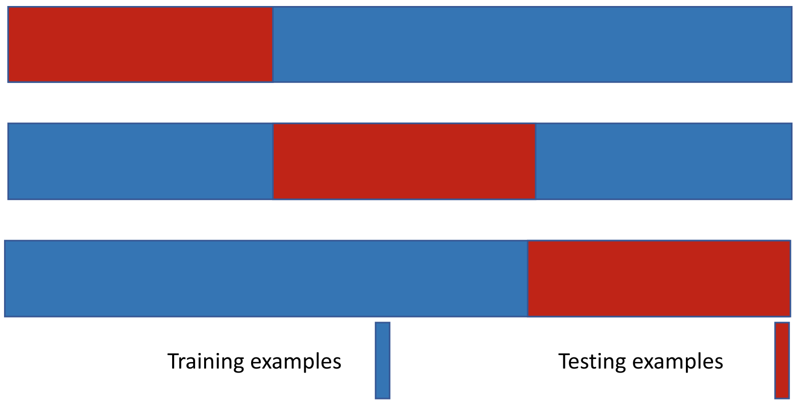

Fig. 5 Three fold cross-validation for splitting the labeled dataset into training and testing

2.3 Cross-validation

Cross validation [14] is one of the most popular techniques for train-test split. Such technique is based on performing the evaluation step multiple times where each single record in the labeled data is assigned at least once to the test set. In cross-validation, the user decides on a fixed number of folds or partitions of the data. The labeled data is then partitioned into that number of partitions. For example, if the user chose five folds, the labeled data is partitioned into five (approximately equal) partitions and each partition is used once for testing and the rest of the partitions are used for training. Figure 5, shows an example for a threefold cross-validation. A stratified 10-fold cross-validation is becoming a standard way of train-test split of the labeled data for the purpose of evaluating the ML models.

Another variation of the cross-validation is called leave-one-out cross-validation [15]. This technique is simply n-fold cross-validation when the labeled data set has n records. In this technique, each record is left out exactly once and the ML model is trained on the rest of the records. The record that is left out is used to test the model and the process is repeated n times to use each record for testing the model exactly once. This technique is preferred by the researchers, in many cases, as it maximizes the number of records that are used for training the model and the error estimation process is deterministic. Figure 4 shows an example of the leave-one-out cross-validation technique.

2.4 Bootstrap

Given a labeled data set with n records (examples), the idea of bootstrap [18] is to sample another data set with n records from the labeled data with replacement. That means, a record from the labeled data can be selected more than once. The new sampled data set is then used for training the ML model. Since sampling the new training set is done with replacement, a set of records will be repeated in the training set. Consequently, there will be a set of records in the original labeled data that have not been selected. These records are used for testing the model.

It can be shown that the probability of a record in the labeled data to be picked more than once is 0.368. That means, only 0.632 of the original data is used for training the model which is quite low compared to the 10-fold cross-validation where 90% of the labeled data is used for training the model. To compensate for this, a weighted average of the error on the training and the testing sets is used as an indicator of the model's performance. The final error is computed as

$$e = 0.632 \times e_{te} + 0.368 \times e_{tr},$$

where e_{te} is the error on the test set and e_{tr} is the error on the training set.

3 Evaluation Measures

As mentioned earlier, the evaluation measure that can be used to determine the performance of a given ML model depends on the nature of the data (labeled/unlabeled), the used ML model and the application. When the data is labeled, a supervised ML model can be used and the selection of the evaluation measure will depend on the nature of the predicted variable if it is continuous or categorical. Moreover, in many classification tasks, we might be interested in predicting the labels of a set of records that belong to a specific class more than the labels of the records that belong to the other classes. For example, predicting if a person is going to develop cancer accurately is more important than predicting if the person is not going to develop cancer. In such case, the performance measure should give more weight for correctly predicting the labels of the records from the desirable class. In the upcoming subsections, we will present and discuss different measures that can be used to evaluate the different ML models.

3.1 Evaluating the Supervised Models

In supervised learning, the ML model is trained to predict the value of the target variable using labeled data. We assume that the labels for the evaluation (test) set are also available so we can draw conclusions about the model's performance before using it in production applications for predicting the values of unseen examples (objects). As mentioned earlier, the labeled data is split into training and testing (sometime validation) sets. Comparing the predicted value with the actual value of the target variable is the logical step when evaluating the supervised ML models. Consequently, the selection of the performance measures that can be used to evaluate the supervised ML model depend on the nature of the target variable.

