

BACHELOR

A systematic determination of metrics for Classification tasks in OpenML

Puertolas Molina, Jorge

Award date:
2022

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

**A systematic
determination of metrics
for Classification tasks in
OpenML**

Jorge Puertolas Molina
j.puertolas.molina@student.tue.nl
1525492

Supervisors:
Joaquin Vanschoren

Contents

Contents	ii
1 Introduction	2
1.1 Problem definition	3
1.2 Research question	3
1.3 Methodology Outline	3
1.3.1 Metric Selection	3
1.3.2 Implementation of Metrics	4
1.3.3 Showing the use of the metrics	4
1.4 Personal Contribution	4
2 Literature Review	5
3 Methodology	7
3.1 Metric Selection	7
3.1.1 Context Measures	7
3.1.2 Imbalanced Datasets	9
3.1.3 Binary	13
3.1.4 Multi Class	15
3.1.5 OpenML Metrics	16
3.1.6 Final Metrics for Classification	16
3.2 Metric Implementation	17
3.3 Function for detection of metrics	18
3.4 Metric Testing	19
3.5 Limitations	19
4 Results	20
4.1 Implementation of Metrics	20
4.2 OpenML Examples	20
4.2.1 Imbalanced Binary Data	20
4.2.2 Binary Balanced Data	21
4.2.3 MultiClass Imbalanced Data	22
5 Conclusions	23
5.1 Conclusion	23
5.2 Discussion	23
5.3 Future Work	24

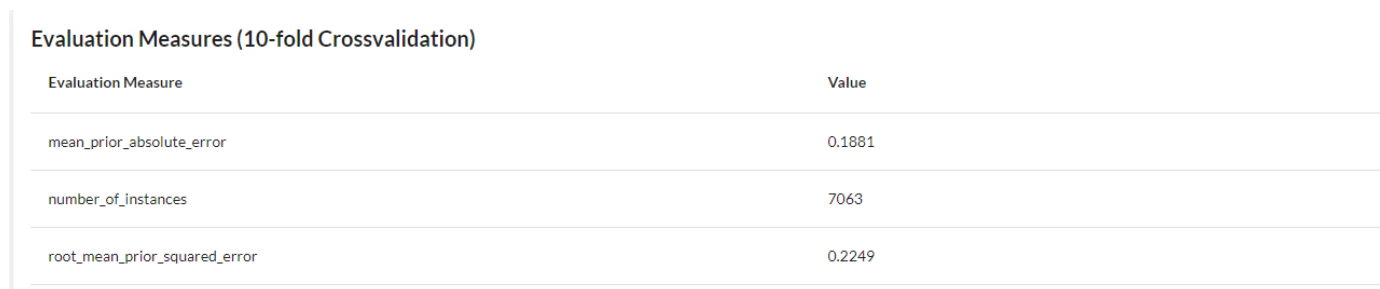
Bibliography	25
A Explanation of Metrics	27
A.1 Code	27
A.1.1 isBalanced() Function	27
A.1.2 Evaluation Metrics	27

Chapter 1

Introduction

The roots of machine learning can be traced back to the 1950s [24]. Since its creation, the usage and creation of machine learning algorithms has been growing rapidly. Nowadays, organizations increasingly rely on these algorithms to provide them with a variety of powerful implementations. These are recommender systems, forecasting, voice recognition, etc. In 2017, the global machine learning market totaled 1.4 billion and is estimated to reach 8.8 billion by 2022 [9]. With this development, the machine learning research field needed a platform that made resources accessible and reusable.

For this purpose, OpenML was created. OpenML is an open platform for sharing datasets, algorithms, and experiments. Older components in the platform are being rewritten into Python to ensure continued adoption by the machine learning community. The evaluation engine is one of these components, which serves as a tool to evaluate models made by many users on existing datasets on the platform. In figure 1, an example of the evaluation engine's function is shown. The evaluation engine returns, for a specific run, the metrics that correspond to the task type's metrics. For instance, runs for a task that has the machine learning task type 'Classification', would return the metric 'Accuracy' (percentage of correct results). For the machine learning task type 'Regression', the metrics used differ, and metrics like Mean absolute error will be used. This is the part of API that we will rebuild in this project.



Evaluation Measure	Value
mean_prior_absolute_error	0.1881
number_of_instances	7063
root_mean_prior_squared_error	0.2249

Figure 1.1: Example of Evaluation Metrics

1.1 Problem definition

The current evaluation component in OpenML is written in Java. A novel, python-based implementation of this evaluation engine that is very well designed, well-implemented, well-tested, well-documented, scalable, and extensible is needed.

This paper will focus on developing the part of the evaluation engine that is dedicated to Classification Tasks. This implies the cautious research and implementation of metrics for Classification tasks. The goal at the end of the project is to have a systematic way of evaluating the model at hand properly, depending on the characteristics of the dataset and the task.

1.2 Research question

Having mentioned the focal goal of the project, we introduce a main research question to guide the process:

- **How can the models for Classification tasks be evaluated systematically?**

The first objective is to find the different scenarios that affect the appropriateness of using evaluation metrics. Then, we make a list of all metrics per scenario. The objective is to narrow down that list of metrics by selecting the best metrics depending on the scenario at hand, so that those metrics are returned by default in the OpenML engine. For this we need to answer the following questions:

- Which task/data characteristics matter when considering which metric to use?
- Which evaluation metrics are better depending on the task/data characteristic?
- How can the metrics be implemented?

1.3 Methodology Outline

To answer the main research question and its derived questions, multiple tasks need to be accomplished. In this outline, all these tasks and procedures needed to complete the project are listed and explained briefly.

1.3.1 Metric Selection

The first step is determining all the metrics that are going to be implemented. The objective is to provide extensive research on the chosen metrics for classification tasks. This research begins in the literature review section, where we introduce the distinct challenges we face when we determine which metrics to use. This examination will enlighten issues with the usage of some metrics. For instance, the issue with the use of accuracy in the case of imbalanced data, as described by Valverde et al [25]. From these issues, we can then obtain directives for the division of the analysis based on the task characteristics. By dividing the analysis, we focus individually on finding the best metrics for each individual case. For this, we explore scientific research relevant to each case to find new metrics. We explain the functioning of

metrics. Then, we proceed to compare the metrics and generate a list of best metrics per use case. This list of metrics can then be used to systematically determine the best metrics to use for the OpenML task.

Also, the evaluation metrics have to cover a variety of functionalities that adapt to all the possible needs a user of OpenML might have. For instance, the Mean Absolute Error (MAE) gives different information than the Mean Squared Error (MSE) does, because the MAE error is less sensitive to outliers than MSE [4]. Different evaluation metrics are used depending on the goal of the user, as this sensitivity to outliers may suit some more than others. For this reason, it is important to ensure that the variety of evaluation metrics, that are included, cover the spectrum of use cases. Finally, the metrics that OpenML is currently implementing will be added to the list.

1.3.2 Implementation of Metrics

Once the definitive set of metrics is finished, as planned in 1.4.1, the implementation of the metrics takes place. A majority of them are likely to be easily implemented by using python libraries such as `scipy.stats`, `scikitlearn`, etc. Others, that are not within those packages, will have to be implemented independently, and will require technical understanding of the metric at hand. After this, the metrics are grouped by type to ensure that the right metrics are computed for the run. This means that if the task is Binary Classification, then all the metrics that are good for Binary Classification will be computed and returned.

