Task: Iris Flower Classification

Data Information

The Data set contains 3 classes of 50 instances each, where each refers to a type of Iris plant. One class is linearly separable from the 2;the latter are NOT linearly separable from each other.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: Iris setosa--Iris Versicolour--Iris Virginica

Importing modules

In [138…

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

Loading the Dataset

In [139…

```
df=pd.read_csv('Iris.csv')
```

In [140…

```
df.head()
```

Out[140]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [141…

```
#delete the colums
df=df.drop(columns=['Id'])
df.head()
```

Out[141]:

	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 [142…

```
# display the summary
df.describe()
```

Out[142]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [143…

```
df.shape
```

Out[143]:

```
(150, 5)
```

In [144…

```
#display basic info about datatype
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 [145…

```
# To display the no.of samples on each speices
df['Species'].value_counts()
```

Out[145]:

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

Preprocessing the Dataset

In [146…

```
# checking the null values
df.isnull().sum()
```

Out[146]:

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

Exploratory Data Analysis

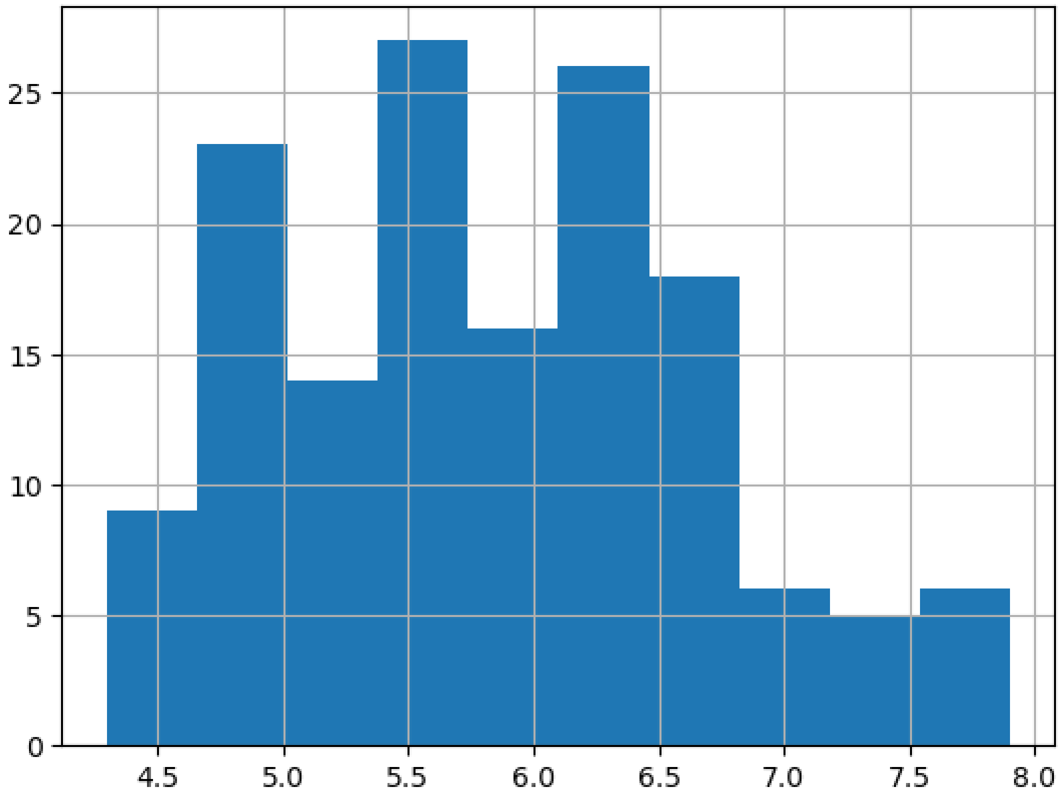
In [147…

```
#histogram plots

df['SepalLengthCm'].hist()
```

Out[147]:

```
<Axes: >
```

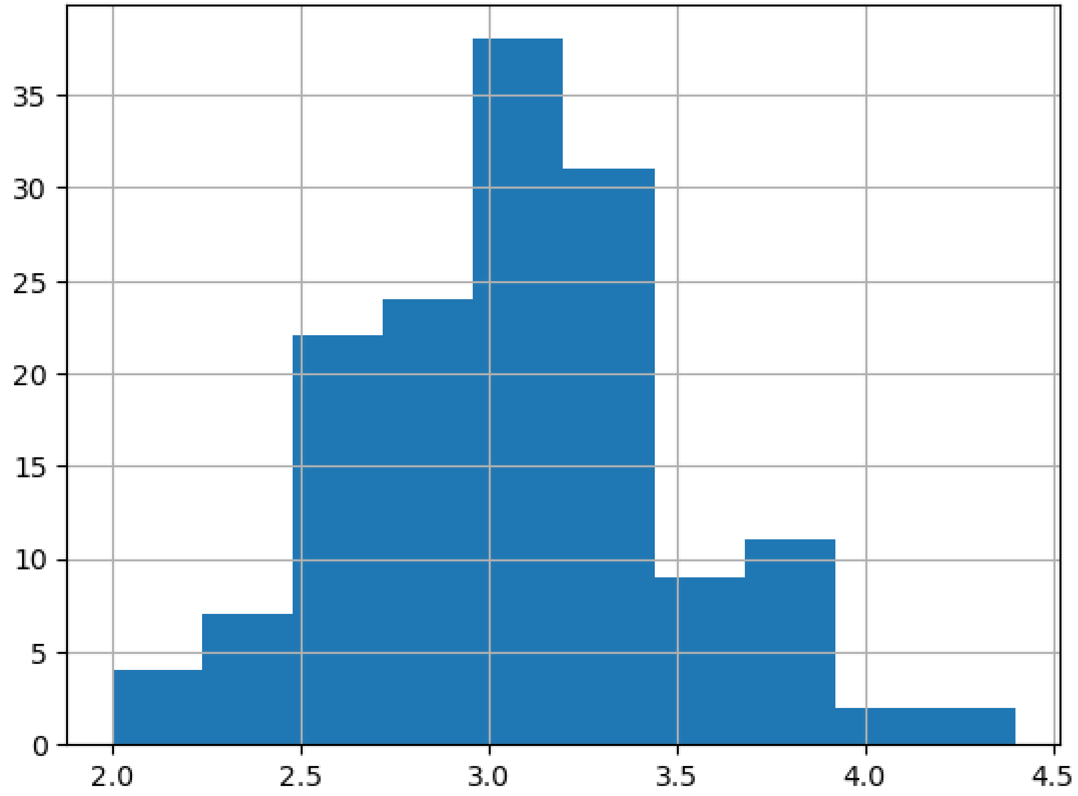


In [148…

```
df['SepalWidthCm'].hist()
```

Out[148]:

```
<Axes: >
```

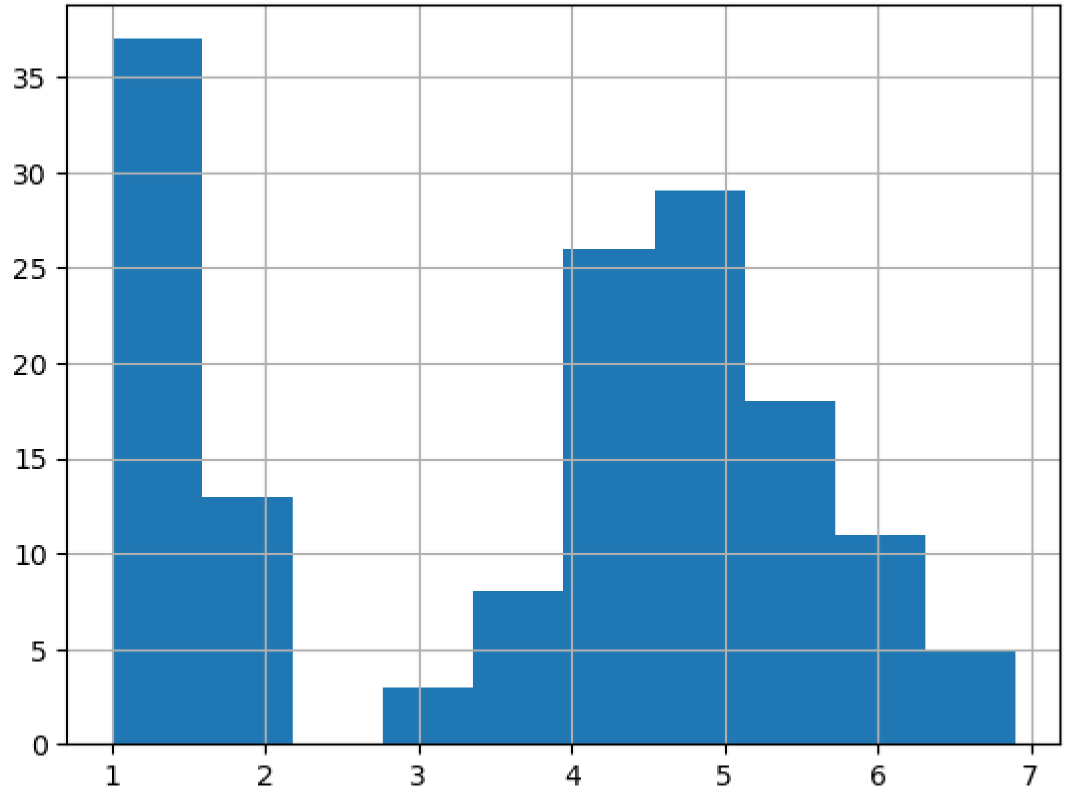


In [149…

```
df['PetalLengthCm'].hist()
```

Out[149]:

```
<Axes: >
```

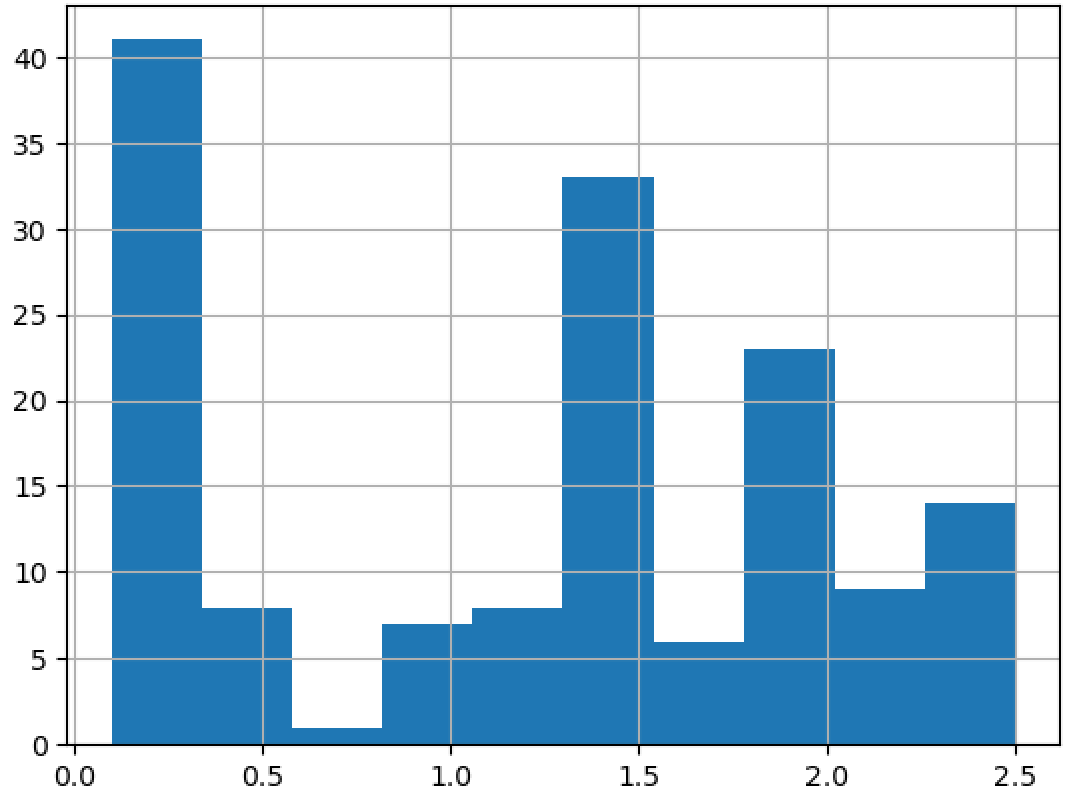


In [150…

```
df['PetalWidthCm'].hist()
```

Out[150]:

```
<Axes: >
```



In [151…

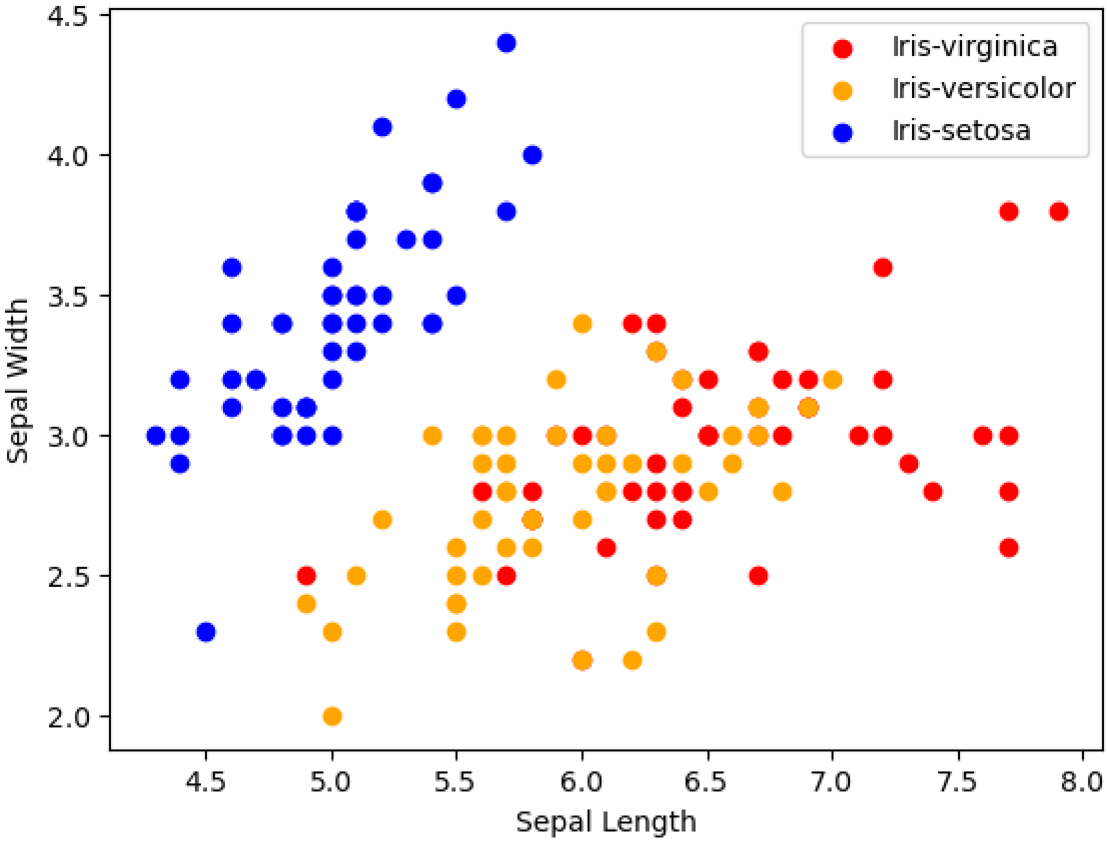
```
# scatterplot
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

In [152…

```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i], label=specie
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

Out[152]:

```
<matplotlib.legend.Legend at 0x1bee1f69810>
```

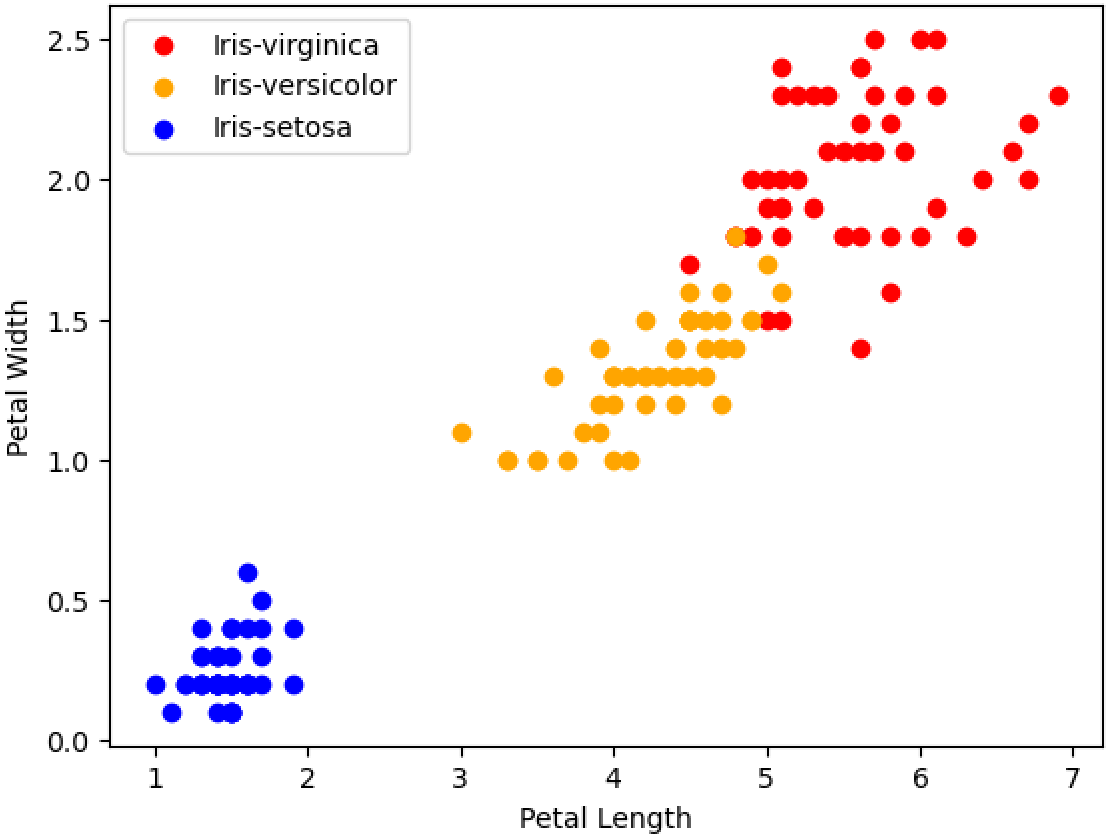


In [153…

