

# DATA MINING

Dosen Pengampu : Dr. Rakhmat Arianto, S.ST., M.Kom



Nama : Nazwa Nurul Wijaya  
NIM : 2341760045  
Kelas : SIB-2F  
No.Absen : 22

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
2025**

## Dataset MovieLens 100k

Link: <https://grouplens.org/datasets/movielens/100k/>

### Jawab

GoogleCollab: [https://colab.research.google.com/drive/1XTdcxMzY9yzg8WVJTeNyaWQY1LABIgJh#scrollTo=nKzTpTAxP\\_en](https://colab.research.google.com/drive/1XTdcxMzY9yzg8WVJTeNyaWQY1LABIgJh#scrollTo=nKzTpTAxP_en)

Langkah	Jawaban/Deskripsi
1	Upload dahulu file zip yang sudah diunduh pada link dataset
2	<p>Menginstal scikit-surprise untuk sistem rekomendasi, lalu mengimpor pustaka seperti numpy, pandas, dan sklearn untuk analisis data dan evaluasi. Nah ini berfungsi untuk regresi, klasifikasi, clustering, visualisasi serta menghilangkan peringatan yang tidak perlu (yang kodenya ada di bawah sendiri)</p> <div><p>▼ Kategori 2: Dataset MovieLens 100k</p><pre>[ ] # Install library     !pip install scikit-surprise  # Import library 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 from sklearn.linear_model import LinearRegression from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score from sklearn.preprocessing import LabelEncoder, StandardScaler from sklearn.ensemble import RandomForestClassifier from sklearn.svm import SVC from sklearn.tree import DecisionTreeClassifier from sklearn.neighbors import KNeighborsClassifier from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_score from sklearn.cluster import KMeans from sklearn.decomposition import PCA from sklearn.metrics import silhouette_score import warnings warnings.filterwarnings('ignore')</pre></div> <p>Output</p> <div><pre>collecting scikit-surprise Downloading scikit_surprise-1.1.4.tar.gz (154 kB) 154.4/154.4 kB 3.4 MB/s eta 0:00:00  Installing build dependencies ... done Getting requirements to build wheel ... done Preparing metadata (pyproject.toml) ... done Requirement already satisfied: joblib&gt;=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-surprise) (1.5.1) Requirement already satisfied: numpy&gt;=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-surprise) (2.0.2) Requirement already satisfied: scipy&gt;=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-surprise) (1.15.3) Building wheels for collected packages: scikit-surprise   Building wheel for scikit-surprise (pyproject.toml) ... done   Created wheel for scikit-surprise: filename=scikit_surprise-1.1.4-cp311-cp311-linux_x86_64.whl size=2469550 sha256=67de4b245c553556f56ff4f4f36c   Stored in directory: /root/.cache/pip/wheels/2a/8f/6e/7e2899163e2d85d8266daab4aa1cdabec7a6c56f83c015b5af Successfully built scikit-surprise Installing collected packages: scikit-surprise Successfully installed scikit-surprise-1.1.4</pre></div>
3	<p>Selanjutnya saya unzip karena file yang saya upload langsung ke local nya adalah zip. Setelah itu ngeload data user, rating, film dan genre yang saya gabungkan ke dalam movies_genres dan menghapus kolom yang tidak saya butuhkan</p>

```

# Unzip dataset yang sudah saya upload
!unzip -q "/content/ml-100k.zip" -d "/content/movie_data"

# Ngeload data
u_cols = ['user_id', 'age', 'sex', 'occupation', 'zip_code']
users = pd.read_csv('/content/movie_data/ml-100k/u.user', sep='|', names=u_cols, encoding='latin-1')
r_cols = ['user_id', 'movie_id', 'rating', 'unix timestamp']
ratings = pd.read_csv('/content/movie_data/ml-100k/u.data', sep='\t', names=r_cols, encoding='latin-1')
m_cols = ['movie_id', 'title', 'release_date', 'video_release_date', 'imdb_url']
movies = pd.read_csv('/content/movie_data/ml-100k/u.item', sep='|', names=m_cols, usecols=range(5), encoding='latin-1')
genres = pd.read_csv('/content/movie_data/ml-100k/u.genre', sep='|', names=['genre', 'genre_id'], encoding='latin-1')
genre_list = genres['genre'].tolist()

# Saya gabungkan genre ke data film
movies_genres = pd.read_csv('/content/movie_data/ml-100k/u.item', sep='|', names=m_cols + genre_list, encoding='latin-1')
movies_genres.drop(columns=['video_release_date', 'imdb_url'], inplace=True)

```

Untuk mengetahui data EDA dengan melihat distribusi datanya

```

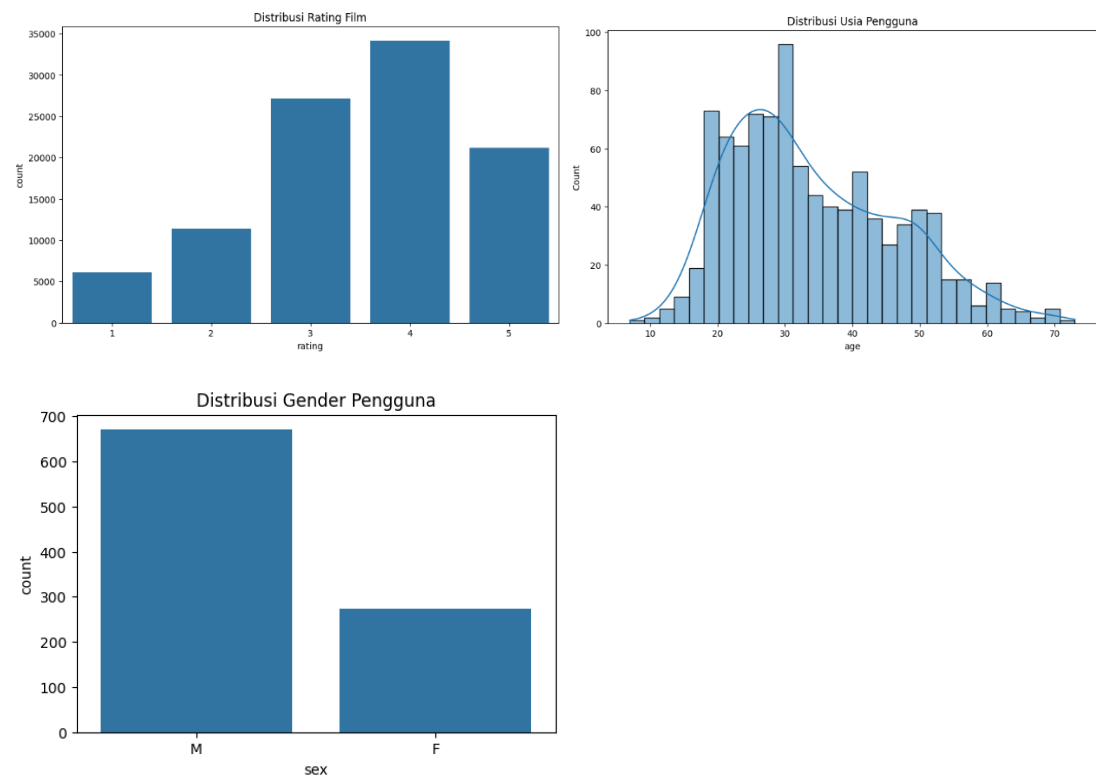
# Distribusi rating
plt.figure(figsize=(10, 6))
sns.countplot(x='rating', data=ratings)
plt.title("Distribusi Rating Film")
plt.show()

# Distribusi usia pengguna
plt.figure(figsize=(10, 6))
sns.histplot(users['age'], bins=30, kde=True)
plt.title("Distribusi Usia Pengguna")
plt.show()

# Distribusi gender
plt.figure(figsize=(6, 4))
sns.countplot(x='sex', data=users)
plt.title("Distribusi Gender Pengguna")
plt.show()

```

Ouput



Dari hasil nya dapat kita interpretasikan bawah:

1. Pada distribusi rating film mayoritas film diberi rating 3 dan 4 yang menunjukkan preferensi pengguna condong ke rating positif

	<p>2. Pada distribusi usia pengguna Sebagian besar pengguna berada di rentang usia 20–35 tahun. Kurva miring ke kanan (positively skewed), artinya makin sedikit pengguna di usia tua.</p> <p>3. Pada distribusi gender pengguna pengguna laki-laki (M) jauh lebih banyak daripada perempuan (F)</p>
--	--

### REGRASI: Prediksi rating film berdasarkan genre dan user profile

Langkah	Jawaban/Deskripsi
1	Pada kode ini, saya mencoba untuk memprediksi rating yang diberikan oleh pengguna terhadap sebuah film berdasarkan genre film dan profil pengguna. Model yang digunakan adalah Linear Regression, yang bertujuan untuk menemukan hubungan linier antara fitur seperti genre dan profil pengguna dengan rating yang diberikan oleh pengguna.
2	<p>Preprocessing data dimana saya menggabungkan dahulu prediksi diantara user dengan genre dari movie lalu saya lakukan cleaning dimana saya hanya menggunakan kolom yang sesuai dengan kebutuhan saya. Lalu saya lakukan encoding data kategorial yaitu sex dan occupation diubah menjadi data numerik menggunakan LabelEncoder agar model dapat memprosesnya. Setelah itu Scalling untuk menstandarisasi fitur fitur yang memiliki skala berbeda</p> <pre> # Saya Gabungkan dahulu prediksi antara user dengan genre dari movie merged_data = ratings.merge(users, on='user_id').merge(movies_genres, on='movie_id')  # Saya mendrop kolom non numerik dan tidak relevan cols_to_drop = [     'movie_id', 'title', 'user_id', 'release_date',     'zip_code', 'unix_timestamp',     'video_release_date', 'imdb_url' ]  # Cleaning cols_to_drop = [col for col in cols_to_drop if col in merged_data.columns] X = merged_data.drop(columns=cols_to_drop + ['rating']) y = merged_data['rating'] print("\nTipe data sebelum encoding dan scaling:") print(X.dtypes)  # Encode kolom sex dan occupation menggunakan LabelEncoder label_encoder = LabelEncoder() X['sex'] = label_encoder.fit_transform(X['sex']) X['occupation'] = label_encoder.fit_transform(X['occupation'])  # Scalling data scaler = StandardScaler() X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) </pre> <p>Output</p>

```
Tipe data sebelum encoding dan scaling:
age          int64
sex          object
occupation  object
unknown      int64
Action       int64
Adventure    int64
Animation    int64
Children's   int64
Comedy       int64
Crime        int64
Documentary  int64
Drama        int64
Fantasy      int64
Film Noir    int64
Horror       int64
Musical      int64
Mystery      int64
Romance      int64
Sci-Fi       int64
Thriller     int64
War          int64
Western      int64
dtype: object
```

Training Model Linear Regression lalu membuat prediksi yang hasilnya dibandingkan dengan rating aktual pada proses evaluasi model dengan menggunakan 3 etrik yaitu MSE (Mengukur seberapa besar kesalahan kuadrat rata-rata antara nilai prediksi dan nilai aktual), MAE (Mengukur rata-rata selisih absolut antara prediksi dan nilai aktual) dan  $R^2$  (Mengukur seberapa baik variabel independen menjelaskan variabilitas data. Nilai yang lebih tinggi menunjukkan model yang lebih baik). Setelah itu adalah visualisasi agar lebih mudah dibaca dan dipahami

```
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Membuat prediksi
y_pred = model.predict(X_test_scaled)

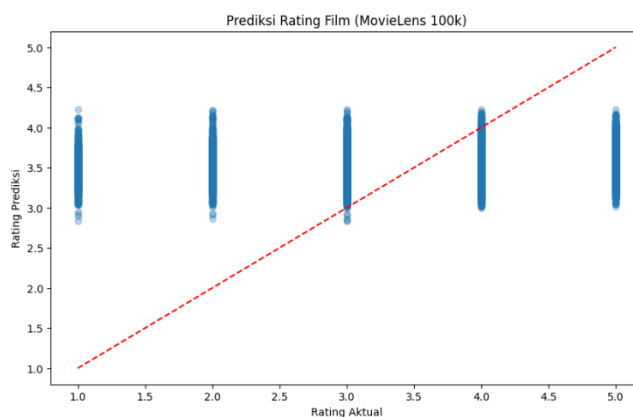
# Evaluasi model
print("\n=== HASIL REGRESI ===")
print(f"MSE: {mean_squared_error(y_test, y_pred):.4f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred):.4f}")
print(f"R²: {r2_score(y_test, y_pred):.4f}")

# VISUALISASI
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.scatter(y_test, y_pred, alpha=0.3)
plt.plot([1, 5], [1, 5], 'r--') # Garis ideal
plt.xlabel('Rating Aktual')
plt.ylabel('Rating Prediksi')
plt.title('Prediksi Rating Film (MovieLens 100k)')
plt.show()
```

Output

```
=== HASIL REGRESI ===
MSE: 1.2254
MAE: 0.9113
R²: 0.0298
```



## KLASIFIKASI: Prediksi apakah user suka film (>3.5)

Langkah	Jawaban/Deskripsi
1	Klasifikasi untuk memprediksi apakah seorang pengguna suka dengan film yang diberi rating lebih dari 3.5 (liked = 1) atau tidak (liked = 0). Model yang digunakan adalah <i>SVM (Support Vector Machine)</i> , <i>Decision Tree</i> , dan <i>KNN (K-Nearest Neighbors)</i> .
2	<p>Melakukan pemilihan target, cleansing dengan menghapus kolom yang tidak saya butuhkan lalu scaling</p> <pre> # Targetnya merged_data['liked'] = (merged_data['rating'] &gt; 3.5).astype(int)  cols_to_drop = ['rating', 'movie_id', 'title', 'user_id', 'release_date', 'zip_code', 'unix_timestamp'] X = merged_data.drop(columns=[col for col in cols_to_drop if col in merged_data.columns]) + ['liked'] y = merged_data['liked']  # Encode if 'sex' in X.columns:     X['sex'] = LabelEncoder().fit_transform(X['sex']) if 'occupation' in X.columns:     X['occupation'] = LabelEncoder().fit_transform(X['occupation'])  # Scaling scaler = StandardScaler() X_scaled = scaler.fit_transform(X)  # Split data buat training dan testing X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y) </pre>
3	<p>Modeling, evaluasi dan visualisasi</p> <pre> # A. SVM svm = SVC(kernel='rbf', random_state=42) svm.fit(X_train, y_train) y_pred_svm = svm.predict(X_test)  # B. Decision Tree dt = DecisionTreeClassifier(max_depth=3, random_state=42) dt.fit(X_train, y_train) y_pred_dt = dt.predict(X_test)  # C. KNN dengan tuning k=3-15 f1_scores = [] accuracy_scores = [] for k in range(3, 16):     knn = KNeighborsClassifier(n_neighbors=k)     knn.fit(X_train, y_train)     y_pred_knn = knn.predict(X_test)      # Hitung F1-Score dan Akurasi     f1_scores.append(f1_score(y_test, y_pred_knn))     accuracy_scores.append(accuracy_score(y_test, y_pred_knn))  # Hasil evaluasi results = {     'SVM': {'Accuracy': accuracy_score(y_test, y_pred_svm), 'F1': f1_score(y_test, y_pred_svm)},     'Decision Tree': {'Accuracy': accuracy_score(y_test, y_pred_dt), 'F1': f1_score(y_test, y_pred_dt)},     'KNN (k=best k)': {'Accuracy': accuracy_score(y_test, y_pred_knn), 'F1': f1_score(y_test, y_pred_knn)} }  # Uji nilai k dari 3 hingga 15 dan simpan hasilnya f1_scores = [] accuracy_scores = []  for k in range(3, 16):     knn = KNeighborsClassifier(n_neighbors=k)     knn.fit(X_train, y_train)     y_pred_knn = knn.predict(X_test)      f1_scores.append(f1_score(y_test, y_pred_knn))     accuracy_scores.append(accuracy_score(y_test, y_pred_knn))  # Menambahkan hasil KNN untuk setiap k ke dalam dictionary results for k in range(3, 16):     results[f'KNN (k={k})'] = {         'Accuracy': accuracy_scores[k-3],         'F1': f1_scores[k-3]     } </pre>

```

# Tampilkan hasil
print("\n== HASIL KLASIFIKASI ==")
display(pd.DataFrame(results).T)

# Confusion Matrix
fig, axes = plt.subplots(1, 3, figsize=(10,5))
models = [('SVM', y_pred_svm), ('Decision Tree', y_pred_dt), ('KNN (k=best k)', y_pred_knn)]
colors = ['Blues', 'Greens', 'Reds']

for (name, pred), ax, color in zip(models, axes, colors):
    cm = confusion_matrix(y_test, pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap=color, ax=ax)
    ax.set_title(name)
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Actual')

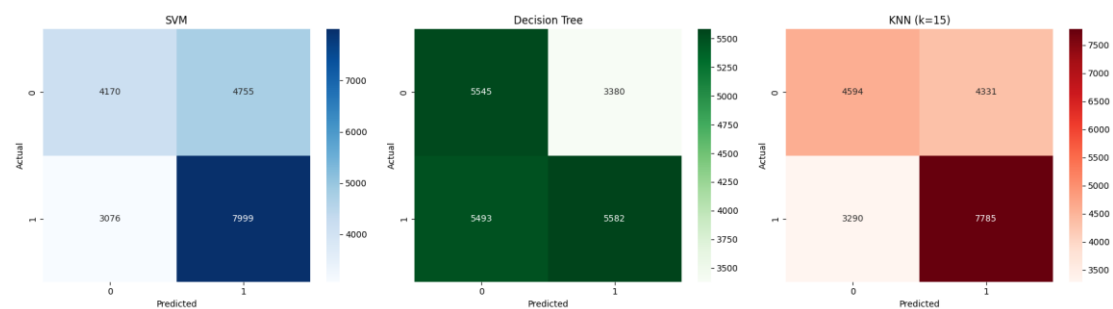
plt.tight_layout()
plt.show()

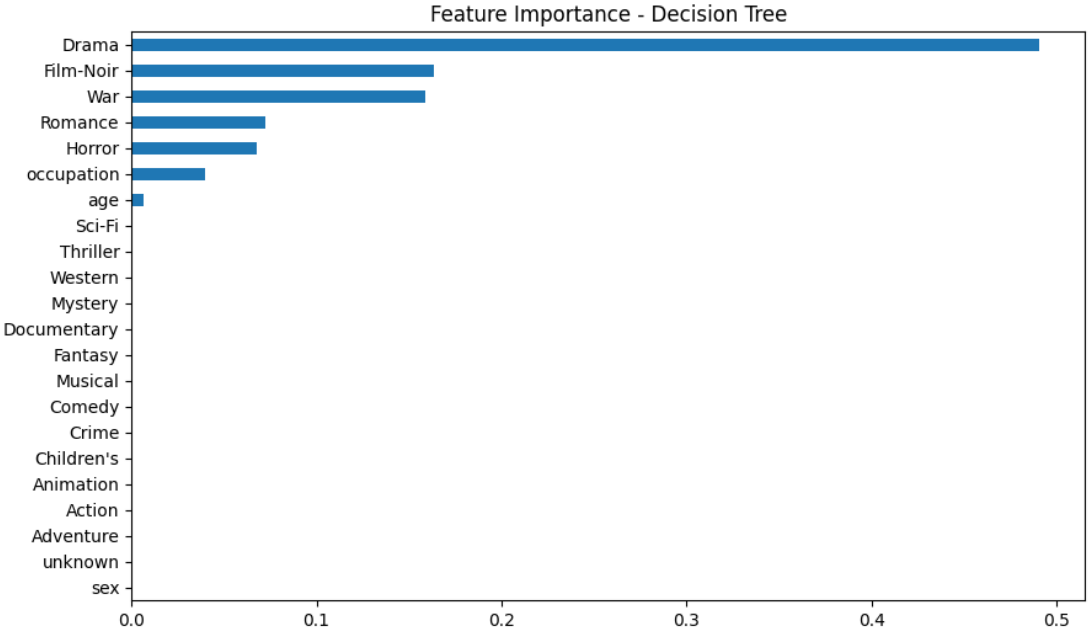
# Feature Importance (Decision Tree)
plt.figure(figsize=(10,6))
pd.Series(dt.feature_importances_, index=X.columns).sort_values().plot.barh()
plt.title('Feature Importance - Decision Tree')
plt.show()

```

## Output

	Accuracy	F1
<b>SVM</b>	0.60845	0.671367
<b>Decision Tree</b>	0.55635	0.557169
<b>KNN (k=15)</b>	0.61895	0.671381
<b>KNN (k=3)</b>	0.59150	0.635528
<b>KNN (k=4)</b>	0.57990	0.566550
<b>KNN (k=5)</b>	0.60215	0.647593
<b>KNN (k=6)</b>	0.59025	0.596445
<b>KNN (k=7)</b>	0.61015	0.658131
<b>KNN (k=8)</b>	0.60070	0.617565
<b>KNN (k=9)</b>	0.61130	0.660998
<b>KNN (k=10)</b>	0.60575	0.631353
<b>KNN (k=11)</b>	0.61825	0.669266
<b>KNN (k=12)</b>	0.60600	0.636900
<b>KNN (k=13)</b>	0.61910	0.671128
<b>KNN (k=14)</b>	0.61245	0.648210



	 <p>SVM &amp; KNN: kotak kanan-bawah (TP) besar → banyak film yang benar-benar disukai terprediksi “liked”. Namun kotak kanan-atas (FP) juga besar → prediksi “liked” untuk film yang sebenarnya tidak disukai. Sedangkan Decision Tree Genre “Drama” dominan (<math>\approx 50\%</math> importance) menjadi split pertama di pohon. Disusul Film-Noir &amp; War juga berpengaruh, lalu Romance &amp; Horror; variabel demografis age/occupation hanya sedikit sex tidak berperan. Preferensi “suka” paling dipicu oleh jenis genre tertentu, bukan profil user.</p>
--	--

## CLUSTERING: Kelompokkan pengguna berdasarkan pola rating film (KMeans)

Langkah	Jawaban/Deskripsi
1	Clustering untuk mengelompokkan pengguna berdasarkan pola rating film mereka. Clustering ini menggunakan algoritma KMeans, yang berfungsi untuk membagi data pengguna ke dalam beberapa kelompok yang memiliki pola rating film yang serupa
2	<pre> Preprocessing Data  # Menggabungkan data pengguna dengan matriks user-item user_item_matrix = ratings.pivot(index='user_id', columns='movie_id', values='rating').fillna(0)  # Menggabungkan dengan data user untuk mendapatkan informasi tentang gender dan occupation user_cluster_data = users.merge(user_item_matrix, left_on='user_id', right_index=True)  # Encoding kolom 'sex' dan 'occupation' user_cluster_data['sex'] = user_cluster_data['sex'].map({'M': 0, 'F': 1}) user_cluster_data['occupation'] = LabelEncoder().fit_transform(user_cluster_data['occupation'])  cols_to_drop = ['user_id', 'zip_code'] cols_to_drop = [col for col in cols_to_drop if col in user_cluster_data.columns] user_cluster_data.drop(columns=cols_to_drop, inplace=True)  [ ] # Mengonversi nama kolom menjadi string (untuk menangani kolom numerik) user_cluster_data.columns = user_cluster_data.columns.astype(str) </pre>



#### Scaling Data

```
# Hanya untuk kolom numerik
scaler_cluster = StandardScaler()
user_cluster_scaled = scaler_cluster.fit_transform(user_cluster_data.select_dtypes(include=['int64', 'float64']))

# Memastikan data sudah distandarisasi
print(user_cluster_scaled[:5]) # Menampilkan 5 baris pertama dari data yang sudah di-scaling
```

```
[[-0.82485939 -0.63832884 1.22705763 ... -0.03258176 -0.03258176
  -0.03258176]
 [ 1.55486734  1.56659263  0.32552639 ... -0.03258176 -0.03258176
  -0.03258176]
 [-0.80691893 -0.63832884 1.37731284 ... -0.03258176 -0.03258176
  -0.03258176]
 [-0.82485939 -0.63832884 1.22705763 ... -0.03258176 -0.03258176
  -0.03258176]
 [-0.88632351  1.56659263  0.32552639 ... -0.03258176 -0.03258176
  -0.03258176]]
```

3

#### KMeans

```
# Menentukan jumlah cluster optimal menggunakan Elbow Method dan Silhouette Score
inertia = []
silhouette_scores = []
k_range = range(2, 11) # Mulai dari 2 cluster

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(user_cluster_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(user_cluster_scaled, kmeans.labels_))

# Visualisasi Elbow Method & Silhouette Score
plt.figure(figsize=(15, 5))

# Plot Elbow Method
plt.subplot(1, 2, 1)
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Jumlah Cluster')
plt.ylabel('Inertia')
plt.title('Elbow Method')

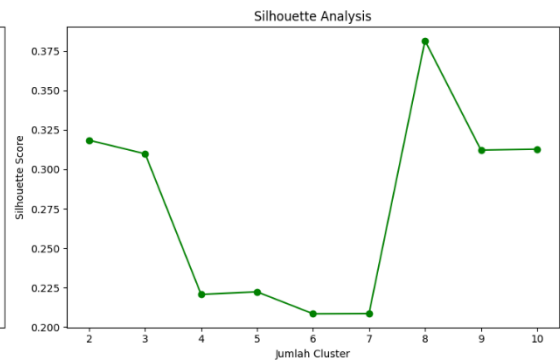
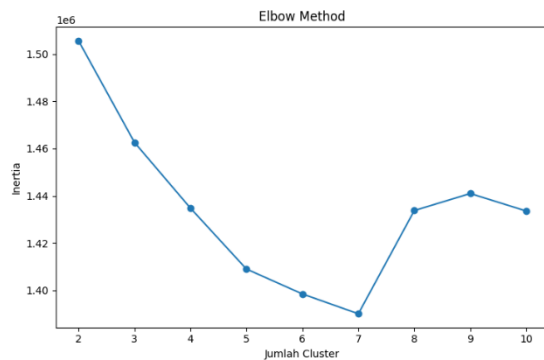
# Plot Silhouette Score
plt.subplot(1, 2, 2)
plt.plot(k_range, silhouette_scores, marker='o', color='green')
plt.xlabel('Jumlah Cluster')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Analysis')

plt.tight_layout()
plt.show()
```

```
# Pilih jumlah cluster optimal, misalnya 5 berdasarkan visualisasi atau intuisi bisnis
best_k = 5
kmeans = KMeans(n_clusters=best_k, random_state=42)
clusters = kmeans.fit_predict(user_cluster_scaled)

# Evaluasi menggunakan Silhouette Score
print(f'\nSilhouette Score (k={best_k}): {silhouette_score(user_cluster_scaled, clusters):.3f}')
```

#### Hasil



**Silhouette Score (k=5): 0.222**

Kualitas pemisahan cluster justru paling tinggi di  $k = 8$  ( $\approx 0,38$ ). Nilai di  $k = 5$  hanya  $\approx 0,22$  masih “cukup”, tetapi jelas di bawah  $k = 8$ .

4

```

from sklearn.decomposition import PCA

# Reduksi Dimensi dengan PCA
pca = PCA(n_components=2) # Mengurangi menjadi 2 dimensi untuk visualisasi 2D
user_cluster_pca = pca.fit_transform(user_cluster_scaled)
users['cluster'] = clusters

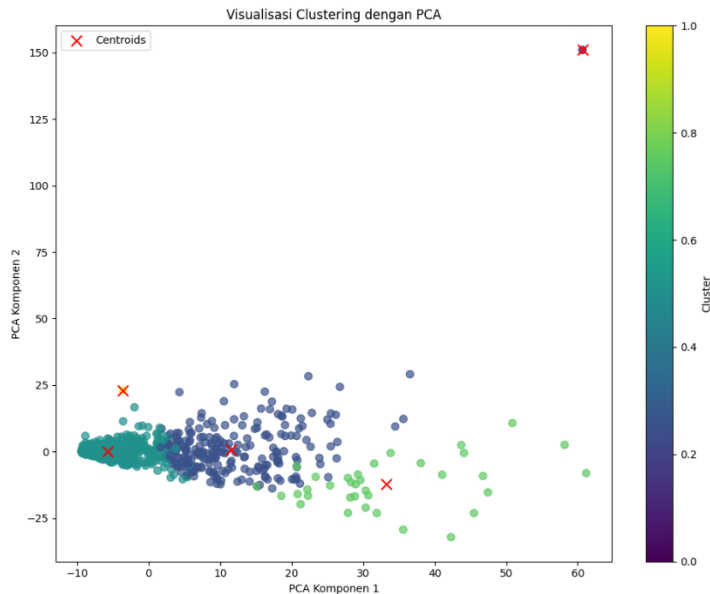
# Visualisasi
plt.figure(figsize=(10, 8))
plt.scatter(user_cluster_pca[:, 0], user_cluster_pca[:, 1], c=users['cluster'], cmap='viridis', s=50, alpha=0.7)
centroids = pca.transform(kmeans.cluster_centers_)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x', s=100, label='Centroids')

plt.title('Visualisasi Clustering dengan PCA')
plt.xlabel('PCA Komponen 1')
plt.ylabel('PCA Komponen 2')
plt.legend()
plt.colorbar(label='Cluster')

plt.tight_layout()
plt.show()

```

Output



Model berhasil memisahkan sebagian besar user ke dalam lima grup yang cukup terpisah

5

```

# Profil Cluster - Rata-rata umur, proporsi gender, dan occupation yang dominan
cluster_profile = users.groupby('cluster').agg([
    'age': ['mean', 'median'],
    'sex': lambda x: (x == 'F').mean(), # Proporsi perempuan
    'occupation': lambda x: x.mode()[0]
]).reset_index()

print("\nProfil Cluster:")
display(cluster_profile)

```

cluster	age	sex	occupation
	mean	median	<lambda>
0	50.000000	50.0	1.000000 healthcare
1	32.853448	30.0	0.224138 student
2	34.807122	32.0	0.310089 student
3	27.371429	25.0	0.285714 student
4	21.000000	21.0	1.000000 artist

## KESIMPULAN

Model	Tugas	Metrik	Skor	Catatan
LinearRegression	Regresi	MAE / RMSE	0.911 / 1.107	$R^2 \approx 0.03$ ; cenderung menebak rata-rata untuk prediksi rating film sehingga kurang presisi

SVM	Klasifikasi	Akurasi / F1	0.608 / 0.671	Recall tinggi untuk kelas <i>liked</i> ; banyak false-positive pada kelas <i>dislike</i> .
Decision Tree	Klasifikasi	Akurasi / F1	0.556 / 0.557	Dengan depth = 3, genre <i>Drama</i> paling menentukan. Lalu sederhana dan interpretatif
KNN	Klasifikasi	Akurasi / F1	0.619 / 0.671	Nilai k dicoba dari 3 sampai 15 Performa terbaik di rentang 11-15, puncak pada k = 15.
KMeans	Clustering	Silhouette	0.222	Cocokkan jumlah kluster dengan intuisi bisnis atau elbow method  Elbow mengarah ke $k \approx 5$ , tetapi Silhouette tertinggi di $k = 8$ ( $\sim 0.38$ ) k dapat disesuaikan intuisi bisnis vs kualitas kluster. Misal Jika bisnis hanya butuh sedikit segmen yang mudah dijelaskan dan dieksekusi seperti paket promo maka ambil $k = 5$ .