Evaluating the Regression Models Regression models are used to predict the values of continuous target variables. For example, predicting the blood pressure based on the lab results of a patient requires using a regression model. Let us consider that $y = \{y_1, y_2, \ldots, y_n\}$ represent the set of actual values of the target variable in the test set and $\hat{y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$ represent the set of predicted values. We compute the difference between the actual values and their corresponding predicted values $e_i = |y_i - \hat{y}_i|, \quad 1 \leq i \leq n$. We define a set of performance measures based on the values of e_i. The most common measures are the mean absolute error (MAE), sum/mean of squared error (SSE/MSE), the l_∞ and the coefficient of determination R^2.

The mean absolute error is defined as $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$. It represents the arithmetic average of the absolute error between the actual and the predicted values. This measure is usually used for computing the forecast error in time series analysis. However, it is used in a lot of applications as an indicator of the regression models' performance. The optimal value for MAE is 0; However, when comparing different regression models, the regression with smallest value for the MAE is considered better than the other models.

The sum/mean squared error are computed as $SSE = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$ and $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$. It is clear that SSE/MSE are the arithmetic summation/average of the squared difference (absolute error) between the actual and the predicted values. Similar to MAE, a value close to 0 means that the model is accurate. However, this measure reduces the contribution of the error values that are close to 0 and gives more weight to the error values that are greater than 1.

In order to make sure that the regression model provides accurate predictions for all examples in the test set, a measure called l_∞ is proposed. The $l_\infty = \max\limits_{1\le i\le n}|y_i - \hat{y}_i|$ error is computed as the maximum value of the absolute error e_i, $1 \le i \le n$. This measure is used to highlight the worst performance of the model.

The coefficient of determination is denoted by R^2 [10] and represents the proportion of the variance in the target variable that is predictable from the independent (determinant or exploratory) variables. It is a measure of how the regression equation accounts for the variation in the dependent variable. It is well-known that the closer the regression curve to the points, the better the models fits the data. The main idea behind R^2 is to determine if a regression model can utilize the knowledge from the independent variables to predict the dependent (target) variable accurately. To compute the R^2 value, we start by considering the average value of the target variable as our baseline predictor. After that, we compute the deviation of the predicted values of the regression model from the mean value and from the actual values, i.e, we compute three quantities

$$\begin{aligned} TSS &= \sum_{i=1}^{n}(y_i - \bar{y})^2 \\ RSS &= \sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2 \\ ESS &= \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \end{aligned} \qquad (1)$$

where $\bar{y}$ is the mean of the target variable y in the test set, TSS refers to the sum of squared deviation and RSS refers to the sum of squared deviation between the predicted values using the regression model and the mean of the target variable. Moreover, ESS is the sum of squared deviation between the actual and the values that predicted using the regression model. From Eq. (1), we can see that $TSS = RSS + ESS$. The coefficient of determination R^2 is then computed as

$$R^2 = \frac{RSS}{TSS} = \frac{TSS - ESS}{TSS} = 1 - \frac{ESS}{TSS}. \qquad (2)$$

The optimal score for R^2 is 1.0, which can be achieved when the value of the term $ESS \to 0$. Smaller values for R^2 means that the model is not accurate.

In Table 1, we summarize the measures that can be used to evaluate the regression models. It is clear that the main term in each measure is the difference between the actual and the predicted value. Usually, we need to compare two different learning models to see which one performs better on a specific problem. To have a better indication on the performance of the different models, we need to apply the techniques that we mentioned earlier such as cross validation and repeat the tests multiple times to choose the model that

Table 1 Summary of regression evaluation measures

Measure	Formula
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
MSE	$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$
SSE	$\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$
l_∞	$\max_{1\le i\le n}\lvert y_i - \hat{y}_i\rvert$
R^2	$1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$

gives the lower estimated error. However, we need also to check if the difference in the error is not happening by chance due to the randomness in the process. We leave this issue as it is out of the scope of this chapter.

Evaluating the Classifiers Classifiers predict a categorical value for each data example that represents the class label for that example. Based on this property, evaluating the classification models can be done by matching the predicted class label with the actual one and counting the number of examples that have the same value for the predicted and the actual labels. The large number of correct predictions indicates high performance of the classification model.

