What metrics should be used for evaluating a model on an imbalanced data set? (precision + recall or ROC=TPR+FPR)

Shir Meir Lador Sep 5, 2017 · 9 min read

I always thought the subject of metrics to be somehow confusing, specifically when the data set is imbalanced (as happens so often in our usual problems). In order to clarify things I've decided to test a few simple examples of an imbalanced data sets with the different type of metrics and see which reflects more correctly the model performance — ROC curve metrics — TPR and FPR or precision or recall.

Definitions

So let's start by reminding ourselves the definitions.

In the ROC curve we look at:
TPR (True Positive Rate) = # True positives / # positives = Recall = TP / (TP+FN)
FPR (False Positive Rate) = # False Positives / # negatives = FP / (FP+TN)

Here we will focus on the TPR (True Positive Rate) and FPR (False Positive Rate) of a single point (this will indicate the general performance of the ROC curve which consists of the TPR and FPR through various probability thresholds).

Precision and recall are:
Precision =# True positives / # predicted positive = TP/(TP+FP)
Recall = # True positives / # positives = TP / (TP+FN)

What is the difference?

The main difference between these two types of metrics is that precision denominator contains the False positives while false positive rate denominator contains the true negatives.
While precision measures the probability of a sample classified as positive to actually be positive, the false positive rate measures the ratio of false positives within the negative samples.

With a large number of negative samples — precision is probably better

If the number of negative samples is very large (a.k.a imbalance data set) the false positive rate increases more slowly. Because the true negatives (in the fpr denominator — (FP+TN)) would probably be very high and make this metric smaller.
Precision however, is not affected by a large number of negative samples, that's because it measures the number of true positives out of the samples predicted as positives (TP+FP).

Precision is more focused in the positive class than in the negative class, it actually measures **the probability of correct detection of positive values**, while FPR and TPR (ROC metrics) measure **the ability to distinguish between the classes**.

Examples

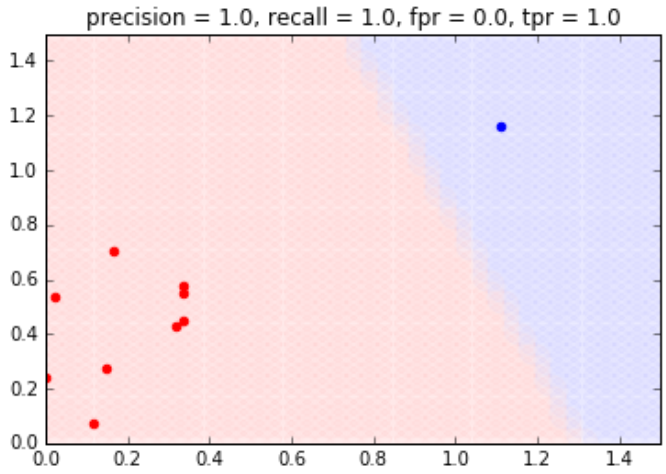
The examples are just a way to illustrate the metrics with a visual image.

For visualisation purposes, Here is a simple function which plot the decision region of a given model.

```
def plot_model_boundaries(model, xmin=0, xmax=1.5, ymin=0, ymax=1.5,
npoints=40):
    xx = np.linspace(xmin, xmax, npoints)
    yy = np.linspace(ymin, ymax, npoints)
    xv, yv = np.meshgrid(xx, yy)
    xv, yv = xv.flatten(), yv.flatten()
    labels = model.predict(np.c_[xv,yv])
    plt.scatter(xv[labels==1],yv[labels==1],color='r', alpha=0.02,
marker='o', s=300)
    plt.scatter(xv[labels==0],yv[labels==0],color='b', alpha=0.02,
marker='o', s=300)
    plt.ylim([xmin, xmax])
    plt.xlim([ymin, ymax])
```

```
from matplotlib import pyplot as plt
from sklearn.svm import LinearSVC
from sklearn.metrics import precision_score, recall_score, roc_curve

np.random.seed(1)
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 0.8*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
y = np.concatenate([np.ones(9), np.zeros(1)])
model = LinearSVC()
model.fit(x,y)
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=1.5, ymin=0, ymax=1.5)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {}, recall = {}, fpr = {}, tpr = {}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[0], tpr[0]));
```



My simple data set and the model decision region

Example #1 — Majority of positive samples and — all positive samples are detected but there are also false positives — ROC is a better metric

