

MINI PROJECT

CLASSIFICATION MODEL WITH IRIS DATASET USING K-NEAREST NEIGHBORS (KNN) ALGORITHM

DIGITAL SKILL FAIR 35.0 - DATA SCIENCE

HINDUN TRIWAHYUNI



ABOUT THE PROJECT

Proyek ini bertujuan untuk membangun model klasifikasi untuk memprediksi spesies bunga Iris (Setosa, Versicolor, Virginica) menggunakan algoritma K-Nearest Neighbors (KNN). Dataset terdiri dari 150 observasi dengan empat fitur utama: panjang sepal, lebar sepal, panjang petal, dan lebar petal.

Proses dimulai dengan pembersihan data, normalisasi fitur, dan analisis eksploratif data (EDA) untuk memahami pola dalam dataset. Data kemudian dibagi menjadi set pelatihan dan set pengujian, diikuti dengan pembangunan model KNN dan optimasi nilai k. Evaluasi model dilakukan menggunakan metrik seperti akurasi, presisi, recall, dan F1-score.

KNN dipilih karena kesederhanaannya dan efektivitasnya pada dataset kecil. Algoritma ini bekerja dengan mengklasifikasikan data berdasarkan kedekatan jarak dengan tetangga terdekatnya, yang sesuai dengan pola dalam dataset Iris. Penyesuaian nilai k dilakukan untuk menghasilkan akurasi terbaik, menjadikan KNN pilihan yang fleksibel dan optimal untuk tugas ini.

TOOLS USED



Untuk menjalankan kode Python berbasis cloud.



Untuk menjalankan kode Python berbasis cloud.



Manipulasi data dan operasi numerik.



Visualisasi data dan hasil evaluasi

IMPORT LIBRARIES AND LOAD DATASET

```
# Importing necessary libraries
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
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.datasets import load_iris

# Load Iris dataset
iris = load_iris()

# Convert to pandas DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
df.head()
```

input

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

output

Langkah pertama adalah mengimpor pustaka yang diperlukan untuk proyek ini, seperti pandas, numpy, seaborn, matplotlib, dan sklearn. Pustaka-pustaka ini memberikan fungsionalitas yang dibutuhkan untuk manipulasi data, visualisasi, serta penerapan algoritma pembelajaran mesin.

Dataset Iris dimuat menggunakan fungsi load_iris dari scikit-learn. Data ini kemudian dianalisis untuk mengeksplorasi fitur-fitur yang ada dan label spesiesnya.

Dataset ini terdiri dari 4 fitur utama, yaitu panjang dan lebar sepal, serta panjang dan lebar petal, yang digunakan untuk mengklasifikasikan tiga spesies iris yang berbeda.



DATA PREPROCESSING (FEATURE NORMALIZATION)

```
# Normalizing features using StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.iloc[:, :-1]) # Excluding 'species' column
df_scaled = pd.DataFrame(scaled_features, columns=iris.feature_names)
df_scaled['species'] = df['species']
df_scaled.head()
```

input

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	-0.900681	1.019004	-1.340227	-1.315444	setosa
1	-1.143017	-0.131979	-1.340227	-1.315444	setosa
2	-1.385353	0.328414	-1.397064	-1.315444	setosa
3	-1.506521	0.098217	-1.283389	-1.315444	setosa
4	-1.021849	1.249201	-1.340227	-1.315444	setosa

Next steps:

[Generate code with df_scaled](#)

[View recommended plots](#)

[New interactive sheet](#)

output

Sebelum memulai pemodelan, fitur perlu dinormalisasi agar berada dalam skala yang sama. Hal ini penting untuk algoritma KNN karena algoritma ini bergantung pada perhitungan jarak antar data.

Kita menggunakan StandardScaler untuk menormalkan data sehingga nilai rata-rata setiap fitur menjadi 0 dan standar deviasi menjadi 1. Ini membuat algoritma KNN lebih efektif karena KNN menggunakan jarak antar data.

