# Binary classification metrics 1/7

based on blog post 22 Evaluation Metrics for Binary Classification (And When to Use Them)

## False positive rate | Type-I error

### When to use it

Usually, it is not used alone but rather with some other metric,
If the cost of dealing with an alert is high you should consider increasing the threshold to get fewer alerts.

### Explanation

How many false alerts your model raises

### Formula

$$FPR = \frac{fp}{fp + tn}$$

### How to calculate

```
from sklearn.metrics import confusion_matrix

tn, fp, fn, tp = confusion_matrix(y_true, y_pred_class).ravel()
false_positive_rate = fp / (fp + tn)
```

## False negative rate | Type-II error

### When to use it

Usually, it is not used alone but rather with some other metric,
If the cost of letting the fraudulent transactions through is high and the value you get from the users isn't you can consider focusing on this number.

### Explanation

How often your model misses trully fraudulent transactions.

### Formula

$$FNR = \frac{fn}{tp + fn}$$

### How to calculate

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_true, y_pred_class).ravel()
false_negative_rate = fn / (tp + fn)
```

## False discovery rate

### When to use it

-

### Explanation

Reversed precision (1-precision).

### Formula

$$FDR = \frac{fp}{tp + fp}$$

### How to calculate

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_true, y_pred_class).ravel()
false_discovery_rate = fp / (tp + fp)
```

## True negative rate | Specificity

### When to use it

Usually, you don't use it alone but rather as an auxiliary metric,
When you really want to be sure that you are right when you say something is safe. A typical example would be a doctor telling a patient "you are healthy". Making a mistake here and telling a sick person they are safe and can go home is something you may want to avoid.

### Explanation

Think of it as recall for negative class.

### Formula

$$TNR = \frac{tn}{tn + fp}$$

### How to calculate

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_true, y_pred_class).ravel()
true_negative_rate = tn / (tn + fp)
```

## Negative predictive value

### When to use it

Usually, you don't use it alone but rather as an auxiliary metric,
When we care about high precision on negative predictions. For example, imagine we really don't want to have any additional process for screening the transactions predicted as clean. In that case, we may want to make sure that our negative predictive value is high.

### Explanation

Think of it as precision for negative class.

### Formula

$$NPV = \frac{tn}{tn + fn}$$

### How to calculate

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_true, y_pred_class).ravel()
negative_predictive_value = tn / (tn + fn)
```

based on blog post 22 Evaluation Metrics for Binary Classification (And When to Use Them)

## Log loss

### When to use it

Usually used as objective not metric.

### Explanation

Averaged difference between ground truth and logarithm of predicted score for every observation. Heavily penalizes when the model is confident about something yet wrong.

### Formula

$$logloss = -(y_{true} * log(y_{pred}) + (1 - y_{true}) * log(1 - y_{pred}))$$

### How to calculate

```
from sklearn.metrics import log_loss

loss = log_loss(y_true, y_pred)
```

## Brier score (loss)

### When to use it

When you care about calibrated probabilities.

### Explanation

Mean squared error between ground truth and predicted score.

### Formula

$$brierloss = (y_{pred} - y_{true})^2$$

### How to calculate

```
from sklearn.metrics import brier_score_loss

brier = brier_score_loss(y_true, y_pred[:,1])
```

## ROC AUC score

### When to use it

You should use it when you ultimately care about ranking predictions.
You should not use it when your data is heavily imbalanced.
You should use it when you care equally about positive and negative class.

### Explanation

Rank correlation between predictions and target. Tells you how good at ranking predictions (positive over negative) your model is.

### Formula

Area under ROC curve

### How to calculate

```
from sklearn.metrics import roc_auc_score

roc_auc = roc_auc_score(y_true, y_pred_pos)
```

## Precison-Recall AUC | Average precision

### When to use it

When you want to communicate precision/recall decision to other stakeholders and want to choose the threshold that fits the business problem.
When your data is heavily imbalanced.
When you care more about positive than negative class.

### Explanation

What is the average precision over all recall values.

### Formula

Area under Precision-Recall curve

### How to calculate

```
from sklearn.metrics import average_precision_score

avg_precision = average_precision_score(y_true, y_pred)
```

# Binary classification metrics 3/7

based on blog post 22 Evaluation Metrics for Binary Classification (And When to Use Them)

## Kolmogorov-Smirnov statistics

### When to use it

when your problem is about sorting/ prioritizing the most relevant observations and you care equally about positive and negative class.

### Explanation

It helps to assess the separation between prediction distributions for positive and negative class.

### Formula

Max distance between KS curves

### How to calculate

```
from scikitplot.helpers import binary_ks_curve

res = binary_ks_curve(y_true, y_pred[:, 1])
ks_stat = res[3]
```

## F beta

### When to use it

When you want to combine precision and recall in one metric and would like to be able to adjust how much focus you put on one or the other.

### Explanation

Geometrically averaged precision and recall with a weight beta. ( 0<beta<1 favours precision; beta>1 favours recall )

### Formula

$$F_{beta} = (1 + \beta^2)\frac{precision * recall}{\beta^2 * precision + recall}$$

### How to calculate

```
from sklearn.metrics import fbeta_score
fbeta_score(y_true, y_pred_class, beta)
```

## F1 score

### When to use it

Pretty much in every binary classification problem.
It is my go-to metric when working on those problems.

### Explanation

Geometric average of precision and recall.

### Formula

$$F_1 = F_{beta=1}$$

### How to calculate

```
from sklearn.metrics import fbeta_score
fbeta_score(y_true, y_pred_class, beta=1)
```

## F2 score

### When to use it

When recalling positive observations (fraudulent transactions) is more important than being precise about it but you still want to have a nice and simple metric that combines precision and recall.

### Explanation

Geometric average of precison and recall with twice as much weight on recall.

### Formula

$$F_2 = F_{beta=2}$$

### How to calculate

```
from sklearn.metrics import fbeta_score
fbeta_score(y_true, y_pred_class, beta=2)
```

## Cohen Kappa

### When to use it

When you want to combine precision and recall in one metric and would like to be able to adjust how much focus you put on one or the other.

### Explanation

How much better is your model over the random classifier that predicts based on class frequencies.

### Formula

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

### How to calculate

```
from sklearn.metrics import cohen_kappa_score
cohen_kappa_score(y_true, y_pred_class)
```

## Matthews correlation coefficient

### When to use it

Pretty much in every binary classification problem.
It is my go-to metric when working on those problems.

### Explanation

Correlation between predicted classes and ground truth.

### Formula

$$MCC = \frac{tp * tn - fp * fn}{(tp + fp)(tp + fn)(tn + fp)(tn + fn))}$$

### How to calculate

```
from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_true, y_pred_class)
```

## True positive rate | Recall | Sensitivity

### When to use it

Usually, you will not use it alone but rather coupled with other metrics like precision.
That being said recall is a go-to metric, when you really care about catching all fraudulent transactions even at a cost of false alerts.
Potentially it is cheap for you to process those alerts and very expensive when the transaction goes unseen.

### Explanation

Put all guilty in prison.

### Formula

$$TPR = \frac{tp}{tp + fn}$$

### How to calculate

```
from sklearn.metrics import recall_score
recall_score(y_true, y_pred_class)
```

## Positive predictive value | Precision

### When to use it

Again, it usually doesn't make sense to use it alone but rather coupled with other metrics like recall.
When raising false alerts is costly and you really want all the positive predictions to be worth looking at you should optimize for precision.

### Explanation

Make sure that people that go to prison are guilty.

### Formula

$$PPV = \frac{tp}{tp + fp}$$

### How to calculate

```
from sklearn.metrics import precision_score
precision_score(y_true, y_pred_class)
```

## Accuracy

### When to use it

When your problem is balanced using accuracy is usually a good start.
When every class is equally important to you.

### Explanation

How good at classyfying both positive and negative cases your model is.

