ORES Custom Documentation IV

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

Goals

1. Adjust crucial metrics list
   - to match the damaging model metrics ✓
   - add 2 last metrics ✓
2. Check out mail attachments ✓
3. Check out new Confluence pages and goals ✓
4. Research
   - Check out FAT Conference Docs
   - In what other cases than confusion matrices are those parameters explained?
   - Are there already visualizations of some of these parameters in any contexts?
   - Are there any applications, where I can filter for these parameters → visualizations or just about anything?

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	✓

The metrics are the same for the damaging and itemquality models, but a few changes will be made to the explanatory parts nontheless (compared to the version in oresDoc3). Also the structure of explanations will be changed to the following:

For each metric (if possible) there will be:

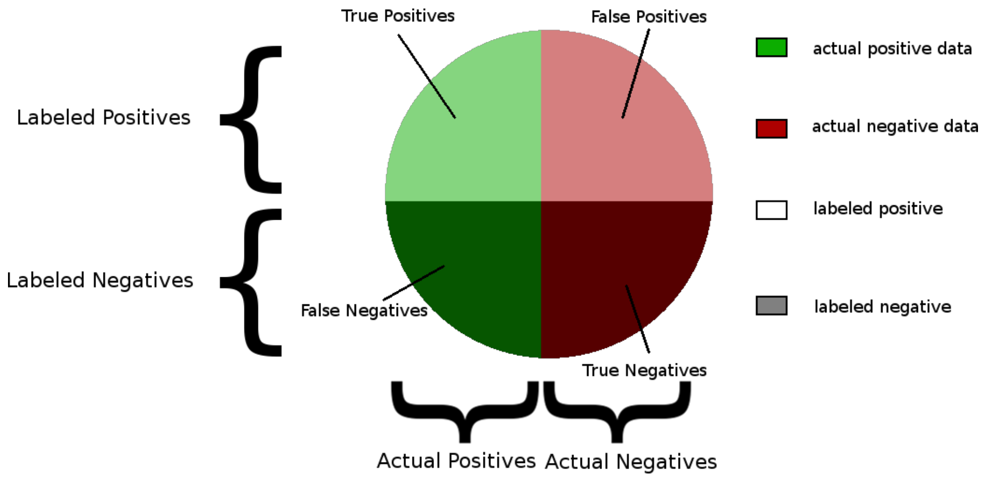
1. An intuitive explanation
2. The formula based on the **confusion matrix**
3. Its meaning based on the "**confusion circle**"
4. Its meaning based on the **loan threshold** representation by Google (Link)

Explanations: References

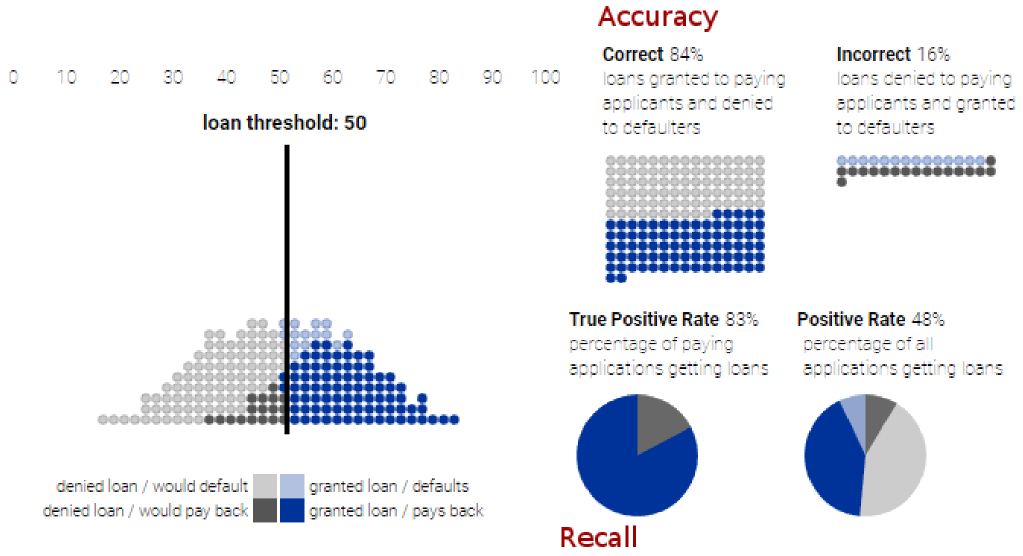
- Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	**True Positive**	**False Positive**
	Negative	**False Negative**	**True Negative**

- “Confusion Circle”

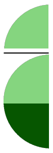
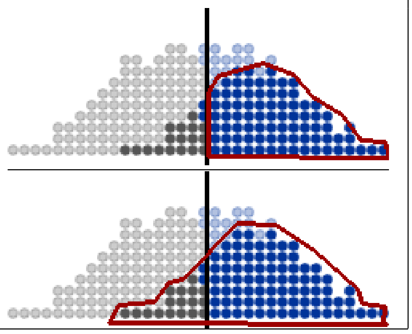


