Tugas Mandiri 6: Machine Learning Support Vector Machine (SVM)

Safinatunnajah - 0110222234

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

E-mail: safi22234ti@student.nurulfikri.ac.id

Abstract. Laporan ini bertujuan untuk mengklasifikasikan rentang harga ponsel berdasarkan karakteristik spesifikasi hardware dengan menggunakan algoritma Support Vector Machine (SVM). Dataset yang digunakan berasal dari kaggle dan berisi berbagai fitur seperti kapasitas RAM, daya baterai, berat ponsel, serta atribut lainnya. Data terlebih dahulu melalui tahap pra-proses yang mencakup standarisasi menggunakan StandardScaler untuk menyamakan skala setiap fitur. Selanjutnya, dilakukan pencarian parameter terbaik melalui metode Grid Search Cross Validation agar diperoleh kombinasi hyperparameter SVM yang paling optimal. Model terbaik kemudian dievaluasi menggunakan data uji dan menghasilkan nilai akurasi yang tinggi dengan distribusi prediksi yang ditunjukkan melalui confusion matrix serta classification report. Visualisasi 3D berdasarkan fitur RAM, battery power, dan mobile weight memperlihatkan persebaran kelas harga yang relatif terpisah, menunjukkan kemampuan SVM dalam mempelajari pola data dengan baik.

1. Pendahuluan

Perkembangan teknologi smartphone yang pesat membuat pasar ponsel semakin kompetitif, dengan variasi harga yang sangat bergantung pada spesifikasi perangkat kerasnya. Oleh karena itu, analisis terhadap faktor-faktor yang mempengaruhi rentang harga menjadi penting, baik bagi produsen dalam menentukan strategi produk maupun bagi konsumen dalam memahami nilai suatu perangkat. Dalam penelitian ini digunakan algoritma Support Vector Machine (SVM), salah satu metode supervised learning yang efektif untuk klasifikasi data dengan batas pemisah non-linear. Dataset ponsel yang digunakan berisi berbagai atribut seperti kapasitas RAM, daya baterai, dan berat ponsel, yang kemudian diolah dan standarisasi agar model dapat belajar secara optimal. Parameter SVM disesuaikan menggunakan Grid Search untuk menemukan konfigurasi terbaik. Hasil pelatihan model menunjukkan bahwa SVM mampu membedakan kelas

harga ponsel dengan tingkat akurasi yang tinggi, yang juga divisualisasikan melalui grafik 3D untuk memberikan gambaran hubungan antara fitur-fitur utama dan kategori harga.

2. Proses pengerjaan

Berikut tahapan pengerjaannya:

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

Kodingan di atas digunakan untuk menyiapkan lingkungan analisis data dan pemodelan klasifikasi menggunakan algoritma Support Vector Machine (SVM). Library pandas dan numpy digunakan untuk memanipulasi serta mengolah data numerik, sementara matplotlib.pyplot dan seaborn membantu menampilkan visualisasi seperti grafik dan heatmap untuk analisis hasil. Modul train_test_split dari sklearn.model_selection digunakan untuk membagi dataset menjadi data latih dan data uji agar evaluasi model lebih objektif, sedangkan GridSearchCV berfungsi melakukan pencarian kombinasi parameter terbaik bagi model SVM. StandardScaler dipakai untuk menstandarkan fitur numerik agar semua memiliki skala yang sebanding sebelum proses pelatihan. Kelas SVC merupakan implementasi SVM untuk klasifikasi, dan modul classification_report, confusion_matrix, serta accuracy_score digunakan untuk mengevaluasi performa model berdasarkan akurasi, distribusi prediksi benar dan salah, serta metrik lainnya.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Kodingan di atas digunakan untuk mengimpor modul drive dari library google.colab. Dimana modul ini berisi fungsi-fungsi untuk mengakses Google Drive.

```
path = ('/content/drive/MyDrive/praktikum_ml/praktikum06')
```

