Ramesh Chandra Soren

Enrollment No.: 2022CSB086
Department: Computer Science and Technology

The objective of this task is to classify the Iris flowers into three distinct species: Iris-setosa, Iris-versicolor, and Iris-virginica. This classification will be based on the morphological measurements of the flowers, specifically the lengths and widths of their petals and sepals.

Cell 1: Install and import libraries

> Add blockquote

```
# pip install pandas numpy scikit-learn matplotlib seaborn tensorflow

import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

Cell 2:

A . Data Acquisition: The Iris dataset, which is publicly available, can be downloaded from the UCI Machine Learning Repository. The dataset consists of 150 samples with 4 features each.

```
iris = load_iris()
X = iris.data
y = iris.target
```

B. Data Preparation: Prepare the dataset by loading it into a suitable data structure, such as a Pandas DataFrame.

```
df = pd.read_csv('/content/iris.data',
names=['sl','sw','pl','pw','class'])

df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 150,\n  \"fields\": [\
n    {\n      \"column\": \"sl\",\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\":
\"sw\",\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\": \"pl\",\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\":
\"pw\",\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\": \"class\",\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"}
```


Cell 3: Split the data

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

print(X)
print(y)
```

```
      sl   sw   pl   pw
0    5.1  3.5  1.4  0.2
1    4.9  3.0  1.4  0.2
2    4.7  3.2  1.3  0.2
3    4.6  3.1  1.5  0.2
4    5.0  3.6  1.4  0.2
..   ...  ...  ...  ...
145  6.7  3.0  5.2  2.3
146  6.3  2.5  5.0  1.9
147  6.5  3.0  5.2  2.0
148  6.2  3.4  5.4  2.3
149  5.9  3.0  5.1  1.8
```

```
[150 rows x 4 columns]
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: class, Length: 150, dtype: object
```

C. Feature Selection: Use the provided features (sepal length, sepal width, petal length, petal width) to build a classification model.

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

Cell 4: Train the model

D. Model Training: Develop a classification model to categorize the Iris flowers into the three species.

```
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)

SVC(kernel='linear', random_state=42)
```

Cell 5: Make predictions

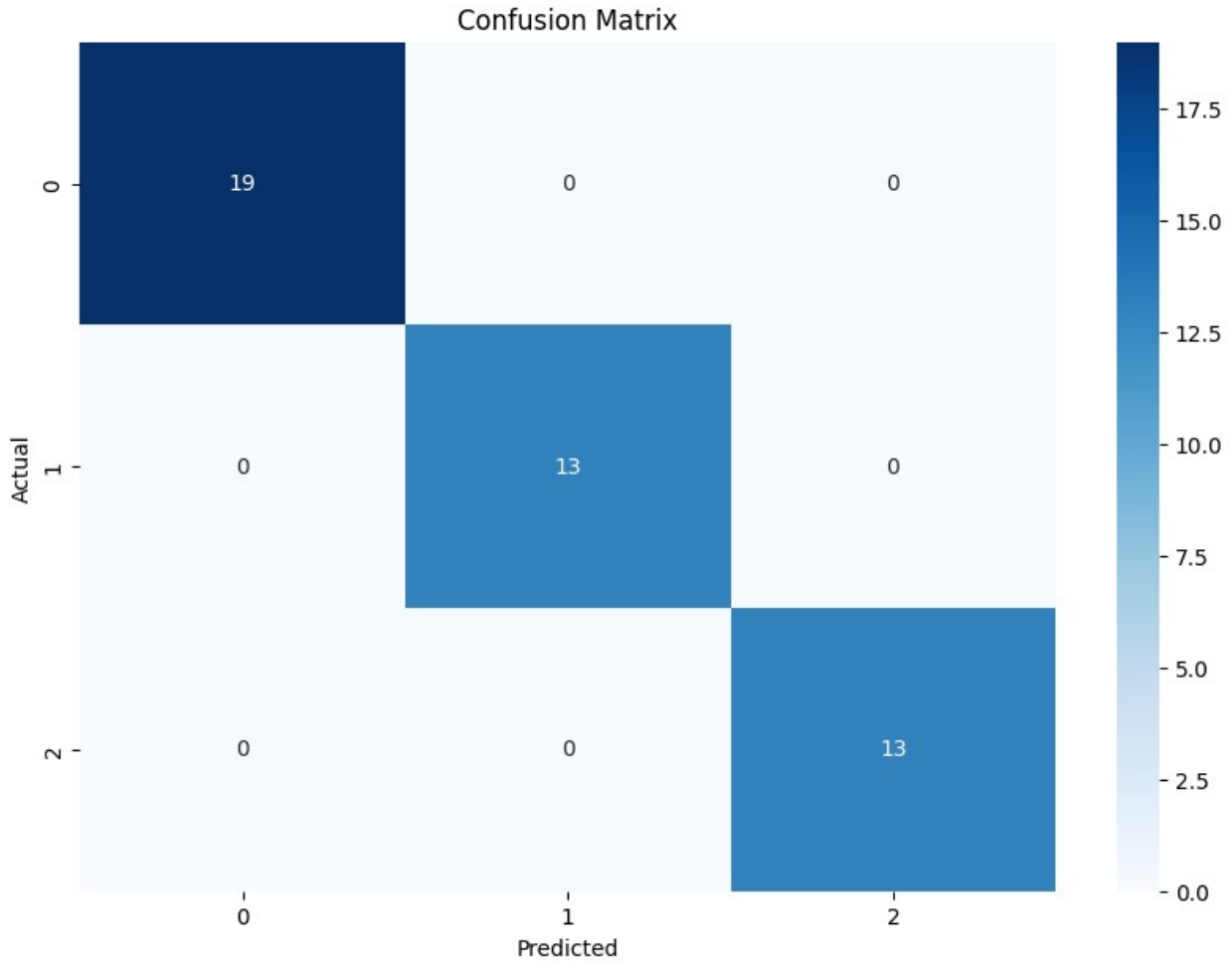
```
y_pred = svm.predict(X_test)
```

Cell 6: Create and plot confusion matrix

E. Model Evaluation: Evaluate the model's performance using appropriate metrics. Present the results using a confusion matrix.

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

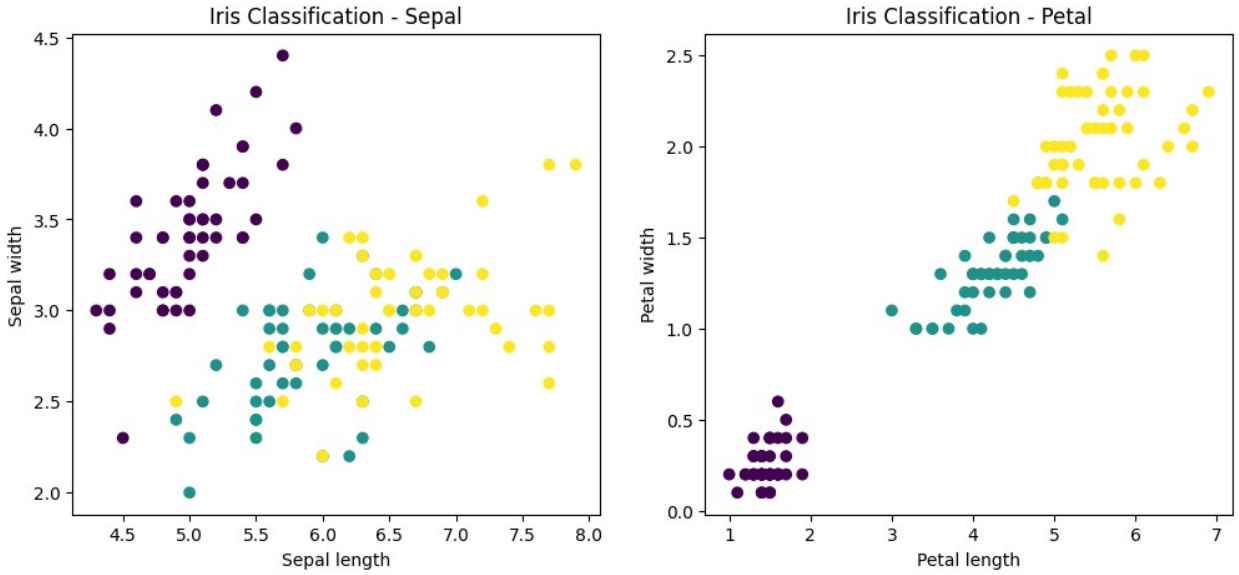


Cell 7: Plot the results

F. Visualization: Plot the classification results to visually assess the performance of the model.

```
plt.figure(figsize=(12,5))
plt.subplot(121)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('Iris Classification - Sepal')

plt.subplot(122)
```

```
plt.scatter(X[:, 2], X[:, 3], c=y, cmap='viridis')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.title('Iris Classification - Petal')

plt.show()
```



Cell 8: Print classification report

```
print(classification_report(y_test, y_pred,
target_names=iris.target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       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
```

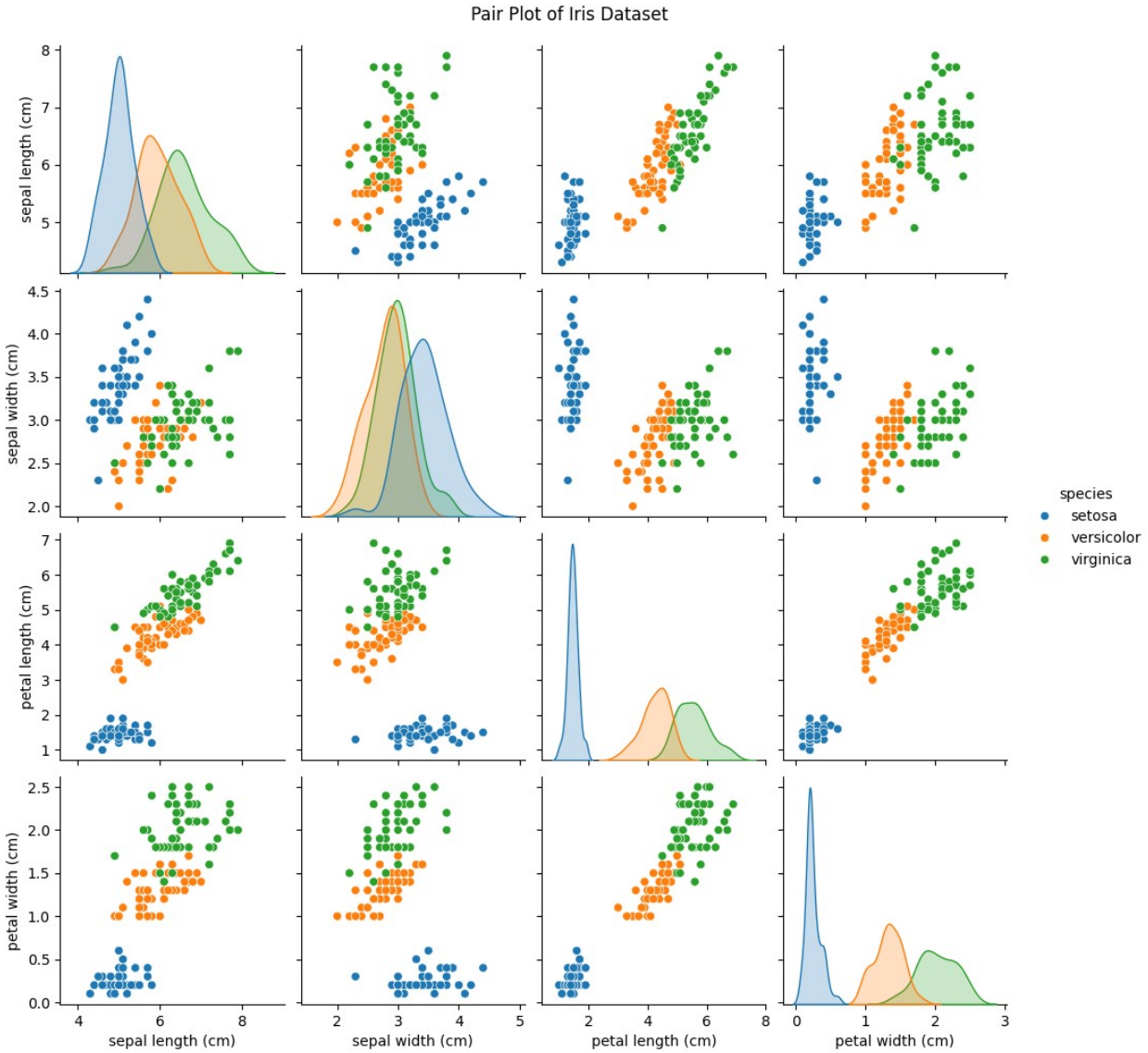
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris

# Load the Iris dataset
```

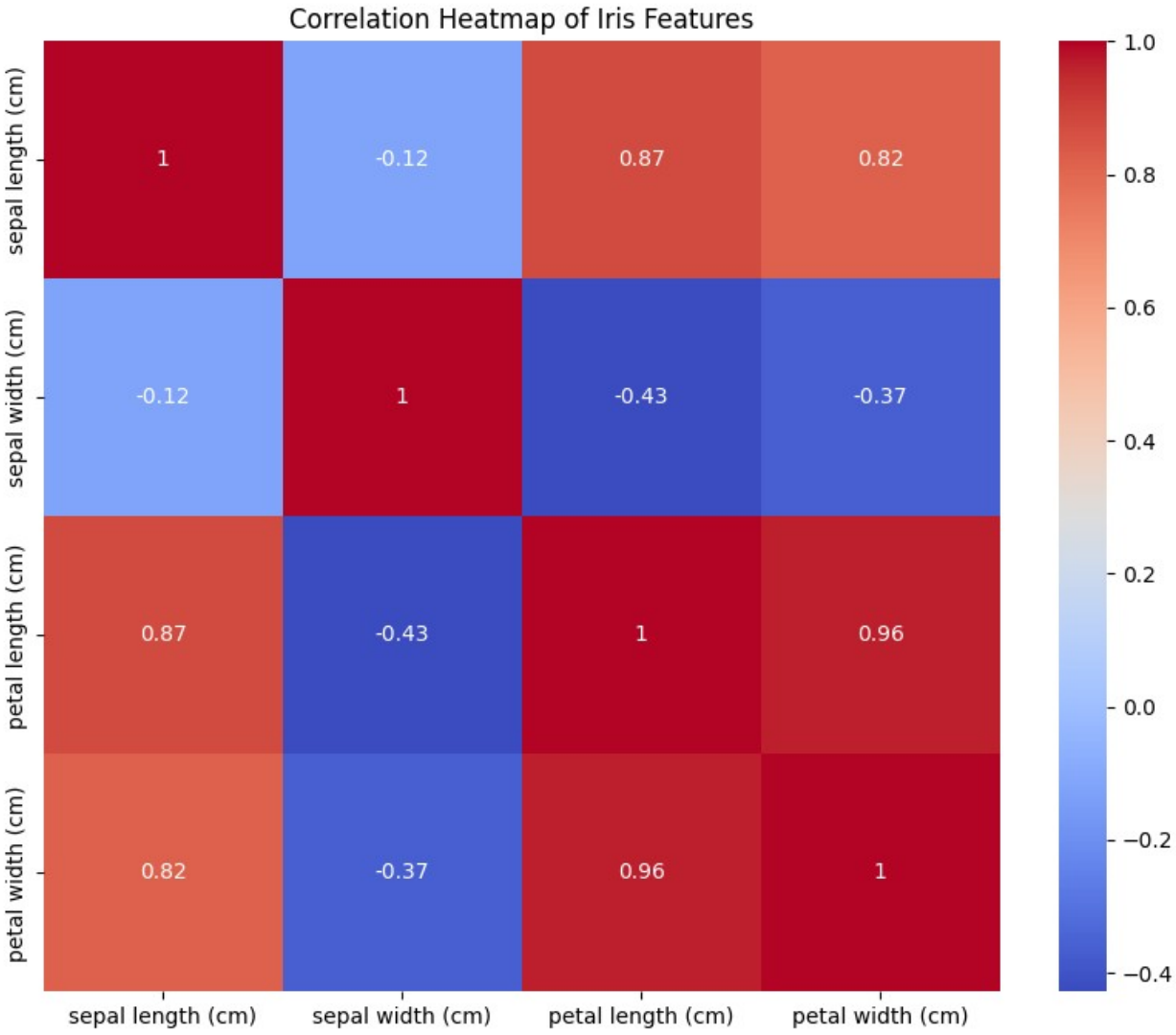
```
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

# Convert to DataFrame for easier handling
df = pd.DataFrame(X, columns=feature_names)
df['species'] = pd.Categorical.from_codes(y, target_names)

# 1. Pair Plot
plt.figure(figsize=(12, 10))
sns.pairplot(df, hue='species', height=2.5)
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()

# 2. Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df.drop('species', axis=1).corr(), annot=True,
cmap='coolwarm')
plt.title('Correlation Heatmap of Iris Features')
plt.show()

# 3. Box Plots
plt.figure(figsize=(12, 8))
for i, feature in enumerate(feature_names):
    plt.subplot(2, 2, i+1)
    sns.boxplot(x='species', y=feature, data=df)
    plt.title(f'Box Plot of {feature}')
plt.tight_layout()
plt.show()

# 4. Violin Plots
plt.figure(figsize=(12, 8))
for i, feature in enumerate(feature_names):
    plt.subplot(2, 2, i+1)
    sns.violinplot(x='species', y=feature, data=df)
    plt.title(f'Violin Plot of {feature}')
plt.tight_layout()
plt.show()

# 5. Histogram
plt.figure(figsize=(12, 8))
for i, feature in enumerate(feature_names):
    plt.subplot(2, 2, i+1)
    for species in target_names:
        sns.histplot(df[df['species'] == species][feature], kde=True,
label=species)
    plt.title(f'Histogram of {feature}')
    plt.legend()
```

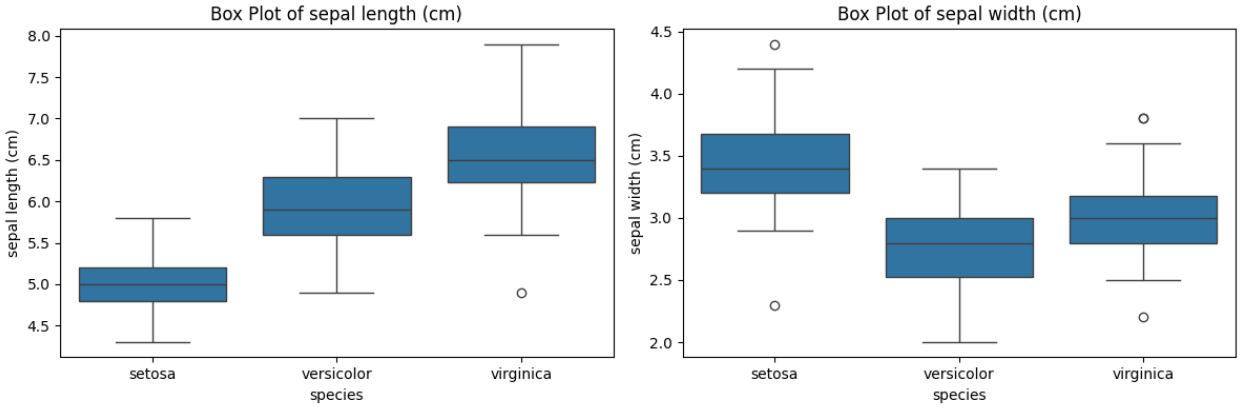
```
plt.tight_layout()
plt.show()

# 6. 3D Scatter Plot
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

colors = ['r', 'g', 'b']
for i, species in enumerate(target_names):
    species_data = df[df['species'] == species]
    ax.scatter(species_data['sepal length (cm)'],
               species_data['sepal width (cm)'],
               species_data['petal length (cm)'],
               c=colors[i], label=species)

ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Sepal Width (cm)')
ax.set_zlabel('Petal Length (cm)')
ax.legend()
plt.title('3D Scatter Plot of Iris Dataset')
plt.show()

# 7. Andrews Curves
from pandas.plotting import andrews_curves

plt.figure(figsize=(12, 6))
andrews_curves(df, 'species')
plt.title('Andrews Curves of Iris Dataset')
plt.show()

# 8. Parallel Coordinates
from pandas.plotting import parallel_coordinates

plt.figure(figsize=(12, 6))
parallel_coordinates(df, 'species')
plt.title('Parallel Coordinates of Iris Dataset')
plt.show()

# 9. Radar Chart
import math # import the math module

def make_spider(df, title, color):
    # Calculate mean values for each feature for the given species
    mean_values = df.drop('species', axis=1).mean()

    # Number of variables
    categories = list(mean_values.index)
    N = len(categories)
```

```
    # What will be the angle of each axis in the plot? (we divide the
plot / number of variable)
    angles = [n / float(N) * 2 * math.pi for n in range(N)] #
reference pi from the math module
    angles += angles[:1]

    # Initialise the spider plot
    ax = plt.subplot(111, polar=True)

    # If you want the first axis to be on top:
    ax.set_theta_offset(math.pi / 2) # reference pi from the math
module
    ax.set_theta_direction(-1)

    # Draw one axe per variable + add labels
    plt.xticks(angles[:-1], categories)

    # Draw ylabels
    ax.set_rlabel_position(0)
    plt.yticks([1,2,3,4,5,6,7], ["1","2","3","4","5","6","7"],
color="grey", size=7)
    plt.ylim(0,7)

    # Plot data
    values = mean_values.values.flatten().tolist()
    values += values[:1]
    ax.plot(angles, values, color=color, linewidth=2,
linestyle='solid')
    ax.fill(angles, values, color=color, alpha=0.4)

    # Add a title
    plt.title(title, size=11, color=color, y=1.1)

# Create radar charts
plt.figure(figsize=(15, 5))

plt.subplot(131, polar=True)
make_spider(df[df['species'] == 'setosa'], 'Setosa', 'blue')

plt.subplot(132, polar=True)
make_spider(df[df['species'] == 'versicolor'], 'Versicolor', 'red')

plt.subplot(133, polar=True)
make_spider(df[df['species'] == 'virginica'], 'Virginica', 'green')

plt.tight_layout()
plt.show()
```

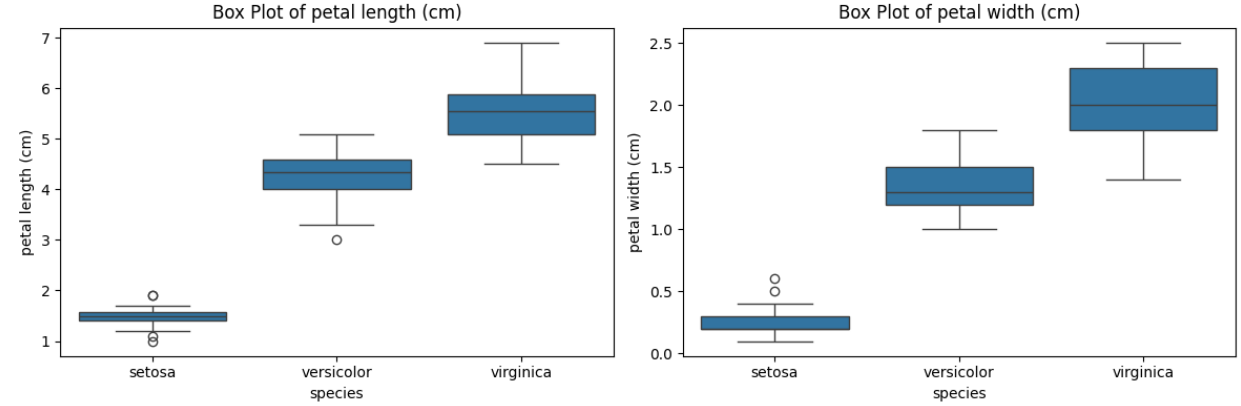
```
<Figure size 1200x1000 with 0 Axes>
```


Pair Plot of Iris Dataset
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
species
setosa
versicolor
virginica

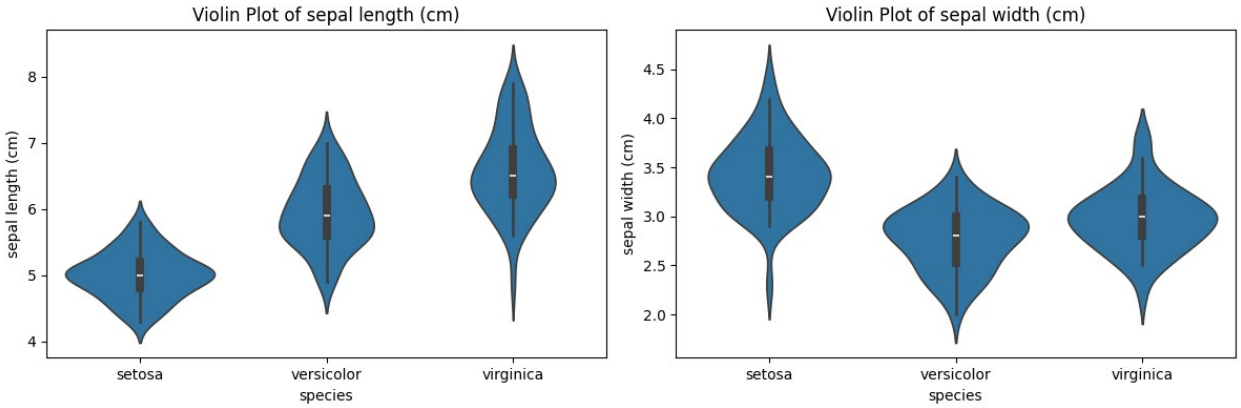

Correlation Heatmap of Iris Features
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
1
-0.12
0.87
0.82
-0.12
1
-0.43
-0.37
0.87
-0.43
1
0.96
0.82
-0.37
0.96
1
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Box Plot of sepal length (cm)
sepal length (cm)
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
setosa
versicolor
virginica
species

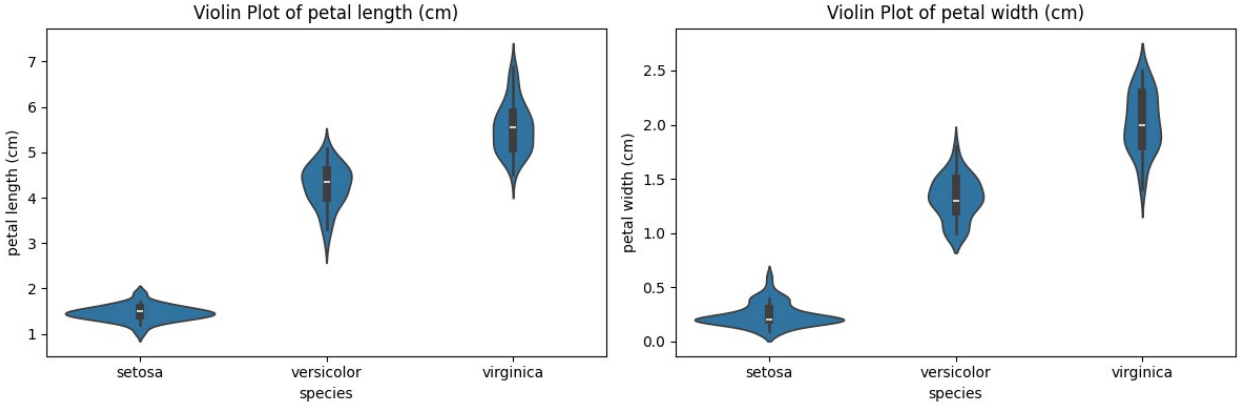
Box Plot of sepal width (cm)
sepal width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
setosa
versicolor
virginica
species


Box Plot of petal length (cm)
petal length (cm)
7
6
5
4
3
2
1
setosa
versicolor
virginica
species

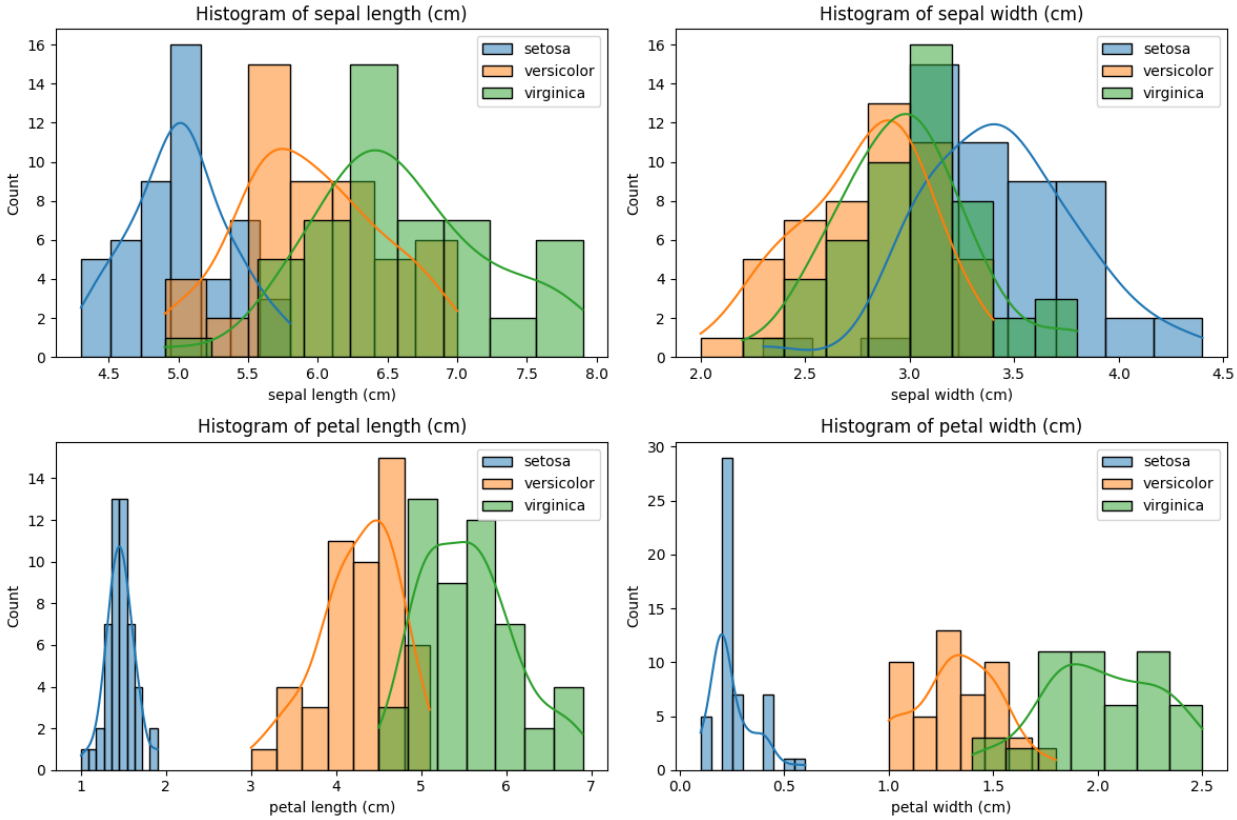
Box Plot of petal width (cm)
petal width (cm)
2.5
2.0
1.5
1.0
0.5
0.0
setosa
versicolor
virginica
species

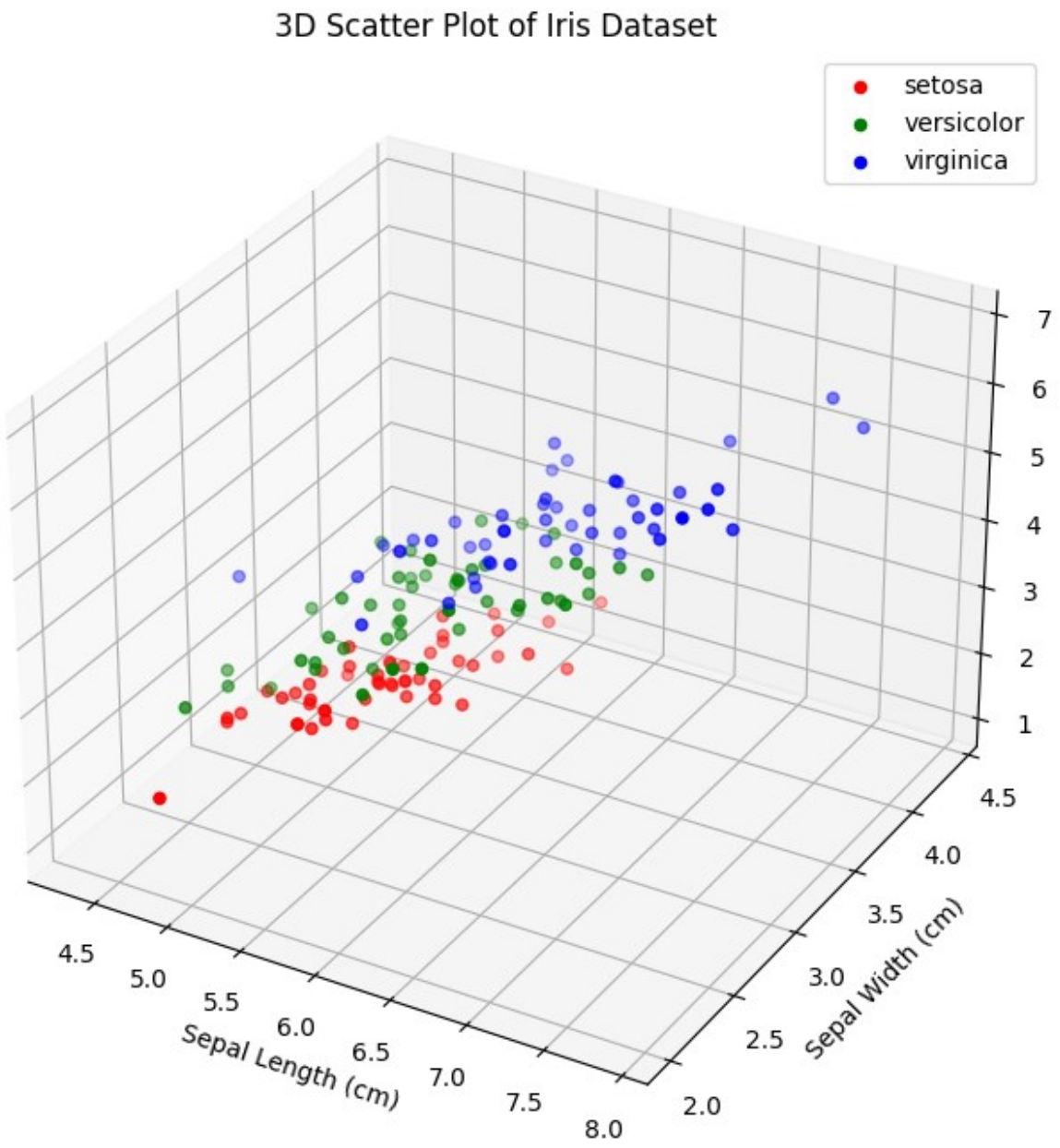

Violin Plot of sepal length (cm)
sepal length (cm)
8
7
6
5
4
setosa
versicolor
virginica
species

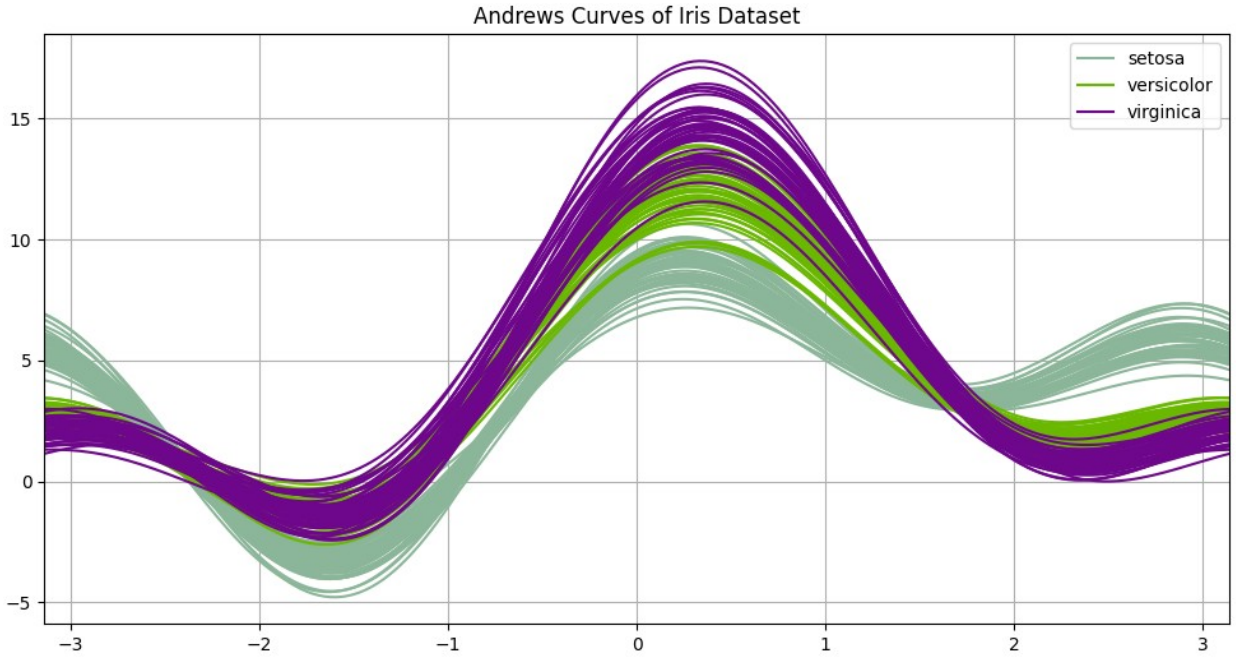
Violin Plot of sepal width (cm)
sepal width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
setosa
versicolor
virginica
species

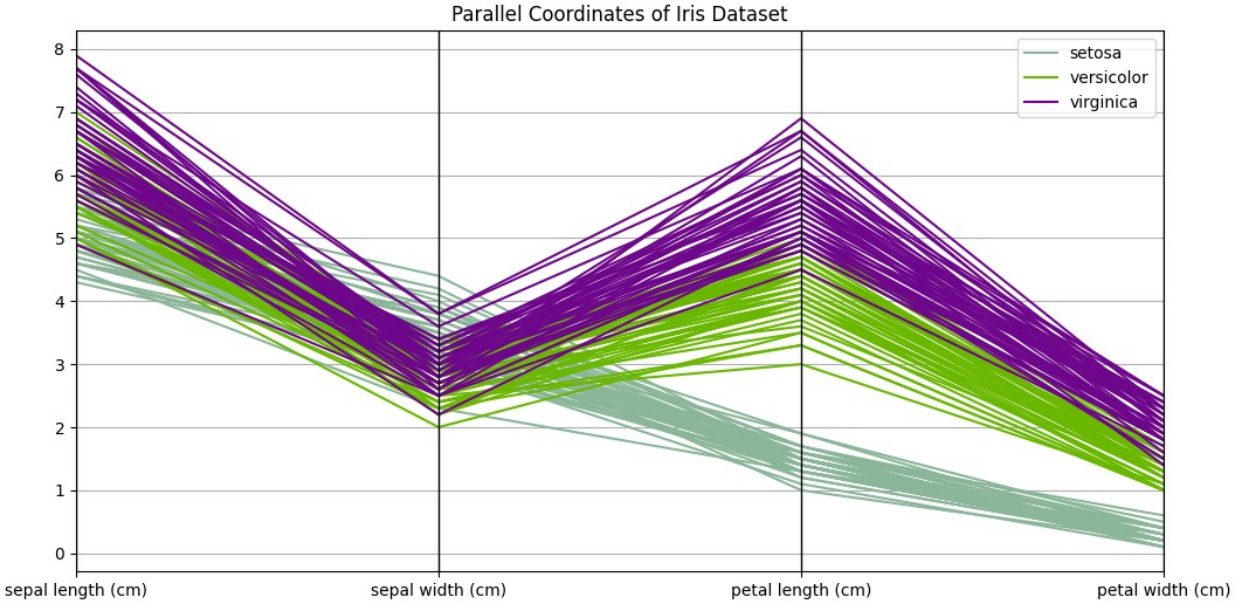

Violin Plot of petal length (cm)
petal length (cm)
7
6
5
4
3
2
1
setosa
versicolor
virginica
species

Violin Plot of petal width (cm)
petal width (cm)
2.5
2.0
1.5
1.0
0.5
0.0
setosa
versicolor
virginica
species


Histogram of sepal length (cm)
setosa
versicolor
virginica
Count
sepal length (cm)
Histogram of sepal width (cm)
setosa
versicolor
virginica
Count
sepal width (cm)
Histogram of petal length (cm)
setosa
versicolor
virginica
Count
petal length (cm)
Histogram of petal width (cm)
setosa
versicolor
virginica
Count
petal width (cm)


3D Scatter Plot of Iris Dataset
setosa
versicolor
virginica
Sepal Length (cm)
Sepal Width (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
2.0
2.5
3.0
3.5
4.0
4.5
1
2
3
4
5
6
7






```
<ipython-input-16-af879c7d5bc5>:112: MatplotlibDeprecationWarning:
Auto-removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  ax = plt.subplot(111, polar=True)
<ipython-input-16-af879c7d5bc5>:141: MatplotlibDeprecationWarning:
Auto-removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  plt.subplot(132, polar=True)
```

```
<ipython-input-16-af879c7d5bc5>:112: MatplotlibDeprecationWarning:
Auto-removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  ax = plt.subplot(111, polar=True)
<ipython-input-16-af879c7d5bc5>:144: MatplotlibDeprecationWarning:
Auto-removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  plt.subplot(133, polar=True)
<ipython-input-16-af879c7d5bc5>:112: MatplotlibDeprecationWarning:
Auto-removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  ax = plt.subplot(111, polar=True)
```