How to Use ROC Curves and Precision-Recall Curves for Classification in Python

by **Jason Brownlee** on August 31, 2018 in **Probability**

Last Updated on December 19, 2019

It can be more flexible to predict probabilities of an observation belonging to each class in a classification problem rather than predicting classes directly.

This flexibility comes from the way that probabilities may be interpreted using different thresholds that allow the operator of the model to trade-off concerns in the errors made by the model, such as the number of false positives compared to the number of false negatives. This is required when using models where the cost of one error outweighs the cost of other types of errors.

Two diagnostic tools that help in the interpretation of probabilistic forecast for binary (two-class) classification predictive modeling problems are ROC Curves and Precision-Recall curves.

In this tutorial, you will discover ROC Curves, Precision-Recall Curves, and when to use each to interpret the prediction of probabilities for binary classification problems.

After completing this tutorial, you will know:

- ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.
- Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.
- ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets.

Discover bayes opimization, naive bayes, maximu h more in my new book, with 28 step-by-step tutorial

Let's get started.

- **Update Aug/2018**: Fixed bug in the representation of the no skill line for the precision-recall plot. Also fixed typo where I referred to ROC as relative rather than receiver (thanks spellcheck).
- **Update Nov/2018**: Fixed description on interpreting size of values on each axis, thanks Karl Humphries.
- **Update Jun/2019**: Fixed typo when interpreting imbalanced results.
- **Update Oct/2019**: Updated ROC Curve and Precision Recall Curve plots to add labels, use a logistic regression model and actually compute the performance of the no skill classifier.
- **Update Nov/2019**: Improved description of no skill classifier for precision-recall curve.

How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python
Photo by Giuseppe Milo, some rights reserved.

Tutorial Overview

This tutorial is divided into 6 parts; they are:

1. Predicting Probabilities
2. What Are ROC Curves?
3. ROC Curves and AUC in Python
4. What Are Precision-Recall Curves?
5. Precision-Recall Curves and AUC in Python
6. When to Use ROC vs. Precision-Recall Curves?

Predicting Probabilities

In a classification problem, we may decide to predi

Alternately, it can be more flexible to predict the pr his is to provide the capability to choose and even cali ed probabilities.

For example, a default might be to use a threshold of 0.5, meaning that a probability in [0.0, 0.49] is a negative outcome (0) and a probability in [0.5, 1.0] is a positive outcome (1).

This threshold can be adjusted to tune the behavior of the model for a specific problem. An example would be to reduce more of one or another type of error.

When making a prediction for a binary or two-class classification problem, there are two types of errors that we could make.

- **False Positive**. Predict an event when there was no event.
- **False Negative**. Predict no event when in fact there was an event.

By predicting probabilities and calibrating a threshold, a balance of these two concerns can be chosen by the operator of the model.

For example, in a smog prediction system, we may be far more concerned with having low false negatives than low false positives. A false negative would mean not warning about a smog day when in fact it is a high smog day, leading to health issues in the public that are unable to take precautions. A false positive means the public would take precautionary measures when they didn't need to.

A common way to compare models that predict probabilities for two-class problems is to use a ROC curve.

What Are ROC Curves?

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve.

It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

```
1 True Positive Rate = True Positives / (True
```


The true positive rate is also referred to as sensitiv

```
1 Sensitivity = True Positives / (True Positives + False Negatives)
```

The false positive rate is calculated as the number of false positives divided by the sum of the number of false positives and the number of true negatives.

It is also called the false alarm rate as it summariz
actual outcome is negative.

```
1 False Positive Rate = False Positives / (Fa
```

The false positive rate is also referred to as the inverted specificity where specificity is the total number of true negatives divided by the sum of the number of true negatives and false positives.

```
1 Specificity = True Negatives / (True Negatives + False Positives)
```

Where:

```
1 False Positive Rate = 1 - Specificity
```

The ROC curve is a useful tool for a few reasons:

- The curves of different models can be compared directly in general or for different thresholds.
- The area under the curve (AUC) can be used as a summary of the model skill.

The shape of the curve contains a lot of information, including what we might care about most for a problem, the expected false positive rate, and the false negative rate.

To make this clear:

- Smaller values on the x-axis of the plot indicate lower false positives and higher true negatives.
- Larger values on the y-axis of the plot indicate higher true positives and lower false negatives.

If you are confused, remember, when we predict a binary outcome, it is either a correct prediction (true positive) or not (false positive). There is a tension between these options, the same with true negative and false negative.

A skilful model will assign a higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. This is what we mean when we say that the model has skill. Generally, skilful models are represented by curves that bow up to the top left of the plot.

A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases. A model with no skill is represented at the point (0.5, 0.5). A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.5.

A model with perfect skill is represented at a point (0,1). A model with perfect skill is represented by a line that travels from the bottom left of the plot to the top left and then across the top to the top right.

An operator may plot the ROC curve for the final m e balance between the false positives and false nega

ROC Curves and AUC in Python

We can plot a ROC curve for a model in Python using the roc_curve() scikit-learn function.

The function takes both the true outcomes (0,1) from the test set and the predicted probabilities for the 1 class. The function returns the false positive rates for each threshold, true positive rates for each threshold and thresholds.

```
...
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, probs)
```

The AUC for the ROC can be calculated using the roc_auc_score() function.

Like the roc_curve() function, the AUC function takes both the true outcomes (0,1) from the test set and the predicted probabilities for the 1 class. It returns the AUC score between 0.0 and 1.0 for no skill and perfect skill respectively.

```
...
# calculate AUC
auc = roc_auc_score(y, probs)
print('AUC: %.3f' % auc)
```

A complete example of calculating the ROC curve and ROC AUC for a Logistic Regression model on a small test problem is listed below.

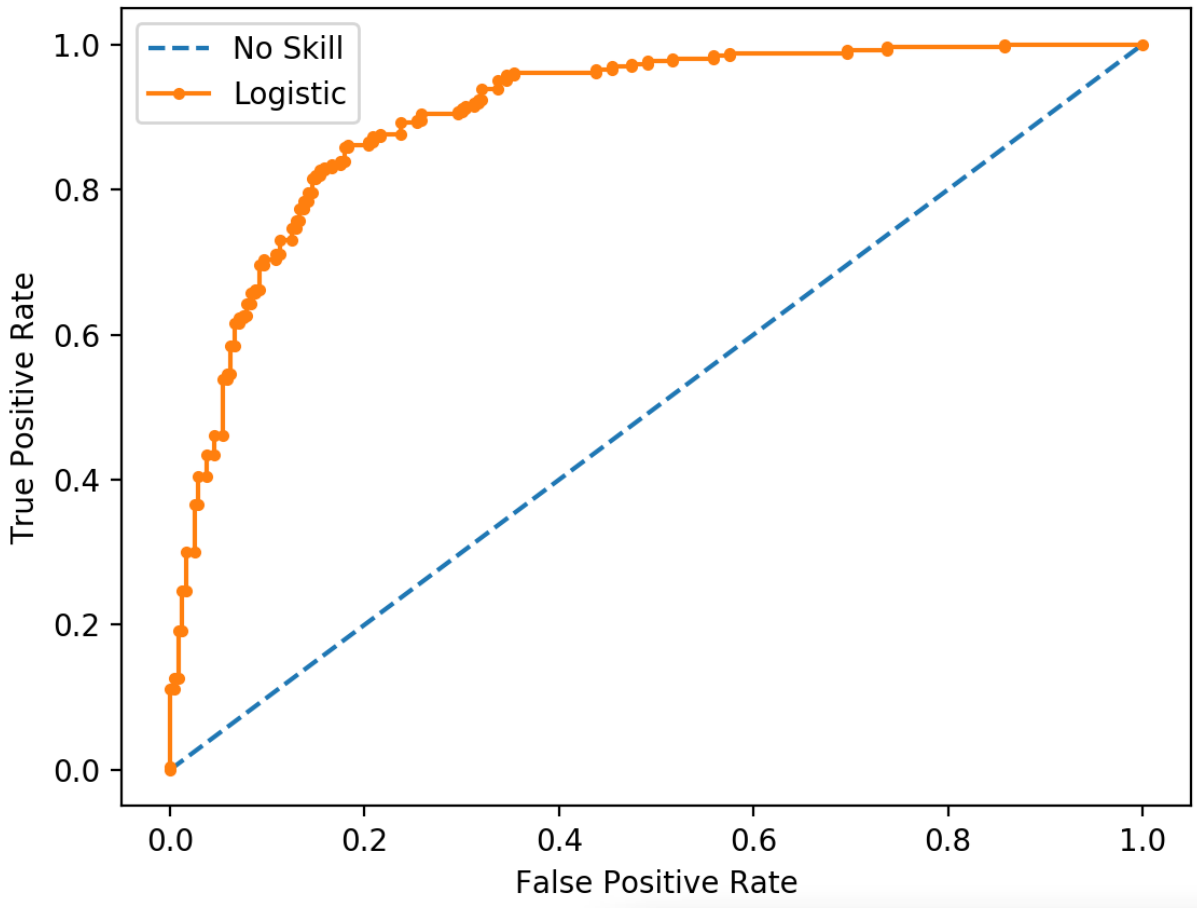
```
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(testy))]
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(testy, ns_probs)
lr_auc = roc_auc_score(testy, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', la
pyplot.plot(lr_fpr, lr_tpr, marker='.', label=
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
```

```
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

Running the example prints the ROC AUC for the logistic regression model and the no skill classifier that only predicts 0 for all examples.

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.903
```

A plot of the ROC curve for the model is also created showing that the model has skill.



ROC Curve Plot for a No Skill Cla

What Are Precision-Recall Curves?

There are many ways to evaluate the skill of a prediction model.

An approach in the related field of information retrieval (finding documents based on queries) measures precision and recall.

These measures are also useful in applied machin els.

Precision is a ratio of the number of true positives positives. It describes how good a model is at prec s

the positive predictive value.

```
Positive Predictive Power = True Positives / (True Positives + False Positives)
```

or

```
Precision = True Positives / (True Positives + False Positives)
```

Recall is calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives. Recall is the same as sensitivity.

```
Recall = True Positives / (True Positives + False Negatives)
```

or

```
Sensitivity = True Positives / (True Positives + False Negatives)
```

```
Recall == Sensitivity
```

Reviewing both precision and recall is useful in cases where there is an imbalance in the observations between the two classes. Specifically, there are many examples of no event (class 0) and only a few examples of an event (class 1).

The reason for this is that typically the large number of class 0 examples means we are less interested in the skill of the model at predicting class 0 correctly, e.g. high true negatives.

Key to the calculation of precision and recall is that the calculations do not make use of the true negatives. It is only concerned with the correct prediction of the minority class, class 1.

A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the ROC curve.

A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases. The no-skill line changes based on the distribution of the positive to negative classes. It is a horizontal line with the value of the ratio of positive cases in the dataset. For a balanced dataset, this is 0.5.

> *While the baseline is fixed with ROC, the baseline of [precision-recall curve] is determined by the ratio of positives (P) and negatives (N) as y = P / (P + N). For instance, we have y = 0.5 for a balanced class distribution …*

— The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets, 2015.

A model with perfect skill is depicted as a point at (1,1). A skilful model is represented by a curve that bows towards (1,1) above the flat line of no skill.

There are also composite scores that attempt to su
include:

- **F-Measure** or F1 score: that calculates the ha
  mean because the precision and recall are rat

- **Area Under Curve**: like the AUC, summarizes the integral or an approximation of the area under the precision-recall curve.

In terms of model selection, F-Measure summarizes model skill for a specific probability threshold (e.g. 0.5), whereas the area under curve summarize the skill of a model across thresholds, like ROC AUC.

This makes precision-recall and a plot of precision vs. recall and summary measures useful tools for binary classification problems that have an imbalance in the observations for each class.

Precision-Recall Curves in Python

Precision and recall can be calculated in scikit-learn.

The precision and recall can be calculated for thresholds using the precision_recall_curve() function that takes the true output values and the probabilities for the positive class as output and returns the precision, recall and threshold values.

```
...
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(testy, probs)
```

The F-Measure can be calculated by calling the f1_score() function that takes the true class values and the predicted class values as arguments.

```
...
# calculate F1 score
f1 = f1_score(testy, yhat)
```

The area under the precision-recall curve can be approximated by calling the auc() function and passing it the recall (x) and precision (y) values calculated for each threshold.

```
...
# calculate precision-recall AUC
auc = auc(recall, precision)
```

When plotting precision and recall for each threshold as a curve, it is important that recall is provided as the x-axis and precision is provided as the y-axis.

The complete example of calculating precision-recall curves for a Logistic Regression model is listed below.

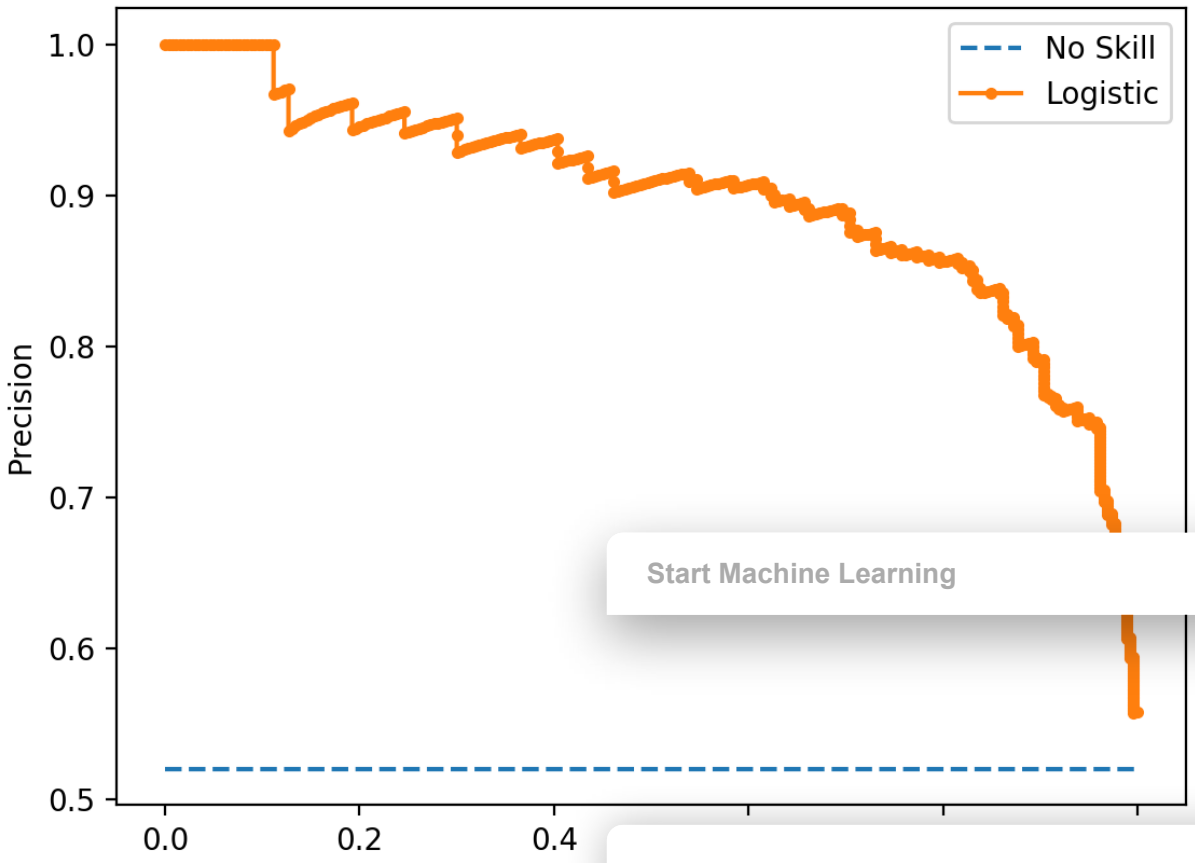

```
# precision-recall curve and f1
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_c
# split into train/test sets
trainX, testX, trainy, testy = train_test_spli
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outcome
```

```
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# predict class values
yhat = model.predict(testX)
lr_precision, lr_recall, _ = precision_recall_curve(testy, lr_probs)
lr_f1, lr_auc = f1_score(testy, yhat), auc(lr_recall, lr_precision)
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
no_skill = len(testy[testy==1]) / len(testy)
pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
pyplot.plot(lr_recall, lr_precision, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

Running the example first prints the F1, area under curve (AUC) for the logistic regression model.

