MAKALAH

Penerapan Algoritma Support Vector Machine (SVM) dalam Mengklasifikasikan Kelas Dokumen Menggunakan Jupyter Notebook

Mata kuliah : Pengantar Kecerdasan Artifical
Dosen Pengampu : Cakra Adipura Wicaksana, ST., MT.

Disusun oleh kelompok 6:

1. **M. Reza Refanda (3337220046)**
2. **Aulia Hoirunnisa (3337220065)**
3. **Widi Tri Nurhasanah (3337220073)**

Program Studi Informatika

Fakultas Teknik

Universitas Sultan Ageng Tirtayasa

Tahun Ajaran 2023

KATA PENGANTAR

Puji Syukur kami panjatkan kehadirat Tuhan Yang Maha Esa atas limpahan Rahmat dan karunia-Nya, sehingga laporan yang berjudul “Penerapan Algoritma Support Vector Machine dalam Mengklasifikasikan Kelas Dokumen Menggunakan Jupyter Notebook” dapat kami selesaikan dengan baik. Makalah ini disusun untuk memenuhi Ulangan Akhir Semester (UAS) mata kuliah Pengantar Kecerdasan Artificial

Pada kesempatan ini, kami mengucapkan terima kasih kepada Bapak Cakra Adipura Wicaksana, ST., MT. sebagai dosen untuk mata kuliah Pengantar Kecerdasan Aritificial yang telah memberikan project ini kepada kami. Ada banyak hal yang kami pelajari dan kembangkan melalui project dalam makalah ini. Dan kami juga berterima kasih kepada teman-teman yang membantu kami dalam berbagai hal. Harapan kami, informasi dan materi yang terdapat dalam makalah ini dapat bermanfaat bagi pembaca.

Setelah berhasil menyelesaikan makalah ini, kami berharap apa yang kami kerjakan bisa membuat kami lebih memahami tentang materi support vector machine. Kami jauh dari kata kesempurnaan sehingga makalah ini mungkin terdapat beberapa kekurangan. Maka dari itu, jika ada kritik dan saran terkait ide tulisan maupun penyusunannya, kami akan menerimanya dengan senang hati

Cilegon, 15 Desember 2023

Kelompok 6

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Support Vector Machine (SVM) adalah salah satu algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. SVM merupakan metode pembelajaran yang diawali dengan mengidentifikasi batas keputusan yang optimal antara kelas-kelas yang ada. SVM digunakan untuk mencari hyperplane terbaik dengan memaksimalkan jarak antar kelas. Hyperplane adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas.

Penerapan algoritma Support Vector Machine dalam mengklasifikasikan kelas gambar dokumen dengan menggunakan jupyter notebook memiliki potensi untuk memberikan kontribusi yang signifikan dalam pengembangan sistem analisis dan dokumen. Melalui penggunaan SVM, kita dapat mengembangkan model klasifikasi yang akurat dan dapat diandalkan untuk berbagai aplikasi, termasuk pengenalan pola, klasifikasi dokumen, dan identifikasi objek visual.

1.2 RUMUSAN MASALAH

1) Bagaimana cara memprediksi gambar dokumen sesuai dengan target?
2) Apa aja parameter yang digunakan?
3) Berapa lama proses training diukur?
4) Apa saja metode validasi yang digunakan?
5) Apa saja kelebihan dan kekurangan dari hasil percobaan?

1.3 TUJUAN PENULISAN

1) Untuk mengetahui cara prediksi gambar sesuai dengan target.
2) Untuk mengetahui parameter, motode validasi yang digunakan.
3) Untuk mengetahui berapa lama proses training diukur.
4) Untuk mengetahui kelebihan dan kekurangan dari hasil percobaan.
5) Untuk memenuhi UAS mata kuliah Pengantar Kecerdasan Artificial.

BAB II

PEMBAHASAN

A. Hasil dan Pembahasan

1. Deskripsi dataset

Dataset	**https://www.kaggle.com/datasets/ritvik1909/document-classification-dataset**
Metode	Support Vector Machine
Instruksi	Akurasi prediksi dengan target, parameter yang digunakan (fungsi aktivasi, fungsi loss), berapa lama proses training diukur, metode validasi yang digunakan apa? Apa saja kelebihan dan kekurangan dari hasil percobaan
Kelas	Terbagi menjadi 3 folder (email, resume, dan scientific_publication)

2. Import library

```
#Impor library yang diperlukan

import cv2
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

- **Cv2 :** library OpenCV untuk pengolahan citra
- **Numpy:** library untuk operasi matematika dan array.
- **Os:** library untuk berinteraksi dengan sistem operasi (digunakan untuk berbagai operasi file dan direktori).
- **Train_test_split:** berfungsi untuk membagi data menjadi set pelatihan dan pengujian.
- **Svm:** modul dari scikit-learn untuk support vector machine.
- **Accuracy_score, precision_score, recall_score, f1_score, confusion_matrix:** metrik evaluasi performa klasifikasi
- **Matplotlib.**pyplot: library untuk membuat visualisasi grafik.
- **Seaborn:** library untuk membuat visualisasi statistic yang cantik.

- **From sklearn.metrics import confusion_matrix:** Baris ini mengimpor fungsi **'confusion_matrix'** dari scikit-learn, yang merupakan alat yang berguna untuk mengevaluasi kinerja algoritma klasifkasi.
- **Import time:** Baris ini mengimpor modul **'time'** yang akan digunakan untuk mengukur waktu eksekusi suatu bagian kode.
- **from sklearn.svm import SVC:** baris ini mengimpor algoritma support vector classification (SVC) dari scikit-learn. SVC adalah algoritma pembelajaran mesin populer yang digunakan untuk tugas klasifikasi.

3. **Mendefinisikan path folder train dan test**

