Import Libraries we want to use

[2]:
```
import pandas as pd  # for data operattions
import matplotlib.pyplot as plt  # for plotting
import numpy as np
```

[3]:
```
import seaborn as sns  # for data visualization .. Seaborn is a powerful Python data visualization library
# built on top of Matplotlib. It provides high-Level functions for creating attractiveand informative statistical graphics.
```

Database loading

[5]:
```
# Load the iris dataset .. in same folder
dataset = pd.read_csv("iris.csv") # load dataset into a pandas DataFrame.
dataset.head() #to ensure read it successfully
```

[5]:

	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

[6]:
```
# Renaming the columns
dataset.columns = ['id','sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']

dataset.head() #to show data
```

[6]:

	id	sepal_length	sepal_width	petal_length	petal_width	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

Exploratory Data Analysis (EDA) of iris dataset

Data Exploration - Information

[9]:
```
dataset.describe()  # describing columns in dataset ..  provides a summary of each feature, including mean, standard deviation, and quartiles.
```

[9]:

	id	sepal_length	sepal_width	petal_length	petal_width
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

[10]:
```
dataset.shape
```

[10]:
```
(150, 6)
```

[11]:
```
dataset['species'].value_counts()
```

[11]:
```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Checking for missing and Duplicate values

[13]:
```
dataset.isnull().sum() #sum of null values
```

[13]:
```
id              0
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[14]:
```
dataset.duplicated().sum() # sum of duplicated values
```

[14]:
```
0
```

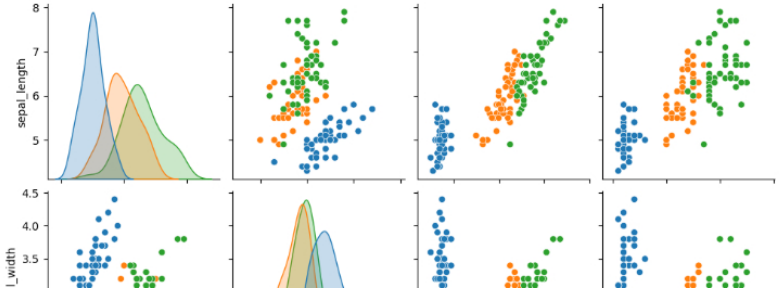
[15]:
```
dataset = dataset.drop(columns=['id']) #not important to Explorationg Data
```

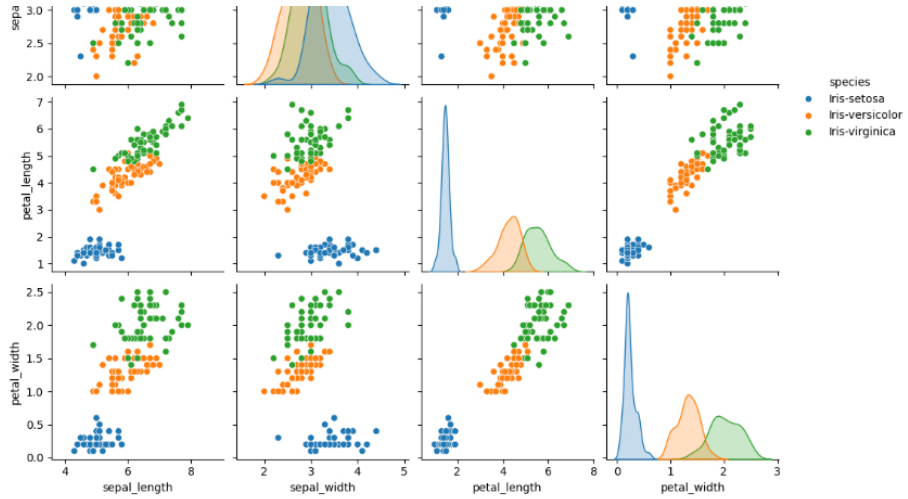
Data Visualizations

[17]:
```
# Visual Analysis - Pair Plots

sns.pairplot(dataset, hue='species')
# The pairplot() function from Seaborn creates grid of scatter plots for pairwise relationships between variables.
# It takes dataframe as input.
# It plots each numeric variable against every other numeric variable in dataframe.

plt.show()
```



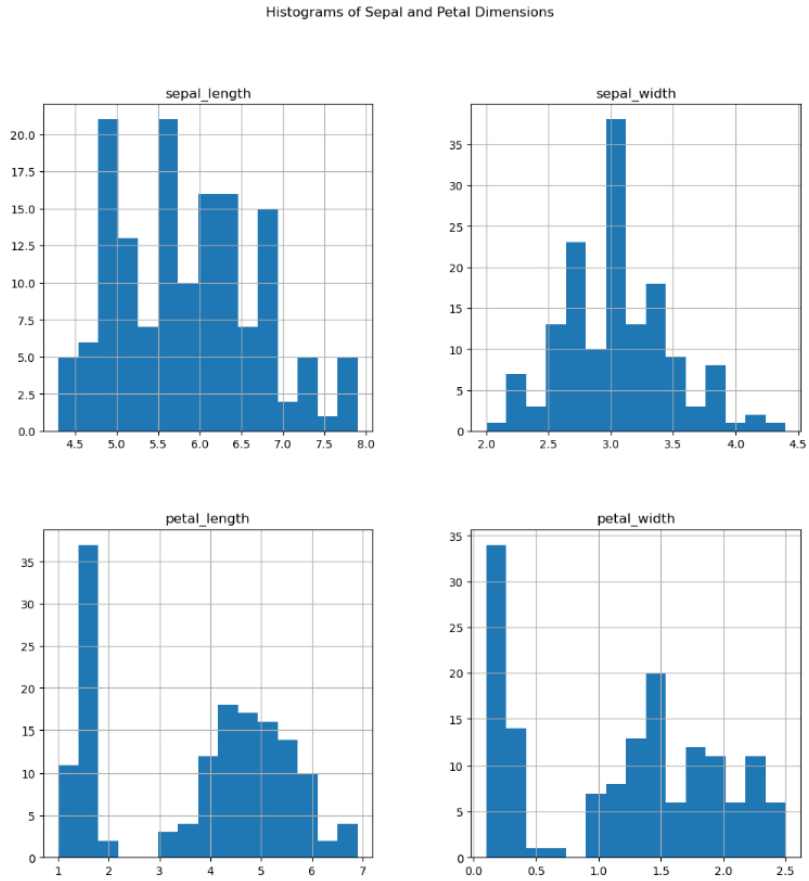


from pairplot can extract - Iris-species Setosa has the smallest of petals widths and lengths.

It also has the smallest sepal length but larger sepal widths. and some other info

[18]:
```
# Visual Analysis - Histograms

# Create histograms for the specified columns
dataset[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].hist(figsize=(12, 12), bins=15)  # Creating histograms
plt.suptitle('Histograms of Sepal and Petal Dimensions', size=12)  #title to the histograms
plt.show()  # Display the histograms
```

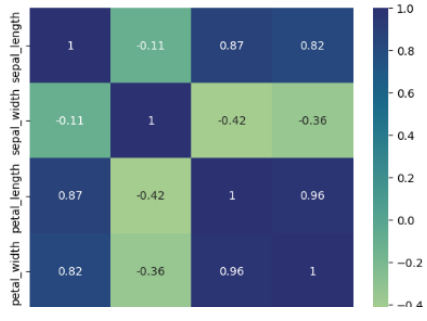


[19]:
```
# Correlation Analysis
# Correlation between species - Heat Maps
corr = dataset[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].corr()
corr
```

[19]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

*[20]:
```
sns.heatmap(corr, annot=True, cmap='crest') #shows a correlation between all numerical variables in the dataset.
plt.show()
```



sepal_length sepal_width petal_length petal_width

So Result Of EDA

- The iris dataset contains 150 samples with 5 basic columns: 'sepal_length', 'sepal_width', 'petal_length', 'petal_width', and 'species'
- There are no missing values or duplicate values
- The dataset is balanced across the three species: iris-setosa, irisversicolor, and iris-virginica

Visual Analysis

- Histograms : Show the distribution of each feature [sepal_length and petal_length] have a wider range compared to sepal_width and petal_width
- Pair Plot : Highlights the relationships between features. petal_length and petal_width show clear separation between species

Correlation Analysis

- Heat Maps to visualize correlation betwwen features

Hypothesis Testing as an Exploratory Data Analysis Technique

[23]:
```
from scipy import stats
# used in Hypothesis Testing to excute for example t-tests, chi-square tests, ANOVA, and more.
```

One Sample T-Test

Null Hypothesis (H0)

$H0: \mu = 5.1$

The sample mean sepal_length is equal to the 5.1 => sepal_length=5.1

Alternative Hypothesis (Ha or H1)

$H1: \mu != 5.1$

The sample mean sepal_length is not equal to the 5.1 "complement of H0" => sepal_length != 5.1

Without using scipy

[25]:
```
# sample of sepal_Lengths from the Iris dataset
sample = dataset[dataset["species"] == "Iris-setosa"]["sepal_length"] # Hypothes here that all Iris-setosa specie lenght = 5.2

hypothetical_value = 5.1  # Hypothetical mean
# to test whether the mean sepal_Length of the Iris-setosa species is different from this hypothetical value

# Calculate sample statistics
sample_mean = np.mean(sample)
sample_std = np.std(sample, ddof=1)
sample_size = len(sample)
# Calculate the t_statistic using eqation  t_statistic =   x̄ − μ 0 / (s/√n)
t_statistic_manual = (sample_mean - hypothetical_value) / (sample_std / np.sqrt(sample_size))
# Calculate the p-value using eqation  p-value = 2 × CDF(−|t|,df)
p_value_manual = 2 * stats.t.cdf(t_statistic_manual, sample_size - 1)

# Output results
print("Sample Mean:", sample_mean)
print("Sample Standard Deviation:", sample_std)
print("Sample Size:", sample_size)
print("T statistic:", t_statistic_manual)
print("P-value:", p_value_manual)
```

```
Sample Mean: 5.006
Sample Standard Deviation: 0.3524896872134512
Sample Size: 50
T statistic: -1.8856732506697746
P-value: 0.06527445885090737
```

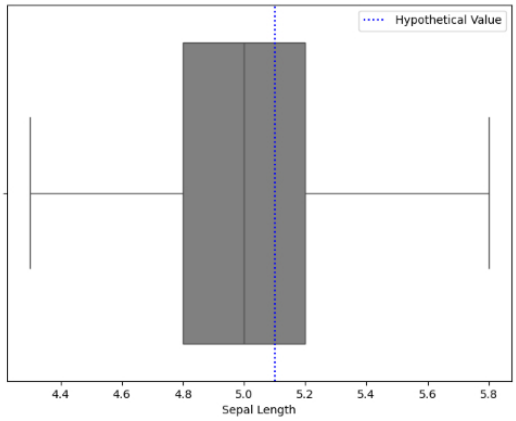
With using scipy library

[26]:
```
# t-test on sample directly using ttest_1samp on scipy library
t_statistic, p_value = stats.ttest_1samp(sample, hypothetical_value)

# ttest_1samp(sample,  popmean) Calculate the T-test for the mean of sample from all data
# test for the null hypothesis that the expected value /mean of a sample of independent observations a is equal to the given population mean
# t_statistic: difference between the sample mean and the hypothetical mean relative to the variation in the sample
# p_value: probability of obtaining the observed data assuming the null hypothesis is true
print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: -1.8856732506697453
P-value: 0.06527445885090742
```

[27]:
```
# Create a box plot
plt.figure(figsize=(8, 6)) # figure with a specified size
sns.boxplot(x=sample, color="gray") # box plot of the sample_data
plt.axvline(hypothetical_value, color="blue", linestyle="dotted" , label="Hypothetical Value" ) # add vertical line at the specified_value
plt.legend()
plt.xlabel("Sepal Length") # Label for the x-axis
plt.show()
```



[28]:
```
# Define the significance Level -- the probability of rejecting the null hypothesis when it the null hypothesis is true
significance_level = 0.05

# Compare the p-value to the significance level
if p_value < significance_level:
    print("Reject the null hypothesis. exsist difference.")
else:
    print("Fail to reject the null hypothesis. no different")
```

```
Fail to reject the null hypothesis. no different
```

One - Sample T-Test "left tail test"

Null Hypothesis (H0)

$H0: \mu \geq 2.5$

The sample mean sepal_width is greter than or equal to the 2.5 => sepal_width ≥ 2.5

Alternative Hypothesis (Ha or H1)

$H1: \mu < 2.5$

The sample mean sepal_width is less than 2.5 "complement of H0" => sepal_width < 2.5

