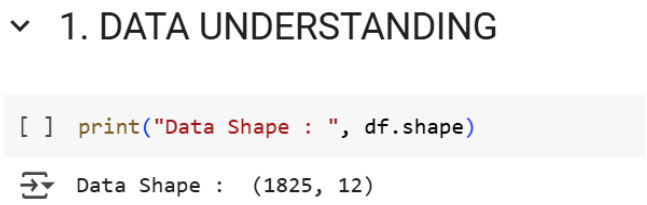
BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pengumpulan data ini dilakukan melalui dataset yang tersedia di portal satu data jakarta. Dataset ini berisikan data Indeks Standar Pencemaran Udara di DKI Jakarta, yang berisikan 1825 catatan pada dataset tersebut. Seperti yang dapat dilihat di Gambar 4.1



Gambar 4.1 Jumlah data yang ada pada dataset

	periode_data	tanggal	stasiun	PM10	PM2.5	so2	co	o3	no2	max	critical	category
0	202302	2/25/2023	DKI5 Kebon Jeruk Jakarta Barat	35	-	13	12	31	18	35	PM10	BAIK
1	202302	2/26/2023	DKI5 Kebon Jeruk Jakarta Barat	23	-	14	9	32	11	32	O3	BAIK
2	202302	2/27/2023	DKI5 Kebon Jeruk Jakarta Barat	20	-	13	8	33	13	33	O3	BAIK
3	202302	2/28/2023	DKI5 Kebon Jeruk Jakarta Barat	30	-	21	11	28	18	30	PM10	BAIK
4	202303	3/1/2023	DKI1 Bunderan HI	38	44	50	8	19	27	50	3	BAIK
...	...	...	...	...	...	...	...	...	...	...	...	...
1820	202311	11/26/2023	DKI5 Kebon Jeruk	48	71	33	21	44	20	71	PM25	SEDANG
1821	202311	11/27/2023	DKI5 Kebon Jeruk	51	76	32	18	53	19	76	PM25	SEDANG
1822	202311	11/28/2023	DKI5 Kebon Jeruk	56	88	33	20	48	21	88	PM25	SEDANG
1823	202311	11/29/2023	DKI5 Kebon Jeruk	56	88	32	18	56	18	88	PM25	SEDANG
1824	202311	11/30/2023	DKI5 Kebon Jeruk	30	57	32	15	61	17	61	O3	SEDANG

1825 rows × 12 columns

Gambar 4.2 Dataset Selection

Berdasarkan data yang diperoleh dari satudata.jakarta.go.id, sebanyak 1.825 data telah dikumpulkan dalam periode Februari 2023 hingga November 2023, sebagaimana ditampilkan pada Gambar 4.2. Dalam tahap pengolahan, data disaring dan jumlah atribut dikurangi agar hanya informasi yang benar-benar relevan yang digunakan. Beberapa variabel utama yang dipilih sebagai input meliputi tanggal,

lokasi, PM10, PM2.5, SO2, CO, O3, dan NO2, karena variabel-variabel ini memiliki pengaruh yang signifikan terhadap kualitas udara.

4.2 Pre-Processing

Pada tahap preprocessing data, dilakukan pembersihan untuk memastikan kualitas data yang optimal. Data yang tidak lengkap, kosong, mengandung noise, duplikat, atau inkonsisten akan diidentifikasi dan ditangani. Selain itu, data yang tidak relevan atau tidak diperlukan untuk analisis akan dibuang agar model dapat bekerja secara lebih akurat dan efisien.

```
[21] df.drop(columns=['periode_data', 'tanggal', 'stasiun', 'max', 'critical'], inplace=True, errors='ignore')
```