Kodingan di atas digunakan untuk menyimpan lokasi folder kerja di Google Drive agar mudah diakses saat membuka atau menyimpan file selama menjalankan program di Google Colab.

```
import pandas as pd

df = pd.read_csv(path + '/data/train.csv')
df.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores
0	842	0	2.2	0	1	0	7	0.6	188	2
1	1021	1	0.5	1	0	1	53	0.7	136	3
2	563	1	0.5	1	2	1	41	0.9	145	5
3	615	1	2.5	0	0	0	10	0.8	131	6
4	1821	1	1.2	0	13	1	44	0.6	141	2

5 rows × 21 columns

Kodingan di atas digunakan untuk membaca dataset dalam format CSV. Baris pertama import pandas as pd berfungsi mengimpor pustaka pandas dan memberinya alias pd agar mudah dipanggil.

df = pd.read_csv(path + '/data/train.csv') membaca file bernama train.csv yang berada di dalam folder data pada direktori yang disimpan dalam variabel path, dan menyimpannya ke dalam variabel df sebagai sebuah DataFrame, yaitu struktur data dua dimensi mirip tabel. df.head() menampilkan lima baris pertama dari dataset tersebut sehingga pengguna bisa melihat sekilas isi dan struktur data seperti nama kolom, tipe data, serta contoh nilai di tiap kolom.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

Kodingan di atas digunakan untuk melihat kondisi dan tipe data dari dataset.

```
df.describe()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000

8 rows × 21 columns

df.describe() digunakan untuk menampilkan statistik deskriptif dari seluruh kolom numerik dalam DataFrame df. Fungsi ini secara otomatis menghitung ukuran-ukuran penting seperti count (jumlah data yang tidak kosong), mean (rata-rata), std (standar deviasi atau seberapa tersebar datanya), min (nilai terkecil), 25%, 50%, 75% (kuartil atau pembagi distribusi data), dan max (nilai terbesar). Hasilnya memberi gambaran umum tentang distribusi, rentang nilai, serta kemungkinan adanya outlier pada dataset.

```
df.isnull().sum()
```

	0
battery_power	0
blue	0
clock_speed	0
dual_sim	0
fc	0
four_g	0
int_memory	0
m_dep	0
mobile_wt	0
n_cores	0

Kodingan di atas digunakan untuk mengecek jumlah data kosong (missing values) di setiap kolom DataFrame df. Dapat dilihat pada hasilnya 0 semua yang artinya, data tersebut tidak memiliki missing value.

```
X = df.drop(columns=['price_range'])
y = df['price_range']
```

Kodingan di atas digunakan untuk memisahkan fitur (variabel input) dan target (variabel output) dari dataset sebelum proses pelatihan model machine learning.

X = df.drop(columns=['price_range']) membuat DataFrame baru bernama X yang berisi semua kolom dari df kecuali kolom price_range, karena kolom tersebut dianggap sebagai label atau hasil yang ingin diprediksi.

y = df['price_range'] menyimpan kolom price_range saja ke dalam variabel y, yang berperan sebagai target klasifikasi.

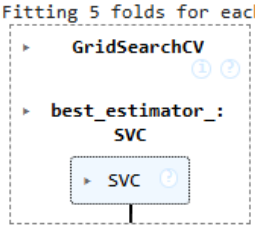
```
# Standardisasi
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, stratify=y, random_state=42)
```

Kodingan di atas digunakan untuk standarisasi fitur dan pembagian dataset.