In the case of two class problem, we can consider the labels for the classes to be positive and negative or 0 and 1. The possible outcomes for matching a predicted class label with the actual one can be one of the the following:

- True positive (TP): the actual and the predicted labels are positive.
- False positive (FP): the actual label is negative while the predicted label is positive.
- False negative (FN): the actual label is positive while the predicted label is negative.
- True negative (TN): both (actual and predicted) labels are negative.

These different outcomes are usually summarized in matrix form, which is called the *confusion matrix* [21] (see Table 2). In Table 2, we see the confusion matrix for two classes with entries labeled as TP, FP, FN, and TN. From the confusion matrix, a set of performance measures can be defined. The basic performance measure for a classifier is its accuracy, which can be defined as follows

Table 2 The confusion matrix

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

$$\text{Accuracy (acc)} = \frac{\text{number of correct prediction}}{\text{size of the test set}} \tag{3}$$

In terms of the confusion matrix entries, the accuracy can be written as

$$\begin{aligned}\text{Accuracy (acc)} &= \frac{TP + TN}{TP + FP + FN + TN} \\ &= \frac{TP + TN}{n},\end{aligned} \tag{4}$$

where n is the number of examples in the test set. As we can see, all misclassification errors are given the same weight. However, optimizing the classifiers to have better accuracy values is usually misleading. This is a well-known problem when the class distribution of the samples (examples) is imbalanced [1]. By imbalanced data, we refer to the situation when the representative samples of the classes are unevenly distributed [19]. Let us consider the example in [4], where a set of images are labeled as either cancerous or noncancerous. The annotation resulted in labeling 10,923 images as noncancerous (majority class) and 260 as cancerous (minority class). When optimizing the ML model for accuracy, the model will tend to classify more examples to be noncancerous. If the ML model classifies every sample to be from the noncancerous (majority) class, it will achieve 99.98 % accuracy. However, in many applications, it is more costly to misclassify the examples from the minority class. For this reason, a set of measures have been proposed to tackle the imbalanced data problem and give better indications about the classifiers' performance.

In the information retrieval community, the *precision (P)* and *recall (R)* [20] are used to evaluate the performance of the information retrieval systems. When the user sends a query to retrieve a set of documents that are related to a specific topic, the precision represents the ratio of the correctly retrieved documents that are related to the topic in the set of the retrieved documents, whereas the recall represents the ratio of correctly retrieved documents in the set of he related documents in the whole dataset. In terms of the confusion matrix entries, we can consider:

- TP = the number of relevant retrieved documents
- FP = the number of irrelevant retrieved documents
- FN = the number of relevant unretrieved documents
- TN = the number of irrelevant unretrieved documents

Using this analogy, the precision and recall can be defined as:

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

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

Since, the precision and the recall can be expressed in terms of the entries in the confusion matrix, they have been used for measuring the performance of the classifiers. Moreover, a measure that combines the values of precision and recall is called the *F-Measure* or *F-Score* [7] is also used for measuring the classifiers' performance. In its generic form, the F-Score is defined as:

$$F_\beta = (1 + \beta^2)\frac{P \times R}{(\beta^2 \times P) + R} \quad ,$$

where P, R are the precision and recall and β is a parameter that controls the importance of the recall compared to the precision when computing the F-Score. When $\beta > 1$, the recall has more weight than the precision and vice versa. The balance between precision and recall is achieved when $\beta = 1$. In this case, F-Score represents the harmonic mean between P and R and is written as:

$$F_1 = 2\frac{P \times R}{P + R} \ .$$

Usually, the discussion of the precision, recall and F-Score focuses on measuring the performance of the ML models with respect to the positive class (+ve). However, they can be used to measure the performance with respect to the negative class (−ve). We can write:

$$P(+ve) = \frac{TP}{TP + FP} \quad \text{and} \quad P(-ve) = \frac{TN}{TN + FN}.$$

Similarly:

$$R(+ve) = \frac{TP}{TP + FN} \quad \text{and} \quad R(-ve) = \frac{TN}{TN + FP}.$$

In general, if we have k classes, we can compute k different values for each of the precision, recall and F-Score as they are associated with the class labels.

