Project Overview

The Iris Classification project involves creating a machine learning model to classify iris flowers into three species (Setosa, Versicolour, and Virginica) based on the length and width of their petals and sepals. This is a classic problem in machine learning and is often used as an introductory example for classification algorithms.

Importing the Necessary Libraries

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

Load the Data

```python
df = pd.read_csv("/content/drive/MyDrive/Unified Mentor/Iris.csv")
```

```python
df.shape
```

```
(150, 5)
```

So our dataset contains 150 rows and five columns.

Let us now find out the column names.

```python
df.columns
```

```
Index(['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm',
       'Species'],
      dtype='object')
```

Let us now find out the number of datapoints in each class.

```python
df["Species"].value_counts()
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

So we can see that this is a balanced dataset because we have 50 datapoints for each class.

```
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 150,\n  \"fields\": [\
n    {\n      \"column\": \"SepalLengthCm\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.828066127977863,\n
\"min\": 4.3,\n        \"max\": 7.9,\n        \"num_unique_values\":
35,\n        \"samples\": [\n          6.2,\n          4.5,\n
5.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SepalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.4335943113621737,\n        \"min\":
2.0,\n        \"max\": 4.4,\n        \"num_unique_values\": 23,\n
\"samples\": [\n          2.3,\n          4.0,\n          3.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"PetalLengthCm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.7644204199522626,\n        \"min\": 1.0,\n        \"max\": 6.9,\n
\"num_unique_values\": 43,\n        \"samples\": [\n          6.7,\n
3.8,\n          3.7\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.7631607417008411,\n        \"min\":
0.1,\n        \"max\": 2.5,\n        \"num_unique_values\": 22,\n
\"samples\": [\n          0.2,\n          1.2,\n          1.3\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Species\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"Iris-
setosa\",\n          \"Iris-versicolor\",\n          \"Iris-
virginica\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

Data Preprocessing

Let us see some high level statistics about the data.

```
df.describe()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"SepalLengthCm\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51.24711349471842,\n
\"min\": 0.828066127977863,\n        \"max\": 150.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
5.843333333333334,\n          5.8,\n          150.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SepalWidthCm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
52.08647211421483,\n        \"min\": 0.4335943113621737,\n
\"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          3.0540000000000003,\n          3.0,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalLengthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 51.835227940958106,\n        \"min\":
1.0,\n        \"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          3.758666666666666,\n          4.35,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 52.636634243409915,\n        \"min\":
0.1,\n        \"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          1.1986666666666668,\n          1.3,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

Let us check the datatypes in the given dataset.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Let us check if the given dataset contains null values. If more than 15% of the enteries in any column are null, we will drop that column. But if the percentage of null values in my column is less than 15%, we will handle the missing values by using the imputation techniques like mean, median and mode.

```
df.isna().sum()

SepalLengthCm    0
SepalWidthCm     0
```

```
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Let us now check if any duplicate enteries exist in the dataset.

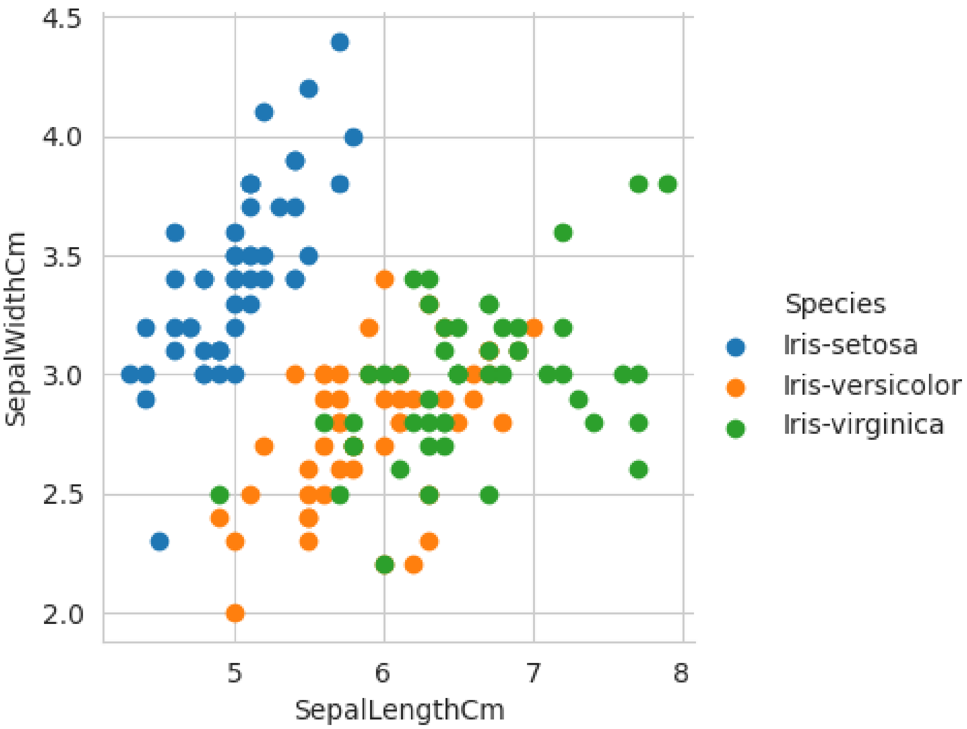
```
df.duplicated().sum()

3
```

Exploratory Data Analysis

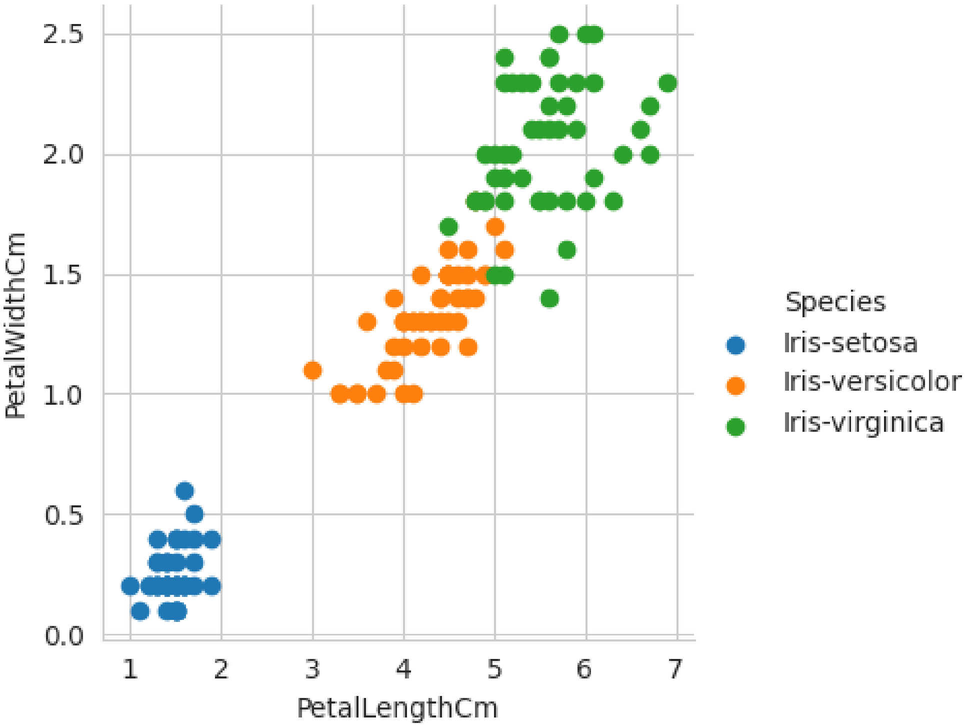
2D Scatter Plot

```
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="Species", height=4) \
   .map(plt.scatter, "SepalLengthCm", "SepalWidthCm") \
   .add_legend();
plt.show();
```



In order to make more sense out of the 2 D scatter plot, we have color coded each class. Note that the setosa datapoints are linearly separable from the versicolor and virginica datapoints. In a similar fashion, we can draw multiple scatter plots for each combination of features. In total, for 4 features we can draw 6 2D scatter plots.

```
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="Species", height=4) \
   .map(plt.scatter, "PetalLengthCm", "PetalWidthCm") \
   .add_legend();
plt.show();
```



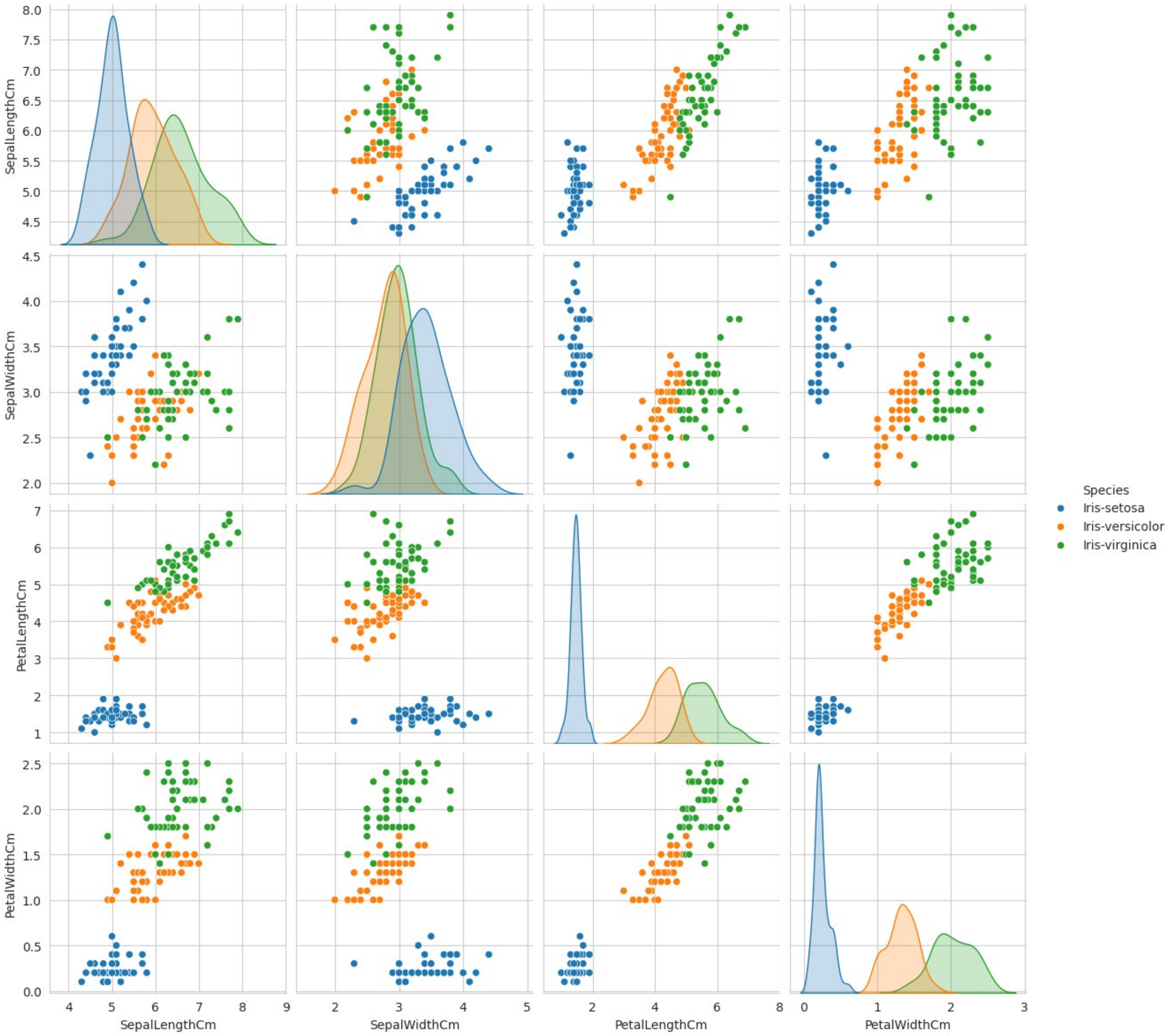
Observations

By using sepal length and sepal width as features, we can easily separate setosa flowers from the other species of iris flowers. It is difficult to separate versicolor flowers from virginica flowers because they overlap considerably.

