Confusion Matrix Practical - 5

- Shivam Tiwari A-58

Aim: Write a python program to evaluate a confusion matrix from given dataset.

Theory:

A confusion matrix is a tabular summary of the number of correct and incorrect prediction made by a classifier.

It can be used to evaluate the performance of classification made through the calculation of performance metric like accuracy, precision, recall and F1 score.

A Good ML Model:

It depends but generally you'll evaluate your machine learning model based some predetermined metrics that you decide to use when it comes to building classification model you will most likely used. Confusion metric are not just useful in most evaluation but also made monitering and model ~~mang~~ management.

Example 1 :

Suppose that a classifier ~~proceedures~~ produce the following results :

Predicted Outcome	Actual Outcome
Class A	Class A
Class B	Class B
Class C	Class C
Class A	Class C
Class B	Class A

Below code to compute confusion matrix and classification report.

```
from sklearn import metrics

y_pred = ['a', 'b', 'c', 'a', 'b']
y_act = ['a', 'b', 'c', 'c', 'a']

print (metrics.confusion_matrix (y_act, y_pred,
                labels = ['a', 'b', 'c']))

print (metrics.classification_report (y_act, y_pred,
                labels = ['a', 'b', 'c']))
```

It calculates performance metric the precision, recall and support.

Positive → Observation is positive
Negative → Observation is negative
True Positive (TP) → Correctly predicted positive class
True Negative (TN) → Correctly predicted negative class
False Positive (FP) → Incorrectly predicted ~~posi~~ positive class
False Negative (FN) → Incorrectly predicted negative class.

Accuracy :

This is simply equal to the proportion of prediction that the model classified correctly.

$$\text{Accuracy} = \frac{\text{Correct Prediction}}{\text{Total Prediction}}$$

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

Precision :

Precision is also known as positive prediction value and is the proportion of relatives instances among the retrived instances.

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

Recall :

Recall also known as ressistivity hit rate the true positive rate (TPR) is the proporitial of total amount of relevent instances that were accurately retrived.

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Result}}$$

F1 Score :

The F1 score is a measure of test accuracy it is the harmonic mean of precision and recall. It can have a maximum score of 1 c perfect precision and ~~rec~~ recall and minimum of a overall it is measure of precision of precision and robustiness of our model.

$$\text{F1 score} = \frac{2 \times (\text{Precision} \times \text{recall})}{\text{Precision} + \text{recall}}$$

$$= \frac{2TP}{2TP + FP + FN}$$

Conclusion : Hence, we have successfully created python program to evaluate confusion metric.

Code and Output:

Practical 5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1]
```python
# Shivam Tawari A-58

# Importing the dependancies
from sklearn import metrics

# Predicted values
y_pred=["a", "b","c", "a", "d"]

# Actual values
y_act=["a","b","c","c","a"]
print(metrics.confusion_matrix(y_act,y_pred, labels=["a", "b", "c"]))

# Printing the precision and recall, among other metrics
print(metrics.classification_report(y_act,y_pred,labels=["a","b","c"]))
```

```
[[1 0 0]
 [0 1 0]
 [1 0 1]]
              precision    recall  f1-score   support

           a       0.50      0.50      0.50         2
           b       1.00      1.00      1.00         1
           c       1.00      0.50      0.67         2

   micro avg       0.75      0.60      0.67         5
   macro avg       0.83      0.67      0.72         5
weighted avg       0.80      0.60      0.67         5
```

[2]
```python
import pandas as pd
data = {'Y_Actual': [1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0],
        'y_Predicted': [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0]}
df = pd.DataFrame(data, columns=['Y_Actual', 'y_Predicted'])
print(df)
```

```
    Y_Actual  y_Predicted
0          1            1
1          0            1
2          0            0
3          1            1
4          0            0
5          1            1
6          0            1
7          0            0
8          1            1
9          0            0
10         1            0
11         0            0
```

[3]
```python
confusion_matrix = pd.crosstab (df['Y_Actual'], df['y_Predicted'],
rownames=['Actual'], colnames=['Predicted'])
print(confusion_matrix)
```

```
Predicted  0  1
Actual
0          5  2
1          1  4
```

```
[4] import pandas as pd
    import seaborn as sn
    import matplotlib.pyplot as plt
    sn.heatmap(confusion_matrix,annot=True)
    plt.show()
```