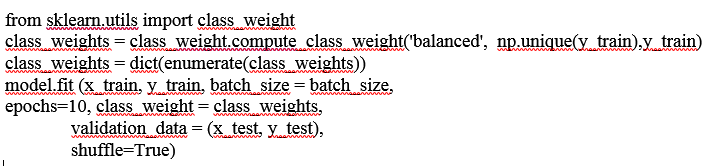
There are several ways to measure a classifier’s generalization quality, i.e.using ([Szymanski](https://arxiv.org/search/cs?searchtype=author&query=Szyma%C5%84ski%2C+P) et al. 2017):

* [*Hamming loss*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html#sklearn.metrics.hamming_loss) that measures how well the classifier predicts each of the labels, averaged over samples, then over labels;
* [*accuracy score*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score) that measures how well the classifier predicts label combinations, averaged over samples;
* [*jaccard similarity*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_similarity_score.html#sklearn.metrics.jaccard_similarity_score) that measures the proportion of predicted labels for a sample to its correct assignment, averaged over samples;
* [*precision*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score) (also called *specificity*) that is a measure of quality and measures how many retrieved samples are relevant: ; the *true positive rate* (TPR) is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive;
* [*recall*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score)(also called *sensitivity, or detection rate* (DR)) that is a measure of quantity and measures how many relevant samples were retrieved: ; the *true negative rate* (TNR) is calculated as the number of true negatives divided by the sum of the number of true negatives and the number of false positives. It describes how good the model is at predicting the negative class when the actual outcome is negative;
* [*F1* score](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score) that measures a weighted average of precision and recall, where both have the same impact on the score.

*Accuracy* (i. e. the number of correct predictions divided by the total number of predictions) can be a useful measure of model performance if we have the same amount of samples per class. If the problem is **balanced** and we **care about both positive and negative predictions, accuracy is a good choice** because it is simple and easy to interpret. But in case of imbalanced data set, accuracy isn't useful. Even more so, a test can have a high accuracy but actually perform worse than a test with a lower accuracy. Sometimes we can compute the accuracy individually per every class by giving the classification function only samples from the same class and remember and count the number of correct classifications and incorrect classifications, and then compute . For imbalanced data sets such performance measures as F1 score and/or Brier score are needed. Both accuracy and F1 score take class predictions as input so we have to adjust the threshold regardless of which one you choose. [Brownlee](https://machinelearningmastery.com/author/jasonb/) J. (2020) also describes approaches that address the problem of imbalanced data sets in machine learning. Amongst them, *cost sensitive learning* is mentioned. In *cost-sensitive learning*, instead of each instance being either correctly or incorrectly classified, each class (or instance) is given a misclassification cost. Instead of trying to optimize the accuracy, the problem is then to minimize the total misclassification cost. Thus, the goal of cost-sensitive learning is to minimize the cost of a model on the training dataset assuming that different types of prediction errors have a different and known associated cost ([Brownlee](https://machinelearningmastery.com/author/jasonb/) J. 2020). The [*Keras Python Deep Learning library*](https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/) provides access to the use of cost-sensitive augmentation for neural networks via the argument on the  [function](https://keras.io/models/sequential/) when training models. Example of the Python code using class weights is provided in the Fig. 1.11.



**Fig. 1.11.** Reguliarisation in case of high biase and high variance

Classifiers can often provide uncertainty estimates of predictions. In practice, we are often interested in how certain a classifier is about each class prediction. Most learning methods can return at least one measure of confidence in their predicions (Vanschoren 2022):

* decision function - the return value of the decision function encodes how strongly the model believes a data point belongs to the “positive” class: the higher the more confident); positive values indicate preference for the positive class; by default, we threshold at 0 for decision function,
* probability: estimated probability for each class; by default, we threshold at 0.5 in case we predict probabilities (they sum up to 1).

Lower threshold yields fewer FN (better recall), more FP (worse precision), and vice-versa.

The *Receiver Operator Characteristic* (ROC) curve may be used as an evaluation metric for binary classification problems. ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: *True Positive Rate* and *False Positive Rate*. It is a plot of the *false positive rate* (x-axis) versus the *true positive rate* (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate. The ROC curve is used to assess the overall diagnostic performance of a test and to compare the performance of two or more diagnostic tests. It is also used to select an optimal cut-off value for determining the presence or absence of an event. The ROC curve shows the trade-off between *sensitivity* (or TPR) and *specificity* (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test. When we predict a binary outcome, it is either a correct prediction (true positive) or not (false positive). There is a tension between these options, the same with true negative and false negative:

* smaller values on the x-axis of the plot indicate lower false positives and higher true negatives;
* larger values on the y-axis of the plot indicate higher true positives and lower false negatives.

