Tugas 6: Super Vector Machine

Aisya Az Zahra - 0110224053

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: azzahrasyaa1715@gmail.com

Abstract.

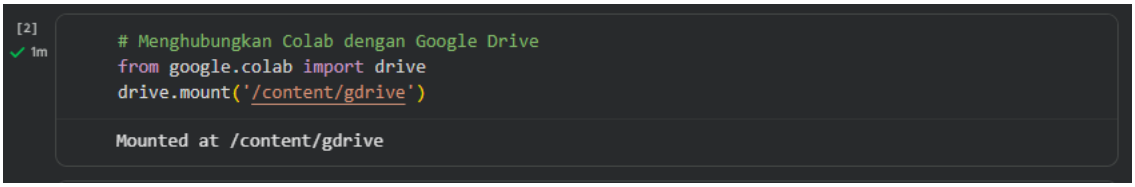
Laporan ini menjelaskan penerapan algoritma Support Vector Machine (SVM) untuk melakukan klasifikasi pada dataset Breast Cancer Wisconsin (Diagnostic) yang diambil dari Kaggle. Dataset ini berisi 569 data dengan 30 fitur numerik yang menggambarkan karakteristik inti sel tumor. Proses yang dilakukan meliputi tahap preprocessing (encoding dan normalisasi data), pembagian data menjadi data latih dan data uji, pelatihan model SVM dengan kernel RBF, serta evaluasi menggunakan metrik akurasi dan laporan klasifikasi. Hasil pengujian menunjukkan bahwa model menghasilkan akurasi sebesar 97.37%, yang berarti model mampu membedakan tumor jinak (Benign) dan ganas (Malignant) dengan sangat baik.

1.
```
[1]
✓ 4s
    # Import Library
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns

    from google.colab import drive
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.svm import SVC
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    import joblib
    import os
```

Penjelasan :

- import pandas as pd = Fungsinya untuk membaca dan mengolah data dalam bentuk tabel (DataFrame).
- import numpy as np = Mengimpor NumPy, library untuk perhitungan numerik seperti array dan operasi matematika.
- import matplotlib.pyplot as plt = Mengimpor modul matplotlib.pyplot untuk membuat grafik dan visualisasi data.
- import seaborn as sns = Mengimpor Seaborn, library visualisasi yang lebih menarik dan mudah digunakan daripada matplotlib.

- from google.colab import drive = Mengimpor fungsi drive dari Google Colab, digunakan untuk menghubungkan Colab dengan Google Drive agar file dataset bisa diakses langsung dari Drive.
- from sklearn.model_selection import train_test_split = Mengimpor fungsi train_test_split dari Scikit-Learn untuk membagi data menjadi data latih (training) dan data uji (testing).
- from sklearn.preprocessing import LabelEncoder, StandardScaler
  Mengimpor dua fungsi preprocessing:
  -LabelEncoder: mengubah data kategori (seperti 'M' dan 'B') menjadi angka.
  -StandardScaler: menormalkan skala data agar semua fitur memiliki rentang yang sama.
- from sklearn.svm import SVC = Mengimpor SVC (Support Vector Classifier), yaitu algoritma Support Vector Machine untuk klasifikasi data.
- from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
  Mengimpor fungsi-fungsi evaluasi model:
  - accuracy_score: menghitung akurasi model.
  - classification_report: menampilkan precision, recall, dan f1-score.
  - confusion_matrix: menampilkan jumlah prediksi benar dan salah dalam bentuk tabel.
- import joblib = Digunakan untuk menyimpan model yang sudah dilatih ke dalam file agar bisa digunakan lagi tanpa harus melatih ulang.
- import os = Library bawaan Python untuk mengatur file dan folder, misalnya membuat folder penyimpanan model.