In the data mining community, the $R(+ve)$ is also known as the *sensitivity* of the ML model while the $R(-ve)$ is known as the *specificity*. The sensitivity and the specificity are used to define another performance measure that is more suitable for evaluating the ML models on biased data, which is called *the balanced accuracy* and is defined as follows:

$$\begin{aligned}\text{Balanced Accuracy (BA)} &= \frac{\text{sensitivity} + \text{specificity}}{2} \\ &= \frac{R(+ve) + R(-ve)}{2}\end{aligned}$$

The balanced accuracy provides a better measure for the performance of the ML models when the dataset is biased (there is a significant difference between the number of representative examples from each class in the dataset).

Example 2 Consider the case of training an ML model for classifying an input record to be cancerous or not using the dataset in [4]. We have 10,923 examples from noncancerous (U) and 260 examples from the cancerous class (C). If we split the dataset using the 70-30 rule (70% for training and 30% for testing) with stratification then we will have the number of records in each subset as in Table 3. We train an ML model (M) using the training set and test it on the test set. We present the output of the ML model in Table 4.

Table 3 Train-test split of the cancer dataset with stratification

	Cancerous (C)	Noncancerous (U)
Training	182	7646
Testing	78	3277

Table 4 The confusion matrix that represents the results of testing M using the test set of the cancer dataset

		Prediction	
		C	U
Actual	C	47	31
	U	327	2950

Table 5 Performance measures of the ML model on the cancer dataset

Acc.	$P(C)$	$P(U)$	$R(C)$	$R(U)$	$F_1(C)$	$F_1(U)$	BA
89.33	12.57	98.96	60.26	90.02	20.80	94.28	75.14

The reported values are out of 100, where 100 is the optimal value

The values for the different performance measures that are used to evaluate the ML model (M) are presented in Table 5. As we can see in the results, considering the values of the measures that are computed for the class (C) are lower than those for the class (U). Moreover, small changes in the classification outcomes would lead to significant changes in the values of the performance measures when considering the class (C) as it has a few examples. The accuracy is an exception.To show that, assume that your ML model is able to classify all the examples from the class (C) correctly. In this case, the value of the accuracy will be $acc. = 90.25\%$ whereas the balanced accuracy will be $BA = 95\%$. As we can see, the balanced accuracy increased by 20% while the accuracy increased by less than 1%. Hence, the balanced accuracy can be considered as a better classification performance measure when the data is biased as it gives more weight for correctly classifying an example from the minority class.

It is worth noting that these measures are easily extendable for the cases when we have more than two classes. To show that, let us consider a dataset that contains k classes $C = \{C_1, C_2, \ldots, C_k\}$. In this case, the confusion matrix can be constructed as in Table 6 and the accuracy can be computed as the summation of the values in the main diagonal over the summation of all entries in the confusion matrix. That is

Table 6 The confusion matrix for the case of k classes

		Prediction			
		C_1	C_2	$\ldots$	C_4
Actual	C_1	C_{11}	C_{12}	$\ldots$	C_{1k}
	C_2	C_{21}	C_{22}	$\ldots$	C_{2k}
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
	C_k	C_{k1}	C_{k2}	$\ldots$	C_{kk}

$$acc. = \frac{\sum_{i=1}^{k} C_{ii}}{\sum_{i=1}^{k} \sum_{j=1}^{k} C_{ij}}$$

Moreover, the precision and recall can be expressed as

$$P(C_i) = \frac{C_i}{\sum_{j=1}^{k} C_{ij}} \quad \text{and} \quad R(C_i) = \frac{C_i}{\sum_{j=1}^{k} C_{ji}},$$

where $P(C_i)$ is the precision and $R(C_i)$ is the recall with respect to (w.r.t) class C_i. The other measures can be expressed in a similar way.

3.2 Evaluating the Unsupervised Models

In unsupervised learning, the training data is not labeled. In this case, there is no error or reward that can help in optimizing the ML model. Instead, the ML techniques learn the patterns that exist in the data in order to categorize the examples (objects) according to a specific geometric or statistical criteria. The ML models are trained to summarize the key features or structures of the data by optimizing for the specified criteria. For example, clustering is an unsupervised technique that tends to increase the intra-cluster similarity and reduce the inter-cluster similarity. The different clustering techniques (algorithms) try to satisfy this criteria using different optimization functions. In this section we will focus on evaluating the clustering techniques since they are the most widely used unsupervised learning techniques.

As the clustering techniques tend to group similar items (objects) together, a similarity measure should be introduced that can determine how two objects are similar to each other. For numerical attributes, the *Minkowski distance* is a well-known similarity measure between the objects. The Minkowski distance is defined as

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p},$$

where $d(\mathbf{x}, \mathbf{y})$ is the distance between two objects $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, d is the data dimensionality and $\mathbb{R}$ is the set of real numbers. For categorical data, a match/mismatch or string dissimilarity functions can be used as dissimilarity (distance) measures. These dissimilarity measures are also used to define similarity measures to determine the membership level of an object to a given cluster. We use the similarity measures to compute *Silhouette coefficient* [13], which is used to measure the quality of a given clustering technique.

When evaluating the different clustering techniques, it is important to define a measure that can check if two clustering techniques (algorithms) produce similar groups (clusters) of the objects in the data. For this purpose, a measure called *Rand Index* is proposed in [12]. To explain how rand index can be used to compare two clustering algorithms, assume that we have a set $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ of objects (examples) that needs to be clustered (grouped). We use two different clustering algorithms $\mathcal{A}_1, \mathcal{A}_2$ to cluster the data into $\mathcal{A}_1(\mathbf{X}) = \{A_{11}, A_{12}, \ldots, A_{1r}\}$ and $\mathcal{A}_2(\mathbf{X}) = \{A_{21}, A_{22}, \ldots, A_{2s}\}$. There are four different types of relations that can be found between any pair of elements in the set $\mathbf{X} \times \mathbf{X}$ (Cartesian product of $\mathbf{X}$ with itself)

$$\begin{aligned}
\Gamma_1 &= \{(\mathbf{x_i}, \mathbf{x_j}) \ : (\exists p, \exists q),\ \mathbf{x_i}, \mathbf{x_j} \in A_{1p} \ \wedge \ \mathbf{x_i}, \mathbf{x_j} \in A_{2q}\} \\
\Gamma_2 &= \{(\mathbf{x_i}, \mathbf{x_j}) \ : (\exists p, \forall q),\ \mathbf{x_i}, \mathbf{x_j} \in A_{1p} \ \wedge \ \mathbf{x_i}, \mathbf{x_j} \notin A_{2q}\} \\
\Gamma_3 &= \{(\mathbf{x_i}, \mathbf{x_j}) \ : (\forall p, \exists q),\ \mathbf{x_i}, \mathbf{x_j} \notin A_{1p} \ \wedge \ \mathbf{x_i}, \mathbf{x_j} \in A_{2q}\} \\
\Gamma_4 &= \{(\mathbf{x_i}, \mathbf{x_j}) \ : (\forall p, \forall q),\ \mathbf{x_i}, \mathbf{x_j} \notin A_{1p} \ \wedge \ \mathbf{x_i}, \mathbf{x_j} \notin A_{2q}\},
\end{aligned}$$

where $p \in \{1, 2, \ldots, r\}$ and $q \in \{1, 2, \ldots, s\}$. Let $\gamma_l = |\Gamma_l|, \ 1 \le l \le 4$, then the values for $\gamma_l, \ 1 \le l \le 4$ can be interpreted as follows: i) γ_1 represents the cardinality of the set that contains the pairs of objects which fall in the same cluster using both algorithms $\mathcal{A}_1, \mathcal{A}_2$; ii) γ_2 is the number of pairs of objects in $\mathbf{X}$ that are in the same cluster according to algorithm $\mathcal{A}_1$, but in different clusters according to algorithm $\mathcal{A}_2$; iii) γ_3 is the number of pairs of elements in $\mathbf{X}$ that are in different clusters according to algorithm $\mathcal{A}_1$, but in the same cluster according to algorithm $\mathcal{A}_2$; and iv) γ_4 is the number of pairs of objects in $\mathbf{X}$ that fall in different clusters according to both algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$. Based on these quantities, the rand index is computed as

$$RI = \frac{\gamma_1 + \gamma_4}{\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4}.$$

The RI takes values in the interval [0, 1], where 1 represents the optimal value and means that both algorithms divided the original dataset $\mathbf{X}$ into the same set of clusters. When $RI = 0$, then the two algorithms are completely different. However, when assigning the objects in $\mathbf{X}$ into clusters randomly, the value of RI will not be 0, which requires correction-for-chance that has been proposed in [8] to define the adjusted random index (ARI). The ARI can be written as [9]

$$ARI = \frac{\binom{n}{2}(\gamma_1 + \gamma_3) - [(\gamma_1 + \gamma_4)(\gamma_1 + \gamma_2) + (\gamma_2 + \gamma_3)(\gamma_3 + \gamma_4)]}{\binom{n}{2}^2 [(\gamma_1 + \gamma_4)(\gamma_2 + \gamma_3) + (\gamma_2 + \gamma_3)(\gamma_3 + \gamma_4)]},$$

where $n = |\mathbf{X}|$ and $\binom{n}{2}$ is the total number of pairs. We consider that the pairs $(\mathbf{x_i}, \mathbf{x_j})$ and $(\mathbf{x_j}, \mathbf{x_i})$ are equal so they are counted only once.

