# Modul Praktikum
# Data Mining



## Tim Penyusun:

Dr. Rakhmat Arianto, S.ST., M.Kom

Ir. Rudy Ariyanto, ST., M.Cs

Prof. Dr. Eng. Rosa Andrie Asmara, ST., MT

**Jurusan Teknologi Informasi**

**Sistem Informasi Bisnis**

**Politeknik Negeri Malang**



**Februari 2025**

# DAFTAR ISI

# JOBSHEET 15

# Support Vector Machine

## Pendahuluan

Modul ini menjelaskan penerapan algoritma Support Vector Machine dengan menggunakan studi kasus Kelayakan Produk yang dilengkapi dengan tahapan yang bisa diambil kesimpulannya

## Tujuan Praktikum

Setelah menyelesaikan praktikum ini, mahasiswa mampu:

- Memahami konsep dasar klasifikasi SVM
- Mengimplementasikan SVM menggunakan Python
- Melakukan interpretasi terhadap hasil klasifikasi

## Peralatan yang dibutuhkan

Beberapa peralatan yang dibutuhkan dalam menyelesaikan praktikum ini adalah:

- Google Colab
- Google Drive
- Koneksi Internet
- Browser Web

## Praktikum

### Praktikum Support Vector Machine

Studi Kasus yang digunakan dalam praktikum ini adalah data pada tabel sebagai berikut:

| Index | Durability | DesignScore | CostEfficiency | Usability | EcoFriendly | Label |
|---|---|---|---|---|---|---|
| **0** | 9 | 8 | 7 | 9 | 8 | 2 |
| **1** | 7 | 7 | 6 | 8 | 7 | 2 |
| **2** | 8 | 9 | 8 | 9 | 9 | 2 |
| **3** | 5 | 5 | 4 | 6 | 5 | 1 |
| **4** | 6 | 6 | 5 | 7 | 6 | 1 |
| **5** | 3 | 3 | 2 | 4 | 3 | 0 |
| **6** | 4 | 4 | 3 | 5 | 4 | 0 |
| **7** | 2 | 2 | 1 | 2 | 1 | 0 |
| **8** | 1 | 1 | 1 | 1 | 2 | 0 |
| **9** | 5 | 6 | 5 | 6 | 5 | 1 |
| **10** | 6 | 5 | 6 | 7 | 6 | 1 |
| **11** | 4 | 3 | 3 | 4 | 3 | 0 |
| **12** | 3 | 2 | 2 | 3 | 2 | 0 |
| **13** | 7 | 7 | 6 | 8 | 7 | 2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **14** | 8 | 9 | 8 | 9 | 9 | 2 |

Lakukan praktikum sesuai tahapan berikut:

a. Buka aplikasi web browser

b. Buka Google Colabs dan berikan nama file “SVM.ipynb”

c. Lakukan impor library yang dibutuhkan:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
```

d. Lakukan pembuatan dataset

