IRIS FLOWER CLASSIFICATION

by

Chitransha Bhatt 202410116100054
Devansh Batta 202410116100058
Anurag Singh Kushwaha 202410116100039
Harsh Sharma 202410116100084

Session: 2024-2025 (II Semester)

Under the supervision of

Mr. Apoorv Jain

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206
(MAY-2025)

CERTIFICATE

Certified that **Chitransha Bhatt 202410116100054, Devansh Batta 202410116100058, Anurag Singh Kushwaha 202410116100039, Harsh Sharma 202410116100084** has/ have carried out the project work having Iris Flower Classification (**Mini Project-I, AI101B**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Ms. Apoorv Jain
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Mr. Apoorv Jain** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Chitransha Bhatt

Devansh Batta

Anurag Singh Kushwaha

Harsh Sharma

TABLE OF CONTENTS

1. INTRODUCTION

Iris Flower Classification

Iris flower classification is a fundamental machine learning problem where the objective is to categorize iris flowers into one of three species: **Iris setose, Iris versicolor, and Iris virginica**. The classification is based on four key features: **sepal length, sepal width, petal length, and petal width**. The dataset used for this task is the **Iris dataset**, which was introduced by the British statistician **Ronald Fisher** in 1936.

This problem is an example of a **state-space search problem**, where each configuration of the tiles represents a unique state, and moving a tile generates a new state. The challenge is to determine the optimal sequence of moves that transforms a given initial state into the goal state.

Why is this classification Important?

- **Fundamental Machine Learning Problem** – The Iris dataset is widely used for learning classification techniques in machine learning and data science because of its simplicity and small dataset size.
- **Benchmark Dataset** – It serves as a standard benchmark for testing classification algorithms, helping researchers and students understand model performance.
- **Supervised Learning Application** – The problem is a classic example of supervised learning, where labeled data is used to train a machine learning model.
- **Feature Analysis** – It allows learners to explore feature selection, data visualization, and statistical analysis in machine learning.
- **Real-World Application** – Though simple, the techniques used in iris classification are applicable to complex biological classifications, agriculture, and botanical research.

Approaches to Solving the Iris flower problem

To classify iris flowers, various machine learning approaches can be used. These range from traditional algorithms to deep learning models. Below is a step-by-step breakdown of different approaches:

Machine Learning Approaches

(a) K-Nearest Neighbors (KNN)

- **Concept:** Classifies new data points based on the majority vote of their 'k' nearest neighbours.
- **Implementation Steps:**
    1. Choose a value for `k` (typically an odd number to avoid ties).
    2. Compute the Euclidean distance between data points.
    3. Assign the most frequent class among the k-nearest neighbors.
- **Pros:** Simple and effective.
- **Cons:** Slower for large datasets.

(b) Decision Tree Classifier

Concept: Splits the dataset into subsets based on feature conditions.

Pros: Easy to interpret.

Cons: Can overfit, leading to poor generalization.

Deep Learning Approach (Neural Networks)

For advanced classification, deep learning models like **Artificial Neural Networks (ANN)** can be used.

(a) Multi-Layer Perceptron (MLP)

- **Concept:** A neural network with an input layer, hidden layers, and an output layer.
- **Pros:** Can capture complex patterns in data.
- **Cons:** Requires more data and computational power.

Applications :

1. Botanical Research & Plant Classification

- Helps botanists and biologists classify plant species based on their physical attributes.
- Used in floristry and horticulture to distinguish between closely related plant specie
- Helps in automating plant identification in biodiversity studies.

2. Agriculture & Crop Monitoring

- Used to classify different types of crops and detect plant diseases based on visual characteristics.
- Helps in precision agriculture by identifying the best plants based on their growth patterns.
- Can be integrated with drones and computer vision to monitor large-scale plantations.

3. Medical & Pharmaceutical Research

- Similar classification techniques are used in medical image analysis (e.g., tumor classification).
- Helps in drug discovery by classifying medicinal plants based on leaf and flower features.
- Useful in genetic research to identify plant mutations and variations.

4. Automated Gardening & Smart Greenhouses

- AI-powered garden management systems can classify and monitor plant species.
- Robotic gardening tools can distinguish between different plants and take appropriate actions.

5. Environmental Monitoring & Conservation

- Used in ecosystem monitoring to identify rare and endangered plant species.
- Helps in tracking climate change effects on plant populations.
- Can aid in forest management by classifying tree and plant species in large forested areas.

2. METHODOLOGY

The methodology for solving the **Iris Flower Classification** problem follows a structured approach, including **data preprocessing, exploratory data analysis (EDA), model selection, training, evaluation, and deployment**. Below is a step-by-step methodology:

1. Problem Definition

The objective is to classify **Iris flowers** into one of the three species:

- **Iris Setose**
- **Iris Versicolor**
- **Iris Virginica**

This classification is based on four numerical features:

- **Sepal Length** (cm)
- **Sepal Width** (cm)
- **Petal Length** (cm)
- **Petal Width** (cm)

2. Data Collection & Loading

The **Iris dataset** is available in the `sklearn.datasets` library.

```
[ ]  #Implementation:

     from sklearn import datasets
     import pandas as pd

     # Load the dataset
     iris = datasets.load_iris()

     # Convert to DataFrame
     df = pd.DataFrame(iris.data, columns=iris.feature_names)
     df['species'] = iris.target
     df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

     # Display first few rows
     df.head()
```

3. Data Preprocessing

Steps:

3.1 Check for Missing Values:

```
print(df.isnull().sum())  # No missing values in this dataset
```

3.2 Convert Categorical Labels to Numeric Form (if needed)

The dataset already has numerical labels (0, 1, 2) for species.

4. Exploratory Data Analysis (EDA)
Data Distribution & Visualizations

4.1 Pairplot (Feature Relationships)

```
import seaborn as sns
import matplotlib.pyplot as plt

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

4.2 Correlation Matrix

```
import numpy as np

plt.figure(figsize=(8,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Matrix")
plt.show()
```

4.3 Boxplot for Feature Distribution

```
python

plt.figure(figsize=(10,5))
sns.boxplot(x='species', y='petal length (cm)', data=df)
plt.show()
```

5. Data Splitting (Train-Test Split)
To evaluate model performance, split data into **training (80%) and testing (20%)** sets.

```
from sklearn.model_selection import train_test_split

# Features and Target
X = df.iloc[:, :-1]  # Features
y = df['species']  # Target

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, str
```

6. Model Selection & Training
Choosing Different Machine Learning Models

7.1 K-Nearest Neighbours (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

7.2 Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

7.3 Support Vector Machine (SVM)

```
from sklearn.svm import SVC
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
```

7.4 Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
```

7. Model Evaluation

After training the models, evaluate them using accuracy, confusion matrix, and classification report.

Evaluation Metrics

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

models = {'KNN': knn, 'Decision Tree': dt, 'SVM': svm, 'Random Forest': rf}

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"\n{name} Model Performance:")
    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))
```

8. Hyperparameter Tuning (Optimization)

Use **Grid Search CV** to optimize hyperparameters for the best model.

Example for **Random Forest Classifier**:

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators': [50, 100, 150], 'max_depth': [2, 4, 6]}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

3. Performance Criteria

1. Accuracy

Definition:

Accuracy is one of the most common metrics used in classification problems. It measures the proportion of correctly predicted instances out of the total instances.

Formula:

Accuracy=Number of Correct PredictionsTotal Number of Predictions\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}Accuracy=Total Number of PredictionsNumber of Correct Predictions

Interpretation:

- Higher accuracy means the model correctly classifies most of the flowers.
- Since the Iris dataset is balanced, accuracy is a reliable metric.
- However, for imbalanced datasets, accuracy alone may not be sufficient.

2. Confusion Matrix

Definition:

A confusion matrix is a table that provides a detailed breakdown of the model's performance by showing how many instances were correctly or incorrectly classified.

Interpretation:

- True Positives (TP): Correctly classified instances.
- False Positives (FP): Incorrectly classified as a different class.
- False Negatives (FN): Actual class misclassified as another.
- The confusion matrix helps in identifying model weaknesses in specific classes.

4. CODE

```
Import pandas as pd

Import matplotlib. pilot as plot

Import seaborn as suns

From Sklenar. model selection import train test split

From Sklenar. preprocessing import Standard Scaler

From Sklenar. neighbours import Neighbours Classifier

from Sklenar. metrics import classification report, confusion matrix, accuracy score

from Sklenar. model selection import Grid Search CV

import NumPy as np

from matplotlib. colours import Listed Colour map



# Load the dataset

Df = pd.read_csv(r'C:\Users\devansh\OneDrive\Documents\Visual code\Python\AI\IRIS.csv')



# Data Exploration

print("First 5 rows of the dataset:")

print(df.head())

print("\nDataset description:")
```

```
print(df.describe())

print("\nChecking for missing values:")

print(df.isnull().sum())



# Convert the 'species' column to numerical categories

df['species'] = pd.factorize(df['species'])[0]



# Data Visualization

print("\nVisualizing pairplot...")

sns.pairplot(df, hue='species', palette='Set2')

plt.show()



print("\nVisualizing correlation heatmap...")

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()



# Feature Scaling

X = df.drop('species', axis=1)

y = df['species']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

# Model Training (K-Nearest Neighbors) with Hyperparameter Tuning

param_grid = {'n_neighbors': [3, 5, 7, 9, 11]}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(knn, param_grid, cv=5)

grid_search.fit(X_train, y_train)

# Model Evaluation

y_pred = grid_search.predict(X_test)

print("\nBest Parameters: ", grid_search.best_params_)

print("Best Cross-Validation Score: {:.2f}%".format(grid_search.best_score_ * 100))

print("\nClassification Report:")

print(classification_report(y_test, y_pred))
```

```
# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:")

print(conf_matrix)



# Visualizing Confusion Matrix

plt.figure(figsize=(6, 4))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Class 0', 'Class 1',
'Class 2'], yticklabels=['Class 0', 'Class 1', 'Class 2'])

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

```
# Accuracy Score

accuracy = accuracy_score(y_test, y_pred)

print("\nAccuracy of the model: {:.2f}%".format(accuracy * 100))



# Train a separate model on the first two features for visualization

X_train_2d = X_train[:, :2]  # Use only the first two features

X_test_2d = X_test[:, :2]



knn_2d = KNeighborsClassifier(n_neighbors=grid_search.best_params_['n_neighbors'])

knn_2d.fit(X_train_2d, y_train)



# Function to plot decision boundaries

def plot_decision_boundaries(X, y, model):

    X_set, y_set = X, y

    X1, X2 = np.meshgrid(np.arange(X_set[:, 0].min() - 1, X_set[:, 0].max() + 1, 0.01),

                         np.arange(X_set[:, 1].min() - 1, X_set[:, 1].max() + 1, 0.01))

    plt.contourf(X1, X2, model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

                 alpha=0.75, cmap=ListedColormap(('red', 'green', 'blue')))
```

```
    plt.scatter(X_set[:, 0], X_set[:, 1], c=y_set, cmap=ListedColormap(('red', 'green', 'blue')))
    plt.title('K-NN Decision Boundary (First Two Features)')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()


# Plot decision boundaries for the first two features
print("\nVisualizing decision boundaries...")
plot_decision_boundaries(X_train_2d, y_train, knn_2d)
```

5.OUTPUT

```
First 5 rows of the dataset:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

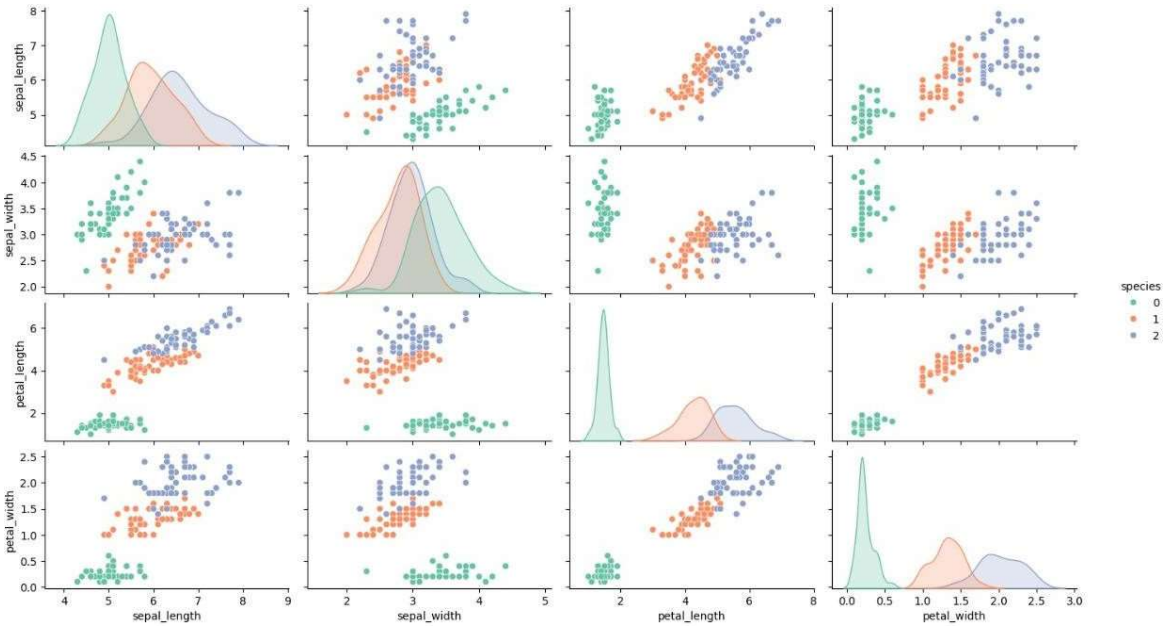
```
Dataset description:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