	PM10	PM2.5	so2	co	o3	no2	category
0	35	<NA>	13	12	31	18	BAIK
1	23	<NA>	14	9	32	11	BAIK
2	20	<NA>	13	8	33	13	BAIK
3	30	<NA>	21	11	28	18	BAIK
4	38	44	50	8	19	27	BAIK
...	...	...	...	...	...	...	...
1820	48	71	33	21	44	20	SEDANG
1821	51	76	32	18	53	19	SEDANG
1822	56	88	33	20	48	21	SEDANG
1823	56	88	32	18	56	18	SEDANG
1824	30	57	32	15	61	17	SEDANG

825 rows × 7 columns

Gambar 4.3 Preprocessing

Pada Gambar 4.3, terlihat bahwa masih terdapat data yang mengandung nilai NaN atau noise. Untuk memastikan kualitas data yang lebih baik, kita akan menghapus seluruh baris yang masih memiliki nilai NaN. Akibat dari proses ini, jumlah total baris dalam dataset mungkin akan berkurang. Namun, langkah ini diperlukan agar data yang digunakan dalam analisis lebih akurat dan tidak memengaruhi hasil model yang akan dibangun.

```
[33] numerical_columns = ['PM10', 'PM2.5', 'so2', 'co', 'o3', 'no2']
     df[numerical_columns] = df[numerical_columns].apply(pd.to_numeric, errors='coerce')
```

```
[41] df = df.dropna(subset=['PM10', 'PM2.5', 'so2', 'co', 'o3', 'no2'])
```

```
[42] print(f"Jumlah data setelah pembersihan: {df.shape[0]} baris")

     Jumlah data setelah pembersihan: 1284 baris
```

Gambar 4.4 Pembersihan data yang memiliki NaN

Langkah pertama yang dilakukan adalah mengubah tipe data dari bentuk objek (string) menjadi tipe numerik. Proses ini bertujuan agar nilai kosong dalam dataset dapat dikenali sebagai NaN (Not a Number). Dengan demikian, data yang tidak valid atau kosong dapat dengan mudah diidentifikasi dan dihapus menggunakan fungsi dropna(). Pengubahan tipe data ini juga penting untuk memastikan bahwa seluruh variabel numerik dapat diproses dengan benar dalam tahap analisis dan pemodelan selanjutnya, sehingga hasil yang diperoleh menjadi lebih akurat dan dapat diandalkan.

	PM10	PM2.5	so2	co	o3	no2	category
4	38.0	44.0	50.0	8.0	19.0	27.0	BAIK
5	29.0	33.0	47.0	11.0	21.0	27.0	BAIK
6	38.0	46.0	49.0	9.0	16.0	25.0	BAIK
8	33.0	41.0	47.0	11.0	21.0	22.0	BAIK
9	31.0	44.0	46.0	9.0	20.0	19.0	BAIK
...	...	...	...	...	...	...	...
1820	48.0	71.0	33.0	21.0	44.0	20.0	SEDANG
1821	51.0	76.0	32.0	18.0	53.0	19.0	SEDANG
1822	56.0	88.0	33.0	20.0	48.0	21.0	SEDANG
1823	56.0	88.0	32.0	18.0	56.0	18.0	SEDANG
1824	30.0	57.0	32.0	15.0	61.0	17.0	SEDANG

1284 rows × 7 columns

Gambar 4.5 Setelah data dibersihkan

Setelah proses pembersihan data dilakukan, jumlah data yang tersisa dapat dilihat pada Gambar 4.5, yaitu sebanyak 1.284 baris. Dengan data yang lebih bersih dan terstruktur, risiko terjadinya permasalahan saat tahap pelatihan (training) dan evaluasi model dapat diminimalkan. Langkah ini sangat penting untuk memastikan bahwa model yang dibangun nantinya bekerja dengan optimal dan menghasilkan prediksi yang lebih akurat. Dengan demikian, data yang telah diproses kini siap digunakan pada tahap selanjutnya, seperti normalisasi, pemisahan data latih dan uji, serta penerapan algoritma pemodelan.

4.3 Training Model

Pada tahap ini, proses pemisahan fitur dan label akan dilakukan untuk memastikan bahwa klasifikasi dapat berjalan secara optimal tanpa adanya kesalahan. Fitur merupakan variabel independen yang digunakan sebagai input dalam model, sementara label adalah variabel target yang ingin diprediksi. Dengan memisahkan keduanya, algoritma dapat mempelajari pola dari fitur tanpa tercampur dengan nilai target, sehingga meningkatkan akurasi dalam proses klasifikasi.

```
[50] X = df[['PM10', 'PM2.5', 'so2', 'co', 'o3', 'no2']]  # Fitur
     y = df['category']  # Label kategori
```

Gambar 4.6 Pemisahan fitur dan label

Selanjutnya, kategori pada dataset akan dikonversi ke dalam bentuk angka agar dapat diproses oleh model dengan lebih optimal. Proses ini dikenal sebagai encoding, yang bertujuan untuk mengubah data kategori menjadi format numerik sehingga dapat digunakan dalam tahap pelatihan dan pengujian model. Dengan melakukan konversi ini, algoritma machine learning dapat lebih mudah mengenali pola dalam data tanpa mengalami kesulitan dalam menangani nilai kategori. Selain itu, teknik encoding juga membantu dalam meningkatkan efisiensi komputasi serta memastikan bahwa model dapat beroperasi dengan lebih akurat dan stabil.

```
[51] from sklearn.preprocessing import LabelEncoder

     le = LabelEncoder()
     y = le.fit_transform(y)  # Mengubah kategori menjadi angka
```

Gambar 4.7 LabelEncoder

Setelah itu, data akan dibagi menjadi dua bagian, yaitu 80% untuk data pelatihan (training data) dan 20% untuk data pengujian (test data). Pembagian ini bertujuan agar model dapat belajar dari sebagian besar data yang tersedia dan kemudian diuji dengan data yang belum pernah dilihat sebelumnya untuk mengevaluasi kinerjanya.

```
from sklearn.model_selection import train_test_split

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

Gambar 4.8 Pembagian data training & test

4.4 Building Model

Pada tahap pembangunan model (building model), dilakukan proses pembentukan model yang akan digunakan untuk melakukan klasifikasi, dalam hal ini menggunakan Support Vector Machine (SVM). SVM adalah salah satu algoritma pembelajaran mesin yang sangat efektif dalam menangani data dengan berbagai dimensi dan kompleksitas. Algoritma ini bekerja dengan mencari hyperplane terbaik yang dapat memisahkan kelas data secara optimal.

```
[53] from sklearn.preprocessing import MinMaxScaler

     scaler = MinMaxScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

Gambar 4.9 Menormalisasi Data

Selanjutnya, dilakukan normalisasi data agar proses pelatihan dan pengujian model dapat berjalan dengan optimal. Normalisasi bertujuan untuk menyamakan

skala semua fitur, sehingga model tidak berat sebelah terhadap fitur dengan nilai yang lebih besar. Selain itu, normalisasi juga membantu mempercepat konvergensi dalam algoritma berbasis gradien dan memastikan bahwa distribusi data tetap terjaga tanpa mengalami perubahan struktur yang signifikan.

Min-Max Scaling dihitung menggunakan rumus berikut:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

di mana:

- Xnorm adalah nilai setelah dinormalisasi.
- X adalah nilai asli dari fitur.
- Xmin dan Xmax adalah nilai minimum dan maksimum dalam dataset.

Dengan normalisasi ini, semua nilai akan berada dalam rentang yang sama, sehingga algoritma machine learning dapat bekerja lebih optimal.