Pair Plot

```
plt.close();
sns.set_style("whitegrid");
sns.pairplot(df, hue="Species", size=3);
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:2100:
UserWarning: The `size` parameter has been renamed to `height`; please
update your code.
  warnings.warn(msg, UserWarning)
```



Observations

The diagonal enteries of the pair plot are the probability distribution functions of the individual features. We can see that petal length and petal width are the most useful features for distinguishing the different flower types.

```
X = df.drop('Species', axis=1)
y = df['Species']

print(X.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

We will now split the given data 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)
```

Standardize the features using standard scaler. This method calculates the mean and the standard deviation to use later for scaling the data. This method fits the parameters of the data and then transforms it.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Model Training

We will be using K-Nearest Neighbors for classification. K-Nearest Neighbors (KNN) is a machine learning algorithm used for classification and regression tasks. It's a supervised learning method that uses proximity to make predictions or classifications.

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# Make predictions
y_pred = knn.predict(X_test)
```

Model Evaluation

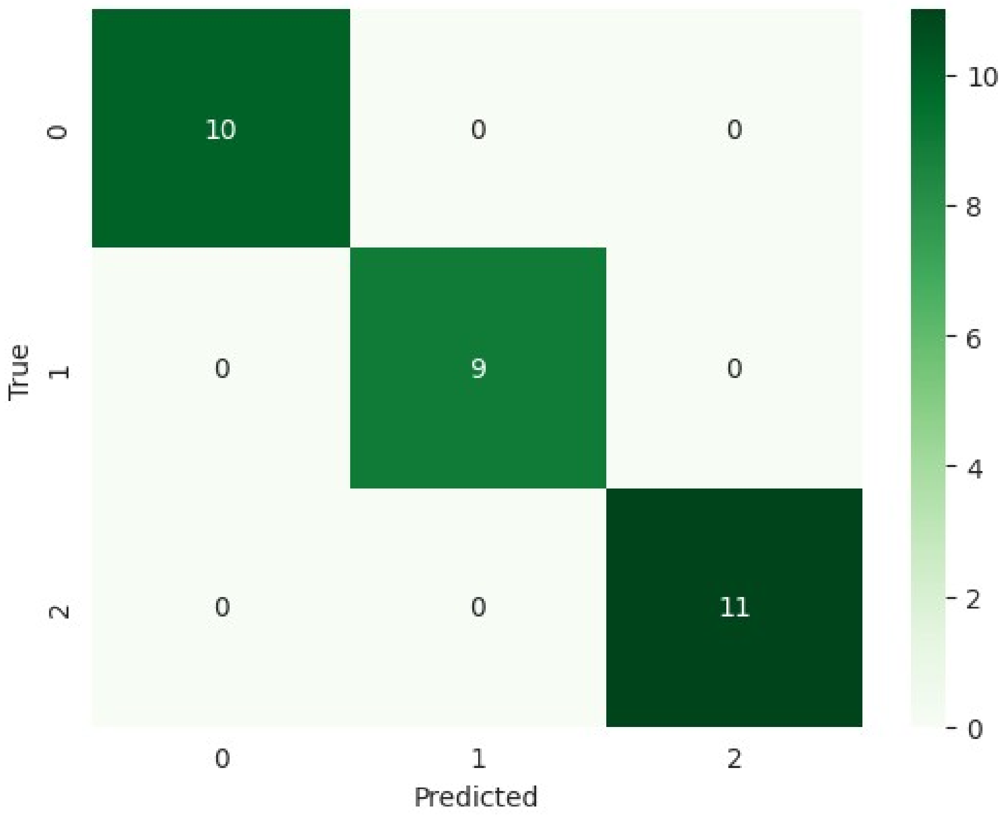
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-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
```

```python
# Plot confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',
cmap='Greens')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



Observations

1) The confusion matrix is a 3x3 matrix because the Iris dataset has three classes (species of Iris flowers): Setosa, Versicolor, and Virginica.

2) The rows of the confusion matrix represent the actual classes (true labels). The columns of the confusion matrix represent the predicted classes (predictions made by the KNN model).

3) The model performed perfectly for Setosa as all the 10 instances were correctly classified.

4) The model performed perfectly for Setosa as all the 9 instances were correctly classified.

5) The model performed perfectly for Setosa as all the 11 instances were correctly classified.

6) The confusion matrix shows that the KNN model performs very well on the iris dataset.