task1

November 5, 2023

[ ]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

## Supress Warnings
import warnings
warnings.filterwarnings("ignore")
```

[4]:
```
from google.colab import files

uploaded = files.upload()
```

```
<IPython.core.display.HTML object>

Saving Iris.csv to Iris.csv
```

[ ]:
```
import pandas as pd

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

[ ]:
```
iris_df = pd.read_csv("Iris.csv")
print("the data has been successfully loaded")
```

```
the data has been successfully loaded
```

[ ]:
```
iris_df
```

[ ]:
```
       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0       1            5.1           3.5            1.4           0.2
1       2            4.9           3.0            1.4           0.2
2       3            4.7           3.2            1.3           0.2
3       4            4.6           3.1            1.5           0.2
4       5            5.0           3.6            1.4           0.2
..    ...            ...           ...            ...           ...
145   146            6.7           3.0            5.2           2.3
146   147            6.3           2.5            5.0           1.9
147   148            6.5           3.0            5.2           2.0
148   149            6.2           3.4            5.4           2.3
149   150            5.9           3.0            5.1           1.8

            Species
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

[150 rows x 6 columns]
```

[ ]:
```
iris_df.shape
```

[ ]:
```
(150, 6)
```

[ ]:
```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[ ]:
```
iris_df.describe()
```

[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

[ ]:
```
iris_df.isnull().sum()
```

[ ]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

[ ]:
```
print("unique number of values in dataset species:", iris_df["Species"].
  ↪nunique())
print("unique species in iris dataset:", iris_df["Species"].unique())
```

```
unique number of values in dataset species: 3
unique species in iris dataset: ['Iris-setosa' 'Iris-versicolor' 'Iris-
virginica']
```

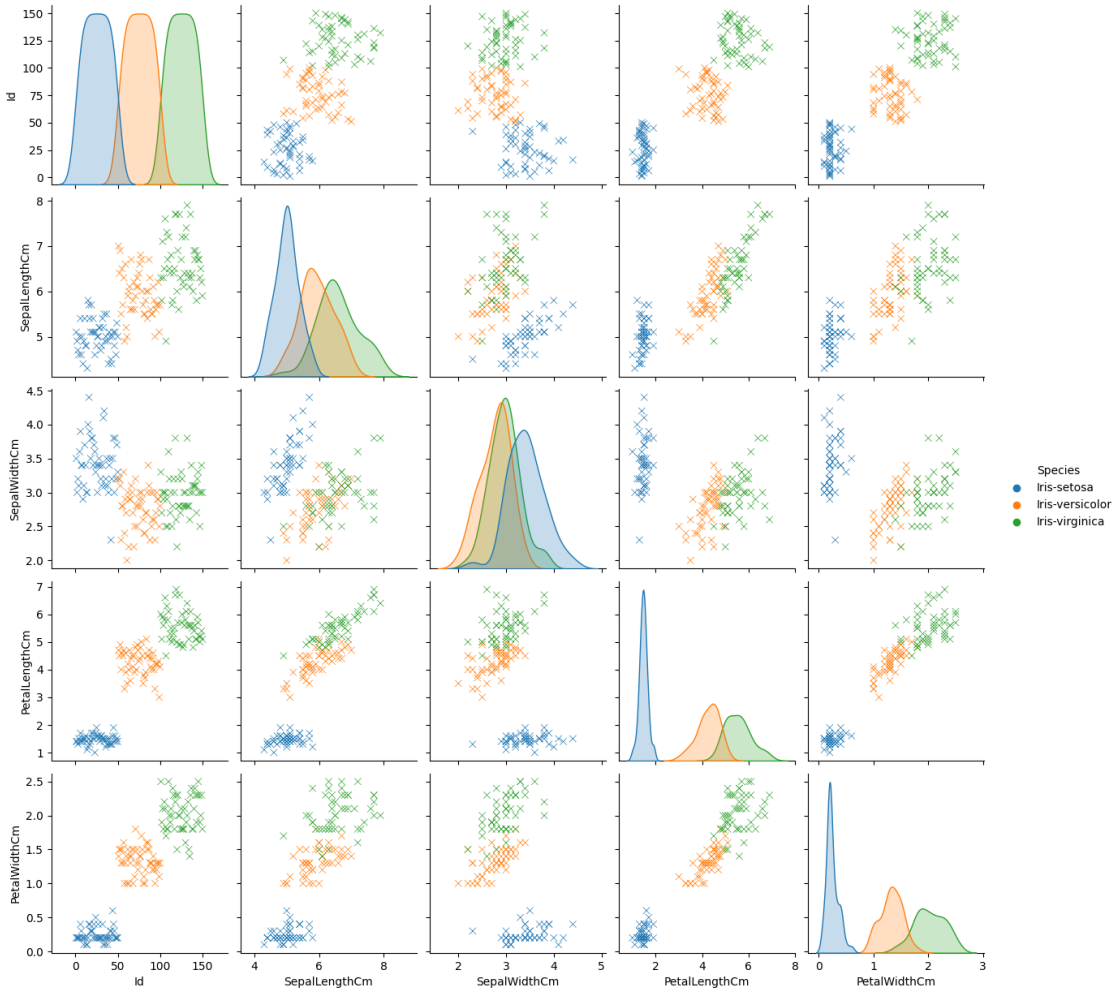
Exploratory Data Analysis

Data Visualisation

[2]:
```
import seaborn as sns
```

[6]:
```
import pandas as pd
iris_df = pd.read_csv('Iris.csv')
```

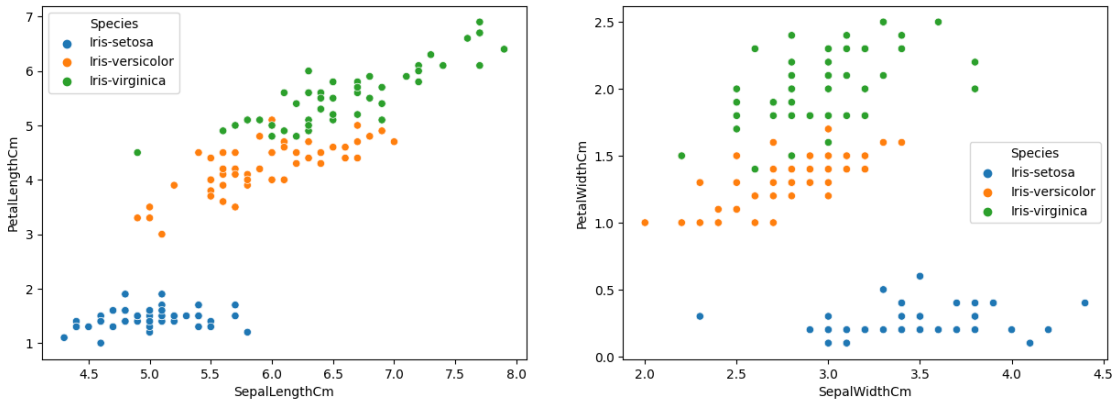
[7]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(iris_df, hue="Species", markers="x")
plt.show()
```