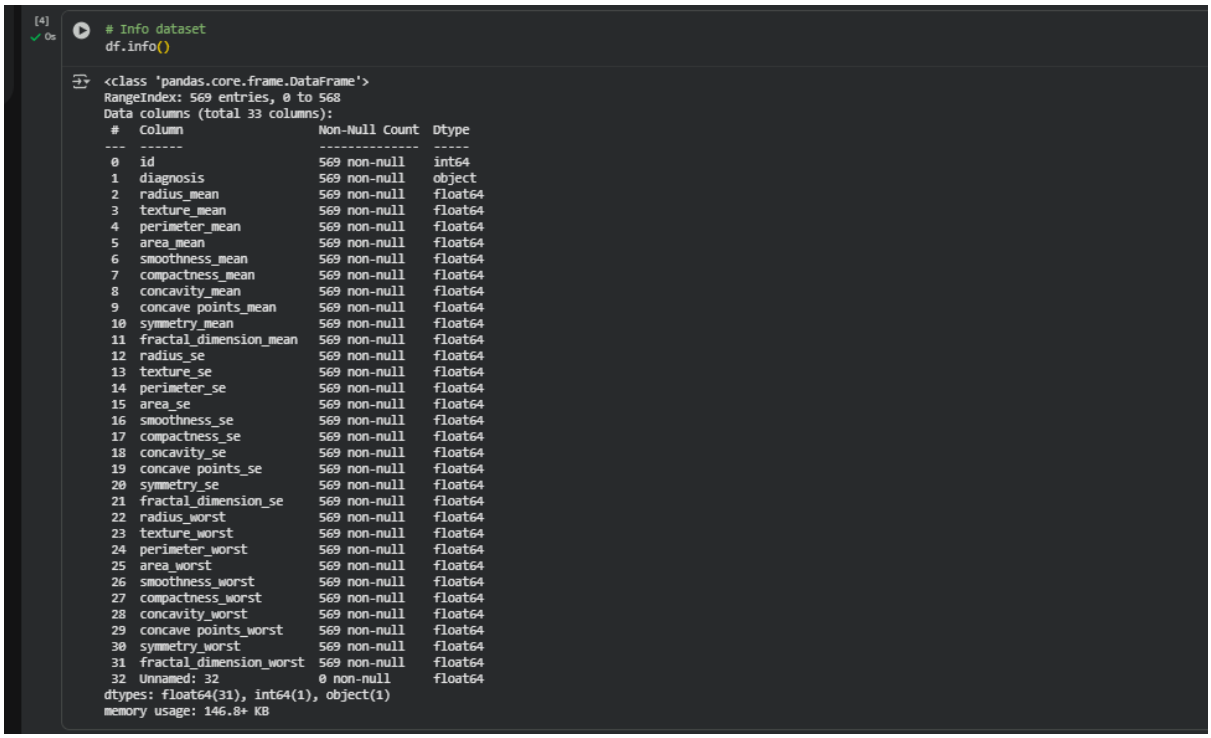
2. 

Penjelasan :
Kode :
Kode ini digunakan untuk menghubungkan google colab dengan google drive, sehingga kita bias membuka google colab melalui google drive.

Output :
Output menunjukan bahwa google colab sudah terhubung dengan google drive.

3. 

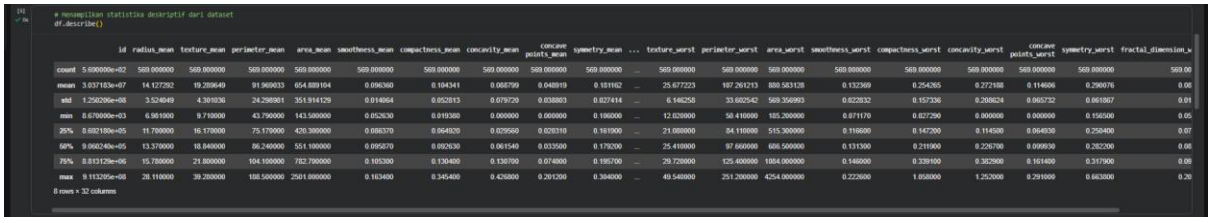
Penjelasan :
Kode tersebut digunakan untuk membaca dataset breast_cancer_wisconsin.csv dari Google Drive menggunakan fungsi pd.read_csv(). Data disimpan dalam variabel df sebagai DataFrame agar mudah diolah dan dianalisis pada tahap berikutnya.

