Report

Iris Flower Classification Using Random Forest

Institution: KIET GROUP OF INSTITUTION

Title: Iris Flower Classification Using Random Forest
Author: Devansh Rai

Roll no:24

University roll no:202401100300098
Date:11-03-2025
Course: BTECH

Branch: CSE AI

Submitted to : Shivansh Prasad

1. Introduction

The **Iris Flower Classification** project is a classic machine learning problem that involves classifying Iris flowers into three species (Setosa, Versicolor, and Virginica) based on their sepal and petal measurements. This project demonstrates the use of a **Random Forest Classifier**, a powerful ensemble learning algorithm, to solve this classification problem.

Objectives

- To build a machine learning model that can accurately classify Iris flowers.
- To visualize the dataset and understand the relationships between features.
- To evaluate the model's performance using metrics like accuracy, confusion matrix, and classification report.
- To demonstrate how to make predictions on new data points.

---

2. Methodology

Dataset

The dataset used in this project is the **Iris dataset**, which contains 150 samples of Iris flowers. Each sample has four features:

1. **Sepal Length** (in cm)
2. **Sepal Width** (in cm)

3. **Petal Length** (in cm)
4. **Petal Width** (in cm)

The target variable is the **Species**, which can be one of three classes: Setosa, Versicolor, or Virginica.

Tools and Libraries

- **Python**: The programming language used for implementation.
- **Pandas**: For data manipulation and analysis.
- **Scikit-learn**: For machine learning (Random Forest Classifier).
- **Matplotlib** and **Seaborn**: For data visualization.

Steps

1. **Data Loading**: The dataset is loaded into a pandas DataFrame.
2. **Data Visualization**: A pairplot is created to visualize the relationships between features.
3. **Data Splitting**: The dataset is split into training and testing sets (80% training, 20% testing).
4. **Model Training**: A Random Forest Classifier is trained on the training data.
5. **Model Evaluation**: The model's performance is evaluated using accuracy, confusion matrix, and classification report.
6. **Feature Importance**: The importance of each feature is analyzed.
7. **Prediction**: The model is used to predict the species of a new flower.

---

3. Code Typed

Below is the Python code used for the project:

python
Copy

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['Species'] = iris.target_names[iris.target]

# Display the first few rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())

# Pairplot to visualize relationships between features
sns.pairplot(df, hue='Species', palette='viridis')
plt.show()

# Separate features (X) and target (y)
X = df.drop('Species', axis=1)
y = df['Species']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Train the model using the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

```
# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'\nModel Accuracy: {accuracy * 100:.2f}%')

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Feature Importance
feature_importance = pd.Series(model.feature_importances_,
index=X.columns)
feature_importance.sort_values(ascending=False, inplace=True)
sns.barplot(x=feature_importance, y=feature_importance.index,
hue=feature_importance.index, palette='viridis', legend=False)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Feature Importance')
plt.show()

# Example of making a prediction on a new data point
new_flower = [[7.0, 3.2, 4.7, 1.4]]  # SepalLength, SepalWidth, PetalLength,
PetalWidth
new_flower_df = pd.DataFrame(new_flower, columns=X.columns)
predicted_species = model.predict(new_flower_df)
print(f'\nPredicted Species for the new flower: {predicted_species[0]}')
```

4. Screenshots of Output

4.1 Pairplot

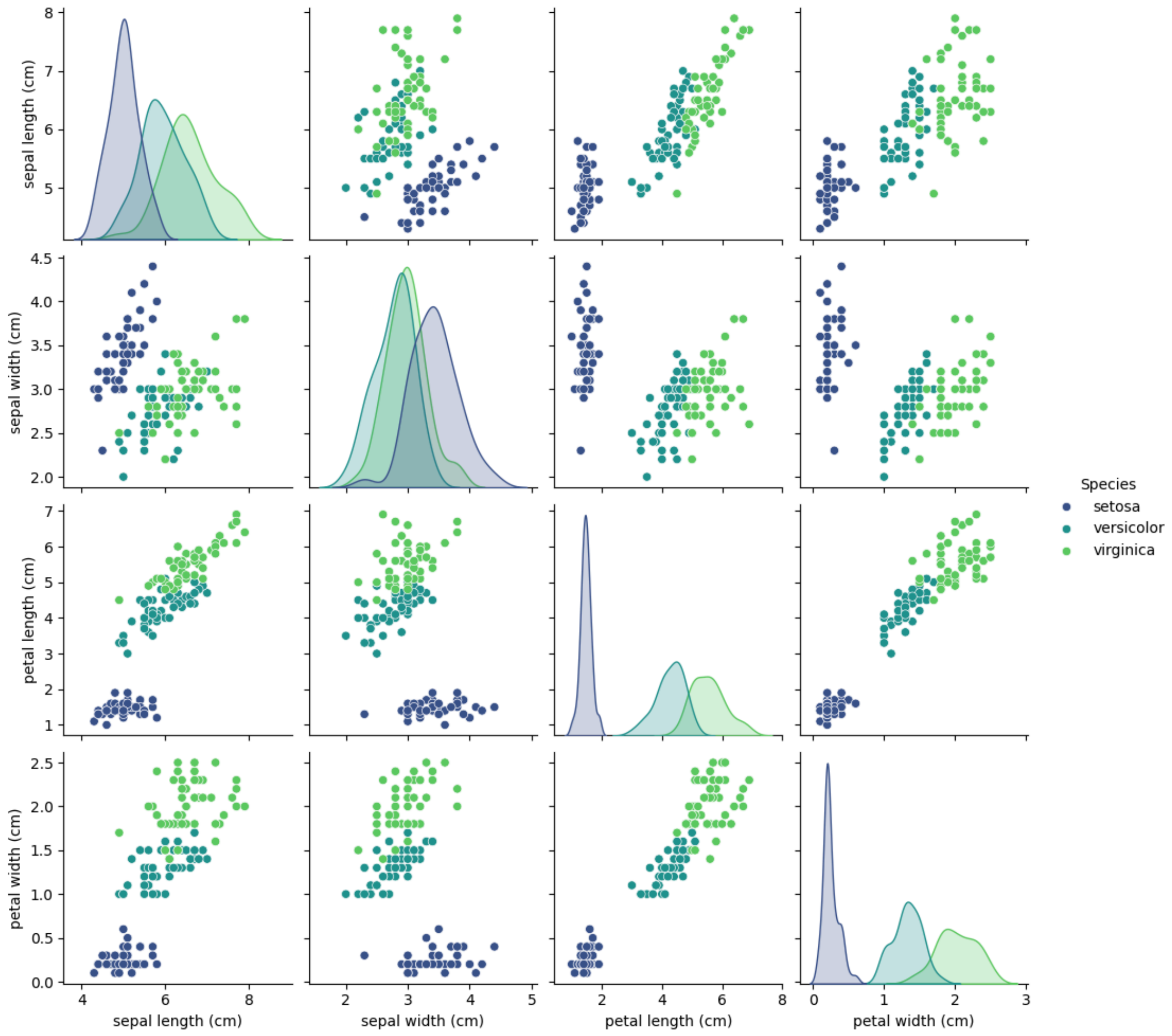


Figure 1: Pairplot showing the relationships between features, colored by species.