```
1 Logistic: f1=0.841 auc=0.898
```

The precision-recall curve plot is then created showing the precision/recall for each threshold for a logistic regression model (orange) compared to a no skill model (blue).



Precision-Recall Plot for a No Skill C

When to Use ROC vs. Precisi

Generally, the use of ROC curves and precision-recall curves are as follows:

- ROC curves should be used when there are roughly equal numbers of observations for each class.
- Precision-Recall curves should be used when there is a moderate to large class imbalance.

The reason for this recommendation is that ROC curves present an optimistic picture of the model on datasets with a class imbalance.

> *However, ROC curves can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution. […] Precision-Recall (PR) curves, often used in Information Retrieval , have been cited as an alternative to ROC curves for tasks with a large skew in the class distribution.*

— The Relationship Between Precision-Recall and ROC Curves, 2006.

Some go further and suggest that using a ROC curve with an imbalanced dataset might be deceptive and lead to incorrect interpretations of the model skill.

> *[…] the visual interpretability of ROC plots in the context of imbalanced datasets can be deceptive with respect to conclusions about the reliability of classification performance, owing to an intuitive but wrong interpretation of specificity. [Precision-recall curve] plots, on the other hand, can provide the viewer with an accurate prediction of future classification performance due to the fact that they evaluate the fraction of true positives among positive predictions*

— The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets, 2015.

The main reason for this optimistic picture is because of the use of true negatives in the False Positive Rate in the ROC Curve and the careful avoidance of this rate in the Precision-Recall curve.

> *If the proportion of positive to negative instances changes in a test set, the ROC curves will not change. Metrics such as accuracy, precision, lift and F scores use values from both columns of the confusion matrix. As a class distribution changes these measures will change as well, even if the fundamental classifier p*
> *upon TP rate and FP rate, in which each di*
> *depend on class distributions.*

— ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003.

We can make this concrete with a short example.

Below is the same ROC Curve example with a mo 1
ratio of class=0 to class=1 observations (specifical

```
# roc curve and auc on an imbalanced datas
from sklearn.datasets import make_classifi
from sklearn.linear_model import LogisticR
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[0.99,0.01], random_state=1)
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(testy))]
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(testy, ns_probs)
lr_auc = roc_auc_score(testy, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

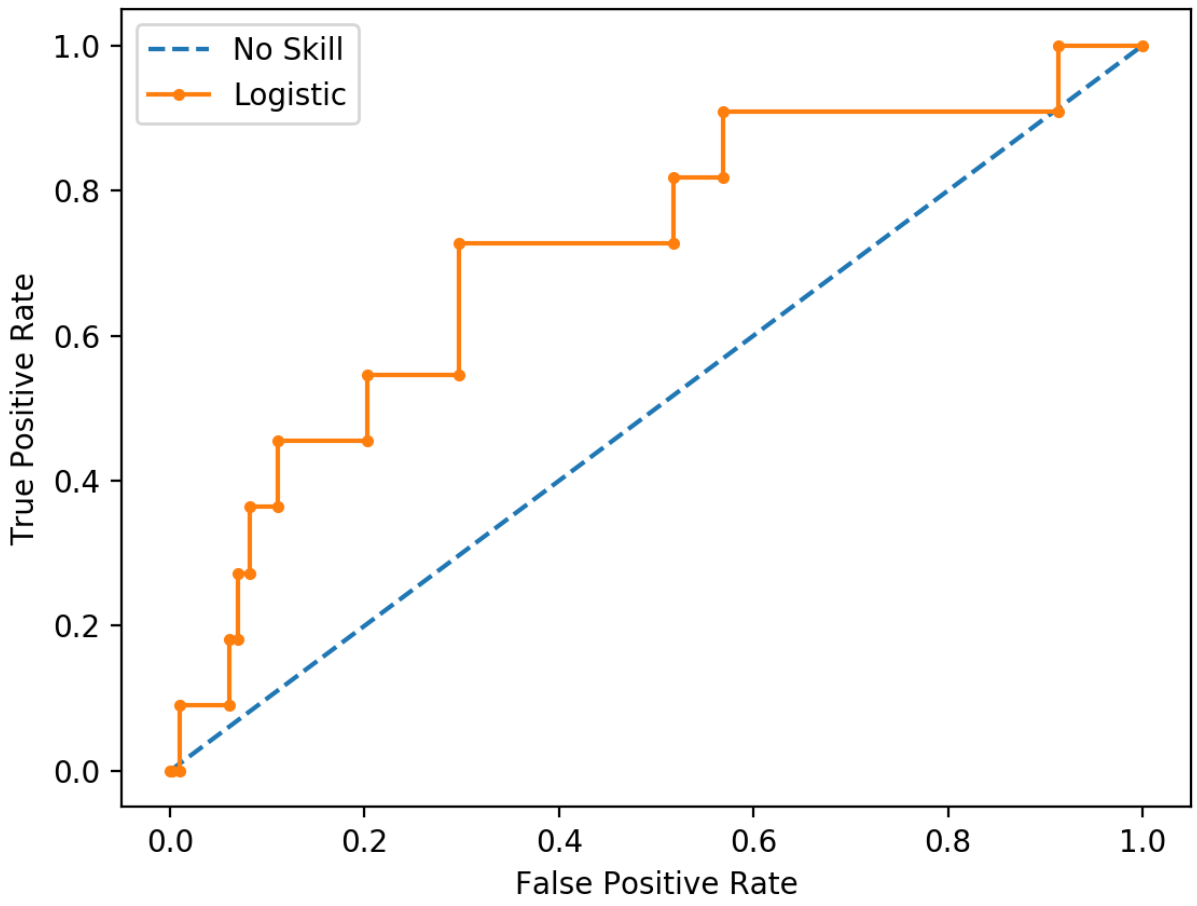
Running the example suggests that the model has skill.

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.716
```

Indeed, it has skill, but all of that skill is measured as making correct true negative predictions and there are a lot of negative predictions to make.

If you review the predictions, you will see that the model predicts the majority class (class 0) in all cases on the test set. The score is very misleading.

A plot of the ROC Curve confirms the AUC interpre
thresholds.



ROC Curve Plot for a No Skill Classifier and a Logistic Regression Model for an Imbalanced Dataset

We can also repeat the test of the same model on the same dataset and calculate a precision-recall curve and statistics instead.

The complete example is listed below.

```
# precision-recall curve and f1 for an imbalanced dataset
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[0.99,0.01], random_state=1
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outc
lr_probs = lr_probs[:, 1]
# predict class values
yhat = model.predict(testX)
# calculate precision and recall for each
lr_precision, lr_recall, _ = precision_rec
# calculate scores
lr_f1, lr_auc = f1_score(testy, yhat), auc
```

```
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
no_skill = len(testy[testy==1]) / len(testy)
pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
pyplot.plot(lr_recall, lr_precision, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