4. 

```
[4]
0s
# Info dataset
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Penjelasan :
Kode diatas digunakan untuk melihat info dari dataset yang telah dimasukan sebelumnya.

Output :
Output diatas menunjukan Hasil output menunjukkan bahwa dataset memiliki 569 baris dan 33 kolom, dengan sebagian besar kolom bertipe data float64. Tidak terdapat nilai kosong (non-null) pada data, sehingga dataset siap digunakan untuk proses analisis dan pembuatan model.

5. 

```
# Menampilkan statistika deskriptif dari dataset
df.describe()
```

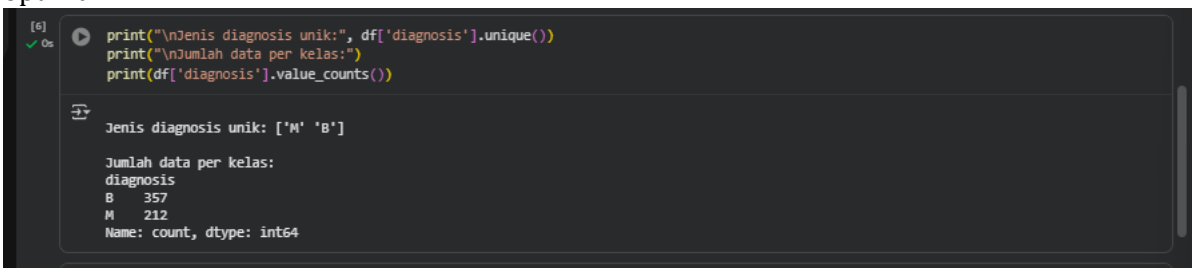
	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_w
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.000000	569.000000	568.000000	569.000000	569.000000	569.00
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606	0.290076	0.08
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.01
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.05
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.064930	0.250400	0.07
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.099930	0.282200	0.08
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.161400	0.317900	0.09
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.20

8 rows × 32 columns

Penjelasan :
Kode di atas digunakan untuk menampilkan statistik deskriptif dari seluruh kolom numerik pada dataset, seperti jumlah data, nilai rata-rata, standar deviasi, nilai minimum, maksimum, dan persentil.

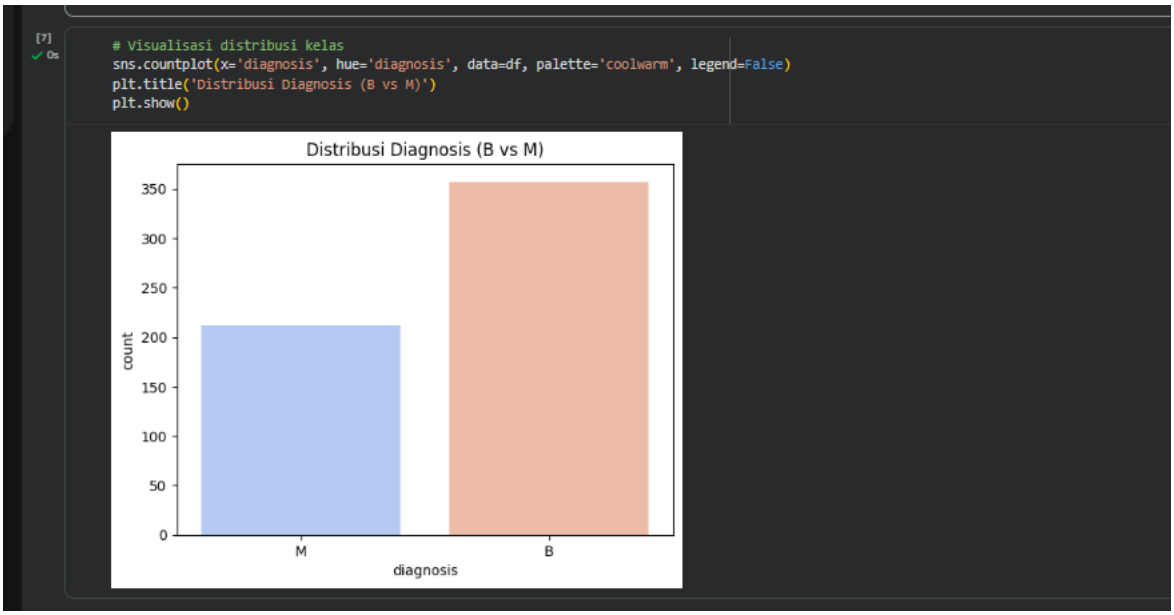
Output :
Hasil output menunjukkan bahwa setiap kolom memiliki 569 data valid tanpa nilai kosong. Nilai rata-rata (mean), standar deviasi (std), serta rentang minimum dan maksimum menggambarkan persebaran data tiap fitur. Perbedaan skala antar fitur

menandakan bahwa normalisasi data diperlukan agar model SVM dapat bekerja lebih optimal.

6. 

Penjelasan :
Kode ini digunakan untuk melihat nilai unik pada kolom diagnosis serta menghitung jumlah data pada setiap kelas. Perintah unique() menampilkan kategori label yang ada, sedangkan value_counts() menghitung berapa banyak data pada masing-masing kategori.

Output :