1.3.3 Showing the use of the metrics

To finish, we will show, with real examples of OpenML datasets and runs, the results of the metrics we have chosen for each specific case. From these, we finish justifying why this range of metrics is optimal for the OpenML user.

1.4 Personal Contribution

I was not involved in developing the testing for the code nor the sphinx documentation, and in developing the main structure of the evaluation engine. I will not describe the development the other members of the group did in those tasks in this report and I do not claim them as mine.

Chapter 2

Literature Review

There is extensive scientific research on the evaluation and usage of metrics for Classification. Hossin et al [13] reviews 13 metrics commonly used in classification, including Accuracy, Error rate, Precision, Recall, F Measure and Area under Curve.

However, these metrics are often misused. A high accuracy is not equivalent to a high classifier performance. This is called the 'accuracy paradox'. [7]. For instance, suppose we have a dataset D, with a target variable that has classes 'A', 'B', 'C'. Now, datapoints with label 'A' make up 80 percent of all instances. If we were to use a classifier that predicts all instances with label 'A', the accuracy would be 80 percent, but classes 'B' and 'C' would never be detected. This phenomena is also associated with the other aforementioned classification metrics, Precision, Recall, F-measure and AUC. For cases in which the dataset is imbalanced, these metrics are often not representative of the classifier performance. The evaluation metric is biased towards the classes that contain the most instances.

We explored the dataset that has the most runs in the OpenML platform, which is 'credit-g-classification'. This dataset has a target variable with 2 classes: 'good' and 'bad', so it is a binary problem. It has 1000 instances, and the 'good' labels make up 70 percent of the total number of labels, which means the data is imbalanced towards the 'good' class.

To find out if imbalanced datasets in OpenML are frequent or occasional, we investigated more datasets. Out of the 10 datasets with the most runs, 8 of them had Imbalanced data. This makes it increasingly important to implement metrics that adjust to imbalanced data.

Added to this, Sokolova et al [22] mentions there are 3 types of classification problems: binary, multiclass, and multi-label classification. Multilabel classification is not currently implemented in OpenML so it is disregarded. A binary problem is when the target variable has two possible classes, often referred as positive (1) or negative (0). In multiclass classification, each input instance is to be classified into one, and only one, of n classes, where $n \geq 3$. Binary problems and Multiclass problems generally use the same metrics. However, there are some differences that will be explained in the methodology section.

Lastly, a multiclass problem can be an ordered classification problem. An example of this is Wu et al [28], where diabetic retinopathy had to be classified on a scale from 0-5, 5 being the most grave case of Diabetic Retinopathy, and 0 meaning that there is no diabetic retinopathy. For this, other metrics are also necessary, since it is also valuable to know how far the prediction is.

The finding of incompatibility of commonly used metrics with imbalanced data, as well as the different needs for metrics in binary and multiclass problems, led to the decision of dividing the metric selection in 3 separate analyses: Binary, Multiclass (Ordered, Non-Ordered), and Imbalanced Data.

Chapter 3

Methodology

3.1 Metric Selection

The principle is to have an appropriate set of metrics for the user. Appropriateness here is defined as having variability in the metrics available. Different metrics measure different things. Giving the opportunity to the user of giving importance to the criteria he/she considers important is what is defined as appropriate. A metric is considered appropriate to include if in a hypothetical case a user would extract value from using it. If adding a new metric doesn't provide value in any hypothetical case, then it is not included.

For finding and selecting metrics that go beyond the usual metrics, we made use of research papers that introduce and explain new metrics. This was done first for Imbalanced Data, and then specific to binary and multiclass problems

Since OpenML does not currently have metrics for imbalanced data, this analysis concerns the evaluation of models for Imbalanced Datasets mostly. Hence we will explain the benefits and usage of every metric for Imbalanced datasets carefully. For Multiclass and Binary classes, we provide a list of the metrics obtained from the research and briefly explain what their purpose is. From there, we select the metrics that are more suitable for the purpose of adding variability to the engine.

3.1.1 Context Measures

In order to understand the metrics that will be proposed for the evaluation engine, we first need a brief introduction of some basic metrics that are used in Machine Learning.

Confusion Matrix

Evaluation metrics are generally derived from a confusion matrix. A confusion matrix can be used for both multi-class and binary problems. Here is an example:

	<u>Predicted 1</u>	<u>Predicted 0</u>
<u>True 1</u>	TP	FN
<u>True 0</u>	FP	TN

Figure 3.1: Example of Confusion Matrix

The y-index represents the true value of the instances, while the x-index indicates the predicted value. From this, we obtain the TN (True Negatives), FN (False Negatives), TP (True Positives), and False Positives (FP). With the TN, TP, FN, FP, now we can proceed to calculate evaluation measures.

Commonly accepted Metrics

Accuracy is the ratio between the total number of correct predictions and the total number of instances, $\frac{TN+TP}{FN+FP+TP+TN}$.

Error rate is equal to $1 - \text{Accuracy}$.

Specificity is the refers to the ratio between the number of predicted true negatives, and the total number of real negatives: $\frac{TN}{TN+FP}$

Precision refers to the ratio between the number of predicted true positives, and the total number of predicted positives: $\frac{TP}{TP+FP}$

Recall refers to the ratio between the number of predicted true positives, and the total number of real positives: $\frac{TP}{TP+FN}$

The F-Measure is computed using Precision and Recall. It calculates the harmonic mean between Recall and Precision. It has parameter β , that specifies the weight of Recall and Precision in the metrics, derived from van Rijsbergen's effectiveness measure [26]. When $\beta = 1$, the F-measure is the F1 Score, and the formula is:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.1)$$

When $\beta < 1$, more weight is given to precision, and oppositely, when $\beta > 1$, more weight is given to recall. For instance, a $F_{1.5}$ score gives 2 times more weight to recall than precision.

AUC Metrics Area under curve metrics are derived from the comparison of two metrics. The most common are Receiver Operating Characteristic AUC score and PR (Precision-Recall) AUC. ROC AUC score is the area under the curve of the True Positive Rate (Recall) against the False Positive Rate (1 - Specificity). When ROC AUC = 1, the ROC Curve looks like this:

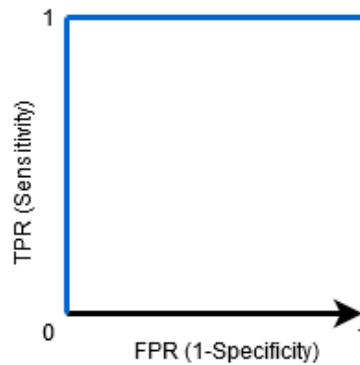


Figure 3.2: Example of ROC Curve with AUC = 1

PR AUC is obtained from plotting Precision against Recall, as shown here:

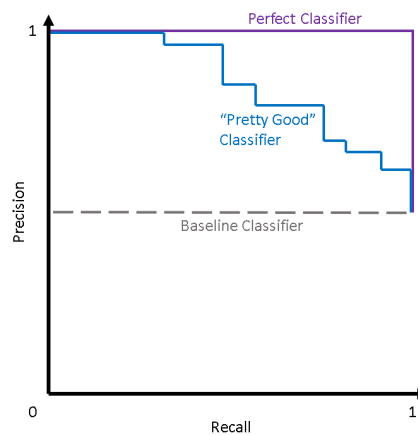


Figure 3.3: Example of PR Curve [23]

All these metrics will be implemented by default in the engine since they are the most basic ones, and users would normally think of these when evaluating their models.

3.1.2 Imbalanced Datasets

In this section, we introduce metrics that tackle the issues that have been encountered with the usage of commonly used metrics in Imbalanced Datasets.

All except 2 metrics in this section are introduced in a binary context. Although they can be applied to multiclass problems, we decide to narrow down the implementation to binary problems. Also, the formula of these metrics lies under the assumption that the data is balanced towards the negative class.

Most of these metrics were found in a compilation of metrics for Imbalanced data learning. [14]

The G-mean metrics

G-mean In maths, a geometric mean is the following: $G\text{-mean} = \sqrt{a \times b}$. In machine learning, the geometric mean is used to evaluate performance by using the true negative rate and the true positive rate, with equal weight. The G-mean was introduced by Kubat et al [16] to deal with imbalanced data. Its formula is:

$$G - \text{mean} = \sqrt{\text{TruePositiveRate} * \text{TrueNegativeRate}} = \sqrt{\text{Recall} * \text{Specificity}} \quad (3.2)$$

G-mean is suitable for imbalanced data since it takes into account equally the Specificity and Recall. Also, G-mean is nonlinear. A change of p percentage points in recall (or specificity) has a different effect on G-mean depending on the magnitude of Recall: the smaller the value of Recall, the greater the change in G-mean. Hence, the more often a classifier fails to recognize a positive, the more this decreases the G-mean. This then means that G-mean really penalizes low accuracy in either Recall or Specificity.

Adjusted G-mean Batuwita et al [1] investigated the usage of common measures for their problem of interest in the domain of bioinformatics, and found out that they're not adequate. The G-mean for example, lead the researcher to choose a model that reduces the weight of specificity too much. To solve this, Batuwita et al [1] proposed the Adjusted Geometric Mean. It was designed to deal with class imbalance problems in which the power of recall on the metric wants to be increased, but without sacrificing the weight of specificity too much. This is the formula for the AGm:

$$AGm = \frac{G - \text{mean} + \text{specificity} \times N_n}{N_n + 1} \quad (3.3)$$

N_n represents the number of negatives. This metric is more sensitive to variations in Specificity. This metric is made such that, the higher the imbalance, the higher the reaction to changes in Specificity. The authors showed the advantages of using this new measure compared to the G-mean and the F-measure.

Alternatives to Accuracy

As mentioned in 2, accuracy is not suitable for imbalanced datasets. Gupta et al [12] explains alternatives to accuracy: Weighted accuracy and balanced accuracy. These are the only metrics introduced in this section that are introduced as a multi class problem.

Balanced Accuracy is the mean of the accuracy for each class. For instance, in the example shown in 2, the accuracy was 100 percent for class 'A' but 0 percent for class 'B' and 'C'. This returned an accuracy of 80 percent because 80 percent of the instances were labeled as class 'A'. Balanced Accuracy would be the average of the accuracies. In our example, the

result would be $\frac{Accuracy(ClassA)+Accuracy(ClassB)+Accuracy(ClassC)}{NumberofClasses} = \frac{100+0+0}{3} = 33.3$ percent, which differs a great deal from the 80 percent returned by accuracy.

Weighted Accuracy is a modified version of the Balanced Accuracy, where one can add weights that signify the importance of the class accuracy. For our case, an example could be to give more weight to the accuracy of Class A, since it has the majority of instances. This could result in a formula similar to this $\frac{Accuracy(ClassA) \times 1.5 + Accuracy(ClassB) \times 0.75 + Accuracy(ClassC) \times 0.75}{3} = \frac{100 \times 1.5 + 0 \times 0.75 + 0 \times 0.75}{3} = 0.5$. This allows the user of the metric to give importance to whichever class accuracy. To illustrate this, in medicine it is well known that false negatives are more harmful than false positives. When a false negative occurs, a disease is left untreated, and can proliferate quickly. An expert in domain would then choose to have more weight in the accuracy of class 0.

Alternative to F-measure

Furthermore, an alternative to F-measure was proposed for imbalanced data by Maratea et al [17]. The metric, named Adjusted F-measure (AGF) is computed as follows: First the F2 measure is computed, which gives recall more importance than precision, and hence gives more strength to the FN values. Then the class labels are switched (Negatives become positives, positives become negatives). Then the F0.5 measure is computed with this inverted data, called Inv F0.5. Finally, the geometric mean, previously seen in this chapter, of the F2 measure and the Inv F0.5 measure, is computed:

$$AGF = \sqrt{F_2 \times InvF_{0.5}} \quad (3.4)$$

This is done to balance out both indices. This metric gives more weight to the minority class in the computation, which makes it more suitable for imbalanced data where the correct prediction of the minority class is valuable. To prove this metric's robustness, a comparison test with the F1-measure was done in a worst-case scenario, where the data was extremely imbalanced. Confusion Matrices for imbalanced data were generated, and then the AGF and the F measure were computed. The final result was: 'The comparison between the F1-measure and the Adjusted F-measure highlights that the more the data are imbalanced, the greater sensitivity to the correct classification of patterns belonging to the minority class is shown from the latter'.

Alternative to Precision

Ranawana and Palade [19] developed Optimized Precision, which also combines recall and specificity. This is the formula:

$$OptimizedPrecision = \frac{specificity \times N_n + recall \times N_p}{N_n + N_p} - \frac{|specificity - recall|}{specificity + recall} \quad (3.5)$$

N_n represents the number of negatives in the dataset, while N_p represents the number of positives in the dataset. The goal is to weigh specificity and recall according to the imbalance of the data. It is more systematic. The authors showed in the conclusion that it is more favorable to use compared to normal precision, since it adjusts for imbalance.

Alternative to any metric

Garcia et al [10] explains a novel measure for imbalanced datasets, called Index of Balanced Measure. It is used to turn any measure into an appropriate measure for an imbalanced dataset. However, it can only be used with binary data. This is the formula:

$$IBM = (1 + \alpha \times Dom) \times M \quad (3.6)$$

M signifies a performance metric (Accuracy, AUC, etc). $Dom = True\ Positive\ Rate - True\ Negative\ Rate$. Dom ranges then from [-1, 1]. When True Positive Rate > True Negative Rate, then $Dom > 0$, and oppositely, when True Positive Rate < True Negative Rate, $Dom < 0$. α is a weighting factor for $Dom \geq 1$.

PR AUC Curve

Saito et Rehmsmeier [21] claim that the Precision Recall AUC metric is more informative than ROC when evaluating binary classifiers on Imbalanced Datasets. PR Plots change with the ratio of positives and negatives. Due to this, PR plots, and by consequence also AUC, change between balanced and imbalanced datasets. As PR Plots are able to show performance differences between balanced and imbalanced datasets, it is the most informative and powerful plot for imbalanced cases and is able to explicitly reveal differences in early-retrieval performance. However, ROC is more frequently used in these problems. [21]. To solve this, we suggest using PR AUC as the preferred AUC metric for imbalanced datasets in binary problems.

