Understanding Correlation and Error Metrics (Part 2)

ISE4132 : AI Application System

AMIN AL (아민알)
Integrated System Engineering (ISE)
010-6853-6648
alaminanik@inha.ac.kr

Classification Evaluation Metrics

How to measure the performance of classification models?

True/False, Positives & Negatives

Confusion Matrix

Confusion matrix is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong. This helps to understand where the model is making mistakes so that we can improve it. It breaks down the predictions into four categories.

Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

True Positive (TP): The model correctly predicted a positive outcome i.e the actual outcome was positive.

True Negative (TN): The model correctly predicted a negative outcome i.e the actual outcome was negative.

False Positive (FP): The model incorrectly predicted a positive outcome i.e the actual outcome was negative. It is also known as a Type I error.

False Negative (FN): The model incorrectly predicted a negative outcome i.e the actual outcome was positive. It is also known as a Type II error.

Accuracy

Accuracy shows how many predictions the model got right out of all the predictions. It gives idea of overall performance but it can be misleading when one class is more dominant over the other. For example a model that predicts the majority class correctly most of the time might have high accuracy but still fail to capture important details about other classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

Precision focus on the quality of the model's positive predictions. It tells us how many of the "positive" predictions were actually correct. It is important in situations where false positives need to be minimized such as detecting spam emails or fraud.

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall measures how how good the model is at predicting positives. It shows the proportion of true positives detected out of all the actual positive instances. High recall is essential when missing positive cases has significant consequences like in medical tests.

$$Recall = \frac{TP}{TP + FN}$$

F1 Score

F1-score combines precision and recall into a single metric to balance their trade-off. It provides a better sense of a model's overall performance particularly for imbalanced datasets. It is helpful when both false positives and false negatives are important though it assumes precision and recall are equally important but in some situations one might matter more than the other.

$$F1 - Score = \frac{2.Precision.Recall}{Precision + Recall}$$

Example: Confusion Matrix For Binary Classification

Confusion Matrix for Dog Image Recognition with Numbers

Index	1	2	3	4	5	6	7	8	9	10
Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result	TP	FN	TP	TN	TP	FP	TP	TP	TN	TN

Actual Dog Counts = 6
Actual Not Dog Counts = 4
True Positive Counts = 5
False Positive Counts = 1
True Negative Counts = 3
False Negative Counts = 1

Example: Confusion Matrix For Binary Classification

		Predicted	
		Dog	Not Dog
Actual	Dog	True Positive (TP =5)	False Negative (FN =1)
	Not Dog	False Positive (FP=1)	True Negative (TN=3)

Code

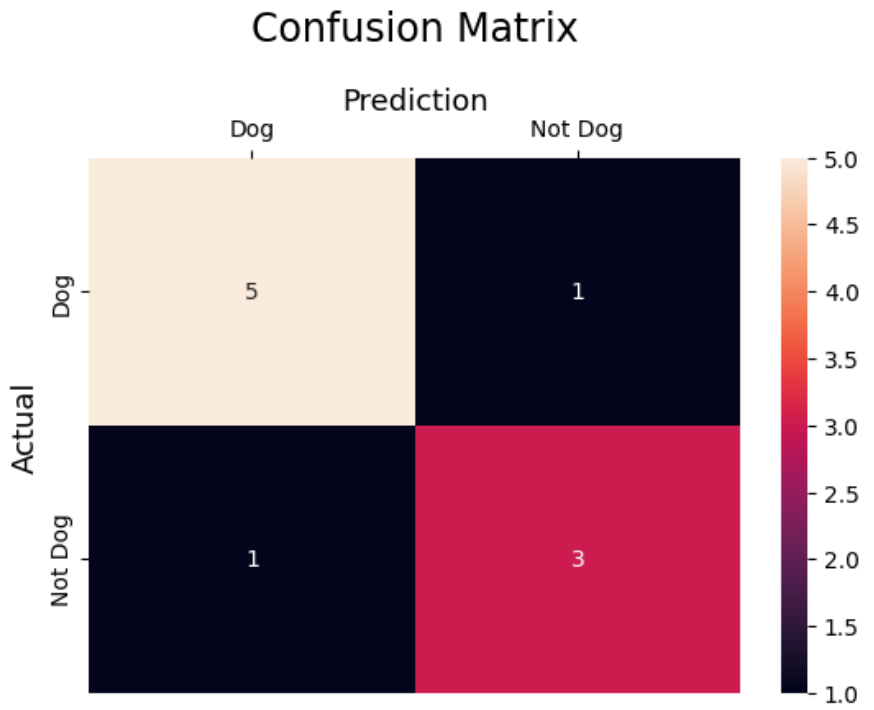
```
import numpy as np
from sklearn.metrics import confusion_matrix,classification_report
import seaborn as sns
import matplotlib.pyplot as plt

#Create the NumPy array for actual and predicted labels
actual    = np.array(['Dog','Dog','Dog','Not Dog','Dog','Not Dog','Dog','Dog','Not Dog','Not Dog’])
predicted = np.array(['Dog','Not Dog','Dog','Not Dog','Dog','Dog','Dog','Dog','Not Dog','Not Dog’])

#Compute the confusion matrix
cm = confusion_matrix(actual,predicted)
```

Code

#Plot the confusion matrix with the help of the seaborn heatmap
sns.heatmap(cm, annot=**True**, fmt='g', xticklabels=['Dog','Not Dog'], yticklabels=['Dog','Not Dog'])

plt.ylabel('Actual', fontsize=13)
plt.title('Confusion Matrix', fontsize=17, pad=20)
plt.gca().xaxis.set_label_position('top')
plt.xlabel('Prediction', fontsize=13)
plt.gca().xaxis.tick_top()

plt.gca().figure.subplots_adjust(bottom=0.2)
plt.gca().figure.text(0.5, 0.05, 'Prediction', ha='center', fontsize=13)
plt.show()



Code

#Classifications Report based on Confusion Metrics
print(classification_report(actual, predicted))

	precision	recall	f1-score	support
Dog	0.83	0.83	0.83	6
Not Dog	0.75	0.75	0.75	4
accuracy			0.80	10
macro avg	0.79	0.79	0.79	10
weighted avg	0.80	0.80	0.80	10

Class: Dog
Support = 6 → There are 6 actual Dog images.
Precision = 0.83 → Out of all predictions labeled Dog, 83% were correct. (1 out of 6 was misclassified as Dog when it was actually Not Dog).
Recall = 0.83 → Out of all 6 actual Dogs, 83% were correctly identified. (1 Dog was missed).
F1 = 0.83 → Balanced measure between precision and recall.

Example: Confusion Matrix for Image Classification (Cat, Dog, Horse)

	Predicted Cat	**Predicted Dog**	**Predicted Horse**
Actual Cat	True Positive (TP) 8	False Negative (FN) 1	False Negative (FN) 1
Actual Dog	False Negative (FN) 2	True Positive (TP) 10	False Negative (FN) 0
Actual Horse	False Negative (FN) 0	False Negative (FN) 2	True Positive (TP) 8

Cats: 8 were correctly identified, 1 was misidentified as a dog and 1 was misidentified as a horse.

Dogs: 10 were correctly identified, 2 were misidentified as cats.

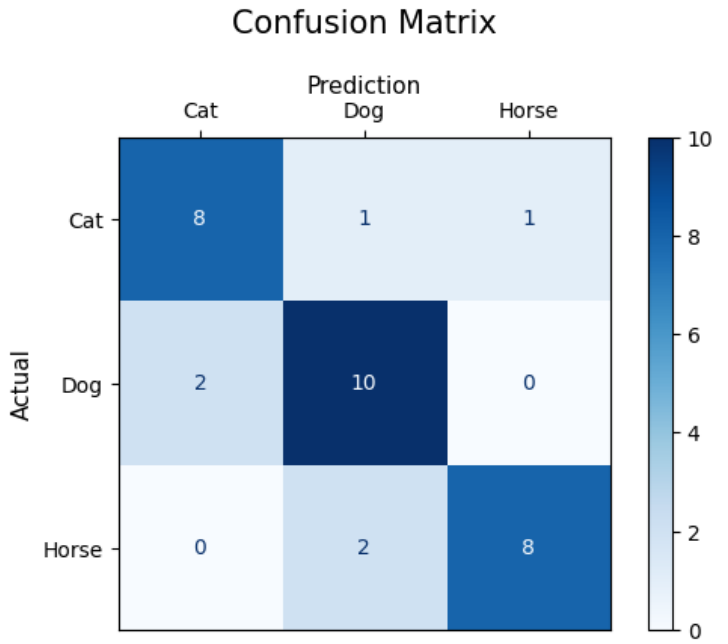
Horses: 8 were correctly identified, 2 were misidentified as dogs.

Code

```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

#Create the NumPy array for actual and predicted labels
y_true = ['Cat'] * 10 + ['Dog'] * 12 + ['Horse'] * 10
y_pred = ['Cat'] * 8 + ['Dog'] + ['Horse'] + ['Cat'] * 2 + ['Dog'] * 10 + ['Horse'] * 8 + ['Dog'] * 2
classes = ['Cat', 'Dog', 'Horse']
```

Code

```
#Generate and Visualize the Confusion Matrix
cm = confusion_matrix(y_true, y_pred, labels=classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix', fontsize=15, pad=20)
plt.xlabel('Prediction', fontsize=11)
plt.ylabel('Actual', fontsize=11)
plt.gca().xaxis.set_label_position('top')
plt.gca().xaxis.tick_top()
plt.gca().figure.subplots_adjust(bottom=0.2)
plt.gca().figure.text(0.5, 0.05, 'Prediction', ha='center', fontsize=13)

plt.show()
```



Code

#Print the Classification Report
print(classification_report(y_true, y_pred, target_names=classes))

	precision	recall	f1-score	support
Cat	0.80	0.80	0.80	10
Dog	0.77	0.83	0.80	12
Horse	0.89	0.80	0.84	10
accuracy			0.81	32
macro avg	0.82	0.81	0.81	32
weighted avg	0.82	0.81	0.81	32

Class Activity

	Predicted Cat	Predicted Dog	Predicted Rabit	Predicted Horse	Predicted Elephant
Actual Cat	1	1	0	0	0
Actual Dog	0	2	0	0	0
Actual Rabit	0	0	2	0	0
Actual Horse	0	0	0	2	0
Actual Elephant	0	1	0	0	1

Manually compute solution:

1. Compute overall accuracy.
2. Per-class metrics (Cat, Dog, Rabbit, Horse, Elephant): Precision, Recall, F1 Score

Now use sklearn and compare the results.