```
# Mendefinisikan path untuk folder train dan test
train_path = '/Users/wtrin/Documents/UAS_PKA/dataset/train'
test_path = '/Users/wtrin/Documents/UAS_PKA/dataset/test'
```

- **'train_path':** ini adalah path (jalur) ke folder yang berisi data pelatihan. Data pelatihan biasanya digunakan untuk melatih model dalam proses pembelajaran mesin.
- **'test_path':** ini adalah path (jalur) ke folder yang berisi data pengujian. Data pengujian digunakan untuk menguji sejauh mana model yang telah dilatih dapat melakukan prediksi dengan benar pada data yang belum pernah dilihat sebelumnya.

4. **Memuat data train**

```
# Memuat data test
test_data = []
test_labels = []

for folder in ['email', 'resume', 'scientific_publication']:
    folder_path = os.path.join(test_path, folder)
    for img in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img)
        image = cv2.imread(img_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (64, 64))
        test_data.append(image)
        test_labels.append(folder)
```

Dengan menggunakan loop, kode diatas membaca setiap gambar dalam setiap kategori, mengubahnya ke format yang sesuai (skala keabuan dan ukuran yang sama yaitu 64, 64), dan kemudian menyimpannya beserta labelnya ke dalam dua list terpisah **('train_data' dan 'train_labels').** Setelah selesai, list **'train_data'** akan berisi data

gambar yang siap digunakan untuk melatih model dan **'train_labels'** akan berisi label atau kategori yang sesuai dengan setiap gambar.

5. **Memuat data test**

```
# Memuat data test
test_data = []
test_labels = []

for folder in ['email', 'resume', 'scientific_publication']:
    folder_path = os.path.join(test_path, folder)
    for img in os.listdir(folder_path):
        img_path = os.path.join(folder_path, img)
        image = cv2.imread(img_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (64, 64))
        test_data.append(image)
        test_labels.append(folder)
```

Dengan menggunakan loop yang serupa dengan kode sebelumnya, kode diatas membaca setiap gambar dalam kategori pada folder pengujian, mengubahnya ke format yang sesuai, dan kemudian menyimpannya beserta labelnya ke dalam dua list terpisah **('test_data' dan 'test_labels')**. Setelah eksekusi, list **'test_data'** akan berisi data gambar yang siap digunakan untuk menguji model dan **'test_labels'** akan berisi label atau kategori yang sesuai dengan setiap gambar pengujian.

6. **Mengkonversikan data train dan test ke dalam bentuk array**

```
# Konversi data train dan test ke dalam bentuk array
X_train = np.array(train_data)
y_train = np.array(train_labels)
X_test = np.array(test_data)
y_test = np.array(test_labels)
```

- **'np.array(train_data)':** Mengonversi list **'train_data'** (yang berisi gambar pelatihan) menjsadi array NumPy, sehingga dpaat diolah lebih efisien oleh operasi matematika dan fungsi yang disediakan oleh NumPy.
- **'np.array(train_labels)':** Mengonversi list **'train_labels'** (yang berisi label atau kategori dari setiap gambar pelatihan) menjadi array NumPy.
- Proses yang serupa juga dilakukan untuk data pengujian **('X_test' dan 'y_test').**

Setelah konversi ini, **'X_train'** dan **'X_test'** akan berisi data gambar dalam bentuk array, sedangkan **'y_train'** dan **'y_test'** akan berisi label atau kategori yang sesuai dalam bentuk array NumPy.

7. **Meratakan data train dan test**

```
# Ratakan data train dan test
X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)
```

'X_train.shape[0]' berfungsi untuk mengambil jumlah sampel gambar dalam data pelatihan **('X_train'),** sedangkan **'-1'** menandakan bahwa dimensi kedua data gambar harus dihitung secara otomatis sehingga total elemen dalam array tetap konstan. Proses yang serupa juga dilakukan untuk data pengujian **('X_test').** Setelah operasi ini, baik **'X_train'** maupun **'X_test'** akan berbentuk array 2D dengan setiap baris mewakili satu sampel gambar yang telah diratakan menjadi vektor satu dimensi.

8. **Menormalisasikan nilai piksel**

```
# Normalisasikan nilai piksel
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

'X_train.astype('float32) mengonversi tipe data dari integer menjadi float32, ini diperlukan karena normalisasi umumnya dilakukan pada data berjenis float. Sedangkan, **'/ 255.0'** membagi setiap nilai piksel dalam **'X_train'** dengan 255.0, dan ini akan menghasilkan nilai piksel yang ternomalisasi dalam rentang 0 hingga 1. Proses yang serupa juga dilakukan untuk pengujian data **('X_test').**

9. **Melatih model dengan menggunakan rbf**

Pada minggu pertama kernel yang digunakan linear

```
# Latih model menggunakan rbf
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

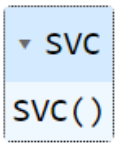
Dan untuk minggu kedua, kita tambahkan kernel yang lain yaitu rbf.

```
clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

- **'SVC(kernel='rbf')':** Membuat objek model Support Vector Machine (SVM) dengan menggunakan kernel radial basis function (RBF).

- **‘clf.fit(X_train, y_train)’:** Melatih model SVM dengan menggunakan data pelatihan **(‘X_train’ dan ‘y_train’).** Model ini untuk mengidentifikasi pola dalam data pelatihan dan membuat keputusan kelas yang sesuai. Kernel RBF digunakan untuk mentransformasikan data ke dimensi yang lebih tinggi agar dapat menangani data yang tidak terpisah secara linear di ruang fitur asli.

Hasil dari kode diatas:



10. Membuat prediksi pada data set

```
# Buat prediksi pada data test
y_pred_test = clf.predict(X_test)
```

Setelah menjalankan kode diatas, **‘y_pred_train’** akan berisi prediksi kelas dari model SVM pada set data pelatihan. Evaluasi performa model biasanya melibatkan perbandingan antara prediksi model dengan label sebenarnya pada data pelatihan, dan ini dapat dilakukan menggunakan berbagai metrik evaluasi tergantung pada jenis tugas klasifikasi yang sedang dijalankan (misalnya, akurasi, presisi, recall, dll).