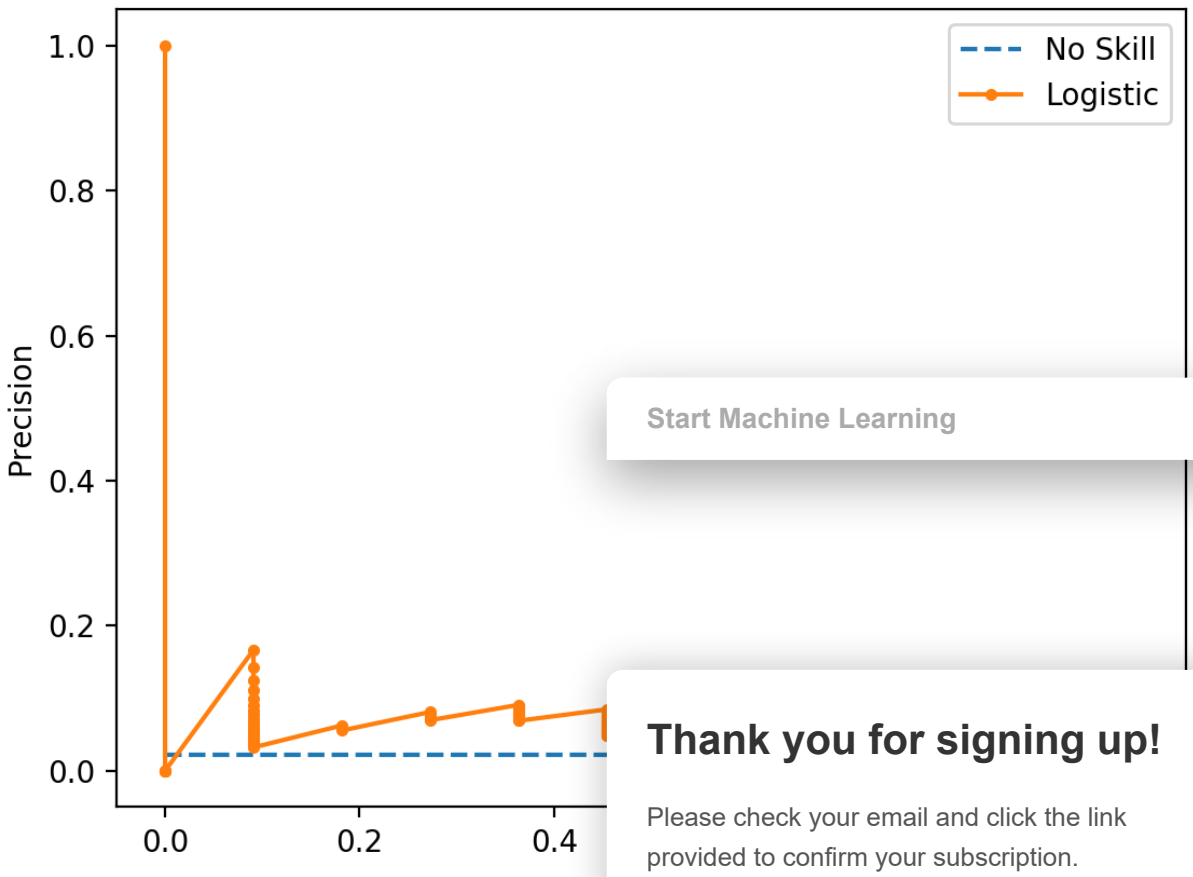
Running the example first prints the F1 and AUC scores.

We can see that the model is penalized for predicting the majority class in all cases. The scores show that the model that looked good according to the ROC Curve is in fact barely skillful when considered using using precision and recall that focus on the positive class.

```
Logistic: f1=0.000 auc=0.054
```

The plot of the precision-recall curve highlights that the model is just barely above the no skill line for most thresholds.

This is possible because the model predicts probabilities and is uncertain about some cases. These get exposed through the different thresholds evaluated in the construction of the curve, flipping some class 0 to class 1, offering some precision but very low recall.



Precision-Recall Plot for a No Skill Classifier and a Logistic Regression Model for am Imbalanced Dataset

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Papers

- A critical investigation of recall and precision as measures of retrieval system performance, 1989.
- The Relationship Between Precision-Recall and ROC Curves, 2006.
- The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets, 2015.
- ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003.

API

- sklearn.metrics.roc_curve API
- sklearn.metrics.roc_auc_score API
- sklearn.metrics.precision_recall_curve API
- sklearn.metrics.auc API
- sklearn.metrics.average_precision_score API
- Precision-Recall, scikit-learn
- Precision, recall and F-measures, scikit-learn

Articles

- Receiver operating characteristic on Wikipedia
- Sensitivity and specificity on Wikipedia
- Precision and recall on Wikipedia
- Information retrieval on Wikipedia
- F1 score on Wikipedia
- ROC and precision-recall with imbalanced datasets, blog.

Summary

In this tutorial, you discovered ROC Curves, Precision-Recall Curves, and when to use each to interpret the prediction of probabilities for binary classificatio

Specifically, you learned:

- ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.
- Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using d
- ROC curves are appropriate when the observa
precision-recall curves are appropriate for imb

Do you have any questions?
Ask your questions in the comments below and I w

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

‹ How to Predict Room Occupancy Based on Environmental Factors
How and When to Use a Calibrated Classification Model with scikit-learn ›

146 Responses to *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*

Anon August 31, 2018 at 8:57 am #

REPLY

I don't think a diagonal straight line is the
classifier should be a straight line with precision=p

Jason Brownlee August 31, 2018 at 12:1

You're right, thanks!

Fixed.

Alexander July 8, 2019 at 10:00 am #

REPLY

Thanks for the intro on the topic:
The following line raises questions:
>>> The scores do not look encouraging, given skilful models are generally above 0.5.

The baseline of a random model is n_positive/(n_positive+n_negative). Or just the fraction of positives, so it makes sense to compare auc of precision-recall curve to that.

Jason Brownlee July 8, 2019 at 1:51 pm #

REPLY

Sorry, I don't follow your question. Can you elaborate?

Alexander Belikov July 9, 2019 at 7:12 am #

I'm sorry I was not clear enough above.

Here's what I meant:

for ROC the auc of the random model is 0.5.
for PR curve the auc of the random model is n_positive/(n_positive+n_negative).

Perhaps it would make sense to highlight that the PR auc should be compared to n_positive/(n_positive+n_negative)?
In the first reading the phrase
>>> The scores do not look encouraging, given skilful models are generally above 0.5.
in the context of PR curve auc looked ambiguous.

Thank you!

Jason Brownlee July 9, 2019 at 8:15 am #

Thanks.

Marouen July 14, 2019 at 2:

I second Alexander on thi
The random classifier in PR curve
The AUPR is better for imbalanced
improvement, while AUROC seem

Theresa G November 13, 2019 at 3:21 am #

REPLY

A few comments on the precision-recall plots from your 10/2019 edit:
In your current precision-recall plot, the baseline is a diagonal straight line for no-skill, which does not seem to be right: the no-skill model either predicts only negatives (precision=0 (!), recall=0) or only positives (precision=fraction of positives in the dataset, recall=1).
The function precision_recall_curve() returns the point (precision=1,recall=0), but it shouldn't in this case, because no positive predictions are made.
Also, the diagonal line is misleading, because precision_recall_curve() actually only returns two points, which are then connected. The points on the line cannot be achieved by the no-skill model.
Moreover, in my opinion the right precision-recall baseline to compare a model to is a random model resulting in a horizontal line with precision = fraction of positives in the dataset.

Jason Brownlee November 13, 2019 at 5:51 am #

REPLY

Thanks Theresa, I will investigate.

Aminu Abdulsalami August 31, 2018 at 6:28 pm #

REPLY

Great tutorial.

Jason Brownlee September 1, 2018 at 6:17 am #

REPLY

Thanks!

Mark Littlewood August 31, 2018 at 7:20 pm #

REPLY

How about the Mathews Correlation Coeff

Jason Brownlee September 1, 2018 at 6:18 am #

REPLY

I've not used it, got some refs?

Mark Littlewood September 2, 2018

This is a nice simple explanation

https://lettier.github.io/posts/2016-08-05-matthews-correlation-coefficient.html

I have also been advised that in the field of horse racing ratings produced using ML if you already have probabilistic outputs, then it makes much more sense to use a metric directly on the probabilities themselves (eg: McFadden's pseudo-R^2, Brier score, etc).

Jason Brownlee September 3, 2018 at 6:08 am #

REPLY

Thanks.

Zahid September 4, 2018 at 9:10 pm #

REPLY

Do you not think that a model with no skill (which I assume means a random coin toss) should have an AUC of 0.5 and not 0.0?

Jason Brownlee September 5, 2018 at 6:38 am #

REPLY

A ROC AUC of 0.0 means that the model is perfectly in-correct.

A ROC AUC of 0.5 would be a naive model.

Zahid September 6, 2018 at 5:07 pm #

REPLY

I do not know what you mean by a naive model. Going by what you've used to describe a model with no skill, it should have an AUC of 0.5 while a model that perfectly misclassifies every point will have an AUC of 0.