Shows that Iris-sentosa is separated from both other soecies in all features

[8]:
```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
sns.scatterplot(x='SepalLengthCm', y='PetalLengthCm', data=iris_df,␣
↪hue='Species')
```

```
plt.subplot(1,2,2)
sns.scatterplot(x='SepalWidthCm', y='PetalWidthCm', data=iris_df, hue='Species')

plt.show()
```



Check Correction in Dataset

[9]:
```
iris_df.corr()
```

```
<ipython-input-9-1b33314f8075>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  iris_df.corr()
```

[9]:

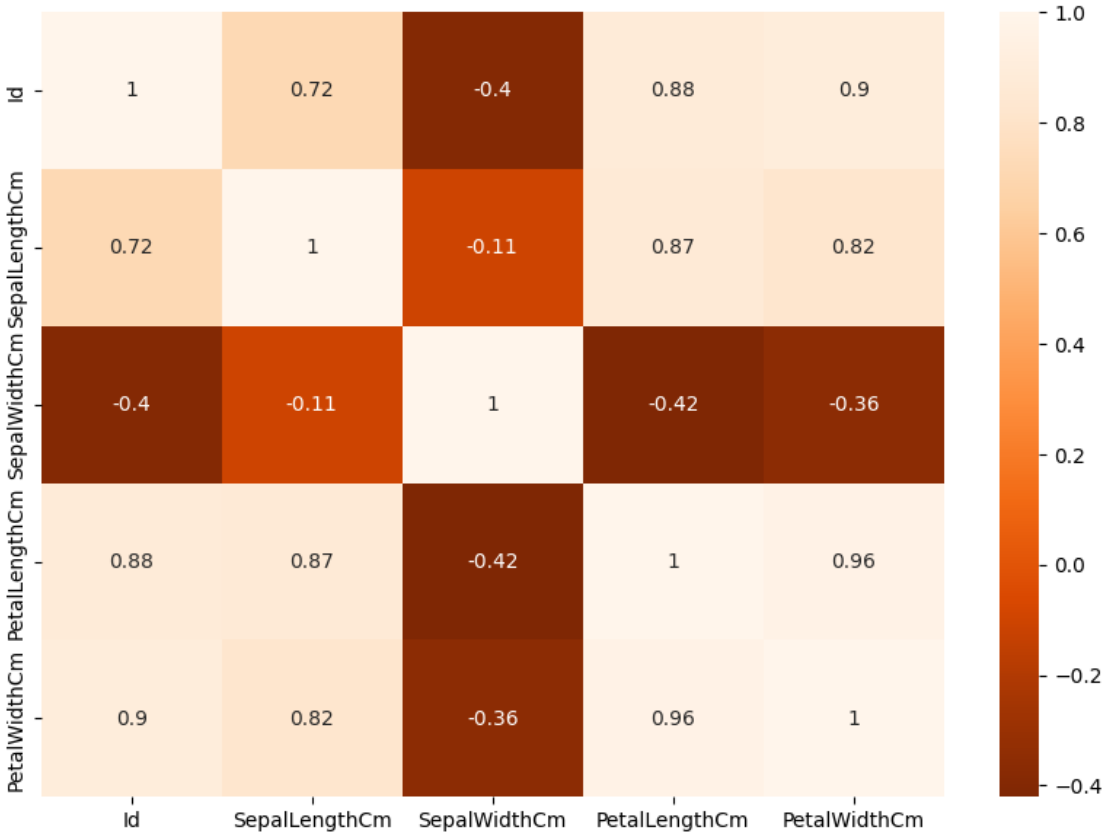
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	\
Id	1.000000	0.716676	-0.397729	0.882747	
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	

	PetalWidthCm
Id	0.899759
SepalLengthCm	0.817954
SepalWidthCm	-0.356544
PetalLengthCm	0.962757
PetalWidthCm	1.000000

[10]:
```
plt.figure(figsize=(10,7))
sns.heatmap(iris_df.corr(),annot = True,cmap = "Oranges_r")
plt.show()
```

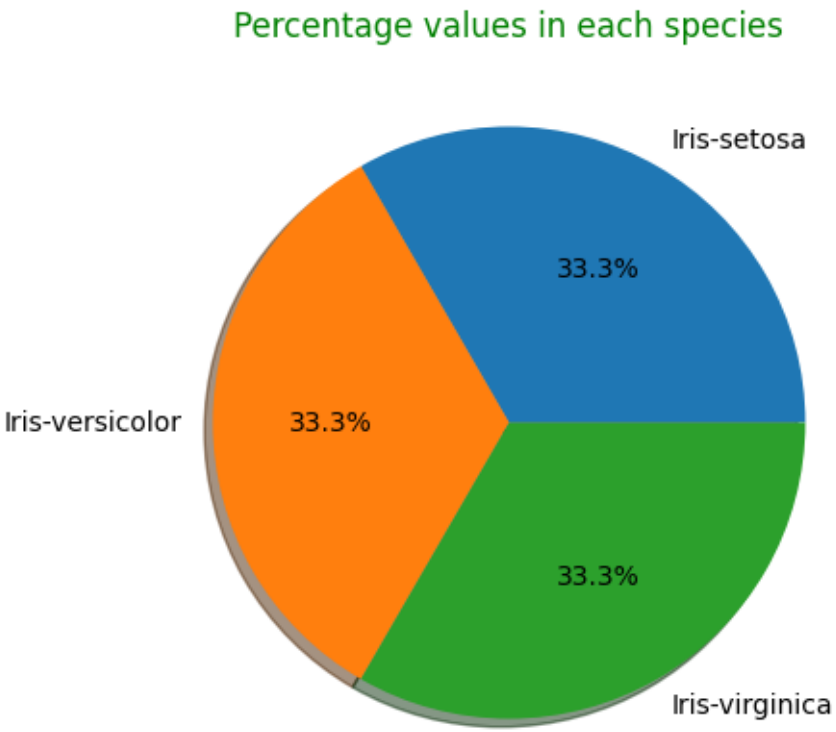
```
<ipython-input-10-fe35941b4079>:2: FutureWarning: The default value of
```

```
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(iris_df.corr(),annot = True,cmap = "Oranges_r")
```



We see petal length and petal width is highly correlated in above heatmap

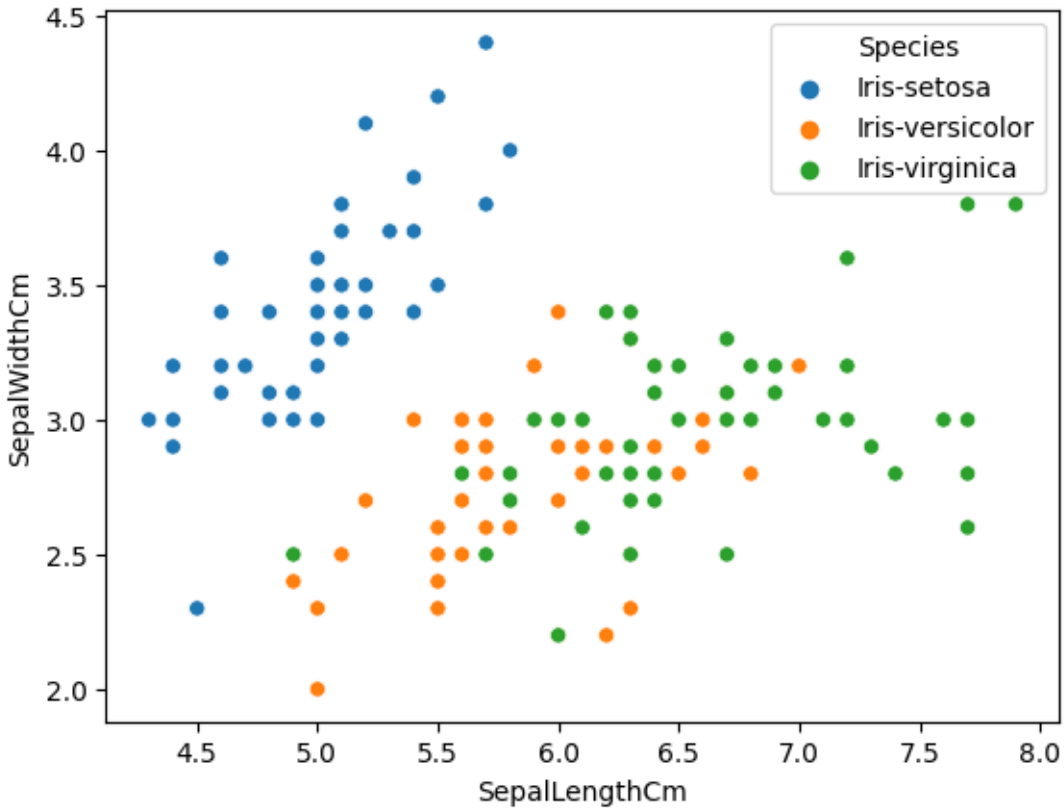
[11]:
```
iris_df["Species"].value_counts().plot(kind="pie",autopct = "%1.
  ↪1f%%",shadow=True, figsize=(5,5))
plt.title("Percentage values in each species", fontsize = 12, c='g')
plt.ylabel("",fontsize=10,c="r")
plt.show()
```



- All species have equal values in dataset
- Iris-Sentosa:50
- Iris-Versicolor:50
- Iris-Virginica:50

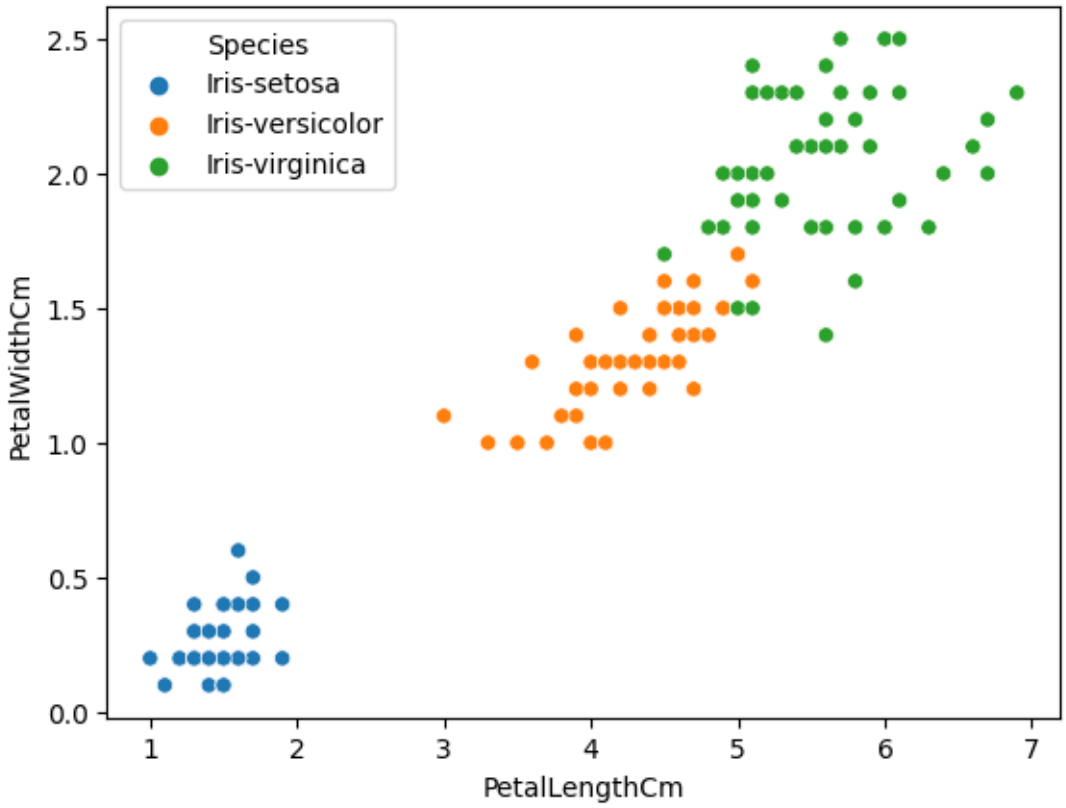
Scatterplot for Sepal Lenth and Sepal Width