11. Mengevaluasi model pada data test

```
# Evaluasi model pada data test
accuracy_test = accuracy_score(y_test, y_pred_test)
precision_test = precision_score(y_test, y_pred_test, average='weighted')
recall_test = recall_score(y_test, y_pred_test, average='weighted')
f1_test = f1_score(y_test, y_pred_test, average='weighted')
```

Kode di atas mencetak hasil evaluasi performa model pada data pengujian, menampilkan nilai akurasi **(`accuracy_test`)**, presisi **(`precision_test`)**, recall **(`recall_test`),** dan skor F1 **(`f1_test`).** Cetakannya memberikan ringkasan singkat tentang kemampuan model Support Vector Machine (SVM) dalam mengklasifikasikan data yang belum pernah dilihat sebelumnya. Melalui informasi ini, dapat diperoleh gambaran tentang seberapa baik model dapat mengatasi tugas klasifikasi pada data pengujian dan sejauh mana generalisasi model terhadap dataset yang berbeda.

12. Mencetak hasil evaluasi pada data test

```
# Cetak hasil evaluasi pada data test
print("Test Accuracy:", accuracy_test)
print("Test Precision:", precision_test)
print("Test Recall:", recall_test)
print("Test F1 Score:", f1_test)
```

Kode di atas mencetak hasil evaluasi performa model pada data pengujian, menampilkan nilai akurasi, presisi, recall, dan skor F1 pada console atau output program. Variabel yang digunakan dalam cetakan tersebut **(`accuracy_test`, `precision_test`, `recall_test`, dan `f1_test`)** diasumsikan telah didefinisikan dan dihitung sebelumnya menggunakan metrik evaluasi yang sesuai pada data pengujian. Melalui cetakan ini, kita dapat dengan cepat mengevaluasi seberapa baik model SVM dapat melakukan klasifikasi pada data pengujian dan membandingkannya dengan performa pada data pelatihan.

Berikut hasil dari kode diatas:

```
Test Accuracy: 0.7272727272727273
Test Precision: 0.7230769230769231
Test Recall: 0.7272727272727273
Test F1 Score: 0.7222222222222221
```

13. Membuat prediksi pada data train

```
# Buat prediksi pada data train
y_pred_train = clf.predict(X_train)
```

Dengan menggunakan metode `predict` dari objek model SVM **(`clf`),** model diterapkan pada data pelatihan **(`X_train`).** Hasil prediksi, yang berisi kelas yang diestimasi oleh model untuk setiap sampel dalam data pelatihan, disimpan dalam variabel **`y_pred_train`.** Proses ini memungkinkan kita untuk mengevaluasi sejauh mana model dapat melakukan klasifikasi pada data yang telah digunakan selama pelatihan.

14. Mengevaluasi model pada data train

```
# Evaluasi model pada data train
accuracy_train = accuracy_score(y_train, y_pred_train)
precision_train = precision_score(y_train, y_pred_train, average='weighted')
recall_train = recall_score(y_train, y_pred_train, average='weighted')
f1_train = f1_score(y_train, y_pred_train, average='weighted')
```

Kode tersebut melakukan evaluasi performa model pada data pelatihan dengan menghitung metrik klasifikasi seperti akurasi **(`accuracy_train`),** presisi **(`precision_train`),** recall **(`recall_train`),** dan skor F1 **(`f1_train`).** Variabel

`y_pred_train` menyimpan prediksi model pada data pelatihan, yang kemudian dibandingkan dengan label sebenarnya **(`y_train`)** untuk menghitung metrik evaluasi tersebut. Evaluasi ini memberikan gambaran tentang sejauh mana model Support Vector Machine (SVM) dapat mengklasifikasikan data pelatihan dan seberapa baik model dapat menangani ketidakseimbangan kelas.

15. **Mencetak hasil evaluasi pada data train**

```
# Cetak hasil evaluasi pada data train
print("Train Accuracy:", accuracy_train)
print("Train Precision:", precision_train)
print("Train Recall:", recall_train)
print("Train F1 Score:", f1_train)
```

Kode tersebut mencetak hasil evaluasi performa model pada data pelatihan. Cetakannya meliputi nilai akurasi **(`accuracy_train`),** presisi **(`precision_train`),** recall **(`recall_train`),** dan skor F1 **(`f1_train`).** Informasi ini memberikan ringkasan tentang kemampuan model Support Vector Machine (SVM) dalam mengklasifikasikan data pelatihan. Melalui cetakan ini, dapat diperoleh pemahaman tentang seberapa baik model dapat memahami pola dalam data yang telah dilihat selama pelatihan dan sejauh mana model dapat melakukan klasifikasi dengan benar pada kelas-kelas yang berbeda.

Berikut hasil dari kode diatas:

```
Train Accuracy: 0.9621212121212122
Train Precision: 0.9659863945578231
Train Recall: 0.9621212121212122
Train F1 Score: 0.9625622091797458
```

16. **Menyiapkan label untuk grafik**

```
# Menyiapkan label untuk grafik
labels = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
```

Kode di atas digunakan untuk menyiapkan label yang akan digunakan pada grafik atau plot. Variabel `labels` merupakan list yang berisi empat string, yaitu **'Accuracy', 'Precision', 'Recall', dan 'F1 Score'.** Label-label ini nantinya dapat digunakan untuk memberikan keterangan atau identifikasi pada sumbu atau elemen-elemen grafik, membantu memahami metrik evaluasi yang ditampilkan secara visual.

17. Menyiapkan data untuk grafik train dan test

```
# Menyiapkan data untuk grafik train dan test
train_scores = [accuracy_train, precision_train, recall_train, f1_train]
test_scores = [accuracy_test, precision_test, recall_test, f1_test]
```

Kode diatas menyiapkan data yang akan digunakan untuk membuat grafik perbandingan performa antara data pelatihan dan data pengujian. Variabel **`train_scores`** dan **`test_scores`** adalah list yang berisi nilai-nilai evaluasi seperti akurasi, presisi, recall, dan skor F1 pada data pelatihan dan pengujian. Data ini nantinya dapat digunakan sebagai titik-titik pada grafik untuk memvisualisasikan sejauh mana model berkinerja pada kedua set data tersebut.

