Syifa Wanda Isnaini

1103201248

UTS Machine Learning

Clustering Model K-Means

Berikan kodingan untuk import necessary libraries

# Importing necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split  # For splitting the data into training and testing sets

from sklearn.tree import DecisionTreeClassifier  # Decision tree classifier for the model

from sklearn.metrics import accuracy\_score, classification\_report  # For model evaluation

from sklearn.preprocessing import LabelEncoder  # To encode non-numeric data

from sklearn.cluster import KMeans  # Importing KMeans

from yellowbrick.cluster import KElbowVisualizer

from google.colab import drive  # For mounting Google Drive

Penjelasan Kode

NumPy (np): Perpustakaan untuk operasi numerik di Python. Ini menyediakan dukungan untuk array dan matriks besar, bersama dengan fungsi matematika untuk beroperasi pada elemen-elemen ini.

Pandas (pd): Perpustakaan untuk manipulasi dan analisis data. Ini menyediakan struktur data seperti DataFrame untuk menyimpan dan memanipulasi data terstruktur dengan efisien.

Matplotlib (plt): Perpustakaan plotting populer untuk membuat visualisasi statis, interaktif, dan beranimasi di Python.

Seaborn (sns): Dibangun di atas Matplotlib, Seaborn adalah perpustakaan visualisasi data statistik. Ini menyederhanakan proses membuat grafik statistik yang informatif dan menarik.

scikit-learn (sklearn): Perpustakaan pembelajaran mesin yang komprehensif. Ini mencakup alat untuk pra-pemrosesan data, pemilihan model, metrik kinerja, dan berbagai algoritma pembelajaran mesin.

Yellowbrick (yellowbrick): Perpustakaan visualisasi untuk scikit-learn. Ini memperluas fungsionalitas Matplotlib dan Seaborn untuk menyediakan alat visual untuk tugas pembelajaran mesin, termasuk evaluasi dan pemilihan model.

Google Colab (google.colab): Perpustakaan untuk berinteraksi dengan Google Colaboratory, platform cloud gratis untuk mengembangkan dan menjalankan kode Python secara kolaboratif.

Penjelasan running :

Pesan "Mounted at /content/drive" menunjukkan bahwa Google Drive Anda telah berhasil dimount (diakses) di lingkungan Google Colab pada direktori "/content/drive". Saat Anda menjalankan perintah drive.mount('/content/drive'), Colab meminta izin akses ke Google Drive Anda melalui antarmuka otorisasi.

Berikan kode untuk mounting google drive to access dataset

# Mounting Google Drive to access the dataset

drive.mount('/content/drive')

# Loading the dataset into a Pandas DataFrame

df = pd.read\_csv('/content/drive/MyDrive/College.csv')

Penjelasan Kode

1. Memasang Google Drive:

drive.mount('/content/drive')

Baris ini memasang Google Drive Anda di direktori '/content/drive' di lingkungan Colab.

Ini meminta Anda untuk membuka tautan, mengotorisasi Google Drive, dan menempelkan kode otorisasi yang dihasilkan kembali ke notebook Colab.

2. Memuat Dataset:

df = pd.read\_csv('/content/drive/MyDrive/College.csv')

Baris ini membaca file CSV bernama 'College.csv' dari direktori 'MyDrive' di Google Drive Anda.

Ini memuat data ke dalam Pandas DataFrame (df), memungkinkan Anda menganalisis dan memanipulasi dataset menggunakan fungsionalitas Pandas.

Yellowbrick (yellowbrick): Perpustakaan visualisasi untuk scikit-learn. Ini memperluas fungsionalitas Matplotlib dan Seaborn untuk menyediakan alat visual untuk tugas pembelajaran mesin, termasuk evaluasi dan pemilihan model.

Penjelasan running

DataFrame memiliki 777 baris dan 19 kolom.

Nama kolom, jumlah nilai non-null, dan tipe data kolom ditampilkan.

Statistik Deskriptif untuk Kolom Numerik:

Rata-rata, deviasi standar, nilai minimum, kuartil ke-1, median, kuartil ke-3, dan nilai maksimum ditampilkan untuk kolom numerik.

Jumlah non-null dan tipe data setiap kolom numerik juga ditampilkan.

Count: Jumlah data non-null.

Mean: Rata-rata.

Std: Deviasi standar.

Min: Nilai minimum.

25%, 50%, 75%: Kuartil pertama, kedua (median), dan ketiga.

Max: Nilai maksimum.

'Apps': Jumlah aplikasi yang diterima.

'Accept': Jumlah aplikasi yang diterima.

'Enroll': Jumlah pendaftar yang terdaftar.

'Top10perc': Persentase siswa dengan peringkat 10% teratas dari sekolah mereka.

Ringkasan statistik ini memberikan gambaran singkat tentang distribusi dan karakteristik dataset numerik. Informasi ini dapat membantu dalam pemahaman awal tentang sifat data di setiap kolom.

Berikan kode untuk EDA

# Exploratory Data Analysis (EDA)

# Displaying the first few rows of the dataset

df.head()

# Providing information about the dataset such as columns, data types, and non-null values

df.info()

# Statistical summary of the dataset

df.describe()

# Histogram for 'Outstate' with differentiation by 'Private'

sns.set\_style('darkgrid')

g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2)

# Exploratory Data Analysis (EDA)

# Displaying the first few rows of the dataset

df.head()

# Providing information about the dataset such as columns, data types, and non-null values

df.info()

# Statistical summary of the dataset

df.describe()

# Histogram for 'Outstate' with differentiation by 'Private'

sns.set\_style('darkgrid')

g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2)

Penjelasan kode :

1. Menampilkan Beberapa Baris Pertama dari Dataset:

df.head()

Baris ini menampilkan beberapa baris pertama dari dataset menggunakan fungsi head(). Ini memberikan pandangan cepat tentang struktur dan isi data.

1. Memberikan Informasi tentang Dataset:

df.info()

Baris ini memberikan informasi tentang dataset, termasuk jumlah nilai non-null, tipe data setiap kolom, dan penggunaan memori. Ini menggunakan metode info().

1. Ringkasan Statistik tentang Dataset:

df.describe()

Baris ini menghasilkan ringkasan statistik tentang dataset. Ini mencakup ukuran seperti mean, deviasi standar, minimum, persentil ke-25, median, persentil ke-75, dan maksimum untuk kolom numerik. Metode describe() digunakan untuk tujuan ini.

Langkah-langkah EDA ini membantu Anda memahami struktur, konten, dan karakteristik statistik dari dataset. Ini penting untuk mendapatkan wawasan tentang data sebelum melanjutkan dengan analisis atau pemodelan lebih lanjut.

Berikan kode untuk data visualisasi

# Data Visualization

# Scatter plot to visualize 'Room.Board' against 'Grad.Rate' with differentiation by 'Private'

sns.set\_style('whitegrid')

sns.lmplot(x='Room.Board', y='Grad.Rate', data=df, hue='Private',

           palette='coolwarm', height=6, aspect=1, fit\_reg=False)

sns.set\_style('whitegrid')  # Setting the visual style for the plot

# Creating a scatter plot using lmplot to visualize 'Room.Board' against 'Grad.Rate'

# Data is differentiated by the 'Private' column using a coolwarm color palette

# The plot is set to a height of 6, with an aspect ratio of 1, and without a regression line (fit\_reg=False)

sns.lmplot(x='Room.Board', y='Grad.Rate', data=df, hue='Private',

           palette='coolwarm', height=6, aspect=1, fit\_reg=False)

Penjelasan kode :

1. Ringkasan Statistik tentang Dataset:

df.describe()

Baris ini menghasilkan ringkasan statistik dari keseluruhan dataset. Ini mencakup ukuran seperti mean, deviasi standar, minimum, persentil ke-25, median, persentil ke-75, dan maksimum untuk kolom numerik.

1. Histogram untuk 'Outstate' dengan Diferensiasi berdasarkan 'Private':

sns.set\_style('darkgrid')

g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2)

g = g.map(plt.hist, 'Outstate', bins=20, alpha=0.7)

Baris-baris ini menggunakan Seaborn untuk membuat plot FacetGrid dengan histogram untuk kolom 'Outstate'.

Data diidentifikasi berdasarkan variabel 'Private', di mana warna yang berbeda menunjukkan nilai 'Private' yang berbeda.

Ini memberikan wawasan tentang distribusi variabel 'Outstate' untuk nilai 'Private' yang berbeda.

1. Histogram untuk 'Grad.Rate' dengan Diferensiasi berdasarkan 'Private':

sns.set\_style('darkgrid')

g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2)

g = g.map(plt.hist, 'Grad.Rate', bins=20, alpha=0.7)

Serupa dengan blok sebelumnya, baris-baris ini membuat plot FacetGrid dengan histogram, namun kali ini untuk kolom 'Grad.Rate'.

Data kembali diidentifikasi berdasarkan variabel 'Private', memungkinkan Anda mengamati distribusi 'Grad.Rate' untuk nilai 'Private' yang berbeda.

Visualisasi ini membantu memahami distribusi variabel numerik dan bagaimana distribusinya berbeda berdasarkan kategori 'Private'.

Penjelasan running :

Hasil scatter plot ini memberikan gambaran visual tentang sebaran titik-titik data untuk 'Room.Board' dan 'Grad.Rate', dengan pembedaan warna berdasarkan variabel 'Private'. Scatter plot seperti ini dapat membantu dalam melihat pola atau perbedaan antar kategori dalam dataset.

Berikan kode untuk cleaning data

# Data Cleaning and Preparation

# Handling 'Grad.Rate' values greater than 100

df[df['Grad.Rate'] > 100]

df['Grad.Rate']['Cazenovia College'] = 100

df[df['Grad.Rate'] > 100]

Penjelasan kode :

Kode tersebut digunakan untuk menemukan baris-baris dalam DataFrame (df) di mana nilai pada kolom 'Grad.Rate' lebih besar dari 100. Ini berguna untuk mengeksplorasi dan menangani nilai yang tidak valid atau tidak sesuai dalam dataset. Dalam konteks data pendidikan atau tingkat kelulusan (sepertinya berdasarkan nama kolom 'Grad.Rate'), seharusnya tidak mungkin memiliki tingkat kelulusan lebih dari 100%. Oleh karena itu, baris-baris dengan nilai 'Grad.Rate' lebih besar dari 100% mungkin merupakan kesalahan input atau data yang tidak valid.

Penjelasan running :

Hasil running tersebut menyertakan pesan peringatan SettingWithCopyWarning yang mengindikasikan bahwa ada percobaan untuk mengubah nilai dalam DataFrame yang merupakan hasil slicing. Pesan ini muncul ketika mencoba mengubah nilai pada DataFrame yang dihasilkan dari operasi slicing (pemotongan) pada DataFrame yang asli.

Secara spesifik, baris ini mencoba mengubah nilai pada kolom 'Grad.Rate' untuk baris yang memiliki nilai 'Cazenovia College' pada kolom 'Unnamed: 0'. Tindakan ini menyebabkan peringatan karena slicing pada DataFrame (df['Grad.Rate']['Cazenovia College']) dapat menghasilkan tampilan (view) DataFrame yang terkait dengan DataFrame asli, dan perubahan pada tampilan tersebut dapat mempengaruhi DataFrame asli.

Untuk menghindari peringatan ini dan memastikan perubahan dilakukan pada DataFrame yang diinginkan, lebih baik menggunakan metode .loc[] atau .iloc[].

Berikan kode untuk visualisasi hostrogram dengan menggunakan seaborn

sns.set\_style('darkgrid')  # Setting the style for the plot

# Creating a FacetGrid plot with differentiation by 'Private' and using a coolwarm color palette

g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2)

# Mapping a histogram onto the FacetGrid for the 'Grad.Rate' column with 20 bins and partial transparency (alpha=0.7)

g = g.map(plt.hist, 'Grad.Rate', bins=20, alpha=0.7)

Penjelasan kode :

1. sns.set\_style('darkgrid'): Baris ini mengatur gaya plot menjadi 'darkgrid'. Seaborn menyediakan beberapa gaya yang dapat Anda atur untuk plot, seperti 'darkgrid', 'whitegrid', 'dark', 'white', dan lain-lain.

2. g = sns.FacetGrid(df, hue="Private", palette='coolwarm', height=6, aspect=2): Baris ini membuat objek FacetGrid dari seaborn, yang memungkinkan Anda membuat subplot berdasarkan variabel kategorikal (dalam hal ini, 'Private'). Parameter lainnya termasuk palet warna ('coolwarm'), tinggi plot (height), dan aspek rasio (aspect).

3. g = g.map(plt.hist, 'Grad.Rate', bins=20, alpha=0.7): Baris ini memetakan histogram ke dalam objek FacetGrid. Ini menggunakan metode .map() untuk memetakan fungsi plot, dalam hal ini, fungsi histogram (plt.hist dari Matplotlib) ke dalam setiap subplot berdasarkan 'Private'. Parameter lainnya termasuk jumlah bins (20) dan transparansi sebagian (alpha=0.7) untuk memberikan efek visual.

Penjelasan running :

Setiap subplot pada grid menampilkan distribusi 'Grad.Rate' untuk kategori 'Private' yang sesuai.

Warna berbeda pada histogram mencerminkan kategori 'Private'.

Grid dan histogram membantu melihat perbedaan dalam distribusi 'Grad.Rate' antara kategori 'Private'.

FacetGrid sangat berguna untuk membandingkan distribusi variabel numerik pada beberapa kelompok atau kategori secara bersamaan.

Berikan kode untuk mengkonversi kolom private dalam data menjadi kolom cluster yang memiliki nilai numerik

# Converting 'Private' column to numerical values (1 for 'Yes' and 0 for 'No')

def converter(cluster):

    if cluster=='Yes':

        return 1

    else:

        return 0

df['Cluster'] = df['Private'].apply(converter)

Penjelasan kode :

Kode yang Anda berikan adalah fungsi Python bernama converter yang mengubah nilai dalam kolom 'Private' dari kategori "Yes" menjadi 1 dan "No" menjadi 0. Fungsi ini dapat digunakan dengan menggunakan metode apply dari Pandas untuk mengaplikasikan transformasi ini pada seluruh kolom 'Private' dalam DataFrame.

Penjelasan running :

Akurasi (Accuracy):

Akurasi model adalah 0.8333 atau 83.33%.

Akurasi mengukur sejauh mana model mampu mengklasifikasikan data dengan benar secara keseluruhan.

Dalam konteks ini, 83.33% dari seluruh data uji diklasifikasikan dengan benar oleh model.

Classification Report:

Precision: Mengukur sejauh mana kelas yang diidentifikasi oleh model benar (positif) dari semua prediksi positif.

Recall: Mengukur sejauh mana model berhasil mengidentifikasi semua instance yang seharusnya termasuk dalam suatu kelas.

F1-score: Rata-rata harmonis antara precision dan recall.

Support: Jumlah instance aktual dalam setiap kelas.

Precision Recall F1-Score Support

0 0.70 0.79 0.74 47

1 0.90 0.85 0.88 109

Accuracy 0.83 156

Interpretasi Classification Report:

Untuk kelas '0' (mungkin representasi non-private), precision sebesar 0.70, recall sebesar 0.79, dan F1-score sebesar 0.74.

Untuk kelas '1' (mungkin representasi private), precision sebesar 0.90, recall sebesar 0.85, dan F1-score sebesar 0.88.

Akurasi weighted avg (rata-rata tertimbang) adalah 0.84.

Kesimpulan:

Model memiliki kinerja yang baik dalam mengklasifikasikan kelas '1' (representasi private), dibuktikan dengan precision, recall, dan F1-score yang tinggi untuk kelas ini.

Meskipun kelas '0' (representasi non-private) memiliki performa yang lebih rendah, model secara keseluruhan memiliki akurasi yang baik.

Catatan:

Performa model dapat berbeda tergantung pada konteks aplikasi dan preferensi terhadap precision atau recall.

Dalam kasus ini, interpretasi dan evaluasi lebih lanjut bergantung pada kebutuhan spesifik tugas klasifikasi.

Berikan kode untuk mentraining dan mengevaluasi

# Exclude non-numeric columns or encode them

non\_numeric\_cols = df.select\_dtypes(include=['object']).columns.tolist()

# Drop non-numeric columns (you can also encode them if they are categorical)

X = df.drop(non\_numeric\_cols + ['Cluster'], axis=1)

y = df['Cluster']

# Encode non-numeric target labels

label\_encoder = LabelEncoder()

y = label\_encoder.fit\_transform(y)

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Model Training

model = DecisionTreeClassifier()

model.fit(X\_train, y\_train)

# Model Prediction and Evaluation

predictions = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, predictions)

report = classification\_report(y\_test, predictions)

print("Accuracy:", accuracy)

print("Classification Report:\n", report)

Penjelasan kode :

1. non\_numeric\_cols = df.select\_dtypes(include=['object']).columns.tolist(): Mengidentifikasi kolom-kolom dalam DataFrame (df) yang memiliki tipe data objek (biasanya kategorikal). Kolom-kolom ini kemudian akan diabaikan atau diencode, tergantung pada kebutuhan.

2. X = df.drop(non\_numeric\_cols + ['Cluster'], axis=1): Membuang kolom-kolom non-numerik dan kolom target ('Cluster') dari dataset untuk menyiapkan fitur (features) untuk model.

3. y = df['Cluster']: Menentukan target yang ingin diprediksi.

4. label\_encoder = LabelEncoder(): Membuat objek LabelEncoder dari scikit-learn untuk mengkodekan label kategori menjadi nilai numerik.

5. y = label\_encoder.fit\_transform(y): Menggunakan LabelEncoder untuk mengkodekan label target ('Cluster') menjadi nilai numerik. Ini diperlukan karena model-machine learning pada umumnya memerlukan input yang berupa nilai numerik.

6. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42): Membagi dataset menjadi data latih (train) dan data uji (test) untuk melatih dan mengevaluasi model.

7. model = DecisionTreeClassifier(): Membuat model klasifikasi pohon keputusan sebagai contoh. Anda dapat menggantinya dengan model lain yang sesuai dengan tugas Anda.

8. model.fit(X\_train, y\_train): Melatih model menggunakan data latih.

9. predictions = model.predict(X\_test): Membuat prediksi menggunakan data uji.

10. accuracy = accuracy\_score(y\_test, predictions): Mengukur akurasi model.

11. report = classification\_report(y\_test, predictions): Menghasilkan laporan klasifikasi yang melibatkan precision, recall, dan f1-score.

Menampilkan hasil akurasi dan laporan klasifikasi.

berikan kode untuk menampilkan metode elbow

# Create KMeans instance

model = KMeans()

# Using the elbow method to find the optimal number of clusters

visualizer = KElbowVisualizer(model, k=(2,10))  # Trying clusters from 2 to 10

visualizer.fit(X)  # Fit the data to the visualizer

visualizer.show()  # Display the Elbow plot

Penjelasan kode :

1. Membuat Instance KMeans:

model = KMeans()

Baris ini membuat sebuah instance dari model pengelompokan KMeans.

1. Menggunakan Metode Elbow:

visualizer = KElbowVisualizer(model, k=(2,10)) # Mencoba clusters dari 2 hingga 10

Kode ini menginisialisasi KElbowVisualizer dengan model KMeans dan menentukan rentang jumlah cluster yang akan dievaluasi (dari 2 hingga 10 dalam hal ini).

1. Melatih Data pada Visualizer:

visualizer.fit(X)

Baris ini melatih data pada visualizer, di mana X mewakili set fitur atau dataset Anda.

1. Menampilkan Plot Elbow:

visualizer.show()

Terakhir, kode ini menampilkan plot elbow, yang membantu mengidentifikasi secara visual jumlah cluster optimal. Titik "elbow" adalah di mana inersia (jumlah kuadrat dalam cluster) mulai berkurang pada tingkat yang lebih lambat.

Metode elbow adalah pendekatan heuristik untuk menemukan jumlah cluster yang sesuai. Ini membantu memilih nilai K yang seimbang antara kompleksitas model dan kemampuan untuk menangkap struktur dasar dari data.

Penjelasan hasil running :

Jumlah klaster yang optimal biasanya terletak di siku pada plot elbow.

Inersia adalah suatu metrik yang mengukur sejauh mana titik-titik dalam klaster. Penurunan inersia menandakan bahwa klaster-klaster semakin kompak.

Pilih jumlah klaster di siku untuk mendapatkan keseimbangan antara kompaknya klaster dan jumlah klaster yang tidak terlalu banyak.

Melalui visualisasi ini, Anda dapat memilih jumlah klaster yang optimal untuk penggunaan lebih lanjut dalam analisis atau pemodelan.

berikan kode untuk menerapkan algoritma K-Means clustering

# Creating a KMeans model with, for instance, 4 clusters

kmeans = KMeans(n\_clusters=4)  # Considering 4 clusters, as previously used for 'Private'

# Fit KMeans to the feature set

kmeans.fit(X)

# Retrieving the cluster labels assigned to each data point

cluster\_labels = kmeans.labels\_

# Displaying the cluster labels

print("Cluster Labels:", cluster\_labels)

Penjelasan kode :

1. Membuat Model KMeans dengan 4 Cluster:

kmeans = KMeans(n\_clusters=4) # Menggunakan 4 cluster, seperti yang sebelumnya digunakan untuk 'Private'

Baris ini membuat model pengelompokan KMeans dengan jumlah cluster yang ditentukan, dalam hal ini, 4. Parameter n\_clusters menentukan berapa banyak cluster yang algoritma akan mencoba temukan.

1. KMeans pada Set Fitur:

kmeans.fit(X)

Metode fit() digunakan untuk menerapkan algoritma KMeans pada set fitur (X). Proses ini melibatkan pencarian pusat cluster dan penugasan titik data ke dalam cluster.

1. Mengambil Label Cluster:

cluster\_labels = kmeans.labels\_

Setelah melatih model, atribut labels\_ digunakan untuk mengambil label cluster yang ditugaskan untuk setiap titik data. Setiap label mewakili cluster ke mana sebuah titik data termasuk.

1. Menampilkan Label Cluster:

print("Cluster Labels:", cluster\_labels)

Baris ini mencetak label cluster yang diperoleh ke konsol.

Output yang dihasilkan akan berupa daftar label cluster yang sesuai dengan setiap titik data dalam set fitur. Label ini menunjukkan penugasan cluster seperti yang ditentukan oleh algoritma KMeans.

Penjelasan hasil running :

Interpretasi Plot Elbow:

Pada plot elbow, mencari di mana ada "siku" atau "elbow" dapat membantu menentukan jumlah klaster optimal.

Pada titik siku, penurunan inersia mulai melambat, dan itu bisa menjadi pilihan jumlah klaster yang baik.

Catatan:

Dalam kasus ini, jumlah klaster optimal tidak ditampilkan dalam cuplikan yang diberikan. Anda dapat mengamati plot elbow dan memilih jumlah klaster yang paling sesuai dengan karakteristik data.

Jumlah klaster yang optimal dapat digunakan dalam pemodelan lanjutan atau analisis lebih lanjut tergantung pada tujuan Anda.

Berikan kode untuk visualisasi hasil clustering

# Perform KMeans clustering (previous code)

# Visualize clustering results

plt.scatter(X['Room.Board'], X['Grad.Rate'], c=cluster\_labels, cmap='viridis')  # Plotting Room.Board vs Grad.Rate

plt.title('KMeans Clustering')

plt.xlabel('Room & Board')

plt.ylabel('Graduation Rate')

plt.show()

Penjelasan kode :

Kode yang diberikan melanjutkan proses setelah pengelompokan KMeans dengan memvisualisasikan hasil pengelompokan. Ini menggunakan kolom 'Room.Board' dan 'Grad.Rate' dari set fitur (X) dan memberi warna pada titik-titik berdasarkan label cluster.

Melakukan Pengelompokan KMeans (Kode Sebelumnya):

Komentar ini menunjukkan bahwa kode sebelumnya termasuk langkah-langkah untuk melakukan pengelompokan KMeans, yang melibatkan pembuatan model KMeans, melatihnya pada set fitur (X), dan mendapatkan label cluster.

Memvisualisasikan Hasil Pengelompokan:

Baris-baris ini membuat plot scatter untuk memvisualisasikan hasil pengelompokan.

Sumbu x mewakili kolom 'Room & Board', sumbu y mewakili kolom 'Graduation Rate'.

Titik-titik diberi warna berdasarkan label cluster (c=cluster\_labels), menggunakan peta warna 'viridis'.

Plot kemudian diberi judul, dan sumbu diberi label sesuai.

Terakhir, plt.show() menampilkan plot tersebut.

Visualisasi ini memungkinkan Anda mengamati bagaimana titik-titik data terdistribusi dalam ruang fitur berdasarkan hasil pengelompokan.

Penjelasan hasil running :

Menggunakan metode elbow untuk menentukan jumlah optimal dari cluster.

Membuat model KMeans dengan jumlah cluster tertentu (misalnya, 4 cluster).

Melatih model KMeans pada fitur set.

Memvisualisasikan hasil clustering dengan scatter plot 'Room.Board' vs 'Grad.Rate'.

Kesimpulan Umum:

Data telah dijelajahi dan dibersihkan dengan beberapa visualisasi untuk pemahaman awal.

Sebuah model Decision Tree telah dibuat dan dievaluasi untuk klasifikasi.

Selain itu, dilakukan clustering menggunakan algoritma KMeans untuk memahami pola grup dalam data.

Namun, pada penggunaan KMeans, hasil visualisasi cluster yang dihasilkan mungkin belum memberikan pemahaman yang mendalam tentang pola di dalam data. Evaluasi lebih lanjut dan interpretasi tergantung pada kebutuhan analisis yang lebih spesifik.