**Iris Flower Classification Using**

**K-Nearest Neighbors (KNN)**

# **1. Project Overview**

This project focuses on classifying Iris flowers into three species—Setosa, Versicolor, and Virginica—based on their sepal and petal measurements (length and width) using the K-Nearest Neighbors (KNN) algorithm. The goal is to build a machine learning model, evaluate its performance, and deploy it via a web interface using Flask.

# **2. Dataset Information**

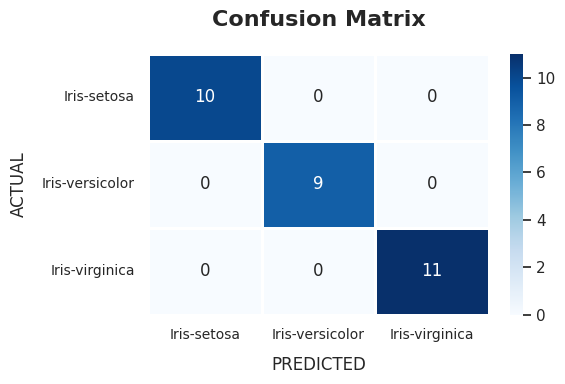
**• Source:** Kaggle - Iris Dataset (IRIS.csv)  
**• Features:**  
 - Sepal Length (cm)  
 - Sepal Width (cm)  
 - Petal Length (cm)  
 - Petal Width (cm)  
**• Target Variable:** Species (Setosa, Versicolor, Virginica)  
**• Total Samples:** 150

# **3. Workflow**

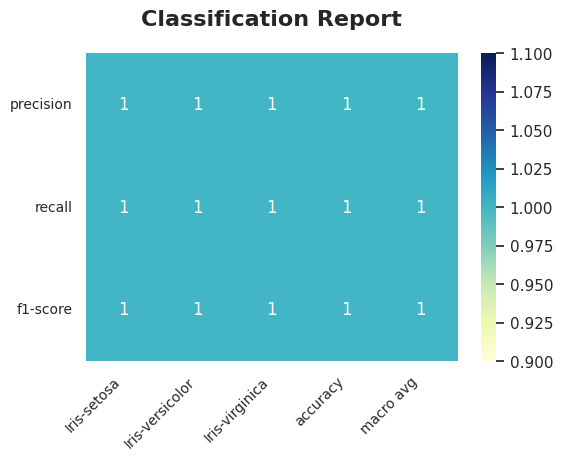
**Step 1: Load and Preprocess Data**  
- Loaded dataset using Pandas.  
- Checked for missing values and ensured data consistency.  
  
**Step 2: Feature-Label Split**  
- Features: Sepal and Petal measurements (length and width).  
- Labels: Species names.  
  
**Step 3: Train-Test Split**  
- Used train\_test\_split() to divide data into 80% training and 20% testing sets.  
  
**Step 4: Data Standardization**  
- Scaled features using StandardScaler to improve model performance.

**Step 5: Train KNN Model**  
- Trained KNN classifier with k=5.  
- Saved the trained model and scaler using joblib.  
  
**Step 6: Model Evaluation**  
- Used accuracy\_score, confusion\_matrix, and classification\_report to evaluate performance.

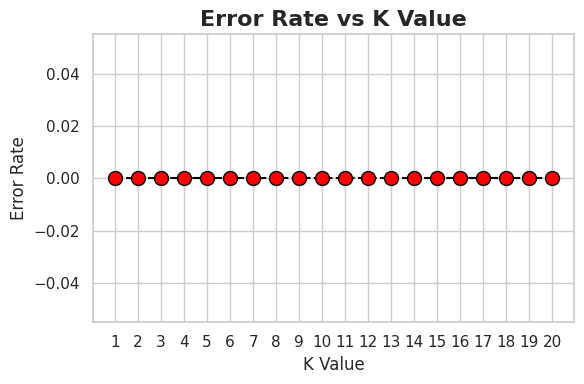
**- Plotted:**  
 1. Confusion Matrix



2. Classification Report Heatmap

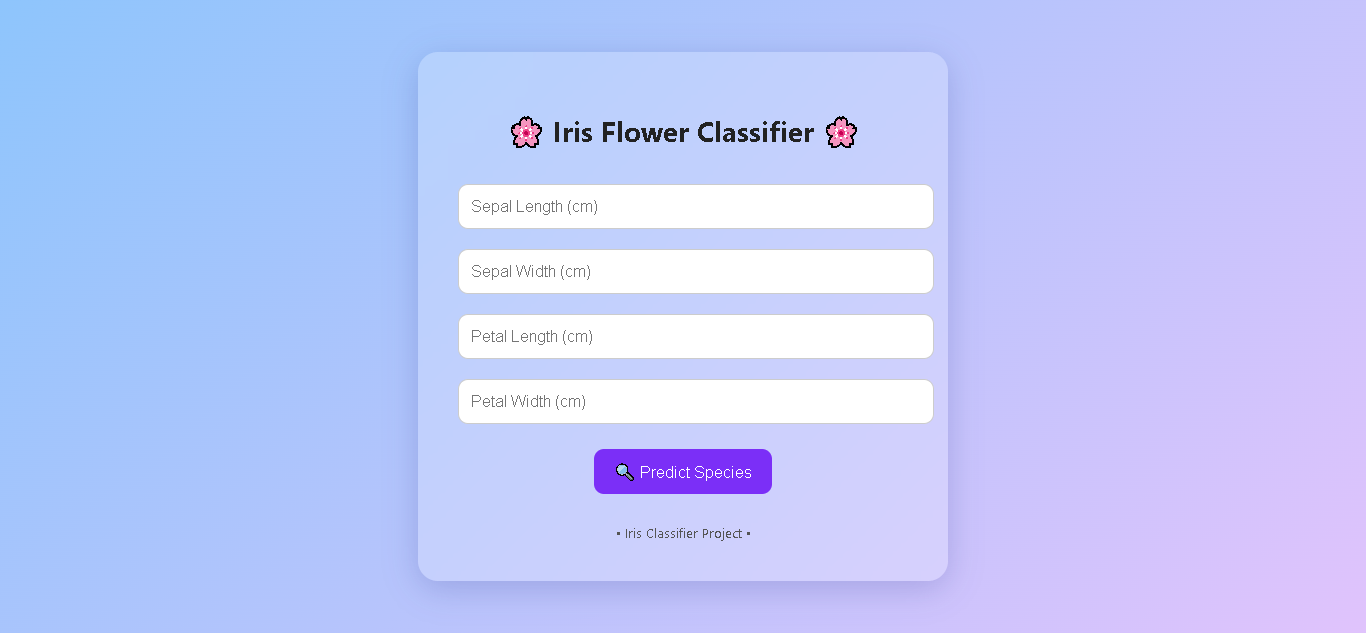


3. Error Rate vs K Plot



# **4. Flask Web Deployment**

**Backend: Flask**  
- Created a Flask app (app.py) to receive input from the user.  
- Loaded the trained model and scaler.  
- Processed form input, made predictions, and returned results to frontend.  
  
**Frontend: HTML**  
- Designed a user-friendly form to collect 4 inputs from the user.  
- Displayed predicted Iris species on submission.

- Frontend:

# **5. Results and Visualizations**

**1. Accuracy:**  
 Achieved a classification accuracy of 100%.  
  
**2. Confusion Matrix:**  
 Visually represented the model's predictions vs actual values.  
  
**3. Classification Report:**  
 Detailed precision, recall, and F1-score for each class.  
  
**4. Error Rate vs K Plot:**  
 Helped visualize how the error rate changes with different k values, identifying the optimal value.

# **6. How to Run the App**

1. Open Command Prompt and navigate to the project directory:  
cd "C:\Users\Administrator\Desktop\AI\AI Project" (Can be different in your device)  
  
2. Run the Flask app:  
python app.py  
  
3. Open browser and go to:  
<http://127.0.0.1:5000>

**OR**

1. Open the app.py file in Visual Studio Code.

2. Run the python file in terminal.

3. In the terminal you will find

Running on <http://127.0.0.1:5000>

4. Open browser and go to:  
<http://127.0.0.1:5000>

# **7. Conclusion**

The K-Nearest Neighbors algorithm proved to be highly effective for classifying Iris flowers with simple yet powerful logic. This project not only demonstrates a complete machine learning pipeline but also highlights the importance of deployment for real-world usability.

# **8. Files in the Project**

|  |  |
| --- | --- |
| File Name | Description |
| IRIS.csv | Iris dataset |
| app.py | Flask application |
| knn\_model.pkl | Trained KNN model |
| scaler.pkl | Standard scaler object |
| templates/index.html | HTML frontend for user input |

# **9. References**

- Scikit-learn Documentation  
- Kaggle Datasets  
- Flask Web Framework