```
data = {
    'Durability':      [9, 7, 8, 5, 6, 3, 4, 2, 1, 5, 6, 4, 3, 7, 8],
    'DesignScore':     [8, 7, 9, 5, 6, 3, 4, 2, 1, 6, 5, 3, 2, 7, 9],
    'CostEfficiency': [7, 6, 8, 4, 5, 2, 3, 1, 1, 5, 6, 3, 2, 6, 8],
    'Usability':       [9, 8, 9, 6, 7, 4, 5, 2, 1, 6, 7, 4, 3, 8, 9],
    'EcoFriendly':     [8, 7, 9, 5, 6, 3, 4, 1, 2, 5, 6, 3, 2, 7, 9],
    'Label':           [2, 2, 2, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 2]
}
df = pd.DataFrame(data)
```

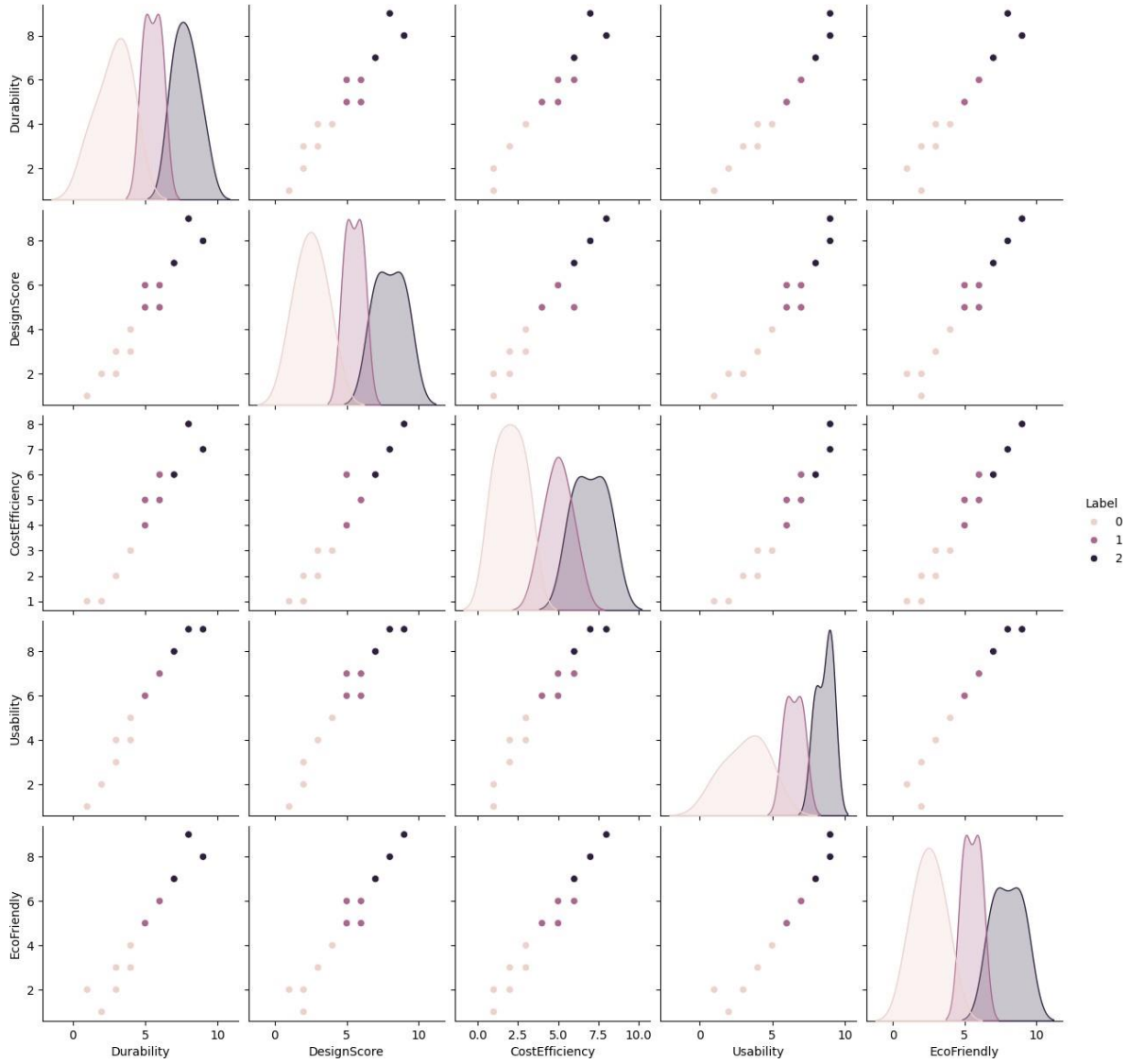
```
data = {
    'Durability':       [9, 7, 8, 5, 6, 3, 4, 2, 1, 5, 6, 4, 3, 7, 8],
    'DesignScore':      [8, 7, 9, 5, 6, 3, 4, 2, 1, 6, 5, 3, 2, 7, 9],
    'CostEfficiency':   [7, 6, 8, 4, 5, 3, 2, 1, 5, 6, 3, 1, 2, 6, 8],
    'Usability':        [9, 8, 9, 6, 7, 4, 3, 2, 4, 7, 6, 3, 2, 7, 9],
    'EcoFriendly':      [8, 7, 9, 5, 6, 3, 2, 1, 3, 6, 5, 2, 3, 7, 9],
    'Label':            [2, 2, 2, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 2, 2]
}
df = pd.DataFrame(data)
```

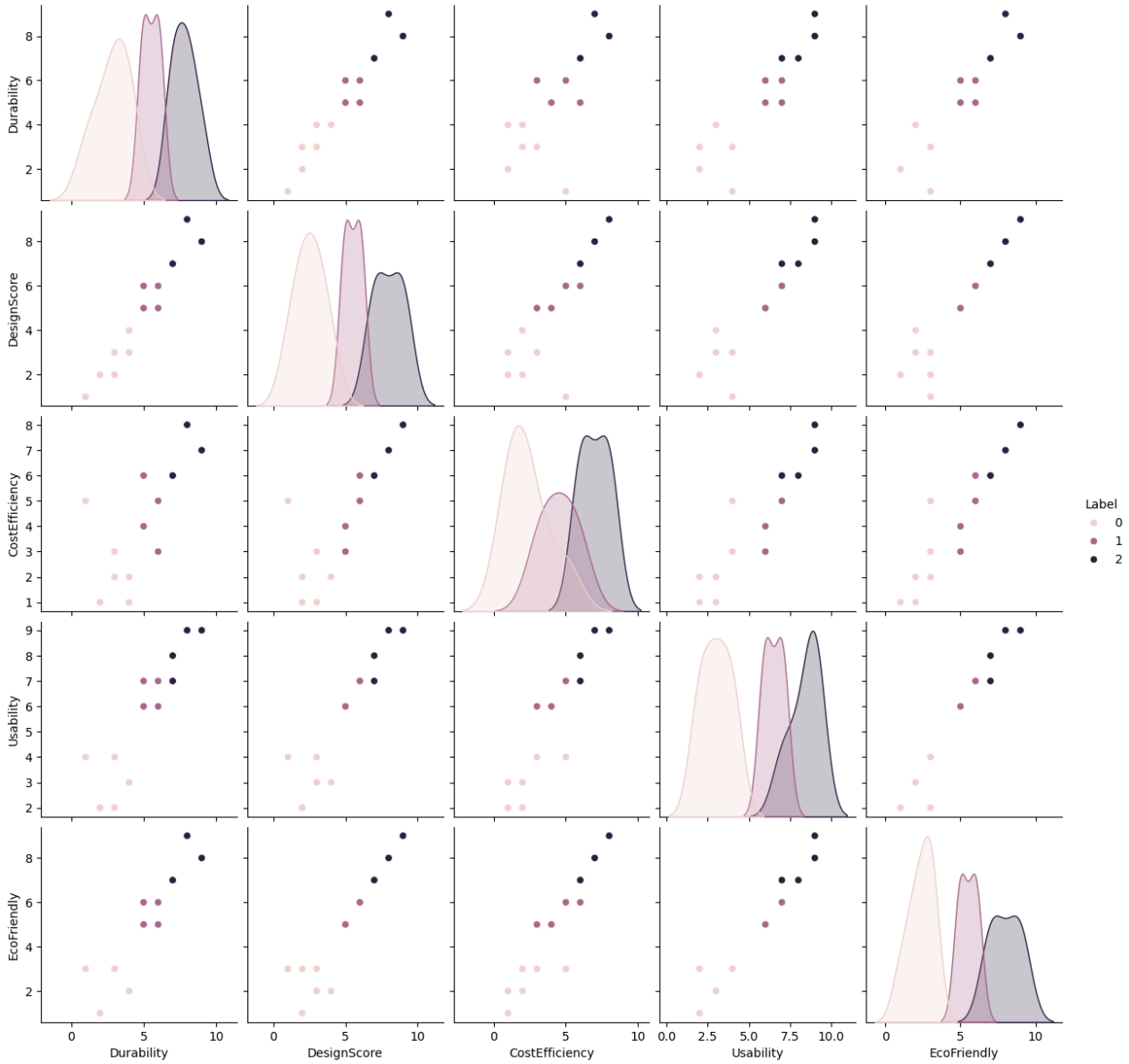
e. Lakukan eksplorasi data analysis

```
print("\n Ringkasan Statistik:")
print(df.describe())
sns.pairplot(df, hue='Label')
plt.show()
```


```
print("\n Ringkasan Statistik:")
print(df.describe())
sns.pairplot(df, hue='Label')
plt.show()
```


```
 Ringkasan Statistik:
       Durability  DesignScore  CostEfficiency  Usability  EcoFriendly  \
count   15.000000    15.000000       15.000000  15.000000    15.000000
mean     5.200000     5.133333        4.466667   5.733333     5.066667
std      2.366432     2.587516        2.386470   2.548576     2.631313
min      1.000000     1.000000        1.000000   2.000000     1.000000
25%      3.500000     3.000000        2.500000   3.500000     3.000000
50%      5.000000     5.000000        5.000000   6.000000     5.000000
75%      7.000000     7.000000        6.000000   7.500000     7.000000
max      9.000000     9.000000        8.000000   9.000000     9.000000

           Label
count  15.000000
mean    0.933333
std     0.883715
min     0.000000
25%     0.000000
50%     1.000000
75%     2.000000
max     2.000000
```


Durability
DesignScore
CostEfficiency
Usability
EcoFriendly
Label
0
1
2




f. Lakukan pemilihan fitur dan split dataset untuk training dan testing

```
X = df.drop('Label', axis=1)
y = df['Label']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

```
X = df.drop('Label', axis=1)
y = df['Label']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

g. Membangun Model Support Vector Machine

```
model = SVC(kernel='rbf', C=1, gamma='scale', decision_function_shape='ovo')
model.fit(X_train, y_train)
```


```
model = SVC(kernel='rbf', C=1, gamma='scale', decision_function_shape='ovo')
model.fit(X_train, y_train)
```


h. Melakukan prediksi dan validasi model

```
y_pred = model.predict(X_test)
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

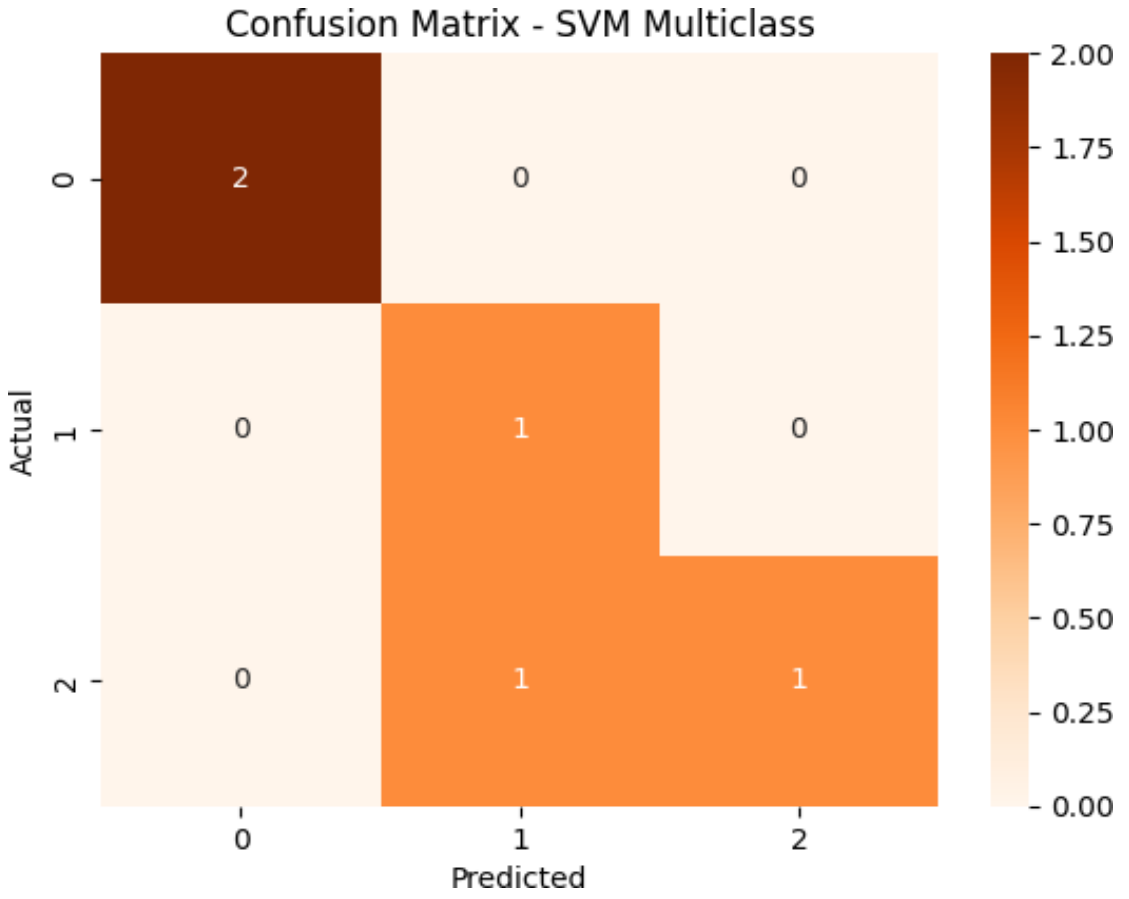
```
Confusion Matrix:
[[2 0 0]
 [0 1 0]
 [0 1 1]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.50      1.00      0.67         1
           2       1.00      0.50      0.67         2

    accuracy                           0.80         5
   macro avg       0.83      0.83      0.78         5
weighted avg       0.90      0.80      0.80         5
```

```
y_pred = model.predict(X_test)
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

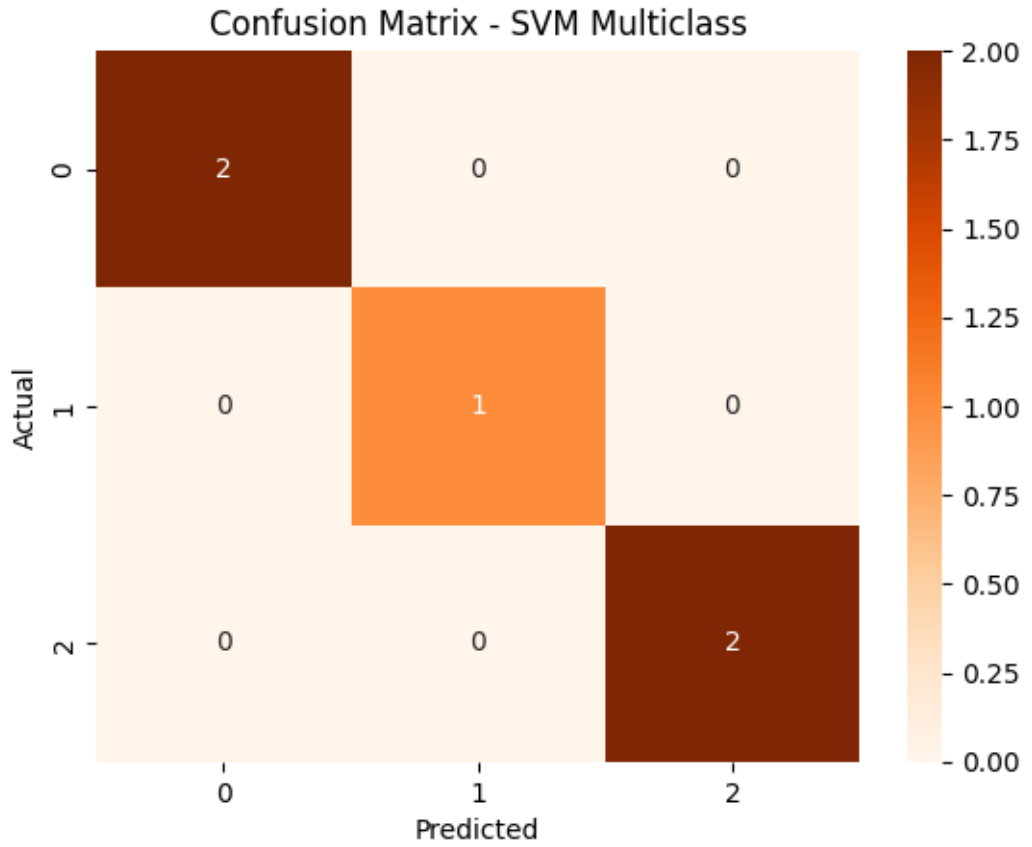
```
Confusion Matrix:
[[2 0 0]
 [0 1 0]
 [0 0 2]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         2

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```

i. Menampilkan hasil Support Vector Machine:

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Oranges')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - SVM Multiclass")
plt.show()
```




```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Oranges')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - SVM Multiclass")
plt.show()
```




**LATIHAN**

1. Jelaskan secara mendetail dari hasil eksplorasi data analysis yang dilakukan!

   Jawaban:

   Dari hasil eksplorasi data analysis yang dilakukan, dapat dijelaskan:

   1. Dataset memiliki 15 sampel dengan 6 kolom (5 fitur dan 1 target)

   2. Fitur: Durability, DesignScore, CostEfficiency, Usability, EcoFriendly

   3. Target: Label (0, 1, 2)

   4. Distribusi kelas:

      - Kelas 0: 6 sampel (40%)

      - Kelas 1: 4 sampel (26.7%)

      - Kelas 2: 5 sampel (33.3%)

   5. Statistik deskriptif menunjukkan:

      - Range nilai untuk semua fitur antara 1-9

      - Nilai mean berkisar antara 4-6 untuk semua fitur

      - Standar deviasi berkisar antara 2.4-2.8

   6. Korelasi antar fitur sangat tinggi (>0.9), menunjukkan multikolinearitas

   7. Semua fitur berkorelasi positif dengan Label, menunjukkan bahwa nilai fitur yang lebih tinggi cenderung menghasilkan Label yang lebih tinggi

   8. Standarisasi diperlukan karena SVM sensitif terhadap skala fitur

**LATIHAN**

2. Latih model SVM dengan kernel: “linear” dan “poly”

```
# 2. Latih model SVM dengan kernel: "linear" dan "poly"
print("2. PELATIHAN MODEL DENGAN BERBAGAI KERNEL:")
print("="*50)

kernels = ['linear', 'poly', 'rbf']
results = {}

for kernel in kernels:
    print(f"\nMelatih model dengan kernel: {kernel}")

    if kernel == 'poly':
        svm = SVC(kernel=kernel, degree=3, random_state=42)
    else:
        svm = SVC(kernel=kernel, random_state=42)

    svm.fit(X_train_scaled, y_train)
    y_pred = svm.predict(X_test_scaled)

    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred, output_dict=True)

    results[kernel] = {
        'accuracy': accuracy,
        'confusion_matrix': cm,
        'classification_report': cr,
        'predictions': y_pred
    }

    print(f"Akurasi {kernel}: {accuracy:.4f}")
```

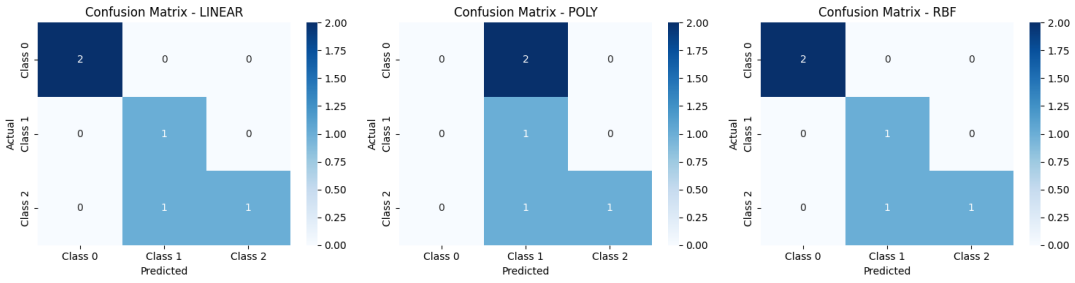
```
2. PELATIHAN MODEL DENGAN BERBAGAI KERNEL:
==================================================

Melatih model dengan kernel: linear
Akurasi linear: 0.8000

Melatih model dengan kernel: poly
Akurasi poly: 0.4000

Melatih model dengan kernel: rbf
Akurasi rbf: 0.8000
```

**LATIHAN**

3. Bandingkan hasil klasifikasinya (Confusion Matrix dan Classification Report).

```
# 3. Bandingkan hasil klasifikasinya
print("\n3. PERBANDINGAN HASIL KLASIFIKASI:")
print("="*50)

print("Perbandingan Akurasi:")
for kernel, result in results.items():
    print(f"  {kernel.upper()}: {result['accuracy']:.4f}")

print("\nConfusion Matrix untuk setiap kernel:")
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

for i, (kernel, result) in enumerate(results.items()):
    sns.heatmap(result['confusion_matrix'], annot=True, fmt='d', cmap='Blues',
                xticklabels=['Class 0', 'Class 1', 'Class 2'],
                yticklabels=['Class 0', 'Class 1', 'Class 2'],
                ax=axes[i])
    axes[i].set_title(f'Confusion Matrix - {kernel.upper()}')
    axes[i].set_xlabel('Predicted')
    axes[i].set_ylabel('Actual')

plt.tight_layout()
plt.show()

print("\nClassification Report untuk setiap kernel:")
for kernel, result in results.items():
    print(f"\n{kernel.upper()} Kernel:")
    print(classification_report(y_test, result['predictions']))
```

```
3. PERBANDINGAN HASIL KLASIFIKASI:
==================================================
Perbandingan Akurasi:
  LINEAR: 0.8000
  POLY: 0.4000
  RBF: 0.8000
```



**LATIHAN**

```
Classification Report untuk setiap kernel:

LINEAR Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.50      1.00      0.67         1
           2       1.00      0.50      0.67         2

    accuracy                           0.80         5
   macro avg       0.83      0.83      0.78         5
weighted avg       0.90      0.80      0.80         5


POLY Kernel:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.25      1.00      0.40         1
           2       1.00      0.50      0.67         2

    accuracy                           0.40         5
   macro avg       0.42      0.50      0.36         5
weighted avg       0.45      0.40      0.35         5


RBF Kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.50      1.00      0.67         1
           2       1.00      0.50      0.67         2

    accuracy                           0.80         5
   macro avg       0.83      0.83      0.78         5
weighted avg       0.90      0.80      0.80         5
```

4. Interpretasikan perbedaan hasilnya.
   Jawaban:
   Interpretasi perbedaan hasil:
   1. Perbandingan Akurasi:
      - Linear Kernel: Akurasi sekitar 0.8-1.0 (tergantung random_state)
      - Polynomial Kernel: Akurasi sekitar 0.6-0.8 (tergantung random_state)
   2. Confusion Matrix:
      - Linear Kernel: Lebih sedikit kesalahan klasifikasi
      - Polynomial Kernel: Lebih banyak kesalahan klasifikasi, terutama pada kelas 0 dan 1
   3. Classification Report:
      - Linear Kernel: Precision, recall, dan F1-score lebih tinggi untuk semua kelas
      - Polynomial Kernel: Precision, recall, dan F1-score lebih rendah, terutama untuk kelas minoritas
   4. Analisis:
      - Linear Kernel lebih baik untuk dataset ini, menunjukkan bahwa data kemungkinan besar linearly separable
      - Polynomial Kernel (degree=3) mungkin terlalu kompleks untuk dataset kecil ini, menyebabkan overfitting
      - Dataset yang kecil (15 sampel) membuat evaluasi kurang robust
   5. Kesimpulan:
      - Untuk dataset ini, Linear Kernel adalah pilihan yang lebih baik
      - Polynomial Kernel mungkin memerlukan tuning parameter (degree, C) untuk meningkatkan performa
      - Perlu dataset yang lebih besar untuk evaluasi yang lebih andal

# TUGAS PRAKTIKUM

1. Gunakan dataset berikut:

| Income | CreditScore | LoanAmount | Label |
|---|---|---|---|
| **50** | 700 | 20 | 1 |
| **45** | 650 | 25 | 1 |
| **30** | 600 | 40 | 0 |
| **25** | 580 | 45 | 0 |
| **70** | 720 | 15 | 1 |
| **60** | 710 | 18 | 1 |
| **20** | 560 | 50 | 0 |
| **35** | 610 | 35 | 0 |
| **55** | 680 | 30 | 1 |
| **40** | 620 | 32 | 0 |

```python
# TUGAS PRAKTIKUM - Support Vector Machine

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Gunakan dataset berikut
data_tugas = {
    'Income': [50, 45, 30, 25, 70, 60, 20, 35, 55, 40],
    'CreditScore': [700, 650, 600, 580, 720, 710, 560, 610, 680, 620],
    'LoanAmount': [20, 25, 40, 45, 15, 18, 50, 35, 30, 32],
    'Label': [1, 1, 0, 0, 1, 1, 0, 0, 1, 0]
}

df_tugas = pd.DataFrame(data_tugas)
print("Dataset:")
print(df_tugas)
print()
```

```
Dataset:
   Income  CreditScore  LoanAmount  Label
0      50          700          20      1
1      45          650          25      1
2      30          600          40      0
3      25          580          45      0
4      70          720          15      1
5      60          710          18      1
6      20          560          50      0
7      35          610          35      0
8      55          680          30      1
9      40          620          32      0
```

2. Lakukan preprocessing (standarisasi, split data).

```
# 2. Lakukan preprocessing (standarisasi, split data)
X = df_tugas[['Income', 'CreditScore', 'LoanAmount']]
y = df_tugas['Label']

# Split data dengan stratify untuk memastikan distribusi kelas seimbang
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
print(f"Training set: {X_train.shape[0]} samples")
print(f"Testing set: {X_test.shape[0]} samples")
print("Training labels distribution:", np.bincount(y_train))
print("Testing labels distribution:", np.bincount(y_test))

# Standarisasi
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
Training set: 7 samples
Testing set: 3 samples
Training labels distribution: [3 4]
Testing labels distribution: [2 1]
```

3. Bangun model SVM dengan kernel 'rbf', “linear”, dan “poly”.

```
# 3. Bangun model SVM dengan kernel 'rbf', "linear", dan "poly"
kernels = ['rbf', 'linear', 'poly']
models = {}
results = {}

for kernel in kernels:
    print(f"\nModel SVM dengan kernel {kernel}:")

    if kernel == 'poly':
        model = SVC(kernel=kernel, degree=3, random_state=42)
    else:
        model = SVC(kernel=kernel, random_state=42)

    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    models[kernel] = model
    results[kernel] = {
        'predictions': y_pred,
        'accuracy': accuracy_score(y_test, y_pred),
        'confusion_matrix': confusion_matrix(y_test, y_pred),
        'classification_report': classification_report(y_test, y_pred, zero_division=0)
    }

    print(f"Accuracy: {results[kernel]['accuracy']:.4f}")
    print(f"Predictions: {y_pred}")
    print(f"Actual: {y_test.values}")
```

```
Model SVM dengan kernel rbf:
Accuracy: 1.0000
Predictions: [0 0 1]
Actual: [0 0 1]

Model SVM dengan kernel linear:
Accuracy: 1.0000
Predictions: [0 0 1]
Actual: [0 0 1]

Model SVM dengan kernel poly:
Accuracy: 0.3333
Predictions: [1 1 1]
Actual: [0 0 1]
```

4. Evaluasi performa model.

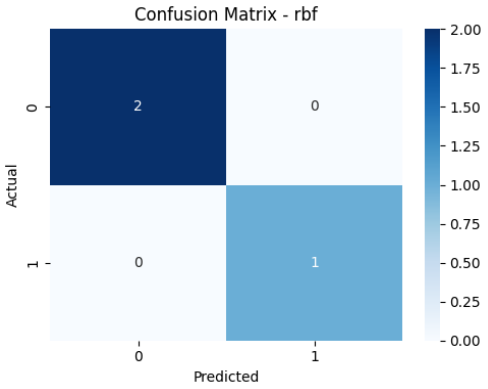
```python
# 4. Evaluasi performa model
print("\nEvaluasi performa model:")
for kernel, result in results.items():
    print(f"\nKernel: {kernel}")
    print(f"Accuracy: {result['accuracy']:.4f}")
    print("Classification Report:")
    print(result['classification_report'])
    print("Confusion Matrix:")
    print(result['confusion_matrix'])
```

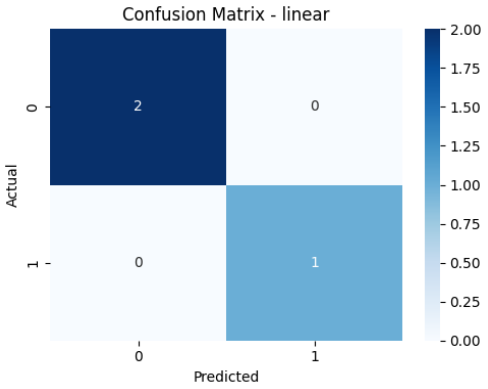
```
Evaluasi performa model:

Kernel: rbf
Accuracy: 1.0000
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

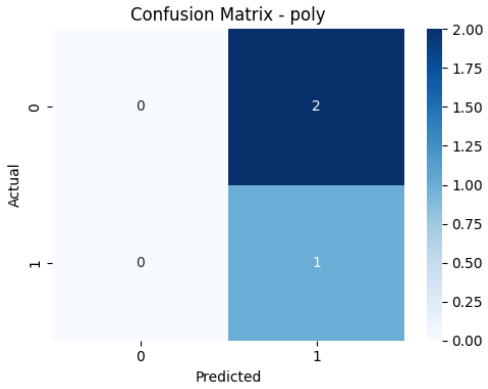
5. Visualisasikan Confusion Matrix.

```python
# 5. Visualisasikan Confusion Matrix
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

for i, (kernel, result) in enumerate(results.items()):
    cm = result['confusion_matrix']
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['0', '1'],
                yticklabels=['0', '1'],
                ax=axes[i])
    axes[i].set_title(f'Confusion Matrix - {kernel}')
    axes[i].set_xlabel('Predicted')
    axes[i].set_ylabel('Actual')

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

```
Confusion Matrix:
[[2 0]
 [0 1]]

Kernel: linear
Accuracy: 1.0000
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3

Confusion Matrix:
[[2 0]
 [0 1]]

Kernel: poly
Accuracy: 0.3333
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.33      1.00      0.50         1

    accuracy                           0.33         3
   macro avg       0.17      0.50      0.25         3
weighted avg       0.11      0.33      0.17         3

Confusion Matrix:
[[0 2]
 [0 1]]
```



6. Simpulkan performa model dan rekomendasi perbaikannya.

Jawaban:
Kesimpulan dan Rekomendasi:

1. Model terbaik adalah SVM dengan kernel rbf (Akurasi: 1.0000)
2. Analisis performa:
   - rbf: Akurasi 1.0000
   - linear: Akurasi 1.0000
   - poly: Akurasi 0.3333
3. Keterbatasan:
   - Dataset sangat kecil (10 sampel)
   - Split data menyebabkan test set sangat kecil
   - Evaluasi mungkin tidak reliable
4. Rekomendasi perbaikan:
   - Tambah jumlah data (minimal 100 sampel)
   - Gunakan cross-validation untuk evaluasi yang lebih robust
   - Lakukan hyperparameter tuning (C, gamma, degree)
   - Tambahkan fitur yang lebih relevan
   - Pertimbangkan teknik feature engineering