StandardScaler() dari sklearn.preprocessing digunakan untuk menstandarkan seluruh variabel numerik dalam X agar tiap kolom memiliki rata-rata 0 dan standar deviasi 1, sehingga semua fitur berada pada skala yang sebanding dan tidak ada satu fitur dengan rentang nilai besar yang mendominasi proses pembelajaran model SVM. Proses ini dilakukan dengan scaler.fit_transform(X) yang menghitung parameter skala dan langsung menerapkannya ke seluruh data. Selanjutnya, perintah train_test_split(...) memecah dataset menjadi dua bagian: 80% untuk pelatihan (X_train, y_train) dan 20% untuk pengujian (X_test, y_test). Parameter stratify=y memastikan proporsi tiap kelas pada target y tetap seimbang di data latih dan data uji, sementara random_state=42 membuat hasil pembagian selalu konsisten setiap kali kode dijalankan.

```
# Grid search parameter SVM
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}

svm = SVC()
grid_search = GridSearchCV(
    svm, param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=1
)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```



Kodingan di atas digunakan untuk melakukan pencarian parameter terbaik pada model Support Vector Machine menggunakan teknik Grid Search dengan validasi silang (cross-validation).
param_grid berisi kombinasi nilai parameter yang akan diuji: C menentukan tingkat regularisasi, kernel menentukan jenis fungsi pemisah yang digunakan, dan gamma mengontrol seberapa jauh pengaruh satu data terhadap lainnya untuk kernel non-linear.
svm = SVC() membuat model SVM dasar dari sklearn.svm. GridSearchCV menjalankan model tersebut berkali-kali untuk setiap kombinasi parameter di param_grid, cv=5 artinya setiap kombinasi diuji menggunakan 5-fold cross-validation agar hasilnya lebih stabil. scoring='accuracy' menilai performa berdasarkan tingkat akurasi, n_jobs=-1 memanfaatkan seluruh inti CPU agar proses lebih cepat. verbose=1 menampilkan progres. grid_search.fit(X_train, y_train) mengeksekusi seluruh pencarian dan menentukan kombinasi parameter terbaik berdasarkan hasil evaluasi cross-validation.

```
# Model terbaik
best_model = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
```

Kodingan di atas digunakan untuk menampilkan model SVM terbaik. Setelah GridSearchCV selesai menguji semua kombinasi parameter, atribut best_estimator_ menyimpan objek model SVM dengan kombinasi parameter yang menghasilkan akurasi tertinggi selama validasi silang, dan variabel best_model menyimpannya agar bisa digunakan langsung untuk prediksi selanjutnya. Sementara grid_search.best_params_ berisi pasangan nilai parameter terbaik

(seperti C, kernel, dan gamma) yang dipilih otomatis berdasarkan performa terbaik. print("Best Parameters:", grid_search.best_params_), program menampilkan kombinasi parameter paling optimal.

```
# Evaluasi
y_pred = best_model.predict(X_test)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Akurasi Model:", accuracy_score(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       100
           1       0.97      0.96      0.96       100
           2       0.96      0.97      0.97       100
           3       0.98      0.99      0.99       100

    accuracy                           0.97       400
   macro avg       0.98      0.98      0.97       400
weighted avg       0.98      0.97      0.97       400

Akurasi Model: 0.975
```

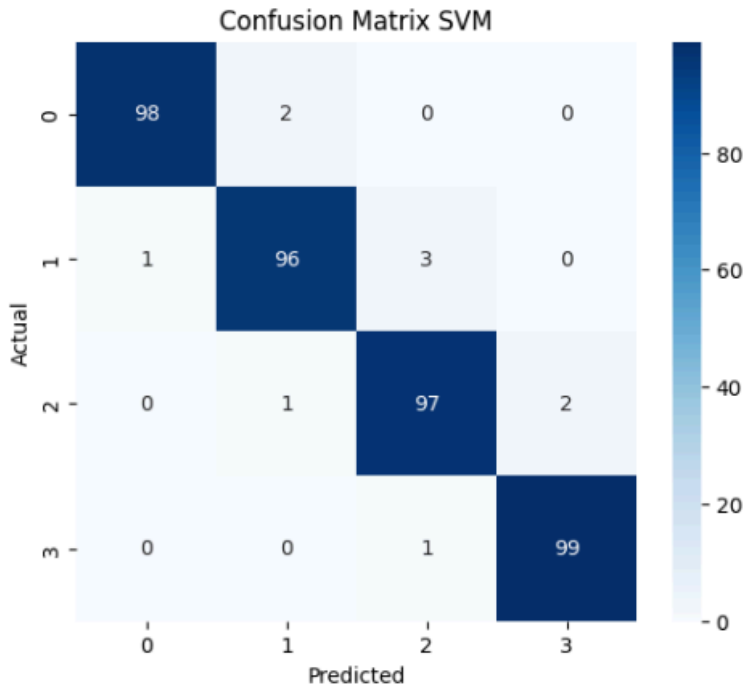
Kodingan di atas digunakan untuk mengevaluasi performa model SVM terbaik yang telah diperoleh dari proses Grid Search.

y_pred = best_model.predict(X_test) menghasilkan prediksi kelas dari data uji berdasarkan model SVM yang sudah dilatih sebelumnya.

classification_report(y_test, y_pred) menampilkan ringkasan metrik evaluasi seperti precision, recall, f1-score, dan support untuk setiap kelas target, yang memberi gambaran detail tentang seberapa baik model mengklasifikasikan masing-masing kategori.

accuracy_score(y_test, y_pred) menghitung proporsi prediksi yang benar terhadap seluruh data uji, dan hasilnya dicetak dengan label “Akurasi Model”. Secara keseluruhan, kode ini menunjukkan sejauh mana model SVM mampu memprediksi price_range dengan tepat pada data yang belum pernah dilihat sebelumnya.

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



Kodingan di atas digunakan untuk menampilkan confusion matrix hasil prediksi model SVM dalam bentuk visual yang mudah dibaca.

cm = confusion_matrix(y_test, y_pred) membuat tabel yang membandingkan antara kelas sebenarnya (y_test) dan kelas hasil prediksi (y_pred), sehingga setiap sel dalam matriks menunjukkan jumlah data yang diklasifikasikan benar maupun salah untuk tiap kategori.

plt.figure(figsize=(6,5)) menentukan ukuran gambar, dan sns.heatmap(cm, annot=True, fmt='d', cmap='Blues') memvisualisasikan matriks tersebut menggunakan heatmap berwarna biru dengan angka di setiap kotaknya. Sumbu X diberi label “Predicted” untuk menunjukkan hasil klasifikasi model, sedangkan sumbu Y “Actual” menunjukkan label sebenarnya dari data.

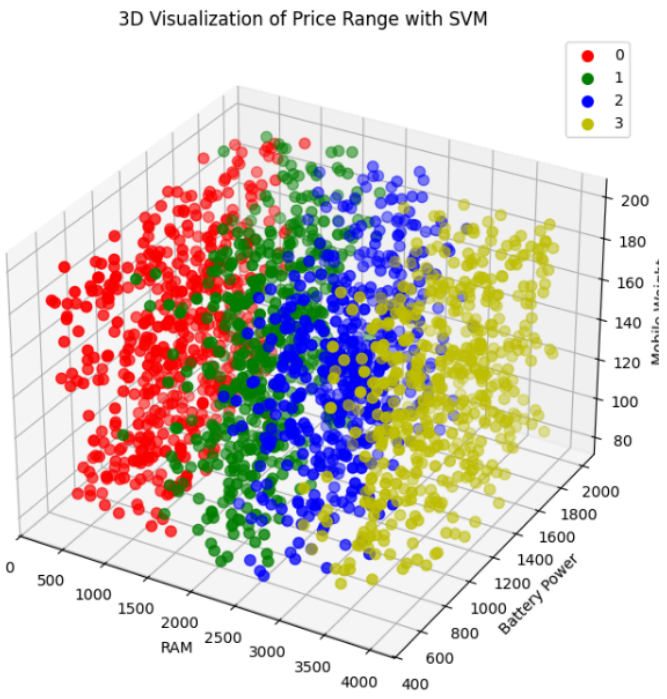
plt.show() menampilkan plot ke layar. Visualisasi ini membantu mengenali kelas mana yang sering salah diprediksi dan seberapa baik model membedakan antar kelas.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from mpl_toolkits.mplot3d import Axes3D

# 3. Encode label (ubah teks jadi angka)
le = LabelEncoder()
df['price_range_encoded'] = le.fit_transform(df['price_range'])

# 8. Plot 3D hasil klasifikasi
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Warna untuk tiap kelas
colors = ['r', 'g', 'b', 'y'] # Added 'y' for the fourth class
labels = le.classes_

# Plot tiap spesies dengan warna berbeda
for i, price_range in enumerate(labels):
    subset = df[df['price_range_encoded'] == i]
    ax.scatter(
        subset['ram'], # Using 'ram' as an example feature
        subset ['battery_power'], # Using 'battery_power' as an example feature
        subset['mobile_wt'], # Using 'mobile_wt' as an example feature
        color=colors[i],
        label=price_range,
        s=50
    )

ax.set_xlabel('RAM')
ax.set_ylabel('Battery Power')
ax.set_zlabel('Mobile Weight')
ax.set_title('3D Visualization of Price Range with SVM')
ax.legend()
plt.show()
```

Kodingan di atas digunakan untuk memvisualisasikan persebaran data hasil klasifikasi SVM dalam bentuk grafik 3 dimensi berdasarkan tiga fitur utama, yaitu ram, battery_power, dan mobile_wt. Pertama,

LabelEncoder() dari sklearn.preprocessing mengubah nilai teks atau kategori pada kolom price_range menjadi angka melalui fit_transform, dan hasilnya disimpan di kolom baru price_range_encoded, agar dapat diproses secara numerik. Selanjutnya dibuat objek figur dan sumbu tiga dimensi menggunakan matplotlib dengan projection='3d'. Variabel colors menyimpan daftar warna untuk empat kelas harga, sementara labels = le.classes_ menyimpan nama asli masing-masing kelas sebelum di-encode. Kemudian dilakukan perulangan untuk setiap kelas harga (price_range), di mana data yang termasuk kelas tersebut disaring dari DataFrame dan ditampilkan sebagai titik-titik (scatter) di ruang tiga dimensi: sumbu X mewakili ram, sumbu Y mewakili battery_power, dan sumbu Z mewakili mobile_wt. Setiap kelas divisualisasikan dengan warna berbeda agar pola persebarannya mudah dibedakan. Hasil plot

ini memperlihatkan bagaimana data ponsel dari berbagai rentang harga terdistribusi di ruang tiga fitur tersebut, dan seberapa jelas pemisahan antar kelas yang mungkin bisa dipelajari oleh model SVM.



Visualisasi tiga dimensi tersebut memperlihatkan distribusi data ponsel berdasarkan tiga fitur utama (RAM, Battery Power, dan Mobile Weight) yang digunakan untuk mengelompokkan rentang harga atau price range menggunakan SVM. Setiap warna merepresentasikan kelas harga berbeda: merah untuk kelas 0 (harga terendah), hijau untuk kelas 1, biru untuk kelas 2, dan kuning untuk kelas 3 (harga tertinggi). Terlihat bahwa data cenderung membentuk pola terpisah secara bertahap dari kiri ke kanan sepanjang sumbu RAM, yang menunjukkan bahwa semakin besar kapasitas RAM, ponsel umumnya berada pada rentang harga lebih tinggi. Selain itu, peningkatan battery power juga tampak berkontribusi pada kenaikan kelas harga, meskipun tidak sejelas pengaruh RAM. Pola warna yang cukup teratur menunjukkan bahwa kombinasi ketiga fitur ini memiliki kemampuan diskriminatif yang baik untuk memisahkan tiap kelas harga, sehingga model SVM dapat mempelajari batas pemisah antar kelompok dengan cukup efektif.

Link Google Colab Praktikum Mandiri:
https://colab.research.google.com/drive/1lLAYn1FH8QvRqE3Pgiqb3y3HZeeB06iS#scrollTo=lObly4PqxkoW

Link Google Colab Praktikum di Kelas:
https://colab.research.google.com/drive/1N_Pzl2obsJemn6gf1ywt_T_m5njV5YhV#scrollTo=2Qznu7T-hhQK

Link Github:
https://github.com/safntunajah/PraktikumMandiri06.git