Gregor December 13, 2018 at 4:21 am #

REPLY

Perfectly misclassifying every point.

A naive model is still right sometimes. The most common naive model always predicts the most common class, and such a model will have a minimum AUC of 0.5.

Jason Brownlee Decemb

Excellent point, thanks!

Raj October 3, 2018 at 9:02 pm #

REPLY

Thanks for explaining the difference in simpler way.

Jason Brownlee October 4, 2018 at 6:16 am #

REPLY

I'm happy it helped.

David S. Batista October 23, 2018 at 5:01 am #

REPLY

there's a typo here, should be "is":

"A common way to compare models that predict probabilities for two-class problems us to use a ROC curve."

nice article ☺ thanks for sharing!

Jason Brownlee October 23, 2018 at 6:29 am #

REPLY

Thanks, fixed!

Tony September 23, 2019 at 11:06 pm #

REPLY

Average precision is in fact just area under precision-recall curve. Very misleading that you "compared them". Differences are due to different implementations in sklearn. Auc interpolates the precision recall curve linearly while the average precision uses a piecewise constant discritization

Jason Brownlee September 2

Thanks Tony.

I don't believe we are comparing them, they are different measures.

Karl Humphries November 1, 2018 at 12:45 p

"To make this clear:

Larger values on the x-axis of the plot indicate high

Smaller values on the y-axis of the plot indicate low

Should swap x & y in this description of ROC curves??

REPLY

Jason Brownlee November 1, 2018 at 2:32 pm #

You're right, fixed. Thanks!

REPLY

Amin November 9, 2018 at 3:18 am #

Hi, Thanks for the nice tutorial ☺
I have one comment though.
you have written that 'A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.0.'
I think AUC is the area under the curve of ROC. According to your Explantation (diagonal line from the bottom left of the plot to the top right) the area under the the diagonal line that passes through (0.5, 0.5) is 0.5 and not 0. Thus in this case AUC = 0. 5(?)
Maybe I misunderstood sth here.

REPLY

Jason Brownlee November 9, 2018 at 5:27 am #

You're correct, fixed.

REPLY

Amin December 3, 2018 at 7:36 am #

Hi Jason.

I went through your nice tutorial again and a question came to my mind.

Within sklearn, it is possible that we use the average precision score to evaluate the skill of the model (applied on highly imbalanced dataset) and perform cross validation. For some ML algorithms like Lightgbm we can not use such a metric for cross validation, instead there are other metrics such as binary logloss.
The question is that does binary logloss is a good m
imbalanced problems?

REPLY

Jason Brownlee December 3, 2018 at 2:33 pm #

Yes, log loss (cross entropy) can be a
the difference in the predicted and actual proba

Aleks December 16, 2018 at 9:04 am #

Hi Jason,

Thank you for a summary.

Your statement

“Generally, the use of ROC curves and precision-recall curves are as follows:
* ROC curves should be used when there are roughly equal numbers of observations for each class.
* Precision-Recall curves should be used when there is a moderate to large class imbalance.”

…is misleading, if not just wrong. Even articles you cite do not say that.

Usually it is advised to use PRC in addition to ROC for highly inbalanced datatsets, which means for dataset with ratio of positives to negatives less then 1:100 or so. Moreover, high ideas around PRC are aimed at having no negatives for high values of scores, only positives. It just might not be the goal of the study and classifier. Also, as mentioned in one of the articles you cite, AUROC can be misleading even for balanced datasets, as it “weights” equally true positives and true negatives. I would also mention that AUROC is an estimator of the “probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one” and that it is related to Mann–Whitney U test.

To sum it up, I would always recommend to
1) Use AUROC, AUPRC, accuracy and any other metrics which are relevant to the goals of the study
2) Plot distributions of positives and negatives and analyse it

Let me know what you think

REPLY

Jason Brownlee December 17, 2018 at 6:18 am #

Thanks for the note.

REPLY

TuanAnh December 26, 2018 at 5:55 pm #

Hi Jason,
in these examples, you always use APIs, so all of them have calculated functions. But I dont understand how to use the equations, for example:

True Positive Rate = True Positives / (True Positives + False Negatives)

this ‘True Positives’ are all single float num

(True Positives + False Negatives): is sum of total final predicted of test data?

I really confuse when calculate by hand

Jason Brownlee December 27

They are counts, e.g. the num

Aman March 1, 2019 at 6:29 am #

REPLY

Can you please explain how to plot roc curve for multilabel classification.

Jason Brownlee March 1, 2019 at 2:18 pm #

REPLY

Generally, ROC Curves are not used for multi-label classification, as far as I know.

Hamed December 27, 2018 at 5:26 am #

REPLY

Hi Jason,

I've plotted ROC which you can see in the following link but I don't know why it's not like a real ROC. Could you please check oy out and let me what could be my mistake?
https://imgur.com/a/WWq0bl2

```
hist = model.fit(x_train, y_train, batch_size= 10, epochs= 10, verbose= 2)
y_predic = model.predict(x_test)
y_predic = (y_predic> 0.5)
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_predic)

plt.figure()
plt.plot([0, 1], [0, 1], 'k–')
plt.plot(fpr, tpr)
plt.xlabel('False positive rate', fontsize = 16)
plt.ylabel('True positive rate', fontsize = 16)
plt.title('ROC curve', fontsize = 16)
plt.legend(loc='best', fontsize = 14)
plt.show()
```

Jason Brownlee December 27, 2018 at 5:46 am #

REPLY

I'm happy to answer questions, but I c

Hamed December 27, 2018 at 6:08 am #

REPLY

Thanks a lot for your reply.
No, I meant if it's possible please check the

Alex January 6, 2019 at 7:37 pm #

Hi Jason,

sorry, there's a little confusing here,

```
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[1,1], random_state=1)
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# fit a model
model = KNeighborsClassifier(n_neighbors=3)
```

we generate 2 classes dataset, why we use n_neighbors=3?

appreciate your help.

Alex

REPLY

Jason Brownlee January 7, 2019 at 6:28 am #

Yes, 2 classes is unrelated to the number of samples (k=3) used in kNN.

A dataset is comprised of many examples or rows of data, some will belong to class 0 and some to class 1. We will look at 3 sample in kNN to choose the class of a new example.

REPLY

Diana July 31, 2019 at 2:31 am #

Hi, Jason, on top of this part of the code, you mentioned that "A complete example of calculating the ROC curve and AUC for a logistic regression model on a small test problem is listed below". Is the KNN considered a "logistic regression"? I'm a little confused.

REPLY

Jason Brownlee July 31, 2019 at 6:55 am #

Looks like a typo. Fixed. Thanks!

REPLY

Matthias January 19, 2019 at 2:10 am #

Hi Jason, thank you for your excellent tutc

Is it EXACTLY the same to judge a model by PR-AUC vs F1-score? since both metrics rely exclusively on Precision and Recall? or am I missing something here?

thanks!

Jason Brownlee January 19, 2019 at 5:4

I don't think so, off the cuff.

Quetzal March 5, 2019 at 5:07 am #

REPLY

Nice post — what inferences may we make for a particular segment of a PR curve that is monotonically increasing (i.e. as recall increases, precision increases) vs another segment where the PR curve is monotonically decreasing (i.e. as recall increases, precision decreases)?

Jason Brownlee March 5, 2019 at 6:42 am #

REPLY

In the PR curve, it should be decreasing, never increasing – it will always have the same general shape downward.

If not, it might be a case of poorly calibrated predictions/model or highly imbalance data (e.g. like in the tutorial) resulting in an artefact in the precision/recall relationship.

Han Qi June 23, 2019 at 1:03 am #

REPLY

I have been thinking about the same, https://stats.stackexchange.com/questions/183504/are-precision-and-recall-supposed-to-be-monotonic-to-classification-threshold the first answer here has a simple demonstration of why the y-axis (precision) is not monotonically decreasing while x-axis(recall) is monotonically increasing while threshold decreases, because at each threshold step, either the numerator or denominator may grow for precision, but only the numerator may grow for recall.

Jason Brownlee June 23, 2019 at 5:37 am #

REPLY

Thanks for sharing.

