# ORES Custom Documentation VI

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

## Goals

1. Metrics List: Create Table as a general quickview ✓
2. Metrics: which combinations are particularly useful, which are nonsensical?
   - Ask for documentation on IRC (✓)
   - Logically exclude combinations?
   - Document outputs
3. Recent Changes filter classes: how are edits assigned to them?
   - Also ask for documentation on IRC ✓
   - Which metrics are included in the process? ✓
   - How are the metrics (precision, recall, threshold) included in the associated API calls? What do the (GET?)-Requests look like? ✓
4. Take a closer look at the Threshold Plot for Logistic Regression (Link)
   - What is the meaning of the areas around the curves? ✓
   - What is queue rate exactly? ✓
5. Take a closer look at the Swagger API Documentation
6. !!! Improve knowledge of ORES Docs (metrics!!)

# 1 Metrics List: Table

| Metric | Quick Definition | Value |
|---|---|---|
| accuracy | Portion of correctly predicted data | $\frac{\texttt{TP+TN}}{\texttt{Total}}$ |
| counts | Number of **F**&**T**-labels and predictions | |
| f1 | Harmonic mean of recall and precision | $2 * \frac{\texttt{rec*prec}}{\texttt{rec+prec}}$ |
| filter_rate | Portion of observations predicted to be negative | $1 - \texttt{match\_rate} = \frac{\texttt{TN+FN}}{\texttt{Total}}$ |
| fpr | Probability of a false alarm | $\frac{\texttt{FP}}{\texttt{FP+TN}}$ |
| match_rate | Portion of observations predicted to be positive | $\frac{\texttt{TP+FP}}{\texttt{Total}}$ |
| pr_auc | Measure of classification performance | |
| precision | Ability to find only relevant cases | $\frac{\texttt{TP}}{\texttt{TP+FP}}$ |
| rates | Proportion of **F**&**T**-labels to the total | |
| recall | Ability to find **all** relevant cases | $\frac{\texttt{TP}}{\texttt{TP+FN}}$ |
| roc_auc | Measure of classification performance | |
| !f1 | Negated f1 | $2 * \frac{\texttt{!rec*!prec}}{\texttt{!rec+!prec}}$ |
| !precision | Negated precision | $\frac{\texttt{TN}}{\texttt{TN+FN}}$ |
| !recall | Negated recall | $\frac{\texttt{TN}}{\texttt{TN+FP}}$ |

# 2 Metrics combinations - or - "syntax for requesting an optimization from ORES" (source: paper)

example: `https://ores.wikimedia.org/v3/scores/enwiki/?models=damaging&model_info=statistics.thresholds.true."maximumrecall@precision>=0.9"` More links for quickstart:

- 1
- 2
- 3
- 4

Querying the API as in the example above, the output, before applying the given filters of **recall** and **precision**, is the list of values of all available metrics for thresholds (Link). Asking for the maximum recall in a specific precision interval (greater or equal 0.9), the final output is just the one set of values (Link).

**Note:**

- The metrics counts and rates are not part of the available parameters for this type of query. Neither is threshold.
- Other than that, all combinations of metrics that can be seen (e.g. in the output of the example link) can be used.
- Queries with values outside of the possible interval (e.g. maximum **recall** @ **precision** >= 1.1) or queries that simply do not return any results (e.g. maximum filter_rate @ f1 >= 0.5; f1 does not yield a value of 0.5 or higher for any threshold) return null:


```
{
  "enwiki": {
    "models": {
      "damaging": {
        "statistics": {
          "thresholds": {
            "true": [
              null
            ]
          }
        }
      }
    }
  }
}
```


## 2.1 (Some) Useful combinations

- maximum **recall** @ **precision** >= very high (e.g. 0.9)
  - Determines a "useful threshold for a counter-vandalism bot".
  - **Recall** and **precision** is one of the most important trade-offs to adjust.
  - A bot that automatically reverts damaging edits should only operate at high **precisions** in order to make the least mistakes while reverting and not to cause lots of frustration for well behaving, misunderstood and sometimes inexperienced users (e.g. the rejection of good-faith newcomers).

