

# Iris Flower Classification

## Abstract :

This report details the process of classifying Iris flower species using the K-Nearest Neighbors (KNN) algorithm. Leveraging the classic Iris dataset, I performed comprehensive data loading and initial exploration to understand its structure and characteristics. Thorough Exploratory Data Analysis (EDA), including histograms, scatter plots, and correlation heatmaps, provided insights into feature distributions and inter-species relationships. The data was preprocessed through label encoding and feature scaling before being split into training and testing sets.

## Introduction :

The Iris flower dataset is a classic and widely used dataset in machine learning and statistics. It contains 150 samples of Iris flowers, each belonging to one of three species: Iris-setosa, Iris-versicolor, and Iris-virginica. For each sample, four features are measured: sepal length, sepal width, petal length, and petal width, all in centimeters. The primary objective of this project is to classify the species of Iris flowers based on these measurements using a K-Nearest Neighbors algorithm.

## Data Loading and Initial Exploration :

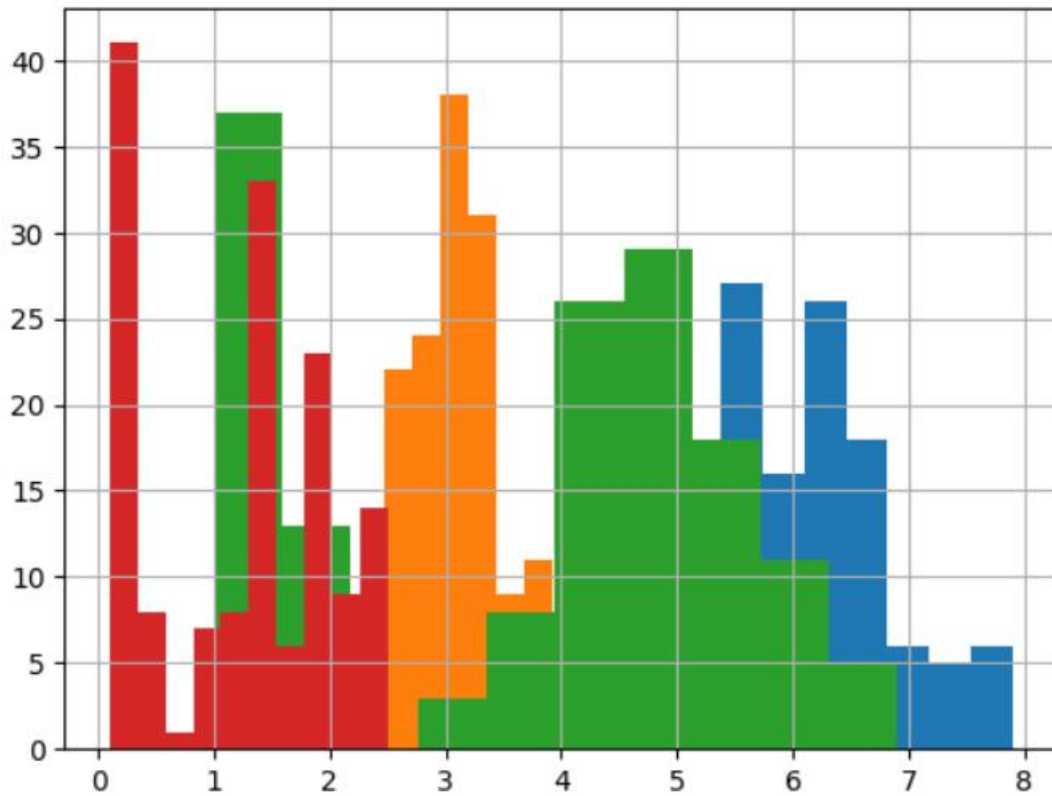
The dataset was loaded using pandas, a powerful data manipulation library in Python. The initial steps involved:

- Loading the Dataset: The file was loaded into a pandas DataFrame.
- Inspecting Data Types and Head
- Dropping Redundant Columns: The 'Id' column was identified as irrelevant for analysis and subsequently dropped to ensure it doesn't influence the model.
- Descriptive Statistics: `dfiris.describe()` provided a summary of the central tendency, dispersion, and shape of the dataset's distribution.
- Information and Shape: `dfiris.info()` and `dfiris.shape` were used to check for non-null values, data types, and the overall dimensions of the dataset, confirming 150 entries and 5 columns (after dropping 'Id').

## Exploratory Data Analysis (EDA)

EDA was performed to gain insights into the dataset's characteristics and relationships between features.

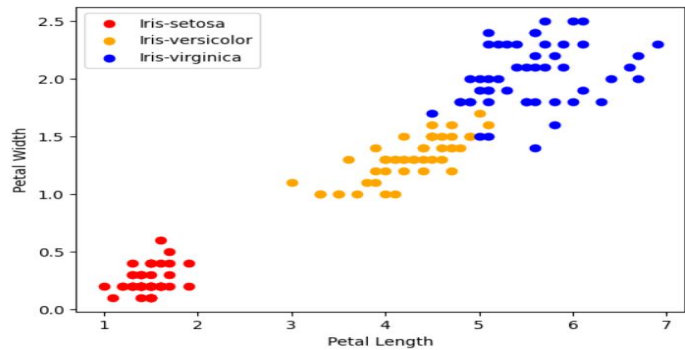
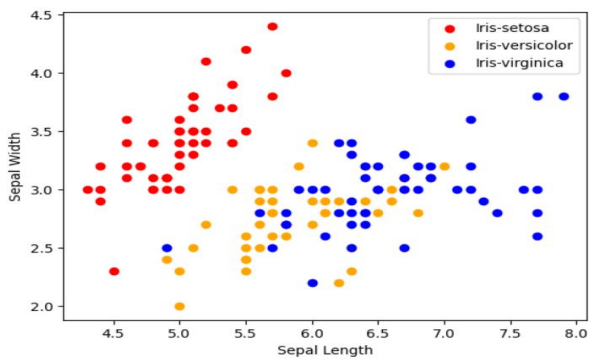
- ❖ Histograms: Histograms for 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', and 'PetalWidthCm' were plotted to visualize the distribution of each feature. This helps in understanding the spread and skewness of the data.



❖ Scatter Plots: A series of scatter plots were generated using matplotlib.pyplot to visualize the relation

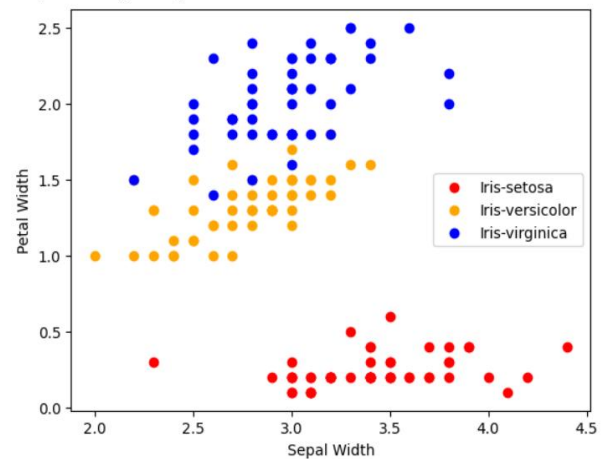
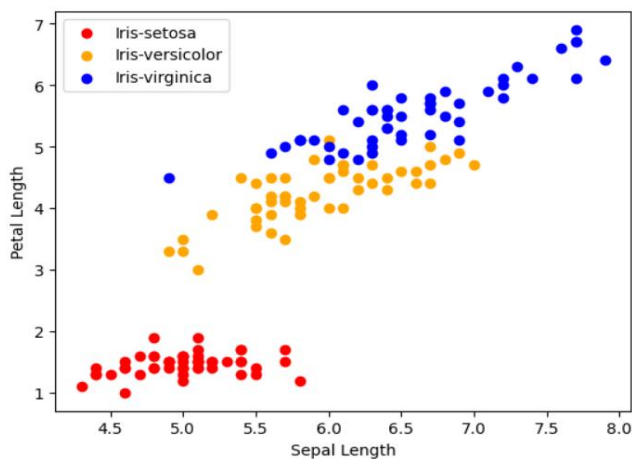
**Sepal Length vs. Sepal Width**