```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i], label=specie
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
```

Out[153]:

```
<matplotlib.legend.Legend at 0x1bee1f1e7d0>
```

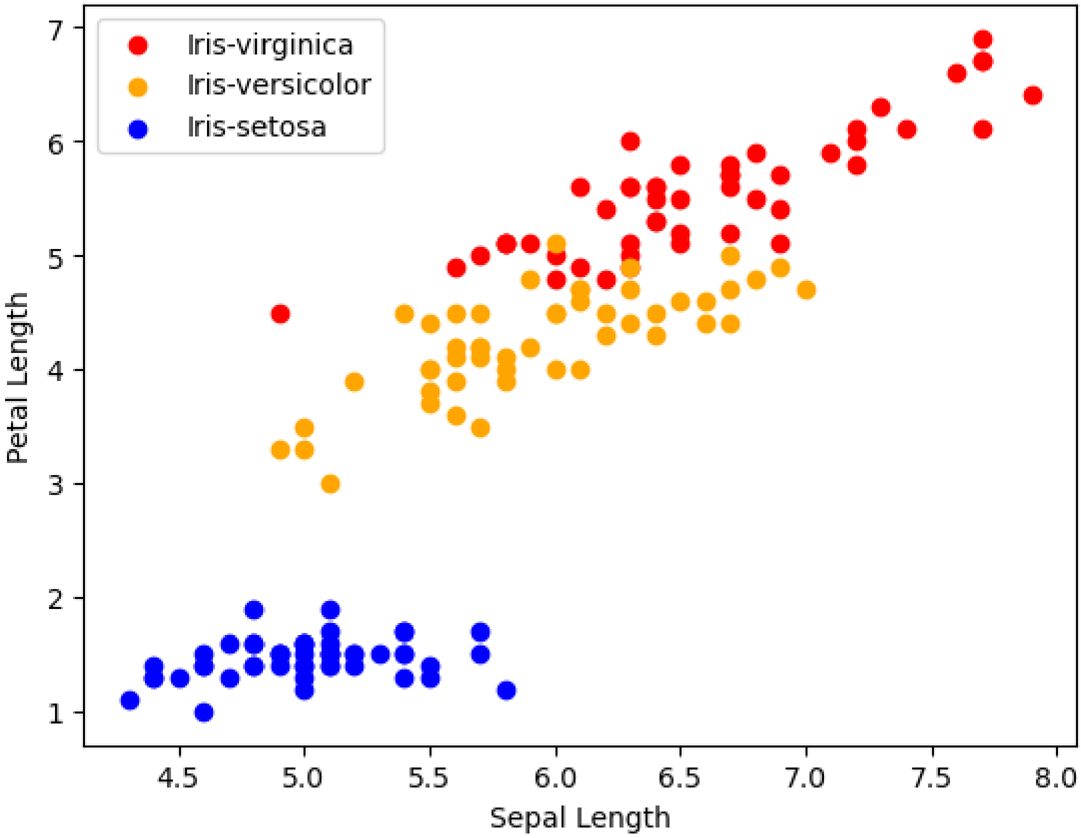


In [154…

```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i], label=speci
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

Out[154]:

```
<matplotlib.legend.Legend at 0x1bee208e7d0>
```

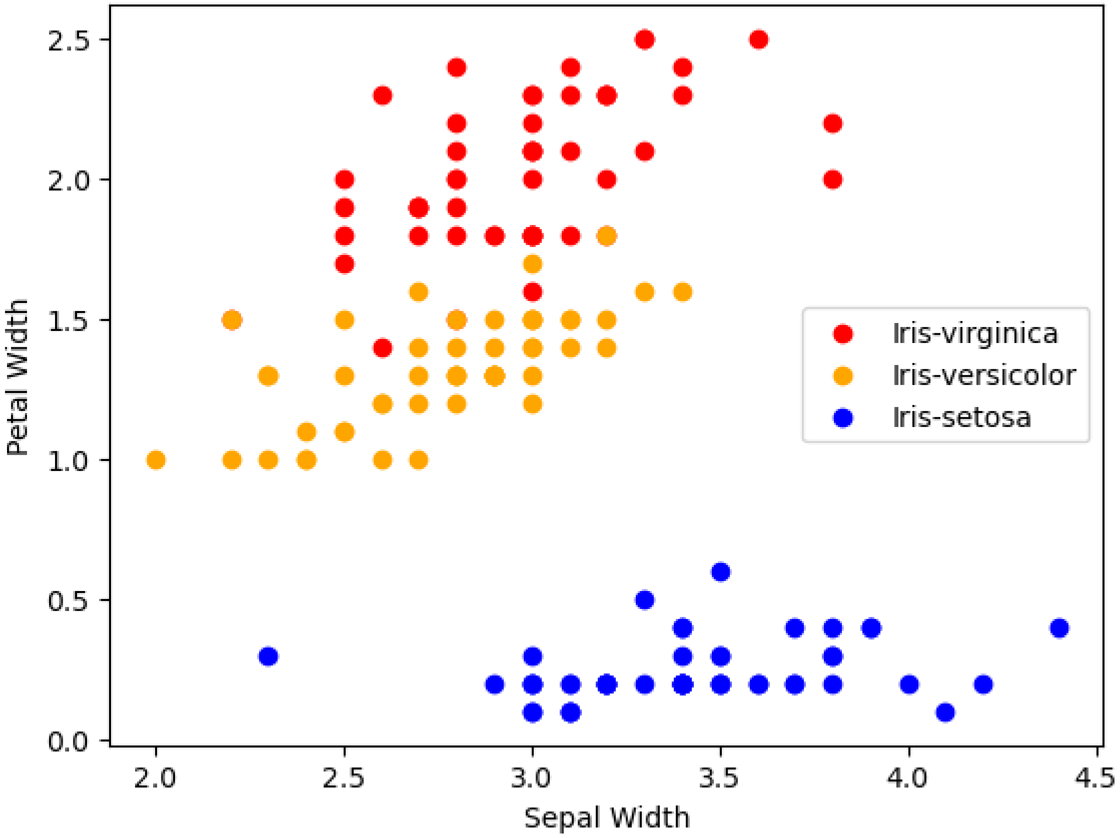


In [156…

```
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalWidthCm'], x['PetalWidthCm'], c = colors[i], label=species
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")
plt.legend()
```

Out[156]:

```
<matplotlib.legend.Legend at 0x1bee1dcd750>
```

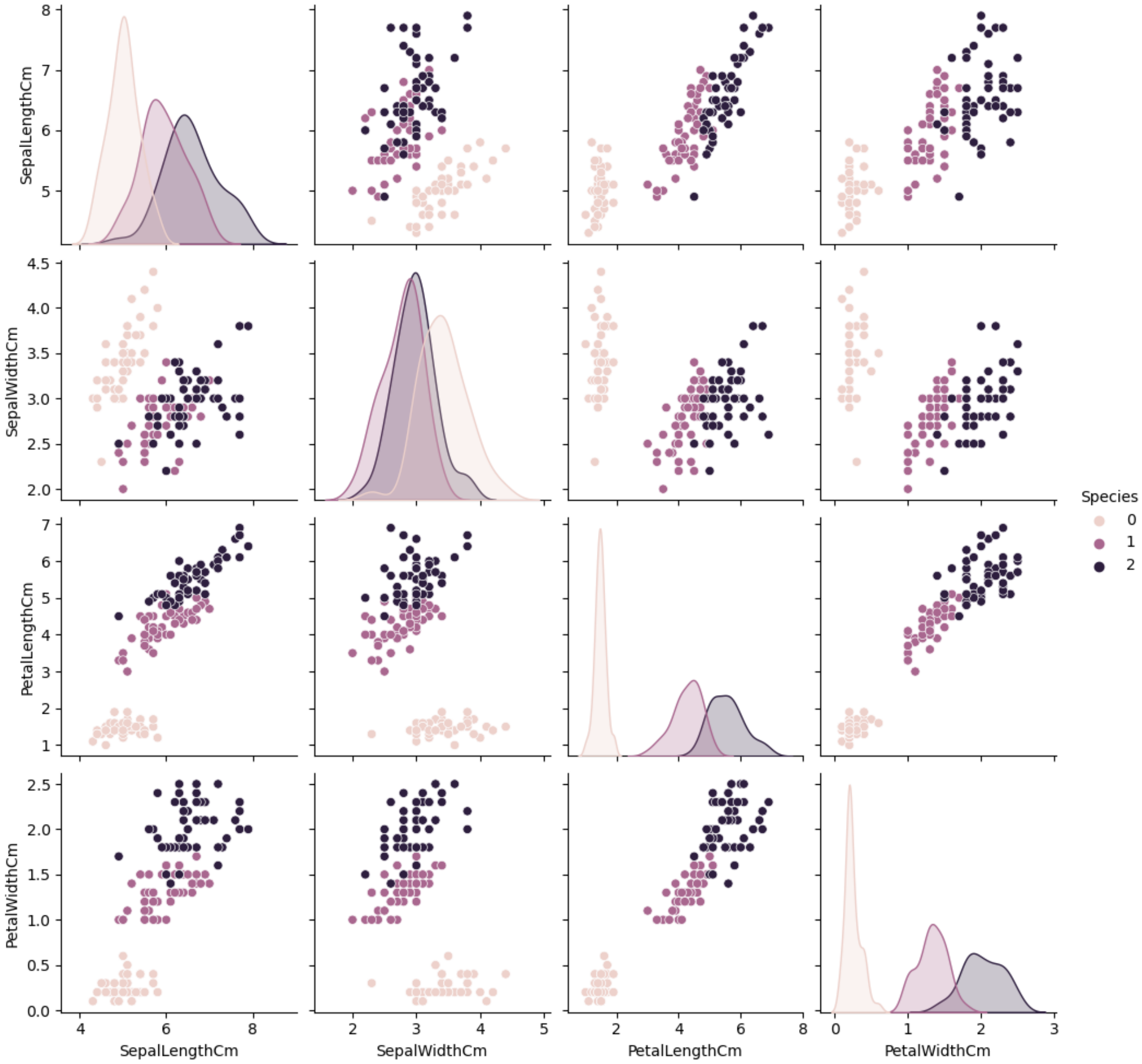


In [171…

```
sns.pairplot(df,hue='Species')
```

Out[171]:

```
<seaborn.axisgrid.PairGrid at 0x1bee3555210>
```



Correlation Matrix

In [172…

```
df.corr()
```

Out[172]:

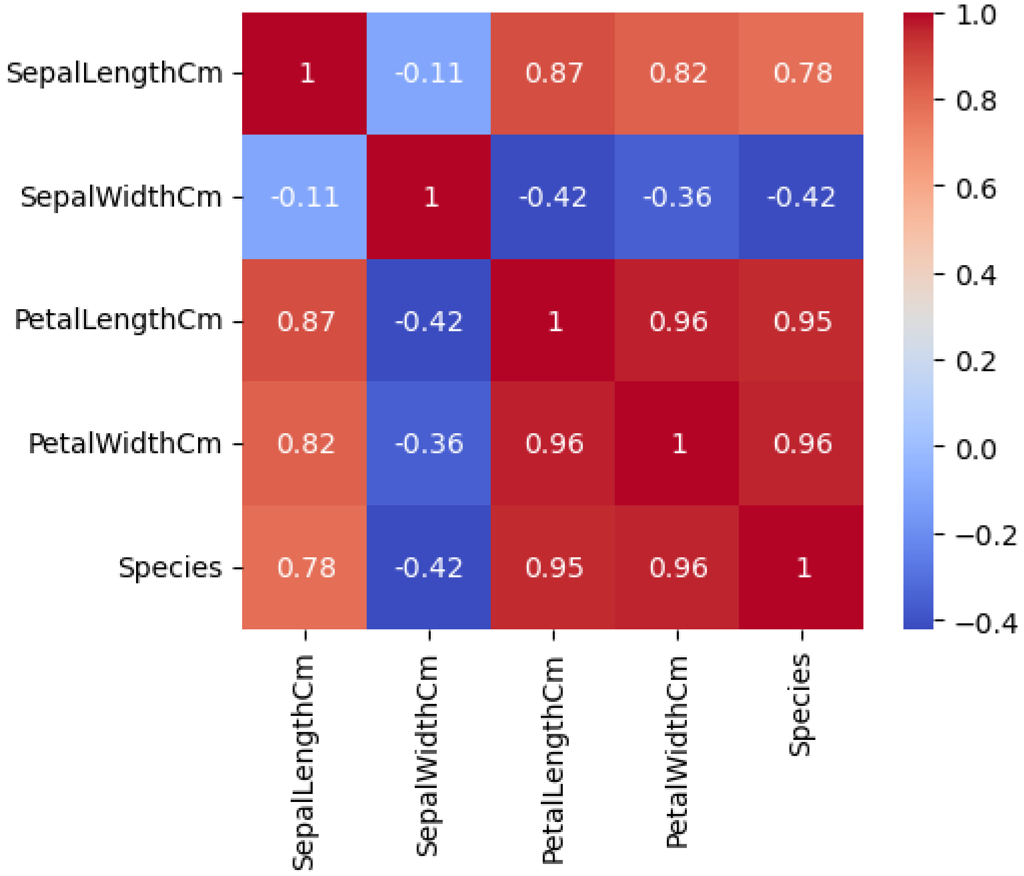
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954	0.782561
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757	0.949043
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000	0.956464
Species	0.782561	-0.419446	0.949043	0.956464	1.000000

In [173…

```
# display the correlation matrix using a heatmap
corr = df.corr()
fig, ax = plt.subplots(figsize=(5, 4))
sns.heatmap(corr, annot=True, ax=ax, cmap='coolwarm')
```

Out[173]:

```
<Axes: >
```



Model Training

In [174…

```
# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split
X = df.drop(columns=['Species'])
Y = df['Species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
```

Model-1

In [175…

```
# Logistic Regression Model
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression()
model1.fit(x_train, y_train)
accuracy_logreg = model1.score(x_test, y_test) * 100
print("Accuracy (Logistic Regression): ", accuracy_logreg)
```

```
Accuracy (Logistic Regression):  96.66666666666667
```

Model-2

In [176…

```
# K-nearest Neighbours Model (KNN)
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier()
model2.fit(x_train, y_train)
accuracy_knn = model2.score(x_test, y_test) * 100
print("Accuracy (KNN): ", accuracy_knn)
```

```
Accuracy (KNN):  100.0
```

Model-3

In [177…

```
# Decision Tree Model
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier()
model3.fit(x_train, y_train)
accuracy_decision_tree = model3.score(x_test, y_test) * 100
print("Accuracy (Decision Tree): ", accuracy_decision_tree)
```

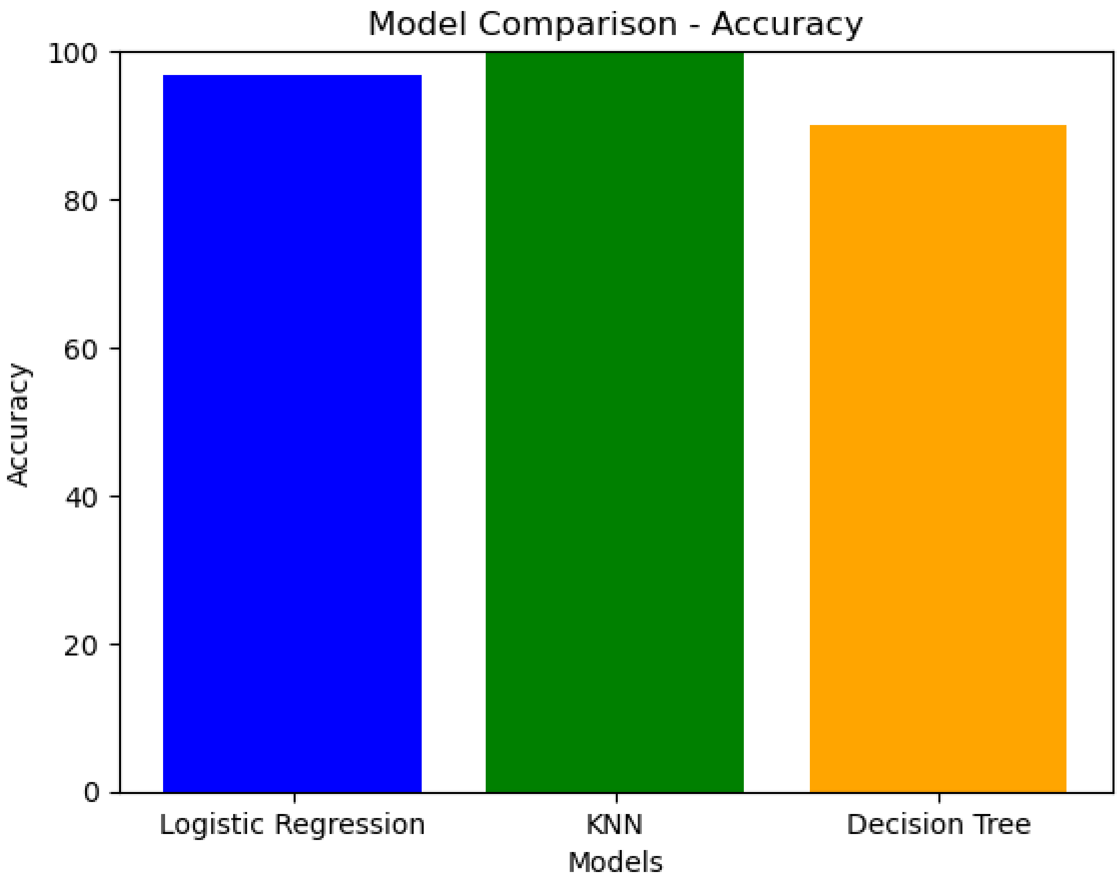
```
Accuracy (Decision Tree):  90.0
```

Report

In [178…

```
# Model Comparison - Visualization
models = ['Logistic Regression', 'KNN', 'Decision Tree']
accuracies = [accuracy_logreg, accuracy_knn, accuracy_decision_tree]

plt.bar(models, accuracies, color=['blue', 'green', 'orange'])
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.title("Model Comparison - Accuracy")
plt.ylim([0, 100])
plt.show()
```



In [ ]:

```
```