18. Menentukan posisi atau lokasi label

```
x = np.arange(len(labels))  # label locations
```

Kode di atas membuat array **`x`** menggunakan NumPy **(`np.arange(len(labels))`)** untuk menentukan posisi atau lokasi label pada sumbu x ketika akan membuat grafik. Panjang array ini sesuai dengan jumlah label yang telah didefinisikan sebelumnya **(`len(labels)`),** yaitu 4. Array ini nantinya dapat digunakan sebagai referensi untuk meletakkan elemen-elemen pada sumbu x pada saat membuat grafik.

19. Mendefinisikan lebar bar pada grafik

```
# Lebar bar pada grafik
width = 0.35
```

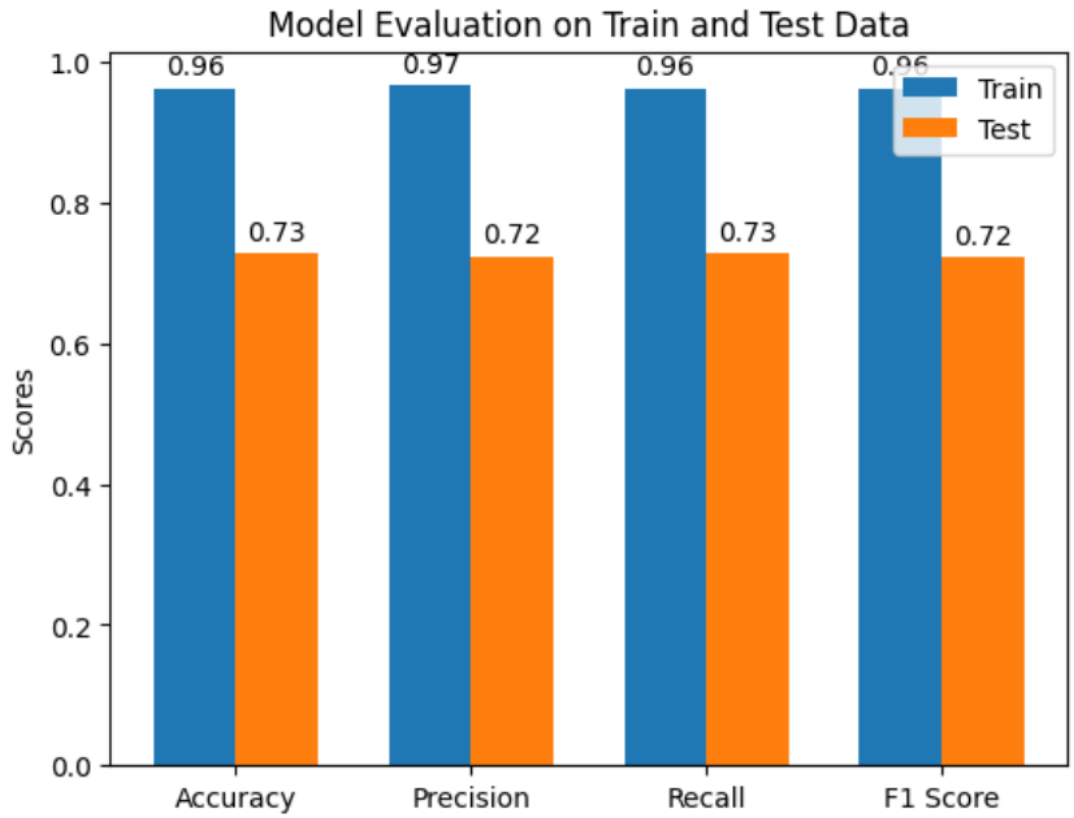
Kode di atas mendefinisikan lebar (width) dari setiap batang (bar) pada grafik. Nilai **`0.35`** yang diberikan pada variabel **`width`** menunjukkan lebar relatif dari setiap batang. Pada grafik batang, lebar ini akan mempengaruhi jarak antara satu grup batang dengan grup batang berikutnya pada sumbu x. Nilai yang sesuai dengan preferensi atau kebutuhan desain dapat dipilih untuk mencapai tata letak grafik yang diinginkan.

20. Menampilkan grafik, menambahkan label,judul, legenda, dan menampilkan nilai pada bar.

```
# Menampilkan grafik
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, train_scores, width, label='Train')
rects2 = ax.bar(x + width/2, test_scores, width, label='Test')

# Menambahkan label, judul, dan legenda
ax.set_ylabel('Scores')
ax.set_title('Model Evaluation on Train and Test Data')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Menampilkan nilai pada bar
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(round(height, 2)),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)

# Menampilkan grafik
plt.show()
```

- **`fig, ax = plt.subplots()`:** Membuat objek gambar (figure) dan sumbu (axes) untuk grafik.
- **`rects1 = ax.bar(x - width/2, train_scores, width, label='Train')` dan `rects2 = ax.bar(x + width/2, test_scores, width, label='Test')`:** Membuat batang pada posisi yang sudah ditentukan untuk data pelatihan dan pengujian. `width/2` digunakan untuk menyesuaikan posisi batang agar berada di sisi kanan atau kiri setiap lokasi label.
- **`ax.set_ylabel('Scores')`, `ax.set_title('Model Evaluation on Train and Test Data')`, `ax.set_xticks(x)`, `ax.set_xticklabels(labels)`, dan `ax.legend()`:** Menambahkan label, judul, dan legenda pada grafik untuk memberikan konteks informasi.
- **`autolabel(rects1)` dan `autolabel(rects2)`:** Menambahkan nilai akurat pada setiap batang untuk memberikan informasi yang lebih jelas.
- **`plt.show()`:** Menampilkan grafik secara keseluruhan.