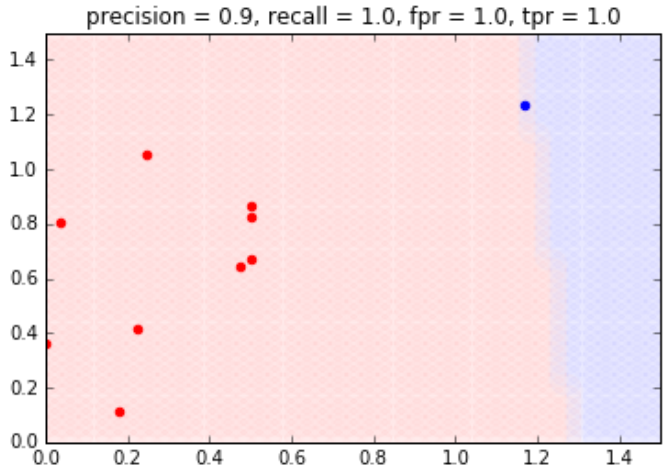
We have 10 samples in test dataset. 9 samples are positive and 1 is negative.

As we saw before — all metrics are perfect in the case of the perfect model, but now we look at a naive model which predicts **everything positive.**

Precision = TP/(TP+FP) = 0.9, **Recall** = TP/(TP+FN)= 1.0.
In this case the precision and recall are both very high, but we have a poor classifier.

TPR = TP/(TP+FN) = 1.0, **FPR** = FP/(FP+TN) = 1.0.
Because the FPR is very high, we can identify that this is not a good classifier.

```
np.random.seed(1)
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 1.2*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
y = np.concatenate([np.ones(9), np.zeros(1)])
model = LinearSVC()
model.fit(x,y)
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=1.5, ymin=0, ymax=1.5)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {}, recall = {}, fpr = {}, tpr = {}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[1], tpr[1]));
```

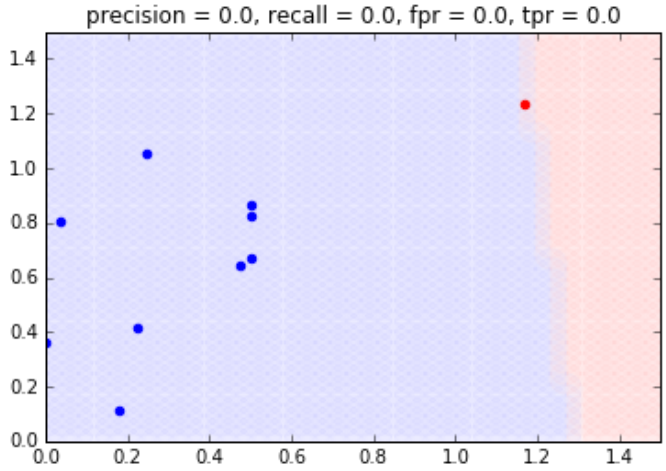


The classifier predicts all the labels as positive

Example #1 —same data set with opposite labels — Both metrics are 0 because there is no “detection”

Now, we switch the labels — 9 samples are negative and 1 is positive.
We have a model which predicts everything negative. Our basic metrics are now: **TP =**

```
np.random.seed(1)
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 1.2*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
y = np.concatenate([np.zeros(9), np.ones(1)])
model = LinearSVC()
model.fit(x,y)
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=1.5, ymin=0, ymax=1.5)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {}, recall = {}, fpr = {}, tpr = {}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[0], tpr[0]));
```



The classifier predicts all the labels as negatives so all the metrics are zeros

Example #2 — Majority of positive samples — all positive samples are detected but there are also false positives— ROC is a better metric

This example is similar to example #1 and shows basically the same thing.

In this case 8 samples are positive and 2 are negative.

The model predicts 9 of the samples as positives (8 true positives and 1 negative) and one as negative.

The basic metrics are: **TP = 8, FP = 1, TN = 1, FN = 0.**

The advanced metrics are:

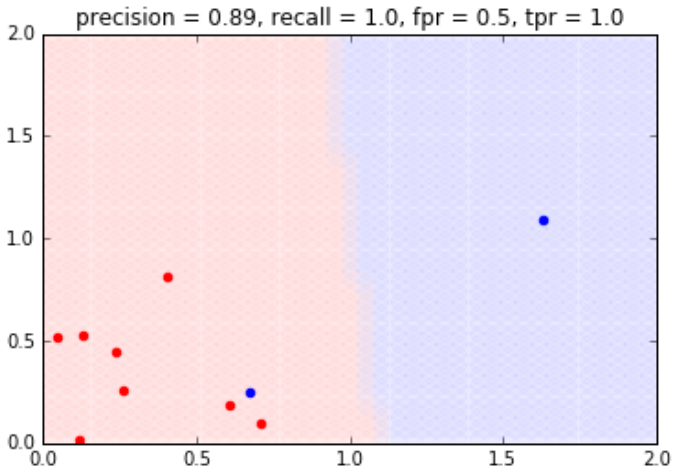
class is good.

