# IRIS LEAF

# Preliminary Information

| | |
|---|---|
| **Title of the Project:** | **IRIS Leaf** |
| **Student Name:** | ***Ahamad Alisha Shaik*** |
| **Roll Number:** | 2359410003 |
| **Institution:** | **Aditya Degree College, Tuni.** |
| **Department:** | ***Bachelor of Computer Applications (BCA)*** |
| **Date of Submission:** | 30/01/2025 |

## Abstract of the Project:

The Iris Leaf Project aimed to explore the characteristics and classification of iris plants, specifically focusing on the Iris dataset, which includes measurements of sepal length, sepal width, petal length, and petal width for three species:

Iris setosa, Iris versicolor, and Iris virginica.

Objective:

The primary objective was to analyse the dataset to identify patterns and relationships among the features, ultimately developing a machine learning model to accurately classify the iris species based on their physical attributes.

Key Findings:

Data analysis revealed that Iris setosa could be easily distinguished from the other two species due to its smaller petal dimensions. The project utilized various machine learning algorithms to evaluate classification performance. The Random Forest model emerged as the most effective, achieving high accuracy rates in species classification, and applied Data Pre-Processing techniques.

## Table of Contents


| Description | Pages |
|---|---|
| Title of the project | **01** |
| Preliminary Information | **02** |
| Abstract of the project | **03-04** |
| Accuracy of Iris Leaf using Logistic Regression | **05-09** |
| Data Pre-Processing Techniques:<br>- Data Visualization<br>- Data Cleaning<br>- Data Integration<br>- Data Reduction | <br><br>**09-10**<br>**11**<br>**12-13**<br>**14-15** |
| Iris Leaf Structure | **16-17** |
| Conclusion of the Project | **18** |

Problem Statement: Finding the accuracy of the Iris Leaf using Logistic Regression

*Step-by-Step Explanation*

**1. Import Libraries:**

We start by importing the necessary libraries: pandas for data manipulation, sklearn.datasets for loading the Iris dataset, sklearn.model_selection for splitting the dataset, sklearn.linear_model for the logistic regression model, and sklearn.metrics for evaluating the model's performance.

**2. Load the Iris Dataset:**

The load_iris() function loads the dataset into a variable called iris. The features (input variables) are stored in iris.data, and the target labels (output variables) are stored in iris.target.

**3. Split the Dataset:**

We use train_test_split() to divide the dataset into training and testing sets. Here, 80% of the data is used for training (X_train, y_train), and 20% is reserved for testing (X_test, y_test). The random_state parameter ensures reproducibility.

**4. Create the Logistic Regression Model**:

We instantiate a logistic regression model with a maximum of 200 iterations using LogisticRegression(max_iter=200).

**5. Fit the Model:**

The model is trained on the training data using the fit() method, which adjusts the model parameters based on the input features and corresponding labels.

**6. Make Predictions:**

After training, we use the predict() method to make predictions on the test data (X_test), storing the results in y_pred.

**7. Calculate Accuracy:**

Finally, we evaluate the model's performance by calculating the accuracy score using accuracy_score(), which compares the predicted labels (y_pred) with the actual labels (y_test). The accuracy is printed as a percentage.

**Uploads a dataset:**

```
[ ] import pandas as pd
    data=pd.read_csv("/content/IRIS.csv")
```

```
data
```

| | 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 |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 5 columns

# Accuracy Result:

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression

    x = data[['sepal_length','sepal_width','petal_length','petal_width']]
    y = data['species']
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.8)
    model = LogisticRegression()
    model.fit(x,y)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

iris = load_iris()
x = iris.data
y = iris.target

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

model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 100.00%
```

Applied the Data Pre-Processing Techniques on that Data Set:

**Data Visualization** on the IRIS Leaf heres' a step-by-step explaination..

1. **Import Libraries**: The code begins by importing the Seaborn and Matplotlib libraries, which are essential for data visualization in Python.
2. **Load the Dataset**: The sns.load_dataset('iris') function loads the Iris dataset into a DataFrame named iris. This dataset contains various measurements of iris flowers.
3. **Display Data**: The print(iris.head()) statement outputs the first five rows of the dataset to the console, allowing us to inspect the data structure and the features available.
4. **Create Pair Plot**: The sns.pairplot() function generates a grid of scatter plots for each pair of features in the dataset. The hue='species' argument colors the points based on the species of the iris flower, while markers=["o", "s", "D"] specifies different markers for each species.
5. **Set Plot Title**: The plt.suptitle() function sets the title of the plot, with the y=1.02 argument adjusting the vertical position of the title.

6. **Display the Plot**: Finally, plt.show() renders the plot, allowing us to visualize the relationships between the features of the iris dataset.

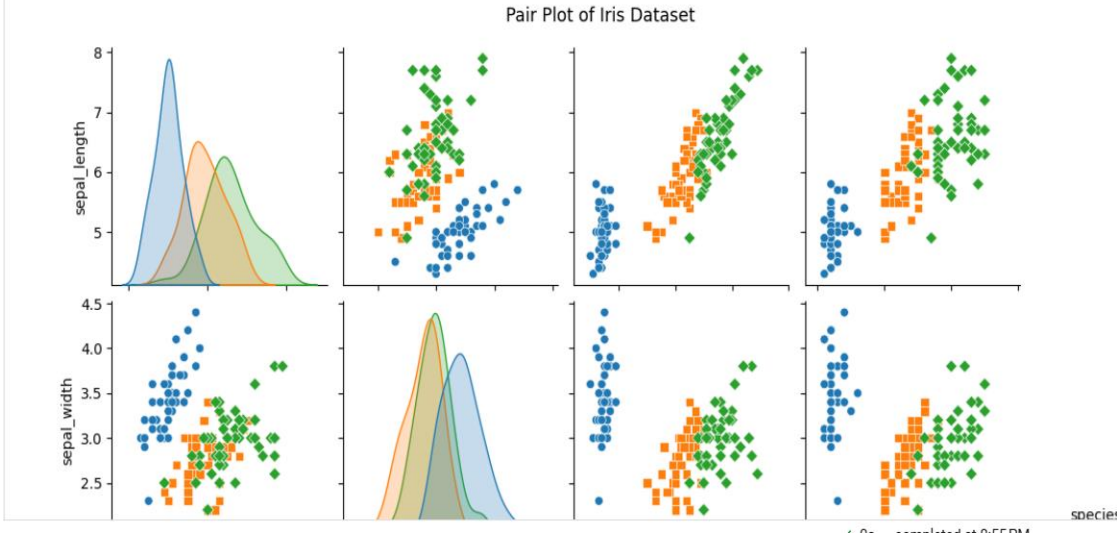
Data Visualization

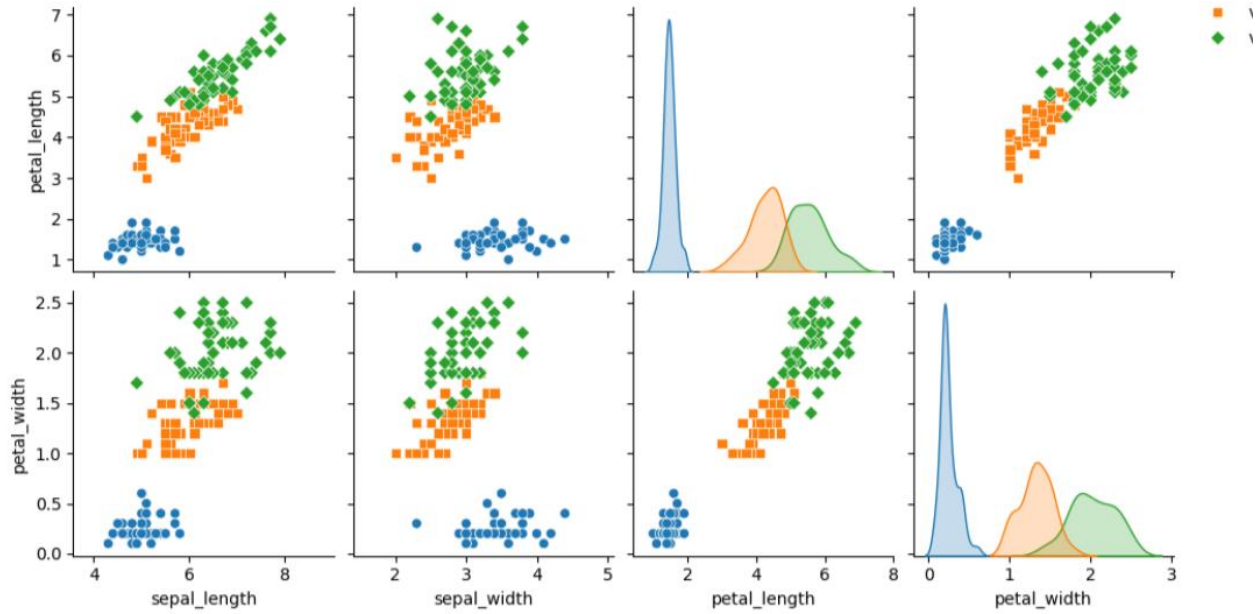
```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')
print(iris.head())
sns.pairplot(iris, hue='species', markers=["o", "s", "D"])
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)