### Formula

$$ACC = \frac{(tp + tn)}{(tp + fp + fn + tn)}$$

### How to calculate

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred_class)
```

## Confusion Matrix

### When to use it

Pretty much always.
Get the gist of model imbalance and where the predictions fall.

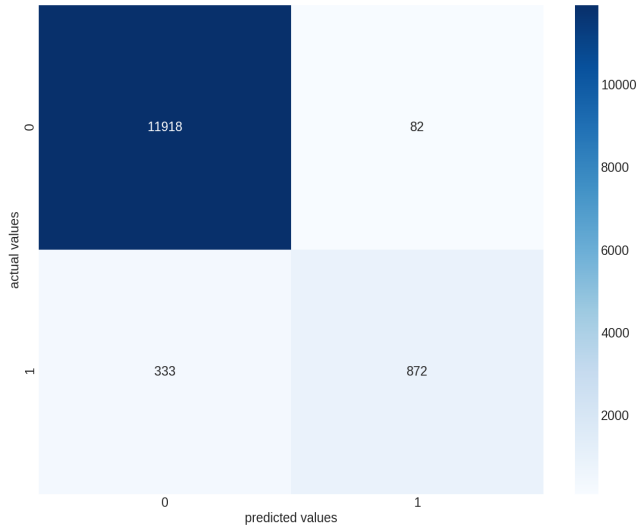
### Explanation

Table that contains true negative (tn), false positive (fp), false negative (fn), and true positive (tp) predictions.

### How to calculate

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_true, y_pred_class)
tn, fp, fn, tp = cm.ravel()
```




## Cumulative gain chart

### When to use it

Whenever you want to use your model to choose the best customers/transactions to target by sorting all predictions you should consider using cumulative gain charts.

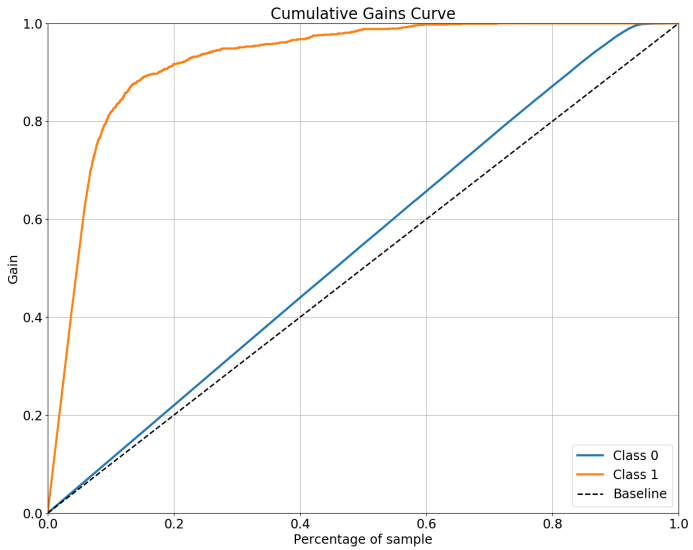
### Explanation

In simple words, it helps you gauge how much you gain by using your model over a random model for a given fraction of top scored predictions.

### How to calculate

```
from scikitplot.metrics import plot_cumulative_gain

fig, ax = plt.subplots()
plot_cumulative_gain(y_true, y_pred, ax=ax)
```



## Lift curve

### When to use it

Whenever you want to use your model to choose the best customers/transactions to target by sorting all predictions you should consider using a lift curve.

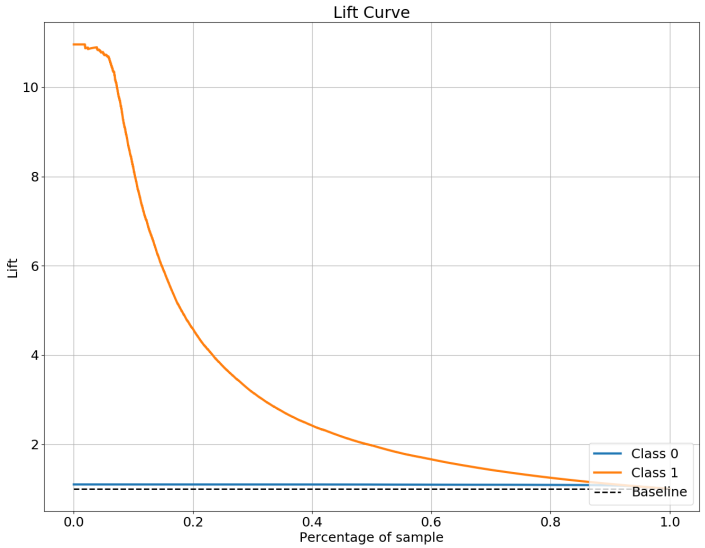
### Explanation

In simple words, it helps you gauge how much you gain by using your model over a random model for a given fraction of top scored predictions.
It tells you how much better your model is than a random model for the given percentile of top scored predictions.