TPR = TP/(TP+FN) = 1, FPR = FP/(FP+TN) = 1/2 = 0.5.

The False Positive rate is 0.5, which is pretty high, because we have 1 False positive out of two negatives — that's a lot!

Notice that high FPR is actually the same thing as a low recall for the negative class.

```
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 0.9*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
y = np.concatenate([np.zeros(1), np.ones(8), np.zeros(1)])
model = LinearSVC()
model.fit(x,y)
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=2, ymin=0, ymax=2)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {:.2f}, recall = {}, fpr = {}, tpr =
{}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[1], tpr[1]));
```



One negative sample is classified correctly as negative and the other is classified as positive — this caused a small decrease in precision and a relatively high value in FPR

Example #2 — opposite labels — Both metrics are the same in this case

Now 8 samples are negative and 2 are positive.

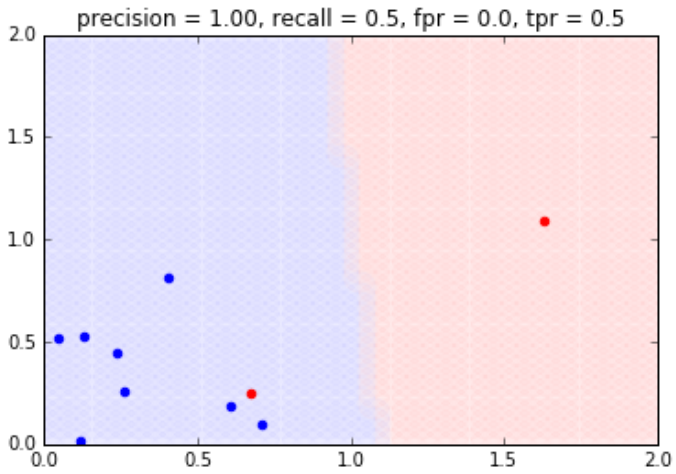
We have a model which predicts everything negative but one positive sample (which is

The advanced metrics are:

Precision = TP/(TP+FP) = 1, Recall = TP/(TP+FN)= 1/(1+1) = 0.5.
TPR = TP/(TP+FN) = 1/2 = 0.5, FPR = FP/(FP+TN) = 0.

In this case both metrics give the same information.

```
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 0.9*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
y = np.concatenate([np.ones(1), np.zeros(8), np.ones(1)])
model = LinearSVC()
model.fit(x,y)
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=2, ymin=0, ymax=2)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {:.2f}, recall = {}, fpr = {}, tpr =
{}".format(
 precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[0], tpr[0]));
```



One positive sample is detected correctly and the other is not

Example #3 — Majority of positive samples and not all positive samples are detected — Both metrics are the same in this case

Now, 9 samples are positive and 1 is negative. The model predicts 7 of the samples as positive (all are positive) and 3 as negative.

The basic metrics are: TP = 7, FP = 0, FN = 2, TN = 2

The advanced metrics are:

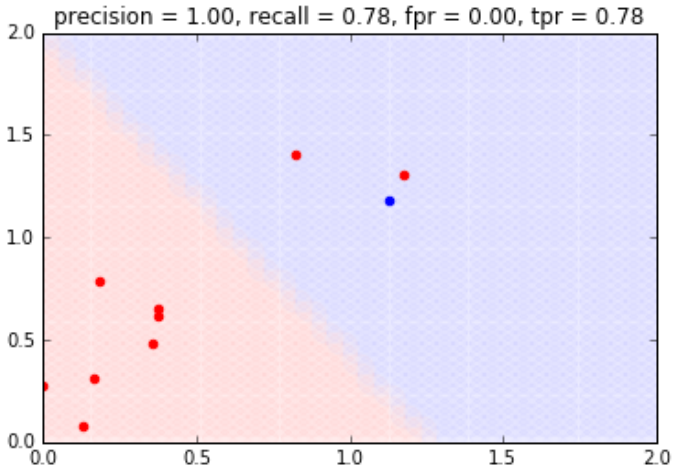
Precision = TP/(TP+FP) = 1, Recall = TP/(TP+FN)= 7/9 = 0.78

The precision and recall are both very high, because the performance on the positive class is good.