[30]:
```
# sepal width data from Iris-virginica species
sample = dataset[dataset["species"] == "Iris-virginica"]["sepal_width"]

hypothetical_value = 2.5  # Hypothetical mean
```

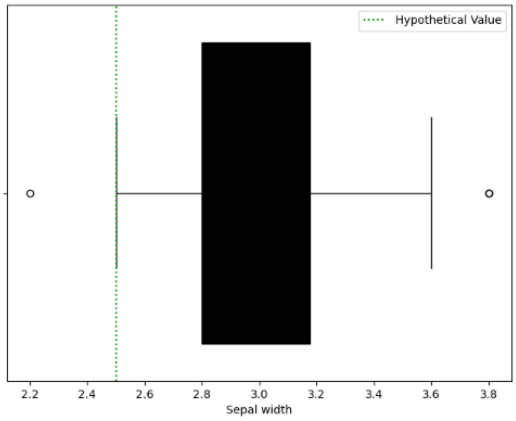
[31]:
```
#  t-test sample
t_statistic, p_value = stats.ttest_1samp(sample, hypothetical_value)

# p-value for a one-sided test "left tail test"
if t_statistic < 0:
    p_value /= 2  # Divide the p-value by 2 for a one-sided test

p_value , t_statistic
```

[31]:
```
(5.5137715014737266e-14, 10.392933587822478)
```

[32]:
```
# Create a box plot
plt.figure(figsize=(8, 6))
sns.boxplot(x=sample, color="black")
plt.axvline(hypothetical_value, color="green", linestyle="dotted" , label="Hypothetical Value" )
plt.legend()
plt.xlabel("Sepal width")
plt.show()
```



[33]:
```
# Define the significance level -- the probability of rejecting the null hypothesis when it the null hypothesis is true
significance_level = 0.05

# Compare the p-value to the significance level
if p_value < significance_level:
    print("Reject the null hypothesis. exsist difference.")
else:
    print("Fail to reject the null hypothesis. no different")
```

```
Reject the null hypothesis. exsist difference.
```

Two-Sample T-Test

Null Hypothesis (H0)

$H0: \mu1 = \mu2$

The means of the two groups['Iris-setosa' , 'Iris-versicolor'] are equal

Alternative Hypothesis (Ha or H1)

$H1: \mu1 != \mu2$

The means of the two groups['Iris-setosa' , 'Iris-versicolor'] are not equal

[35]:
```
#  samples for setosa and versicolor species
setosa_sample = dataset[dataset['species'] == 'Iris-setosa']['petal_length']
versicolor_sample = dataset[dataset['species'] == 'Iris-versicolor']['petal_length']

# Perform two-sample t-test
t_statistic, p_value = stats.ttest_ind(setosa_sample, versicolor_sample)
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Define the significance level -- the probability of rejecting the null hypothesis when it the null hypothesis is true
significance_level = 0.05

# Compare the p-value to the significance level
if p_value < significance_level:
    print("Reject the null hypothesis. exsist difference.")
else:
    print("Fail to reject the null hypothesis. no different")
```

```
T-statistic: -39.46866259397272
P-value: 5.717463758170621e-62
Reject the null hypothesis. exsist difference.
```

ANOVA TEST

Null Hypothesis (H0)

$H0: \mu$ Setosa = μ Versicolor = μ Virginica

There is no significant difference in mean petal lengths among the three species.

Alternative Hypothesis (Ha or H1)

$H1: \mu$ Setosa != μ Versicolor != μ Virginica

There is a significant difference in mean petal lengths among the three species.

[37]:
```
anova_result = stats.f_oneway(
    dataset[dataset['species'] == 'Iris-setosa']['petal_length'],
    dataset[dataset['species'] == 'Iris-versicolor']['petal_length'],
    dataset[dataset['species'] == 'Iris-virginica']['petal_length']
)

print("T-statistic:", anova_result.statistic)
print("P-value:", anova_result.pvalue)

# Define the significance level -- the probability of rejecting the null hypothesis when it the null hypothesis is true
significance_level = 0.05

# Compare the p-value to the significance level
if p_value < significance_level:
    print("Reject null hypothesis: at least one group mean is different.")
else:
    print("Fail to reject null hypothesis: all group means are the same")
```

```
T-statistic: 1179.0343277002194
P-value: 3.0519758018278374e-91
```

```
Reject null hypothesis: at least one group mean is different.
```

[ ]: