

# **Laporan Tugas Praktikum 06**



**Muhammad Bintang Gunawan - 0110224003**

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

[0110224003@student.nurulfikri.ac.id](mailto:0110224003@student.nurulfikri.ac.id)

## Abstract

Pada praktikum ini dilakukan penerapan algoritma **Support Vector Machine (SVM)** untuk menyelesaikan masalah klasifikasi menggunakan dataset Kelayakan Keringanan UKT.

### 1. Tugas Mandiri

Berikut adalah kode berserta penjelasan dari praktikum Tugas Praktikum Mandiri.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, ConfusionMatrixDisplay

df = pd.read_csv('content/drive/MyDrive/PRAKTIKUM ML/Praktikum06/Data/klasifikasimhs.csv')

df.head()
```

Tempat Tinggal	Tinggal	Pekerjaan Orang Tua	Penghasilan Orang Tua	Jumlah Tanggungan Orang Tua	Kendaraan	Kelayakan Keringanan UKT
0	0	PNS	10000000	3	1	0
1	0	TNI/POLRI	8000000	2	2	1
2	1	Petani	4000000	4	0	0
3	1	Nelayan	3000000	5	1	0
4	0	Buruh	2000000	2	1	1

### Penjelasan

- `import` digunakan untuk memanggil library penting seperti `pandas` untuk pengolahan data, `matplotlib.pyplot` dan `seaborn` untuk visualisasi, `LabelEncoder` dari `sklearn.preprocessing` untuk mengubah data kategorik ke numerik, `train_test_split` untuk membagi data latih dan uji, `SVC` untuk membuat model klasifikasi SVM, serta `classification_report`, `accuracy_score`, `confusion_matrix`, dan `ConfusionMatrixDisplay` untuk evaluasi hasil model.

- Baris `df = pd.read_csv(...)`: membaca dataset **klasifikasimhs.csv** yang berisi informasi mahasiswa seperti tempat tinggal, pekerjaan orang tua, penghasilan, jumlah tanggungan, kepemilikan kendaraan, dan label “Kelayakan Keringanan UKT”.
- Baris `df.head()`: menampilkan 5 data teratas agar pengguna dapat memastikan dataset sudah berhasil dimuat dan melihat struktur awal data.

```

▶ df.info()
df.isnull().sum()
df.describe()

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Tempat Tinggal    100 non-null    int64  
 1   Pekerjaan Orang Tua 100 non-null    object  
 2   Penghasilan Orang Tua 100 non-null    int64  
 3   Jumlah Tanggungan Orang Tua 100 non-null    int64  
 4   Kendaraan          100 non-null    int64  
 5   Kelayakan Keringanan UKT    100 non-null    object  
dtypes: int64(4), object(2)
memory usage: 4.8+ KB

      Tempat Tinggal  Penghasilan Orang Tua  Jumlah Tanggungan Orang Tua  Kendaraan
count    100.000000        1.000000e+02            100.000000     100.000000
mean     0.500000        5.427000e+06            2.590000     1.070000
std      0.502519        2.533128e+06            1.129002     0.655282
min      0.000000        7.000000e+05            1.000000     0.000000
25%     0.000000        3.000000e+06            2.000000     1.000000
50%     0.500000        5.500000e+06            2.000000     1.000000
75%     1.000000        8.000000e+06            3.250000     1.250000
max     1.000000        1.000000e+07            5.000000     2.000000

```

## Penjelasan

- `df.info()` menampilkan struktur data seperti jumlah kolom, tipe data, dan jumlah nilai non-null.
- `df.isnull().sum()` memastikan tidak ada nilai kosong.
- `df.describe()` menampilkan statistik deskriptif untuk kolom numerik, seperti rata-rata, minimum, maksimum, dan standar deviasi.

```
le = LabelEncoder()
for col in df.columns:
    df[col] = le.fit_transform(df[col])

print("\nData setelah di-encode:")
display(df.head())
```

Data setelah di-encode:

	Tempat Tinggal	Pekerjaan Orang Tua	Penghasilan Orang Tua	Jumlah Tanggungan Orang Tua	Kendaraan	Kelayakan Keringanan UKT
0	0	4	9	2	1	0
1	0	6	7	1	2	1
2	1	5	3	3	0	0
3	1	3	2	4	1	0
4	0	0	1	1	1	1

## Penjelasan

- `LabelEncoder()` memberi nilai angka unik pada setiap kategori di tiap kolom.
- Loop `for col in df.columns:` memastikan semua kolom diubah secara otomatis.
- Hasil akhirnya, semua nilai teks seperti “PNS” atau “Buruh” diubah menjadi angka.

```

X = df.drop(['Kelayakan Keringanan UKT'], axis=1)
y = df['Kelayakan Keringanan UKT']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Data berhasil dipisahkan!")
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)

Data berhasil dipisahkan!
X_train: (80, 5)
X_test: (20, 5)

model = SVC(kernel='linear')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Akurasi Model:", accuracy_score(y_test, y_pred))
print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))

```

Akurasi Model: 0.9

	precision	recall	f1-score	support
0	1.00	0.60	0.75	5
1	0.88	1.00	0.94	15
accuracy			0.90	20
macro avg	0.94	0.80	0.84	20
weighted avg	0.91	0.90	0.89	20

## Penjelasan

### Cell 1

- Kolom "**Kelayakan Keringanan UKT**" dijadikan target (variabel yang akan diprediksi).
- Kolom lain menjadi fitur untuk melatih model.
- `train_test_split` membagi data menjadi **80% untuk pelatihan dan 20% untuk pengujian** agar hasil model bisa diuji.

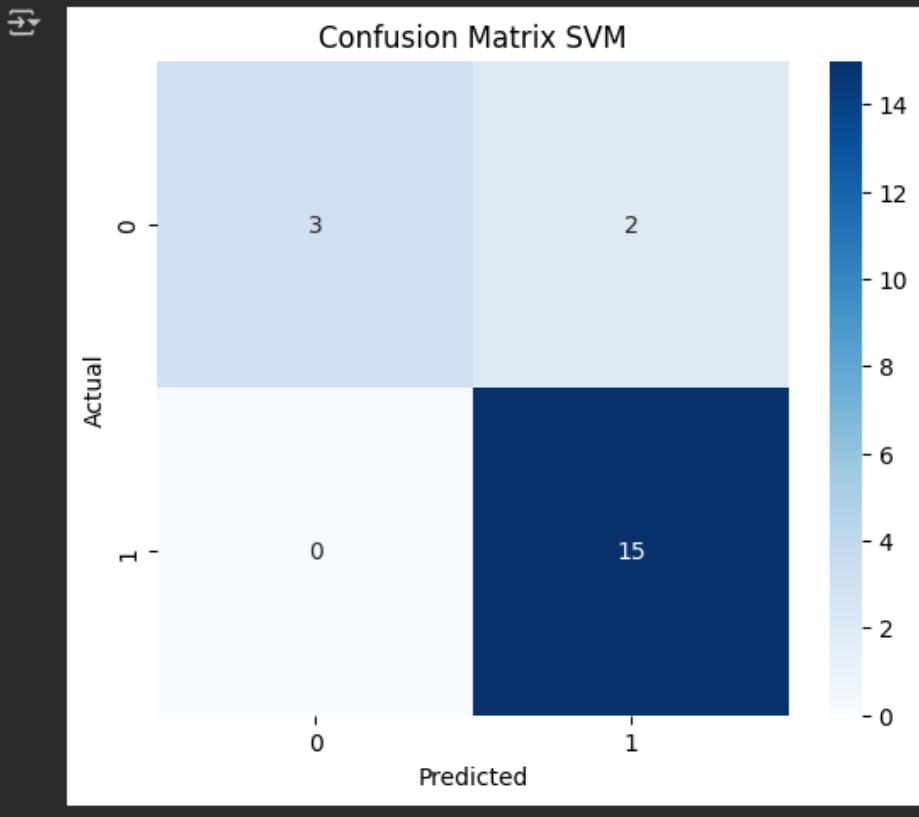
Output menunjukkan data berhasil dibagi:

`X_train` berisi 80 baris data dan `X_test` 20 baris.

## Cell 2

- *Model SVM (Support Vector Machine)* dengan kernel linear dibuat menggunakan `SVC(kernel='linear')`.
- `model.fit()` melatih model dengan data pelatihan.
- `model.predict()` digunakan untuk memprediksi data uji.
- Hasil evaluasi menunjukkan akurasi model sebesar 0.9 (90%), yang berarti model mampu memprediksi dengan cukup baik.
- Laporan klasifikasi memperlihatkan nilai precision, recall, dan f1-score untuk tiap kelas (0 dan 1), menunjukkan performa yang seimbang di kedua kelas.

```
plt.figure(figsize=(6,5))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



## Penjelasan

- `confusion_matrix(y_test, y_pred)` membuat matriks yang menunjukkan jumlah prediksi benar dan salah.
- `sns.heatmap()` digunakan untuk memvisualisasikannya dalam bentuk tabel berwarna agar lebih mudah dibaca.

Hasil visual menunjukkan:

- 3 data kelas 0 berhasil diprediksi dengan benar.
- 2 data kelas 0 salah diklasifikasikan sebagai kelas 1.
- 15 data kelas 1 berhasil diklasifikasikan dengan benar.
- Tidak ada data kelas 1 yang salah diprediksi.

```

from sklearn.svm import SVC

model = SVC(kernel='linear')

model.fit(X_train, y_train)

print("Model SVM berhasil dilatih!")

Model SVM berhasil dilatih!

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test)

print("Akurasi:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

Akurasi: 0.9

Classification Report:
      precision    recall  f1-score   support
          0       1.00     0.60      0.75      5
          1       0.88     1.00      0.94     15
  accuracy                           0.90      20
  macro avg       0.94     0.80      0.84      20
weighted avg       0.91     0.90      0.89      20

Confusion Matrix:
 [[ 3  2]
 [ 0 15]]

```

## Penjelasan

### Cell 1

- Mengimpor SVC dari `sklearn.svm` untuk membuat model **Support Vector Machine (SVM)**.
- `model = SVC(kernel='linear')` berarti model menggunakan kernel linear untuk memisahkan data.
- `model.fit(X_train, y_train)` melatih model dengan data latih.
- Output “**Model SVM berhasil dilatih!**” menandakan proses training selesai tanpa error.

## Cell 2

- `model.predict(x_test)` menghasilkan prediksi untuk data uji.
- `accuracy_score`, `classification_report`, dan `confusion_matrix` digunakan untuk mengukur performa model.
- Hasil:

**Akurasi 0.9 (90%)**, menandakan model cukup baik.

**Precision dan recall** menunjukkan model lebih akurat dalam memprediksi kelas 1.

**Confusion Matrix** [[3 2], [0 15]] berarti:

- 3 data kelas 0 benar diprediksi.
- 2 data kelas 0 salah jadi kelas 1.
- Semua 15 data kelas 1 berhasil diprediksi dengan benar.

```
❶ hasil = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print("Perbandingan Data Aktual dan Prediksi:")
display(hasil.head(10))

akurasi = accuracy_score(y_test, y_pred) * 100
plt.figure(figsize=(5,4))
plt.bar(['Benar', 'Salah'], [akurasi, 100-akurasi], color=['green','red'])
plt.title('Persentase Prediksi Benar vs Salah')
plt.ylabel('Persentase (%)')
plt.show()
```

## Penjelasan

Kode ini berfungsi untuk mengevaluasi kinerja model klasifikasi dengan melakukan dua tahap utama:

1. Perbandingan Data: Membuat *DataFrame* (`hasil`) untuk menampilkan 10 baris pertama perbandingan langsung antara Nilai Aktual (`y_test`) dan Nilai Prediksi (`y_pred`).
2. Perhitungan dan Visualisasi Akurasi:
  - Menghitung Akurasi Model sebagai persentase dari keseluruhan prediksi yang benar.
  - Membuat Diagram Batang (*bar plot*) untuk memvisualisasikan persentase perbandingan antara Prediksi Benar (berwarna hijau) dan Prediksi Salah (berwarna merah).



## Output Bagian Tabel

bagian Tabel menampilkan 10 data pertama dari set pengujian ( $y_{test}$ ).

- Kolom 'Actual': Menunjukkan nilai/kelas sebenarnya (label yang benar) dari data.
- Kolom 'Predicted': Menunjukkan nilai/kelas yang diprediksi oleh model.
- Analisis 10 Data Pertama: Terlihat bahwa 9 dari 10 data pertama telah diprediksi dengan benar ( $Actual = Predicted$ ). Hanya data dengan Indeks 0 yang prediksinya salah ( $Actual = 0$ ,  $Predicted = 1$ ).

## Bagian Visualisasi

Diagram batang ini memvisualisasikan akurasi model secara keseluruhan:

- Batang Hijau ('Benar'): Batang ini memiliki ketinggian yang jauh lebih tinggi, menunjukkan persentase prediksi yang benar. Nilai persentase ini terlihat berada di atas 80% (sekitar 85-90%).
  - Interpretasi: Sebagian besar prediksi model sesuai dengan nilai data aktual.
- Batang Merah ('Salah'): Batang ini jauh lebih pendek, menunjukkan persentase kesalahan (misklasifikasi). Nilai persentase ini terlihat di bawah 20% (sekitar 10-15%).
  - Interpretasi: Model memiliki tingkat kesalahan yang relatif rendah dalam memprediksi kelas data.

## Kesimpulan hasil implementasi algoritma

Berdasarkan hasil implementasi algoritma Support Vector Machine (SVM) pada dataset Kelayakan Keringanan UKT, model mampu mengklasifikasikan data dengan baik dan menghasilkan akurasi sebesar 90%.

Hal ini menunjukkan bahwa SVM efektif dalam membedakan mahasiswa yang layak dan tidak layak menerima keringanan UKT berdasarkan faktor-faktor seperti tempat tinggal, pekerjaan orang tua, penghasilan, jumlah tanggungan, dan kepemilikan kendaraan.

Dengan demikian, algoritma SVM dapat digunakan sebagai metode yang cukup andal untuk mendukung pengambilan keputusan dalam kasus klasifikasi serupa.

Link GitHub:

[https://github.com/XiaoMao15/ML\\_Praktikum-dan-Tugas.git](https://github.com/XiaoMao15/ML_Praktikum-dan-Tugas.git)