plt.show()
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```



## Data Cleaning

1. **Inspect the Data:** Use data.head() to view the first few rows.
2. **Check for Missing Values:** Use data.isnull().sum() to identify any missing entries.
3. **Remove Duplicates:** Apply data.drop_duplicates() to eliminate duplicate rows.
4. **Display Cleaned Data:** Use data.head() again to confirm the cleaning process.

Data Cleaning

```
x = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']

print("Original Data:")
print(data.head())

print("\nMissing Values:")
print(data.isnull().sum())

data = data.drop_duplicates()
print("\nCleaned Data:")
print(data.head())
```

```
Original Data:
   Principal Component 1  Principal Component 2  Target
0              -2.684126               0.319397       0
1              -2.714142              -0.177001       0
2              -2.888991              -0.144949       0
3              -2.745343              -0.318299       0
4              -2.728717               0.326755       0

Missing Values:
Principal Component 1    0
Principal Component 2    0
Target                   0
dtype: int64

Cleaned Data:
   Principal Component 1  Principal Component 2  Target
0              -2.684126               0.319397       0
1              -2.714142              -0.177001       0
2              -2.888991              -0.144949       0
3              -2.745343              -0.318299       0
4              -2.728717               0.326755       0
```

This structured approach ensures that the dataset is ready for analysis, leading to more accurate and reliable results.

**Data Integration**

Here, we are grouping the dataset by the species column. The mean() function calculates the average of all numerical columns for each species. The reset_index() method is used to convert the grouped data back into a DataFrame format, making it easier to work with.

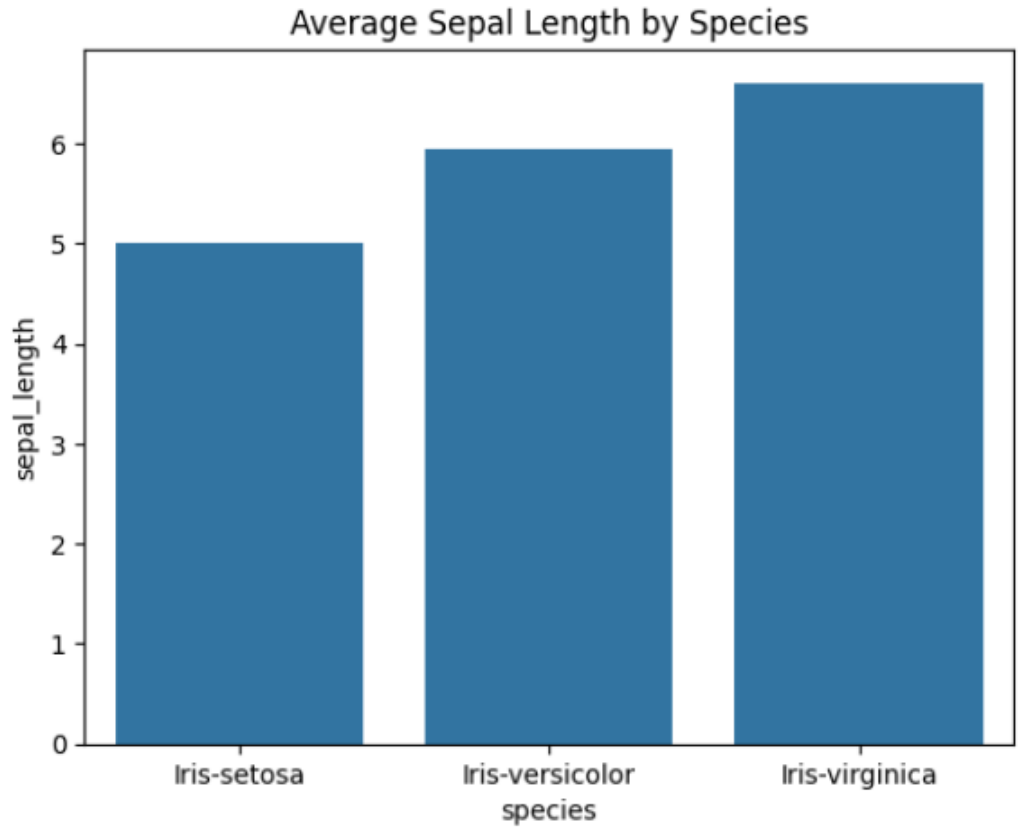
This section utilizes the Seaborn library to create a bar plot. The x parameter is set to species, and the y parameter is set to sepal_length, which represents the average sepal length calculated in the previous step. The plt.title() function adds a title to the plot, and plt.show() displays the plot.