Conclusion

To conclude, Jeni et al [15] compared a variety of metrics in both scenarios: Balanced and Imbalanced Binary Data. The reviewed metrics were: Accuracy, F1 Score, Cohen's Kappa, Krippendorf's Alpha, ROC AUC and Precision Recall AUC. The final conclusion was "We discovered that with exception of area under the ROC curve, all performance metrics were attenuated by imbalanced distributions; in many cases, dramatically so." This confirms that the traditional metrics are not suitable for imbalanced data. Hence, all the metrics that were introduced in this section will be included in the evaluation engine.

The use of this conclusion is to provide a summary on which metrics are suitable for imbalanced data. Also, to provide an explanation on the usage case of these metrics for the user/researcher using OpenML. For instance, the G-mean is a good metric when the researcher is dealing with imbalanced data, and the prediction of both classes is equally important. It is also an appropriate metric when poor performance in specificity and recall is really not desired.

Metric	Usage Case
G-mean	<ul style="list-style-type: none"> • Equal weight to recall and specificity • Penalizes low specificity and recall
AGm	<ul style="list-style-type: none"> • Same as G-mean but gives more weight to the majority class
Balanced Accuracy	<ul style="list-style-type: none"> • Equal weight to all classes
Weighted Accuracy	<ul style="list-style-type: none"> • User can customize the weights according to domain expertise
Adjusted F-Measure	<ul style="list-style-type: none"> • Minority classes have more weight than normal F-measure
Optimized Precision	<ul style="list-style-type: none"> • Systematically Balanced Metric • Weighs the metric according to the degree of imbalance
Index of Balanced Measure	<ul style="list-style-type: none"> • Flexible, turns any metric into balanced • Customizable by user
PR AUC	<ul style="list-style-type: none"> • The most robust AUC metric for imbalanced datasets

These metrics are all a great inclusion, as they provide flexibility to OpenML's platform. Users can rely on the repertoire of evaluation metrics to adjust to their preferences. Especially, since most users are researchers, they have an in-depth understanding of evaluation metrics. The inclusion of these metrics would be even greater as researchers know exactly how to use the metrics according to their specific problems. Furthermore, Index of Balanced Measure and Weighted Accuracy have parameters of weight that are customizable by the user. This is a great asset because it can never be known which particular wants a user has.

3.1.3 Binary

Metrics

Now we proceed to find metrics for binary problems, which are neither in 3.1.1 nor in 3.1.2. For simplicity, here is a table of the Metrics found, with their respective explanation:

Metric	Explanation
Cohen's Kappa	Measure of agreement between the prediction and the true value
Brier Score	MSE between probability and true value
Matthews Correlation Coefficient	Correlation between observed and predicted classifications
Youden's Index	Evaluates the algorithm's ability to avoid failure
Discriminant Power	Evaluates how well an algorithm distinguishes between positives and negatives
Jaccard Score	Calculates only the model's ability to predict positives

Analysis

Ferri et al [8] studied the relationships between the most common performance measures for classifiers. In this study, a set of experiments was designed to analyse the correlations between measures and their sensitivity to several identified traits. These were the results:

- Accuracy and Cohen's kappa show high correlations
- Accuracy, Cohen's kappa are best when noise is present on the dataset. (If no imbalance)

- Accuracy and Cohen’s kappa are very bad with small training datasets or when a bad algorithm is used
- AUC measures are the best with small training datasets
- Macro Averages are not good when there are very few examples from one class

The last finding that was relevant from this paper, is that it is not significantly different to use different variants of AUC, since the correlations with one another were high. This is also why it was decided not to include more AUC metrics.

Chicco et Jurman [5] made an extensive study comparing the Matthews Correlation Coefficient (will be referred as MCC) to the F1 Score and Accuracy, in binary datasets. The metrics were compared in 6 cases: 2 cases of positively imbalanced datasets, 2 cases of negatively imbalanced datasets, and 2 cases of balanced datasets. In all the cases, the Matthew’s Correlation Coefficient provided an informative response. On the other hand, Accuracy failed to give informative responses in half of the cases, and the F-1 Score in 2 of them. The conclusion is that accuracy and F-1 can mislead, but the Matthews Correlation Coefficient doesn’t. The MCC was also compared with the Brier Score and Cohen’s Kappa by Chicco et al [6].

Cohen’s kappa was found to provide misleading information when true positives and true negatives are zero. The Brier Score can generate an ambiguous outcome when its value is close to 0.25, as it can correspond both to a very good prediction and to a very bad prediction. MCC does not share these flaws with Cohen’s Kappa and the Brier Score.

Youden’s Index is simply the sum of Recall and Specificity, minus 1. Biggerstaff et Brad [2] claims Youden’s Index is the best available summary measure. The other reason why Youden’s index is appreciated, is because of its simplicity and clarity, which makes it easily interpretable [18]. The disadvantage is that it hides high performance in either recall or specificity, when one is very high and the other very low.

Selection of Metrics

Now, from the metrics that were presented in the previous section, we decide which metrics will be implemented.

The Brier Score, very often, returns counter-intuitive results. Low values of the Score can either mean a very good performance, or a very bad one. However, it can be used to complement the Matthews Correlation Coefficient for model selection. When two models have a similar MCC, these models can have different Brier Scores, and this can help the user select the model with the best Brier Score. For this reason, it will be implemented.

Cohen’s kappa not onlt has a high correlation with Accuracy, but also because Chicco et Jurman [6] clearly demonstrated that it provided no leverage over the Matthews Correlation Coefficient. However, when 2 classifiers have the same accuracy, it can be used to determine imbalances in errors. When the errors are more balanced towards one class, cohen’s kappa tends to increase

Matthew's Correlation Coefficient seems to be the most consistent metric of all for binary problems. It is more consistent than Accuracy and F-1 [5] and more informative than the Brier Score and Cohen's Kappa [6]. It will be implemented.

Youden's index will be implemented because of its simplicity. It provides a good overall summary of the performance of the model. Recall and Specificity are given equal weight, and they are summed up, which makes it linear. However, it is recommended to use it with other metrics as well, as a big difference between recall and specificity is not reflected in this metric.

The Discriminant Power is used for feature selection. It could be implemented, but in the future. OpenML could create an interactive feature selection dashboard where the user can obtain the Discriminant Power of the feature he/she selects in the dashboard

No other metric gives as much importance to the correct prediction of positives as the Jaccard score does. This could be of benefit in a hypothetical case to a researcher, when only the rate of correct predictions of positives matters. It is implemented to add flexibility to the engine.

Except for the Discriminant Power, all the metrics will be implemented.

3.1.4 Multi Class

Non Ordinal

To understand how to use metrics in a multi-class context, we use the concepts described in [11]. Here, the F1-Score, the MCC and Cohen's Kappa are extrapolated to a multiclass context. We explain only how the the F1 score is compute in multiclass classification, since the same principle applies to the others as well.

There are two ways of computing the F-measure for multi-class: Micro and Macro F-1. Micro F-1 is equal to Accuracy. Now we explain how the Macro F-1 is obtained, since it is the same method for other metrics used in binary problems as well.