### How to calculate

```
from scikitplot.metrics import plot_lift_curve

fig, ax = plt.subplots()
plot_lift_curve(y_true, y_pred, ax=ax)
```




## Kolmogorov-Smirnov chart

### When to use it

When your problem is about sorting/prioritizing the most relevant observations and you care equally about positive and negative class.

### Explanation

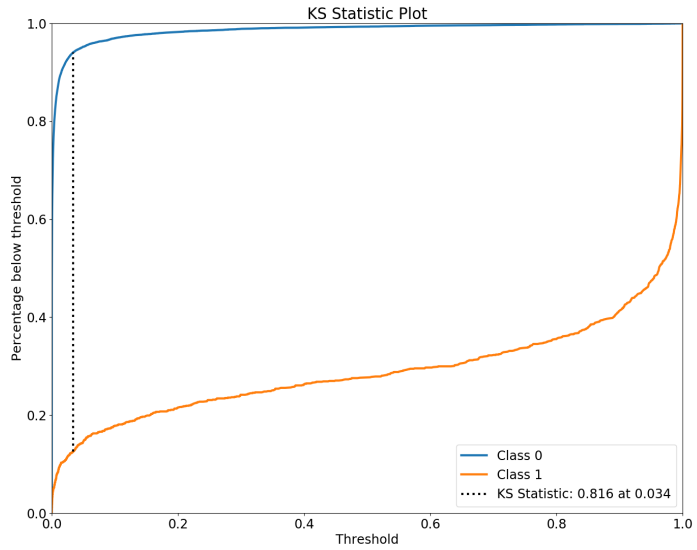
It helps to assess the separation between prediction distributions for positive and negative class.

So it works similarly to Cumulative gain chart but instead of just looking at positive class it looks at the separation between positive and negative class.

### How to calculate

```
from scikitplot.metrics import plot_ks_statistic

fig, ax = plt.subplots()
plot_ks_statistic(y_true, y_pred, ax=ax)
```



## ROC curve

### When to use it

You should use it when you ultimately care about ranking predictions.
You should not use it when your data is heavily imbalanced.
You should use it when you care equally about positive and negative class.

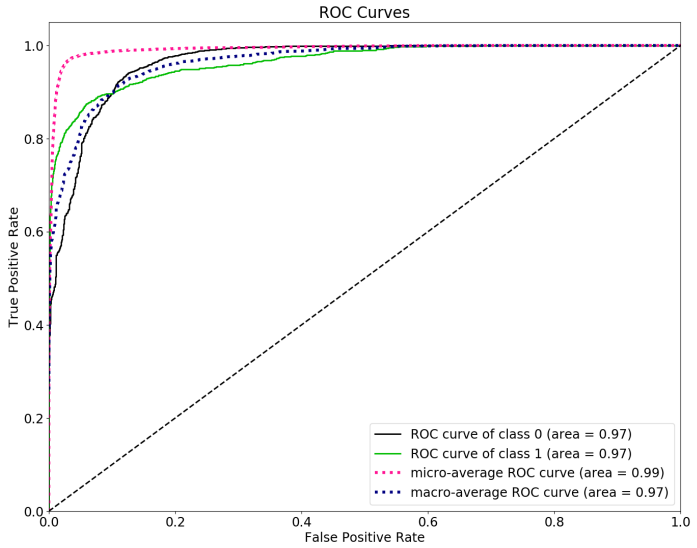
### Explanation

It is a chart that visualizes the tradeoff between true positive rate (TPR) and false positive rate (FPR). Basically, for every threshold, we calculate TPR and FPR and plot it on one chart.

### How to calculate

```
from scikitplot.metrics import plot_roc

fig, ax = plt.subplots()
plot_roc(y_true, y_pred, ax=ax)
```




## Precision-Recall curve

### When to use it

It is a curve that combines precision (PPV) and Recall (TPR) in a single visualization. For every threshold, you calculate PPV and TPR and plot it. The higher on y-axis your curve is the better your model performance.

### Explanation

When you want to communicate precision/recall decision to other stakeholders and want to choose the threshold that fits the business problem.
When your data is heavily imbalanced.
When you care more about positive than negative class.

### How to calculate

```
from scikitplot.metrics import plot_precision_recall

fig, ax = plt.subplots()
plot_precision_recall(y_true, y_pred, ax=ax)
```