

How to Interpret the Classification Report in sklearn (With Example)

When using **classification models** in machine learning, there are three common metrics that we use to assess the quality of the model:

1. Precision: Percentage of correct positive predictions relative to total positive predictions.

2. Recall: Percentage of correct positive predictions relative to total actual positives.

3. F1 Score: A weighted harmonic mean of precision and recall. The closer to 1, the better the model.

- F1 Score: $2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

Using these three metrics, we can understand how well a given classification model is able to predict the outcomes for some **response variable**.

Fortunately, when fitting a classification model in Python we can use the **classification_report()** function from the **sklearn** library to generate all three of these metrics.

The following example shows how to use this function in practice.

Example: How to Use the Classification Report in sklearn

For this example, we'll fit a logistic regression model that uses points and assists to predict whether or not 1,000 different college basketball players get drafted into the NBA.

First, we'll import the necessary packages to perform logistic regression in Python:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

Next, we'll create the data frame that contains the information on 1,000 basketball players:

```
#make this example reproducible
np.random.seed(1)

#create DataFrame
df = pd.DataFrame({'points': np.random.randint(30, size=1000),
                    'assists': np.random.randint(12, size=1000),
                    'drafted': np.random.randint(2, size=1000)})

#view DataFrame
df.head()

      points  assists  drafted
0        5         1         1
1       11         8         0
2       12         4         1
3        8         7         0
4        9         0         0
```

Note: A value of **0** indicates that a player did not get drafted while a value of **1** indicates that a player did get drafted.

Next, we'll split our data into a training set and testing set and fit the logistic regression model:

```
#define the predictor variables and the response variable
X = df[['points', 'assists']]
y = df['drafted']

#split the dataset into training (70%) and testing (30%) sets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)

#instantiate the model
logistic_regression = LogisticRegression()

#fit the model using the training data
logistic_regression.fit(X_train,y_train)
```

```
#use model to make predictions on test data
y_pred = logistic_regression.predict(X_test)
```

Lastly, we'll use the **classification_report()** function to print the classification metrics for our model:

```
#print classification report for model
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.51	0.58	0.54	160
1	0.43	0.36	0.40	140
accuracy			0.48	300
macro avg	0.47	0.47	0.47	300
weighted avg	0.47	0.48	0.47	300

Here's how to interpret the output:

Precision: Out of all the players that the model predicted would get drafted, only **43%** actually did.

Recall: Out of all the players that actually did get drafted, the model only predicted this outcome correctly for **36%** of those players.

F1 Score: This value is calculated as:

- F1 Score: $2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$
- F1 Score: $2 * (.43 * .36) / (.43 + .36)$
- F1 Score: **0.40**.

Since this value isn't very close to 1, it tells us that the model does a poor job of predicting whether or not players will get drafted.

Support: These values simply tell us how many players belonged to each class in the test dataset. We can see that among the players in the test dataset, **160** did not get drafted and **140** did get drafted.

Note: You can find the complete documentation for the `classification_report()` function [here](#).

Additional Resources

The following tutorials provide additional information on how to use classification models in Python:

[How to Perform Logistic Regression in Python](#)

[How to Create a Confusion Matrix in Python](#)

[How to Calculate Balanced Accuracy in Python](#)



Published by Zach

[View all posts by Zach](#)