[13]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x=iris_df["SepalLengthCm"], y=iris_df["SepalWidthCm"],␣
 ↪hue=iris_df["Species"])
plt.show()
```

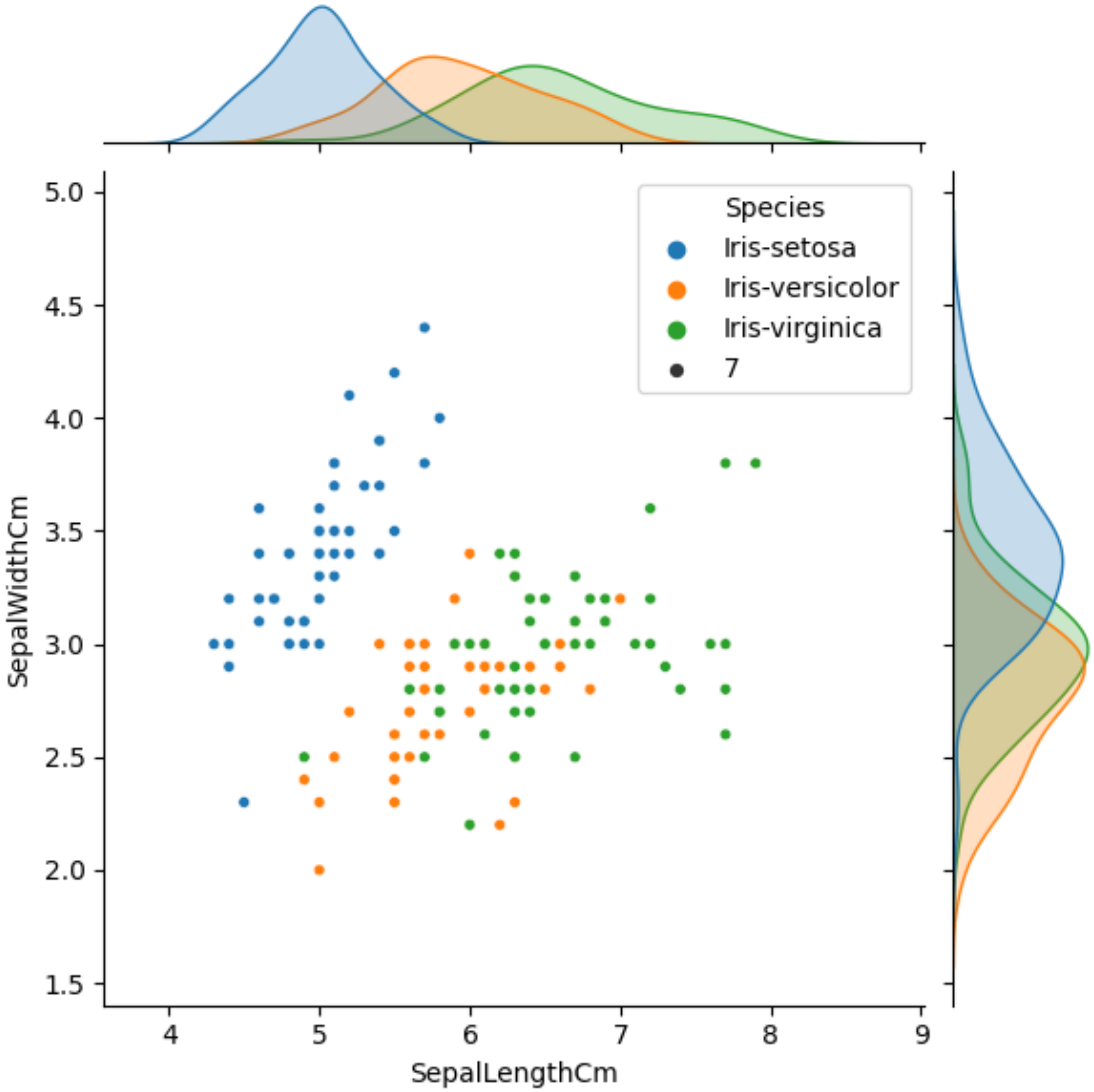


Scatterplot for Petal Length and Petal Width

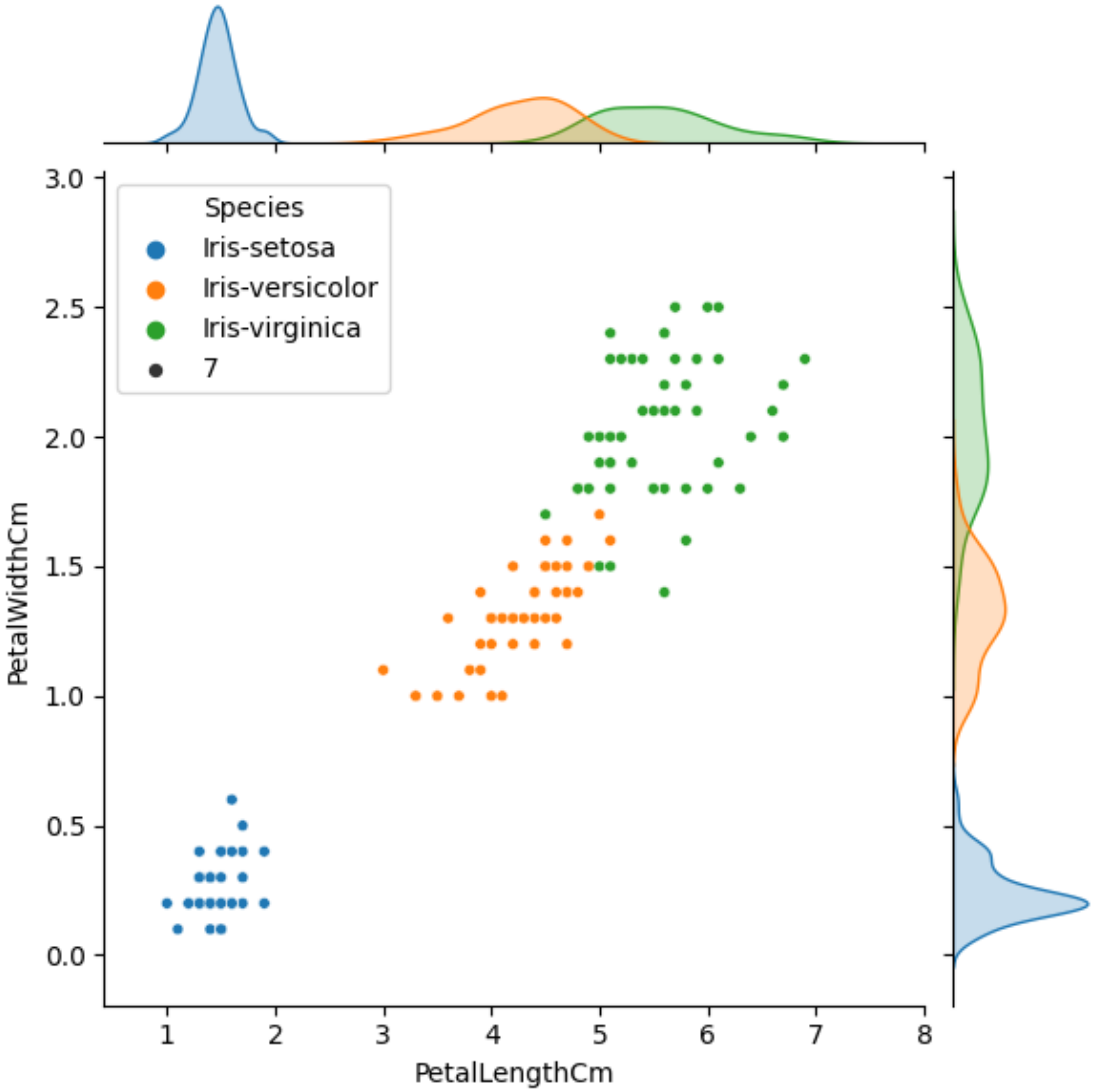
[15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(x=iris_df["PetalLengthCm"], y=iris_df["PetalWidthCm"],␣
↪hue=iris_df["Species"])
plt.show()
```



[16]:
```
sns.jointplot(data = iris_df, x="SepalLengthCm", y="SepalWidthCm", size = 7,␣
↪hue = "Species")

plt.show()
```

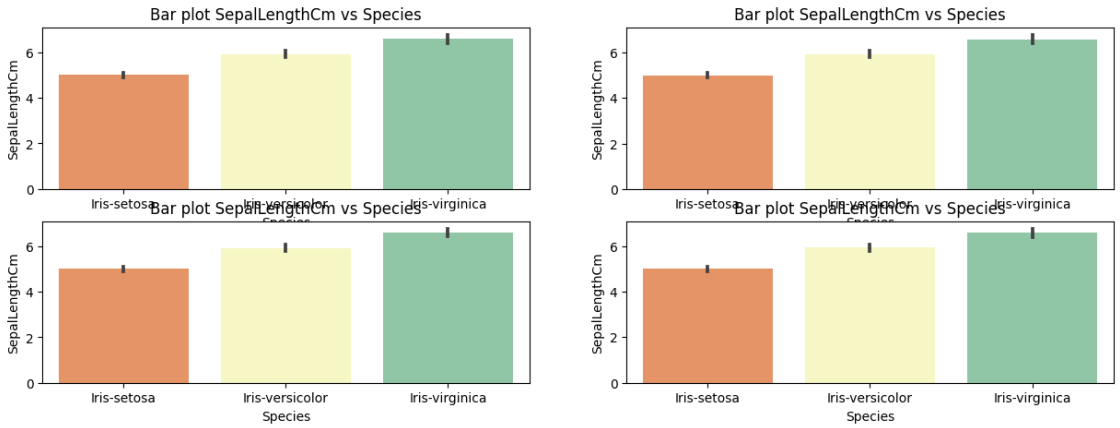


[17]:
```
sns.jointplot(data = iris_df, x = "PetalLengthCm", y = "PetalWidthCm", size =␣
↪7, hue = "Species")
plt.show()
```

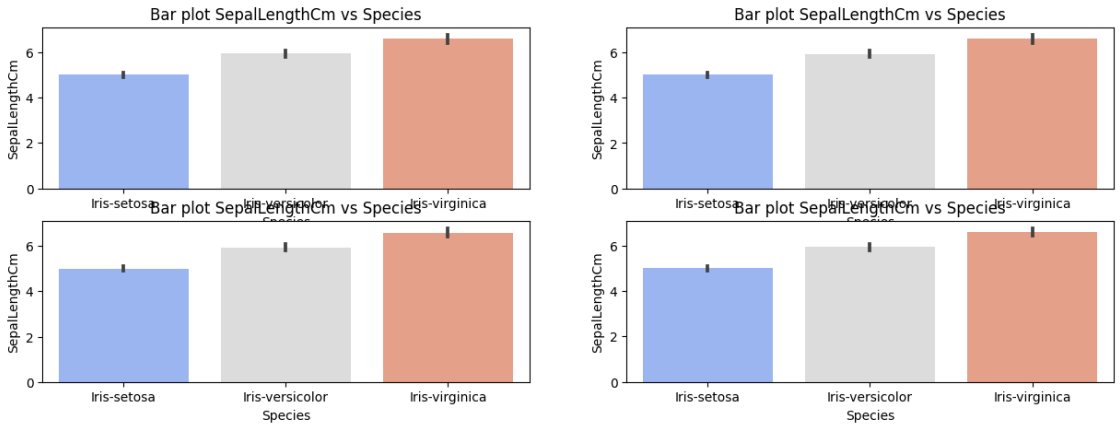


[18]:
```
plt.figure(figsize = (15,5))
plt.subplot(2,2,1)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("Spectral"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.subplot(2,2,2)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("Spectral"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.subplot(2,2,3)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("Spectral"))
plt.title("Bar plot SepalLengthCm vs Species")
```

```
plt.subplot(2,2,4)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("Spectral"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.show()
```



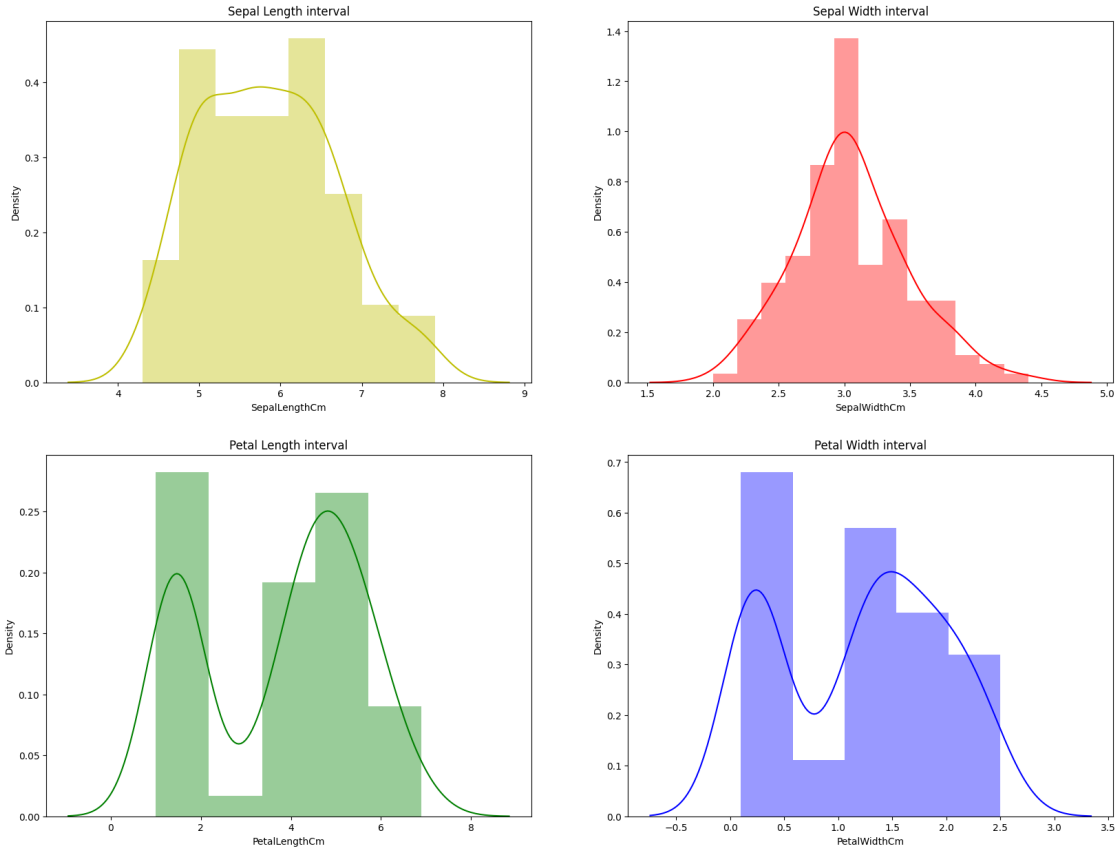
[19]:
```
plt.figure(figsize = (15,5))
plt.subplot(2,2,1)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("coolwarm"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.subplot(2,2,2)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("coolwarm"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.subplot(2,2,3)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("coolwarm"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.subplot(2,2,4)
sns.barplot(x = "Species", y="SepalLengthCm",data=iris_df, palette=("coolwarm"))
plt.title("Bar plot SepalLengthCm vs Species")

plt.show()
```



[22]:
```
plt.figure(figsize=(20,15))
plt.subplot(2,2,1)
sns.distplot(iris_df["SepalLengthCm"],color="y").set_title("Sepal Length␣
 ↪interval")

plt.subplot(2,2,2)
sns.distplot(iris_df["SepalWidthCm"],color="r").set_title("Sepal Width␣
 ↪interval")

plt.subplot(2,2,3)
sns.distplot(iris_df["PetalLengthCm"],color="g").set_title("Petal Length␣
 ↪interval")

plt.subplot(2,2,4)
sns.distplot(iris_df["PetalWidthCm"],color="b").set_title("Petal Width␣
 ↪interval")

plt.show()
```

```
<ipython-input-22-d489ffb1486c>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(iris_df["SepalLengthCm"],color="y").set_title("Sepal Length
interval")
<ipython-input-22-d489ffb1486c>:6: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(iris_df["SepalWidthCm"],color="r").set_title("Sepal Width
interval")
<ipython-input-22-d489ffb1486c>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(iris_df["PetalLengthCm"],color="g").set_title("Petal Length
interval")
<ipython-input-22-d489ffb1486c>:12: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(iris_df["PetalWidthCm"],color="b").set_title("Petal Width
interval")
```



