# ORES Custom Documentation V

*Disclaimer: No guarantee for the correctness of information / explanations / sources is given.*

## Goals

1. Metric list:
   - Add examples to each metric (if possible)
   - Adjust the description of counts → labels, predictions
2. Research what form of revision data is needed (for existing visualizations, but also in general) → also check RecentChanges and new Filters and what API calls look like in that context
3. Watch "ROC curves and Area Under the Curve explained" and think about what parameters could be used in which ways to filter the output of the current UI (currently: X inputs → X outputs)

   Also check out: Precision-Recall AUC vs ROC AUC discussion
4. Read A Review of User Interface Design for Interactive Machine Learning (carefully!)
5. Think about what could be the goal of this thesis

# 1 Crucial metrics: damaging-model

Metrics simple list:

| | |
|---|---|
| !f1 | ✓ |
| !precision | ✓ |
| !recall | ✓ |
| accuracy | ✓ |
| counts | ✓ |
| f1 | ✓ |
| filter_rate | ✓ |
| fpr | ✓ |
| match_rate | ✓ |
| pr_auc | ✓ |
| precision | ✓ |
| rates | ✓ |
| recall | ✓ |
| roc_auc | ✓ |

Again, changes have been made to the list and explanations, compared to the version in oresDoc4. The structure of explanations will be as follows:

For each metric (if possible) there will be:

1. The formula based on the **confusion matrix**
2. A definition
3. An intuitive explanation with an example
4. Its meaning based on the **loan threshold** representation by Google (Link)

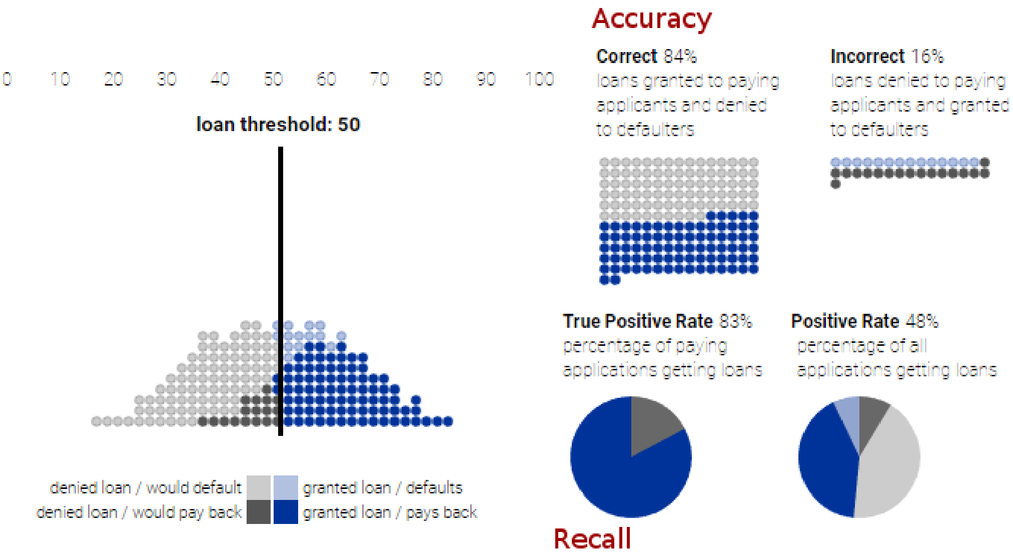
## Explanations: References

- Confusion Matrix

| | | Actual | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | **True Positive** | **False Positive** |
| | Negative | **False Negative** | **True Negative** |

Abbreviations: **TP**, **FP**, **FN** and **TN**.

- Loan Threshold

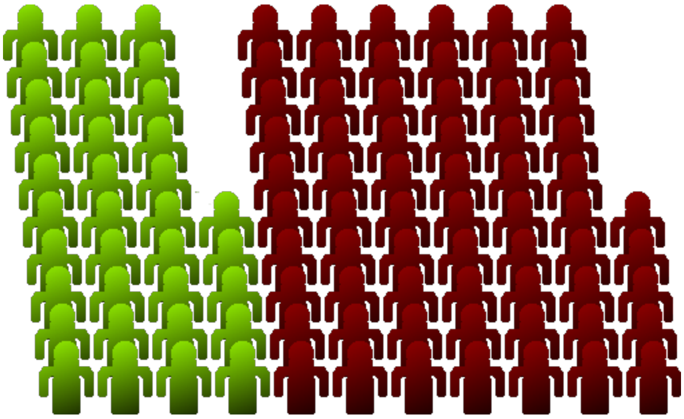


## The example scenario

To ease the understanding, let's stick to the following scenario and refer to it for each metric:

- 100 people represent our total population
- 35% of our population is infected with disease X

  That leaves us with the following **labels**: **35 positives** and **65 negatives** (being tested for disease X;)



- A classifier is now supposed to classify every person of our population (based on their visible symptoms for example). This is the **prediction**:

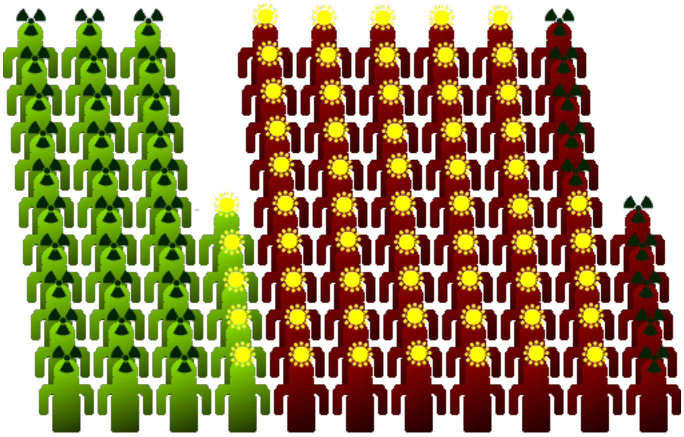
  - If our algorithm says a person is infected, that person will be classified as a **positive**, and marked with radioactive symbol:

    ☢

  - If the prediction results in a **negative**, the person will be marked with a sun symbol:

    ☀

- The classifier may predict that:
  - out of the 35 infected people, **30** are infected (those 30 are what we call **true positives**) and **5** are not (those 5 are **false negatives**)
  - out of the 65 non infected people, **10** are infected (**false positives**) and **55** are not (**true negatives**)



**true positive**: infected and correctly predicted

**false negative**: infected and incorrectly predicted

**true negative**: not infected and correctly predicted

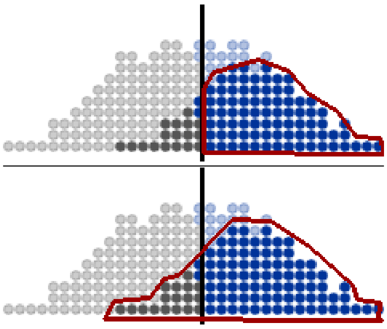
**false positive**: not infected and incorrectly predicted

**Let's get started.**

## 1.1 recall

1. $\frac{\texttt{TP}}{\texttt{TP+FN}}$
2. Recall (≡ true positive rate ≡ “sensitivity”) is the ability of a model to find **all** relevant cases within the dataset.
3. Now, in our example scenario, the relevant cases are the infected people. We absolutely want to identify those: 🟢. The ability of the model to identify those depends on how many will be **correctly** predicted to be infected: 🟩.

   In other words, we are looking for the ratio of correctly predicted to be infected people to all infected people.

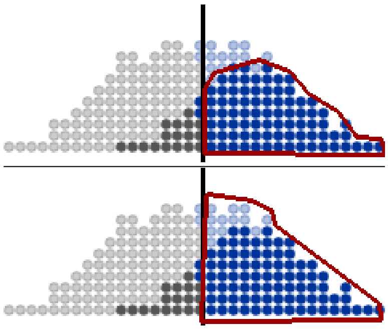
   That leads to $\frac{\sum \text{🟩}}{\sum \text{🟢}}$, with 🟢 $=$ 🟩 $+$ 🟨, which is equivalent to the formula in **1.**, if you replace the symbols with their confusion matrix counterpart according to the legend in **The example scenario**.
4. 

## 1.2 precision

1. $\frac{\texttt{TP}}{\texttt{TP+FP}}$
2. Ability of the model to find **only** relevant cases within the dataset
3. Again, we take a look at the relevant cases, the infected people: 🟢. This time around though, we are **not** interested in the ratio of correctly predicted to be infected people to **all** infected people. Instead we want to know how good the model is at only predicting those to be infected, that actually are. Therefore, we want the ratio of all 🟩 to all those predicted to be infected: 🟩 + 🟥

$$= \frac{\sum \text{🟩}}{\sum \text{🟩} + \sum \text{🟥}}$$

4.



## 1.3 f1

1. F1-Score, the harmonic mean of recall and precision, a metric from 0 (worst) to 1 (best), used to evaluate the accuracy of a model by taking recall and precision into account

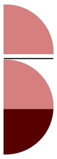
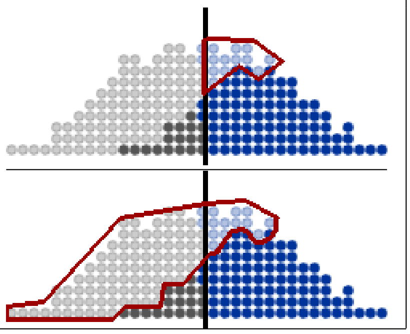
| 2. | 3. | 4. |
|---|---|---|
| - | - | - |

$$= 2 * \frac{\texttt{precision}*\texttt{recall}}{\texttt{precision}+\texttt{recall}}$$

Compared to the simple average (of recall and precision), the harmonic mean punishes extreme values (e.g. precision 1.0 and recall 0.0 → average 0.5, but F1 = 0)

## 1.4 fpr

1. The false positive rate (**FPR**) is the probability of a false alarm

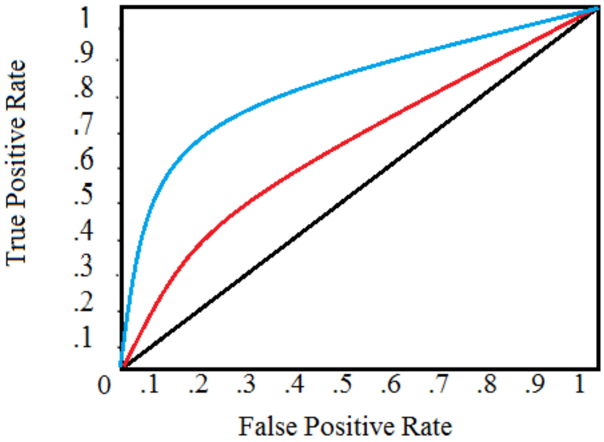
| 2. | 3. | 4. |
|---|---|---|
| $= \frac{\texttt{FP}}{\texttt{FP}+\texttt{TN}}$ | =  | =  |

## 1.5 roc_auc

1. The **area under** the **curve** of the **ROC**-curve, a measure between 0.5 (worthless) and 1.0 (perfect: getting no FPs), rates the ability of a model to achieve a blend of recall and precision

| 2. | 3. | 4. |
|---|---|---|
| - | - | - |

The receiver operating characteristic (ROC) curve plots the TPR versus FPR as a function of the model's threshold for classifying a positive



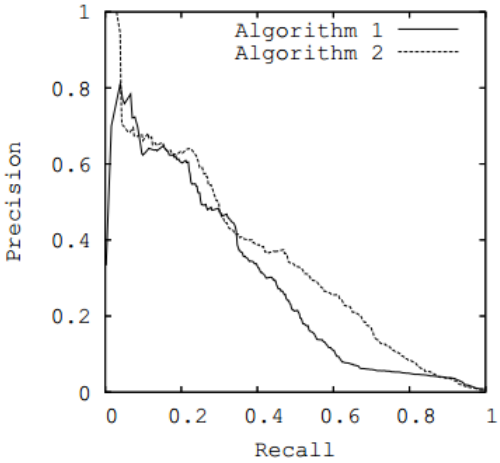
Increasing the threshold → moving up a curve (≡ model) to the top right corner, where all data is predicted as positive (threshold = 1.0) and vice versa

## 1.6 pr_auc

(see: link 1 and link 2)

1. The **area under** the **curve** of the **PR**-curve, same: similar objective as the **roc_auc**, but PR curves are better than ROC curves if the populations are imbalanced

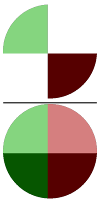
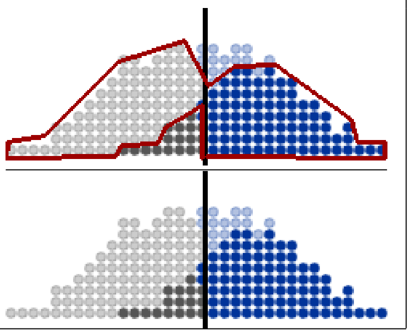
The PR-curve plots the Precision versus the Recall



Instead of the top left corner for the ROC-curve, here, we want to be in the top right corner for our classifier to be perfect

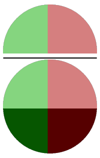
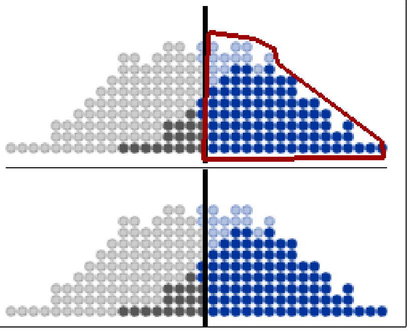
## 1.7 accuracy

1. Measuring the portion of correctly predicted data

| | | |
|---|---|---|
| 2.<br>$= \frac{\mathtt{TP+TN}}{\mathtt{Total}}$ | 3.<br>=  | 4.<br>=  |

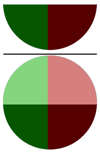
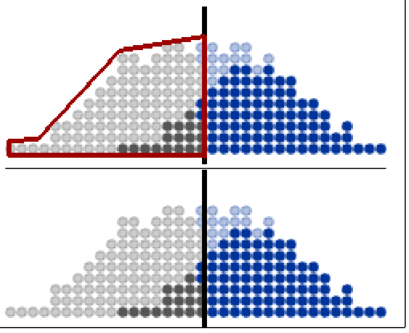
## 1.8 match_rate

1. The proportion of observations matched/not-matched

| 2. | 3. | 4. |
|---|---|---|
| $= \frac{\texttt{TP+FP}}{\texttt{Total}}$ | =  | =  |

## 1.9 filter_rate

1. The proportion of observations filtered/not-filtered

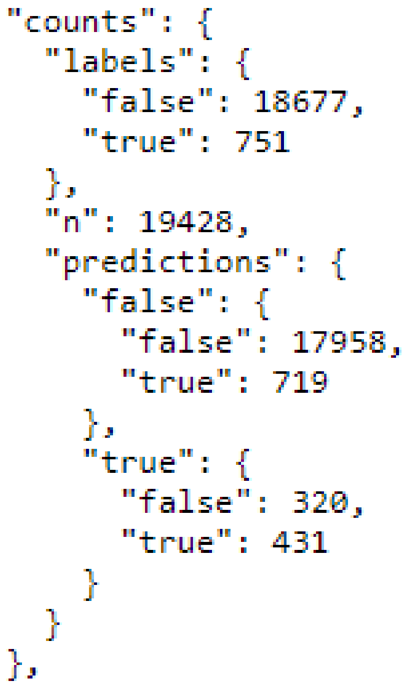
| 2. | 3. | 4. |
|---|---|---|
| $= 1 - \texttt{match\_rate} = \frac{\texttt{TN+FN}}{\texttt{Total}}$ | =  | =  |

## 1.10 counts

1. The number of edits labeled as *false* and *true*

| 2. | 3. | 4. |
|---|---|---|
| - | - | - |

When calling the enwiki damaging model for example (Link), you'll notice the following under counts:



```
"counts": {
  "labels": {
    "false": 18677,
    "true": 751
  },
  "n": 19428,
  "predictions": {
    "false": {
      "false": 17958,
      "true": 719
    },
    "true": {
      "false": 320,
      "true": 431
    }
  }
},
```

The information stored in *labels* is what we defined above under **1.**

*n* is the total number of edits taken into account

*predictions* is where it gets interesting: *false* and *true* as parents of another two of those are the labels and the values of their "child"-booleans are the actual values of their edits

⇒ e.g. of 18677 edits that were labeled as *false*, 719 were false negatives

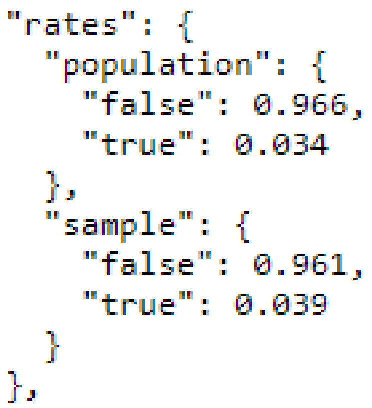
## 1.11 rates

1. The rates simply equal $\frac{\texttt{label}}{\texttt{n}}$, both from **counts** for *rates*: *sample*: *label*, and with *label* = *false* or *true*

   So *rates*: *sample*: *false*: $0.961 = \frac{18677}{19428}$

| 2. | 3. | 4. |
|---|---|---|
| - | - | - |

Calling the API the same way as for **counts**, we get:



```
"rates": {
  "population": {
    "false": 0.966,
    "true": 0.034
  },
  "sample": {
    "false": 0.961,
    "true": 0.039
  }
},
```

As already mentioned, the size of *sample* equals our $n$ in **counts** $= 19428$.

But why have a sample and the whole population? Because there is a significant number of bot edits and edits that don't need reviewing (admins, autopatrolled users). The sample of edits does not contain any of those.

## 1.12 !<metric>

- Any <metric> with an exclamation mark is the same metric for the negative class
- e.g. recall $= \frac{\texttt{TP}}{\texttt{TP+FN}} \Rightarrow$ **!**recall $= \frac{\texttt{TN}}{\texttt{TN+FP}}$
- Example usage: find all items that are not "E" class → look at **!**recall for "E" class.

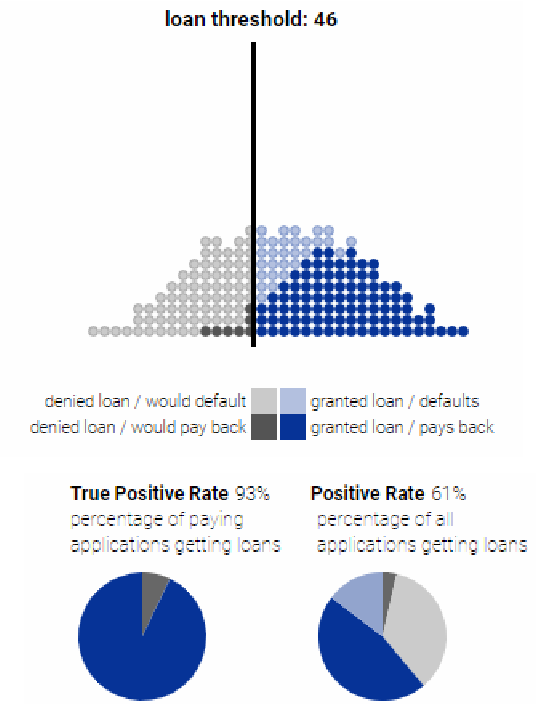
### 1.12.1 Existing !<metric>s

- !f1
- !precision
- !recall

## 1.13 Additional explanations

### 1.13.1 recall vs precision

When increasing one of these two, the other one naturally decreases. For an intuitive example, let's take a look at Google's Loan Threshold Simulation:



The dark grey / dark blue dots, representing clients that would actually pay back their loan, are more and more included (→ given loans) if we move the threshold further to the left.

But so are clients that would not. Thus moving the threshold to the left increases the **recall** (**tpr**) but decreases the **precision** and vice versa when moving to the right.

### 1.13.2 roc_auc vs pr_auc

see: `https://www.kaggle.com/general/7517`

- tl;dr: if the class imbalance problem exists, **pr_auc** is more appropriate than **roc_auc**

  If TNs are not meaningful to the problem or there are a lot more negatives than positives, **pr_auc** is the way to go (it does not account for TNs).

- In other words:
  - If the model needs to perform equally on the positive and negative class → **roc_auc**

  - If it's not interesting how the model performs on negative class → **pr_auc** (example: detecting cancer; find all positives and make sure they're correct!)

## Questions

- Q: Should I ask Aaron how he would like us to work together? I'm not sure how he meant it.

  A:

- Q: In what situations exactly do we want to optimize the threshold in the context of user centered threshold optimization?

  A: