# Iris Classification

## Overview

This project aims to classify Iris flower species using a RandomForest model. The dataset consists of features such as sepal length, sepal width, petal length, and petal width. The goal is to train a machine learning model to accurately predict the species of a given iris flower based on these features.

## Task Objectives

- Develop a machine learning model to classify Iris flower species.  
- Ensure clean, well-structured, and modular code.  
- Implement proper data preprocessing, training, evaluation, and prediction.

## Dataset

The dataset used is IRIS.csv, which contains:  
- Sepal length  
- Sepal width  
- Petal length  
- Petal width  
- Species (target variable)

## Installation

To set up the project, install the required dependencies:  
  
```  
pip install pandas numpy seaborn matplotlib scikit-learn joblib  
```

## Implementation Details

## Data Preprocessing

- Load the dataset using pandas.  
- Handle missing values (if any) and normalize numerical values.  
- Split the dataset into training and testing sets.

## Model Training

- The script uses RandomForestClassifier from scikit-learn.  
- The model is trained using 80% of the dataset, while the remaining 20% is used for testing.  
- Hyperparameters such as n\_estimators can be tuned for better performance.

## Model Evaluation

- The model's accuracy is measured using:  
 - Accuracy Score  
 - Classification Report  
 - Confusion Matrix  
 - Feature Importance Analysis

## Model Prediction

You can test the saved model with a sample input:  
  
```python  
loaded\_model, loaded\_scaler = load\_model()  
sample = [5.1, 3.5, 1.4, 0.2]  
print("Predicted species:", predict\_species(loaded\_model, loaded\_scaler, sample))  
```

## Expected Outputs

- Accuracy Score: Displays how well the model performs on test data.  
- Confusion Matrix: Visualizes classification performance.  
- Feature Importance Analysis: Identifies the most influential features.  
- Example Prediction:  
 ```  
 Predicted species: Setosa  
 ```

## Code Quality and Best Practices

- The code is modularized into functions for better readability and reusability.  
- Exception handling is included for dataset loading and model inference.  
- The project follows best practices for structuring machine learning scripts.