LAPORAN

RENCANA TUGAS MAHASISWA (RTM) Ke-5

MATA KULIAH DATA MINING

"UNSUPERVISED LEARNING"

DISUSUN OLEH:

Rahmah Lidya Nastiti (22083010102)

DOSEN PENGAMPU:

Kartika Maulida Hindrayani, S.Kom., M.Kom. (NIP. 199209092022032009)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2024

UNSUPERVISED LEARNING

Dataset hepatitis (hanya menggunakan variabel prediktor, tanpa menggunakan variabel dependen) dibangun pemodelan datanya menggunakan task klasteriasi yakni metode K-means termasuk mengevaluasi model menggunakan Silhouette Coefficient.

```
[1]: pip install ucimlrepo

Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.0.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Instal ‘pip install ucimlrepo’ untuk menginstal paket python untuk ucimlrepo dimana paket ini merupakan kumpulan utilitas atau fungsi yang berkaitan dengan akses dan pengelolaan dataset dari UC Irvine Machine Learning Repository (UCI ML Repo).

```
15]: from ucimlrepo import fetch_ucirepo
     import pandas as pd

     hepatitis = fetch_ucirepo(id=46)

     data = hepatitis.data.features
     data.head()
```

15]:

	Age	Sex	Steroid	Antivirals	Fatigue	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Spiders	Ascites	Varices	Bilirubin	Alk Phosphate	Sgot	Albumin	Protime	
0	30	2	1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0	NaN	
1	50	1	1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5	NaN	
2	78	1	2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0	NaN	
3	31	1	NaN	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0	80.0	
4	34	1	2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	NaN	200.0	4.0	NaN	

Pertama mengimpor fungsi ‘fetch_ucirepo’ dari modul ucimlrepo dan pandas. Selanjutnya mengambil dataset Hepatitis menggunakan fungsi ‘fetch_ucirepo’ dan menetapkannya ke variabel hepatitis. Kemudian mengekstrak atribut fitur dari atribut data objek Bunch hepatitis dan menampilkan 5 baris pertama dari fitur DataFrame.

```
16]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155 entries, 0 to 154
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              155 non-null    int64
 1   Sex              155 non-null    int64
 2   Steroid          154 non-null    float64
 3   Antivirals       155 non-null    int64
 4   Fatigue          154 non-null    float64
 5   Malaise          154 non-null    float64
 6   Anorexia         154 non-null    float64
 7   Liver Big        145 non-null    float64
 8   Liver Firm       144 non-null    float64
 9   Spleen Palpable  150 non-null    float64
 10  Spiders          150 non-null    float64
 11  Ascites          150 non-null    float64
 12  Varices          150 non-null    float64
 13  Bilirubin        149 non-null    float64
 14  Alk Phosphate    126 non-null    float64
 15  Sgot             151 non-null    float64
 16  Albumin          139 non-null    float64
 17  Protime          88 non-null     float64
 18  Histology        155 non-null    int64
dtypes: float64(15), int64(4)
memory usage: 23.1 KB
```

Dari hasil diatas untuk menampilkan informasi tentang struktur dataset tersebut, termasuk jumlah entri sebanyak 155, tipe data dari setiap kolom yaitu float64 dan int64, jumlah nilai non-null, dan jumlah kolom sebanyak 19.

```
7]: data.isnull().sum()
```

```
7]: Age                0
    Sex                0
    Steroid            1
    Antivirals         0
    Fatigue            1
    Malaise            1
    Anorexia           1
    Liver Big         10
    Liver Firm        11
    Spleen Palpable    5
    Spiders            5
    Ascites            5
    Varices            5
    Bilirubin          6
    Alk Phosphate     29
    Sgot               4
    Albumin           16
    Protime           67
    Histology          0
    dtype: int64
```

Hasil diatas merupakan jumlah nilai yang hilang (NULL) di setiap kolom dataset.