  - It therefore makes sense to set a **precision** lower bound, high enough so that every value above seems acceptable, and then search for the threshold, that returns the highest possible **recall**.

- maximum filter_rate @ recall >= high (e.g. 0.75)
  - Determines "a useful threshold for semi-automated edit review".
  - With semi-automated tools, we have our algorithms marking edits as potentially damaging and then specific users looking into those. We can therefore typically afford lower **precision** than in fully automated review tools and, as a consequence, we can expect higher **recall**.
  - Having fixed a reasonably high **recall** lower bound, we still want to minimize the work necessary by reviewers. That is why we could maximize the **filter_rate**

# 3 Recent Changes Quality Prediction Filters

The Recent Changes quality prediction filters are a helpful tool in varying the precision and recall of catching damaging edits. They can be applied on the Recent changes site (Link).

## Contribution quality predictions

| Filter | Precision | Recall | Threshold range | |
|---|---|---|---|---|
| **Very likely good** | 99% | 91.1% | 0 | 0.315 |
| **May have problems** | 15% | 86.3% | 0.144 | 1 |
| **Likely have problems** | 45.7% | 48.1% | 0.612 | 1 |
| **Very likely have problems** | 90% | 8.2% | 0.912 | 1 |

Wikipedia Source

## 3.1 Precision and Recall

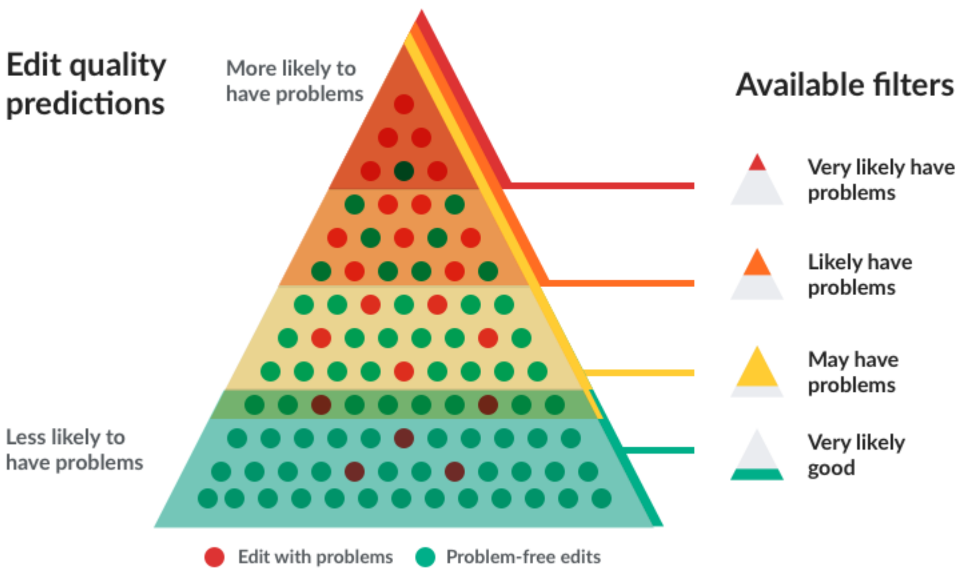
To put those numbers into context: we can expect that, for example, the *Likely have problems* filter will be right about 45.7% of the time, classifying a contribution as damaging while catching 48.1% of problem edits.

## 3.2 Threshold ranges

Keep in mind that the damaging model classifier is binary, deciding wether a contribution is damaging or not, but does not only output 0 or 1, but instead a probability (between 0 and 1) of how probable it is, that a contribution is damaging. The threshold then decides at which probability the contributions are split into damaging and non-damaging (e.g. a threshold of 0.6 means that every contribution with a value of under 0.6 will be classified as non-damaging and vice versa). Threshold ranges therefore signify the following:

- *Very likely good*: Contributions that score between 0 and 0.315 on the probability-of-being-damaging scale
- *May have problems*: Contributions that score between 0.144 and 1
- *Likely have problems*: Contributions that score between 0.612 and 1
- *Very likely have problems*: Contributions that score between 0.912 and 1

To better understand threshold ranges it's helpful to also take a look at the following graphic:




Wikimedia Source, MediaWiki Source
Note two things:

1. The *May have problems*, *Likely have problems*, and *Very likely have problems* filters overlap.

   More precisely, the most general filter *May have problems* contains all edits that also are part of the other two filters and, similarly, edits of *Very likely have problems* are a subset of *Likely have problems*.

2. The *Very likely good* and *May have problems* filters overlap.

   This overlap is referred to as "the indeterminate zone between problem and problem-free edits" and happens in order to achieve a "broader recall".

## 3.3 Highlighting

It is also possible to apply filters, for example the broadest one *May have problems* and then highlight edits marked as particularly alarming by using color highlighting for the *Likely have problems* and *Very likely have problems* filters.
Read more on highlighting: Mediawiki Link.

## 3.4 Recent Changes ORES Requests

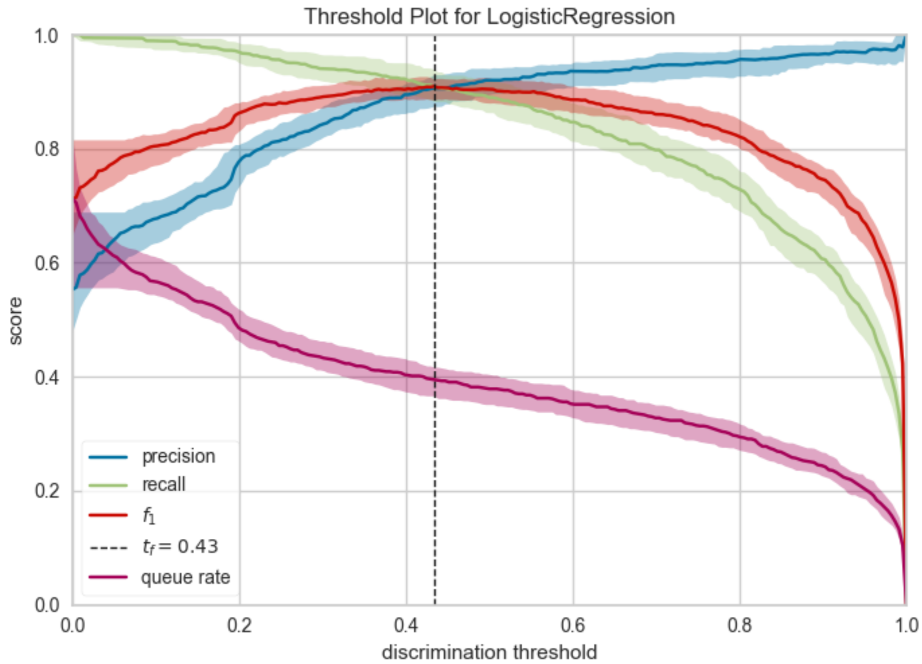
Here is how it is decided to which filter category contributions belong:

1. MediaWiki sends requests to ORES, as soon as edits are saved, to get the **score.probability.true** of edits. (See source: MediaWiki Request to ORES)

2. The value of **score.probability.true** is then stored in a table (ores_classification table) that can be joined to the recent_changes table.

3. Having the threshold ranges for all available filters, an edit's **score.probability.true** has to be a value in the threshold ranges interval for the edit to be a part of that filter category.

# 4 Discrimination Threshold Visualisation (Logistic Regression)




## 4.1 Areas - or bands - around the curves

The model will split the data multiple times, differently, into train and test sets and then run the trials. This ensures a certain amount of variability being visualized. Corresponding section on the site:

> "*The visualizer also accounts for variability in the model by running multiple trials with different train and test splits of the data. The variability is visualized using a band such that the curve is drawn as the median score of each trial and the band is from the 10th to 90th percentile.*"

## 4.2 Queue rate

"This metric describes the percentage of instances that must be reviewed." It can be helpful to think about the costs of reviewing whatever it is that must be reviewed in the context of business decisions, where the ability to review is a limited resource and might be a factor in adjusting the threshold in order to find a favourable outcome.