When the dataset $\mathbf{X}$ is labeled, we can select algorithm $\mathcal{A}_1$ as the dummy clustering algorithm that assigns each object in $\mathbf{X}$ to its class and creates a number of clusters that is equivalent to the number of the classes in the dataset. In this case, the value of RI that is used to compare a given clustering algorithm $\mathcal{A}_2$ with the dummy algorithm $\mathcal{A}_1$ is exactly the accuracy that we discussed when evaluating the classification techniques earlier. Based on this observation, the other performance measures that we defined to evaluate the classification techniques, can be used to evaluate the clustering algorithms when the labels of the object are available. However, the performance measures have been given different names when they are used to evaluate the clustering algorithms. For example, the precision is called *purity* or *homogeneity*, the recall is called the *completeness* and the F-Score is called the *V-measure*.

When the labels of the objects in the dataset are available, we count the number of objects that belong to each class in a given cluster and associate that cluster with the class which includes the majority of the objects. A cluster is said to satisfy the purity (homogeneity) criterion if all the values in that cluster belong to the same class. Moreover, a cluster is said to satisfy the completeness criterion if all examples that belong to the class associated with that cluster are included in the cluster. To compute the purity and completeness of cluster $\mathbb{C}_i$, we assume that $\mathbb{C}_i$ is associated with class C_j.[2] In this case, the purity of $\mathbb{C}_i$ is defined as $purity(\mathbb{C}_i) = \frac{|\{\mathbf{x}:\mathbf{x}\in\mathbb{C}_i \wedge \mathbf{x}\in C_j\}|}{|\mathbb{C}_i|}$ and the completeness is defined as $completeness(\mathbb{C}_i) = \frac{|\{\mathbf{x}:\mathbf{x}\in\mathbb{C}_i \wedge \mathbf{x}\in C_j\}|}{|C_j|}$. The V-measure is defined similar to the F_1-Score as follows

$$\text{V-measure} = \frac{2 \times purity \times completeness}{purity + completeness}.$$

The purity, completeness and V-measure take values in the interval $[0, 1]$ where 1 represents the optimal outcome of the clustering algorithm. Obviously, these measures will not take the value of 0 in case of random clustering. Instead, their values will increase as the number of clusters increases, which could give misleading indication about the goodness of the clustering algorithm. However, this problem can be overcome when the number of the objects in the dataset $\mathbf{X}$ is large and the number of clusters is small.

Another measure that can be used to evaluate the goodness of the clustering algorithm is the *Silhouette coefficient* [13]. This measure determines how similar an example is to the examples in its own cluster compared to the examples in the other clusters without using the labels in the dataset. To compute the Silhouette coefficient for a given object (example $\mathbf{x_i}$) in the dataset, we compute two quantities $a(\mathbf{x_i})$ and $b(\mathbf{x_i})$ as follows:

$$a(\mathbf{x_i}) = \frac{1}{|\mathbb{C}_k| - 1} \sum_{\mathbf{x_j}\in\mathbb{C}_k \wedge \mathbf{x_j}\neq\mathbf{x_i}} d(\mathbf{x_i}, \mathbf{x_j}),$$

where $\mathbb{C}_k$ is the cluster that contains the object $\mathbf{x_i}$ and $d(\mathbf{x_i}, \mathbf{x_j})$ is a dissimilarity (distance) measure. The quantity $a(\mathbf{x_i})$ represents the average distance between $\mathbf{x_i}$ and all other objects in the same cluster. We define $\alpha_{im}(\mathbf{x_i}, \mathbb{C}_m)$ to be the average dissimilarity between the object $\mathbf{x_i}$ and all other objects in the cluster $\mathbb{C}_m$. That is

$$\alpha_{im}(\mathbf{x_i}, \mathbb{C}_m) = \frac{1}{|\mathbb{C}_m|} \sum_{\mathbf{x_j}\in\mathbb{C}_m} d(\mathbf{x_i}, \mathbf{x_j}).$$