Dataset yang sudah dinormalisasi ditampilkan di sini untuk memastikan semuanya telah diproses dengan benar.

EXPLORATORY DATA ANALYSIS (EDA)

Sebelum melanjutkan dengan tugas klasifikasi, penting untuk memproses data terlebih dahulu dan melakukan analisis data eksploratori untuk mendapatkan wawasan tentang dataset.

Fungsi pairplot dari pustaka seaborn digunakan untuk membuat plot sebar berpasangan dari dataset, dengan warna yang berbeda mewakili spesies. Visualisasi ini memberikan gambaran visual tentang hubungan antar fitur yang berbeda dan distribusinya, memungkinkan kita untuk mengidentifikasi pola dan potensi pemisahan antar kelas.

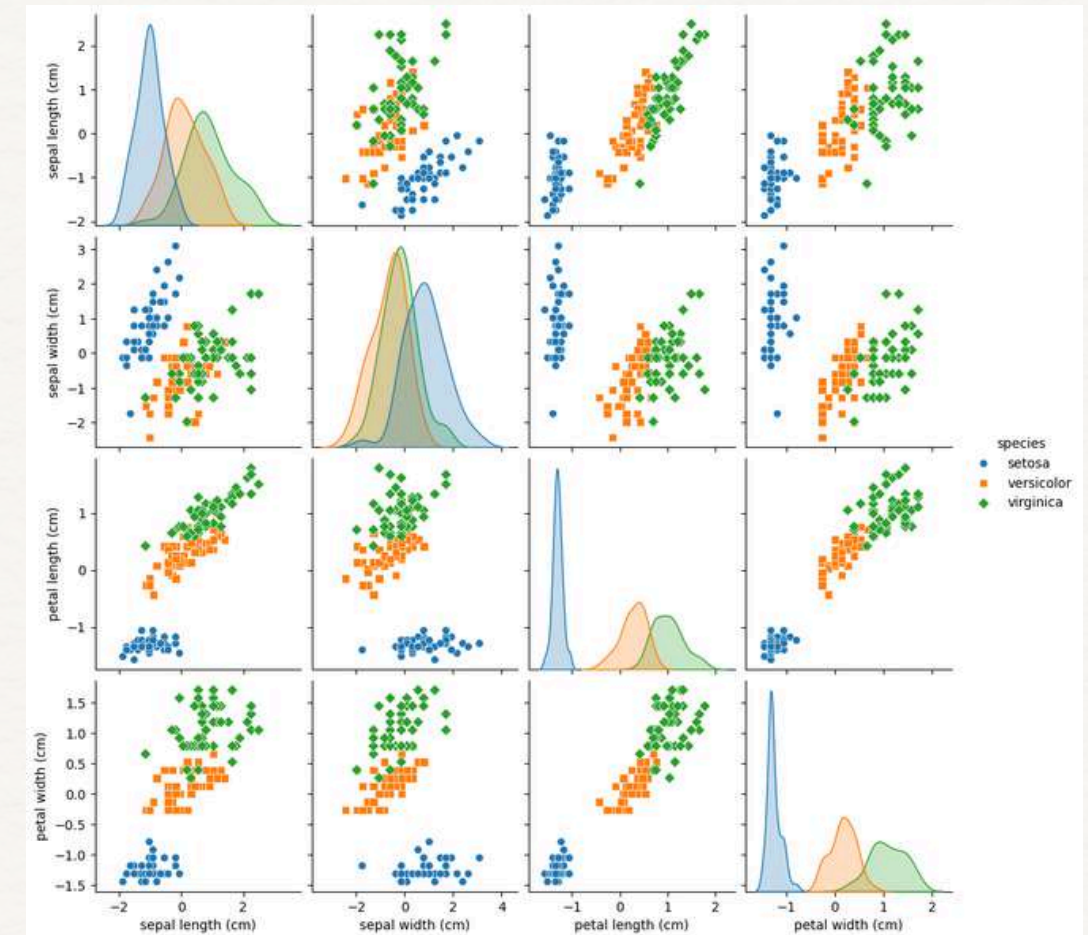


```
# Pairplot to visualize relationships between features
sns.pairplot(df_scaled, hue='species', markers=["o", "s", "D"])
plt.show()

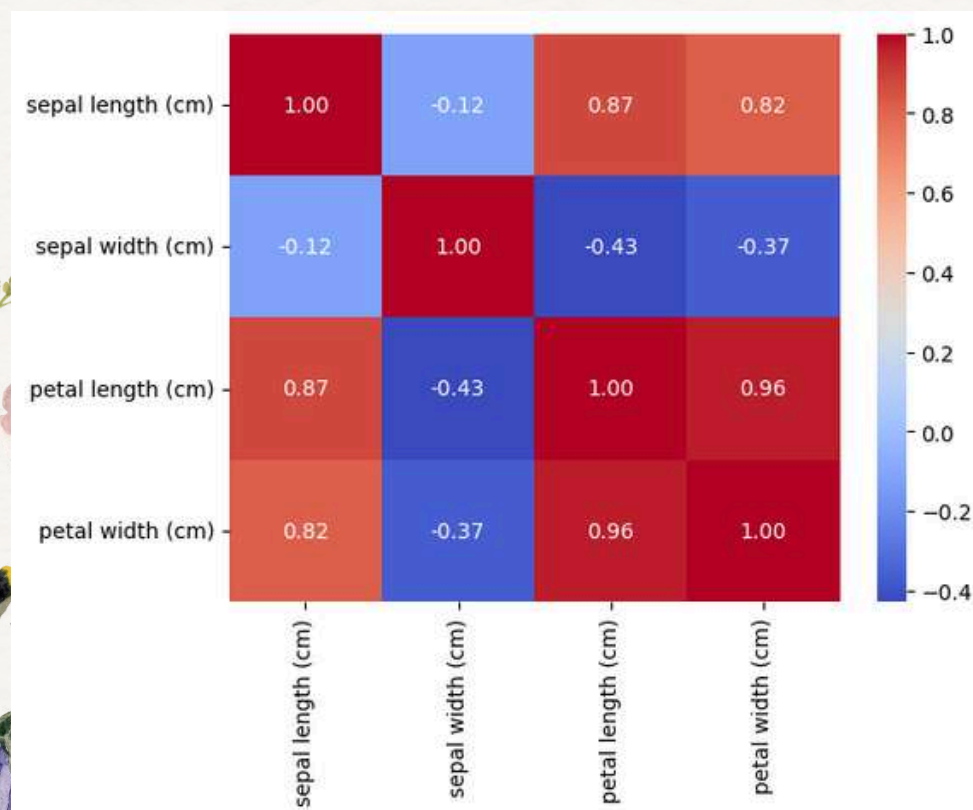
# Heatmap to show correlation between features
sns.heatmap(df_scaled.iloc[:, :-1].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.show()
```


EXPLORATORY DATA ANALYSIS (EDA)

Pairplot memberikan gambaran tentang bagaimana setiap fitur terkait satu sama lain. Di sini, kita mewarnai titik berdasarkan kelas bunga. Berdasarkan plot yang dibuat, dapat dilihat bahwa kelas iris-setosa selalu terpisah dari kelas yang lain. Artinya saat melakukan klasifikasi terdapat kemungkinan besar bahwa model akan selalu dapat membedakan spesies setosa dengan baik. Dapat dilihat juga distribusi data untuk petal-length, spesies setosa terpisah dari kelas yang lain.



pairplot



heatmap

Heatmap menampilkan matriks korelasi antar fitur. Nilai mendekati 1 atau -1 menunjukkan hubungan kuat, sementara nilai mendekati 0 menunjukkan hubungan lemah. Dari heatmap, terlihat bahwa terdapat korelasi positif yang kuat antara petal length dan petal width.

SPLIT DATASET INTO TRAINING AND TEST SETS

```
# Splitting the dataset into features (X) and target (y)
X = df_scaled.iloc[:, :-1] # Features
y = df_scaled['species']    # Target

# Splitting dataset into training (70%) and test (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Print the shape of the resulting datasets
print(f"Training set shape: {X_train.shape}, Test set shape: {X_test.shape}")
```

input

Dataset dibagi menjadi dua bagian: data pelatihan untuk membangun model dan data pengujian untuk mengevaluasi model. Di sini, kita membagi data menjadi 70% data training dan 30% data testing. Fungsi `train_test_split` juga memastikan hasil yang konsisten setiap kali kita membagi dataset dengan menggunakan parameter `random_state`.

```
➡ Training set shape: (105, 4), Test set shape: (45, 4)
```

output

Interpretasi:

Banyaknya data training adalah sebanyak 105 dan banyaknya data testing adalah sebanyak 45.

MODEL BUILDING: K-NEAREST NEIGHBORS (KNN)

```
# Creating a KNN model with k=5
knn = KNeighborsClassifier(n_neighbors=5)

# Training the model on the training data
knn.fit(X_train, y_train)

# Predicting flower species for the test data
y_pred = knn.predict(X_test)

# Print the first few predictions to verify the output
print(f"Predictions: {y_pred[:10]}") # Print first 10 predictions
```

```
➡ Predictions: ['versicolor' 'setosa' 'virginica' 'versicolor' 'versicolor' 'setosa'
               'versicolor' 'virginica' 'versicolor' 'versicolor']
```

Pada tahap ini, kita membuat model KNN, KNN membutuhkan suatu nilai konstanta k untuk menentukan berapa banyak tetangga yang akan digunakan oleh model. Kode di bawah ini akan melakukan training sebanyak 5 kali. Angka 5 ini bersifat bebas dan dapat diganti dengan angka lain.

input

output

Interpretasi:

Model KNN dengan k=5 memprediksi spesies bunga pada data uji menjadi tiga kategori ('versicolor', 'setosa', dan 'virginica') berdasarkan 5 tetangga terdekat. Prediksi ini akan dievaluasi dengan label sebenarnya untuk menilai akurasi model.

MODEL EVALUATION

Setelah model dibuat, kita perlu mengevaluasi model dengan menggunakan akurasi, classification report, dan confusion matrix untuk memahami seberapa baik model kita.

input

```
# Evaluating accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi: {round(accuracy * 100)}%")

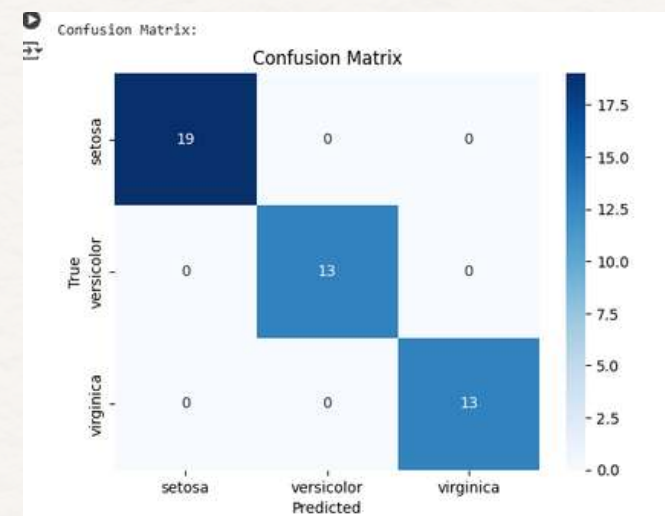
# Displaying classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Displaying confusion matrix
print("\nConfusion Matrix:")
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

output

Akurasi: 100%

Classification Report:				
	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	19
versicolor	1.00	1.00	1.00	13
virginica	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45



Interpretasi:

Akurasi Model

Model KNN mengklasifikasikan seluruh data uji dengan akurasi 100%, tanpa kesalahan prediksi. Ini menunjukkan dataset Iris yang sederhana sangat cocok untuk KNN.

Classification Report

Semua metrik evaluasi (precision, recall, F1-score) bernilai 100% untuk setiap kelas, menandakan performa sempurna pada data uji.

Confusion Matrix

Angka besar di diagonal utama menunjukkan prediksi yang benar untuk semua kelas, tanpa ada kesalahan klasifikasi.

Catatan Penting:

Hasil sempurna ini wajar untuk dataset kecil seperti Iris, tetapi performa model bisa berbeda pada dataset lebih kompleks atau berdimensi tinggi.

COMPARING ERROR RATE WITH THE K VALUE

Algoritma KNN memiliki kelemahan, seperti kesulitan pada dataset dengan dimensi fitur tinggi atau banyak fitur kategorikal, serta potensi overfitting jika akurasi mencapai 100%, terutama dengan data uji yang kecil. Untuk mengatasi hal ini, dilakukan analisis Comparing Error Rate with the K Value, yaitu memplot grafik antara nilai K dan tingkat kesalahan untuk dataset. Dengan menghitung kesalahan rata-rata untuk nilai K antara 1 hingga 40, kita dapat menemukan nilai K yang optimal dan mengurangi risiko overfitting atau underfitting.



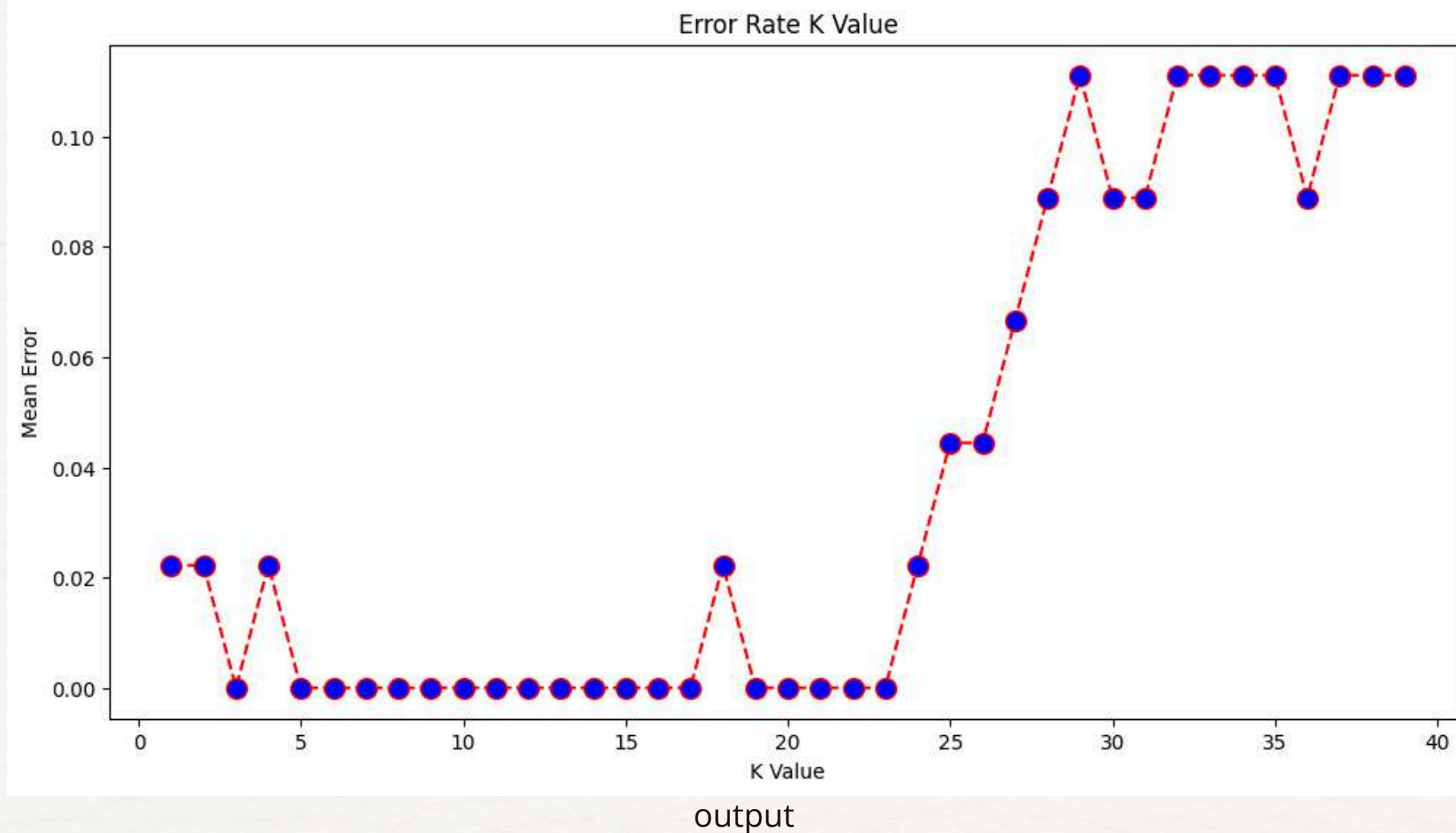
```
error = []
# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

input

COMPARING ERROR RATE WITH THE K VALUE

Text(0, 0.5, 'Mean Error')



Berdasarkan grafik yang dihasilkan, error rate terendah tercatat antara $K=5$ hingga $K=17$, yang menunjukkan bahwa model dengan nilai K dalam rentang tersebut memberikan performa terbaik. Setelah K mencapai 25, error rate mulai meningkat dan mencapai puncaknya sekitar 0.11 pada $K>30$. Untuk mengoptimalkan model, sebaiknya memilih nilai K dalam rentang 5 hingga 17, karena nilai K yang lebih besar dapat menyebabkan peningkatan error rate, yang berisiko mengarah pada overfitting atau underfitting.



Model K-Nearest Neighbors (KNN) berhasil mengklasifikasikan dataset Iris dengan akurasi tinggi, namun perlu diingat bahwa akurasi 100% bisa menjadi indikasi adanya overfitting, terutama pada dataset yang sederhana. Dengan melakukan Comparing Error Rate with the K Value, kita dapat melihat bahwa nilai K antara 5 hingga 17 menghasilkan kesalahan rata-rata yang rendah, menunjukkan performa terbaik. Hal ini menegaskan bahwa pemilihan nilai K yang tepat sangat penting untuk menghindari overfitting dan memastikan model dapat menggeneralisasi data dengan lebih baik.

CONCLUSION




About Me



HINDUN TRIWAHYUNI

 www.linkedin.com/in/hinduntriwahyuni

 hinduntriw@gmail.com

Saya adalah mahasiswa semester 6 jurusan Statistika di Universitas Terbuka yang sangat tertarik untuk mengeksplorasi dunia data. Dengan semangat belajar yang terus berkembang, saya bercita-cita untuk menjadi seorang Data Analyst atau Data Scientist. Saat ini, saya fokus pada pengolahan, analisis, dan visualisasi data untuk membantu pengambilan keputusan yang lebih tepat dan berbasis data.

Saya percaya bahwa data memiliki peran penting untuk memahami banyak hal di sekitar kita, dan saya berkomitmen untuk terus mengembangkan kemampuan di bidang teknologi dan analitik untuk bisa berkontribusi di dunia profesional.





THANK YOU