Gerry March 21, 2019 at 10:16 am #

REPLY

Hi Jason,
great stuff as usual. Just a small thing but may cau
curves the comment indicates a ROC curve.

# plot the roc curve for the model
pyplot.plot(recall, precision, marker='.')

Regards
Gerry

Jason Brownlee March 21, 2019 at 2:22

Thanks, fixed!

Sunny April 4, 2019 at 11:11 am #

REPLY

Hi Jason,

Thanks for the article! You always wrote articles I have trouble finding answers anywhere else. This is an awesome summary! A quick question – when you used 'smog system' as an example to describe FP vs. FN cost, did you mean we will be more concerns about HIGH FN than HIGH FP? Correct me if I did not get what you meant.

Regards,
Sunny

Jason Brownlee April 4, 2019 at 2:09 pm #

REPLY

Thanks.

Yes, it might be confusing. I was saying we want (are concerned with) low false neg, not false pos.

High false neg is a problem, high false pos is less of a problem.

Akhil April 11, 2019 at 2:05 pm #

REPLY

Hi Jason,

How do we decide on what is a good operating point for precision% and recall %? I know it depends on the use case, but can you give your thoughts on how to approach it?

Thanks!

Jason Brownlee April 11, 2019 at 2:22 pm #

REPLY

Yes, establish a baseline score with a naive method, and compare more sophisticated methods to the baseline.

Prashanth April 11, 2019 at 7:57 pm #

REPLY

Great post. Thank you Jason.
One query. what is the difference between area un
Both have similar definitions I guess.

Prashanth April 11, 2019 at 9:25 pm #

Also what approach do you recommend for selecting a threshold from the precision-recall curve, like the way we can use Youden's index for ROC curve?

Jason Brownlee April 12, 2019 at 7:45 am #

REPLY

I'd recommend looking at the curve for your model and choose a point where the trade off makes sense for your domain/stakeholders.

Jason Brownlee April 12, 2019 at 7:44 am #

REPLY

Great question. They are similar.

I can't give a good answer off the cuff, I'd have to write about about it and go through worked examples.

Prashanth April 22, 2019 at 8:21 pm #

REPLY

I am guessing both average precision score and area under precision recall curve are same. The difference arises in the way these metrics are calculated. As per the documentation page for AUC, it says

"Compute Area Under the Curve (AUC) using the trapezoidal rule

This is a general function, given points on a curve. For computing the area under the ROC-curve, see roc_auc_score. For an alternative way to summarize a precision-recall curve, see average_precision_score."

So i guess, it finds the area under any curve using trapezoidal rule which is not the case with average_precision_score.

Dana Averbuch April 24, 2019 at 1:57 am #

REPLY

Thanks for the nice and clear post ☺
Shouldn't it be "false negatives" instead of "false positives" in the following phrase:
"here is a tension between these options, the same with true negative and false positives."

Jason Brownlee April 24, 2019 at 8:07 ar

REPLY

I think you're right. Fixed.

ziad June 14, 2019 at 9:35 am #

Thanks for the nice and clear article.
i used GaussianNB model, i got the thresholds [2.00000e+000, 1.00000e+000, 1.00000e+000, 9.59632e-018].

is it noraml that the thresholds have very small value??

thanx in advance

Han Qi June 23, 2019 at 1:08 am #

REPLY

This line makes no sense to me at all : “Indeed, it has skill, but much of that skill is measured as making correct false negative predictions”

What is a “correct false negative”? The “correct” to my current understanding consist TP and TN, not FP or FN. If it’s correct, why is it false? If it’s false, how can it be correct?

Could you explain correct according to what? y_true or something else?

Jason Brownlee June 23, 2019 at 5:40 am #

REPLY

Looks like a typo, I believe I wanted to talk about true negatives, e.g. the abundant class.

Fixed. Thanks.

Ayoyinka June 24, 2019 at 7:59 pm #

REPLY

Great post, I found this very intuitive.

But why keep probabilities for the positive outcome only for the precision_recall_curve?
I tried with the probabilities for the negative class and the plot was weird. Please, I will like you to explain the intuition behind using the probabilities for the positive outcome and not the one for the negative outcome?

Abdur Rehman Nadeem July 22, 201

Actually scikit learn “predict_proba()” predict probability for each class for a row and it sums upto 1. In binary classification case, it predicts the probability for an example to be negative and positive and 2nd column shows how much probability of an example belongs to positive class. When we pass only positive probability, ROC evaluate on different thresholds and check if given probability > threshold (say 0.5), it belongs to p
Similarly, it evaluates on different thresholds ar

Zaki July 29, 2019 at 10:10 pm #

Thanks for explaining the ROC curve, i would like to aske how i can compare the Roc curves of many algorithms means SVM knn, RandomForest and so on.

REPLY

Jason Brownlee July 30, 2019 at 6:13 am #

Typically they are all plotted together.

You can also compare the Area under the ROC Curve for each algorithm.

REPLY

krs reddy July 29, 2019 at 11:41 pm #

can anyone explain whats the significance of average precision score?

REPLY

Jason Brownlee July 30, 2019 at 6:15 am #

Yes, see this:
https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision

REPLY

Walid August 10, 2019 at 1:07 am #

Thanks a lot for this tutourial. There are actually not a lot of resources like this.

REPLY

Jason Brownlee August 10, 2019 at 7:21 am #

Thanks, I'm glad it helped!

REPLY

Gianinna September 25, 2019 at 11:41 pm #

Hi Jason,

thank you for your tutorial!

I have a question about the F1 score, because i know the best value is 1 (perfect precision and recall) and worst value is 0, but i'm wondering if there is a minimun standard value.

I'm obtaining a F score of 0.44, because i have hig
know if 0.44 is enough to say that i have a good m
literature? for example in ROC about 0.75 is good

Thanks

Jason Brownlee September 26, 2019 at 6:41 am #

REPLY

Yes, the F1 score returned from a naive classification model:
https://machinelearningmastery.com/how-to-develop-and-evaluate-naive-classifier-strategies-using-probability/

Bishal Mandal September 26, 2019 at 8:59 pm #

REPLY

Hi Jason,

Thanks for the explaining these concepts in simple words!

I have a little confusion. You mentioned Roc would be a better choice if we have a balanced dataset, and precision-recall for an imbalanced dataset. But if we get an imbalanced dataset, will we not try to balance it out first and then start with the models?

Regards,
Bishal Mandal

Jason Brownlee September 27, 2019 at 8:00 am #

REPLY

We may decide to balance the training set, but not the test set used as the basis for making predictions.

Erfan Basiri October 5, 2019 at 8:21 am #

REPLY

i was thinking about this whole day ☺ .Thanks

Erfan Basiri October 5, 2019 at 8:19 am #

REPLY

HI.could you tell me whether i can create roc curve in this way or not ?

y_test=my test data ground truth (6500,1)
x_test= my test data (6500,3001)

prediction=model.predict(x_test)

fpr,tpr,thresholds=roc_curve(y_test,prediction)

plt.plot(fpr,tpr)
plt.show()

auc = roc_auc_score(y_test, prediction)

Jason Brownlee October 6, 2019 at 8:13 am #

REPLY

I can't debug your example sorry.

Perhaps try it?

Erfan Basiri October 8, 2019 at 9:59 am #

REPLY

Thanks,it's solved.

Jason Brownlee October 8, 2019 at 1:17 pm #

REPLY

Happy to hear that.

Erfan Basiri October 5, 2019 at 8:32 am #

REPLY

Sorry i have another question .when i saw the thresholds array elements , i noticed that its first element is about 1.996 . How is it possible ? thresholds should be between 0 and 1 , isn't it ?

Thanks again

Erfan Basiri October 5, 2019 at 8:48 am #

REPLY

Also i check it in your code , and it is the same , you're first element of thresholds is 2 !

Could yo tell me how the number of thresholds elements are obtained ? for example in your code you have thresholds have 5 elements but in my problem it has 1842 element

Jason Brownlee October 6, 2019 at 8:14

REPLY

Perhaps double check the value?