Assuming that we have λ clusters, we select $b(\mathbf{x_i})$ to be the minimum value of $\alpha_{it}(\mathbf{x_i}, \mathbb{C}_t)$, $1 \le t \le \lambda \wedge t \neq k$. Using the quantities $a(\mathbf{x_i})$ and $b(\mathbf{x_i})$, we define the Silhouette coefficient for $\mathbf{x_i}$ as:

$$Sil(\mathbf{x_i}) = \frac{b(\mathbf{x_i}) - a(\mathbf{x_i})}{\max(b(\mathbf{x_i}), a(\mathbf{x_i}))}.$$

The Silhouette coefficient takes values in the interval $[-1, 1]$, where a value of 0 means that the object is in the border between two clusters, a negative value means that the object is more similar to objects in the nearest cluster than the objects in its own cluster. The average Silhouette coefficient over all objects in a given cluster determines the goodness of the cluster where a value close to 1 would mean a compact cluster. The average Silhouette coefficient over all clusters defines the quality of the clustering algorithm.

4 Conclusion

In this chapter, we provided a brief summary about the different machine learning approaches including the supervised, unsupervised and reinforcement learning. We introduced different performance measures that can be used to evaluate the ML models. Our main focus was on the regression, classification and clustering as these are the most widely used ML techniques. We showed that the values of some measures can be misleading when the dataset has specific characteristics. For example, the accuracy is not a good measure for

[2] We use the symbol $\mathbb{C}$ to represent a cluster while the regular C is used to represent a class.

the classification performance when the dataset is biased (the majority of the examples in the dataset belong to one class). Consequently, selecting the performance measure should consider the ML technique, the characteristics of the dataset and the task at hand. A good performance measure would lead to better optimization of the ML model to produce high quality results especially in fields where ML models have high impact people's lives such as in the health domain.

References

1. A comprehensive data level analysis for cancer diagnosis on imbalanced data. J Biomed Inf. 2019;90.
2. César CC, Carvalho MS. Stratified sampling design and loss to follow-up in survival models: evaluation of efficiency and bias. BMC Med Res Methodol. 2011;11(1):1–9.
3. Cox DR. The regression analysis of binary sequences. J Roy Stat Soc: Seri B (Methodol). 1958;20(2):215–32.
4. Fotouhi S, Asadi S, Kattan MW. A comprehensive data level analysis for cancer diagnosis on imbalanced data. J Biomed Inf. 2019;90.
5. Hartigan JA, Wong MA. A k-means clustering algorithm. JSTOR: Appl Stat. 1979;28(1):100–108.
6. Ho TK. Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. vol. 1. IEEE; 1995. p. 278–282.
7. Hripcsak G, Rothschild AS. Agreement, the f-measure, and reliability in information retrieval. J Ame Med Inf Assoc. 2005;12:296–8.
8. Hubert L, Arabie P. Comparing partitions. J Classif. 1985;2(1):193–218.
9. Igual L, Seguí S. Introduction to data science. 2017.
10. Lewis-Beck C, Lewis-Beck M. Applied regression: an introduction, vol. 22. Sage Publications; 2015.
11. McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys. 1943;5(4):115–33.
12. Rand WM. Objective criteria for the evaluation of clustering methods. J Am Stat Assoc. 1971;66(336):846–50.
13. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65.
14. Sammut C, Webb GI, editors. Cross-validation. Boston, MA, USA: Springer; 2010.
15. Sammut C, Webb GI editors. Leave-one-out cross-validation. 2010. p. 600–601.
16. Sammut C, Webb GI editors. Mean absolute error. 2010.
17. Scott DW. Multivariate density estimation: theory, practice, and visualization. Wiley; 1992.
18. Stine R. An introduction to bootstrap methods: examples and ideas. Soc Methods Res. 1989;18(2–3):243–91.
19. Sun Y, Wong AK, Kamel MS. Classification of imbalanced data: a review. Int J Pattern Recognit Artif Intell. 2009;23(04):687–719.
20. Ting KM. Precision and recall. 2010.
21. Ting KM. Confusion matrix. Boston, MA, USA: Springer; 2017.
22. Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Philip SY, et al. Top 10 algorithms in data mining. Knowl Inf Syst. 2008;14(1):1–37.