Berikut hasil dari kode hasil:



21. Menyimpan confusion matrix untuk data train dan test

```
# Mendapatkan confusion matrix untuk data train dan test
conf_matrix_train = confusion_matrix(y_train, y_pred_train, labels=['email', 'resume', 'scientific_publication'])
conf_matrix_test = confusion_matrix(y_test, y_pred_test, labels=['email', 'resume', 'scientific_publication'])
```

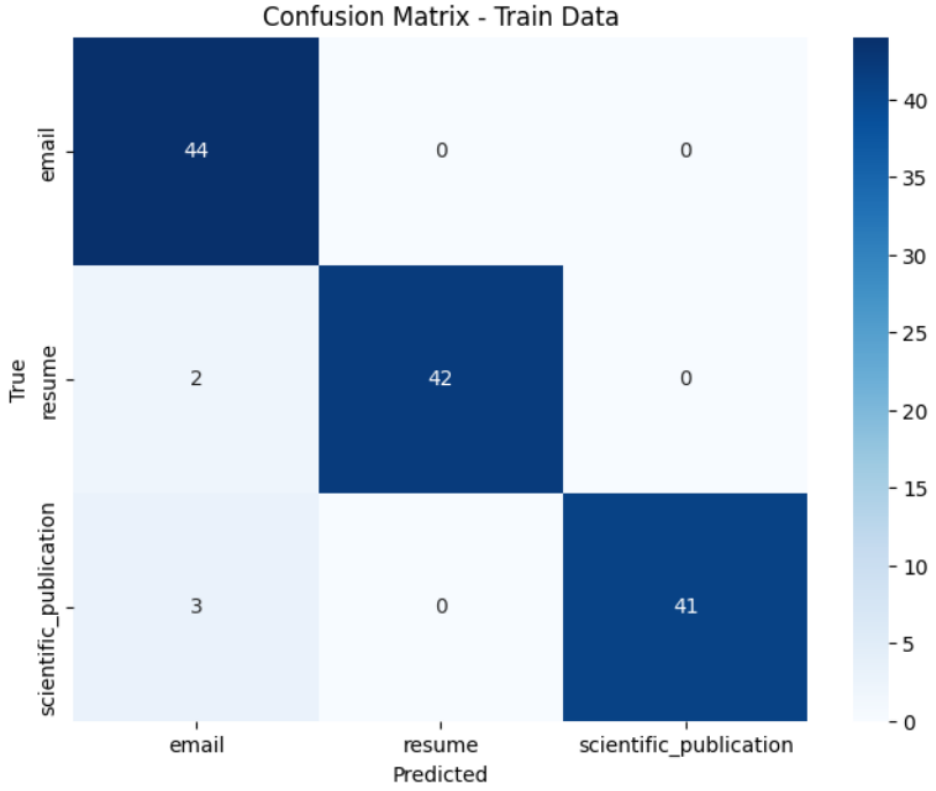
Kode di atas menggunakan fungsi **`confusion_matrix`** dari scikit-learn untuk menghitung matriks kebingungan (confusion matrix) pada data pelatihan dan pengujian. Matriks kebingungan menyajikan jumlah prediksi yang benar dan salah untuk setiap kelas pada model klasifikasi. Variabel **`conf_matrix_train`** dan **`conf_matrix_test`** menyimpan matriks kebingungan untuk data pelatihan dan pengujian, dengan kelas-kelas yang diidentifikasi sebagai 'email', 'resume', dan 'scientific_publication'.

22. Menampilkan confusion matrix untuk data train

```
# Menampilkan confusion matrix untuk data test
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['email', 'resume', 'scientific_publication'],
            yticklabels=['email', 'resume', 'scientific_publication'])
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Kode di atas menggunakan library seaborn dan matplotlib untuk membuat dan menampilkan heatmap (peta panas) dari matriks kebingungan pada data pelatihan. Dengan menggunakan **`sns.heatmap`,** matriks kebingungan **`conf_matrix_train`** disajikan secara visual dengan anotasi nilai, format angka desimal **(`fmt='d'`),** dan skema warna **'Blues'**. Label pada sumbu x dan y diatur sesuai dengan kelas-kelas yang diidentifikasi ('email', 'resume', dan 'scientific_publication'). Ukuran gambar diatur dengan **`plt.figure(figsize=(8, 6))`.** Judul, label sumbu x dan y, serta penampilan heatmap ditentukan melalui kode selanjutnya.

Berikut hasil dari kode diatas:



23. Menyimpan confusion matrix untuk data train sebagai array

```
# Menyimpan confusion matrix untuk data train sebagai array
conf_matrix_train_array = np.array(conf_matrix_train)
print("Confusion Matrix - Train Data (as array):")
print(conf_matrix_train_array)
```

Kode di atas mengonversi matriks kebingungan untuk data pelatihan **(`conf_matrix_train`)** menjadi array NumPy dan mencetaknya ke konsol. Variabel **`conf_matrix_train_array`** menyimpan representasi array dari matriks kebingungan tersebut. Langkah ini memungkinkan kita untuk menggunakan data matriks kebingungan dalam operasi atau analisis lanjutan yang memerlukan struktur array

NumPy. Cetakannya memberikan tampilan matriks kebingungan dalam bentuk array pada konsol.

Berikut hasil dari kode diatas:

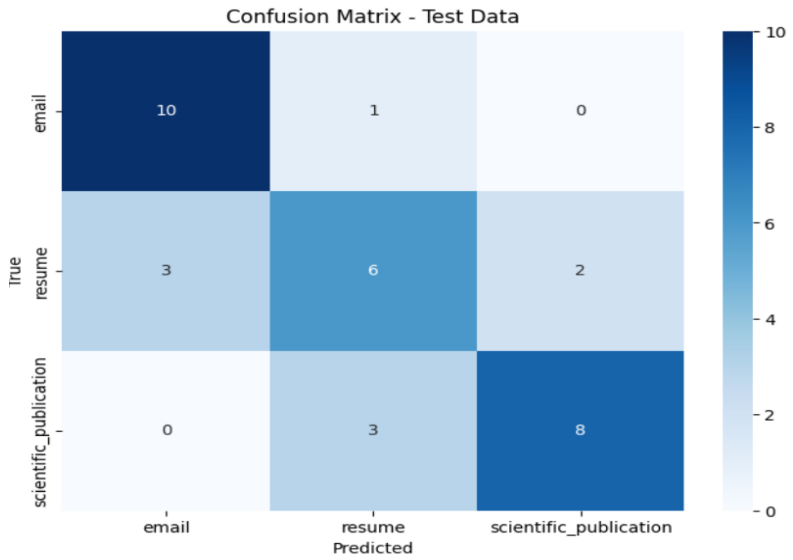
```
Confusion Matrix - Train Data (as array):
[[44  0  0]
 [ 2 42  0]
 [ 3  0 41]]
```

24. Menampilkan confusion matrix untuk data test