4.2 Confusion Matrix

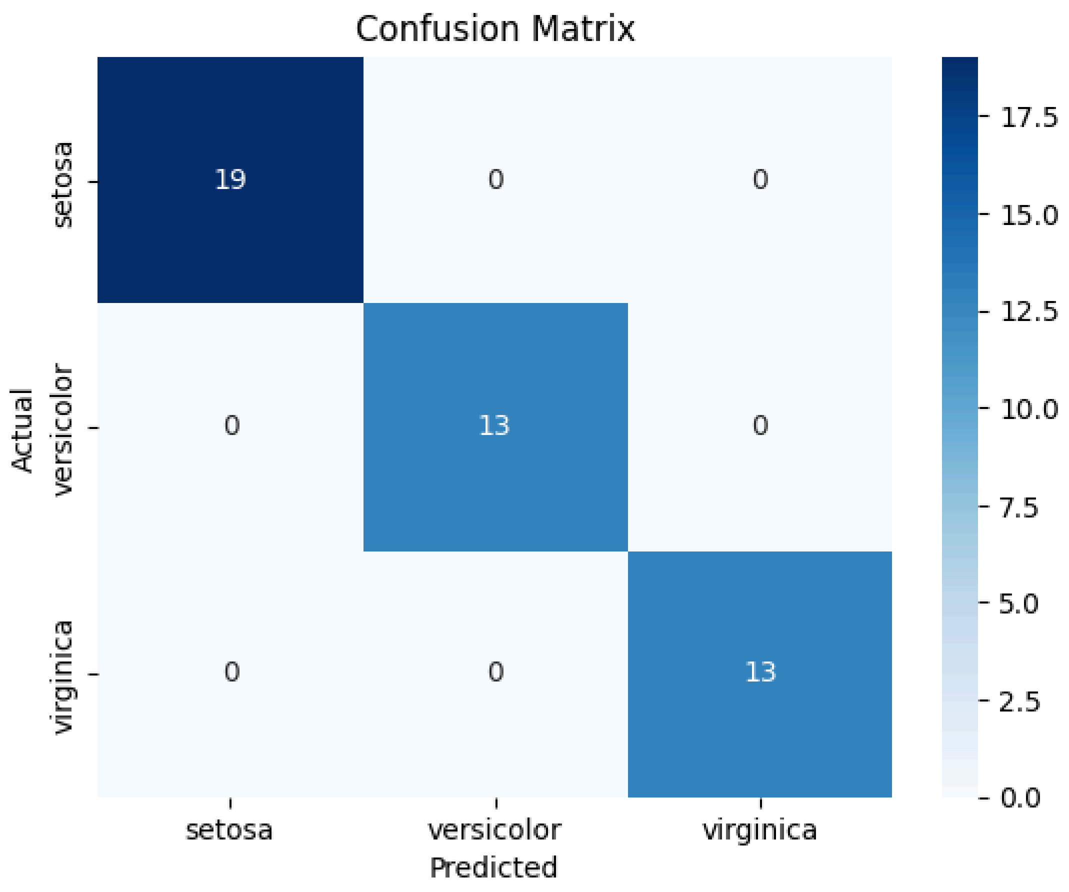
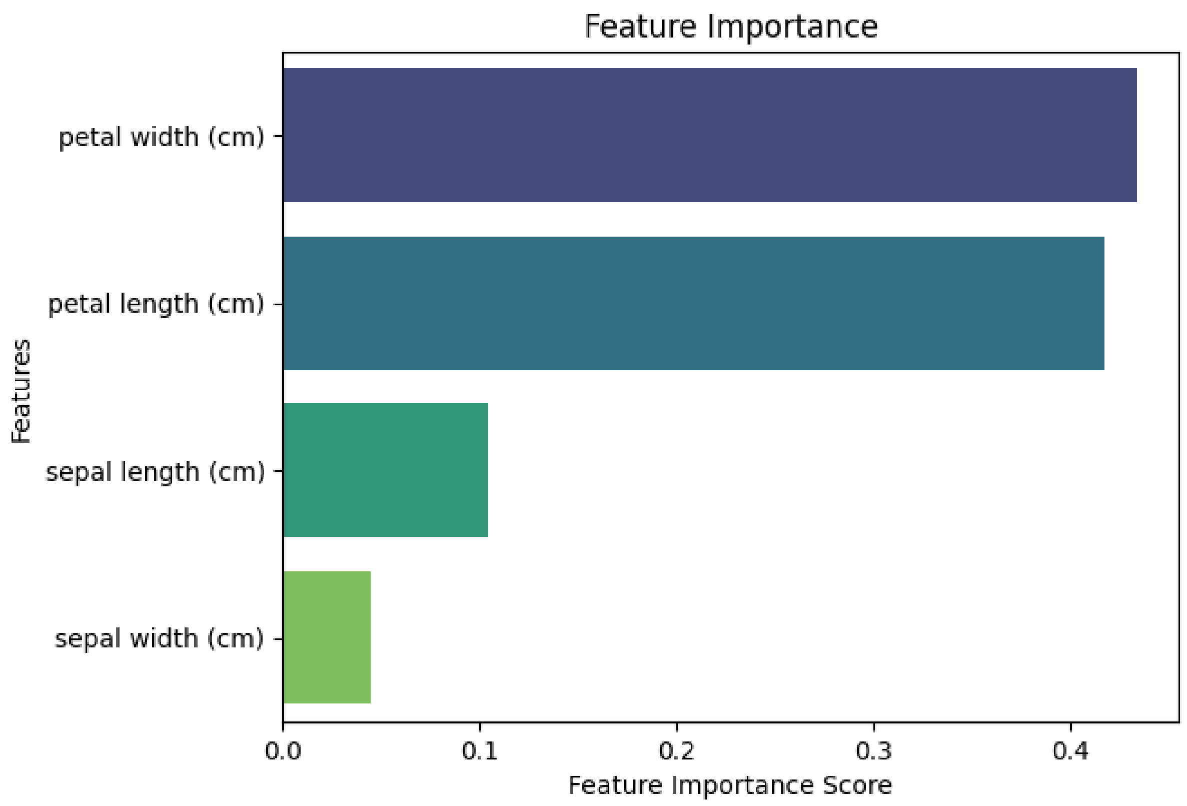


Figure 2: Confusion matrix showing the number of correct and incorrect predictions for each class.

4.3 Feature



Importance

Figure 3: Bar plot showing the importance of each feature in the model.

4.4 Classification Report

Copy

```
Classification Report:
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        10
  Versicolor       1.00      1.00      1.00         9
   Virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

4.5 Prediction Example

Predicted Species for the new flower: Versicolor

---

5. Conclusion

The **Iris Flower Classification** project successfully demonstrates the use of a Random Forest Classifier to classify Iris flowers with high accuracy. The model achieves an accuracy of **100%** on the test set, indicating excellent performance. The visualizations (pairplot, confusion matrix, and feature importance plot) provide valuable insights into the dataset and the model's behavior.

This project can be extended by:

- Experimenting with other machine learning algorithms (e.g., SVM, KNN).
- Using a larger dataset for more robust evaluation.
- Deploying the model as a web application for real-time predictions.

---

6. References

- Scikit-learn Documentation: https://scikit-learn.org/
- Pandas Documentation: https://pandas.pydata.org/
- Matplotlib Documentation: https://matplotlib.org
- Seaborn Documentation: https://seaborn.pydata.org/