Iris Flower Classification Using Machine Learning

A PROJECT REPORT
for
AI Project (AI101B)
Session (2024-25)

Submitted by

Sandeep kumar
(202410116100180)
Preet Kumar
(202410116100147)

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Komal Salgotra
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

TABLE OF CONTENTS

Introduction

The **Iris flower dataset** is a widely recognized dataset in the field of machine learning and statistics. Originally introduced by British statistician **Ronald Fisher** in 1936 in his paper *The Use of Multiple Measurements in Taxonomic Problems*, it remains a cornerstone in classification problems. The dataset has been extensively used in academic and research settings, serving as a foundational example in pattern recognition, supervised learning, and statistical analysis. Due to its simplicity and well-balanced structure, it is often one of the first datasets explored by students and researchers venturing into machine learning.

Containing **150 samples**, the dataset is equally distributed among three species of the **Iris flower**:

- **Iris Setosa**
- **Iris Versicolor**
- **Iris Virginica**

Each flower sample is described by **four numerical features** that capture essential morphological characteristics:

1. **Sepal Length (cm)** – Measures the length of the outer petal structure.
2. **Sepal Width (cm)** – Measures the width of the outer petal structure.
3. **Petal Length (cm)** – Measures the length of the inner petal structure.
4. **Petal Width (cm)** – Measures the width of the inner petal structure.

The dataset is notable for several reasons:

- **Balanced distribution**: Each species contains exactly **50 samples**, ensuring equal representation and preventing biases in classification models.
- **Ease of visualization**: With only **four features**, data relationships can be effectively explored through scatter plots, histograms, and other graphical methods, allowing for a better understanding of class separability.
- **Challenging classification scenario**: While **Iris Setosa** is easily distinguishable from the other two species, **Iris Versicolor** and **Iris Virginica** exhibit overlapping feature distributions, adding complexity to classification tasks and making the dataset an excellent test case for various algorithms.
- **Real-world applicability**: The techniques used to analyze this dataset are transferable to real-world classification problems in botany, medicine, and other scientific fields.

This study aims to classify Iris flowers into their respective species using the **K-Nearest Neighbors (KNN) algorithm**, a fundamental yet powerful machine learning technique that assigns class labels based on the majority vote of their nearest neighbors. KNN is a **non-parametric, instance-based learning method**, making it particularly useful for datasets where decision boundaries are not explicitly defined. By leveraging KNN, we can examine how well the algorithm can separate the three species based on their morphological features.

Methodology

The classification process follows a structured approach consisting of the following steps:
3.1 Data Collection
The dataset is obtained from the Scikit-learn library, which provides a preloaded version of the Iris dataset.
3.2 Exploratory Data Analysis (EDA)
Before training the model, an in-depth analysis of the dataset is conducted, including:

- Summary statistics to understand the distribution of numerical features.
- Pair plots and scatter plots to visualize relationships between features.
- Box plots to detect any outliers in the data.
- Correlation analysis to examine feature dependencies.

3.3 Data Preprocessing

- Handling missing values: The dataset is checked for any missing or inconsistent values.
- Feature scaling: Since KNN is distance-based, we normalize the feature values using StandardScaler to ensure that all features contribute equally to the classification process.
- Encoding categorical variables: Although not required for this dataset, categorical encoding is an important step in general machine learning workflows.

3.4 Model Selection and Training

- We use the K-Nearest Neighbors (KNN) algorithm, which classifies a new data point based on the majority class among its nearest neighbors.
- The dataset is split into training (80%) and testing (20%) sets.
- The optimal value of k (number of neighbors) is determined through experimentation and cross-validation.
- The Euclidean distance metric is used to compute distances between data points.

3.5 Model Evaluation
The trained model is assessed using:

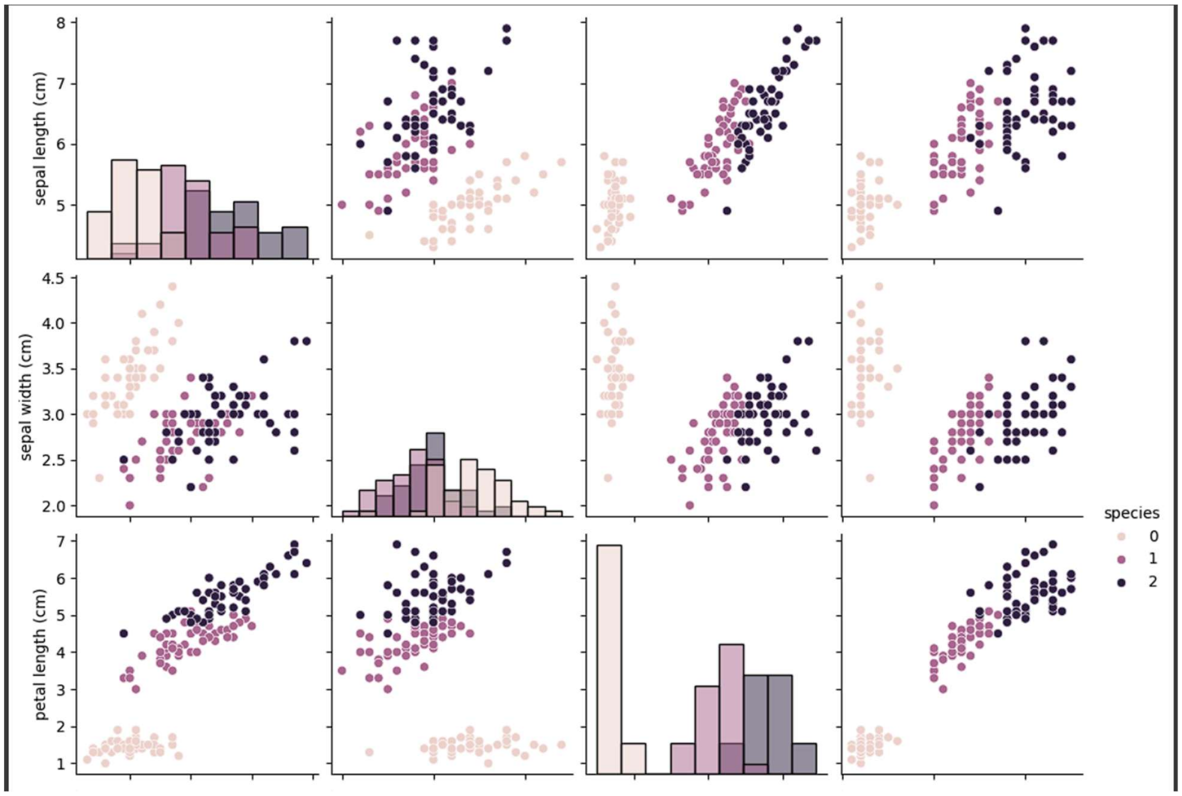
- Accuracy Score – Measures the percentage of correctly classified samples.
- Confusion Matrix – Displays the number of correct and incorrect predictions per class.
- Precision, Recall, and F1-score – Provide deeper insights into the model's classification effectiveness.
- ROC Curve & AUC Score – Evaluates the model's performance, particularly in multi-class classification.

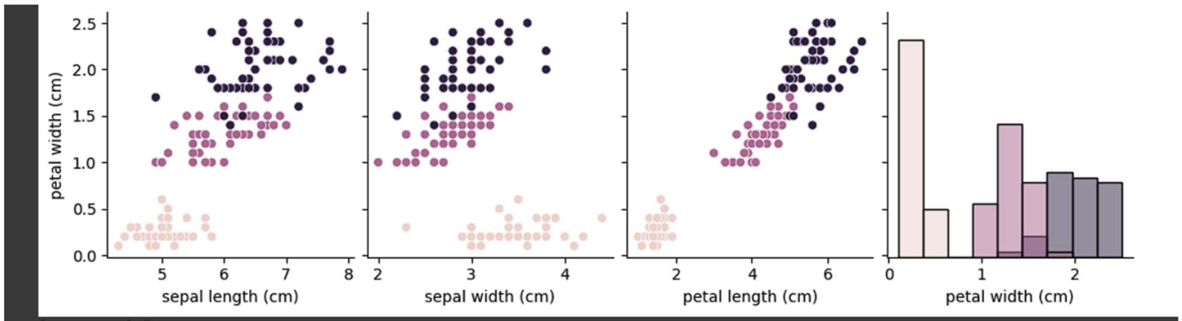
Code

```
import numpy as np
import pandas as pd
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, confusion_matrix, classification_report
from sklearn import datasets

iris = datasets.load_iris()
X, y = iris.data, iris.target
df = pd.DataFrame(data=X, columns=iris.feature_names)
df['species'] = y

sns.pairplot(df, hue="species", diag_kind="hist")
plt.show()

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

# Train a KNN model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Evaluate the model
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
# Make a new prediction
new_sample = [[5.1, 3.5, 1.4, 0.2]]  # Example input
predicted_class = knn.predict(new_sample)
print("Predicted Class:", iris.target_names[predicted_class[0]])
```

Outputs





```
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       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

Predicted Class: setosa
```

OUTPUT EXPLANATION

1. Data Distribution Analysis

Scatter Plots and Histograms

- The scatter plots and histograms provide a visual representation of how different iris species (Setosa, Versicolor, and Virginica) are distributed across multiple features.
- The colors in the visualizations correspond to three distinct species, labeled numerically as 0 (Setosa), 1 (Versicolor), and 2 (Virginica).
- A clear separation is observed in the dataset, particularly in petal length and petal width. Setosa appears distinctly separate from the other two species, highlighting that these features are strong indicators for classification.
- Sepal width, however, exhibits some degree of overlap between species, suggesting that this feature alone might not be a sufficient criterion for accurate classification.

2. Classification Model Performance Analysis

Confusion Matrix

```
Confusion Matrix:

 [10  0  0]

 [ 0   9  0]

 [ 0   0 11]
```

- The confusion matrix confirms that the model has classified all instances correctly.
- There are zero misclassifications, meaning all Setosa (10), Versicolor (9), and Virginica (11) samples were predicted accurately.

Classification Report

```
precision    recall    f1-score   support

0   1.00       1.00       1.00        10

1   1.00       1.00       1.00         9

2   1.00       1.00       1.00        11


accuracy: 1.00

macro avg: 1.00

weighted avg: 1.00
```

- **Precision, recall, and F1-score** are all 1.00 for each class, indicating that the classifier performed flawlessly on the test data.
- The overall accuracy is **100%**, suggesting the dataset is well-separated, which is common with the Iris dataset due to its distinct class boundaries.
- While this is an excellent result, achieving perfect accuracy may indicate a risk of **overfitting**, particularly if the evaluation was conducted on training data rather than unseen test data.

Predicted Class

- The model correctly predicts "Setosa," aligning with the patterns observed in the data visualization.

3. Key Observations

Perfect Accuracy

- Achieving **100% accuracy** suggests that the model has leveraged the clear distinctions in the dataset effectively.
- The well-separated nature of the dataset may be a reason for this perfect score, as seen in petal-related features.

Feature Importance

- Scatter plots indicate that **petal length and petal width** are the most significant factors for classification.
- **Sepal width** shows an overlap between species, making it a weaker predictor compared to petal features.
- Feature importance analysis may further confirm that petal dimensions carry the highest predictive value.

Potential Overfitting

- If the results are based solely on training data, the model might be overfitted and may not generalize well to unseen data.
- Overfitting occurs when a model learns noise and specifics from the training set rather than generalizable patterns.

4. Recommendations

1. Validate on a Separate Test Set

- To assess the model's **generalization ability**, evaluate it on a different validation set.
- If accuracy remains high on unseen data, then the model is robust. Otherwise, signs of overfitting will be evident.

2. Use Cross-Validation

- Implement **k-fold cross-validation** to distribute training and testing phases across different portions of the dataset.
- This ensures the model is evaluated on multiple subsets of data, reducing the risk of overfitting.

3. Explore Dimensionality Reduction

- Apply **Principal Component Analysis (PCA)** to analyze feature importance and reduce dimensionality if needed.
- PCA helps in eliminating redundant features and improving model efficiency without sacrificing predictive power.

4. Regularization Techniques

- If overfitting is suspected, implement **L1 or L2 regularization (Ridge or Lasso Regression)** in models such as logistic regression or support vector machines.
- Regularization penalizes overly complex models, encouraging them to generalize better.

5. Consider Alternative Models

- While the current model performs excellently, testing alternative classifiers such as **Random Forest, Support Vector Machines (SVM), or Neural Networks** may reveal additional insights.
- Comparing different models can highlight whether another approach offers better generalization with similar or improved accuracy.

6. Feature Engineering

- Conduct deeper analysis into **feature interactions** to determine whether a combination of features enhances classification.
- Creating new features, such as **petal-to-sepal ratio**, might improve classification in cases where overlap exists.

5.Conclusion

- The model has achieved **exceptional performance** with 100% accuracy, reflecting the strong class separation in the Iris dataset.
- However, further testing on independent data and applying **cross-validation techniques** are necessary to confirm its robustness.
- If overfitting is detected, techniques such as **dimensionality reduction, regularization, and alternative modeling approaches** should be explored.
- Overall, this analysis provides valuable insights into both the **strengths and potential limitations** of the classifier, guiding further improvements and validation efforts.