Hasil output menunjukkan bahwa terdapat dua jenis diagnosis, yaitu Malignant (M) dan Benign (B). Jumlah data untuk kelas Benign adalah 357, sedangkan kelas Malignant sebanyak 212. Hal ini berarti dataset memiliki distribusi kelas yang cukup seimbang untuk digunakan dalam model klasifikasi.

7. 

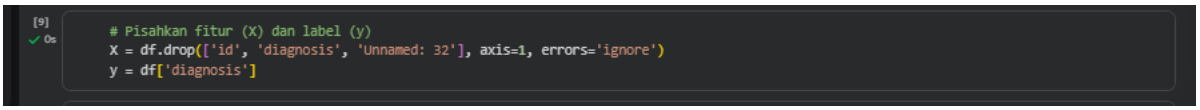
Penjelasan :
Kode di atas digunakan untuk menampilkan distribusi jumlah data pada setiap kelas diagnosis menggunakan diagram batang (countplot) dari library Seaborn. Parameter x='diagnosis' menentukan kolom yang akan divisualisasikan, sedangkan palette='coolwarm' memberikan warna berbeda untuk setiap kelas.

Output :
Hasil visualisasi menunjukkan bahwa kelas Benign (B) memiliki jumlah data lebih banyak dibandingkan kelas Malignant (M). Meskipun terdapat perbedaan jumlah data, distribusi

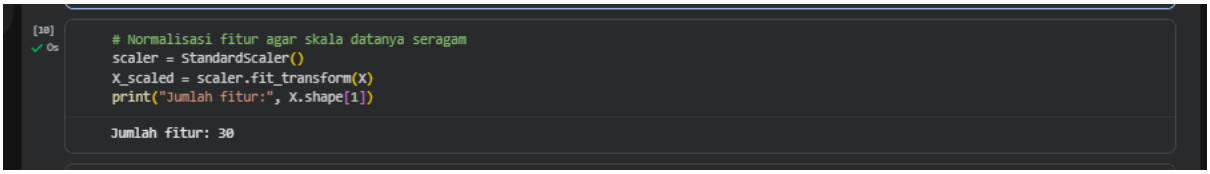
kedua kelas masih cukup seimbang, sehingga dataset tetap baik digunakan untuk proses pelatihan model Support Vector Machine (SVM).

8. 
```
[8]
✓ 0s
    # Ubah diagnosis (M=Malignant, B=Benign) menjadi numerik (1 dan 0)
    le = LabelEncoder()
    df['diagnosis'] = le.fit_transform(df['diagnosis'])
```

Penjelasan :
Kode di atas digunakan untuk mengubah data kategori pada kolom diagnosis menjadi nilai numerik menggunakan fungsi LabelEncoder dari Scikit-Learn. Nilai Malignant (M) diubah menjadi 1, dan Benign (B) diubah menjadi 0, agar data dapat diproses oleh model Support Vector Machine (SVM) yang hanya menerima input numerik.

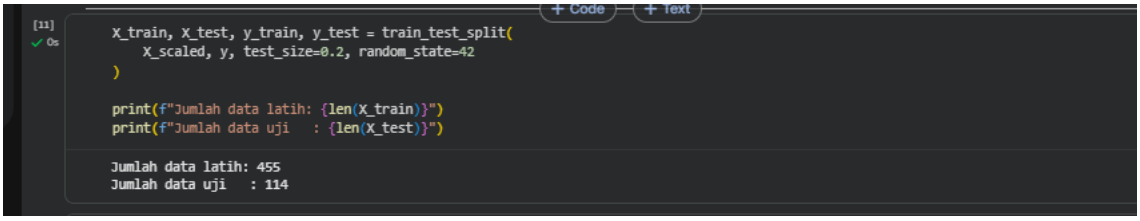
9. 
```
[9]
✓ 0s
    # Pisahkan fitur (X) dan label (y)
    X = df.drop(['id', 'diagnosis', 'Unnamed: 32'], axis=1, errors='ignore')
    y = df['diagnosis']
```

Penjelasan :
Kode di atas digunakan untuk memisahkan antara fitur (X) dan label target (y). Variabel X berisi seluruh kolom yang digunakan sebagai fitur input, sedangkan y berisi kolom diagnosis yang menjadi target prediksi. Kolom seperti id dan Unnamed: 32 dihapus karena tidak relevan dengan proses klasifikasi.

10. 
```
[10]
✓ 0s
    # Normalisasi fitur agar skala datanya seragam
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    print("Jumlah fitur:", X.shape[1])