**Petal Length vs. Petal Width**



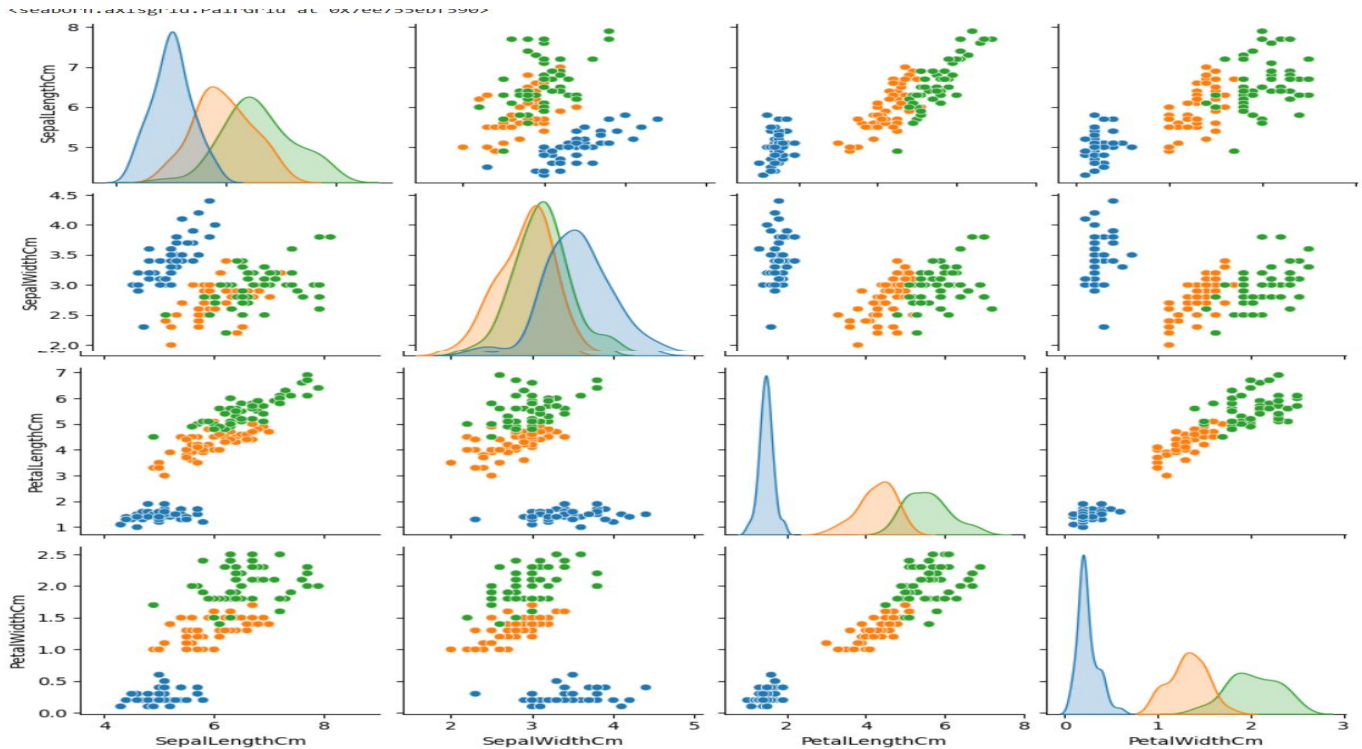
**Sepal Length vs. Petal Length**

**Sepal Width vs. Petal Width**

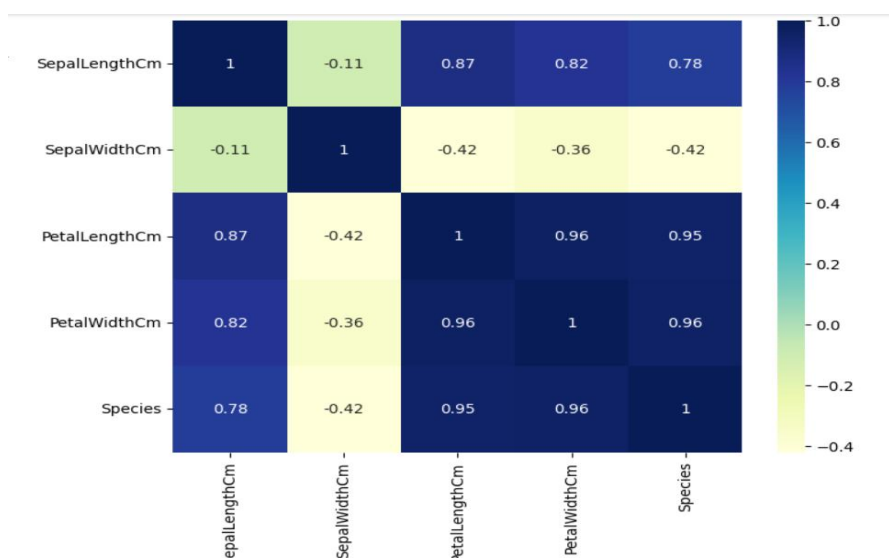


These plots highlighted how different species tend to cluster based on their feature measurements, with Iris-setosa often being distinctly separable from the other two species.

- ❖ Seaborn Scatter Plots: Additional scatter plots using seaborn further emphasized the species separation, particularly for sepal dimensions, providing a more aesthetically pleasing visualization.



- ❖ Correlation Matrix and Heatmap: The correlation matrix was computed using `dfiris.corr(numeric_only=True)` to quantify the linear relationships between numerical features. A heatmap generated using `seaborn.heatmap` visually represented these correlations. It was observed that 'PetalLengthCm' and 'PetalWidthCm' exhibit a strong positive correlation, suggesting their importance in distinguishing between species.



## Data Preprocessing and Model Preparation :

Before model training, the data underwent essential preprocessing:

- Categorical to Numerical: The 'Species' column was converted into numerical format using LabelEncoder.
- Null Value Verification: The dataset was checked for null values to ensure data quality.
- Train-Test Split: The dataset was split into 60% training and 40% testing sets (random\_state=42).
- Feature Scaling: Features were standardized using StandardScaler to ensure uniform variance, crucial for distance-based algorithms like KNN.

## Model Training (K-Nearest Neighbors) :


The K-Nearest Neighbors (KNN) algorithm was chosen for classification:

- Model Initialization: A KNeighborsClassifier model was initialized.
- Model Training: The model was trained on the preprocessed training data (x\_train.values, y\_train.values). The KNN algorithm learns by memorizing the training data, and during prediction, it classifies a new data point based on the majority class of its 'k' nearest neighbors in the training set.

## Prediction and Evaluation :

- Prediction on New Data: A prediction was made for a new, unseen data point [[5, 2.9, 1, 0.2]] to demonstrate the model's ability to classify new samples.

```
[ ] #Prediction
    features = np.array([[5,2.9,1,0.2]])
    prediction = model.predict(features)
    print("Prediction {}".format(prediction))
```

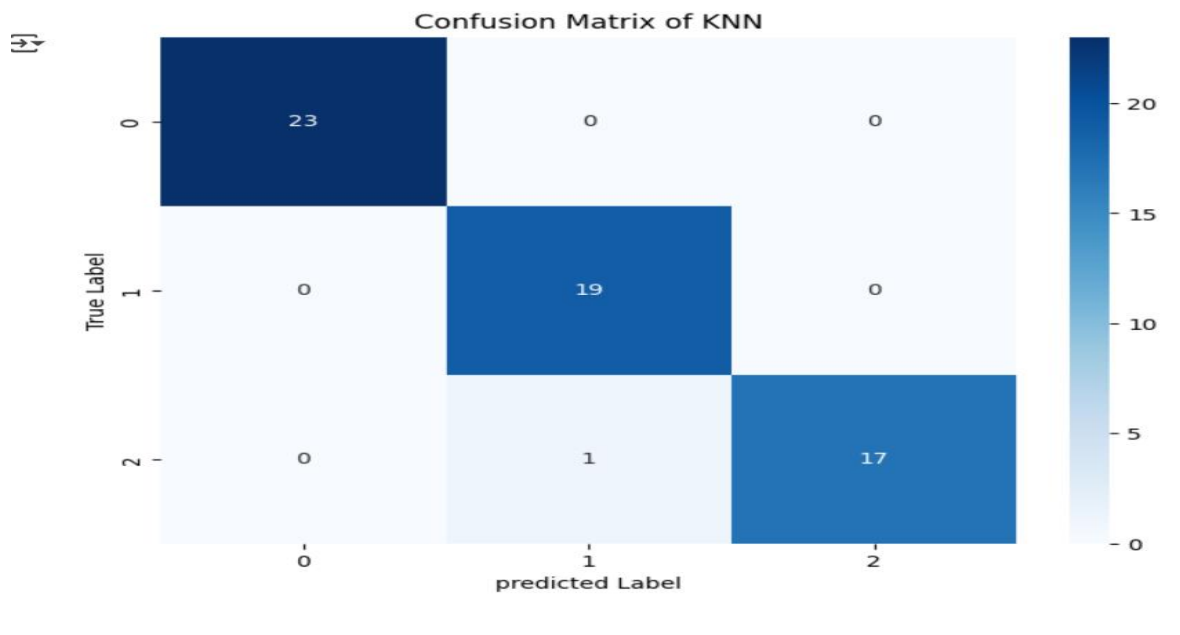
 Prediction [0]

- Model Accuracy: The accuracy of the trained KNN model was evaluated on the test set using model.score(x\_test.values, y\_test.values). The accuracy represents the proportion of correctly classified instances.

```
#Model Training
#K-nearest Neighbours model(KNN)
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(x_train.values,y_train.values)
print("Accuracy (KNN): ", model.score(x_test.values,y_test.values)*100)
```

Accuracy (KNN): 98.33333333333333

- **Confusion Matrix:** A confusion matrix was generated using `confusion_matrix` from `sklearn.metrics`. This matrix provides a detailed breakdown of correct and incorrect predictions for each class. A heatmap visualization of the confusion matrix was created using `seaborn.heatmap`, making it easy to see where the model made errors.



## Conclusion :

The Iris flower classification project successfully demonstrated the application of the K-Nearest Neighbors algorithm. Through comprehensive exploratory data analysis, we gained valuable insights into the feature distributions and inter-species relationships. The preprocessing steps, including label encoding and feature scaling, ensured that the data was in an optimal format for model training. The KNN model achieved a good accuracy, as indicated by its score on the test set and the detailed insights provided by the confusion matrix. This project serves as an excellent example of a complete machine learning workflow, from data loading to model evaluation, on a well-understood dataset.

**By Jalamana Sirisha**