Project Name - Iris Flower Classification

Project Type - Classification

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Project Summary -

Project Description:

The Iris Flower Classification project aims to create a machine learning model capable of predicting the species of iris flowers using their physical measurements. The dataset includes three iris species: Setosa, Versicolor, and Virginica, each having unique measurement patterns that distinguish them from one another.

Objective:

The key objective is to utilize machine learning algorithms to develop an efficient classification model that accurately determines the species of an iris flower based on its measurements. This project provides an automated approach to species identification, making the classification process faster and more reliable.

Key Project Details:

Iris flowers are categorized into three species: Setosa, Versicolor, and Virginica.

These species are differentiated using key attributes such as sepal length, sepal width, petal length, and petal width.

The project focuses on training a machine learning model using a dataset containing these measurements along with their corresponding species labels.

Once trained, the model can predict and classify any given iris flower into one of the three species based on its measurements.

Tech Stack:

Python, NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn (Logistic Regression, Decision Tree, Random Forest, SVM, Naive Bayes, MLP Classifier), XGBoost, LabelEncoder, GridSearchCV, RandomizedSearchCV, Jupyter Notebook/Google Colab

Let's Begin:-

```
In [ ]:
```

```
# Import Libraries
# Importing Numpy & Pandas for data processing & data wrangling
import numpy as np
import pandas as pd

# Importing  tools for visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Import evaluation metric libraries
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_sc

# Library used for data preprocessing
from sklearn.preprocessing import LabelEncoder

# Import model selection libraries
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV,

# Library used for ML Model implementation
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb

# Library used for ignore warnings
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Dataset Loading

In [ ]:

```
# Load Dataset
df = pd.read_csv("/content/Iris.csv")
```

DataSet View and Information

In [ ]:

```
# Dataset First Look
# View top 5 rows of the dataset
df.head()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

```
In [ ]:
```

```python
# Dataset Info
# Checking information about the dataset using info
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
```

```
In [ ]:
```

```python
# Dataset Rows & Columns count
# Checking number of rows and columns of the dataset using shape
print("Number of rows are: ",df.shape[0])
print("Number of columns are: ",df.shape[1])
```

```
Number of rows are:  150
Number of columns are:  5
```

Finding Dublicate and Null Values

```
In [ ]:
```

```python
# Dataset Duplicate Value Count
dup = df.duplicated().sum()
print(f'number of duplicated rows are {dup}')
```

```
number of duplicated rows are 3
```

```
In [ ]:
```

```python
# Missing Values/Null Values Count
df.isnull().sum()
```

```
Out[ ]:
```

	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

Exploring the Variables

```
In [ ]:
```

```
# Dataset Columns
df.columns
```

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

```
In [ ]:
# Dataset Describe (all columns included)
df.describe(include= 'all').round(2)
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.00	150.00	150.00	150.00	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.84	3.05	3.76	1.20	NaN
std	0.83	0.43	1.76	0.76	NaN
min	4.30	2.00	1.00	0.10	NaN
25%	5.10	2.80	1.60	0.30	NaN
50%	5.80	3.00	4.35	1.30	NaN
75%	6.40	3.30	5.10	1.80	NaN
max	7.90	4.40	6.90	2.50	NaN

Check Unique Values for each variable.

```
In [ ]:
# Check Unique Values for each variable.
for i in df.columns.tolist():
  print("No. of unique values in",i,"is",df[i].nunique())
```

```
No. of unique values in SepalLengthCm is 35
No. of unique values in SepalWidthCm is 23
No. of unique values in PetalLengthCm is 43
No. of unique values in PetalWidthCm is 22
No. of unique values in Species is 3
```

Data Wrangling

```
In [ ]:
# We don't need the 1st column so let's drop that
data=df.iloc[:,1:]
```

```
In [ ]:
# New updated dataset
data.head()
```

Out[ ]:

	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	3.5	1.4	0.2	Iris-setosa
1	3.0	1.4	0.2	Iris-setosa
2	3.2	1.3	0.2	Iris-setosa
3	3.1	1.5	0.2	Iris-setosa
4	3.6	1.4	0.2	Iris-setosa

Data Presentation, Narrative Building & Chart Exploration

Chart - 1 : Sepal Length vs Sepal Width

In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm',
hue='Species', data=df, )
# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

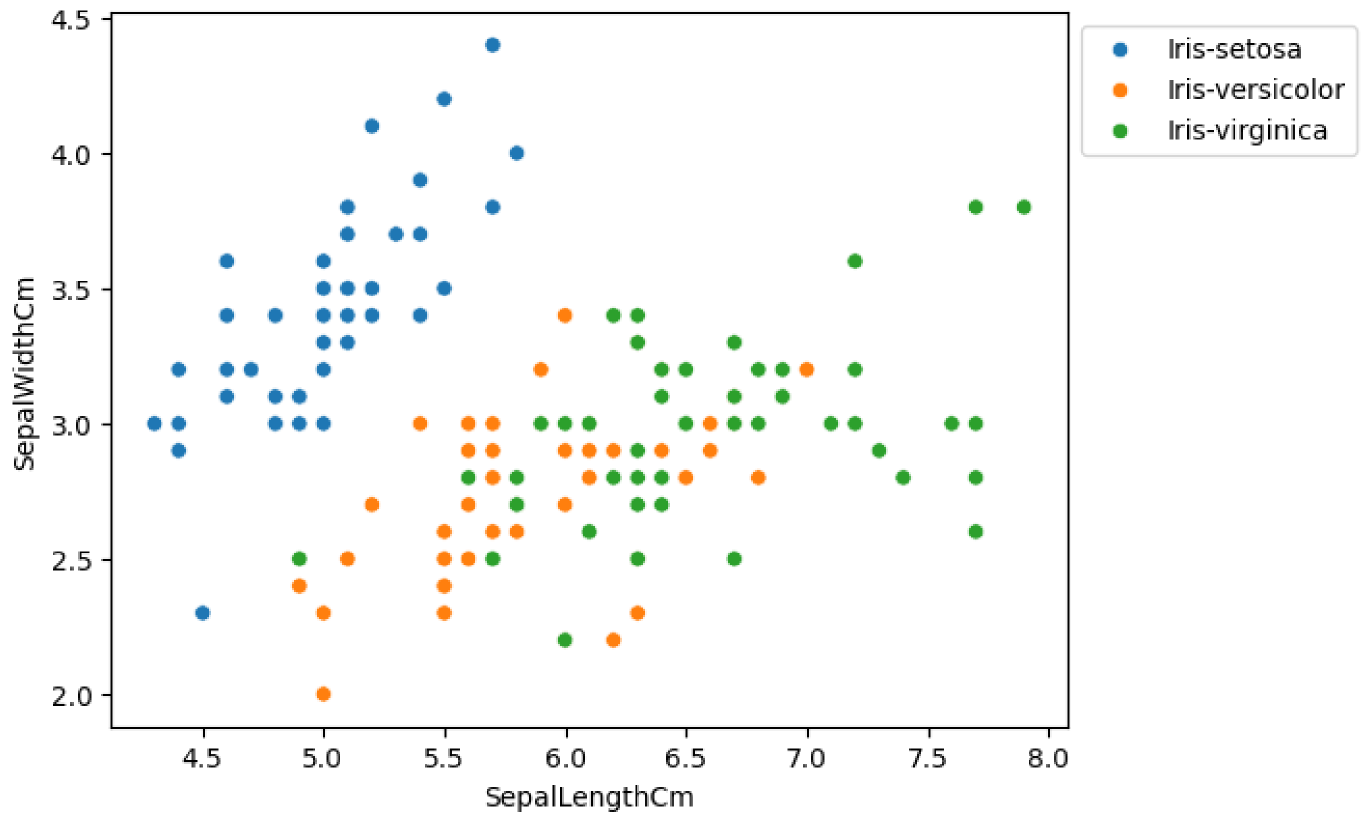


Chart - 2 : Distribution of Numerical Variables

In [ ]:

```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Create a 2x2 grid of subplots with a specified figure size
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

# Plot histogram for Sepal Length
axes[0, 0].set_title("Sepal Length")  # Set the title for the subplot
axes[0, 0].hist(df['SepalLengthCm'], bins=7)  # Plot histogram with 7 bins

# Plot histogram for Sepal Width
axes[0, 1].set_title("Sepal Width")  # Set the title for the subplot
axes[0, 1].hist(df['SepalWidthCm'], bins=5)  # Plot histogram with 5 bins

# Plot histogram for Petal Length
axes[1, 0].set_title("Petal Length")  # Set the title for the subplot
axes[1, 0].hist(df['PetalLengthCm'], bins=6)  # Plot histogram with 6 bins

# Plot histogram for Petal Width
axes[1, 1].set_title("Petal Width")  # Set the title for the subplot
axes[1, 1].hist(df['PetalWidthCm'], bins=6)  # Plot histogram with 6 bins

# Optional: add spacing between plots for better layout
plt.tight_layout()
```

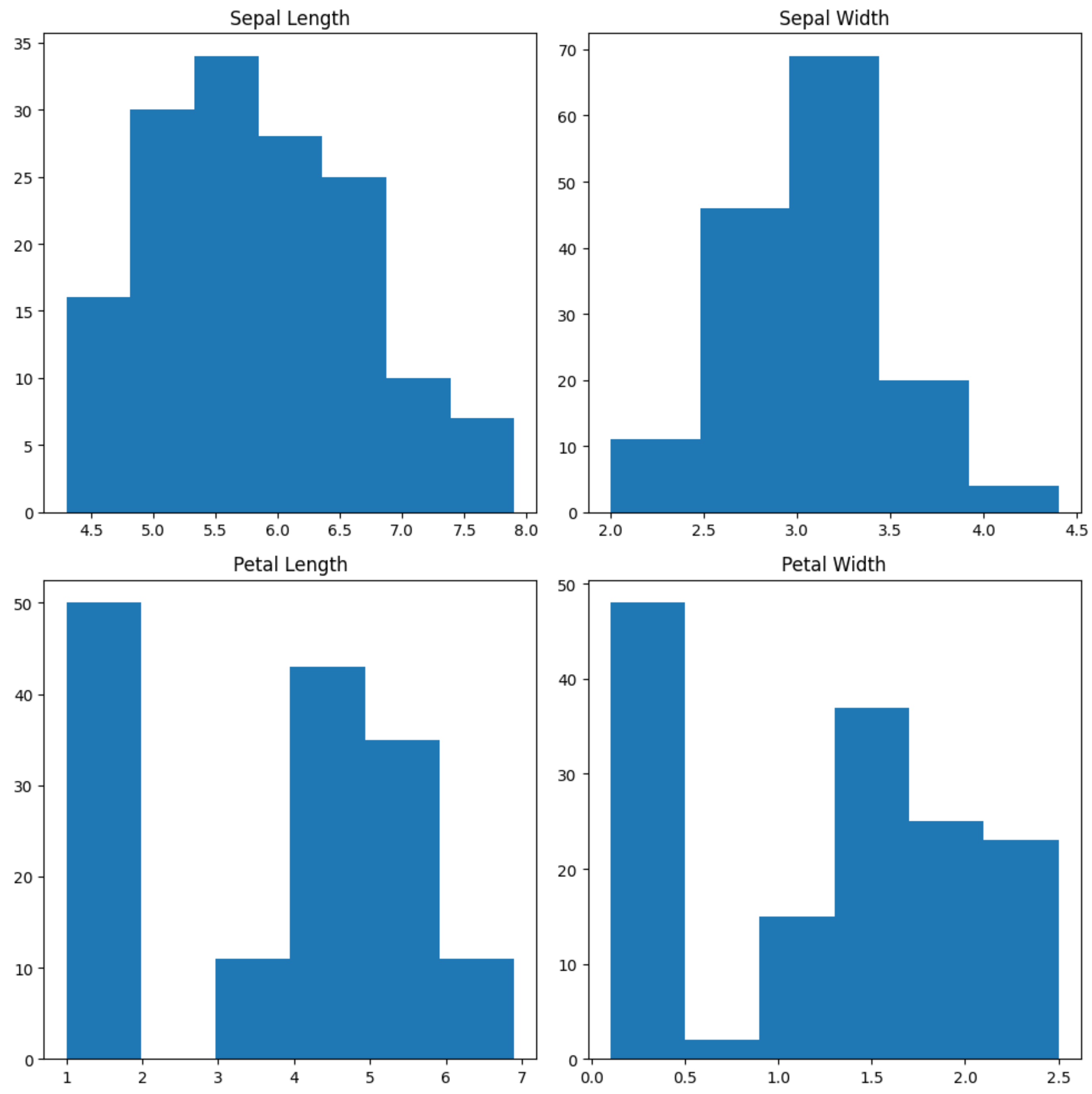


Chart - 3 : Sepal Length vs Petal Length

In [ ]:

```
y=df.iloc[0:100,4].values
y=np.where(y=='Iris-setosa',-1,1)
X=df.iloc[0:100,[0,2]].values
```

In [ ]:

```
plt.scatter(X[:50,0],X[:50,1],color='red',marker='o',
label='setosa')
plt.scatter(X[50:100,0],X[50:100,1],color='blue',marker='x',
label='versicolor')
plt.xlabel('sepallength[cm]')
plt.ylabel('petallength[cm]')
plt.legend(loc='upper left')
plt.show()
```

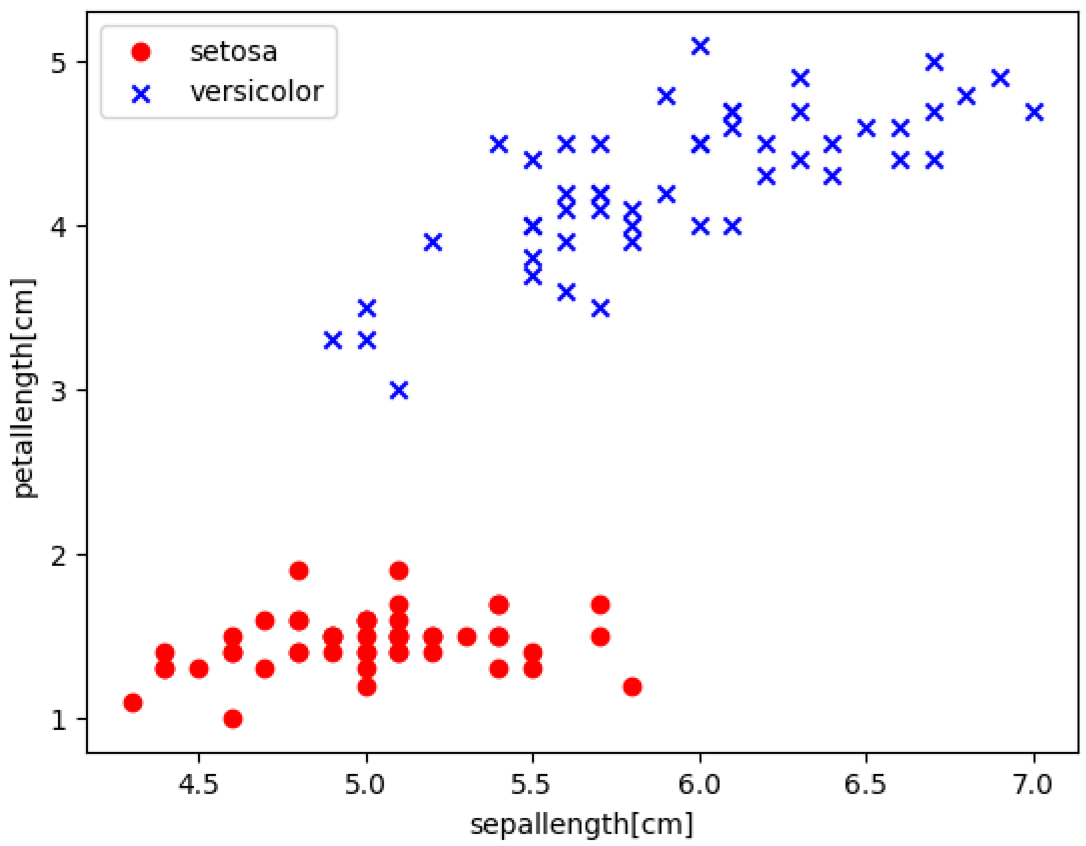


Chart - 4 : Number of Weight Updates per Epoch During Perceptron Training

In [ ]:

```
import numpy as np

class Perceptron(object):
    """
    Perceptron Classifier.

    Parameters
    ----------
    eta : float
        Learning rate (between 0.0 and 1.0)
    n_iter : int
        Passes over the training dataset (epochs)
    random_state : int
        Random number seed for weight initialization.

    Attributes
    ----------
    w_ : 1D array
        Weights after fitting.
    errors_ : list
        Number of misclassifications in every epoch.
    """

    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        self.eta = eta
        self.n_iter = n_iter
        self.random_state = random_state

    def fit(self, X, y):
        """Fit training data.

        Parameters
        ----------
        X : array-like, shape = [n_samples, n_features]
            Training vectors.
        y : array-like, shape = [n_samples]
            Target values.

        Returns
        -------
        self : object
        """
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale=0.01, size=1 + X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[1:] += update * xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self

    def net_input(self, X):
        """Calculate net input (z = w·x + b)"""
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        """Return class label after applying unit step function"""
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

In [ ]:

```
# Create an instance of the Perceptron with learning rate = 0.1 and 10 epochs
ppn = Perceptron(eta=0.1, n_iter=10)

# Fit the model with training data X and y (already defined earlier)
ppn.fit(X, y)
```

Out[ ]:

```
<__main__.Perceptron at 0x7f21690839b0>
```

In [ ]:

```
plt.plot(range(1, len(ppn.errors_) + 1),ppn.errors_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Number of updates')
plt.show()
```

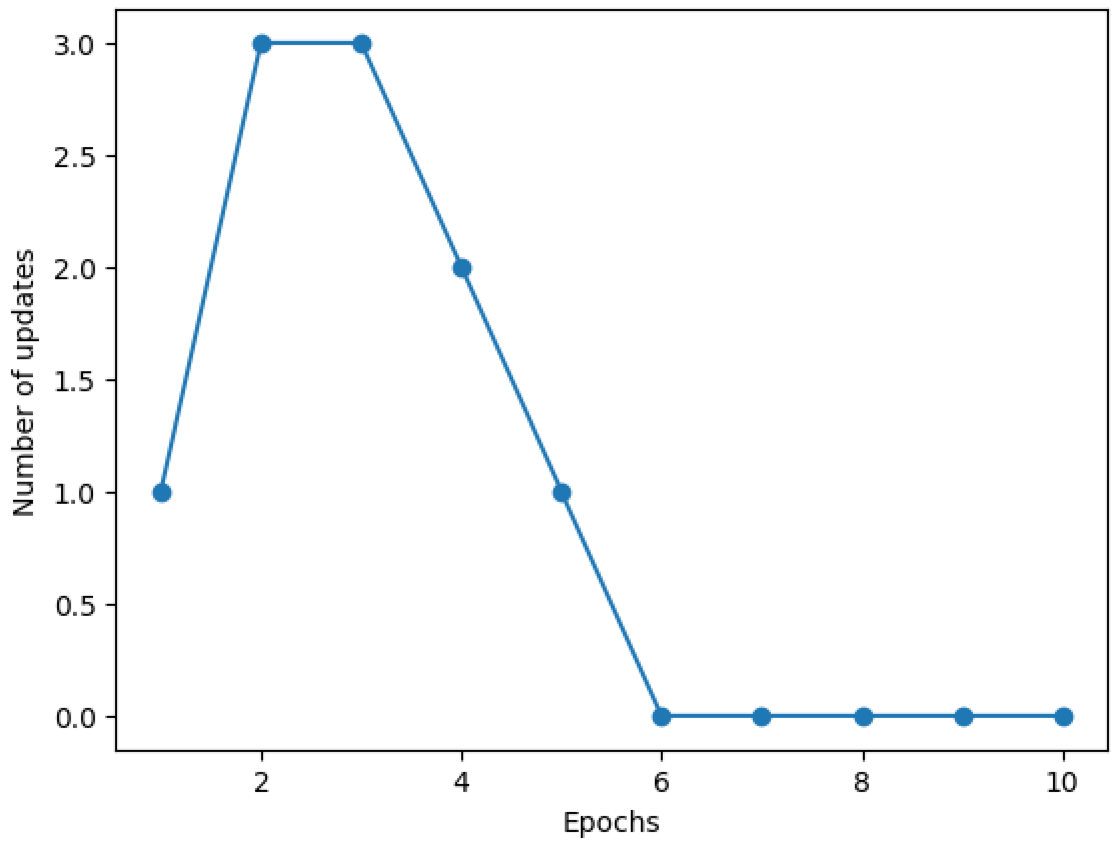


Chart 5: Count of Each Iris Flower Species

In [ ]:

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

# Load the dataset
df = pd.read_csv('Iris.csv')

# Create the count plot with custom colors
sns.countplot(x='Species', data=df, palette=['#1f77b4', '#ff7f0e', '#2ca02c'])  # Blue,

# Add labels and title
plt.xlabel('Species')
plt.ylabel('Count')
plt.title('Count of Each Iris Flower Species')

# Show the plot
plt.show()
```

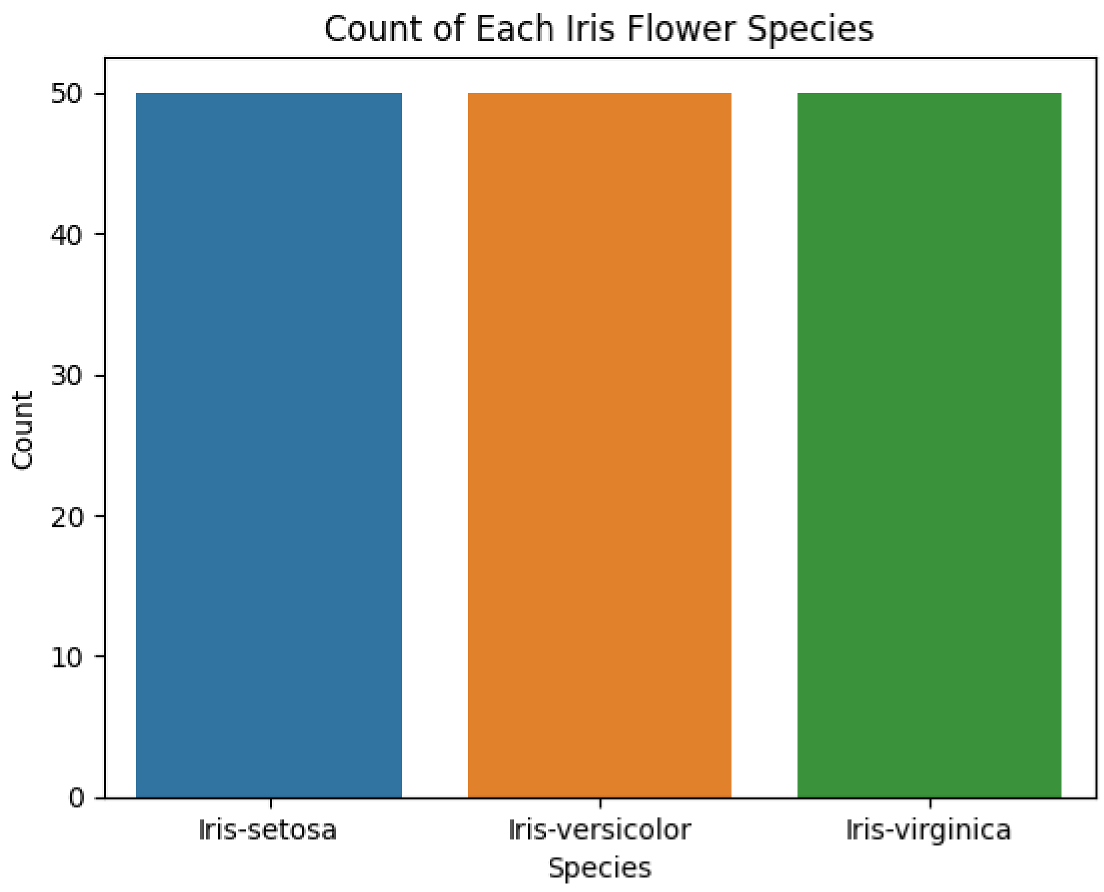


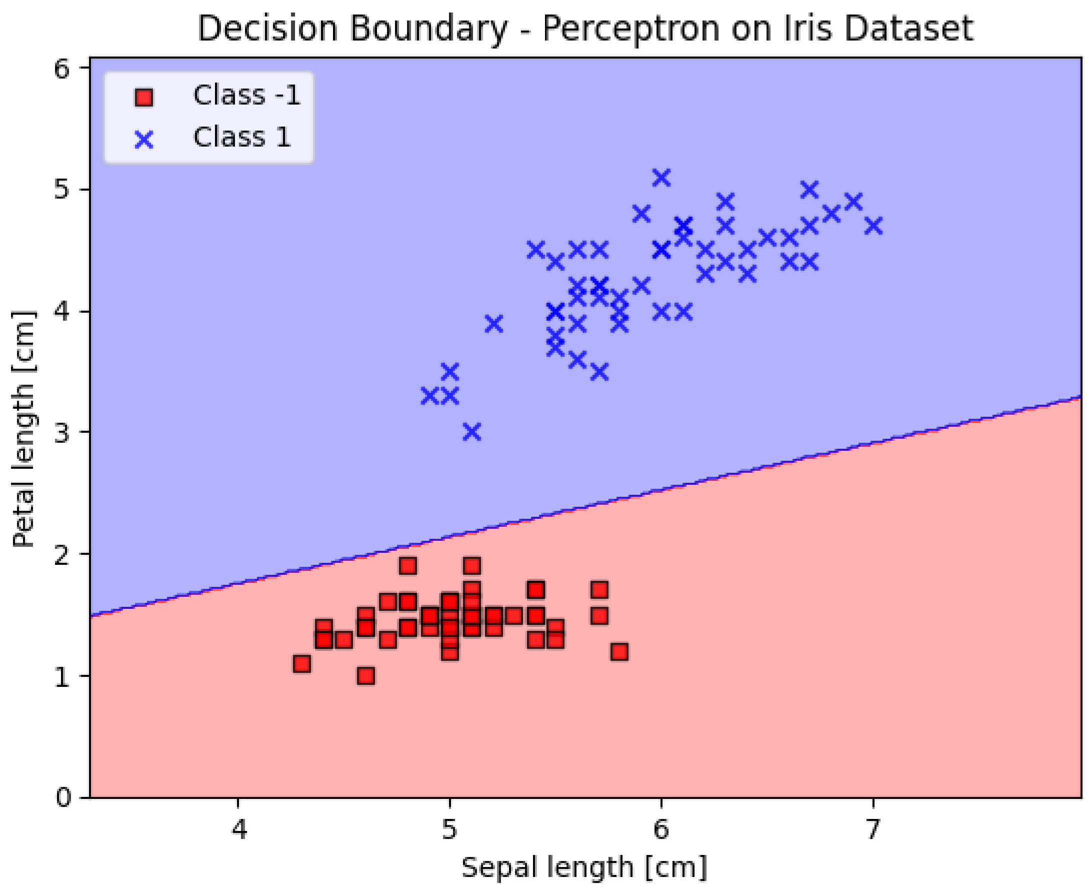
Chart 6 : Perceptron Model: Sepal Length vs. Petal Length Decision Boundary

In [ ]:

