0.1 Performance Evaluation

The Evaluation of machine learning models is core part of building an effective and robust model. It is not only a technique to get feedback, at the end of a machine learning training process, wich shows how good the quality of the models results are. The performance evaluation is also part of the optimization process while training a machine learning model. This can be an iterative process, in wich a model is trained, after that feedback of quality is obtained through metrics, then the models hyper-parameters or features are improved (depending on the actual phase of process of creating a machine learning model) and this is repeated until a solid model has been found. [**analyticsvidhya˙evaluation:2022**]
The metrics used to evaluate supervised machine learning models are also divided into evaluation of classification and regression models **jeremyjordan˙evaluation:2022**. Causing the fact that the prediction problem of this study is a classification problem, only evaluation metrics wich are used to evaluated classifier models are mentioned in this topic.
The outcome of a binary machine learning classifier prediction has one of the four following types **jeremyjordan˙evaluation:2022**:

- True positive (TP): The model predicts that an observation belongs to a class and it actually belongs to that class.
- False positive (FP): The model predicts that an observation belongs to a class and it actually does not belong to that class.
- True negative (TN): The model predicts that an observation not belongs to a class and it does not belog to that class.
- False negative (FN): The model predicts that an observation not belongs to a class and it does belog to that class.

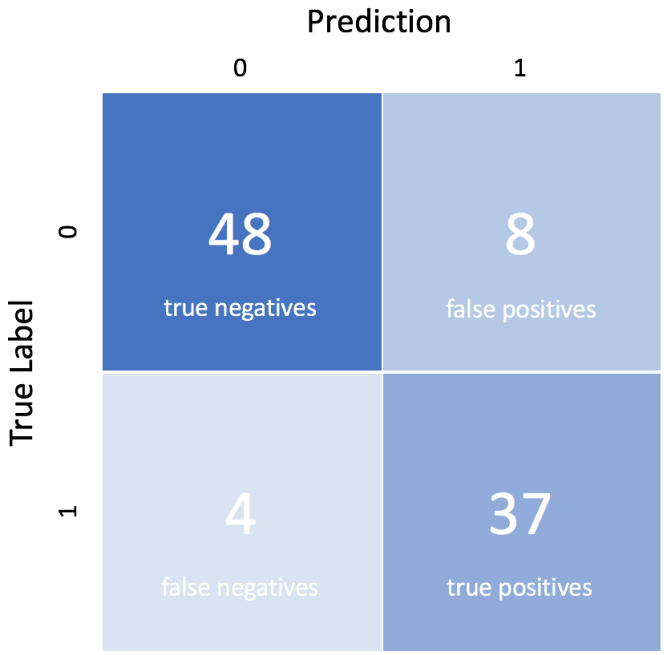
This four values can be plotted on a confusion matrix as shown in Figure 10. The matrix is generated by making predictions on the test data and assigning the results of the individual samples to the four types. Also different classification model evaluation metrics, like the three main scores accuracy, precision and recall score are calculated with these values. [**jeremyjordan˙evaluation:2022**]

0.1.1 Accuracy

The Accuracy score is one of the most common evaluation metrics, when it comes to the evaluation of binary classifiers **Kartik˙evaluation:2022**. Also a lot of the related works use this score to evaluate and compare their machine learning models. Jeremy Jordans defines accuracy as follows:

> Äccuracy is defined as the percentage of correct predictions for the test data."[**jeremyjordan˙evaluation:**

The definition can also be represented like this:
$Accuracy = \frac{Numberofcorrectpredictions}{Numberoftotalpredictions}$ **Google˙Acurracy:2022**



Source: [**jeremyjordan˙evaluation:2022**]

Abbildung 1: A confusion matrix wich shows the four types of a classifier outcomes.

For binary classifiers accuracy is can be calculated as follows:
$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$ **Kartik˙evaluation:2022**

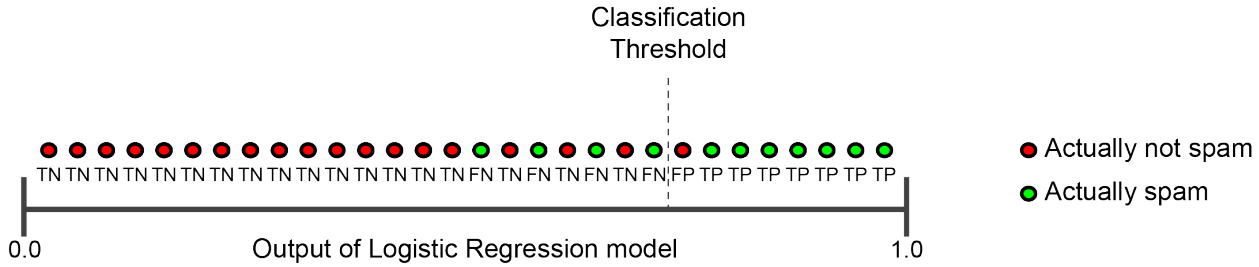
As the definitions describe, the score represents the percentage of the correctness of all predictions. This implies that in the case of an unbalanced class size, the accuracy of a class can be disregarded **Kartik˙evaluation:2022**. As an example in the context of avalanche prediction: There are 90 samples that do not contain an avalanche event and only 10 wich represent an avalanche event. In the case that the algorithm predicts no avalanche event for all samples, it has an accuracy of 90%.
This does not mean that the accuracy score is not useful, the metrics gives a validation of the overall prediction performance of the model. It only signifies that it should not be the only score used for the evaluation, especially in cases of unbalanced class sizes. [**Google˙Acurracy:2022**]

0.1.2 Precision

The Precision score is the percentage of how many positive predicted observations actually are positive **Kartik˙evaluation:2022**.
It is defined as follows:

$Precision = \frac{TP}{TP+FP}$ **Kartik˙evaluation:2022**

The score can be a balancing validation metric for the accuracy score, since it covers exactly the cases in which the accuracy score has the problem described above. So to extend the example mentioned in section 3.3.1, in wich the Number of Avalanches is 10 and ne number of non avalanches is 90. The model only predicts all non-avalanches correctly so the accuracy is 90% but the precision score is 0%. So in this case the accuracy shows that the model has a high prediction quality, but the precision score clarifies that none of the avalanches was predicted.
The fact the use of the Precision Score without the evaluation with another metric, has a similar problem [**Google˙Precision˙Recall:2022**]. In case of the example if only one positive samples is correctly predicted as avalanche, the precision score is 100% but nine of ten avalanche samples are rated false. So the same balancing characteristic applies the other way around from accuracy score to precision score. Figure 11 shows a set of datapoints and the precision score evaluates the right side of the classification threshold line. Since the precision score only targets the samples predicted as positive.



Source: [**Google˙Precision˙Recall:2022**]

Abbildung 2: A set of datapoints split by the classifier and marked as one of the four classifier output types.

0.1.3 Recall

The Recall is another evaluation score, calculated out of the four values represented in the confusion matrix. It is defined as the percentage of actual positive datapoints predicted as positive **Google˙Precision˙Recall:2022**. In figure 11 the recall is represented as all green marked datapoints on the right sight of the classification threshold line divided by the all green marked samples.
The recall evaluation metric is calculated as follows:

$Recall = \frac{TP}{TP+FN}$ **Kartik˙evaluation:2022**

0.1.4 ROC-Curve and AUC

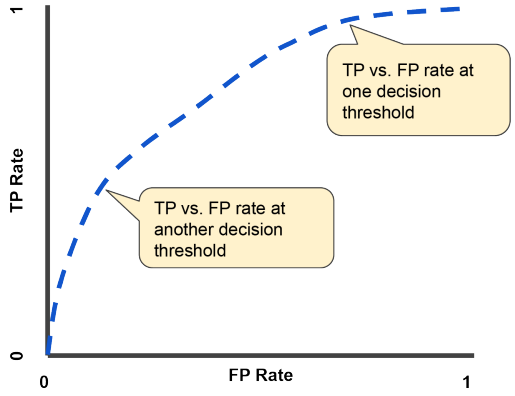
The ROC curve (Receiver operating characteristic) is a graph representing the True Positive Rate (TPR) compared to the False Positive Rate (FPR) for different classification thresholds from a model **Google˙ROC˙AUC:2022**.

The two parameters TPR and FPR are defined as follows:

$TPR = \frac{TP}{TP+FN}$ **Google˙ROC˙AUC:2022**

$FPR = \frac{FP}{FP+TN}$ **Google˙ROC˙AUC:2022**

If the threshold is lower, the model classifies more samples as positive. The consequence of this is that both the TPR and the FPR increase. The same happens in reverse with a higher threshold. [**Google˙ROC˙AUC:2022**] [**Kartik˙evaluation:2022**]
For machine learning classifiers, wich have a class as output and do not use a threshold, the ROC curve will be represented as a single point in the plot **analyticsvidhya˙evaluation:2022**.

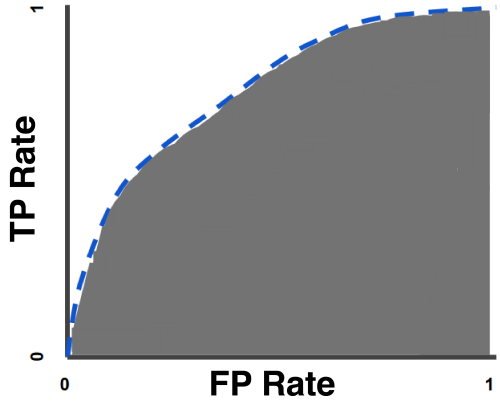


Source: [**Google˙ROC˙AUC:2022**]

Abbildung 3: The ROC curve plots the TPR vs. the FPR on all different thresholds.

An interactive approach, in which a classifier model is evaluated many times with different thresholds, would be associated with high computational costs. However, there is also a more efficient approach called AUC, which can also determine this information. [**Google˙ROC˙AUC:2022**]

The AUC (Area under the Curve) statistic represents the integral calculus of the ROC curve from (0,0) to (1,1). An Example is represented in figure 13. It gives an aggregate measure of the models prediction quality about the whole range of possible classification thresholds. So with the AUC statistic, the ROC curve is represented by a single number. The value of AUC can be in the range between 0 and 1. If the Value is 0.0, the predictions are 1000% false. If the value is 1.0, all predictions are correct. [**Google˙ROC˙AUC:2022**, **analyticsvidhya˙evaluation:2022**]
The value The higher the numerical values of the AUC statistic the better is the models performance **analyticsvidhya˙evaluation:2022** . This number is definitely meaningful, however, the entire ROC curve should always be considered as there are models that perform better in certain areas and other models in other regions **analyticsvidhya˙evaluation:2022**.



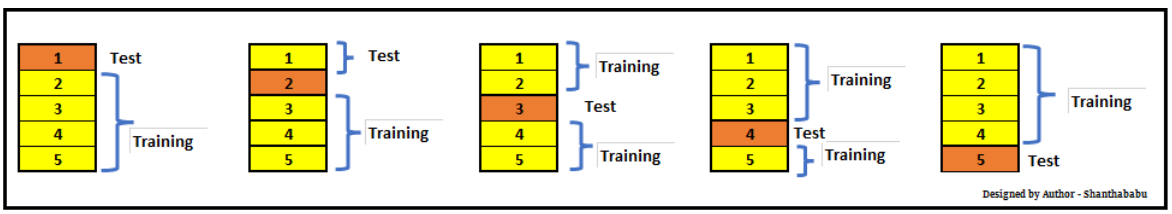
Source: [**Google˙ROC˙AUC:2022**]

Abbildung 4: The AUC statistic represents the grey marked area under the ROC curve.

In Context of this work, the implementation of the ROC-curve as well as the AUC value included in the same python library (Sklearn.metrics) as the other metrics mentioned before are used. This implementation of the ROC-curve is restricted to binary classification tasks. [**Scikit-learn-roc-curve:2022**]

0.1.5 K-Fold Cross-Validation

The k-fold cross-validation is a technique to evaluate machine learning classifier models in a balanced way and to avoid the risk of a random training test split variant with a splitting that is not meaningful. This balance is caused by the fact that with this method the model is both trained and validated with every data sample of the set. [**Refaeilzadeh2009**]
In the k-fold cross-validation, the dataset is split into k equally sized subsets. After that the model is trained iteratively with k-1 folds of the dataset and the last fold is used for validation. The one fold wich is held-out change every iteration until the model is validated with every sample of the set. The performance of each iteration is tracked by an evaluation metric like accuracy or precision. This process is shown for the example of an 5-fold cross validation in Figure 13. [**Refaeilzadeh2009**]
The 10-fold cross validation is a popular variant in terms of machine learning. the deci-



Source: [**analyticsvidhya˙cross˙validation:2022**]

Abbildung 5: The iteration process of a 5-fold cross-validation.

sive advantage compared to, for example, a 70/30 train test split is that you have a large

train set of 90% for each iteration. So the machine learning model does have more samples to learn from. At the same time k-fold cross-validation provides a precise test coverage. [**analyticsvidhya˙cross˙validation:2022**] In the case of this master thesis, the accuracy, precision and recall metric are all used in combination with the cross-validation implementation of the Python library sklearn.model_selection, in wich the metrics can be selected.