Lecture 17
Evaluating Classification Models

Dennis Sun
Stanford University
DATASCI / STATS 112

February 22, 2023

1 Review

2 Precision and Recall

3 Announcements

Case Study: Credit Card Fraud

Dataset of credit card transactions in September 2013 by European cardholders.

```python
import pandas as pd
df_fraud = pd.read_csv(
    "https://datahub.io/machine-learning/creditcard/r/creditcard.csv")
df_fraud
```

	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053	149.62	'0'
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724	2.69	'0'
2	1.0	-1.358354	-1.340163	1.773209	...	-0.055353	-0.059752	378.66	'0'
3	1.0	-0.966272	-0.185226	1.792993	...	0.062723	0.061458	123.50	'0'
4	2.0	-1.158233	0.877737	1.548718	...	0.219422	0.215153	69.99	'0'
...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	...	0.943651	0.823731	0.77	'0'
284803	172787.0	-0.732789	-0.055080	2.035030	...	0.068472	-0.053527	24.79	'0'
284804	172788.0	1.919565	-0.301254	-3.249640	...	0.004455	-0.026561	67.88	'0'
284805	172788.0	-0.240440	0.530483	0.702510	...	0.108821	0.104533	10.00	'0'
284806	172792.0	-0.533413	-0.189733	0.703337	...	-0.002415	0.013649	217.00	'0'

284807 rows × 31 columns

Goal: Predict `Class`, where 1 indicates a fraudulent transaction.

Training a Classifier

```
X_train = df_fraud.loc[:, "V1":"V28"]
y_train = df_fraud["Class"]
```

Last time, we saw how k-nearest neighbors could be used for classification:

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=20)
cross_val_score(model, X_train, y_train,
                scoring="accuracy", cv=10).mean()
```

```
0.9992731942525058
```

How is the accuracy so high?

A Closer Look

Let's take a closer look at the labels.

```
y_train.value_counts()
```

```
'0'    284315
'1'       492
Name: Class, dtype: int64
```

Almost all of the transactions are normal!

We could get 99.8% accuracy just by predicting that every transaction is normal.

Although this model is accurate *overall*, it is inaccurate for fraudulent transactions. A good model is "accurate for every class".

1. Review
2. Precision and Recall
3. Announcements

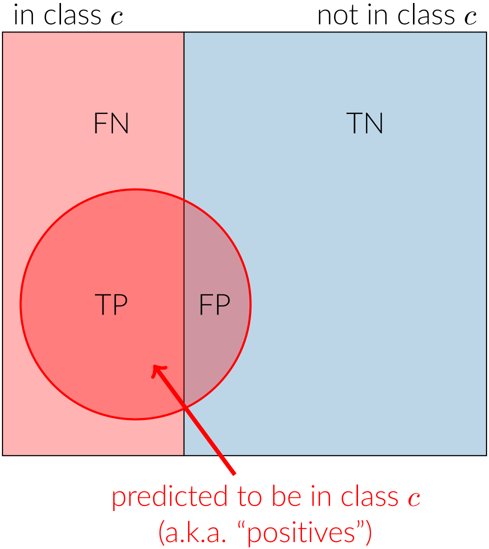
Precision and Recall

We need a score that measures "accuracy for class c".

There are at least two reasonable definitions:

- **precision**: $p(\text{correct}|\text{predicted class } c)$
  Among the observations that were predicted to be in class c, what proportion actually were?
- **recall**: $p(\text{correct}|\text{actual class } c)$.
  Among the observations that were actually in class c, what proportion were predicted to be?

A Geometric Look at Precision and Recall



$$\textbf{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \textbf{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Exercise: Calculating Precision and Recall

To check our understanding of these definitions, let's calculate a few precisions and recalls by hand.

First, summarize the results by the **confusion matrix**.

```
from sklearn.metrics import confusion_matrix
model.fit(X_train, y_train)
y_train_ = model.predict(X_train)
confusion_matrix(y_train, y_train_)
```

```
array([[284267,     48],
       [   116,    376]])
```

← actually in class 0

← actually in class 1

- What is the (training) accuracy? 99.99%
- What's the precision for normal transactions? 99.96%
- What's the recall for normal transactions? 99.98%
- What's the precision for fraudulent transactions? 88.68%
- What's the recall for fraudulent transactions? 76.42%

Note that each class has its own precision and recall!

Tradeoff between Precision and Recall

Can you imagine a classifier that always has 100% recall for class c, no matter the data?

In general,

- precision increases if we classify fewer observations as c
- recall increases if we classify more observations as c

How do we compare two classifiers, if one has higher precision and the other has higher recall?

The **F1 score** combines precision and recall into a single score:

$$\text{F1 score} = \text{harmonic mean of precision and recall}$$
$$= 1\Big/\frac{1}{2}\Big(\frac{1}{\text{precision}} + \frac{1}{\text{recall}}\Big)$$

So the F1 score of the classifier for fraudulent transactions is

$$1\Big/\frac{1}{2}\Big(\frac{1}{.8868} + \frac{1}{.7642}\Big) \approx 82.1\%.$$

To achieve a high F1 score, both precision and recall have to be high. If either one is low, then the harmonic mean will be low.

Precision, Recall, and F1 in Scikit-Learn

Remember that each class has its own precision, recall, and F1.

For `cross_val_score`, the `scoring=` parameter must be a single number.
For this, we can use

- `"precision_macro"`
- `"recall_macro"`
- `"f1_macro"`

which averages the score over the classes.

Precision-Recall Curve

Another way to illustrate the tradeoff between precision and recall is to graph the **precision-recall curve**.

First, we need the predicted probabilities.

```
y_train_probs_ = model.predict_proba(X_train)
y_train_probs_
```

```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

So far, we have been implicitly using a threshold of 0.5 to classify a transaction as fraud.

But what if we instead used a different threshold t? Depending on what t we pick, we'll get a different precision and recall. We can graph the tradeoff.

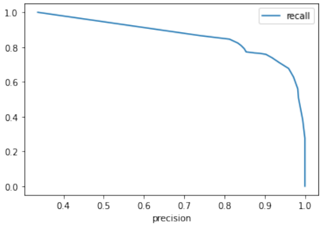
Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve

precision, recall, thresholds = precision_recall_curve(
    y_train, y_train_probs_[:, 1], pos_label="'1'")
precision
```

```
array([0.33606557, 0.73958333, 0.8125    , 0.83333333, 0.84143763,
       0.84934498, 0.85393258, 0.86136364, 0.86896552, 0.87470998,
       0.88679245, 0.9031477 , 0.91898734, 0.93582888, 0.95965418,
       0.97169811, 0.98220641, 0.98418972, 0.99468085, 1.        ,
       1.        ])
```

```
pd.DataFrame({ "precision": precision, "recall": recall
}).plot.line(x="precision", y="recall")
```



1. Review
2. Precision and Recall
3. Announcements

Reminders

- Final project examples posted to website.
- Assignment 5 is due next Tuesday. There's a new Kaggle competition and a new prize for winning this one.
- Don't forget to try the Colab for section tomorrow.
- Exam 2 is Monday 3/6. More details (including a practice exam) will be released on Monday.
- Office hours right now!