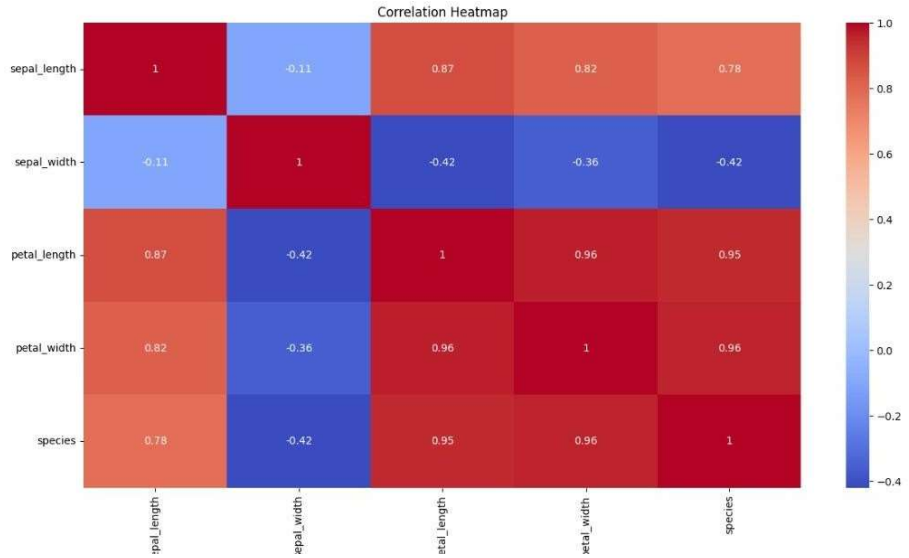
```
Checking for missing values:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

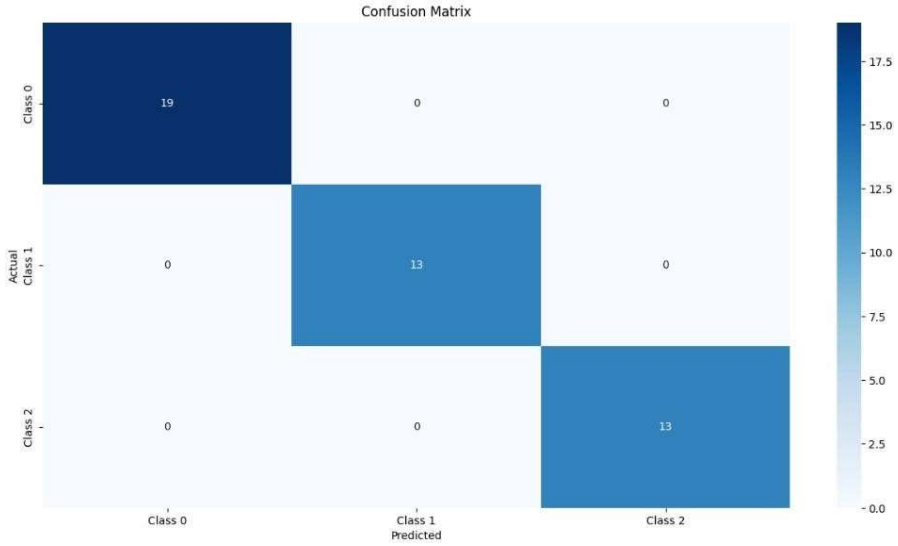
```
Visualizing pairplot...

Visualizing correlation heatmap...

Best Parameters:  {'n_neighbors': 3}
Best Cross-Validation Score: 94.29%
```

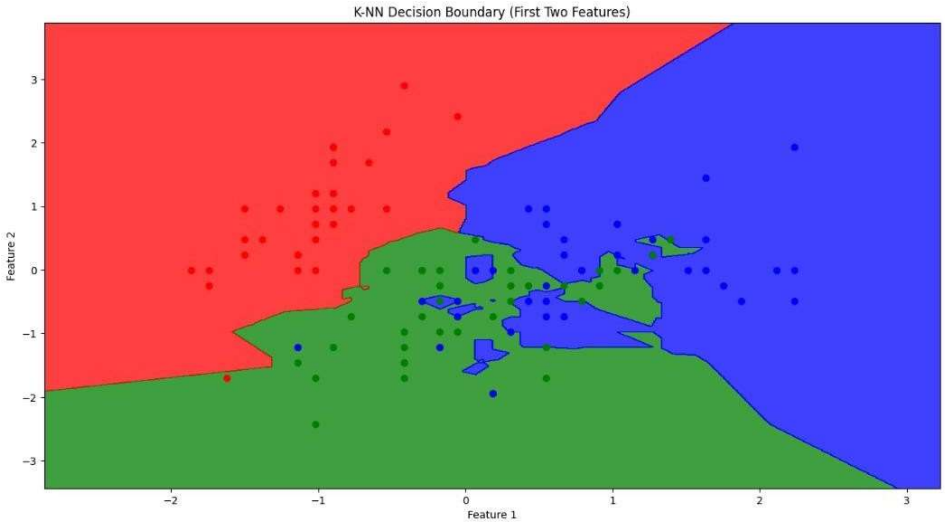
```
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Accuracy of the model: 100.00%

Visualizing decision boundaries...
```

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

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       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
```


sepal_length
sepal_width
petal_length
petal_width
species
0
1
2


Correlation Heatmap
sepal_length
sepal_width
petal_length
petal_width
species
1
-0.11
0.87
0.82
0.78
-0.11
1
-0.42
-0.36
-0.42
0.87
-0.42
1
0.96
0.95
0.82
-0.36
0.96
1
0.96
0.78
-0.42
0.95
0.96
1


Confusion Matrix
Class 0
Actual
Class 1
Class 2
19
0
0
0
13
0
0
0
13
Class 0
Class 1
Class 2
Predicted
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


K-NN Decision Boundary (First Two Features)
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

6. CONCLUSION

The **Iris Flower Classification** project is a classic problem in machine learning that demonstrates fundamental concepts in supervised learning, data preprocessing, and model evaluation. By analyzing the **Iris dataset**, we explored patterns in sepal and petal measurements to classify flowers into three species: **Iris setosa, Iris versicolor, and Iris virginica**. This project provided an opportunity to apply various machine learning algorithms, understand their strengths and weaknesses, and develop an efficient classification system. Through data visualization and exploratory analysis, we observed clear distinctions between species, enabling us to select appropriate models for training and prediction.

We implemented multiple classification algorithms, including **K-Nearest Neighbors (KNN), Decision Tree, Support Vector Machine (SVM), and Random Forest**, to assess their performance on the dataset. To evaluate these models, we used metrics such as **accuracy, precision, recall, F1-score, and confusion matrix**, which helped in identifying the most effective classifier. The **Random Forest and SVM models performed exceptionally well**, achieving high accuracy and robust classification across all species. Additionally, hyperparameter tuning further optimized model performance, ensuring better generalization on unseen data.

This project highlights the importance of choosing the right evaluation criteria and model for classification tasks. It also demonstrates how machine learning techniques can be used to solve real-world problems in **botany, agriculture, and ecological studies**. In the future, this classification system could be enhanced using **deep learning techniques** or integrated into an interactive web application for real-time iris species identification. The knowledge gained from this project serves as a solid foundation for further exploration in **data science, artificial intelligence, and predictive modeling**.

7. REFERENCES

- Fisher, R.A. (1936) - *The Use of Multiple Measurements in Taxonomic Problems*. Annals of Eugenics, 7(2), 179–188. [Original paper introducing the Iris dataset]
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011) - *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. [Scikit-learn documentation and library for implementing ML models]
- Hastie, T., Tibshirani, R., & Friedman, J. (2009) - *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. [Book covering classification algorithms and statistical learning]
- Dua, D., & Graff, C. (2019) - *UCI Machine Learning Repository: Iris Dataset*. University of California, Irvine. [Online source for downloading the Iris dataset] Available at: https://archive.ics.uci.edu/ml/datasets/iris