Sebagai contoh, misalkan terdapat data suhu udara dalam derajat Celcius:

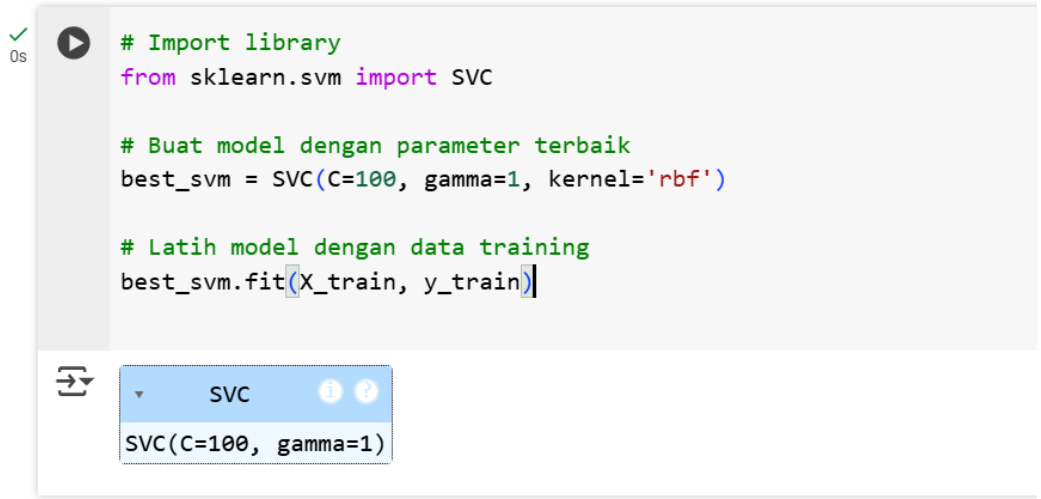
X=[30,35,40,45,50]X = [30, 35, 40, 45, 50]X=[30,35,40,45,50]

Jika diterapkan Min-Max Scaling dengan Xmin=30 dan Xmax=50, maka hasil normalisasi menjadi:

Nilai Asli	Perhitungan	Hasil Normalisasi
30	(30 - 30) / (50 - 30)	0.00
35	(35 - 30) / (50 - 30)	0.25
40	(40 - 30) / (50 - 30)	0.50
45	(45 - 30) / (50 - 30)	0.75
50	(50 - 30) / (50 - 30)	1.00

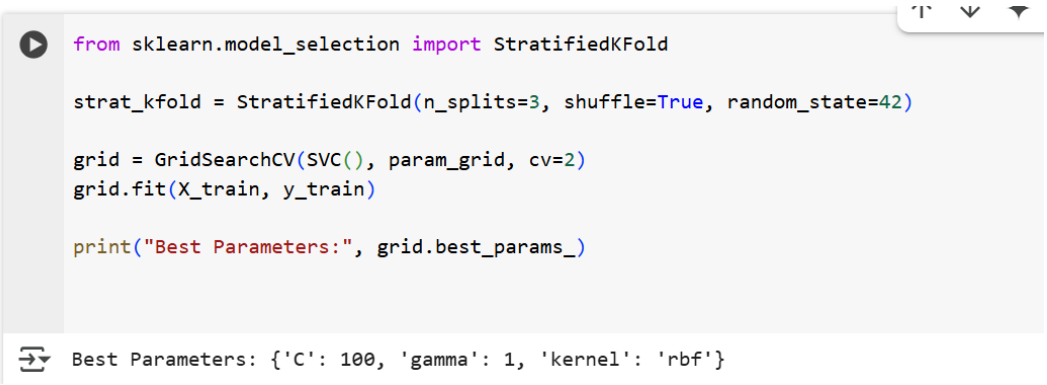
Tabel 4.1 Hasil Perhitungan Normalisasi

Berdasarkan tabel di atas, nilai suhu 30°C dinormalisasi menjadi 0.00, sementara 50°C menjadi 1.00, dan nilai lainnya disesuaikan dalam rentang tersebut.