```
]: data=data.fillna(data.median())
   data
```

]:

	Age	Sex	Steroid	Antivirals	Fatigue	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Spiders	Ascites	Varices	Bilirubin	Alk Phosphate	Sgot	Albumin	Protime
0	30	2	1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0	61.0
1	50	1	1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5	61.0
2	78	1	2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0	61.0
3	31	1	2.0	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0	80.0
4	34	1	2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	200.0	4.0	61.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
150	46	1	2.0	2	1.0	1.0	1.0	2.0	2.0	2.0	1.0	1.0	1.0	7.6	85.0	242.0	3.3	50.0
151	44	1	2.0	2	1.0	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	0.9	126.0	142.0	4.3	61.0
152	61	1	1.0	2	1.0	1.0	2.0	1.0	1.0	2.0	1.0	2.0	2.0	0.8	75.0	20.0	4.1	61.0
153	53	2	1.0	2	1.0	2.0	2.0	2.0	2.0	1.0	1.0	2.0	1.0	1.5	81.0	19.0	4.1	48.0
154	43	1	2.0	2	1.0	2.0	2.0	2.0	2.0	1.0	1.0	1.0	2.0	1.2	100.0	19.0	3.1	42.0

155 rows × 19 columns

Kode diatas untuk mengisi nilai yang hilang dalam DataFrame ‘data’ dengan nilai median dari setiap kolom.

```
]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155 entries, 0 to 154
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              155 non-null    int64
 1   Sex              155 non-null    int64
 2   Steroid          155 non-null    float64
 3   Antivirals       155 non-null    int64
 4   Fatigue          155 non-null    float64
 5   Malaise          155 non-null    float64
 6   Anorexia         155 non-null    float64
 7   Liver Big        155 non-null    float64
 8   Liver Firm       155 non-null    float64
 9   Spleen Palpable  155 non-null    float64
 10  Spiders          155 non-null    float64
 11  Ascites          155 non-null    float64
 12  Varices          155 non-null    float64
 13  Bilirubin        155 non-null    float64
 14  Alk Phosphate    155 non-null    float64
 15  Sgot             155 non-null    float64
 16  Albumin          155 non-null    float64
 17  Protime          155 non-null    float64
 18  Histology        155 non-null    int64
dtypes: float64(15), int64(4)
memory usage: 23.1 KB
```

Dari hasil diatas untuk menampilkan informasi tentang struktur dataset yang telah diperbaiki.

```
]: import numpy as np
import matplotlib.pyplot as plt

dataset = data

X = dataset.values

class KMeans:
    def __init__(self, n_clusters, max_iter=300):
        self.n_clusters = n_clusters
        self.max_iter = max_iter

    def fit(self, X):
        n_samples, n_features = X.shape

        # Inisialisasi centroid secara acak
        self.centroids = X[np.random.choice(n_samples, self.n_clusters, replace=False)]

        # Iterasi untuk mengoptimalkan centroid
        for _ in range(self.max_iter):
            # Menghitung jarak setiap titik data ke setiap centroid
            distances = np.sqrt(((X - self.centroids[:, np.newaxis])**2).sum(axis=2))

            # Memperbarui label cluster untuk setiap titik data
            labels = np.argmin(distances, axis=0)

            # Memperbarui centroid dengan rata-rata titik data dalam setiap cluster
            new_centroids = np.array([X[labels == k].mean(axis=0) for k in range(self.n_clusters)])

            # Jika tidak ada perubahan dalam centroid, keluar dari iterasi
            if np.allclose(self.centroids, new_centroids):
                break

            self.centroids = new_centroids

        # Menghitung inertia (WCSS: Within-Cluster-Sum-of-Squares)
        self.inertia = 0
        for k in range(self.n_clusters):
            cluster_points = X[labels == k]
            self.inertia += np.sum((cluster_points - self.centroids[k])**2)
```

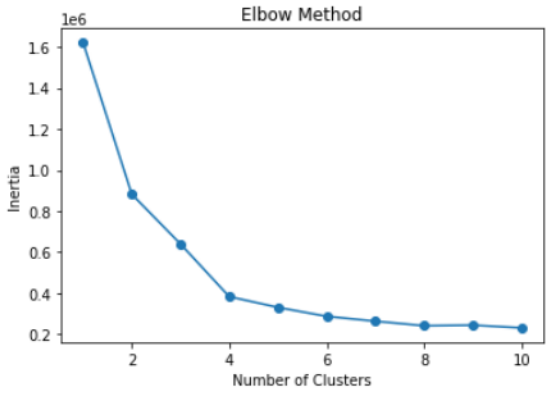
```
        return self.inertia

# Menentukan jumlah cluster optimal menggunakan metode Elbow
def find_optimal_clusters(X, max_clusters):
    inertias = []
    for n_clusters in range(1, max_clusters+1):
        kmeans = KMeans(n_clusters=n_clusters)
        inertia = kmeans.fit(X)
        inertias.append(inertia)

    return inertias

# Contoh penggunaan:
if __name__ == "__main__":
    max_clusters = 10
    inertias = find_optimal_clusters(X, max_clusters)

    # Plot Elbow Method
    plt.plot(range(1, max_clusters+1), inertias, marker='o')
    plt.title('Elbow Method')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.show()
```



Dari hasil plot diatas dapat dijelaskan bahwa plot yang menunjukkan inersia untuk setiap jumlah cluster dari 1 sampai 10. Plot dimulai dengan penurunan inersia yang curam seiring dengan bertambahnya jumlah klaster, yang mengindikasikan bahwa model meningkatkan kecocokannya. Namun, ketika jumlah cluster terus meningkat, penurunan inersia menjadi kurang jelas, dan kurva mulai mendatar.

1]:
```python
import numpy as np
import pandas as pd

# Import dataset (ganti 'nama_file.csv' dengan nama file dataset Anda)
dataset = data

# Ubah dataset menjadi bentuk numpy array
X = dataset.values

class KMeans:
    def __init__(self, n_clusters, max_iter=300, random_state=None):
        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.random_state = random_state

    def fit(self, X):
        n_samples, n_features = X.shape

        # Inisialisasi seed acak (jika random_state diberikan)
        if self.random_state is not None:
            np.random.seed(self.random_state)

        # Inisialisasi centroid secara acak
        self.centroids = X[np.random.choice(n_samples, self.n_clusters, replace=False)]

        # Iterasi untuk mengoptimalkan centroid
        for _ in range(self.max_iter):
            # Menghitung jarak setiap titik data ke setiap centroid
            distances = np.sqrt(((X - self.centroids[:, np.newaxis])**2).sum(axis=2))

            # Memperbarui label cluster untuk setiap titik data
            labels = np.argmin(distances, axis=0)

            # Memperbarui centroid dengan rata-rata titik data dalam setiap cluster
            new_centroids = np.array([X[labels == k].mean(axis=0) for k in range(self.n_clusters)])
            # Jika tidak ada perubahan dalam centroid, keluar dari iterasi
            if np.allclose(self.centroids, new_centroids):
                break

            self.centroids = new_centroids
```

```python
    def predict(self, X):
        # Menghitung jarak setiap titik data ke setiap centroid
        distances = np.sqrt(((X - self.centroids[:, np.newaxis])**2).sum(axis=2))

        # Mengembalikan label cluster dengan jarak terdekat
        return np.argmin(distances, axis=0)

# Contoh penggunaan:
if __name__ == "__main__":
    # Inisialisasi dan latih model KMeans dengan n cluster
    kmeans = KMeans(n_clusters=4, random_state=42)
    kmeans.fit(X)

    # Prediksi label cluster untuk setiap titik data
    labels = kmeans.predict(X)
    print(labels)
```

```
[1 1 1 1 3 1 1 1 1 1 1 3 1 3 1 1 1 3 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 2
 2 1 2 1 1 1 1 1 1 1 0 1 1 1 1 3 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 3 3 1
 1 1 3 1 1 1 1 3 1 2 3 1 2 2 1 1 2 1 1 1 3 1 3 1 1 1 3 1 3 3 1 1 2 0 3 1 2
 1 1 1 2 3 1 2 1 1 3 2 1 3 1 1 1 2 1 1 1 1 3 1 2 3 1 3 2 1 2 1 1 1 1 3 1 0
 3 1 3 3 1 1 1]
```

Import dataset, kemudian dikonversi ke dalam bentuk numpy yang disebut ‘X’. kelas ‘KMean’ mengambil tiga parameter yaitu ‘n_clusters’ (jumlah cluster yang diinginkan), ‘max_iter ‘(jumlah maksimum iterasi untuk proses konvergensi), dan ‘random_state’. Kemudian metode fit(X) digunakan untuk melatih model K-Means pada data. Dengan cara Inisialisasi seed acak jika random_state diberikan dan Inisialisasi Centroid (pusat klaster) diinisialisasi secara acak, Iterasi dilakukan hingga mencapai jumlah maksimum iterasi yaitu dengan Jarak setiap titik data

ke setiap centroid dihitung menggunakan jarak Euclidean, setiap titik diberi label cluster berdasarkan centroid terdekat, centroid diperbarui dengan menghitung rata-rata titik data dalam setiap cluster, iterasi berhenti jika tidak ada perubahan signifikan dalam posisi centroid. Selanjutnya Metode prediksi digunakan untuk memprediksi label klaster untuk titik data baru. Metode ini menghitung jarak antara setiap titik data dan setiap centroid dan mengembalikan label dari centroid terdekat.

```
2]: # Tambahkan output array ke dalam DataFrame
data['Cluster'] = labels

data = pd.DataFrame(data)
data
```

2]:

	Age	Sex	Steroid	Antivirals	Fatigue	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Spiders	Ascites	Varices	Bilirubin	Alk Phosphate	Sgot	Albumin	Protime
0	30	2	1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0	61.0
1	50	1	1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5	61.0
2	78	1	2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0	61.0
3	31	1	2.0	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0	80.0
4	34	1	2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	200.0	4.0	61.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
150	46	1	2.0	2	1.0	1.0	1.0	2.0	2.0	2.0	1.0	1.0	1.0	7.6	85.0	242.0	3.3	50.0
151	44	1	2.0	2	1.0	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	0.9	126.0	142.0	4.3	61.0
152	61	1	1.0	2	1.0	1.0	2.0	1.0	1.0	2.0	1.0	2.0	2.0	0.8	75.0	20.0	4.1	61.0
153	53	2	1.0	2	1.0	2.0	2.0	2.0	2.0	1.0	1.0	2.0	1.0	1.5	81.0	19.0	4.1	48.0
154	43	1	2.0	2	1.0	2.0	2.0	2.0	2.0	1.0	1.0	1.0	2.0	1.2	100.0	19.0	3.1	42.0

155 rows × 20 columns

Kode tersebut bertujuan untuk menambahkan tentang cluster ke dalam DataFrame data yang berisi dataset hepatitis. Ini dilakukan dengan menambahkan kolom baru yang disebut 'Cluster' ke DataFrame ‘data’.

```
3]: import numpy as np

def silhouette_coefficient(data, labels):
    n_samples = len(data)
    silhouette_scores = []

    # Iterasi melalui setiap sampel
    for i in range(n_samples):
        # Ambil label cluster untuk sampel i
        cluster_label = labels.iloc[i]
        # Hitung a(i): rata-rata jarak dari sampel i ke sampel lain dalam cluster yang sama
        samples_in_cluster = data[labels == cluster_label]
        a_i = np.mean(np.linalg.norm(data.iloc[i] - samples_in_cluster, axis=1))

        # Hitung b(i): jarak rata-rata dari sampel i ke semua sampel dalam cluster lain terdekat
        other_clusters = np.unique(labels[labels != cluster_label])
        b_i_values = []
        for other_cluster in other_clusters:
            samples_in_other_cluster = data[labels == other_cluster]
            b_i_values.append(np.mean(np.linalg.norm(data.iloc[i] - samples_in_other_cluster, axis=1)))
        b_i = min(b_i_values)

        # Hitung Silhouette Coefficient untuk sampel i
        silhouette_coefficient_i = (b_i - a_i) / max(a_i, b_i)
        silhouette_scores.append(silhouette_coefficient_i)

    # Hitung Silhouette Coefficient rata-rata dari semua sampel
    silhouette_avg = np.mean(silhouette_scores)
    return silhouette_avg

# Data
data = data
labels = data.Cluster

# Hitung Silhouette Coefficient
silhouette_avg = silhouette_coefficient(data, labels)
print("Silhouette Coefficient:", silhouette_avg)
```

```
Silhouette Coefficient: 0.501726236435136
```

Pertama import numpy, kemudian Fungsi ‘silhouette_coefficient’ digunakan untuk menghitung nilai Silhouette Coefficient dari sebuah clustering. Selanjutnya jumlah sampel dalam data dihitung (n_samples). Untuk setiap sampel, hitung nilai a(i) yang merupakan rata-rata jarak dari sampel tersebut ke sampel lain dalam cluster yang sama dan hitung nilai b(i) yang merupakan jarak rata-rata dari sampel tersebut ke semua sampel dalam cluster lain terdekat. Setelah itu, dihitung Silhouette Coefficient untuk setiap sampel menggunakan rumus: (b(i) - a(i)) / max(a(i), b(i)). Nilai Silhouette Coefficient untuk setiap sampel dimasukkan ke dalam list silhouette_scores. Manggil fungsi silhouette_coefficient dengan parameter data dan labels. Hasilnya disimpan dalam variabel silhouette_avg. Kemudian print hasil Silhouette Coefficient rata-rata dicetak dan hasilnya adalah 0.501726236435136.