TPR = TP/(TP+FN) = 7/9 = 0.78, FPR = FP/(FP+TN) = 0.

Both metrics give similar results in this case.

```
np.random.seed(1)
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 0.9*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
x[-2,0]+=0.8
x[-2,-1]+=0.8
x[-3,0]+=0.8
x[-3,-1]+=0.8
y = np.concatenate([np.zeros(7), np.ones(3)])
y = 1-y
model = LinearSVC()
model.fit(x,y)
y = np.concatenate([np.zeros(9), np.ones(1)])
y = 1-y
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=2, ymin=0, ymax=2)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {:.2f}, recall = {:.2f}, fpr = {:.2f}, tpr =
{:.2f}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[0], tpr[0]));
```



Example #3 — opposite labels —Majority of negative samples and not all positive samples are detected— precision and recall are better

Now we switch the labels — 9 samples are negative and 1 is positive.
The model predicts 3 of the samples as positive (only one of them is actually positive) and 7 as negative.
The basic metrics are: **TP = 1, FP = 2, TN = 7, FN = 0.**

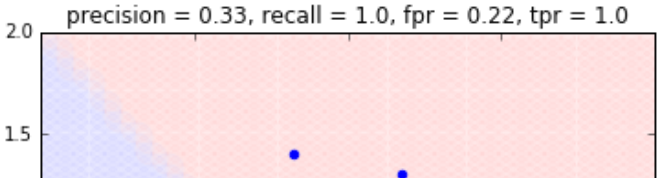
The advanced metrics are:

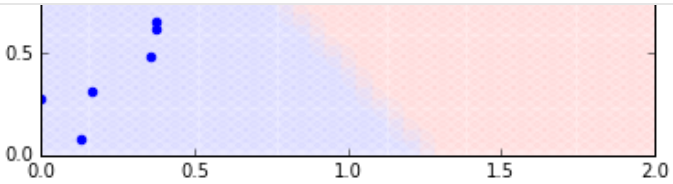
Precision = TP/(TP+FP) = 0.33, Recall = TP/(TP+FN)= 1.

TPR = TP/(TP+FN) = 1, FPR = FP/(FP+TN) = 2/9 = 0.22.

In this case the precision is low and the FPR is not as high as we would want it to be. **The FPR is not high because we have a lot of true negatives, due to the imbalanced data set. In this case the poor detection ability is reflected best in the precision.**

```
np.random.seed(1)
x = np.concatenate([np.zeros((8, 2)), np.zeros((1,2)),
np.zeros((1,2))]) + 0.9*np.random.rand(10,2)
x[-1,0]+=1
x[-1,-1]+=1
x[-2,0]+=0.8
x[-2,-1]+=0.8
x[-3,0]+=0.8
x[-3,-1]+=0.8
y = np.concatenate([np.zeros(7), np.ones(3)])
model = LinearSVC()
model.fit(x,y)
y = np.concatenate([np.zeros(9), np.ones(1)])
predicted_labels = model.predict(x)
plot_model_boundaries(model, xmin=0, xmax=2, ymin=0, ymax=2)
plt.scatter(x[y==0,0],x[y==0,1], color='b')
plt.scatter(x[y==1,0],x[y==1,1], color='r');
fpr, tpr, thresholds = roc_curve(y, predicted_labels)
plt.title("precision = {:.2f}, recall = {}, fpr = {:.2f}, tpr =
{}".format(
    precision_score(y, predicted_labels), recall_score(y,
predicted_labels), fpr[1], tpr[1]));
```





In this case the poor detection ability is reflected best in the precision while the FPR is relatively not that high because of the large amount of negative samples

Final intuition to metric selection

1. **Use precision and recall to focus on small positive class —** When the positive class is smaller and the ability to detect correctly positive samples is our main focus (correct detection of negatives examples is less important to the problem) we should use precision and recall.

2. **Use ROC when both classes detection is equally important —** When we want to give equal weight to both classes prediction ability we should look at the ROC curve.

3. **Use ROC when the positives are the majority or switch the labels and use precision and recall —** When the positive class is larger we should probably use the ROC metrics because the precision and recall would reflect mostly the ability of prediction of the positive class and not the negative class which will naturally be harder to detect due to the smaller number of samples. If the negative class (the minority in this case) is more important, we can switch the labels and use precision and recall (As we saw in the examples above — switching the labels can change everything).