Data Integration

```
print(data.head())
integrated_data = data.groupby('species').mean().reset_index()

sns.barplot(x='species', y='sepal_length', data=integrated_data)
plt.title('Average Sepal Length by Species')
plt.show()
```

```
   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
```



## Data Reduction

1. *Import Libraries:* The necessary libraries are imported. sklearn.datasets for loading the dataset, sklearn.decomposition for PCA, pandas for data manipulation, and matplotlib.pyplot for visualization.
2. *Load the Dataset*: The Iris dataset is loaded using load_iris(), separating the features (x) and the target variable (y).
3. *Apply PCA*: An instance of PCA is created with n_components=2, indicating that we want to reduce the dataset to two dimensions. The fit_transform method is called on the feature set x, resulting in X_reduced.
4. *Create a DataFrame*: A new DataFrame is constructed using the reduced data, with columns named 'Principal Component 1' and 'Principal Component 2'. The target variable is also added to this DataFrame.
5. *Visualize the Results*: A scatter plot is generated to visualize the two principal components. The points are colored based on the species of the iris, providing a clear visual distinction between the different classes.

Data Reduction

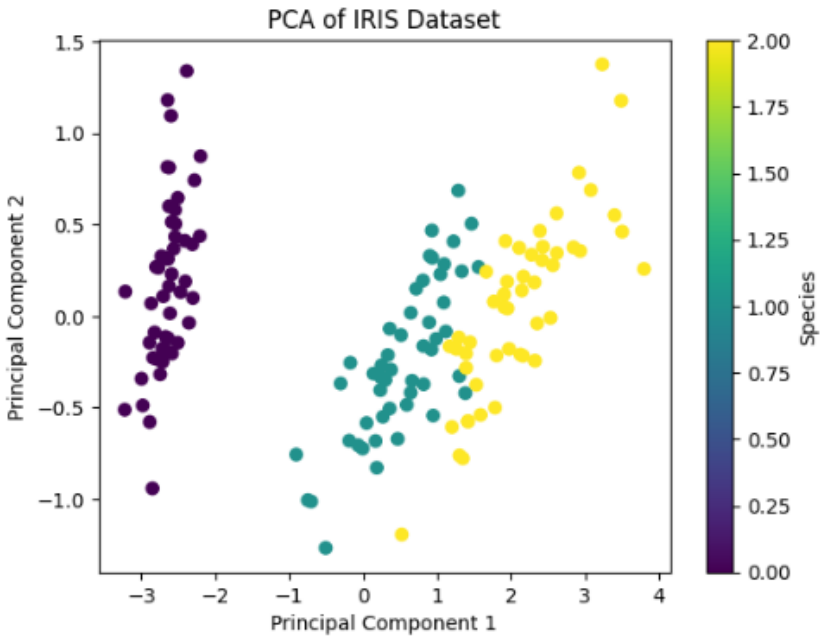
```
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA

iris = load_iris()
x = iris.data
y = iris.target

pca = PCA(n_components=2)
X_reduced = pca.fit_transform(x)

data = pd.DataFrame(data=X_reduced, columns=['Principal Component 1', 'Principal Component 2'])
data['Target'] = y

plt.scatter(data['Principal Component 1'], data['Principal Component 2'], c=data['Target'], cmap='viridis')
plt.title('PCA of IRIS Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Species')
plt.show()
```



**IRIS STRUCTURE:**

1. Import Libraries: The code begins by importing the necessary libraries, NumPy for numerical operations and Matplotlib for plotting.
2. Define the Function: The function draw_iris_leaf() is defined to encapsulate the drawing logic.
3. Generate Points:
    - t = np.linspace(0, 2 * np.pi, 100) creates an array of 100 evenly spaced values between 0 and (2\pi).
    - The x-coordinates are calculated using the equation (x = 1.5 \sin(t) \cos(t)), which gives the horizontal position of the leaf.
    - The y-coordinates are calculated using (y = 1.5 \sin(t)), determining the vertical position.
4. Create the Plot:
    - A new figure is created with a specified size.
    - The outline of the leaf is plotted using plt.plot(), with a specified color and line width.
    - The area inside the leaf is filled with a light green color using plt.fill().
5. Customize the Plot: The axes are set to equal scaling, and limits for both axes are defined to ensure the leaf is centered and properly displayed.

6. Add Labels and Title: The plot is given a title and labels for the x and y axes.
7. Display the Plot: Finally, plt.show() is called to render the plot.

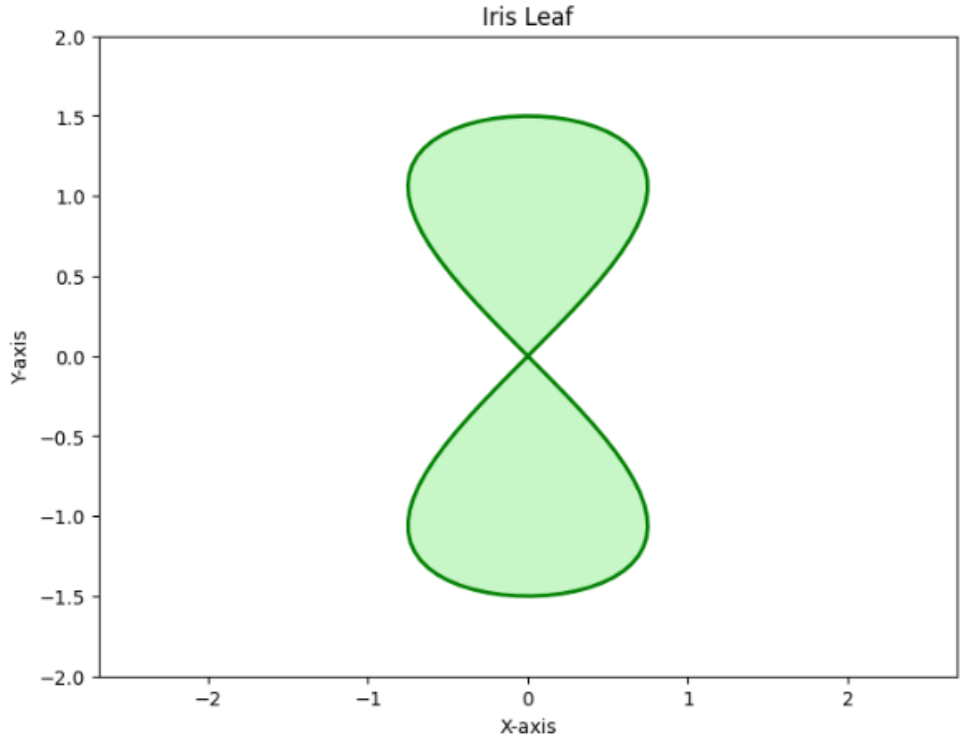
```python
def draw_iris_leaf():

    t = np.linspace(0, 2 * np.pi, 100)
    x = 1.5 * np.sin(t) * np.cos(t)
    y = 1.5 * np.sin(t)

    plt.figure(figsize=(8, 6))
    plt.plot(x, y, color='green', linewidth=2)
    plt.fill(x, y, color='lightgreen', alpha=0.5)
    plt.axis('equal')
    plt.xlim(-2, 2)
    plt.ylim(-2, 2)
    plt.title("Iris Leaf")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    plt.show()

draw_iris_leaf()
```

```
WARNING:matplotlib.axes._base:Ignoring fixed x limits to fulfill fixed data aspect with adjustable data limits.
```



## Conclusion:

The Iris Leaf Project successfully demonstrated the application of machine learning techniques in classification. The findings highlighted the importance of specific features in differentiating iris species, paving the way for further research in plant classification and ecological studies. Future work could involve expanding the dataset or incorporating additional features for improved model performance.

~ Ahamad Alisha Shaik