Chris October 5, 2019 at 8:20 pm #

REPLY

I suggest carefully rereading Aleks post ar
ROC is just for balanced data at the end, which it is
ROC in cases of N >> P can give a high AUC, but
to that. But it all depends on your objectives and I r
the many preferable statistical properties of the RO

https://stats.stackexchange.com/questions/7207/roc-vs-precision-and-recall-curves, he has written an excellent straightforward summary here which could be used to improve this blog post.

Great blog btw.

REPLY

Jason Brownlee October 6, 2019 at 8:17 am #

Thanks Chris, I'll take another run at the paper.

REPLY

jack October 8, 2019 at 8:14 pm #

my precision and recall curve goes up to the end but at the end it crashs. do you know why and is this ok? thanks.

REPLY

Jason Brownlee October 9, 2019 at 8:11 am #

Sorry to hear it crashes, why error does it give?

REPLY

Lilly Wilnson October 10, 2019 at 8:02 pm #

Many thanks Jason for this tutorial

I would like to ask about the ROC Curves and Precision-Recall Curves for deep multi-label Classification.

I was able to plot the confusion matrices for all the labels with sklearn.matrics (multilabel_confusion_matrix).. My question is, How can I plot the ROC Curves and Precision-Recall Curves for all the labels?

I appreciate any help.
Thanks a lot.

REPLY

Jason Brownlee October 11, 2019 at 6:16 am #

Generally PR Curves and ROC Curves are for 2-class problems only.

Lilly Wilnson October 11, 2019 at 11

Thanks a lot Jason for your reply.

so what are the evaluation matrices for mu
Just the ACC and Loss?

Jason Brownlee October 12, 2019 at 7:03 am #

REPLY

It depends on what is important for your problem.

Log loss is a good place to start for multiclass. For multilabel (something else entirely), average precision or F1 is good.

My best advice is to go back to stakeholders or domain experts, figure out what is the most important about the model, then choose a metric that captures that. You can get a good feeling for this by taking a few standard measures and running mock predictions through it to see what scores it gives and whether it tells a good story for you/stakeholders.

Lilly October 12, 2019 at 5:59 pm #

Great!

Many thanks Jason.

Jason Brownlee October 13, 2019 at 8:29 am #

You're welcome.

Saeed October 10, 2019 at 11:00 pm #

REPLY

Hello sir , thank you for your excellent tutorials!
i am facing a problem when ever i show the roc curve i get the following error
File "C:\Program Files\Python37\lib\site-packages\sklearn\metrics\base.py", line 73, in _average_binary_score raise ValueError("{0} format is not supported".format(y_type))

ValueError: multiclass format is not supported

sir please help me to solve this issue. Thanks

Jason Brownlee October 11, 2019 at 6:20 am #

REPLY

ROC Curves can only be used for binary (2 class) classification problems.

Brindha November 2, 2019 at 8:17 am #

It was very useful. Thanks for helping beg

Jason Brownlee November 3, 2019 at 5:41 am #

REPLY

I'm happy to hear that.

Shirin November 6, 2019 at 12:53 am #

REPLY

Hi Jason,

I am dealing with a medical database for prediction of extreme rare event (0.6% chance of occurrence). I have 10 distinct features, 28,597 samples from class 0 and 186 from class 1.

I have developed several models. Also, I have tried to downsample the training set to make a balanced training set and tested on imbalanced test set.

Unfortunately regardless of my model the PR curve is similar to "Precision-Recall Plot for a No Skill Classifier and a Logistic Regression Model for am Imbalanced Dataset" figure in this post.

Any idea how can I deal with this database? I would appreciate any suggestion!

Jason Brownlee November 6, 2019 at 6:35 am #

REPLY

Yes, some ideas:

– try standard models with a range of data scaling
– try class weighted versions of models like logistic regression, svm, etc.
– try undersampling methods
– try over sampling methods
– try combinations of over and under sampling on the same training set
– try one class classifiers
– try ensemble methods designed for imbalanced data
– try an alternate performance metric
– try k-fold cross validation to estimate pr auc

I hope that helps as a start.

Shirin Najdi November 12, 2019 at 2:30 am #

REPLY

Thanks a lot. I already started to use your suggestions. Wish me luck and patience 😀

Jason Brownlee November 12

Good luck!

Christian Post November 8, 2019 at 12:28 am #

REPLY

I have the same problem in the data I am dealing with (around 0.3% occurrence).

I assume you have some expert knowledge about the biological connections between the features and your event, but try calculating the Pearson (point biserial) correlation of each feature with the target, this should give you a hint whether there is any connection between your features and the event you want to predict.

You could also try unsupervised learning (clustering) to see if the event is only located within certain clusters.

Validating with ROC can be a bit tricky in the case that not enough positive events end up in the validation data set.
Though this is a bit cheaty because you would make assumptions about the validation data beforehand, split the negative and positive cases seperately so that you end up with the same prevalence in training and validation data.

Jason Brownlee November 8, 2019 at 6:43 am #

REPLY

Great tips.

A LOOCV evaluation is a good approach with limited data.

Shirin Najdi November 13, 2019 at 2:09 am #

REPLY

Thanks for the tips. I am considering them.

Christian Post November 8, 2019 at 12:37 am #

REPLY

Hello Jason,

great article.
I stumbled upon the PLoS One paper (Saito and Rehmsmeier 2015) myself and I have one question regarding the evaluation of the PRC.

The authors state:

“Nonetheless, care must be taken when interpolations between points are performed, since the interpolation methods for PRC and ROC curves differ—ROC analysis uses linear and PRC analysis uses non-linear interpolation. Interpolation between two points A and B in PRC space can be represented as a function y = (TPA + x) / {TPA + x + FPA + ((FPB – FPA) * x) / (TPB – TPA)} where x can be any value between TPA and TPB [26].”

In your article, you calculated the AUC (PRC) with
Is this in conflict with the quoted statement since I
TPR)?

I don't think this matters much when I am comparing models within my own trial, but what about comparing the AUC to other papers?

REPLY

Jason Brownlee November 8, 2019 at 6:48 am #

Hmmm. Yes, you may be right.

Here comes a rant.

As a general rule, repeat an an experiment and compare locally.

Comparing results to those in papers is next to useless as it is almost always the case that it is insufficiently described – which in turn is basically fraud. I'm not impressed with the computational sciences.

REPLY

Pradeep November 12, 2019 at 12:00 pm #

how to calculate the probabilities that i need to pass for below funciton

roc_curve(y, probs)

REPLY

Jason Brownlee November 12, 2019 at 2:06 pm #

You can call model.predict_proba() to predict probabilities.

REPLY

Im November 18, 2019 at 2:10 am #

Can I ask why you said that in the case of precision -recall we're less interested in high true negative? Is it because you took class 0 to be the dominant class? But isn't the choice of class 0 as being the dominant class just an example? So, in the case of class 1 being the dominant class, that would mean that the model will be less interested in true positives. And in this case your point about true negatives not figuring in the precision and recall formulas wouldn't be relevant

REPLY

Jason Brownlee November 18, 2019 at 6:48 am #

In binary classification, class 0 is the negative/majority class and class 1 is always the positive/minority class. This is convention.

Surajit Chakraborty December 4, 2019 at 8:

Hi,

Is there any formula to determine the optimal threshold from an ROC Curve ?

Thanks
Surajit

REPLY

Jason Brownlee December 4, 2019 at 8:43 am #

Yes, I have a post scheduled on the topic.

You can use the j-statistic:
https://en.wikipedia.org/wiki/Youden%27s_J_statistic

REPLY

Surajit Chakraborty December 4, 2019 at 8:53 am #

Thanks for your reply. How shall i come to know about your post on this topic ?

REPLY

Jason Brownlee December 4, 2019 at 1:57 pm #

You can follow the site via email/rss/twitter/facebook/linkedin.

See the links at the bottom of every page.

REPLY

Surajit Chakraborty December 4, 2019 at 8:57 am #

Also, how to determine the optimal threshold from a PR Curve ? Is it the F-Score or something else ?

REPLY

Jason Brownlee December 4, 2019 at 1:58 pm #

Excellent question!

You can test each threshold in the curve using the F-measure.

I cover this in an upcoming post as well.

