Tugas 2 - Performance Evaluation

IS184620 - Analitika Bisnis [A]
Muhammad Fathurrahman
5026201139

1. Email Dataset

Dalam kurun waktu 2 minggu terakhir, diambil sampel sebanyak 20 *email* yang masuk ke dalam kotak surat. Di antara sampel yang diambil, ada yang tergolong sebagai *spam* dan bukan *spam*.

Email yang mencurigakan entah baik dari isinya maupun dari pengirimnya akan secara otomatis digolongkan sebagai *spam email* oleh sistem. Begitupun sebaliknya untuk *email* yang bukan *spam*.

2. Classification Table (Actual & Predicted)

Berdasarkan *email* yang masuk, dibuat tabel klasifikasi untuk mengkategorikan *email* yang sesuai dan tidak sesuai dengan kategori yang sudah digolongkan oleh sistem.

No	Actual	Predicted
1	*Spam*	*Not Spam*
2	*Not Spam*	*Spam*
3	*Not Spam*	*Not Spam*
4	*Not Spam*	*Not Spam*
5	*Not Spam*	*Spam*
6	*Spam*	*Spam*
7	*Spam*	*Spam*
8	*Spam*	*Not Spam*
9	*Not Spam*	*Spam*
10	*Not Spam*	*Spam*

No	Actual	Predicted
11	*Spam*	*Spam*
12	*Not Spam*	*Not Spam*
13	*Spam*	*Not Spam*
14	*Spam*	*Spam*
15	*Not Spam*	*Not Spam*
16	*Not Spam*	*Spam*
17	*Not Spam*	*Not Spam*
18	*Not Spam*	*Spam*
19	*Spam*	*Not Spam*
20	*Spam*	*Not Spam*

3. Confusion Classification (TP, TN, FP, FN)

Berdasarkan tabel klasifikasi, dapat dihitung berapa nilai dari **TP**, **TN**, **FP**, dan **FN**.

True Positive : *Spam* yang digolongkan *spam* (**Actual true**, **Predicted true**)
True Negative : *Not spam* yang digolongkan *not spam* (**Actual false**, **Predicted false**)
False Positive : *Not spam* yang digolongkan *spam* (**Actual false**, **Predicted true**)
False Negative : *Spam* yang digolongkan *not spam* (**Actual true**, **Predicted false**)

Apabila penghitungan dilakukan menggunakan *python* akan seperti ini:

```python
#!/usr/bin/python

actual_email = [1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1]
predicted_email = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]

# Cari Confusion Matrix (TP, TN, FP, FN)
def confusion_matrix(actual, predicted):
    TP, TN, FP, FN = 0, 0, 0, 0

    for i in range(len(actual)):
        if actual[i] == 1 and predicted[i] == 1:
            TP += 1
        elif actual[i] == 0 and predicted[i] == 0:
            TN += 1
        elif actual[i] == 0 and predicted[i] == 1:
            FP += 1
        elif actual[i] == 1 and predicted[i] == 0:
            FN += 1

    return TP, TN, FP, FN

TP, TN, FP, FN = confusion_matrix(actual_email, predicted_email)

print("TP: {}, TN: {}, FP: {}, FN: {}".format(TP, TN, FP, FN))
```

Output dari kode tersebut akan memberikan nilai berikut:

```
TP: 4, TN: 5, FP: 6, FN: 5
```

4. Confusion Matrix

		Predicted		
		True	**False**	
Actual	**True**	4	5	9
	False	6	5	11
		10	10	20

5. Performance Evaluation (Own Function)

Berdasarkan data pada *confusion matrix*, dapat dihitung berapa nilai dari *accuracy*, *precision*, *recall*, dan *f-measure* menggunakan *python* seperti ini:

```
# Cari Accuracy
def accuracy(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return (TP + TN) / (TP + TN + FP + FN)

print("Accuracy: {0:.3g}".format(accuracy(actual_email, predicted_email)))

# Cari Precision
def precision(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return TP / (TP + FP)

print("Precision: {0:.3g}".format(precision(actual_email, predicted_email)))

# Cari Recall
def recall(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return TP / (TP + FN)

print("Recall: {0:.3g}".format(recall(actual_email, predicted_email)))

# Cari F-Measure
def f_measure(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return 2 * (precision(actual, predicted) * recall(actual, predicted)) /
(precision(actual, predicted) + recall(actual, predicted))

print("F-Measure: {0:.3g}".format(f_measure(actual_email, predicted_email)))
```

Output dari kode tersebut akan memberikan nilai berikut:

```
Accuracy: 0.45
Precision: 0.4
Recall: 0.444
F-Measure: 0.421
```

6. Performance Evaluation (Built-In Function)

Penghitungan nilai dari *accuracy*, *precision*, *recall*, dan *f-measure* dapat menjadi lebih ringkas dengan menggunakan *built-in function* dari *scikit-learn* (*classification_report*).

```python
#!/usr/bin/python

from sklearn.metrics import classification_report

actual_email = [1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1]
predicted_email = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]

# Accuracy, Precision, Recall, dan F-Measure menggunakan scikit-learn
print(classification_report(actual_email, predicted_email))
```

Output dari kode tersebut akan memberikan nilai berikut:

```
              precision    recall  f1-score   support

           0       0.50      0.45      0.48        11
           1       0.40      0.44      0.42         9

    accuracy                           0.45        20
   macro avg       0.45      0.45      0.45        20
weighted avg       0.45      0.45      0.45        20
```

7. Balanced or Imbalanced Dataset

Dataset email pada studi kasus ini termasuk *dataset* yang *imbalanced*, hal ini dikarenakan perbandingan varietas sampel, *true* dengan *false*, tidak sebanding (**9:11**).

Dataset yang *imbalanced* sangat mempengaruhi performa dari mesin, salah satunya pada nilai *accuracy* yang kecil (**0.45**) (Pradeep Kumar et al 2021).

8. Evaluation

Pada studi kasus ini, didapati bahwa nilai *precision* dari model ke-2 (*spam email* diasumsikan sebagai positif) memiliki nilai terendah yaitu **0.40**.

Hal ini dapat terjadi dikarenakan banyaknya jumlah *email* yang diklasifikasikan sebagai *False Positive* (*not spam email* digolongkan *spam*) diikuti dengan sedikitnya jumlah *email* yang diklasifikasikan sebagai *True Positive* (*spam email* digolongkan *spam*).

Source Code

```python
#!/usr/bin/python

from sklearn.metrics import classification_report

actual_email = [1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1]
predicted_email = [0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0]

# Cari Confusion Matrix (TP, TN, FP, FN)
def confusion_matrix(actual, predicted):
    TP, TN, FP, FN = 0, 0, 0, 0
    for i in range(len(actual)):
        if actual[i] == 1 and predicted[i] == 1:
            TP += 1
        elif actual[i] == 0 and predicted[i] == 0:
            TN += 1
        elif actual[i] == 0 and predicted[i] == 1:
            FP += 1
        elif actual[i] == 1 and predicted[i] == 0:
            FN += 1
    return TP, TN, FP, FN

TP, TN, FP, FN = confusion_matrix(actual_email, predicted_email)

print("TP: {}\nTN: {}\nFP: {}\nFN: {}\n".format(TP, TN, FP, FN))

# Cari Accuracy
def accuracy(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return (TP + TN) / (TP + TN + FP + FN)

print("Accuracy: {0:.3g}".format(accuracy(actual_email, predicted_email)))

# Cari Precision
def precision(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return TP / (TP + FP)

print("Precision: {0:.3g}".format(precision(actual_email, predicted_email)))

# Cari Recall
def recall(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return TP / (TP + FN)

print("Recall: {0:.3g}".format(recall(actual_email, predicted_email)))

# Cari F-Measure
def f_measure(actual, predicted):
    TP, TN, FP, FN = confusion_matrix(actual, predicted)
    return 2 * (precision(actual, predicted) * recall(actual, predicted)) /
(precision(actual, predicted) + recall(actual, predicted))

print("F-Measure: {0:.3g}\n".format(f_measure(actual_email, predicted_email)))

# Accuracy, Precision, Recall, dan F-Measure menggunakan scikit-learn
print(classification_report(actual_email, predicted_email))
```