- Loan Threshold



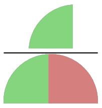
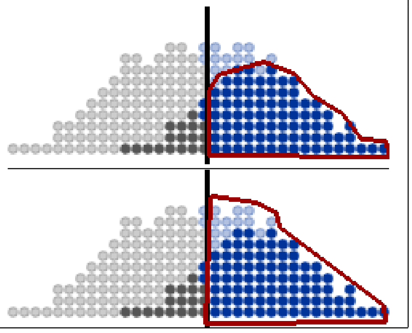
1.1 recall

1. Recall (≡ True Positive Rate) is defined as the ability of a model to find all relevant cases within the dataset.

2.	3.	4.
$= \frac{\texttt{TP}}{\texttt{TP+FN}}$	$=$ 	$=$ 

1.2 precision

1. Ability of the model to find only relevant cases within the dataset

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

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

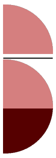
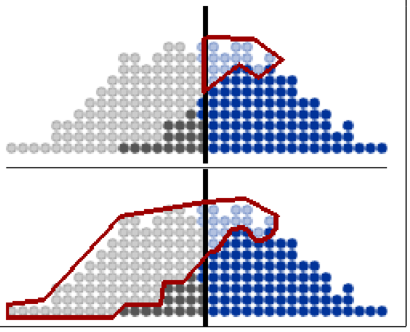
2.	3.	4.
-	-	-

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

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

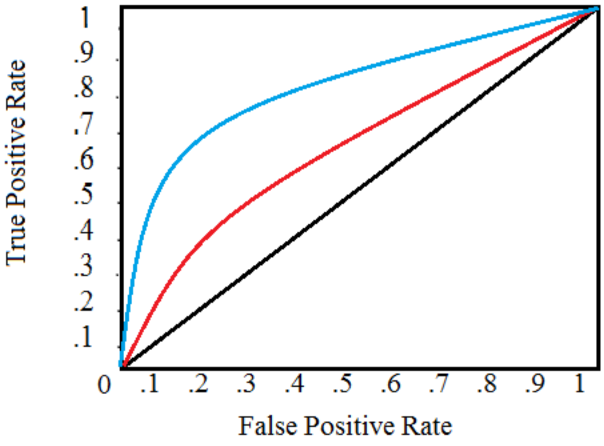
2.	3.	4.
$= \frac{\texttt{FP}}{\texttt{FP+TN}}$	= 	= 

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



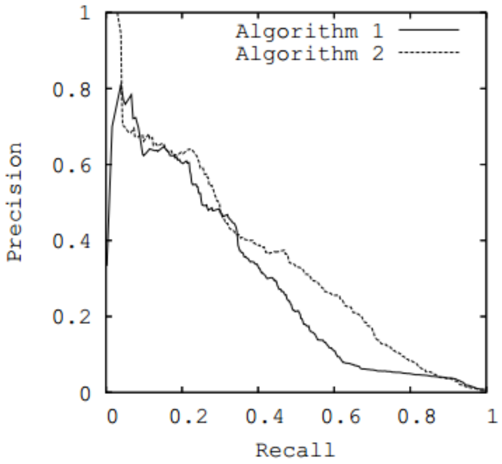
Increasing the threshold → moving up a curve (≡ model) to the top right corner, where all data is predicted as positive (threshold = 1.0) and vice versa

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

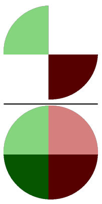
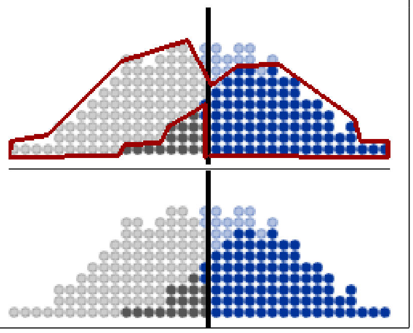
The PR-curve plots the Precision versus the Recall



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

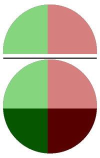
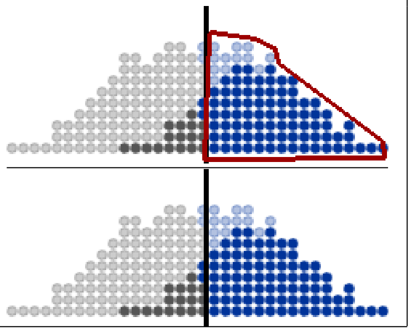
1.7 accuracy

1. Measuring the portion of correctly predicted data

2. $= \frac{\texttt{TP+TN}}{\texttt{Total}}$	3. = 	4. = 

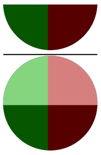
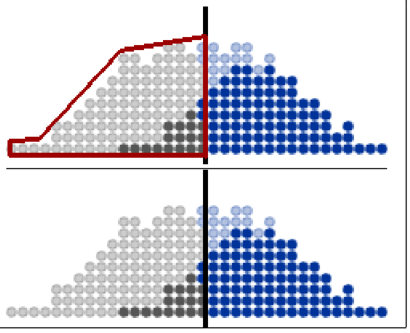
1.8 match_rate

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

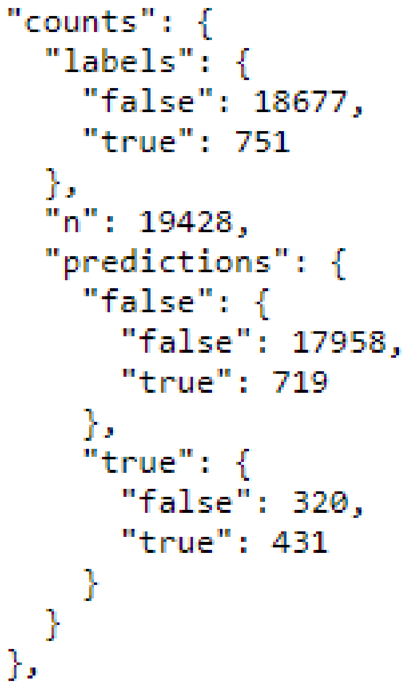
2.	3.	4.
$= 1 - \texttt{match\_rate} = \frac{\texttt{TN+FN}}{\texttt{Total}}$	$=$ 	$=$ 

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

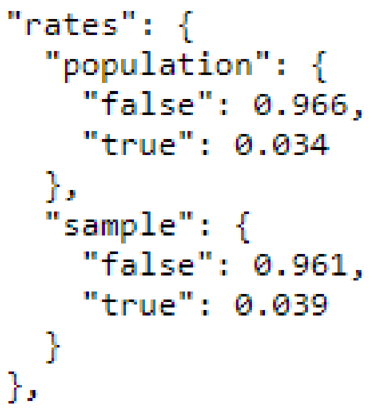
1.11 rates

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 $= \frac{\mathtt{TP}}{\mathtt{TP+FN}} \Rightarrow$ **!**recall $= \frac{\mathtt{TN}}{\mathtt{TN+FP}}$
- Example usage: find all items that are not "E" class → look at **!**recall for "E" class.

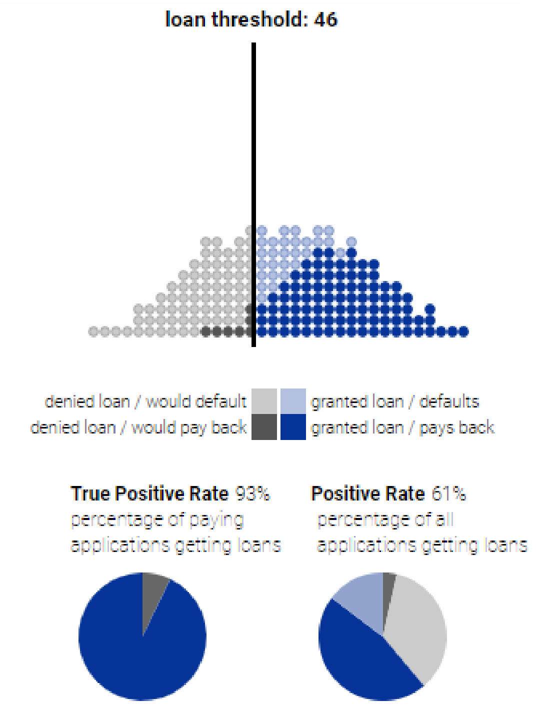
1.12.1 Existing !<metric>s

- !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).

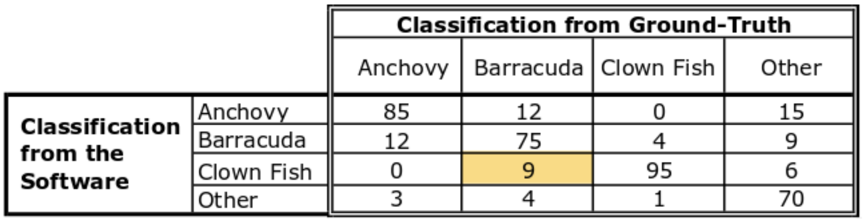
- In other words:
  - If the model needs to perform equally on the positive and negative class → **roc_auc**

  - 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!)

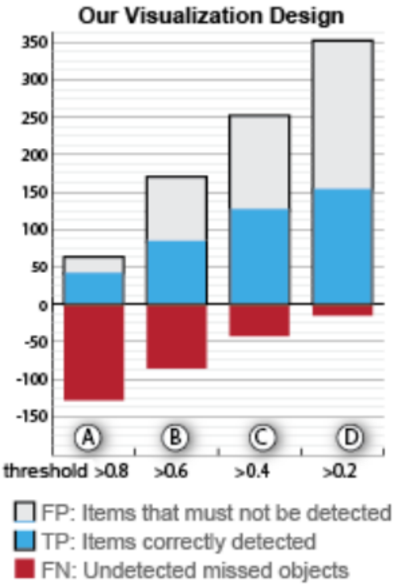
2 Research

2.1 Confusion Matrices Alternatives

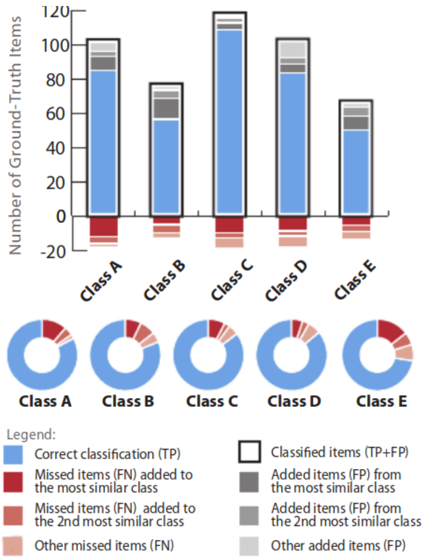
- Already mentioned before: Confused Pie Plots (Link)
- "Visualization of Confusion Matrix for Non-Expert Users" (Link)



		Classification from Ground-Truth			
		Anchovy	Barracuda	Clown Fish	Other
Classification from the Software	Anchovy	85	12	0	15
	Barracuda	12	75	4	9
	Clown Fish	0	9	95	6
	Other	3	4	1	70



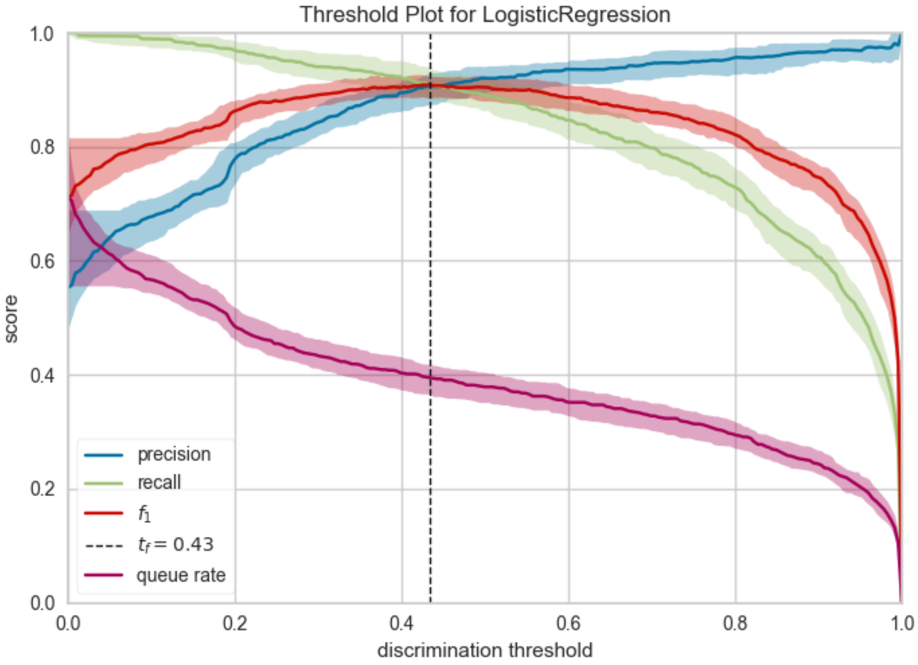
- Same project: (Link)



Personal comment: Since the most commonly mentioned problem with confusion matrices is the readability with a high amount of classes, these (and probably most other) alternative representations aim to improve that aspect, thus they might not be extremely useful for visualizations in the context of the **damaging**-model.

2.2 Classifier Visualizations

- Scikit Discrimination Threshold Visualization (Link)



For binary probabilistic classifiers only (fits the **damaging**-model).

→ Is this interesting to us? We need the dataset (see link for code)!

If it is, also check other visualizations on the site!

Additional Information

- ORES Threshold Link
- Confluence Link (Bachelor Thesis)
- Amir’s mail links
  - New Filters for Edit Review Documentation Link
  - ORES API Call Link

    > “Basically it asks for threshold from the API when ”recall is at its maximum when precision is at least 0.995“’
- Bowen Yu: Applying Value-Sensitive Algorithm Design to ORES

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:

- Q: VPN recommendation?

  A: