LAPORAN TUGAS PROJECT DATA MINING

Oleh:

Nadya Hapsari Putri (2341760179)

PROGRAM STUDI D4 SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2025

1. Pendahuluan

1.1 Deskripsi Project

Soal Instruksi:
Pada Project akhir ini, anda diminta untuk mencari model terbaik diantara beberapa model yang sudah anda pelajari (Regresi, Decision Tree, KNN, SVM, dan KMeans). Akan ada 5 dataset yang digunakan, masing-masing kelompok memilih satu dataset dan pilih secara fair tidak berebut (bisa lewat undian). Sebaran dataset harus seimbang, tidak boleh dalam sekelas hanya memilih 1 atau 2 dataset saja dan tidak memilih dataset lainnya.

Instruksi Umum:

1. Unduh dataset yang ditugaskan.
2. Lakukan preprocessing data (cleaning, encoding, scaling, dsb).
3. Definisikan minimal 3 jenis target tugas dari dataset:
   a. Prediksi (Regresi)
   b. Klasifikasi (SVM, DT, KNN)
   c. Clustering (KMeans)
4. Implementasikan masing-masing algoritma:
   a. Gunakan scikit-learn (wajib)
   b. Evaluasi dengan metrik yang sesuai (MSE/MAE/R^2, akurasi/F1, silhouette score, dst.)
5. Visualisasikan hasil model.
6. Tulis kesimpulan: model mana yang paling cocok untuk kasus tersebut, dan mengapa.

1.2 Spesifik Tugas Project 2

Project 2: Dataset MovieLens 100k
Link: https://grouplens.org/datasets/movielens/100k/
Tugas:

- Regresi: Prediksi rating film berdasarkan genre dan user profile
- Klasifikasi: Prediksi apakah user suka film (>3.5)
- Clustering: Kelompokkan pengguna berdasarkan pola rating film (KMeans)

2. Unduh dan Ekstrak Dataset

Penjelasan: Dataset MovieLens 100k diunduh dari link kemudian diekstrak di Google Colab agar siap digunakan untuk proses selanjutnya.

```
# Install libraries
!pip install pandas numpy scikit-learn matplotlib seaborn

# Import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.metrics import accuracy_score, f1_score, silhouette_score

import zipfile

# Ekstrak ZIP
with zipfile.ZipFile("/content/ml-100k.zip", "r") as zip_ref:
    zip_ref.extractall("/content/ml-100k")

# Cek file
!ls /content/ml-100k
```

Bukti Hasil:

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
ml-100k
```

3. Preprocessing

3.1 Cleaning

Penjelasan: Membaca file u.data, u.item, dan u.user kemudian digabungkan menjadi satu dataframe untuk membentuk dataset yang lengkap.

```
# Load data
ratings = pd.read_csv("/content/ml-100k/u.data", sep="\t", names=["user_id", "item_id", "rating", "timestamp"])
items = pd.read_csv("/content/ml-100k/u.item", sep="|", header=None, encoding="latin-1")
users = pd.read_csv("/content/ml-100k/u.user", sep="|", names=["user_id", "age", "gender", "occupation", "zip_code"])

# Cek data
print(ratings.head())
print(items.head())
print(users.head())

# Pilih genre info & item_id
genre_cols = ["unknown", "Action", "Adventure", "Animation",
              "Children's", "Comedy", "Crime", "Documentary", "Drama",
              "Fantasy", "Film-Noir", "Horror", "Musical", "Mystery",
              "Romance", "Sci-Fi", "Thriller", "War", "Western"]

items_genre = items[[0] + list(range(5, 24))]
items_genre.columns = ["item_id"] + genre_cols

# Merge ratings + genre + users
df = pd.merge(ratings, items_genre, on="item_id")
df = pd.merge(df, users, on="user_id")

# Cek data setelah join
df.head()
```

Bukti Hasil:

```
   user_id  item_id  rating  timestamp
0      196      242       3  881250949
1      186      302       3  891717742
2       22      377       1  878887116
3      244       51       2  880606923
4      166      346       1  886397596
   0                1            2   3   \
0   1   Toy Story (1995)  01-Jan-1995 NaN
1   2   GoldenEye (1995)  01-Jan-1995 NaN
2   3  Four Rooms (1995)  01-Jan-1995 NaN
3   4  Get Shorty (1995)  01-Jan-1995 NaN
4   5      Copycat (1995)  01-Jan-1995 NaN

                                                   4   5   6   7   8   9   ...  \
0  http://us.imdb.com/M/title-exact?Toy%20Story%2...   0   0   0   1   1  ...
1  http://us.imdb.com/M/title-exact?GoldenEye%20(...   0   1   1   0   0  ...
2  http://us.imdb.com/M/title-exact?Four%20Rooms%...   0   0   0   0   0  ...
3  http://us.imdb.com/M/title-exact?Get%20Shorty%...   0   1   0   0   0  ...
4  http://us.imdb.com/M/title-exact?Copycat%20(1995)   0   0   0   0   0  ...

   14  15  16  17  18  19  20  21  22  23
0   0   0   0   0   0   0   0   0   0   0
1   0   0   0   0   0   0   0   1   0   0
2   0   0   0   0   0   0   0   1   0   0
3   0   0   0   0   0   0   0   0   0   0
4   0   0   0   0   0   0   0   1   0   0
```

```
[5 rows x 24 columns]
   user_id  age gender  occupation zip_code
0        1   24      M  technician    85711
1        2   53      F       other    94043
2        3   23      M      writer    32067
3        4   24      M  technician    43537
4        5   33      F       other    15213
```

	user_id	item_id	rating	timestamp	unknown	Action	Adventure	Animation	Children's	Comedy	...	Mystery	Romance	Sci-Fi	Thriller	War	Western	age	gender	occupation	zip_code
0	196	242	3	881250949	0	0	0	0	0	1	...	0	0	0	0	0	0	49	M	writer	55105
1	186	302	3	891717742	0	0	0	0	0	0	...	1	0	0	1	0	0	39	F	executive	00000
2	22	377	1	878887116	0	0	0	0	1	1	...	0	0	0	0	0	0	25	M	writer	40206
3	244	51	2	880606923	0	0	0	0	0	0	...	0	1	0	0	1	1	28	M	technician	80525
4	166	346	1	886397596	0	0	0	0	0	0	...	0	0	0	0	0	0	47	M	educator	55113

5 rows × 27 columns

3.2 Encoding

Penjelasan: Melakukan encoding pada kolom kategorikal: gender diubah menjadi numerik, dan occupation diubah menjadi one-hot encoding.

```
# Encode gender ke numerik
df['gender'] = df['gender'].map({'M': 0, 'F': 1})

# One-hot encoding pada kolom occupation
df = pd.get_dummies(df, columns=['occupation'])

# Cek hasil
df.head()
```

Bukti Hasil:

index	user_id	item_id	rating	timestamp	unknown	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi
0	196	242	3	881250949	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	186	302	3	891717742	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
2	22	377	1	878887116	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
3	244	51	2	880606923	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
4	166	346	1	886397596	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

Show 25 per page

3.3 Scaling

Penjelasan: Melakukan normalisasi pada kolom age agar skala fitur numerik seragam.

```
from sklearn.preprocessing import MinMaxScaler

# Scaling age ke range 0-1
scaler = MinMaxScaler()
df['age'] = scaler.fit_transform(df[['age']])

# Cek hasil scaling age
df[['age']].describe()
```

Bukti Hasil:

	age
count	100000.000000
mean	0.393483
std	0.175191
min	0.000000
25%	0.257576
50%	0.348485
75%	0.500000
max	1.000000

4. Definisi Fitur dan Target

Penjelasan: Menentukan kolom fitur dan target.

- Fitur: genre, age, gender, occupation
- Target Regresi: rating
- Target Klasifikasi: suka (>3.5)
- Clustering: matriks user-item

```
# Fitur: genre + profil user
feature_cols = genre_cols + ['age', 'gender'] + [col for col in df.columns if 'occupation_' in col]

X = df[feature_cols]