```python
from matplotlib.colors import ListedColormap
import numpy as np
import matplotlib.pyplot as plt

def plot_decision_regions(X, y, classifier, resolution=0.02):
    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx1, xx2 = np.meshgrid(
        np.arange(x1_min, x1_max, resolution),
        np.arange(x2_min, x2_max, resolution)
    )

    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)

    # Plot contour
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    plt.show()
```

```
In [ ]:
```

```
# Plot decision regions for the perceptron model
plot_decision_regions(X, y, classifier=ppn)

# Axis labels
plt.xlabel('Sepal length [cm]')
plt.ylabel('Petal length [cm]')

# Add a legend and title
plt.title('Decision Boundary - Perceptron on Iris Dataset')
plt.legend(loc='upper left')

# Show the plot
plt.show()
```



```
In [ ]:
```

```
df.isnull().sum()
```

```
Out[ ]:
```

	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

```
In [ ]:
```

```
data=df.drop_duplicates(subset="Species",)
data
```

```
Out[ ]:
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor
100	6.3	3.3	6.0	2.5	Iris-virginica

```
In [ ]:
```

```
df.value_counts("Species")
```

```
Out[ ]:
```

	count
Species	
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

Chart - 7 : Petal Length vs Petal Width

```
In [ ]:
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='PetalLengthCm', y='PetalWidthCm',
hue='Species', data=df, )
# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

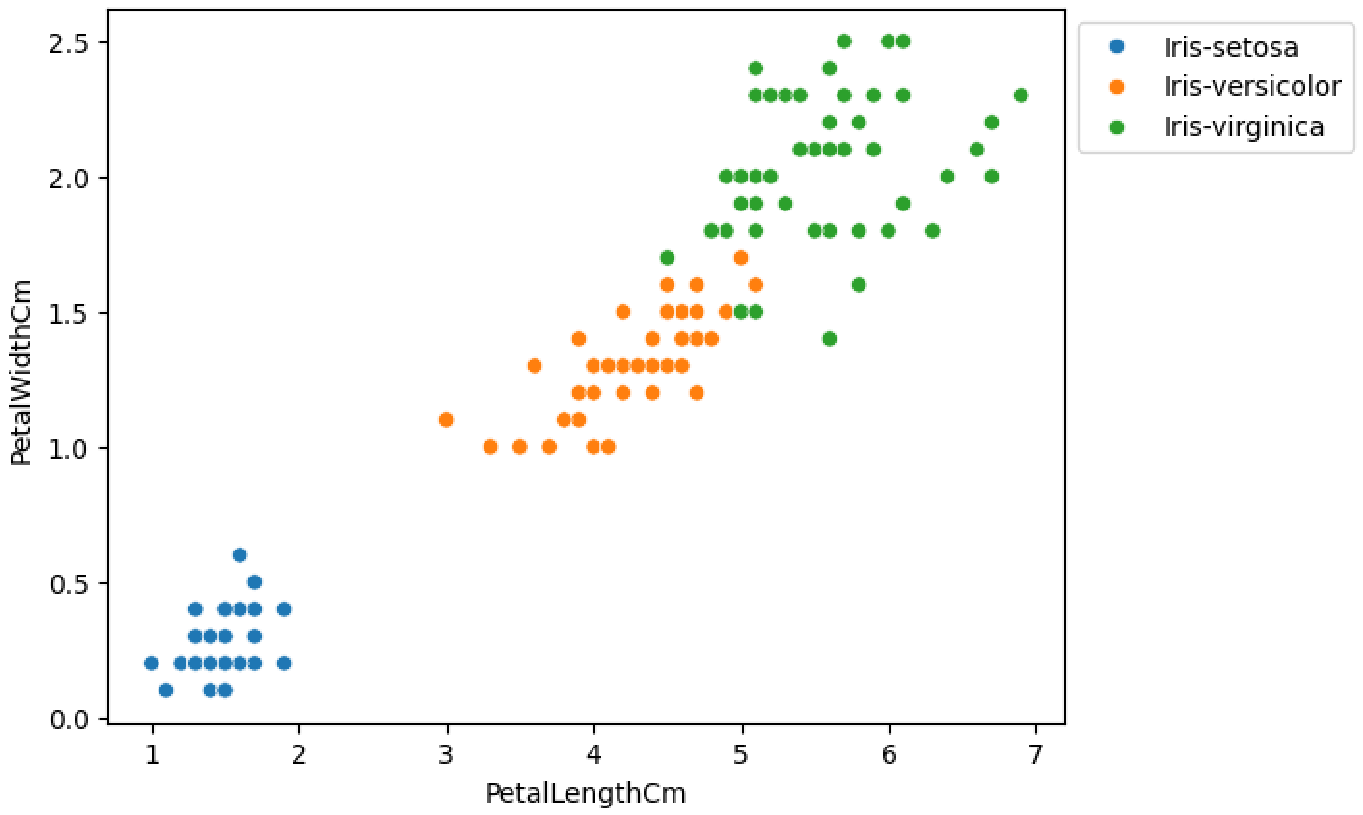


chart-8 : Comparison of Iris Flower Features Across Species

In [ ]:

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

# Use Seaborn style
sns.set(style="whitegrid")

def graph(column, palette):
    """Function to plot a boxplot for a given column with custom palette."""
    sns.boxplot(x="Species", y=column, data=df, palette=palette)

# Define different color palettes for each subplot
palettes = [
    "Blues",      # For Sepal Length
    "Greens",     # For Sepal Width
    "Oranges",    # For Petal Length
    "Purples"     # For Petal Width
]

# Create a larger figure
plt.figure(figsize=(12, 8))

# Plot Sepal Length
plt.subplot(221)
graph('SepalLengthCm', palettes[0])
plt.title("Sepal Length by Species")

# Plot Sepal Width
plt.subplot(222)
graph('SepalWidthCm', palettes[1])
plt.title("Sepal Width by Species")
```

```
# Plot Petal Length
plt.subplot(223)
graph('PetalLengthCm', palettes[2])
plt.title("Petal Length by Species")

# Plot Petal Width
plt.subplot(224)
graph('PetalWidthCm', palettes[3])
plt.title("Petal Width by Species")

# Adjust spacing between subplots
plt.tight_layout()
plt.show()
```

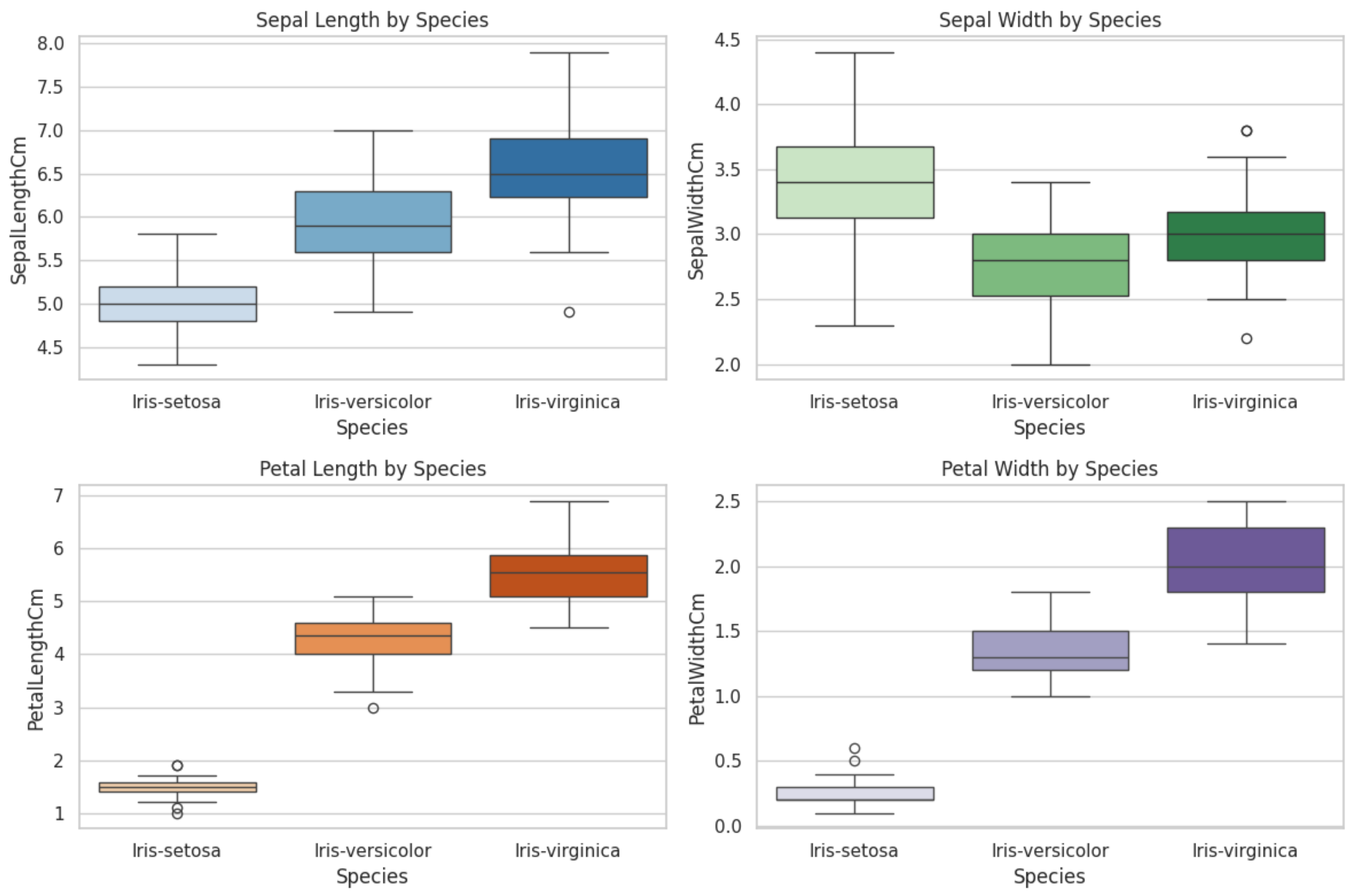
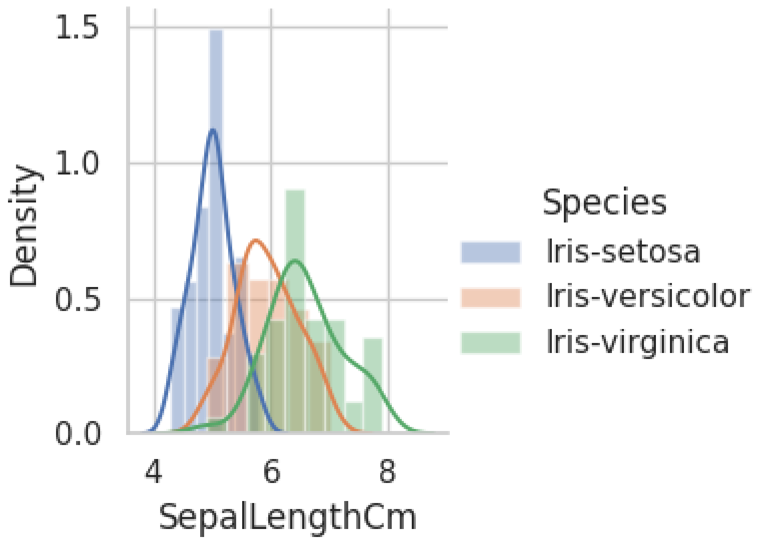


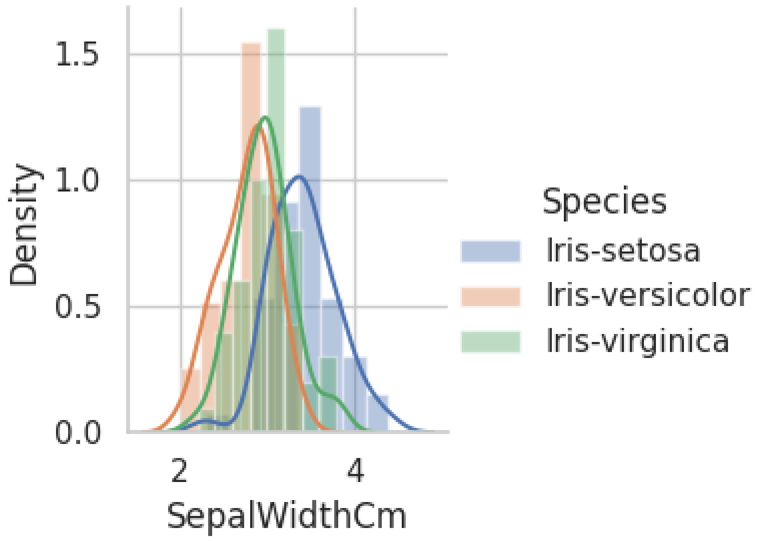
chart-9 : Distribution of Iris Flower Features by Species

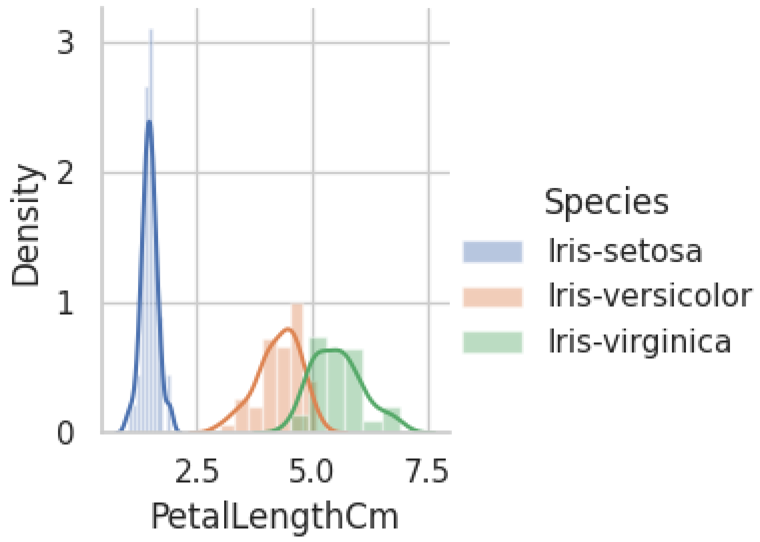
In [ ]:

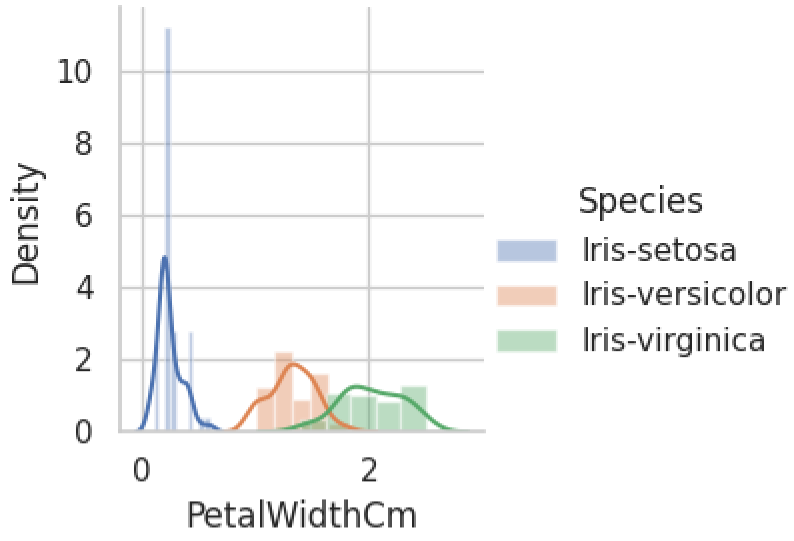
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting distribution of Sepal Length for each species
plot = sns.FacetGrid(df, hue="Species")   # Create FacetGrid with hue as species
plot.map(sns.distplot, "SepalLengthCm").add_legend()  # Map distribution plot and add le

# Plotting distribution of Sepal Width for each species
plot = sns.FacetGrid(df, hue="Species")   # Create FacetGrid again for Sepal Width
plot.map(sns.distplot, "SepalWidthCm").add_legend()  # Map distribution plot and add leg

# Plotting distribution of Petal Length for each species
plot = sns.FacetGrid(df, hue="Species")   # Create FacetGrid again for Petal Length
```

```
plot.map(sns.distplot, "PetalLengthCm").add_legend()  # Map distribution plot and add le

# Plotting distribution of Petal Width for each species
plot = sns.FacetGrid(df, hue="Species")   # Create FacetGrid again for Petal Width
plot.map(sns.distplot, "PetalWidthCm").add_legend()  # Map distribution plot and add leg

# Display all the plots
plt.show()
```









In [ ]:

```
# Select only numeric columns and then calculate correlation
numeric_data = data.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numeric_data.corr(method='pearson')
correlation_matrix
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.999226	0.795795	0.643817
SepalWidthCm	-0.999226	1.000000	-0.818999	-0.673417
PetalLengthCm	0.795795	-0.818999	1.000000	0.975713
PetalWidthCm	0.643817	-0.673417	0.975713	1.000000

chart 10: Boxplot of Sepal Width in Iris Dataset

In [ ]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
# Load the dataset
```

```
df = pd.read_csv('/content/Iris.csv')
sns.boxplot(x='SepalWidthCm', data=df)
```

Out[ ]:

```
<Axes: xlabel='SepalWidthCm'>
```

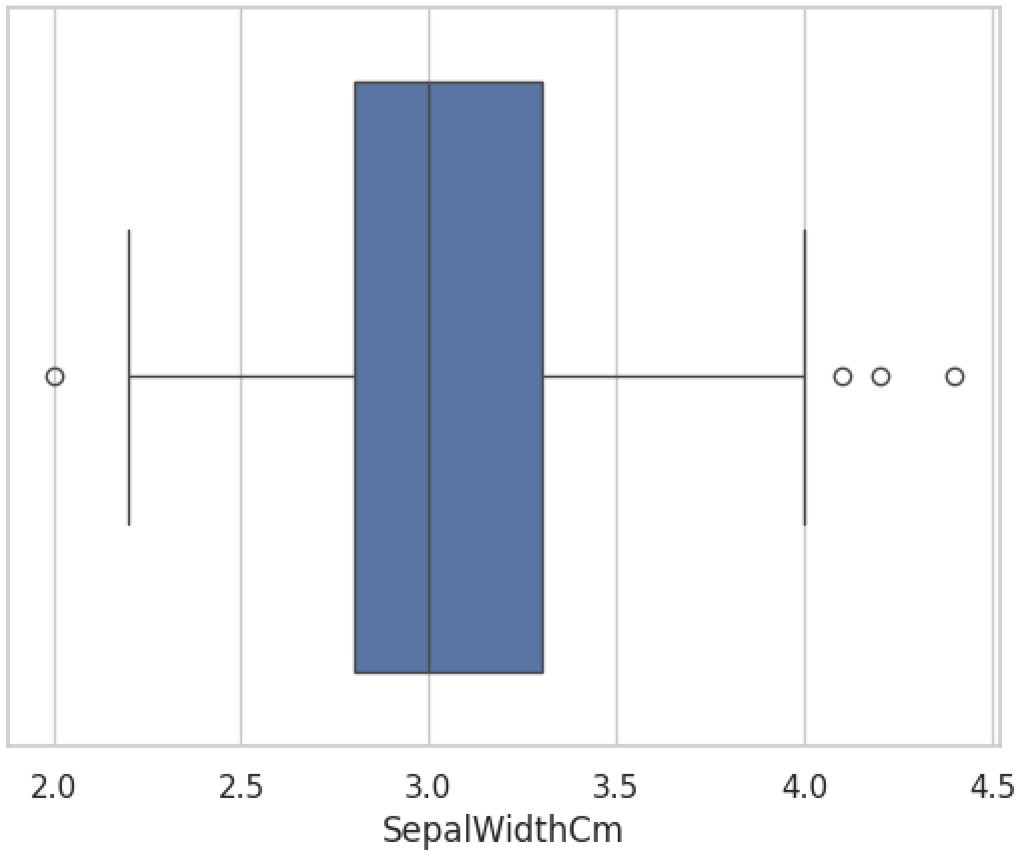


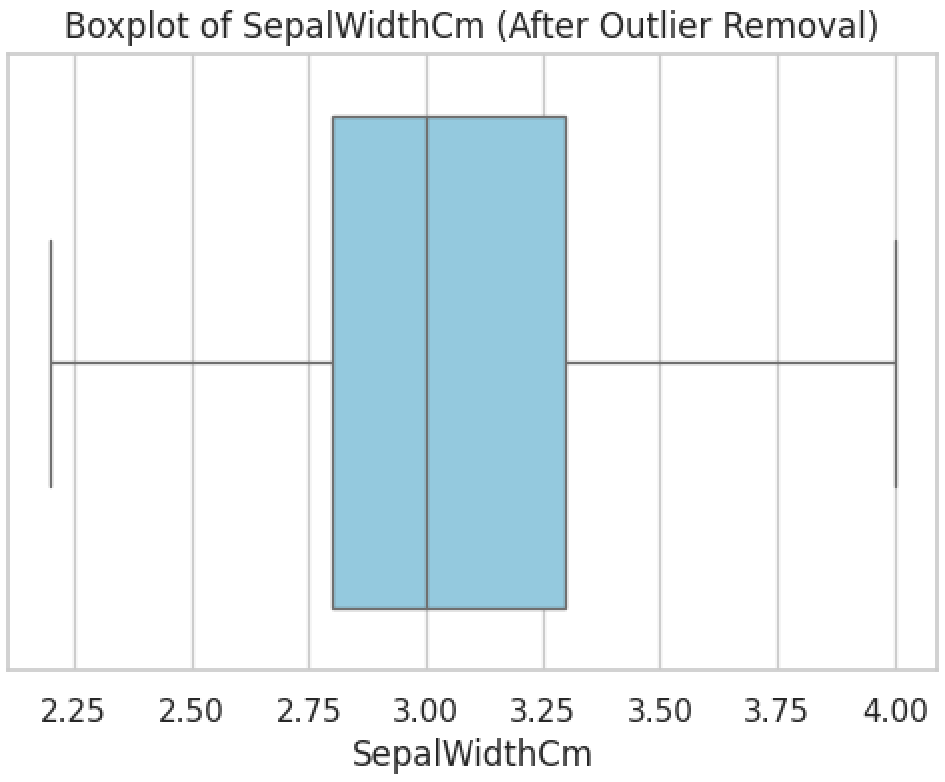
Chart - 11 Sepal Width Distribution After Outlier Removal

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('/content/Iris.csv')

# IQR for SepalWidthCm
Q1 = np.percentile(df['SepalWidthCm'], 25, interpolation='midpoint')
Q3 = np.percentile(df['SepalWidthCm'], 75, interpolation='midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper and lower bounds for outliers
upper = np.where(df['SepalWidthCm'] >= (Q3 + 1.5 * IQR))
lower = np.where(df['SepalWidthCm'] <= (Q1 - 1.5 * IQR))

# Removing the Outliers
df.drop(upper[0], inplace=True)
df.drop(lower[0], inplace=True)

print("New Shape: ", df.shape)
```

```python
# Plot boxplot with custom color palette
plt.figure(figsize=(6, 4))
sns.boxplot(x='SepalWidthCm', data=df, color='skyblue')   # Single color
# Alternative: palette for multiple colors
# sns.boxplot(x='SepalWidthCm', data=df, palette="Blues")

plt.title("Boxplot of SepalWidthCm (After Outlier Removal)")
plt.show()
```

```
Old Shape:  (150, 5)
New Shape:  (146, 5)
```


Boxplot of SepalWidthCm (After Outlier Removal)
SepalWidthCm