    Jumlah fitur: 30
```

Penjelasan :
Kode di atas digunakan untuk menormalkan nilai setiap fitur agar berada pada skala yang sama menggunakan StandardScaler(). Normalisasi ini penting supaya model Support Vector Machine (SVM) dapat bekerja lebih optimal tanpa bias terhadap fitur dengan rentang nilai yang besar. Perintah print("Jumlah fitur:", X.shape[1]) digunakan untuk menampilkan jumlah kolom fitur dalam dataset.

Output :
Hasil output menunjukkan bahwa dataset memiliki 30 fitur yang akan digunakan sebagai variabel input dalam proses pelatihan model SVM.

11. 
```
[11]
✓ 0s
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42
    )

    print(f"Jumlah data latih: {len(X_train)}")
    print(f"Jumlah data uji   : {len(X_test)}")

    Jumlah data latih: 455
    Jumlah data uji   : 114
```

Penjelasan :
Kode di atas digunakan untuk membagi dataset menjadi data latih (training) dan data uji (testing) menggunakan fungsi train_test_split() dari Scikit-Learn. Parameter test_size=0.2

berarti 20% data digunakan untuk pengujian dan 80% untuk pelatihan. Parameter random_state=42 memastikan pembagian data tetap sama setiap kali dijalankan.
Output :
Hasil output menunjukkan bahwa terdapat 455 data latih dan 114 data uji. Data latih akan digunakan untuk melatih model Support Vector Machine (SVM), sedangkan data uji digunakan untuk mengukur performa model terhadap data yang belum pernah dilihat sebelumnya.

12.
```
[12]
✓ 0s
model = SVC(kernel='rbf', random_state=42)
model.fit(X_train, y_train)

SVC
SVC(random_state=42)
```

Penjelasan :
Kode di atas digunakan untuk membangun dan melatih model klasifikasi menggunakan algoritma Support Vector Machine (SVM). Parameter kernel='rbf' (Radial Basis Function) digunakan karena cocok untuk data yang tidak terpisah secara linear. Setelah model dibuat, fungsi fit() digunakan untuk melatih model menggunakan data latih (X_train dan y_train).

Output :
Output menunjukkan bahwa model SVC (Support Vector Classifier) berhasil dibuat dan dilatih menggunakan data latih. Tidak ada pesan error berarti proses pelatihan model telah berjalan dengan baik dan siap digunakan untuk melakukan prediksi pada data uji.

13.
```
[14]
✓ 0s
y_pred = model.predict(X_test)

#akurasi
print(f"Akurasi: {accuracy_score(y_test, y_pred) * 100:.2f}%")
#laporan klasifikasi
print("\nLaporan Klasifikasi:\n", classification_report(y_test, y_pred))

Akurasi: 97.37%

Laporan Klasifikasi:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        71
           1       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Penjelasan :
Kode di atas digunakan untuk mengevaluasi performa model SVM terhadap data uji. Baris :
- model.predict(X_test) menghasilkan hasil prediksi dari model.
- accuracy_score() menghitung persentase ketepatan model
- classification_report() menampilkan metrik evaluasi seperti precision, recall, dan f1-score untuk setiap kelas.

Output :

Hasil output menunjukkan bahwa model memiliki akurasi sebesar 97.37%, menandakan bahwa model mampu memprediksi dengan sangat baik.
Nilai precision, recall, dan f1-score untuk kedua kelas (0 = Benign, 1 = Malignant) juga tinggi, yaitu sekitar 0.97–0.98, menunjukkan bahwa model SVM mampu membedakan kedua jenis tumor dengan tingkat kesalahan yang sangat kecil dan performa yang seimbang.

14.
```
[13]
✓ 0s
    # Simpan model ke folder model/
    os.makedirs('/content/gdrive/MyDrive/Praktikum02/Praktikum6/model', exist_ok=True)
    joblib.dump(model, '/content/gdrive/MyDrive/Praktikum02/Praktikum6/model/svm_breast_cancer.pkl')

    ['/content/gdrive/MyDrive/Praktikum02/Praktikum6/model/svm_breast_cancer.pkl']
```

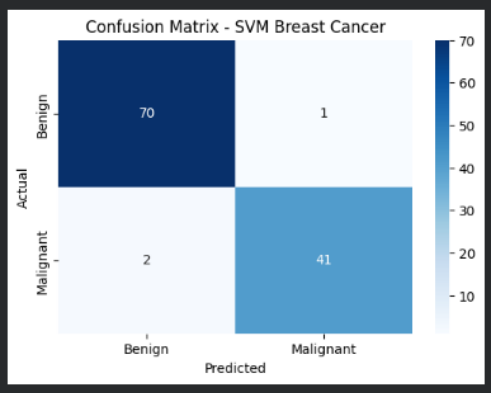
Penjelasan :
Kode ini digunakan untuk menyimpan model SVM yang telah dilatih ke dalam folder model di Google Drive. Fungsi os.makedirs() memastikan folder tujuan tersedia, sedangkan joblib.dump() menyimpan model dalam format .pkl agar dapat digunakan kembali tanpa perlu dilatih ulang.

Output :
Output menunjukkan bahwa file model berhasil disimpan pada direktori tersebut dan siap digunakan untuk proses prediksi atau evaluasi di tahap selanjutnya.

15.

```
[14]
✓ 0s
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAkurasi Model: {accuracy:.4f}")

print("\n Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Benign', 'Malignant']))

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d',
            xticklabels=['Benign','Malignant'],
            yticklabels=['Benign','Malignant'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - SVM Breast Cancer')
plt.show()
```

```
Akurasi Model: 0.9737

 Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.99      0.98        71
   Malignant       0.98      0.95      0.96        43

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```



Penjelasan :

Kode di atas digunakan untuk mengevaluasi performa model SVM.

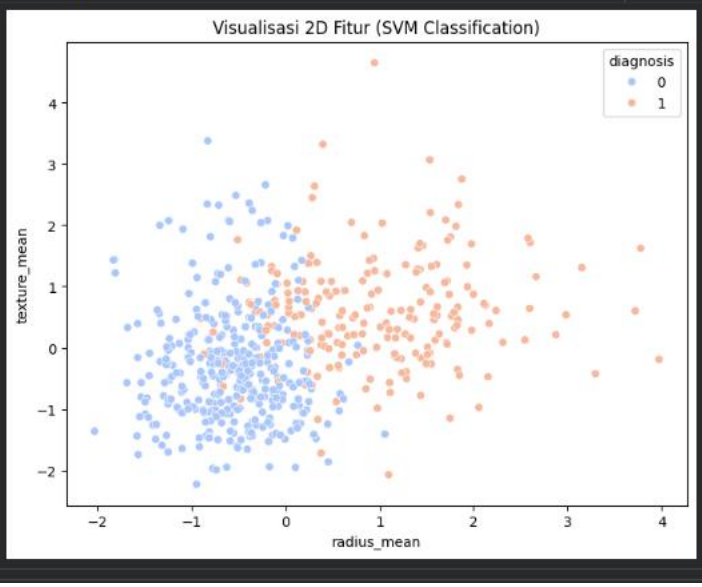
- model.predict(X_test) menghasilkan prediksi dari data uji.
- accuracy_score() menghitung tingkat akurasi model.
- classification_report() menampilkan metrik evaluasi seperti *precision*, *recall*, dan *f1-score*.
- confusion_matrix() digunakan untuk melihat jumlah prediksi benar dan salah antar kelas, yang divisualisasikan menggunakan *heatmap* dari Seaborn.

Output :

Hasil output menunjukkan bahwa model SVM memiliki akurasi sebesar 97,37%, yang berarti model mampu mengklasifikasikan data dengan sangat baik. Nilai precision, recall, dan f1-score pada kedua kelas juga tinggi (sekitar 0.97–0.98), menandakan performa model stabil dan seimbang dalam mengenali kasus Benign maupun Malignant.

Dari Confusion Matrix, hanya terdapat tiga kesalahan prediksi dari total 114 data uji, menunjukkan model bekerja dengan sangat baik.

16.
```
[15]
✓ 0s
# Visualisasi data
plt.figure(figsize=(8,6))
sns.scatterplot(x=X_scaled[:, 0], y=X_scaled[:, 1], hue=y, palette='coolwarm')
plt.title('Visualisasi 2D Fitur (SVM Classification)')
plt.xlabel('radius_mean')
plt.ylabel('texture_mean')
plt.show()
```



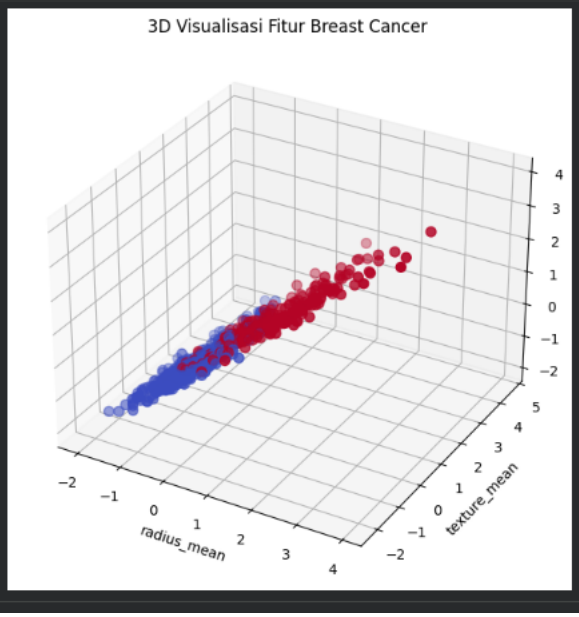
Penjelasan :

Kode ini digunakan untuk memvisualisasikan data dalam bentuk 2D scatter plot menggunakan dua fitur pertama, yaitu radius_mean dan texture_mean. Warna pada plot dibedakan berdasarkan label diagnosis (y), sehingga distribusi antara kelas Benign dan Malignant dapat terlihat lebih jelas.

Output :

Hasil visualisasi menunjukkan bahwa titik-titik data dengan label 0 (Benign) dan 1 (Malignant) cenderung membentuk kelompok (cluster) yang berbeda. Hal ini memperlihatkan bahwa kedua kelas dapat dipisahkan dengan cukup baik oleh model Support Vector Machine (SVM).

17.
```
[16]
✓ 1s
from mpl_toolkits.mplot3d import Axes3D

# Ambil tiga fitur pertama buat visualisasi 3D
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(
    X_scaled[:, 0],  # radius_mean
    X_scaled[:, 1],  # texture_mean
    X_scaled[:, 2],  # perimeter_mean
    c=y, cmap='coolwarm', s=50
)

ax.set_xlabel('radius_mean')
ax.set_ylabel('texture_mean')
ax.set_zlabel('perimeter_mean')
ax.set_title('3D Visualisasi Fitur Breast Cancer')
plt.show()
```



Penjelasan :
Kode ini digunakan untuk membuat visualisasi data dalam bentuk 3 dimensi (3D) menggunakan tiga fitur pertama pada dataset (radius_mean, texture_mean, dan perimeter_mean). Fungsi Axes3D dari mpl_toolkits.mplot3d digunakan untuk membuat tampilan grafik 3D, sementara warna titik-titik ditentukan berdasarkan nilai diagnosis (y).

Output :
Output berupa grafik 3D yang menampilkan sebaran data pasien kanker payudara berdasarkan tiga fitur utama. Titik berwarna berbeda menunjukkan dua kelas diagnosis, yaitu Benign dan Malignant. Terlihat bahwa kedua kelompok data membentuk pola terpisah, yang menandakan model Support Vector Machine (SVM) dapat membedakan kedua kelas dengan baik.

Kesimpulan :
Berdasarkan hasil percobaan yang telah dilakukan, algoritma Support Vector Machine (SVM) berhasil mengklasifikasikan data kanker payudara dengan sangat baik menggunakan dataset Breast Cancer Wisconsin (Diagnostic) dari Kaggle. Model yang dibangun dengan kernel RBF

menghasilkan akurasi sebesar 97.37%, menunjukkan bahwa SVM mampu membedakan dengan akurat antara tumor jinak (Benign) dan ganas (Malignant). Tahapan preprocessing seperti normalisasi dan encoding berperan penting dalam meningkatkan performa model. Secara keseluruhan, metode SVM dapat diandalkan untuk menyelesaikan permasalahan klasifikasi biner pada data medis.

Link Kaggle =
Breast Cancer Wisconsin (Diagnostic) Data Set
https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

Link GitHub =
Aisyaramli15/Sem3ML_Praktikum6

Link GitHub Praktikum Mandiri =
Sem3ML_Praktikum6/notebook/Praktikummandiri.ipynb at main · Aisyaramli15/Sem3ML_Praktikum6

Link GitHub Praktikum DiKelas =
Sem3ML_Praktikum6/notebook/Prakdikls.ipynb at main · Aisyaramli15/Sem3ML_Praktikum6