Macro F-1 In multiclass tasks, measures are also calculated from a confusion matrix, but the computation differs a bit. This is how TP, FP, TN, FN are calculated in multiclass tasks:

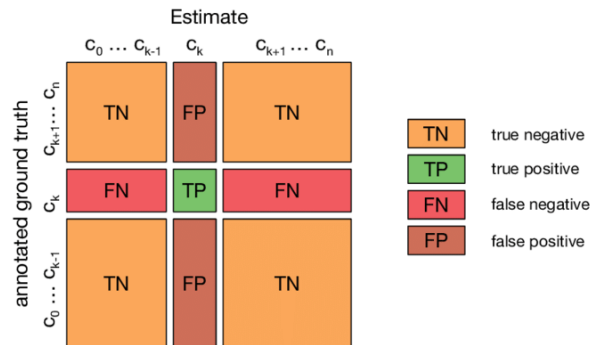


Figure 3.4: Confusion Matrix Multi-class classification

For calculating the Macro F-1, we calculate the Precision and Recall of each class, and then we use the average Precision and average Recall for all classes to calculate the harmonic mean, just like in the normal F-1

The MCC, Cohen's Kappa and the AUC scores, are calculated in the same way (when using macro averaging)

Ordered Classification

As mentioned in 2, Classification can be ordinal. In these tasks, it is also valuable to know how far from the true value the classification is. For this, metrics that are commonly used in regression can be used [8]:

- Mean Absolute Error
- Mean Squared Error
- Root Mean Squared Error

It is normally preferred to use Mean Absolute Error over the other 2 [27]. RMSE, and hence MSE as well, "tends to become increasingly larger than MAE as the distribution of error magnitudes becomes more variable, and it tends to grow larger than MAE".

3.1.5 OpenML Metrics

Metrics that are currently being implemented for Classification Tasks in OpenML also needed to be implemented. The only metrics that weren't mentioned in this paper and are currently in OpenML are: Kononenko Bratko, and Number of instances. Kononenko Bratko was not implemented because of time constraints, but Number of instances was (it's simple).

3.1.6 Final Metrics for Classification

The analysis showed that most of the measures used in machine learning and pattern recognition for evaluating classifiers really measure different things, especially for multiclass problems and problems with imbalanced class distribution, where correlations are worse. [8]

That's it was needed to add many metrics that cover different use cases. In the end, these were the metrics chosen for each:

Imbalanced	Binary	Multi Class
G-mean	Accuracy	<u>Non-Ordered</u>
Adjusted G-mean	ROC AUC	Macro F-1
Balanced Accuracy	PR AUC	Micro F-1
Weighted Accuracy	F-1	ROC AUC
Adjusted F-1	Precision	PR AUC
Optimized Precision	Recall	Kappa
Index of Balanced Measure	Specificity	Accuracy
	MCC	MCC
	Conf Matrix	Conf Matrix
	Brier Score	<u>Ordered</u>
	Youden's Index	RMSE
	Jaccard	MSE
	Kappa	MAE

3.2 Metric Implementation

To implement all these metrics, we had to find out how. Here is a table that shows 1)The metric 2) The problem they're implemented for 3) How it is implemented:

Metric	Problem/s	Method
G-mean	Binary	From scratch
Adjusted G-mean	Binary	From scratch
Balanced Accuracy	Binary, Multiclass	Sklearn
Weighted Accuracy	Binary, Multiclass	From scratch
Adjusted F-1	Binary	From scratch
Optimized Precision	Binary	From scratch
Index of Balanced Measure	Binary	From scratch
Accuracy	Binary, Multiclass	Sklearn
ROC AUC	Binary, Multiclass	Sklearn
PR AUC	Binary, Multiclass	Sklearn
Macro F-1	Multiclass	Sklearn
Micro F-1	Multiclass	Sklearn
F-1	Binary	Sklearn
Precision	Binary, Multiclass	Sklearn
Recall	Binary, Multiclass	Sklearn
Specificity	Binary	Sklearn
MCC	Binary, Multiclass	Sklearn
Youden's Index	Binary	From scratch
Brier Score	Binary	Sklearn
Jaccard	Binary	Sklearn
Kappa	Multiclass	Sklearn
Confusion Matrix	Binary, Multiclass	Sklearn
MAE	Ordinal Multiclass	Sklearn
RMSE	Ordinal Multiclass	Sklearn
MSE	Ordinal Multiclass	Sklearn

The imbalanced metrics can all be implemented for Binary and Multiclass problems. However, only balanced accuracy and weighted accuracy were implemented for Multiclass. The others were not a valuable addition since in multiclass problems there are normally no negatives as such. Also, their implementation in multiclass problems is not priority since there aren't many multiclass datasets.

These metrics are all documented with a docstring. The docstrings contain a description of the evaluation metric. Metrics that are implemented 'From scratch' are metrics that use the confusion matrix function from sklearn to calculate the metrics. Hence, the runtime and efficiency of these metrics is preserved.

3.3 Function for detection of metrics

In order to detect the appropriate metrics for the dataset at hand, we have to detect first whether the data is Binary or Multiclass. This is easily done by calculating the length of the unique values in the column that contains the true predictions. Then we determine if the data is imbalanced with a function called `isBalanced()` (found in appendix), with parameter `thresh`. `Thresh` determines the difference in percentage of instance between the majority class and the minority class in order to call the data imbalanced. After this, we know which metrics apply to the case, and we run them systematically. This would be the decision tree:

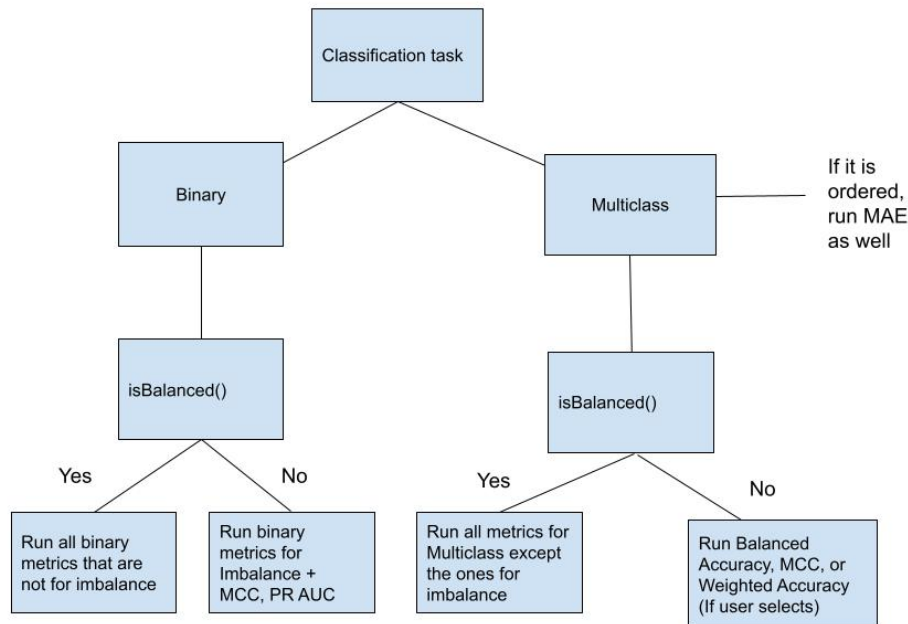


Figure 3.5: Example of Systematic Determination of Metrics

This Decision Tree shows a systematic way of evaluation models for classification tasks. Our evaluation engine would run different metrics in each different scenario. These are the factors that influence the used metrics: when the data is imbalanced, data is either binary or multi-class, and when the multi-class classification is ordinal.