A skillful model will assign a higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. This is what we mean when we say that the model has skill. Generally, skillful models are represented by curves that bow up to the top left of the plot. A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases. A model with no skill is represented at the point A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an Area Under Curve (AUC) of . A model with perfect skill is represented at a point . A model with perfect skill is represented by a line that travels from the bottom left of the plot to the top left and then across the top to the top right. Summarizing, the area under the ROC curve (AUC) gives the best overall model. It is frequently used for evaluating models on imbalanced data. What is common between ROC AUC and PR AUC is that they both look at prediction scores of classification models and not thresholder class assignments. What is different however is that **ROC AUC looks at**a true positive rate **TPR and** false positive rate **FPR** while **PR AUC looks at** positive predictive value (**PPV)** and true positive rate (**TPR)**. Because of that**if we care more about the positive class, then using PR AUC**, which is more sensitive to the improvements for the positive class, is a better choice.

While ROC curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds, a *Precision-Recall* curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds. They often are being used in information retrieval and are treated as an alternative to ROC curves for tasks with a large skew (e. i. distortion) in the class distribution. ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets, i. e. when there is a moderate to large class imbalance. The reason for this recommendation is that ROC curves present an optimistic picture of the model on datasets with a class imbalance. The main reason for this optimistic picture is because of the use of True Negatives in the False Positive Rate in the ROC Curve and the careful avoidance of this rate in the Precision-Recall curve. If the proportion of positive to negative instances changes in a test set, the ROC curves will not change. Metrics such as accuracy, precision, recall and F1 scores use values from both columns of the confusion matrix. As a class distribution changes these measures will change as well, even if the fundamental classifier performance does not. ROC graphs are based upon TP rate and FP rate, in which each dimension is a strict columnar ratio, so do not depend on class distributions.

Vanschoren (2022) states that in binary classification, we have a positive and a negative class and therefore 2 different kind of errors that may be not equally important: false positive (type I error) when model predicts positive while true label is negative; false negative (type II error) when model predicts negative while true label is positive. In multi-class classification, when we train models per class (i. e. apply *One-vs-Rest* strategy), one class is viewed as positive, other(s) as negative. Then we can (Vanschoren 2022):

* *micro-average* TP, FP, TN, FN and calculate micro-precision, micro-recall, micro-F1, accuracy,
* *macro-average*, i.e. average scores obtained for each class (preferable for imbalanced classes if all classes are equally important);
* compute weighted average: in this case, weight is a ratio of examples of the particular class, aka suppor; *macro-averaged recall* is a balanced accuracy.

For ROC in multi-class classification, we need to choose between micro- or macro averaging true positive rates (TPR=TP/TP+FN) and false positive rates (FPR=FP/FP+TN). In multi-class classification we micro-average if every sample is equally important (irrespective of class); we macro-average if every class is equally important, especially for imbalanced data.

It is important to make a distinction between performance evaluation and model selection. Reminding: plotting TPR against FPR for all possible thresholdsyields a Receiver Operating Characteristics curve (ROC) - the area under the ROC curve (AUC) gives the best overall model. So, we change the treshold until we find a sweet spot in the TPR-FPR trade-off during model selection (of course, hyperparameters affect predictions and hence also the shape of the curve). Lower thresholds yield higher TPR (recall), higher FPR, and vice versa. Quite often it is better to have a proper scoring rule for model selection in order to achieve maximum performance according to the metric of real interest (for example, accuracy). Any measure of optimality for classification which ignores costs does so at its own risk. Even the ROC AUC which tells how much the model is capable of distinguishing between classes fails to be cost-invarian.

For some models, the predicted uncertainty does not reflect the actual uncertainty. Overfitted models also tend to be over-confident (Vanschoren 2022).

*Definition.* A model is called calibrated if the reported uncertainty actually matches how correct it is. For example, *LogisticRegression* models are well calibrated since they learn probabilities; SVMs are not well calibrated – they are *b*iased towards points close to the decision boundary.

If it is required to select models based on how accurate the class confidences are, *Brier score* which measures the mean squared difference between the predicted probability and the actual outcome may be used. The problem with the *Brier score* is that it ignores the true needs of the application, and no amount of adjusting the threshold will compensate for the selection of the wrong model. ***Brier score loss*** is squared loss between predicted probability and actual outcome. We can post-process trained models to make them more calibrated using various calibration techniques.

For monitoring of the machine learning model(s) the two main matrics might be used: *Population Stability Index* (PSI) and *Characteristic Stability Index* (CSI). PSI measures how much a variable has shifted in distribution between two samples over time. It is widely used for monitoring changes in the characteristics of a population and for diagnosing possible problems in model performance. CSI specifies which variable is causing a shift in population distribution. It compares the distribution of an **independent variable** in the scoring data set to a development data set and detects shifts in the distributions of input variables that are submitted for scoring over time.

Multi-label classification requires different evaluation measures than multi-class classification – the categories of measures are: example based (based on the average differences of the actual and the predicted sets of labels over all samples, e. g. F1 score, hamming loss, accuracy score, etc.), label-based (decomposing the evaluation process into separate evaluations for each label and then averaging over all labels) and ranking-based (Tsoumakas et al. 2007).