Data Cleaning

[24]:
```
from sklearn.preprocessing import LabelEncoder
```

[25]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

iris_df["Species"] = le.fit_transform(iris_df["Species"])
iris_df.head()
```

[25]:

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

[26]:
```
X = iris_df.iloc[:,[0,1,2,3]]
X.head()
```

[26]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4

[27]:
```
y = iris_df.iloc[:, -1]
y.head()
```

[27]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int64
```

[28]:
```
print(X.shape)
print(y.shape)
```

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

Model Building

[29]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
  ↪2,random_state=0)
```

[30]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 4)
(30, 4)
(120,)
(30,)
```

Logistic Regression

[33]:
```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()

lr.fit(X_train, y_train)
print("Logistic regression successfully implemented")

y_pred = lr.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix : - ")
print(cm)

accuracy = accuracy_score(y_test,y_pred)

print("accuracy is:-", accuracy*100)

print("Classification Report:-")
from sklearn.metrics import classification_report

# ...

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

```
Logistic regression successfully implemented
Confusion Matrix : -
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
accuracy is:- 100.0
Classification Report:-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Random Forest Classifier

[35]:
```
from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier()

rfc.fit(X_train, y_train)
print("Random Forest Classifier successfully implemented")

y_pred = rfc.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix : - ")
print(cm)

accuracy = accuracy_score(y_test,y_pred)

print("accuracy is:-", accuracy*100)

print("Classification Report:-")
from sklearn.metrics import classification_report

# ...

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

```
Random Forest Classifier successfully implemented
Confusion Matrix : -
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
accuracy is:- 100.0
Classification Report:-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Decision Tree

[36]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
```

```
dtc.fit(X_train, y_train)

print("Decision Tree Classifier successfully implemented")

y_pred = dtc.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy*100, "%")

from sklearn.metrics import classification_report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Decision Tree Classifier successfully implemented
Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
Accuracy: 96.66666666666667 %
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.93      1.00      0.96        13
           2       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

[37]:
```
from sklearn.tree import plot_tree
```

[1]:
```
feature = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
classes = ['Iris-Sentosa','Iris-Versicolor','Iris-Virginica']
```

Support Vector Machine

[22]:
```
from google.colab import files

uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Iris.csv to Iris.csv
```

[23]:
```
import pandas as pd

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

[27]:
```
import seaborn as sns

# Load the Iris dataset
iris_df = sns.load_dataset('iris')

# Now you can proceed with your machine learning tasks using `iris_df`
```

[28]:
```
import pandas as pd

# Load your dataset
your_dataset_df = pd.read_csv('Iris.csv')

# Now you can proceed with your machine learning tasks using `your_dataset_df`
```

[33]:
```
X = iris_df.drop(columns=["species"])  # Features
y = iris_df["species"]  # Labels

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
from sklearn.svm import SVC

svc = SVC()
svc.fit(X_train, y_train)


svc.fit(X_train, y_train)
print("Support vector classifier is successfully implemented")

y_pred = svc.predict(X_test)

cm = confusion_matrix(y_test, y_test)
print("Confusion Matrix:-")
print(cm)

accuracy = accuracy_score(y_test,y_pred)

print("accuracy:-", accuracy*100)
print("Classification Report:-")
print( classification_report(y_test, y_pred))
```

```
Support vector classifier is successfully implemented
Confusion Matrix:-
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
accuracy:- 100.0
Classification Report:-
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   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
```

[32]:
```
print(iris_df.columns)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

K-NN Classifier

[34]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

knn = KNeighborsClassifier(n_neighbors=5)  # You can change the number of␣
 ↪neighbors if needed
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy*100, "%")

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy: 100.0 %
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   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
```

Naive Bayes

[35]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

naive_bayes = GaussianNB()

naive_bayes.fit(X_train, y_train)

y_pred = naive_bayes.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(cm)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy*100, "%")

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Accuracy: 100.0 %
Classification Report:
              precision    recall  f1-score   support
```

```
      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   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
```

Test Model

[38]:
```
import numpy as np
```

[46]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

X = iris_df.drop(columns=["species"])
y = iris_df["species"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

dtc = DecisionTreeClassifier()

dtc.fit(X_train, y_train)
input_data = [[4.9, 3.0, 1.4, 0.2]]

input_data_as_nparray = np.asarray(input_data)
input_data_reshaped = input_data_as_nparray.reshape(1,-1)

prediction = dtc.predict(input_data_reshaped)
print("The category is", prediction)
```

```
The category is ['setosa']

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but DecisionTreeClassifier was fitted with feature
names
  warnings.warn(
```

[ ]: