Evaluations

Release 0.0.1

Yaroslav Isaienkov

Aug 14, 2020

MODULES

1 Medical Metrics **3**
1.1 Concordance Index 3

2 Kaggle 2020 **5**
2.1 Global Average Precision 5
2.2 Row-Wise Micro Averaged F1 6

3 Classification **9**
3.1 Accuracy 9
3.2 Confusion Matrix 9

4 Text Extraction **11**
4.1 Jaccard Word Level 11

Index **13**

Evaluations library implements various metrics for evaluating machine learning algorithms

GitHub repository: https://github.com/yisaienkov/evaluations

CHAPTER

ONE

MEDICAL METRICS

1.1 Concordance Index

`evaluations.medical.concordance_index.`**`concordance_index`**(*events: List[int], risks: List[float]*) → float

Compute Concordance index (C-index)

The concordance index is a value between 0 and 1 where:

- 1.0 - all risk scores for happened events higher than for unhappened,
- 0.0 - all risk scores for unhappened events higher than for happened,
- 0.5 - random risk scores.

The formula is:

$$C\text{-}Index = \frac{n\_concordant\_pairs + 0.5 * n\_risk\_ties}{n\_permissible\_pairs}$$

where:

- n_concordant_pairs - number of permissible pairs where the score is bigger for the event with label 1 than 0
- n_risk_ties - number of permissible pairs with equal scores
- n_permissible_pairs - number of pairs with different events

Parameters

- **events** (*List[int]*) – If the some event happened - 1 or not happened - 0
- **risks** (*List[int]*) – Risks scores for each event

Returns Concordance index

Return type float

Examples

```
>>> from evaluations.medical import concordance_index
>>> events = [1, 0, 1, 1, 0]
>>> risks = [0.8, 0.43, 0.62, 0.58, 0.62]
>>> concordance_index(events, risks)
0.75
```

CHAPTER

TWO

KAGGLE 2020

2.1 Global Average Precision

The metric for Kaggle Competition Google Landmark Recognition 2020

2.1.1 Global Average Precision Score

N predictions (label/confidence pairs) sorted in descending order by their confidence scores, then the Global Average Precision is computed as:

$$GAP = \frac{1}{M} \sum_{i=1}^{N} P(i) rel(i)$$

- N is the total number of predictions returned by the system, across all queries
- M is the total number of queries with at least one sample from the training set visible in it (note that some queries may not depict samples)
- P(i) is the precision at rank i. (example: consider rank 3 - we have already made 3 predictions, and 2 of them are correct. Then P(3) will be 2/3)
- rel(i) denotes the relevance of prediciton i: it's 1 if the i-th prediction is correct, and 0 otherwise

`evaluations.kaggle_2020.global_average_precision.`**`global_average_precision_score`**(*y_true: Dict[Any, Any], y_pred: Dict[Any, Tuple[Any, float]]*) → float

Compute Global Average Precision score (GAP)

Parameters

- **y_true** (`Dict[Any, Any]`) – Dictionary with query ids and true ids for query samples
- **y_pred** (`Dict[Any, Tuple[Any, float]]`) – Dictionary with query ids and predictions (predicted id, confidence level)

Returns GAP score

Return type float

Examples

```
>>> from evaluations.kaggle_2020 import global_average_precision_score
>>> y_true = {
...         'id_001': 123,
...         'id_002': None,
...         'id_003': 999,
...         'id_004': 123,
...         'id_005': 999,
...         'id_006': 888,
...         'id_007': 666,
...         'id_008': 666,
...         'id_009': None,
...         'id_010': 666,
...     }
>>> y_pred = {
...         'id_001': (123, 0.15),
...         'id_002': (123, 0.10),
...         'id_003': (999, 0.30),
...         'id_005': (999, 0.40),
...         'id_007': (555, 0.60),
...         'id_008': (666, 0.70),
...         'id_010': (666, 0.99),
...     }
>>> global_average_precision_score(y_true, y_pred)
0.5479166666666666
```

2.2 Row-Wise Micro Averaged F1

The metric for Kaggle Competition Cornell Birdcall Identification

For each row we compute number of True Positives (TP), False Positives (FP) and False Negatives (FN) predictions.

For example, if true label is "bird1 bird2", and we predict "bird2 bird3 bird4", then we have:

- TP = 1 (bird2)
- FP = 2 (bird3, bird4)
- FN = 1 (bird1)

Then we calculate F1 score by the next formula:

$$F1\ Score = \frac{2 * TP}{2 * TP + FN + FP}$$

At the end we compute mean by all row scores.

2.2.1 Row-Wise Micro Averaged F1 Score

`evaluations.kaggle_2020.row_wise_micro_averaged_f1.`**`row_wise_micro_averaged_f1_score`**(*y_true: List[str], y_pred: List[str]*) → float

Compute row-wise micro averaged f1 score

Parameters

- **y_true** (*List[str]*) – Target list of strings of a space separated birds names
- **y_pred** (*List[str]*) – Predicted list of strings of a space separated birds names

Returns Row-wise micro averaged F1 score

Return type float

Examples

```
>>> from evaluations.kaggle_2020 import row_wise_micro_averaged_f1_score
>>> y_true = [
...         'amecro',
...         'amecro amerob',
...         'nocall',
...     ]
>>> y_pred = [
...         'amecro',
...         'amecro bird666',
...         'nocall',
...     ]
>>> row_wise_micro_averaged_f1_score(y_true, y_pred)
0.8333333333333333
```

2.2.2 Micro F1 Similarity

`evaluations.kaggle_2020.row_wise_micro_averaged_f1.`**`micro_f1_similarity`**(*y_true: str, y_pred: str*) → float

Compute micro f1 similarity for 1 row

Parameters

- **y_true** (*str*) – True string of a space separated birds names
- **y_pred** (*str*) – Predicted string of a space separated birds names

Returns Micro F1 similarity

Return type float

Examples

```
>>> from evaluations.kaggle_2020 import micro_f1_similarity
>>> y_true = 'amecro amerob'
>>> y_pred = 'amecro bird666'
>>> micro_f1_similarity(y_true, y_pred)
0.5
```

CHAPTER

THREE

CLASSIFICATION

3.1 Accuracy

3.1.1 Accuracy Score

$$Accuracy\ Score = \frac{The\ number\ of\ samples\ predicted\ correctly}{Total\ number\ of\ samples}$$

`evaluations.classification.accuracy.`**`accuracy_score`**(*y_true: List*, *y_pred: List*) → float

Compute accuracy score

Parameters

- **`y_true`** (*`list`*) – True labels
- **`y_pred`** (*`list`*) – Predicted labels

Returns Accuracy score

Return type float

Examples

```
>>> from evaluations.classification import accuracy_score
>>> accuracy_score([1, 1, 0, 0], [1, 1, 1, 0])
0.75
```

3.2 Confusion Matrix

3.2.1 Confusion Matrix Binary

`evaluations.classification.confusion_matrix.`**`confusion_matrix_binary`**(*y_true: List[int]*, *y_pred: List[int]*) → Dict[str, int]

Compute tp, tn, fp, fn

Parameters

- **y_true** (*list of ints*) – True labels
- **y_pred** (*list os ints*) – Predicted labels

Returns Dictionary with number of samples of tp, tn, fp, fn

Return type Dict[str, int]

Examples

```
>>> from evaluations.classification import confusion_matrix_binary
>>> confusion_matrix_binary([1, 1, 0, 0], [1, 0, 0, 1])
{'tp': 1, 'tn': 1, 'fp': 1, 'fn': 1}
```

CHAPTER

FOUR

TEXT EXTRACTION

4.1 Jaccard Word Level

4.1.1 Jaccard Word Level Similarity

`evaluations.text_extraction.jaccard_word_level.`**`jaccard_word_level_similarity`**(*y_true: str, y_pred: str*) → float

Compute word level Jaccard similarity

Parameters

- **y_true** (*str*) – True text
- **y_pred** (*str*) – Predicted text

Returns Word level Jaccard similarity

Return type float

Examples

```
>>> from evaluations.text_extraction import jaccard_word_level_similarity
>>> assert jaccard_word_level_similarity(
...         "Be happy my friend",
...         "be happy"
...     )
0.5
```

4.1.2 Jaccard Word Level Score

`evaluations.text_extraction.jaccard_word_level.`**`jaccard_word_level_score`**(*y_true: List[str], y_pred: List[str]*) → float

Compute word level Jaccard score

Parameters

- **y_true** (*List of str*) – True labels
- **y_pred** (*List of str*) – Predicted labels

Returns Word level Jaccard score

Return type float

Examples

```
>>> from evaluations.text_extraction import jaccard_word_level_score
>>> jaccard_word_level_score(
...         [
...             "Hello, how are you?",
...             "Be happy my friend",
...             "It's good.",
...         ],
...         [
...             "Hello, how are you?",
...             "be happy",
...             "Have a nice day!"
...         ]
...     )
0.5
```

INDEX

A

`accuracy_score()` (*in module evaluations.classification.accuracy*), 9

C

`concordance_index()` (*in module evaluations.medical.concordance_index*), 3

`confusion_matrix_binary()` (*in module evaluations.classification.confusion_matrix*), 9

G

`global_average_precision_score()` (*in module evaluations.kaggle_2020.global_average_precision*), 5

J

`jaccard_word_level_score()` (*in module evaluations.text_extraction.jaccard_word_level*), 11

`jaccard_word_level_similarity()` (*in module evaluations.text_extraction.jaccard_word_level*), 11

M

`micro_f1_similarity()` (*in module evaluations.kaggle_2020.row_wise_micro_averaged_f1*), 7

R

`row_wise_micro_averaged_f1_score()` (*in module evaluations.kaggle_2020.row_wise_micro_averaged_f1*), 7