3.4 Metric Testing

Finally, we test the metrics by showing the results of the metrics with real runs and datasets in OpenML. We show the results for 3 different cases:

- Binary with balanced data
- Binary with imbalanced data
- Multiclass with imbalanced data

3.5 Limitations

No datasets for ordinal classification were found in the OpenML website. However, in case ordinal classification is implemented in the future, the metrics were still included.

Also, not many Multiclass datasets are available in the OpenML website. This made it hard to find a dataset where we can run an example of the metrics on imbalanced data.

Chapter 4

Results

4.1 Implementation of Metrics

The implementation of all the metrics is shown in the Appendix at the end of the paper. They were not yet included in the package because they weren't sufficiently tested

4.2 OpenML Examples

Here, we run metrics in 4 cases: Imbalanced Binary Data, Balanced Binary Data, Imbalanced Multiclass Data, and Balanced Multiclass data

4.2.1 Imbalanced Binary Data

For this example we used the following dataset: kc2. The kc2 dataset has 522 instances: 415 have the label 'no', while 107 have the label 'yes'. We run the following metrics on one run: Metrics for Imbalance (the non customizable ones), MCC, Accuracy, F1-Score, ROC AUC and PR AUC. The results for the Confusion Matrix were:

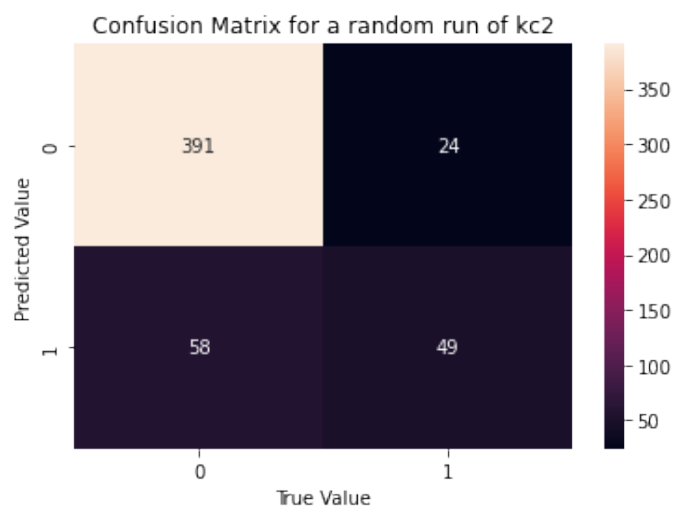


Figure 4.1: Confusion Matrix for run of kc2

This confusion matrix clearly shows a difference in the accuracy for predicting 'no' and predicting 'yes'. The result of the metrics is the following:

Gmean	AdjustedGmean	BalancedAccuracy	AdjustedF_1	OptimizedPrecision	Accuracy	ROC_AUC	PR_AUC	F_1	matthews_correlation_coefficient
0.656856	0.783222	0.700056	0.754452	0.497065	0.842912	0.700056	0.418498	0.905093	0.465708

Figure 4.2: Metrics for run of kc2

As expected, Accuracy and F1 show high scores, 0.84 and 0.9 respectively. This is the 'accuracy paradox' in action. Since most values are 'no', although the accuracy for detection of 'yes' is low, these measures return a high score. This is fine when the user considers it more important to predict 'no' correctly. Matthew's Correlation Coefficient also presents a pretty decent score, but lower since poor prediction of 'yes' is accounted for more than in Accuracy. The G-mean is the lowest of the scores for the Imbalanced metrics, because it penalizes highly a low accuracy of 'yes'. The Adjusted G-mean is higher than the G-mean because it gives more importance to the majority class. AdjustedF1 shows a lower score than the normal one, which is more realistic. Same applies to BalancedAccuracy. Optimized Precision is very low because the difference between recall and specificity is high. Finally, the PR AUC's score is much lower than the ROC's score. This is explained by the minority class, that holds more weight than in ROC's score. These results confirm that until now, the metrics that were being used for Imbalanced Datasets (which seem to be a large majority of the datasets in the platform) were not accurately representing the model's performance. We can confirm that the new metrics are all a great inclusion in the engine, since the user can choose the appropriate criteria for a model's performance. For instance, G-mean is ideal when low performance wants to be specifically penalized, or BalancedAccuracy is ideal when equal importance of correct predictions is wanted.

4.2.2 Binary Balanced Data

For this case, the dataset kr-vs-kp was chosen. This dataset is balanced: it has 1669 'won' labels and 1527 'nowin' labels. Here are the confusion matrix and the metric results:

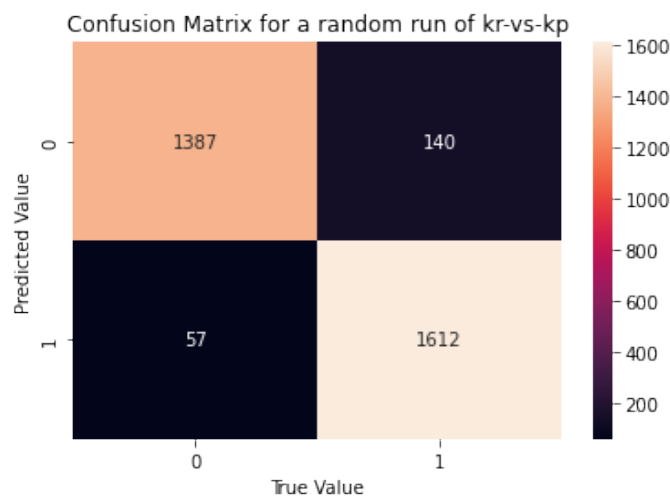


Figure 4.3: Confusion Matrix for random run of kr-vs-kp

Gmean	AdjustedGmean	BalancedAccuracy	AdjustedF_1	OptimizedPrecision	Accuracy	ROC_AUC	PR_AUC	F_1	matthews_correlation_coefficient
0.936641	0.946661	0.937082	0.923581	0.907664	0.93836	0.937082	0.906503	0.933692	0.877385

Figure 4.4: Metrics for run of kr-vs-kp

This example is just to show that when data is balanced, the new metrics do not provide any extra benefit. All the alternatives return a similar result to their original one. Hence, they should not be run in balanced data.

4.2.3 MultiClass Imbalanced Data

There are not many MultiClass tasks in OpenML. We used the most imbalanced dataset for multiclass classification there is in the platform. It is GCM: has 14 labels, with the majority class containing 30 instances, and the minority 10.

BalancedAccuracy	Accuracy	MacroF_1	matthews_correlation_coefficient
0.587121	0.663158	0.582891	0.632662

Figure 4.5: Metrics for run of GCM

For Multiclass, the only metric for imbalanced data that we could test was the balanced accuracy. Weighted accuracy is also implemented, but it is completely customized by the user. Still, we see a considerable difference between the performance of Balanced Accuracy and Accuracy. When the prediction of all the classes is equally crucial, the BalancedAccuracy Score provides a more informative, representative score.

Chapter 5

Conclusions

5.1 Conclusion

In this paper, we have extensively reviewed the literature on evaluation metrics for classification. More than 20 scientific papers were read in order to find metrics that could be implemented for the OpenML evaluation engine. We discovered a popular issue with common metrics: their inability to be informative when the data is imbalanced. For this reason, we sought alternatives to the common metrics that could resolve these issues. Alternatives to accuracy (Balanced and Weighted Accuracy), to F-measure (Adjusted F-measure), to Precision (Optimized Precision) were presented. We also introduced the G-mean metric, that gives equal weight to true negative and positive rates, and the adjusted G-mean, which gives more weight to the majority class.

Afterwards, we presented the findings of the research on metrics used for binary problems. Matthew's correlation coefficient was shown to outperform Accuracy, F-measure, the Brier Score, and Cohen's kappa in binary problems. For this reason, we added it to the engine. Youden's index, the Jaccard Score, and the Brier Score were also added for the purpose of adding variability to the engine.

The last step was finding metrics for Multiclass problems. These metrics are not inherently associated with multiclass problems. They are metrics for binary problems that were extrapolated to Multi class problems. This was the case for the AUC scores, Accuracy, MCC, Kappa, Macro/Micro F1. For ordered classifications however, metrics used for regression (MAE, MSE, RMSE) are used.

At the end of the research we had a definitive list of metrics that were going to be included in the engine. We then proceeded to implement them via sklearn. Once this was done, we designed an algorithm that detects which metrics should run depending on the task characteristics

Finally, in the Results section, we showed examples of these new metrics, and juxtaposed these metrics to the confusion matrix. We successfully showed why the new metrics for imbalanced datasets are a better fit for OpenML than the common measures.

5.2 Discussion

The measures we implemented for imbalanced data are unconventional. The disadvantage of including these measures is that their results are hard to interpret. Common measures such

as accuracy or the f-measure are straightforward because they have been extensively used. This makes them an easier and more familiar choice. It might result in users not using the measures for imbalanced data since they're accustomed to others.

Also, the tests we provided were very limited. In the future, the metrics could be compared with more datasets (imbalanced or balanced). The results that we found clearly showed the benefits of using metrics for imbalanced data, but this was just for one run. It might not be statistically significant.

5.3 Future Work

There are multiple possibilities for future work. First, the systematic determination of metrics shown in Figure 3.4 has not been fully implemented. The current implementation is just dividing the metrics used for each Binary and Multiclass problems.

We introduced a variety of metrics for Imbalanced Data. The majority of these metrics are not currently implemented for Multiclass Problems due to time constraints. This is also a point of interest for future work.

The Index of Balanced Measure and the Weighted Accuracy have been implemented. However, a way to integrate the possibility of customizing them is still needed. An option for this would be to add a spot where the user can type in the value of the weights.

In section 3.1.3, we introduced the Discriminant Power. The Discriminant Power is largely used for feature selection in machine learning. Feature selection could potentially be added to the OpenML platform via the Discriminant Power. This could be done in a dashboard, where the user selects a feature from the data, and can see its discriminant power.

Finally, another potential improvement would be adding visualizations to the platform. The confusion matrix is integrated as a metric, but it is not visual. A visualization of the confusion matrix could be done with a heatmap. Furthermore, the initial idea of this bachelor project was to include visualizations. These visualizations would inform the user on how its model performs relative to other users' models. These ideas could be of interest to OpenML since metrics alone are less interpretable. Visualizations are easy to read, which makes points of interest to analyse more apparent.

Bibliography

- [1] R. Batuwita and V. Palade. A new performance measure for class imbalance learning. application to bioinformatics problems. In *2009 International Conference on Machine Learning and Applications*, pages 545–550. IEEE, 2009.
- [2] B. J. Biggerstaff. Comparing diagnostic tests: a simple graphic using likelihood ratios. *Statistics in medicine*, 19(5):649–663, 2000.
- [3] K. Blagec, G. Dorffner, M. Moradi, and M. Samwald. A critical analysis of metrics used for measuring progress in artificial intelligence. *arXiv preprint arXiv:2008.02577*, 2020.
- [4] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae). *Geoscientific Model Development Discussions*, 7(1):1525–1534, 2014.
- [5] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.
- [6] D. Chicco, M. J. Warrens, and G. Jurman. The matthews correlation coefficient (mcc) is more informative than cohen’s kappa and brier score in binary classification assessment. *IEEE Access*, 9:78368–78381, 2021.
- [7] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.
- [8] C. Ferri, J. Hernández-Orallo, and R. Modroi. An experimental comparison of performance measures for classification. *Pattern recognition letters*, 30(1):27–38, 2009.
- [9] Forbes. Roundup of machine learning forecasts and market estimates, 2020.
- [10] V. García, R. A. Mollineda, and J. S. Sánchez. Theoretical analysis of a performance measure for imbalanced data. In *2010 20th International Conference on Pattern Recognition*, pages 617–620. IEEE, 2010.
- [11] M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.
- [12] A. Gupta, N. Tatbul, R. Marcus, S. Zhou, I. Lee, and J. Gottschlich. Class-weighted evaluation metrics for imbalanced data classification. *arXiv preprint arXiv:2010.05995*, 2020.

-
- [13] M. Hossin and M. N. Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.
- [14] N. Japkowicz. Assessment metrics for imbalanced learning. *Imbalanced learning: Foundations, algorithms, and applications*, pages 187–206, 2013.
- [15] L. A. Jeni, J. F. Cohn, and F. De La Torre. Facing imbalanced data—recommendations for the use of performance metrics. In *2013 Humaine association conference on affective computing and intelligent interaction*, pages 245–251. IEEE, 2013.
- [16] M. Kubat, R. C. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2):195–215, 1998.
- [17] A. Maratea, A. Petrosino, and M. Manzo. Adjusted f-measure and kernel scaling for imbalanced data learning. *Information Sciences*, 257:331–341, 2014.
- [18] N. Pearce, R. Beasley, and J. PEKKANEN. Role of bronchial responsiveness testing in asthma prevalence surveys, 2000.
- [19] R. Ranawana and V. Palade. Optimized precision—a new measure for classifier performance evaluation. In *2006 IEEE international conference on evolutionary computation*, pages 2254–2261. IEEE, 2006.
- [20] K. Ruffbach. Use of brier score to assess binary predictions. *Journal of clinical epidemiology*, 63(8):938–939, 2010.
- [21] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.
- [22] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.
- [23] D. Steen. Precision recall curve, 2020.
- [24] A. M. Turing. Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer, 2009.
- [25] F. J. Valverde-Albacete and C. Peláez-Moreno. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox. *PloS one*, 9(1):e84217, 2014.
- [26] C. J. Van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of documentation*, 1977.
- [27] C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.
- [28] L. Wu, P. Fernandez-Loaiza, J. Sauma, E. Hernandez-Bogantes, and M. Masis. Classification of diabetic retinopathy and diabetic macular edema. *World journal of diabetes*, 4(6):290, 2013.

Appendix A

Explanation of Metrics

A.1 Code

A.1.1 isBalanced() Function

```
from collections import Counter
def isBalanced(df, thresh = 0.70):
    """Function that returns whether the data is imbalanced.
    Parameters:  df : Input data
                thresh : max proportion of minority class instances over
                        majority class instance to be considered imbalanced."""
    counts = Counter(list(df['truth']))
    maxclass = max(counts, key = counts.get)
    minclass = min(counts, key = counts.get)
    if counts[maxclass] * thresh > counts[minclass]:
        return False
    else:
        return True
```

A.1.2 Evaluation Metrics

```
import pandas as pd
from sklearn import metrics
import numpy as np
import math
def Gmean(self):
    """Geometric mean of Specificity (TN / (TN + FP)) and Recall (TP / (TP + FN))
    in Binary Problems"""
    tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
        ['prediction']).ravel()
    specificity = tn / (tn+fp)
    recall = tp / (tp + fn)
    return math.sqrt(recall * specificity)

def AdjustedGmean(self):
    """Geometric mean of Specificity and Recall but adjusted so that the majority
    class's weight is not sacrificed as much. Compatible only in Binary (yet)"""
```

```

tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
    ['prediction']).ravel()
Nn = (tn + fp) / (tn+ fp + tp + fn)
Np = (tp + fn) / (tn+ fp + tp + fn)
if Nn > Np:
    specificity = tn / (tn+fp)
    Agm = (Gmean(self) + (specificity * Nn)) / (Nn + 1)
else:
    recall = tp / (tp+fn)
    Agm = (Gmean(self) + (recall * Np)) / (Np + 1)
return Agm

def BalancedAccuracy(self):
    """Mean of the accuracy of every class. All classes counted equally.
    Compatible with MultiClass and Binary"""
    return metrics.balanced_accuracy_score(self.results['truth'], df['prediction'
        ])

def WeightedAccuracy(self, weights = [1, 1]):
    """Weighted Accuracy customizable by user. Compatible with MultiClass and
    Binary.
    weights: a list with the weight of the accuracy of the class (ordered).
    By default, WeightedAccuracy = BalancedAccuracy"""
    nr_labels = len(self.results['truth'].unique())
    if len(weights) != nr_labels:
        weights = [1] * nr_labels
    elif sum(weights) != nr_labels:
        raise ValueError('Weights are too large')
    else:
        conf_matrix = metrics.confusion_matrix(self.results['truth'], self.
            results['prediction'])
        weighted_accuracy = 0
        for i in range(nr_labels):
            tp = conf_matrix[i][i]
            all = 0
            for n in range(nr_labels):
                all += conf_matrix[n][i]
            class_accuracy = tp * weights[i] / all
            weighted_accuracy += class_accuracy
        weighted_accuracy = weighted_accuracy / nr_labels
    return weighted_accuracy

def AdjustedF1(self, pos_label = self.results['truth'].unique()[0]):
    """Adjusted F1 that gives more weight to the minority class than normal F1.
    It is compatible in Binary only. pos_label is user defined"""
    F2measure = metrics.fbeta_score(self.results['truth'], self.results['
        prediction'], beta = 2, pos_label = pos_label)
    InvF0_5 = metrics.fbeta_score(self.results['truth'], self.results['prediction
        '], beta = 0.5, pos_label = self.results['truth'].unique()[1])
    return math.sqrt(F2measure * InvF0_5)

def OptimizedPrecision(self):
    """Optimizes Precision based on the imbalance of the data and, specificity
    and recall. It is supported in binary only. No parameters"""
    tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
        ['prediction']).ravel()
    specificity = tn / (tn+fp)

```

```

    recall = tp / (tp + fn)
    Nn = tn + fp
    Np = tp + fn
    OP = (specificity * Nn + recall * Np) / (Nn + Np) - abs(specificity - recall)
        / (specificity + recall)
    return OP

def IBalancedMeasure(self, alpha = 0.2, measure = 'accuracy_score'):
    """Index of Balanced Measure. Turns any metric into a metric for balanced
    data. It is supported for binary only. 2 parameters customizable by user:
    measure, and alpha. Highly customizable"""
    tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
        ['prediction']).ravel()
    specificity = tn / (tn+fp)
    recall = tp / (tp + fn)
    measure_result = eval("metrics.{}(df['correct'], df['prediction']).format(
        measure))
    Dom = recall - specificity
    IBM = (1 + alpha * Dom) * measure_result
    return IBM

def Accuracy(self):
    """Accuracy Score. Compatible with multiclass and binary"""
    return metrics.accuracy_score(self.results['truth'], self.results['prediction
        '])

def ROCAUC(self):
    """The Area under the Receiver Operating Characteristic Curve. Ranges from
    0–1. average = 'macro'.
    Compatible with Binary and Multiclass"""
    return metrics.roc_auc_score(self.results['truth'], self.results['prediction'
        ])

def PRAUC(self):
    """The Area under the Precision Recall Curve. Ranges from 0–1. average = '
    macro'
    Compatible with Binary and Multiclass"""
    return metrics.average_precision_score(self.results['truth'], self.results['
        prediction'])

def MacroF1(self):
    """Macro F1 Score for multiclass data"""
    return metrics.f1_score(self.results['truth'], self.results['prediction'],
        average = 'macro')

def MicroF1(self):
    """Micro F1 Score for multiclass data"""
    return metrics.f1_score(self.results['truth'], self.results['prediction'],
        average = 'micro')

def F1(self, pos_label = df['truth'].unique()[0]):
    """F1 Score for binary data"""
    return metrics.f1_score(self.results['truth'], self.results['prediction'],
        average = 'binary', pos_label = pos_label)

def Precision(self):
    """Precision for both Binary and Multiclass Data"""

```



```

if len(df['correct'].unique()) == 2:
    return metrics.precision_score(self.results['truth'], self.results['
        prediction'])
else:
    return metrics.precision_score(self.results['truth'], self.results['
        prediction'], average = 'macro')

def Recall(self):
    """Recall for both Binary and Multiclass Data"""
    if len(df['correct'].unique()) == 2:
        return metrics.recall_score(self.results['truth'], self.results['prediction
            '])
    else:
        return metrics.recall_score(self.results['truth'], self.results['prediction
            '], average = 'macro')

def Specificity(self):
    """Specificity for both Binary data"""
    tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
        ['prediction']).ravel()
    specificity = tn / (tn+fp)
    return specificity

def matthews_correlation_coefficient(self):
    """Matthews Correlation Coefficient calculates the correlation between the
        true values and the observed values.
        It is supported in both Multiclass and Binary data"""
    return metrics.matthews_corrcoef(self.results['truth'], self.results['
        prediction'])

def Youdens_Index(self):
    """Youdens Index is the sum of recall and specificity. It is a summary
        measure. Only for binary."""
    tn, fp, fn, tp = metrics.confusion_matrix(self.results['truth'], self.results
        ['prediction']).ravel()
    specificity = tn / (tn+fp)
    recall = tp / (tp + fn)
    Yindex = recall + specificity - 1
    return Yindex

def Brier_Score(self):
    """The Brier Score is the MSE between the true value and the predicted
        probability of the event happening"""
    return metrics.brier_score_loss(self.results['truth'], self.results['
        prediction'])

def Jaccard(self):
    """Jaccard score measures a model's ability to predict positives. Binary only
        """
    if len(df['correct'].unique()) == 2:
        return metrics.jaccard_score(self.results['truth'], self.results['
            prediction'])

def Kappa(self):
    """Measure of agreement between the correct values and the predicted values.
        Suggested for Multiclass only."""

```

```
    return metrics.cohen_kappa_score(self.results['truth'], self.results['
        prediction'])

def MAE(self):
    """The mean difference between the prediction and the true value"""
    return metrics.mean_absolute_error(self.results['correct'], self.results['
        prediction'])

def RMSE(self):
    """The root of the mean squared difference between the prediction and the
        true value"""
    return metrics.mean_squared_error(df['correct'], df['prediction'], squared =
        False)

def MSE(self):
    """The mean squared difference between the prediction and the true value"""
    return metrics.mean_squared_error(self.results['truth'], self.results['
        prediction'])
```