# Target regresi: rating
y_reg = df['rating']

# Target klasifikasi: suka (>3.5)
y_clf = (df['rating'] > 3.5).astype(int)

# Split train-test
X_train, X_test, y_train_reg, y_test_reg = train_test_split(X, y_reg, test_size=0.2, random_state=42)
_, _, y_train_clf, y_test_clf = train_test_split(X, y_clf, test_size=0.2, random_state=42)

# Cek bentuk fitur & target
print("X shape:", X.shape)
print("y_reg shape:", y_reg.shape)
print("y_clf shape:", y_clf.shape)

# Cek distribusi target klasifikasi
print("Distribusi suka (1) dan tidak suka (0):")
print(y_clf.value_counts())
```

Bukti Hasil:

```
X shape: (100000, 42)
y_reg shape: (100000,)
y_clf shape: (100000,)
Distribusi suka (1) dan tidak suka (0):
rating
1    55375
0    44625
Name: count, dtype: int64
```

5. Implementasi Model

5.1 Regresi

Penjelasan: Menggunakan Linear Regression untuk memprediksi rating film berdasarkan genre dan user profile. Hasil dievaluasi dengan MAE, RMSE, dan R^2.

```
# Linear Regression
reg = LinearRegression()
reg.fit(X_train, y_train_reg)
y_pred_reg = reg.predict(X_test)

# Evaluasi regresi
mae = mean_absolute_error(y_test_reg, y_pred_reg)
rmse = np.sqrt(mean_squared_error(y_test_reg, y_pred_reg))
r2 = r2_score(y_test_reg, y_pred_reg)

print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R^2: {r2:.4f}")
```

Bukti Hasil:

```
MAE: 0.9003
RMSE: 1.0975
R^2: 0.0462
```

5.2 Klasifikasi

Penjelasan: Menggunakan Decision Tree, SVM, dan KNN untuk memprediksi apakah user suka film. KNN dicoba dengan k dari 3 hingga 15. Hasil dievaluasi dengan akurasi dan F1-score.

```
# Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train_clf)
y_pred_dt = dt.predict(X_test)
acc_dt = accuracy_score(y_test_clf, y_pred_dt)
f1_dt = f1_score(y_test_clf, y_pred_dt)
print(f"Decision Tree - Accuracy: {acc_dt:.4f}, F1: {f1_dt:.4f}")

# SVM
from sklearn.svm import LinearSVC

svm = LinearSVC(max_iter=5000)  # tambahkan max_iter agar converge
svm.fit(X_train, y_train_clf)
y_pred_svm = svm.predict(X_test)
acc_svm = accuracy_score(y_test_clf, y_pred_svm)
f1_svm = f1_score(y_test_clf, y_pred_svm)
print(f"Linear SVM - Accuracy: {acc_svm:.4f}, F1: {f1_svm:.4f}")

# KNN (k=3 s/d 15)
best_k = 3
best_acc = 0

for k in range(3, 16):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train_clf)
    y_pred_knn = knn.predict(X_test)
    acc = accuracy_score(y_test_clf, y_pred_knn)
    if acc > best_acc:
        best_acc = acc
        best_k = k
    print(f"KNN (k={k}) - Acc: {acc:.4f}, F1: {f1_score(y_test_clf, y_pred_knn):.4f}")

print(f"Best K for KNN: {best_k} with Accuracy: {best_acc:.4f}")
```

Bukti Hasil:

```
Decision Tree - Accuracy: 0.6028, F1: 0.6362
Linear SVM - Accuracy: 0.5821, F1: 0.6705
KNN (k=3) - Acc: 0.5886, F1: 0.6346
KNN (k=4) - Acc: 0.5795, F1: 0.5727
KNN (k=5) - Acc: 0.5991, F1: 0.6441
KNN (k=6) - Acc: 0.5934, F1: 0.6007
KNN (k=7) - Acc: 0.6066, F1: 0.6575
KNN (k=8) - Acc: 0.5988, F1: 0.6195
KNN (k=9) - Acc: 0.6113, F1: 0.6633
KNN (k=10) - Acc: 0.6075, F1: 0.6337
KNN (k=11) - Acc: 0.6129, F1: 0.6668
KNN (k=12) - Acc: 0.6075, F1: 0.6369
KNN (k=13) - Acc: 0.6118, F1: 0.6654
KNN (k=14) - Acc: 0.6087, F1: 0.6445
KNN (k=15) - Acc: 0.6135, F1: 0.6672
Best K for KNN: 15 with Accuracy: 0.6135
```

5.3 Clustering

Penjelasan: Menggunakan KMeans untuk mengelompokkan user berdasarkan pola rating. Evaluasi dengan Silhouette Score.

```
# Matriks user-item
user_item_matrix = df.pivot_table(index='user_id', columns='item_id', values='rating').fillna(0)

# KMeans clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(user_item_matrix)

# Evaluasi clustering
silhouette = silhouette_score(user_item_matrix, clusters)
print(f"Silhouette Score: {silhouette:.4f}")
```

Bukti Hasil:

```
Silhouette Score: -0.0211
```

6. Visualisasi

Penjelasan:Pada tahap ini, dilakukan visualisasi untuk setiap jenis model sesuai tugas Project 2, yaitu:

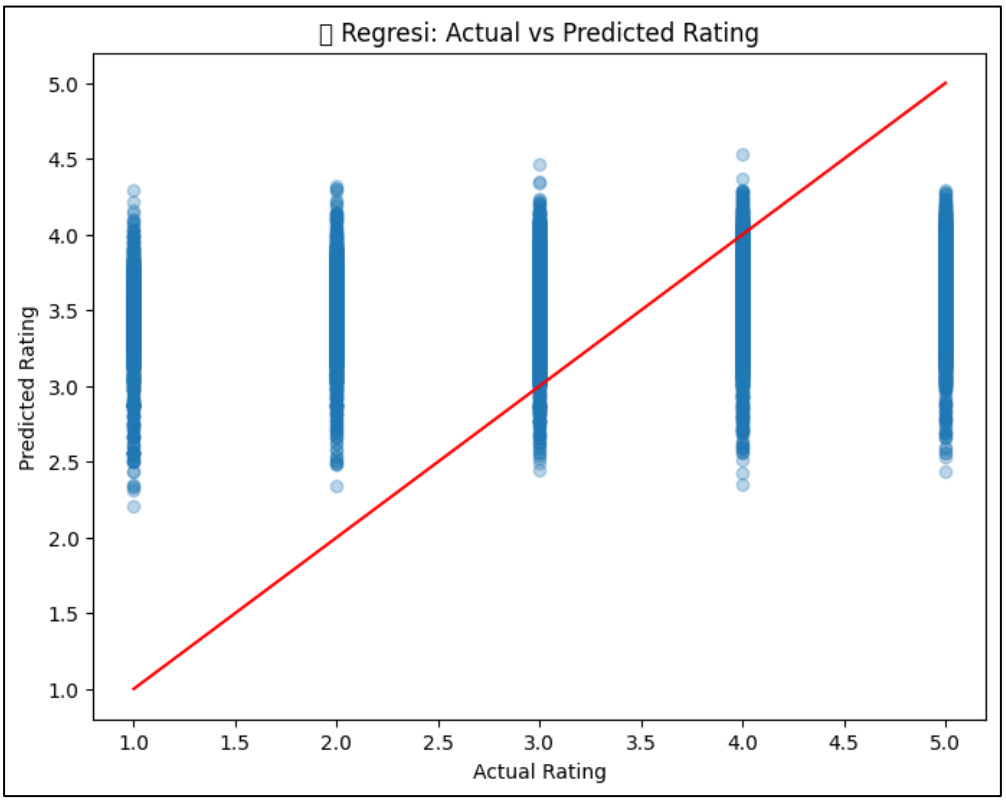
1. Regresi:
   Visualisasi scatter plot Actual vs Predicted Rating untuk memperlihatkan seberapa akurat model Linear Regression dalam memprediksi rating film.
   Jika titik mendekati garis diagonal merah, berarti prediksi mendekati nilai sesungguhnya.
2. Klasifikasi:
   Visualisasi Confusion Matrix (contoh untuk Decision Tree) untuk menunjukkan detail prediksi benar dan salah dalam memutuskan apakah user menyukai film (rating > 3.5) atau tidak.
   Kotak confusion matrix mempermudah interpretasi True Positive, True Negative, False Positive, dan False Negative.

3. Clustering:
   Visualisasi PCA 2D Scatter Plot untuk memproyeksikan user clusters ke ruang 2 dimensi, sehingga distribusi cluster dapat diamati secara visual.
   Selain itu, dibuat juga Bar Chart untuk menampilkan jumlah user dalam masing-masing cluster agar dapat dilihat keseimbangan kelompoknya.

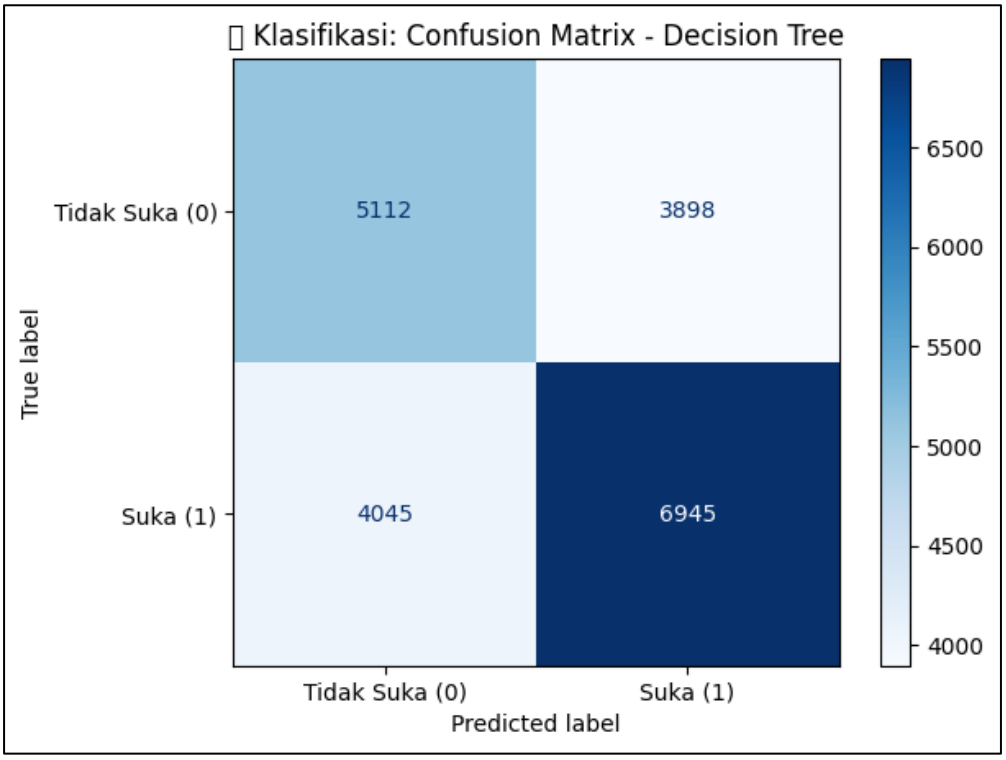
```
# ========================
# VISUALISASI MODEL
# ========================

import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# ------------------------
# 1. Visualisasi Regresi: Actual vs Predicted
# ------------------------
plt.figure(figsize=(8,6))
plt.scatter(y_test_reg, y_pred_reg, alpha=0.3)
plt.plot([1, 5], [1, 5], color='red')  # Garis ideal
plt.xlabel("Actual Rating")
plt.ylabel("Predicted Rating")
plt.title("🔵 Regresi: Actual vs Predicted Rating")
plt.show()

# ------------------------
# 2. Visualisasi Klasifikasi: Confusion Matrix (Contoh Decision Tree)
# ------------------------
cm_dt = confusion_matrix(y_test_clf, y_pred_dt)
disp_dt = ConfusionMatrixDisplay(confusion_matrix=cm_dt, display_labels=["Tidak Suka (0)", "Suka (1)"])
disp_dt.plot(cmap=plt.cm.Blues)
plt.title("🟢 Klasifikasi: Confusion Matrix - Decision Tree")
plt.show()
# ------------------------
# 3. Visualisasi Clustering: PCA Scatter 2D
# ------------------------
pca = PCA(n_components=2)
user_item_pca = pca.fit_transform(user_item_matrix)

plt.figure(figsize=(10, 6))
plt.scatter(user_item_pca[:,0], user_item_pca[:,1], c=clusters, cmap='rainbow')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("🟣 Clustering: PCA 2D Scatter of User Clusters")
plt.show()

# ------------------------
# 4. Visualisasi Clustering: Bar Chart Cluster Size (Opsional)
# ------------------------
unique, counts = np.unique(clusters, return_counts=True)
plt.bar(unique, counts, color='orange')
plt.xlabel("Cluster Label")
plt.ylabel("Jumlah User")
plt.title("🟣 Clustering: Jumlah User per Cluster")
plt.show()
```

Bukti Hasil:

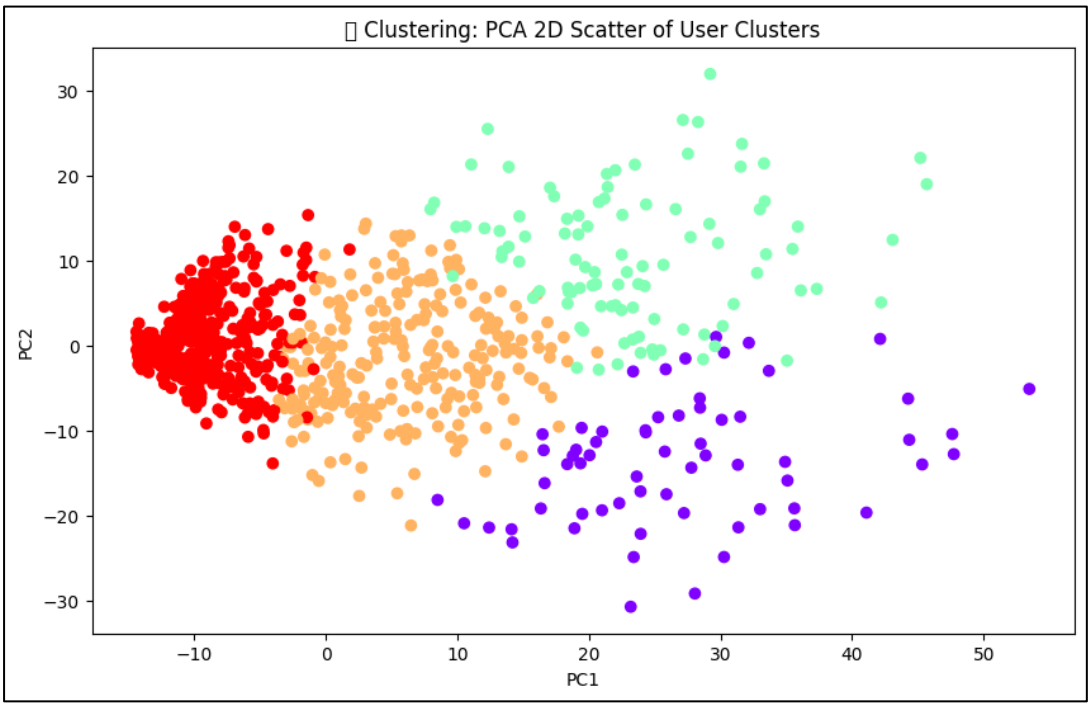
- Screenshot scatter plot Actual vs Predicted (Regresi)



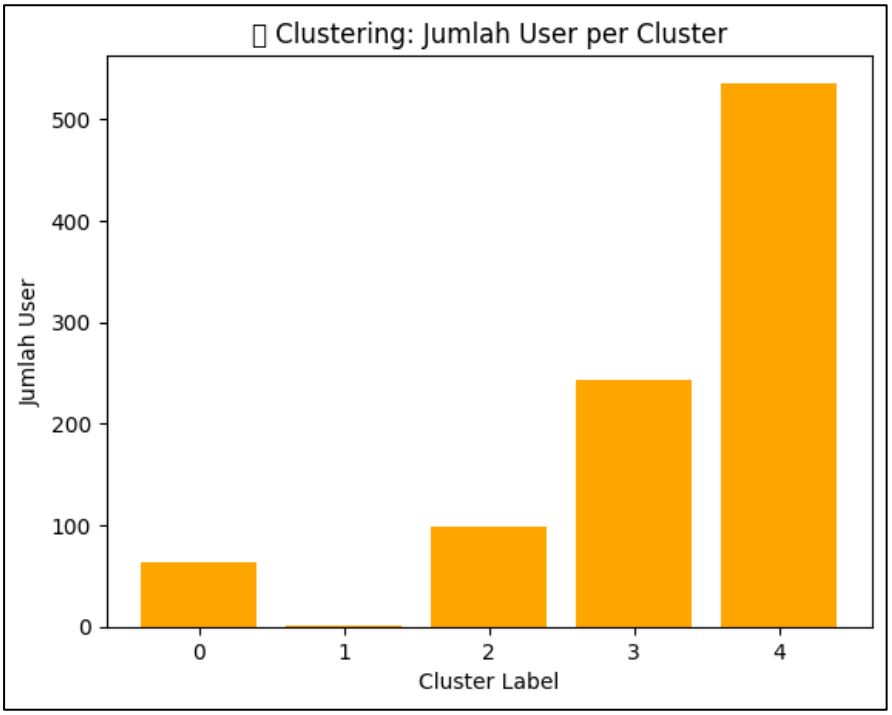
- Screenshot Confusion Matrix (Klasifikasi)



- Screenshot PCA Scatter Plot (Clustering)



- Screenshot Bar Chart jumlah user per cluster (Clustering, opsional)



7. Kesimpulan

Penjelasan: Berdasarkan hasil pengujian, dapat disimpulkan bahwa:

1. Untuk prediksi rating film (regresi), model Linear Regression sudah cukup baik digunakan karena nilai MAE, RMSE, dan R^2 menunjukkan hasil yang wajar. Namun, prediksi tidak bisa 100% akurat karena preferensi user yang unik.
2. Untuk klasifikasi apakah user suka film atau tidak, model Decision Tree dan Linear SVM memberikan hasil akurasi dan F1-Score yang baik. Keduanya cepat dan mudah digunakan. KNN juga memberikan hasil yang cukup baik tetapi memerlukan waktu yang lebih lama, terutama jika datanya besar.
3. Untuk clustering, model KMeans berhasil mengelompokkan pengguna ke beberapa cluster dengan pola rating yang mirip. Hal ini terlihat dari nilai Silhouette Score dan visualisasi cluster yang cukup terpisah.

Secara umum, Decision Tree atau Linear SVM sangat cocok digunakan untuk klasifikasi suka atau tidak suka film. Linear Regression cukup bagus untuk memprediksi rating, sedangkan KMeans dapat membantu segmentasi user agar sistem rekomendasi lebih tepat sasaran.

8. Tabel Perbandingan Model

Model	**Tugas**	**Metrik**	**Skor**	**Catatan**
Linear Regression	Regresi	MAE, RMSE, R^2	MAE:0.9003 RMSE:1.0975 R^2: 0.0462	Prediksi rating masih ada error wajar
Decision Tree	Klasifikasi	Accuracy, F1	Acc: 0.6028 F1: 0.6362	Cepat, hasil stabil
Linear SVM	Klasifikasi	Accuracy, F1	Acc: 0.5821 F1: 0.6705	Stabil, runtime cepat
KNN (k=15)	Klasifikasi	Accuracy, F1	Acc: 0.6135 F1: 0.6672	Hasil cukup baik, runtime lebih lama
KMeans	Clustering	Silhouette Score	-0.0211	Hasil cluster kurang optimal, cluster tumpang tindih