Gambar 4.10 Melatih Model

Setelah data dinormalisasi, langkah berikutnya adalah melatih model menggunakan Support Vector Machine (SVM). Seperti yang telah dijelaskan sebelumnya, SVM merupakan salah satu algoritma yang andal dalam melakukan klasifikasi, terutama pada data dengan dimensi tinggi dan pola yang kompleks. SVM bekerja dengan mencari hyperplane terbaik yang memisahkan kelas-kelas dalam data secara optimal. Dengan menggunakan kernel yang tepat, model ini mampu menangani data yang tidak terpisahkan secara linear, sehingga meningkatkan akurasi dalam proses klasifikasi.



Gambar 4.11 Hyperparameter tuning

Hyperparameter tuning digunakan untuk mengoptimalkan performa model dengan mencari kombinasi parameter terbaik, sehingga model dapat menghasilkan prediksi yang lebih akurat dan optimal.

a. StratifiedKFold digunakan untuk membagi dataset menjadi 3 bagian (folds) dengan proporsi kelas yang sama di setiap fold.
b. shuffle=True memastikan data diacak sebelum dibagi menjadi fold.
c. random_state=42 memastikan hasil pembagian data tetap konsisten setiap kali kode dijalankan.
d. **GridSearchCV** mencoba berbagai kombinasi parameter dari **param_grid** untuk menemukan kombinasi terbaik.
e. **SVC()** adalah model Support Vector Classification yang akan diuji dengan berbagai parameter.
f. **cv=2** berarti **cross-validation** dilakukan dengan **2 fold**, artinya data dibagi menjadi **2 bagian**:
g. **50% data** untuk pelatihan
h. **50% data** untuk pengujian
i. grid.fit(X_train, y_train) menjalankan GridSearchCV untuk mencari parameter terbaik berdasarkan akurasi atau metrik lain.
j. grid.best_params_ menampilkan kombinasi parameter terbaik yang ditemukan.

4.5 Evaluation Model

Pada tahap evaluasi model, data yang telah dipisahkan sebelumnya sebagai data uji akan digunakan untuk mengukur kinerja model yang telah dilatih. Proses ini bertujuan untuk melihat seberapa baik model dalam mengklasifikasikan data baru yang belum pernah dipelajari sebelumnya. Metode evaluasi yang digunakan dapat berupa akurasi, precision, recall, dan F1-score, sehingga kita dapat memahami sejauh mana model mampu memberikan prediksi yang akurat dan andal.

```
# Prediksi data uji
y_pred = best_svm.predict(X_test)

# Evaluasi model
from sklearn.metrics import accuracy_score, classification_report

print("Akurasi:", accuracy_score(y_test, y_pred))
print("Laporan Klasifikasi:\n", classification_report(y_test, y_pred))
```

```
Akurasi: 0.9610894941634242
Laporan Klasifikasi:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86        24
           2       0.98      0.98      0.98       211
           3       0.95      0.91      0.93        22

    accuracy                           0.96       257
   macro avg       0.92      0.92      0.92       257
weighted avg       0.96      0.96      0.96       257
```

Gambar 4.12 Evaluasi Model

Setelah dilakukan pengujian, diperoleh hasil akurasi sebesar 96%, yang menunjukkan bahwa model memiliki tingkat ketepatan yang tinggi dalam melakukan klasifikasi. Akurasi yang tinggi ini menandakan bahwa model telah berhasil belajar dengan baik dari data latih dan mampu memberikan prediksi yang akurat pada data uji.

Selanjutnya, hasil evaluasi model akan dipresentasikan menggunakan perhitungan *confusion matrix*. Pada tahap ini, performa model yang telah dilatih menggunakan algoritma *Support Vector Machine* akan diuji dengan data uji yang telah dipisahkan sebelumnya. Analisis ini bertujuan untuk melihat seberapa baik model dapat mengklasifikasikan data dengan benar, termasuk mengidentifikasi kesalahan prediksi yang mungkin terjadi.

Gambar 4.13 Confusion Matrix

Hasil evaluasi model klasifikasi menggunakan Support Vector Machine (SVM) ditampilkan dalam bentuk Confusion Matrix. Confusion Matrix ini menggambarkan jumlah prediksi benar dan salah untuk setiap kelas dalam data uji.

Dalam tabel tersebut, terdapat tiga kelas yaitu Baik, Sedang, dan Tidak Sehat. Masing-masing sel dalam matriks menunjukkan jumlah data yang diklasifikasikan ke setiap kategori berdasarkan prediksi model dibandingkan dengan label aslinya.

4.6 Analisa Hasil Pengujian

Hasil klasifikasi pada data Indeks Standar Pencemaran Udara (ISPU) dari Februari hingga November 2023 menggunakan metode Support Vector Machine (SVM) menunjukkan tingkat akurasi sebesar 96.1%, yang menandakan bahwa model ini mampu melakukan klasifikasi dengan tingkat kesalahan yang sangat rendah.

Evaluasi menggunakan macro average menunjukkan bahwa kategori Baik (0) memiliki precision 84%, recall 88%, dan f1-score 86%, yang menunjukkan bahwa model dapat mengklasifikasikan kategori ini dengan cukup baik, meskipun masih terdapat beberapa kesalahan klasifikasi. Sementara itu, kategori Sedang (2) mencapai precision 98%, recall 98%, dan f1-score 98%, yang menunjukkan bahwa model hampir sempurna dalam mengenali dan mengklasifikasikan kategori ini. Sedangkan kategori Tidak Sehat (3) memperoleh precision 95%, recall 91%, dan f1-score 93%, yang menandakan bahwa model dapat mengenali pola dari kategori ini dengan cukup baik, meskipun masih terdapat sedikit kekeliruan dalam klasifikasi.

Secara keseluruhan, rata-rata macro average menunjukkan precision 92%, recall 92%, dan f1-score 92%, yang menunjukkan bahwa model memiliki performa yang cukup stabil di setiap kategori, meskipun terdapat ketidakseimbangan jumlah data antar kategori. Di sisi lain, weighted average menghasilkan precision 96%, recall

96%, dan f1-score 96%, yang menunjukkan bahwa model lebih cenderung dipengaruhi oleh kategori dengan jumlah data yang lebih besar, dalam hal ini kategori Sedang (2).

Meskipun model ini telah menunjukkan performa yang sangat baik, terdapat beberapa aspek yang dapat ditingkatkan lebih lanjut, terutama dalam menangani kategori yang memiliki jumlah data lebih sedikit, seperti kategori Baik (0) dan Tidak Sehat (3). Salah satu solusi yang dapat diterapkan adalah melakukan penyeimbangan data dengan teknik seperti oversampling untuk kategori yang lebih kecil atau undersampling untuk kategori yang lebih dominan, sehingga model dapat mengenali pola dari setiap kategori secara lebih seimbang.

Selain itu, pengoptimalan hyperparameter lebih lanjut dapat dilakukan untuk meningkatkan performa model, misalnya dengan menyesuaikan parameter kernel pada SVM agar lebih sesuai dengan karakteristik data yang digunakan. Penggunaan teknik feature selection juga dapat membantu mengurangi dimensi fitur yang kurang relevan, sehingga model dapat bekerja lebih efisien dan akurat dalam melakukan klasifikasi.

Tabel 4.2 Hasil Uji

Kategori	Precision	Recall	F1-Score	Support
Baik(0)	84%	88%	86%	24
Sedang(2)	98%	98%	98%	211
Tidak Sehat(3)	95%	91%	93%	22
Macro Average	92%	92%	92%	257
Weighted Average	96%	96%	96%	257
Akurasi			96%	257

Dengan hasil evaluasi yang diperoleh, model ini dapat digunakan sebagai alat prediksi yang cukup andal dalam memantau kualitas udara berdasarkan data ISPU. Namun, untuk meningkatkan generalisasi model terhadap data baru, disarankan untuk terus melakukan pembaruan data dan validasi model secara berkala guna

memastikan bahwa model tetap memiliki performa yang optimal dalam berbagai kondisi lingkungan yang mungkin berubah seiring waktu.