REPLY

Isabell Orlis December 7, 2019 at 5:55 am #

Hey there!
You obtain the thresholds as "_" through the call "lr
lr_probs)" – but you never use them when plotting
thresholds? I mean, you have to be using them in s

Jason Brownlee December 8, 2019 at 5:59 am #

REPLY

Correct, we are not using the thresholds directly, rather a line plot of recall vs precision.

voodoo December 15, 2019 at 1:28 pm #

REPLY

well, does it then mean for roughly balanced dataset I can safely ignore Precision Recall curve score and for moderately (or largely) imbalances dataset, I can safely ignore AUC ROC curve score?

Jason Brownlee December 16, 2019 at 6:09 am #

REPLY

No, you must select the metric that is most appropriate for your task, then use it to evaluate and choose a model.

Metric first.

voodoo December 16, 2019 at 2:28 pm #

REPLY

thank you, I was talking about specifically binary classification task. And I have two datasets. one is imbalanced (1:2.7) and the second one is almost perfectly balanced. which metric should I choose for the two? Thank you once again, cheers!

Jason Brownlee December 17, 2019 at 6:28 am #

REPLY

I recommend choosing a metric that best captures the requirements of the project for you and the stakeholders.

A good starting point is to think about what is important about classification and misclassification errors. Are errors symmetrical? Are both classes important, etc.

Some metrics to consider include roc a

voodoo December 17, 2019 at 10:56 am #

you're awesome, thank yo

Jason Brownlee Decemb

You're welcome.

Amani December 18, 2019 at 6:49 am #

REPLY

Hi, Thanks for your excellent post.
My question is if I do resampling to my imbalance dataset, can I use AUC in this case to evaluate the model?

Jason Brownlee December 18, 2019 at 1:26 pm #

REPLY

Yes.

Amani December 19, 2019 at 1:25 am #

REPLY

Sorry I meant can I use the ROC curves to evaluate the model in this case?

Jason Brownlee December 19, 2019 at 6:33 am #

REPLY

Yes. Any metric you want.

Sampling only impacts the training set, not the test set used for evaluation.

Mujeeb December 18, 2019 at 6:07 pm #

REPLY

Hi, Jasono, you explain in a way that I always write ‘machinelearningmastery’ in the end of my query at the end. Thanks

My probllem is: I have imbalance dataset of (22:1, positive:negative) and I train the neural network model with the ‘sigmoid’ as a activation function in last (output) layer. After training I call the function ‘model.predict’ as below:

y_prediction = model.predict(X_test)

and when I print y_prediction it shows me float values between 1 and 0 (I am thinking that these are the probabilities of class 1(positive) for every X_test sample).

After that I convert y_prediction to 1s and 0s by threshold of 0.5 as blow:

```
y_pred = np.zeros_like(y_prediction)
y_idx = [y_prediction >= 0.5]
y_pred[y_idx] = 1
```

After that I draw Precision-Recall Curve (PR-Curve

Now I have following Questions:
Q1: How I have to get the best threshold from my P
y_prediction(mentioned above) and results me in g

Q2: How I became satisfied that this precision and recall or F1-score are good and model perform well.

Thanks

REPLY

Jason Brownlee December 19, 2019 at 6:28 am #

Thanks!

You can test all thresholds using the F-measure, and use the threshold with the highest F-measure score. I have a tutorial on this scheduled.

Not sure I understand the second question. Use lots of data and repeated evaluation to ensure the score is robust?

REPLY

Alpha9 January 9, 2020 at 5:15 pm #

How to plot ROC when I have a predicted output as a reconstructed image (e.g. Autoencoder), suppose I calculate some score of reconstructed an consider as a binary classification. I have tried using scikit-learn package and the result is almost the same with my training experiment.
my model structure:
Image sequence as Input -> Encoder -> ConvLSTM -> Decoder -> output a reconstructed image sequence

REPLY

Jason Brownlee January 10, 2020 at 7:23 am #

Not sure that is possible.

REPLY

ARSHAD ALI January 10, 2020 at 3:54 pm #

Sir how i can draw the ROC Curve by using PROMISE DATASET KC1, Kindly help.

REPLY

Jason Brownlee January 11, 2020 at 7:21 am #

A ROC curve is drawn for predictions, not for a dataset.

First model the dataset, then make predictions on a test set, then use the code above to draw a ROC curve from the predictions.

lee January 14, 2020 at 2:14 pm #

great

Jason Brownlee January 15, 2020 at 8:17 am #

REPLY

Thanks!

Jens Kaae January 23, 2020 at 8:59 pm #

REPLY

Hi Jason Brownlee,

First, I regret the loooong question, below. Having said that….;o)

…I cannot escape the fact that you represent a significant capacity in the field of data science and that you show an impressive willingness to share your valuable knowledge with others. In light of that, I hope that you could possible assist me providing some advice. I have two questions.

I am currently in the process of defining a KNN based classification of a relatively imbalanced, some might say highly imbalanced dataset, consisting of 2K+ articles of various scientific content. Purpose of algo is to clarify which articles to scrap and, more interestingly, which ones to proceed with. These ones are highly outnumbered by the ones not of interest. I've managed to make the algo run, and it does a pretty fine job. I think. This is where I'd like you to tell me, after having pre-processed text, vectorized, how I can evaluate on the KNN model's performance. Reading your article, I seem to understand that precision-recall curves are better at showing results when working with (highly?) imbalanced datasets.

Therefore, and first:
Do I need to do a confusion matrix before a precision-recall curve, and do both these exist as standard components of the scikit-learn module?

Second, I seem to face an issue when executing on the KNN classification part of the algo. I am importing a csv, adding a unique id to each line/text representation, and then–when testing relationship of a new observation–I am adding a number from the list of unique id's I added. Then–in presenting result–I concatenate to df's and get an overview of e.g. 20 nearest neighbors. Could be 10 or whatever. This works fine. It points back to a list where I can see that my preferred articles are highlighted, typically at rate of precision betw 75 and 80 per cent. My issue is really that I'd like to be able to add a piece of text/search word in stead of a number. From the tutorial I 'inherited' my own adaptation, it says 'Barack Obama's nearest neighbors are this and that, but that is only because the data applied i structured in a 'name' and 'text' feature. My articles cannot be split like that. While overall covering the same scientific area, they are different, very different. So my quest
index by that new one and then use text as search
'get_closest_neighs' result?

Jason Brownlee January 24, 2020 at 7:51 am #

REPLY

Thanks.

Start with a good idea of what is important to th

This will help a lot, specifically the end:
https://machinelearningmastery.com/tour-of-ev

Not sure I follow your second question sorry. It might be more of an engineering question than a modeling question.

jens January 24, 2020 at 9:21 am #

REPLY

Hi, this is the tutorial I used, re question 2 – http://andrewgaidus.com/Finding_Related_Wikipedia_Articles/

Niwhset February 11, 2020 at 8:45 pm #

REPLY

You mentioned “This is possible because the model predicts probabilities and is uncertain about some cases. These get exposed through the different thresholds evaluated in the construction of the curve, flipping some class 0 to class 1, offering some precision but very low recall.”

Shouldn’t this offer some recall but very low precision as depicted by the graph?

Erick Almaraz February 19, 2020 at 2:55 am #

REPLY

Very illuminating!

Jason Brownlee February 19, 2020 at 8:07 am #

REPLY

Thanks!

Leave a Reply

Website

SUBMIT COMMENT

Welcome!

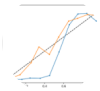
My name is *Jason Brownlee* PhD, and I **help developers** get results with **machine learning**.

Read more

Never miss a tutorial:

Picked for you:

How to Use ROC Curves and Precision-Recall Curves for Classification in Python



How and When to Use a Calibrated Classification Model with scikit-learn

A Gentle Introduction to Probability Scoring Methods in Python

How to Develop and Evaluate Naive Classifier Strategies Using Probability

A Gentle Introduction to Cross-Entropy for Machine Learning

Start Machine Learning

Loving the Tutorials?

The Probability for Machine Learning EBook is where I keep the ***Really Good*** stuff.

SEE WHA

Thank you for signing up!

Please check your email and click the link provided to confirm your subscription.