MODUL 2. Measuring Data Similarity and Dissimilarity

I. Tujuan Praktikum

1. Memahami konsep similarity dan dissimilarity pada data.
2. Mempelajari beberapa metode untuk mengukur similarity dan dissimilarity.
3. Menggunakan Python untuk menghitung similarity dan dissimilarity pada dataset.

II. Teori

Pada analisis data, pengukuran **similarity** (kesamaan) dan **dissimilarity** (perbedaan) antara objek data sangat penting, terutama dalam tugas seperti clustering, rekomendasi, dan pencarian informasi. Ukuran similarity menunjukkan seberapa mirip dua objek data, sedangkan ukuran dissimilarity menunjukkan seberapa berbeda dua objek tersebut.

2.1 **Similarity**: Pengukuran ini menentukan kesamaan antara dua objek. Nilai similarity biasanya berada pada skala 0 hingga 1, di mana 1 berarti sangat mirip dan 0 berarti tidak mirip sama sekali. Ada dua sifat atribut *binary* yang dipakai pada *similarity,* yaitu simetris dan asimetri.

a. **Binary Symmetric Similarity** dipakai untuk menghitung kemiripan untuk data biner tanpa membedakan apakah nilai tersebut adalah 0 atau 1. Digunakan jika kedua nilai sama-sama penting. Contohnya **Simple Matching Coeeficient.**
Menghitung Jarak untuk variabel biner simetris dapat menggunakan rumus

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- **q**: Jumlah atribut di mana kedua objek memiliki nilai 1 (kedua vektor bernilai 1).
- **r**: Jumlah atribut di mana objek pertama memiliki nilai 1 dan objek kedua memiliki nilai 0.
- **s**: Jumlah atribut di mana objek pertama memiliki nilai 0 dan objek kedua memiliki nilai 1.
- **t**: Jumlah atribut di mana kedua objek memiliki nilai 0 (kedua vektor bernilai 0).

b. **Binary Asymmetric Similarity** dipakai disaat hanya memperhitungkan fitur bernilai 1 (positif) dan mengabaikan fitur bernilai 0. Pendekatan ini lebih relevan jika 1 (keberadaan fitur) lebih penting daripada 0 (ketiadaan fitur). Contohnya adalah **Jaccard Coefficient**.
Menghitung jarak untuk variabel biner asimetris dapat menggunakan rumus

$$d(i,j) = \frac{r+s}{q+r+s}$$

- **q**: Jumlah atribut di mana kedua objek memiliki nilai 1 (kedua vektor bernilai 1).
- **r**: Jumlah atribut di mana objek pertama memiliki nilai 1 dan objek kedua memiliki nilai 0.
- **s**: Jumlah atribut di mana objek pertama memiliki nilai 0 dan objek kedua memiliki nilai 1.
- **t**: Jumlah atribut di mana kedua objek memiliki nilai 0 (diabaikan dalam ukuran ini).

Beberapa metric pada similary, yaitu :

1) **Simple Matching Coefficient (SMC)**
   SMC menghitung proporsi dari nilai yang sama antara dua objek di seluruh fitur biner, tanpa membedakan apakah keduanya sama-sama bernilai 0 atau 1.

$$\text{SMC} = \frac{(a+d)}{(a+b+c+d)}$$

   Dimana:

   - **a**: Jumlah atribut di mana kedua objek bernilai 1.
   - **d**: Jumlah atribut di mana kedua objek bernilai 0.
   - **b**: Jumlah atribut di mana objek pertama bernilai 1 dan objek kedua bernilai 0.
   - **c**: Jumlah atribut di mana objek pertama bernilai 0 dan objek kedua bernilai 1.

```python
import numpy as np
from sklearn.metrics import jaccard_score

# Dataset biner
data = np.array([
    [1, 0, 1, 1, 0],  # Objek A
    [1, 1, 0, 1, 1]   # Objek B
])

# Menghitung Simple Matching Coefficient (SMC)
def simple_matching_coefficient(x, y):
    a = np.sum(np.logical_and(x == 1, y == 1))  # Jumlah atribut di mana kedua objek bernilai 1
    d = np.sum(np.logical_and(x == 0, y == 0))  # Jumlah atribut di mana kedua objek bernilai 0
    b = np.sum(np.logical_and(x == 1, y == 0))  # Jumlah atribut di mana x bernilai 1 dan y bernilai 0
    c = np.sum(np.logical_and(x == 0, y == 1))  # Jumlah atribut di mana x bernilai 0 dan y bernilai 1
    smc = (a + d) / (a + b + c + d)  # Rumus SMC
    return smc

# Menghitung SMC antara Objek A dan Objek B
smc_value = simple_matching_coefficient(data[0], data[1])
print(f"Simple Matching Coefficient (SMC) antara Objek A dan Objek B: {smc_value}")
```

2) **Cosine Similarity**: Mengukur sudut kosinus antara dua vektor.

$$\text{Cosine Similarity} = \frac{A \cdot B}{||A|| \cdot ||B||}$$

```python
# Membuat dua vektor
vector_1 = np.array([2, 1, 0, 2, 0, 1, 1, 0, 0, 0])
vector_2 = np.array([2, 1, 1, 2, 0, 1, 0, 0, 1, 0])

# Menghitung Cosine Similarity
cos_sim = cosine_similarity([vector_1], [vector_2])
print(f'Cosine Similarity: {cos_sim[0][0]}')
```

```
Cosine Similarity: 0.8703882797784892
```

3) **Jaccard Similarity**: Mengukur similarity antara dua set data berdasarkan elemen yang dimiliki keduanya.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

```
[13] # Membuat dua set data
     set_1 = np.array([0, 1, 1, 0, 1, 1, 0, 0])
     set_2 = np.array([1, 1, 0, 0, 1, 0, 1, 1])

     # Menghitung Jaccard Distance
     jaccard_dist = distance.jaccard(set_1, set_2)
     jaccard_sim = 1 - jaccard_dist
     print(f'Jaccard Similarity: {jaccard_sim}')
```

```
Jaccard Similarity: 0.2857142857142857
```

2.2 **Dissimilarity**: Dissimilarity mengukur seberapa jauh atau berbedanya dua objek. Semakin besar nilai dissimilarity, semakin tidak mirip dua objek tersebut. Metrik yang sering digunakan :

1) **Euclidean Distance**: Mengukur jarak lurus antara dua titik pada ruang fitur.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

```
# Membuat dua vektor
point_1 = np.array([2, 3])
point_2 = np.array([6, 7])

# Menghitung Euclidean Distance
euclidean_dist = np.linalg.norm(point_1 - point_2)
print(f'Euclidean Distance: {euclidean_dist}')
```

```
Euclidean Distance: 5.656854249492381
```

2) **Manhattan Distance**: Jarak total dari satu titik ke titik lain mengikuti garis lurus (jarak blok kota).

$$d(p,q) = \sum_{i=1}^{n}|p_i - q_i|$$

```
# Menghitung Manhattan Distance
manhattan_dist = distance.cityblock(point_1, point_2)
print(f'Manhattan Distance: {manhattan_dist}')
```

```
Manhattan Distance: 8
```

III. Praktikum

Cobalah Pemrograman berikut:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances
from scipy.spatial import distance

# Generate 100 data points for 2 numerical features (random numbers between 0 and 10)
np.random.seed(42)
data_numerik = np.random.randint(0, 10, size=(100, 2))

# Generate 100 binary data points for the binary feature (0 or 1)
data_biner = np.random.randint(0, 2, size=(100, 1))

# Combine numerical and binary data into one dataset
data = np.hstack((data_numerik, data_biner))
df = pd.DataFrame(data, columns=['Feature_1', 'Feature_2', 'Binary_Feature'])

# Display the first 5 rows of the dataset
print("Dataset with 100 data points:")
print(df.head())

# 1. Euclidean Distance Calculation
euclidean_matrix = euclidean_distances(data_numerik)
print("\nEuclidean Distance Matrix (first 5 rows):\n", euclidean_matrix[:5, :5])

# Visualizing Euclidean Distance Matrix using heatmap
sns.heatmap(euclidean_matrix[:10, :10], annot=True, cmap="Blues")
plt.title("Euclidean Distance Matrix (Subset)")
plt.show()
```

```python
# 2. Cosine Similarity Calculation
cosine_matrix = cosine_similarity(data_numerik)
print("\nCosine Similarity Matrix (first 5 rows):\n", cosine_matrix[:5, :5])

# Visualizing Cosine Similarity Matrix using heatmap
sns.heatmap(cosine_matrix[:10, :10], annot=True, cmap="Greens")
plt.title("Cosine Similarity Matrix (Subset)")
plt.show()

# 3. Jaccard Similarity Calculation (using only binary feature)
jaccard_matrix = np.zeros((len(data_biner), len(data_biner)))
for i in range(len(data_biner)):
    for j in range(len(data_biner)):
        jaccard_matrix[i, j] = 1 - distance.jaccard(data_biner[i], data_biner[j])

print("\nJaccard Similarity Matrix (first 5 rows):\n", jaccard_matrix[:5, :5])

# Visualizing Jaccard Similarity Matrix using heatmap
sns.heatmap(jaccard_matrix[:10, :10], annot=True, cmap="Oranges")
plt.title("Jaccard Similarity Matrix (Subset)")
plt.show()
```

IV. Penutup

Dengan modul praktikum ini, Anda diharapkan dapat memahami dan menerapkan metode pengukuran similarity dan dissimilarity pada dataset nyata menggunakan Google Colab. Selamat belajar dan bereksperimen!