```
# Menampilkan confusion matrix untuk data test
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['email', 'resume', 'scientific_publication'],
            yticklabels=['email', 'resume', 'scientific_publication'])
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Kode di atas menggunakan library seaborn dan matplotlib untuk membuat dan menampilkan heatmap (peta panas) dari matriks kebingungan pada data pengujian. Dengan menggunakan **`sns.heatmap`,** matriks kebingungan **`conf_matrix_test`** disajikan secara visual dengan anotasi nilai, format angka desimal **(`fmt='d'`),** dan skema warna **'Blues'.** Label pada sumbu x dan y diatur sesuai dengan kelas-kelas yang diidentifikasi ('email', 'resume', dan 'scientific_publication'). Ukuran gambar diatur dengan **`plt.figure(figsize=(8, 6))`.** Judul, label sumbu x dan y, serta penampilan heatmap ditentukan melalui kode selanjutnya. Ini memberikan visualisasi yang informatif tentang seberapa baik model dapat mengklasifikasikan setiap kelas pada data pengujian.

Berikut hasil dari kode diatas:



25. Menyimpan confusion matrix untuk data test sebagai array

```
# Menyimpan confusion matrix untuk data test sebagai array
conf_matrix_test_array = np.array(conf_matrix_test)
print("Confusion Matrix - Test Data (as array):")
print(conf_matrix_test_array)
```

Kode di atas mengonversi matriks kebingungan untuk data pengujian **(`conf_matrix_test`)** menjadi array NumPy dan mencetaknya ke konsol. Variabel **`conf_matrix_test_array`** menyimpan representasi array dari matriks kebingungan tersebut. Langkah ini memungkinkan kita untuk menggunakan data matriks kebingungan dalam operasi atau analisis lanjutan yang memerlukan struktur array NumPy. Cetakannya memberikan tampilan matriks kebingungan dalam bentuk array pada konsol.

Berikut hasil dari kode diatas:

```
Confusion Matrix - Test Data (as array):
[[10  1  0]
 [ 3  6  2]
 [ 0  3  8]]
```

26. Mengukur waktu pelatihan

```
# Ukur waktu pelatihan
start_time_train = time.time()
clf.fit(X_train, y_train)
end_time_train = time.time()
training_time = end_time_train - start_time_train
```

Kode di atas digunakan untuk mengukur waktu pelatihan model. **`start_time_train`** menandai waktu awal sebelum pelatihan dimulai, **`end_time_train`** menandai waktu setelah pelatihan selesai, dan **`training_time`** menyimpan selisih waktu antara keduanya. Pengukuran waktu ini memberikan informasi tentang seberapa cepat atau lambat model dapat dilatih pada data pelatihan yang diberikan. Pemantauan waktu pelatihan menjadi penting terutama saat bekerja dengan dataset yang lebih besar atau kompleks untuk memahami efisiensi dan kinerja dari suatu model.

27. Membuat prediksi pada data set

```
# Buat prediksi pada data test
start_time_test = time.time()
y_pred_test = clf.predict(X_test)
end_time_test = time.time()
testing_time = end_time_test - start_time_test
```

Kode di atas digunakan untuk membuat prediksi pada data pengujian menggunakan model yang telah dilatih sebelumnya. `start_time_test` menandai waktu awal sebelum prediksi dimulai, `end_time_test` menandai waktu setelah prediksi selesai, dan `testing_time` menyimpan selisih waktu antara keduanya. Pengukuran waktu ini memberikan informasi tentang seberapa cepat atau lambat model dapat melakukan prediksi pada data pengujian. Pemantauan waktu prediksi menjadi penting untuk mengevaluasi keefisienan model, terutama saat menangani aplikasi real-time atau besar volume data uji.

28. Mencetak waktu pelatihan dan pengujian

```
# Cetak waktu pelatihan dan pengujian
print(f"Training Time: {training_time:.2f} seconds")
print(f"Testing Time: {testing_time:.2f} seconds")
```

Kode di atas mencetak waktu pelatihan dan pengujian model ke konsol. Format f-string **(`f"Training Time: {training_time:.2f} seconds"`)** digunakan untuk menyajikan waktu dengan dua desimal di bagian pecahan. Informasi waktu pelatihan dan pengujian memberikan gambaran tentang efisiensi dan kinerja model. Cetakannya memberikan waktu pelatihan dan pengujian dalam format detik dengan dua desimal angka di belakang koma pada konsol.

Berikut hasil dari kode diatas:

```
Training Time: 0.05 seconds
Testing Time: 0.02 seconds
```

29. Mengimpor dan menormalisasikan nilai piksel

```
from sklearn.decomposition import PCA  # Import PCA
```

```
# Normalisasikan nilai piksel
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

Kode di atas menggunakan library scikit-learn untuk mengimpor modul PCA (Principal Component Analysis) dan melakukan normalisasi nilai piksel pada data pelatihan **(`X_train`)** dan data pengujian **(`X_test`).** Normalisasi dilakukan dengan mengonversi nilai piksel ke dalam rentang 0 hingga 1, di mana setiap nilai piksel dibagi oleh 255.0 (maksimum nilai piksel). Proses normalisasi ini berguna untuk membantu model dalam konvergensi lebih baik selama pelatihan dan memfasilitasi pemrosesan yang lebih efisien.

30. Menggunakan PCA untuk visualisasi

```
# Menggunakan PCA untuk visualisasi
pca = PCA(n_components=2)  # Jumlah komponen dapat disesuaikan
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

Kode di atas menggunakan modul PCA dari scikit-learn untuk melakukan reduksi dimensi pada data pelatihan dan data pengujian. Objek PCA **(`pca`)** dibuat dengan menyertakan parameter **`n_components=2`**, yang menentukan jumlah komponen utama yang akan dipertahankan setelah reduksi dimensi. Proses reduksi dimensi dilakukan dengan memanggil metode `fit_transform` pada data pelatihan **(`X_train`)** dan metode **`transform`** pada data pengujian **(`X_test`).** Hasilnya, **`X_train_pca`** dan **`X_test_pca`,** menyimpan representasi data yang telah direduksi dimensinya dengan jumlah komponen utama yang telah ditentukan. Reduksi dimensi dengan PCA sering digunakan untuk visualisasi data pada ruang dua dimensi.

31. Menvisualisasikan data train

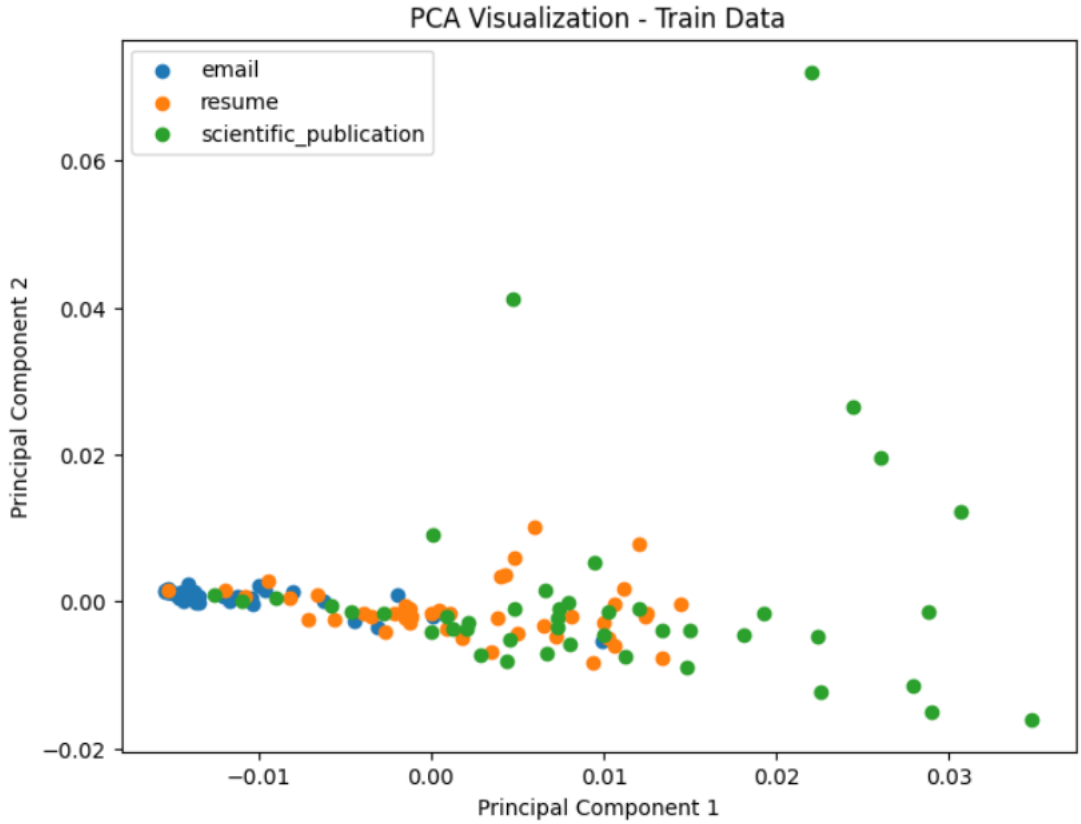
```
# Visualisasi data train
plt.figure(figsize=(8, 6))
for label in np.unique(y_train):
    indices = np.where(y_train == label)
    plt.scatter(X_train_pca[indices, 0], X_train_pca[indices, 1], label=label)
plt.title('PCA Visualization - Train Data')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

Kode di atas digunakan untuk membuat visualisasi data pelatihan setelah menjalani reduksi dimensi menggunakan PCA. Dalam grafik ini, setiap kelas pada data pelatihan **(`y_train`)** direpresentasikan dengan titik-titik yang tersebar pada bidang dua dimensi yang merupakan hasil dari PCA **(`X_train_pca`).**

- **`plt.figure(figsize=(8, 6))`:** Mengatur ukuran gambar grafik.
- **`for label in np.unique(y_train):`:** Melakukan iterasi untuk setiap kelas unik pada data pelatihan.
- **`indices = np.where(y_train == label)`:** Mengidentifikasi indeks di mana kelas pada data pelatihan sesuai dengan kelas saat iterasi.
- **`plt.scatter(X_train_pca[indices, 0], X_train_pca[indices, 1], label=label)`:** Membuat scatter plot untuk setiap kelas dengan menggunakan komponen utama hasil reduksi dimensi PCA.

- **`plt.title('PCA Visualization - Train Data')`:** Menambahkan judul pada grafik.
- **`plt.xlabel('Principal Component 1')` dan `plt.ylabel('Principal Component 2')`:** Menambahkan label sumbu x dan y pada grafik.
- **`plt.legend()`:** Menampilkan legenda berdasarkan kelas.
- **`plt.show()`:** Menampilkan grafik visualisasi data pelatihan menggunakan dua komponen utama hasil PCA.

  Berikut hasil dari kode diatas:

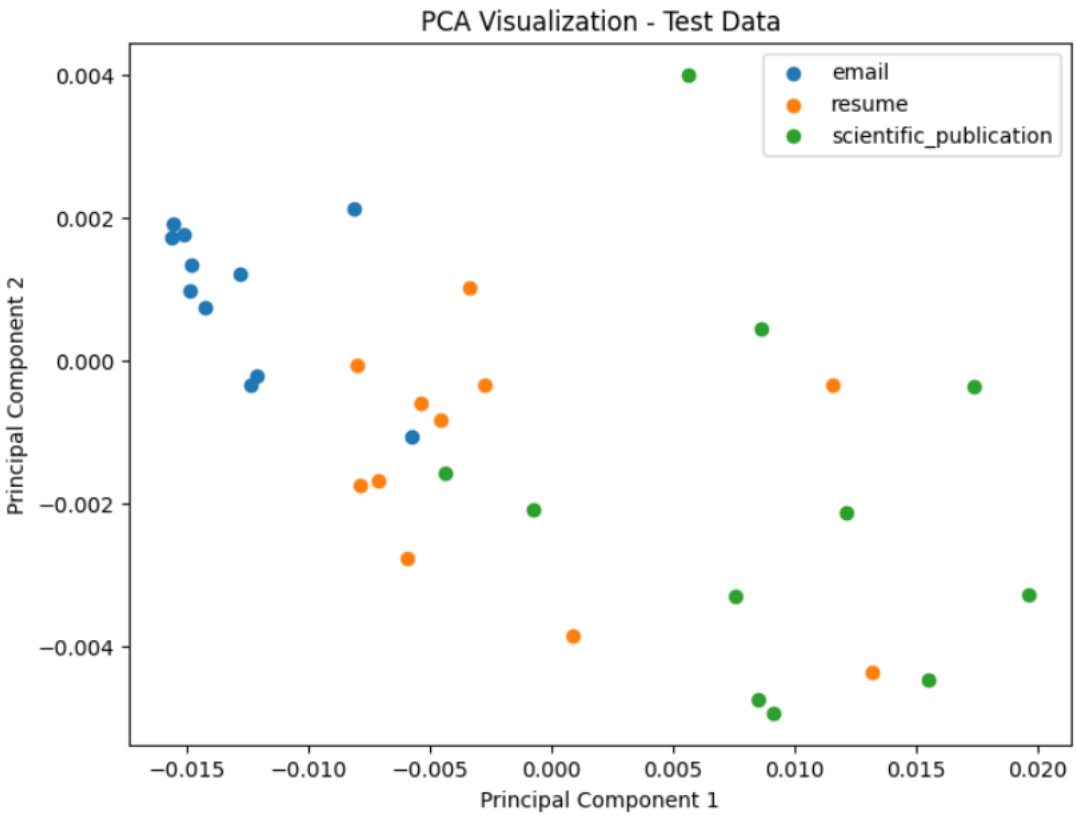


32. Menvisualisasikan data test

```
# Visualisasi data test
plt.figure(figsize=(8, 6))
for label in np.unique(y_test):
    indices = np.where(y_test == label)
    plt.scatter(X_test_pca[indices, 0], X_test_pca[indices, 1], label=label)
plt.title('PCA Visualization - Test Data')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

Kode di atas digunakan untuk membuat visualisasi data pengujian setelah mengalami reduksi dimensi menggunakan PCA. Dalam grafik ini, setiap kelas pada data pengujian **(`y_test`)** direpresentasikan oleh titik-titik yang tersebar pada bidang dua dimensi, yang merupakan hasil dari PCA **(`X_test_pca`).** Scatter plot dibuat untuk setiap kelas, dan legenda ditampilkan berdasarkan label kelas. Judul, label sumbu x dan y, serta ukuran gambar grafik diatur melalui kode tersebut. Cetakannya memberikan visualisasi yang intuitif tentang bagaimana data pengujian terdistribusi setelah mengalami reduksi dimensi menggunakan PCA.

Berikut hasil dari kode diatas:



BAB III

PENUTUP

3.1 Kesimpulan

Cara mengukur akurasi prediksi target dari dataset yaitu pertama kita harus membagi dataset menjadi data train dan test secara manual, setelah itu panggil path yang memuat folder train dan test, konversi data train dan data test ke dalam bentuk array, lalu ratakan data train dan test lalu normalisasikan nilai pikselnya, kemudian latih modelnya menggunakan rbf. Buat prediksi pada data train dan test, selanjutnya evaluasi modelnya, untuk melihat hasilnya kita cetak dengan mengetikkan 'printf'. Untuk prediksinya sendiri kita perlu memanggil variabel counfusion matrix train untuk melihat hasil dari akurasi dan prediksinya.

Parameter fungsi aktivasi dan fungsi loss tidak digunakan di algoritma support vector machine (SVM), parameter yang digunakan dalam program yang kami buat yaitu kernel SVM, Normalisasi data, Matrik Evaluasi, dan PCA. Waktu yang dibutuhkan untuk mentrain data yaitu 0.05 seconds karena dataset relatif kecil sehingga waktu pengujiannya lebih cepat. Metode validasi yang di gunakan adalah metode Hold Out. Kelebihan dari hasil percobaan diatas waktu training sangat cepat, namun itu bisa menjadi kekurangan karena waktu train yang sangat cepat itu bisa disebabkan oleh ukuran dataset yang kecil sehingga bisa menyebabkan overfitting yaitu dimana model Machine Learning mempelajari data dengan terlalu detail, sehingga yang ditangkap bukan hanya datanya saja namun noise yang ada juga direkam.

DAFTAR PUSTAKA

[1] P. A. Octaviani, Y. Wilandari dan D. Ispiyanti, "Penerapan Metode Klasifikasi Support Vector Machine (SVM) Pada Data Akreditasi Sekolah Dasar (SD) DI Kabupaten Magelang," *Jurnal Gaussian*, 2014.

[2] N. S. Dwi, "Penerapan Algoritma Support Vector Machine untuk Prediksi Harga Emas".