Course:CSCE 5215 Machine Learning

Professor: Zeenat Tariq

Week3 Day-2

In this activity we are going to understand about Data visualization and its significance.For this exercise, we will be iris dataset. As you know, It contains measurements of four features (sepal length, sepal width, petal length, and petal width) of three different species of Iris flowers (Setosa, Versicolor, and Virginica).

Also we are going to look at implementing different graphs using Matpplotlib and Seaborn.

### What is Data visualization?

Data visualization refers to the graphical representation of data to communicate information and insights effectively. It involves using visual elements such as charts, graphs, maps, and plots to represent data patterns, trends, and relationships. By presenting data visually, it becomes easier to understand complex information, identify patterns, and draw meaningful conclusions.

### Data Visualization in Python

Data visualization in Python can be accomplished using several libraries that provide a wide range of tools and functions. Some popular libraries for data visualization in Python include: Matplotlib, Seaborn, Plotly etc.

## 1.Matplotlib

It is an open source drawing library which supports rich drawing types Its is used to draw 2D and 3D graphics You can generate plots, histograms, bar charts and many other.

Lets load the iris dataset and extract each feature

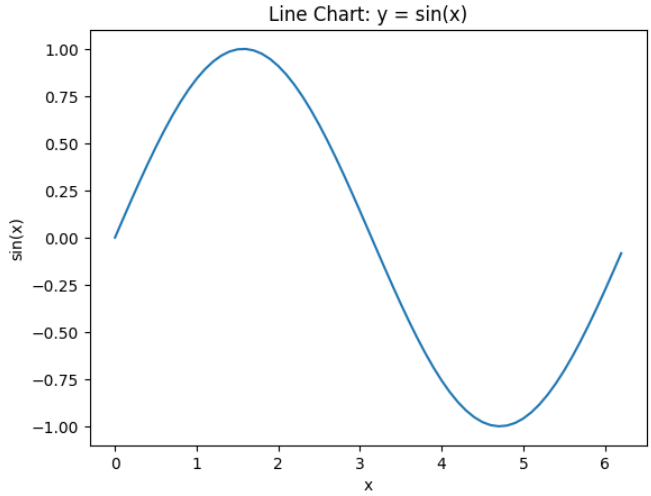
```
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Load the Iris dataset
iris = load_iris()


# Extract the features and target variable
X = iris.data
y = iris.target


sepal_length = X[:, 0]  #All Rows, 1st Columnn
sepal_width = X[:, 1]   #All Rows, 2st Columnn
petal_length = X[:, 2]  #All Rows, 3st Columnn
petal_width = X[:, 3]   #All Rows, 4st Columnn
sepal_length
```

```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8,
       4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. ,
       5. , 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4,
       5.1, 5. , 4.5, 4.4, 5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. , 7. , 6.4,
       6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5. , 5.9, 6. , 6.1, 5.6,
       6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7,
       6. , 5.7, 5.5, 5.5, 5.8, 6. , 5.4, 6. , 6.7, 6.3, 5.6, 5.5, 5.5,
       6.1, 5.8, 5. , 5.6, 5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8, 7.1, 6.3,
       6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5,
       7.7, 7.7, 6. , 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2,
       7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6. , 6.9, 6.7, 6.9, 5.8,
       6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9])
```

- First, let's look at creating a line chart for sin fuction for simplicity

```
# Generate x values from 0 to 2*pi with a step of 0.1
x = np.arange(0, 2 * np.pi, 0.1) # x is a numpy array from 0 -> 2 * pi with 0.1 increment for each element

# Calculate y values for sin(x)
y = np.sin(x)

# Create a line chart
plt.plot(x, y)

# Add labels and title
plt.xlabel('x')  # showing xlabel in the plot
plt.ylabel('sin(x)') # showing ylabel in the plot
plt.title('Line Chart: y = sin(x)') # showing title in the plot
```

```
Text(0.5, 1.0, 'Line Chart: y = sin(x)')
```



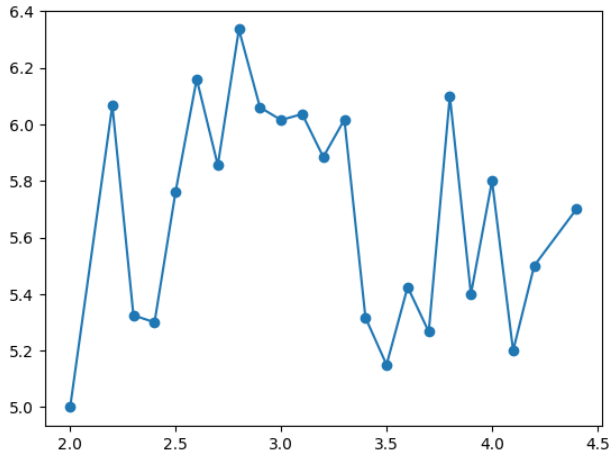

## Line Charts using matplotlib

A Line chart is a graph that represents information as a series of data points connected by a straight line.

1.A Calculate the average sepal length for each unique sepal width and plot Average Sepal lenghts vs uinque Sepal width

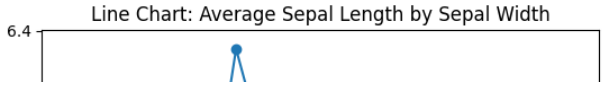
```
unique_widths = np.unique(sepal_width)  #Get unique speal width
avg_lengths = [np.mean(sepal_length[sepal_width == width]) for width in unique_widths] # Getting the array of mean of Sepal lengths for each un

# Create a line chart of average sepal length by sepal width
plt.plot(unique_widths, avg_lengths, marker='o', linestyle='-')
```

```
[<matplotlib.lines.Line2D at 0x7e455e8c1cc0>]
```




```
unique_widths = np.unique(sepal_width)  #Get unique speal width
avg_lengths = [np.mean(sepal_length[sepal_width == width]) for width in unique_widths] # Getting the array of mean of Sepal lengths for each un

# Create a line chart of average sepal length by sepal width
plt.plot(unique_widths, avg_lengths, marker='o', linestyle='-')

# Add labels and title
plt.xlabel('Sepal Width')
plt.ylabel('Average Sepal Length')
plt.title('Line Chart: Average Sepal Length by Sepal Width')
```

```
Text(0.5, 1.0, 'Line Chart: Average Sepal Length by Sepal Width')
```



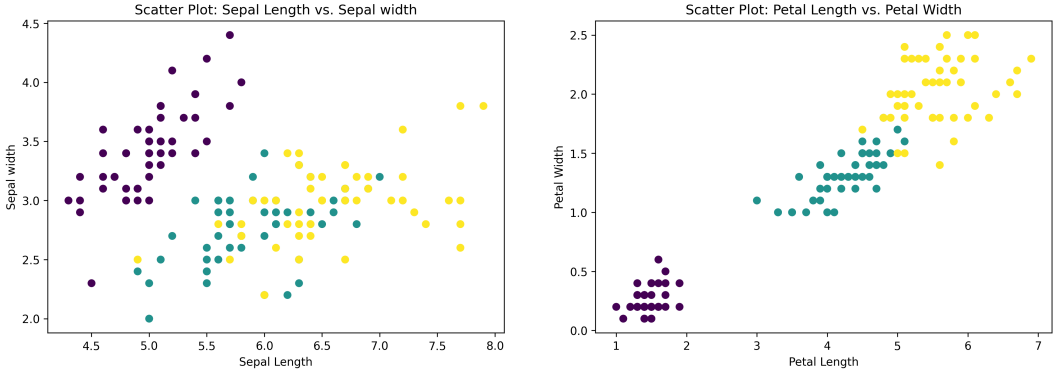

## Scatter Plot:

A scatter plot is a type of plot used to visualize the relationship between two numerical variables. It displays individual data points as dots on a graph, where the x-axis represents one variable and the y-axis represents the other variable. Scatter plots are useful for identifying patterns, trends, and the distribution of data points.

1.B Scatter plot Sepal Length vs. Sepal width & Petal Length vs. Petal Width

```
# Create a figure with two subplots
fig, axes = plt.subplots(1,2, figsize=(16,5), dpi=300)

# Scatter plot on the first subplot: Sepal Length vs. Sepal width
# c=y is a parameter in the plt.scatter() function that specifies the color of each data point based on the y variable.
axes[0].scatter(sepal_length,sepal_width,c=y)
#google cloud retains the assined values
#y=np.sin(x) got rewritten. So run y=np.target first then run this.

axes[0].set_xlabel('Sepal Length')
axes[0].set_ylabel('Sepal width')
axes[0].set_title('Scatter Plot: Sepal Length vs. Sepal width')


# Scatter plot on the second subplot: Petal Length vs. Petal Width
axes[1].scatter(petal_length,petal_width,c=y)
axes[1].set_xlabel('Petal Length')
axes[1].set_ylabel('Petal Width')
axes[1].set_title('Scatter Plot: Petal Length vs. Petal Width')
```

```
Text(0.5, 1.0, 'Scatter Plot: Petal Length vs. Petal Width')
```




```
y.shape, X.shape
```

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

## Bar Chart

is a type of chart that presents categorical data with rectangular bars, where the length of each bar represents the quantity or value associated with that category. Bar plots are useful for comparing different categories or showing the distribution of a variable across categories.

1.C Create a bat chart of Each species vs Species Count

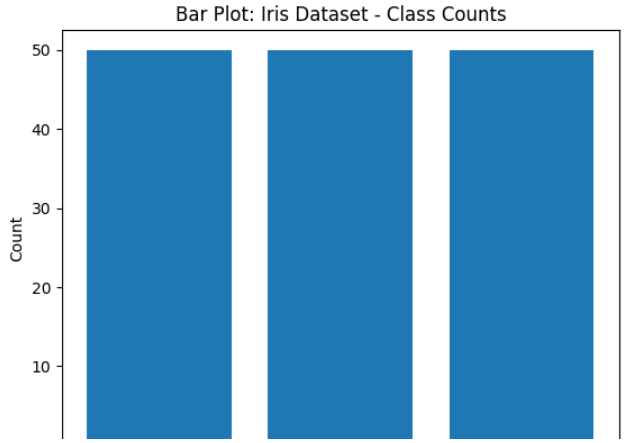
```
unique_species, species_counts = np.unique(y, return_counts=True) # Extract unique species and count
```

```
unique_species, species_counts
```

```
(array([0, 1, 2]), array([50, 50, 50]))
```

```
# Create a bar plot
plt.bar(unique_species, species_counts)

# Add labels and title
plt.xlabel('Target Class')
plt.ylabel('Count')
plt.title('Bar Plot: Iris Dataset - Class Counts')
```

```
Text(0.5, 1.0, 'Bar Plot: Iris Dataset - Class Counts')
```




## Histogram

A histogram is a graphical representation of the distribution of a dataset. It displays the frequencies or counts of data points falling into different intervals, known as bins. In a histogram, the x-axis represents the range of values in the dataset, divided into bins, while the y-axis represents the frequency or count of data points falling into each bin.

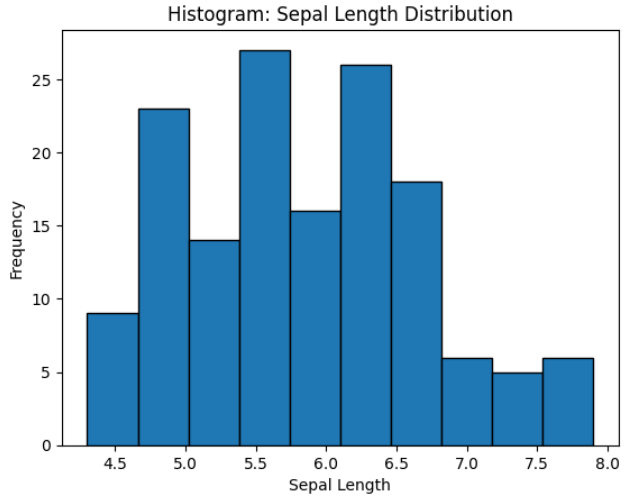
1.D Create histogram for Sepal Length

```
sepal_length
```

```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8,
       4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. ,
       5. , 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4,
       5.1, 5. , 4.5, 4.4, 5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. , 7. , 6.4,
       6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5. , 5.9, 6. , 6.1, 5.6,
       6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7,
       6. , 5.7, 5.5, 5.5, 5.8, 6. , 5.4, 6. , 6.7, 6.3, 5.6, 5.5, 5.5,
       6.1, 5.8, 5. , 5.6, 5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8, 7.1, 6.3,
       6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5,
       7.7, 7.7, 6. , 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2,
       7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6. , 6.9, 6.7, 6.9, 5.8,
       6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9])
```

```
# Create a histogram of sepal length
plt.hist(sepal_length, bins=10, edgecolor='black')

# Add labels and title
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.title('Histogram: Sepal Length Distribution')
```

```
Text(0.5, 1.0, 'Histogram: Sepal Length Distribution')
```




How to define number of bins

1 - square root rule

2 - auto

```
np.round(np.sqrt(len(sepal_length)))
```

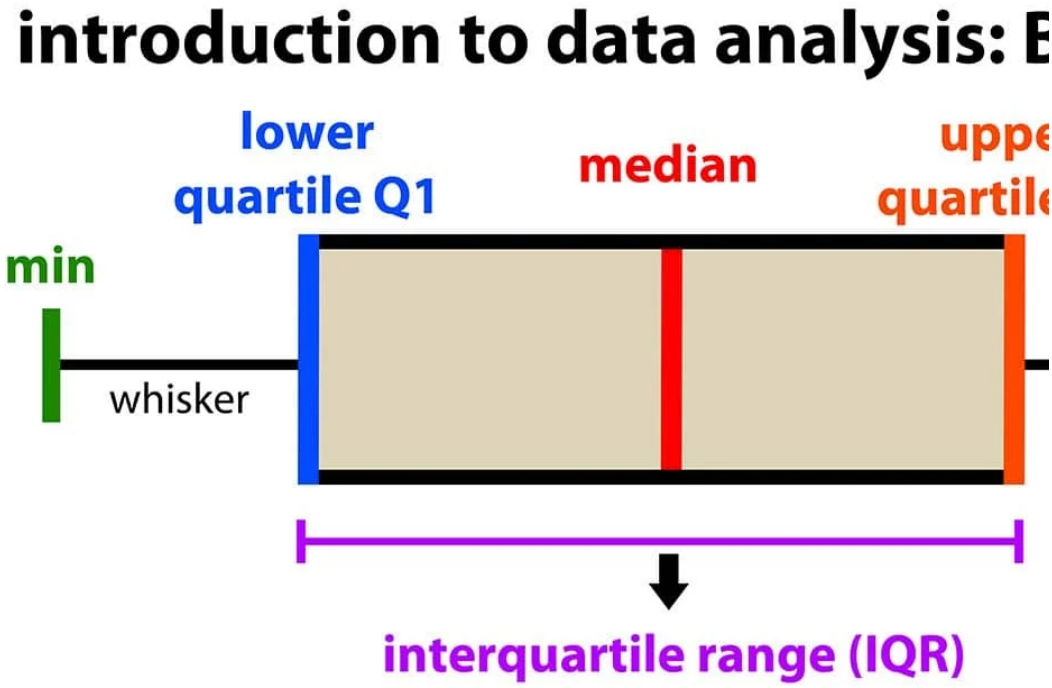
```
12.0
```

Boxplot

Boxplots visually show the distribution of numerical data and skewness by displaying the data quartiles (or percentiles) and averages.

It shows: minimum score, first (lower) quartile, median, third (upper) quartile, and maximum score.

```
from IPython.display import Image
Image(url="https://www.simplypsychology.org/wp-content/uploads/box-whisker-plot.jpg")
```




```
colors = ["blue", "red", "green"]
df = pd.DataFrame(
    data=np.c_[X, y], columns=iris["feature_names"] + ["target"]
)
df.boxplot(by="target", layout=(2, 2), figsize=(10, 10))
plt.show()
```

```
import seaborn as sns
iris_test_df = sns.load_dataset("iris")           #Load the data in really nice way
iris_test_df[["sepal_length", "species"]].head()
```

| | sepal_length | species |
|---|---|---|
| 0 | 5.1 | setosa |
| 1 | 4.9 | setosa |
| 2 | 4.7 | setosa |
| 3 | 4.6 | setosa |
| 4 | 5.0 | setosa |



```
iris_test_df["species"].unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
print(min(iris_test_df.loc[iris_test_df["species"] == "virginica"]["sepal_length"]))
print(max(iris_test_df.loc[iris_test_df["species"] == "virginica"]["sepal_length"]))
```

```
4.9
7.9
```



```
species_df = iris_test_df.loc[iris_test_df["species"] == "virginica"]["sepal_length"]
median = np.median(species_df)
print (f"Median: {median}")
upper_quartile = np.percentile(species_df, 75)
print (f"Upper quartile: {upper_quartile}")
lower_quartile = np.percentile(species_df, 25)
print (f"Lower quartile: {lower_quartile}")

iqr = upper_quartile - lower_quartile
upper_whisker = species_df[species_df<=upper_quartile+1.5*iqr].max()      #1.5 is changable. If you make too low -> you cannot see outliers, to
print (f"Upper whisker: {upper_whisker}")
lower_whisker = species_df[species_df>=lower_quartile-1.5*iqr].min()
print (f"Lower whisker: {lower_whisker}")
```

```
Median: 6.5
Upper quartile: 6.9
Lower quartile: 6.225
Upper whisker: 7.9
Lower whisker: 5.6
```

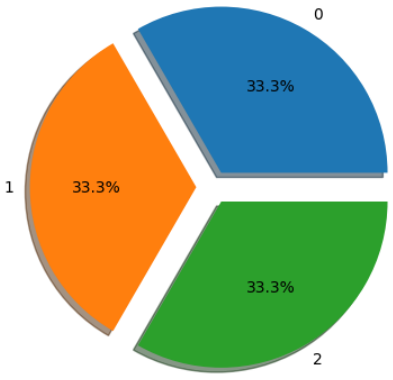
Pie

Pie charts are often used in business. It is helpful showing the relationship of parts to the whole when there are a small number of levels.

```
np.unique(y, return_counts=True)
```

```
(array([0, 1, 2]), array([50, 50, 50]))
```

```
uniq = np.unique(y, return_counts=True)[0]
count = np.unique(y, return_counts=True)[1]

plt.pie(count, labels = uniq, explode=[0.1,0.1,0.1],autopct='%1.1f%%',shadow=True)
# plt.pie(count, labels = uniq, explode=[0.1,0.5,0.1],autopct='%1.1f%%',shadow=True)

plt.show()
```

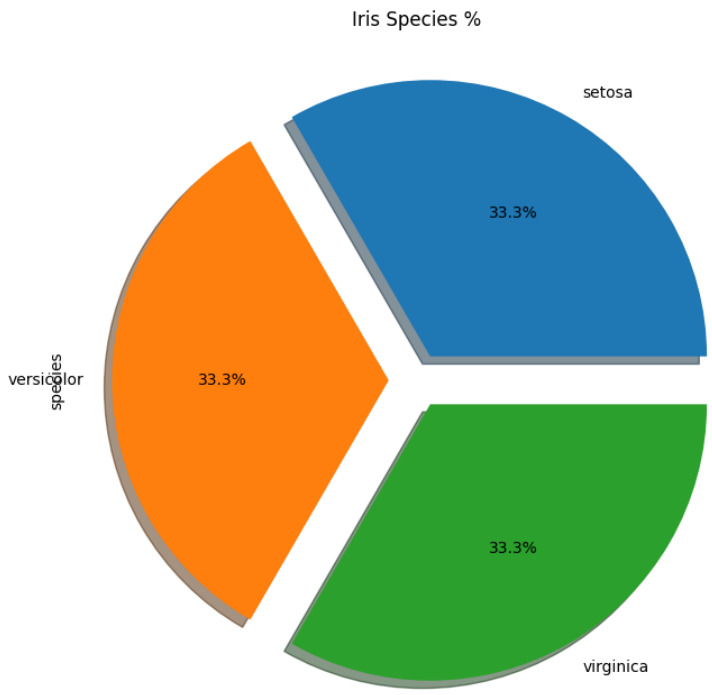



```
iris_sns = sns.load_dataset("iris")
iris_sns
```

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

```
iris_sns['species'].value_counts().plot.pie(explode=[0.1,0.1,0.1],autopct='%1.1f%%',shadow=True,figsize=(10,8))
plt.title("Iris Species %")
plt.show()
```




**2. Seaborn Library**

Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and visually appealing statistical graphics. Seaborn is particularly useful for exploratory data analysis and for creating complex visualizations with minimal code.

2.A Create Bar chart for Average sepal length of each species

```
# Calculate the average sepal length for each species
average_sepal_length = iris_sns.groupby("species")["sepal_length"].mean()
```

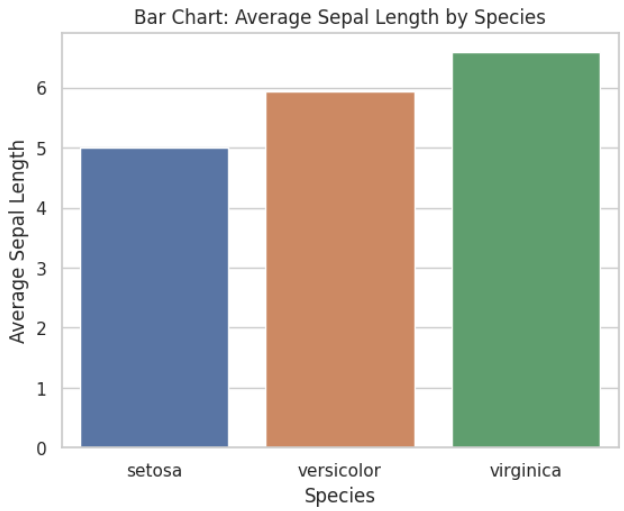
```
average_sepal_length
```

```
species
setosa        5.006
versicolor    5.936
virginica     6.588
Name: sepal_length, dtype: float64
```

```
# Set the Seaborn style
sns.set(style="whitegrid")

# Create a bar chart using Seaborn
sns.barplot(x=average_sepal_length.index, y=average_sepal_length.values)

# Set labels and title
plt.xlabel("Species")
```

```
plt.ylabel("Average Sepal Length")
plt.title("Bar Chart: Average Sepal Length by Species")
```

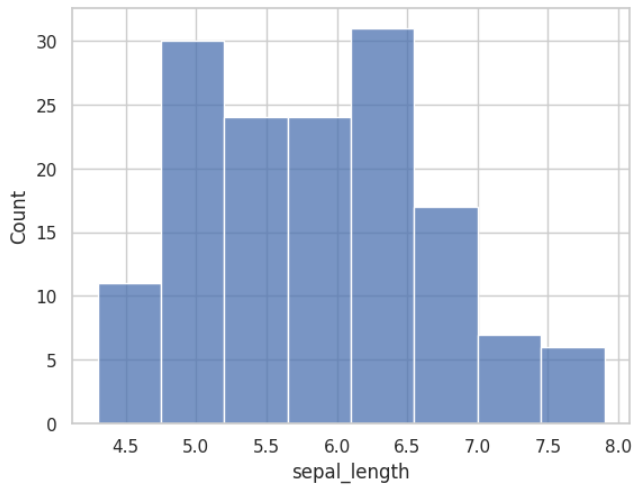
```
Text(0.5, 1.0, 'Bar Chart: Average Sepal Length by Species')
```




2.B Histogram plot of Sepal Length using Seaborn

```
sns.histplot(sepal_length,bins=8)
plt.xlabel("sepal_length")
```

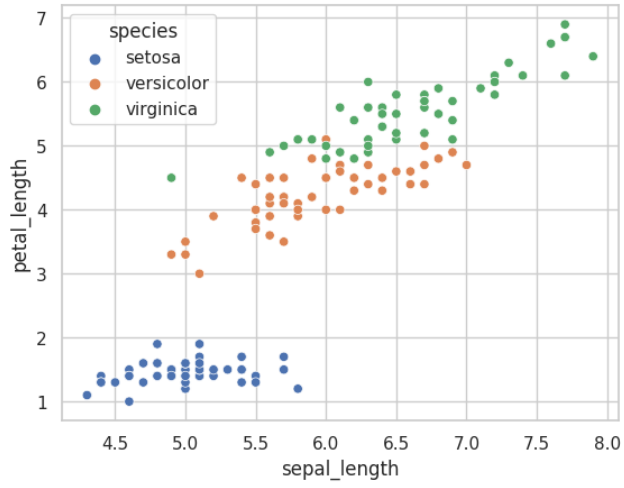
```
Text(0.5, 0, 'sepal_length')
```




2.C Create Scatter plot for sepal length vs petal length

```
sns.scatterplot(x='sepal_length', y='petal_length',data=iris_sns,hue='species')    #"hue" = label name
```

```
<Axes: xlabel='sepal_length', ylabel='petal_length'>
```



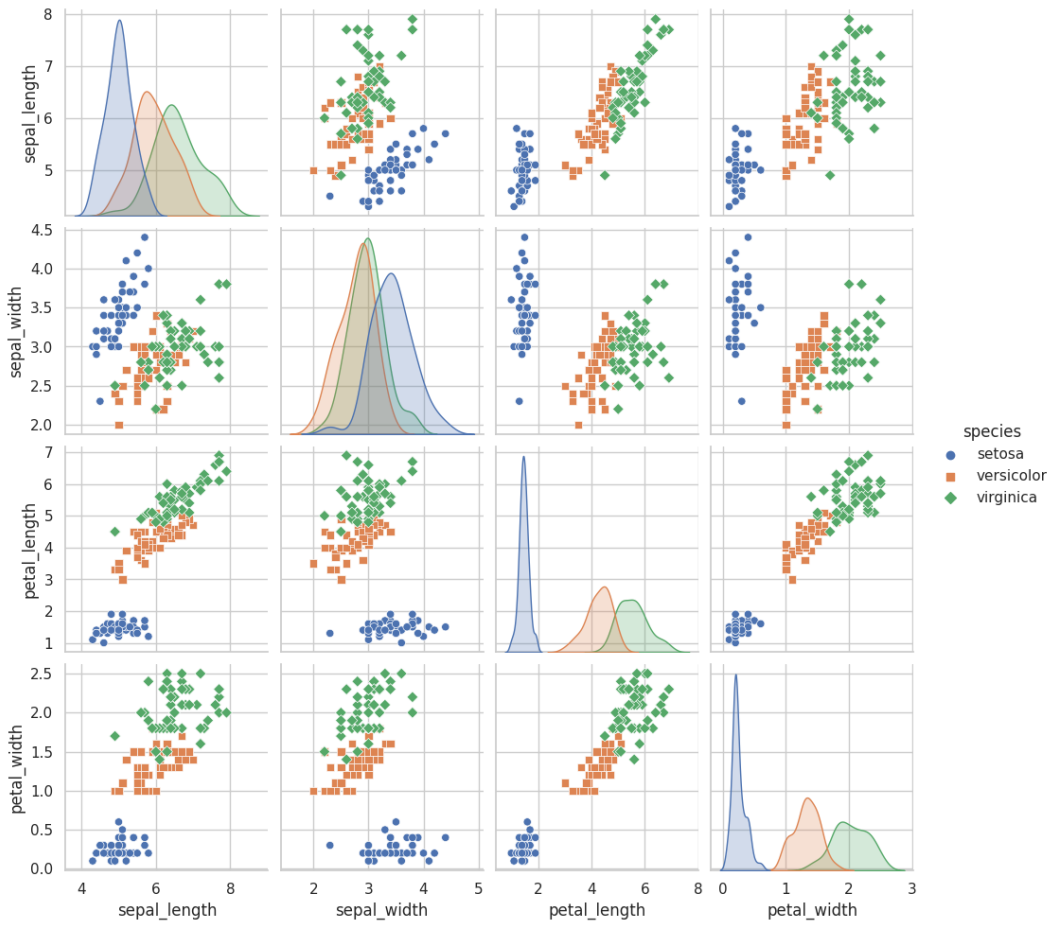

## Pair Plot Seaborn

A pair plot is a type of plot in seaborn that allows you to visualize pairwise relationships between variables in a dataset. It creates a grid of scatter plots and histograms, showing the relationship between each pair of variables.

2.D Create a pair plot for iris feature set

```
# hue helps to use a different color
sns.pairplot(iris_sns,hue='species', markers= ["o", "s", "D"], diag_kind="auto")    #markers : Gives different types of representaions like cir
```

```
<seaborn.axisgrid.PairGrid at 0x7e455b1a1060>
```




## Box Plot using Seaborn

A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset. It displays a summary of the data's central tendency, spread, and outliers.

- The middle line in the box of the represents the median
- Upper horizontal line represents maximum sepal width and lower horizontal line represents minimum sepal width
- The Dots the boxes are the outliers.
- The Extend vertical lines for each box are whiskers

2.E Create box plot for speal width for each species using seaborn

```
sns.boxplot(x='species',y='sepal_width',data=iris_sns)
```

```
<Axes: xlabel='species', ylabel='sepal_width'>
```


## HeatMap

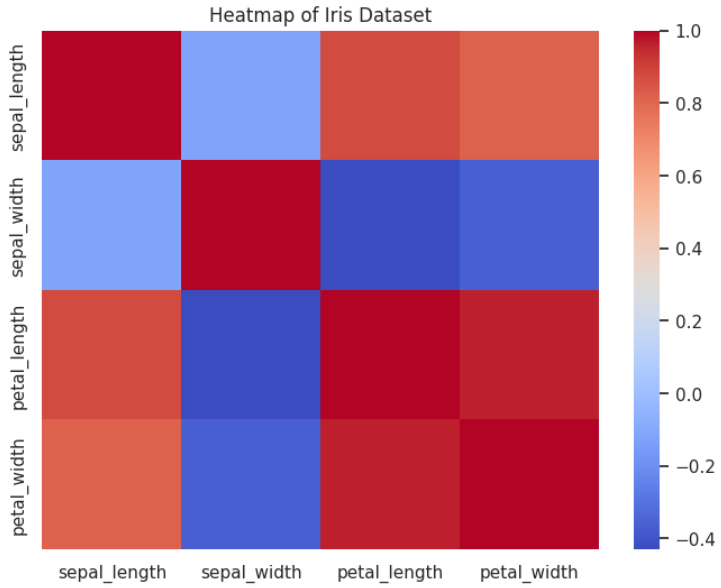
A heat map is a graphical representation of data in which values are displayed as colors within a two-dimensional grid. It provides a visual summary of the data, allowing patterns, relationships, and variations to be easily identified

2.F Create a heat map for iris datset using seaborn



```
# Calculate the correlation matrix
corr_matrix = iris_sns.corr()

# Create a heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, cmap='coolwarm')

# Set title
plt.title('Heatmap of Iris Dataset')
```

```
<ipython-input-37-931f7cda0636>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  corr_matrix = iris_sns.corr()
Text(0.5, 1.0, 'Heatmap of Iris Dataset')
```




Swarmplot

a type of scatter plot that is used for representing categorical values

**It is not advisable to use this type of graph when the sample size is large**

```
sns.set(style="whitegrid")
fig=plt.gcf()
fig.set_size_inches(10,7)
fig = sns.swarmplot(x="species", y="petal_length", data=iris_sns)
```


```
print(min(iris_test_df.loc[iris_test_df["species"] == "virginica"]["petal_length"]))
print(max(iris_test_df.loc[iris_test_df["species"] == "virginica"]["petal_length"]))
```

```
4.5
6.9
```



```
df = sns.load_dataset("penguins")
df
```

| | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 3750.0 | Male |
| **1** | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 3800.0 | Female |
| **2** | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 3250.0 | Female |
| **3** | Adelie | Torgersen | NaN | NaN | NaN | NaN | NaN |
| **4** | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 3450.0 | Female |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **339** | Gentoo | Biscoe | NaN | NaN | NaN | NaN | NaN |
| **340** | Gentoo | Biscoe | 46.8 | 14.3 | 215.0 | 4850.0 | Female |
| **341** | Gentoo | Biscoe | 50.4 | 15.7 | 222.0 | 5750.0 | Male |
| **342** | Gentoo | Biscoe | 45.2 | 14.8 | 212.0 | 5200.0 | Female |
| **343** | Gentoo | Biscoe | 49.9 | 16.1 | 213.0 | 5400.0 | Male |

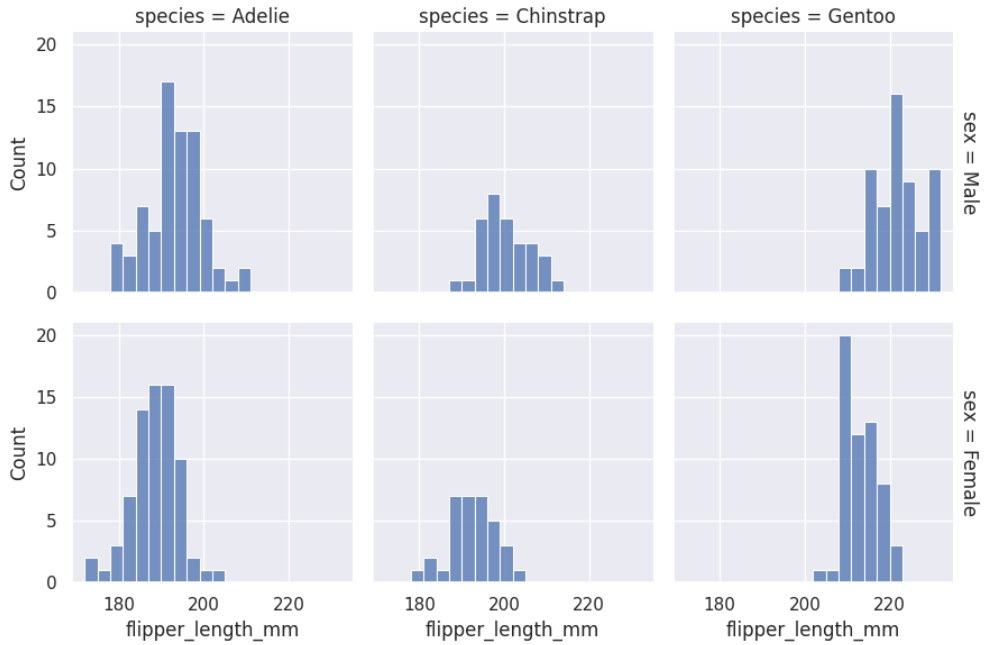
344 rows × 7 columns

Displot

Used to represent data in histogram form, where the data distribution of one variable will be shown against another variable.

```
sns.set_theme(style="darkgrid")
df_penguins = sns.load_dataset("penguins")
sns.displot(
    df_penguins, x="flipper_length_mm", col="species", row="sex",
    binwidth=3, height=3, facet_kws=dict(margin_titles=True),
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e455b6c89d0>
```




```
test_df_penguins = df_penguins[["flipper_length_mm", "sex", "species"]]
test_df_penguins
```

| | flipper_length_mm | sex | species |
|---|---|---|---|
| **0** | 181.0 | Male | Adelie |
| **1** | 186.0 | Female | Adelie |
| **2** | 195.0 | Female | Adelie |
| **3** | NaN | NaN | Adelie |
| **4** | 193.0 | Female | Adelie |
| **...** | ... | ... | ... |

```
test_df_penguins.loc[(test_df_penguins["species"] == "Adelie")  & (test_df_penguins["sex"] == "Female")]["flipper_length_mm"].value_counts()
```

```
187.0    9
190.0    8
185.0    6
191.0    6
186.0    5
193.0    5
195.0    5
181.0    4
189.0    4
188.0    3
184.0    3
192.0    2
178.0    2
182.0    2
180.0    1
172.0    1
196.0    1
174.0    1
202.0    1
183.0    1
199.0    1
198.0    1
176.0    1
Name: flipper_length_mm, dtype: int64
```

Plotly

It is a free and open-source graphing library for Python

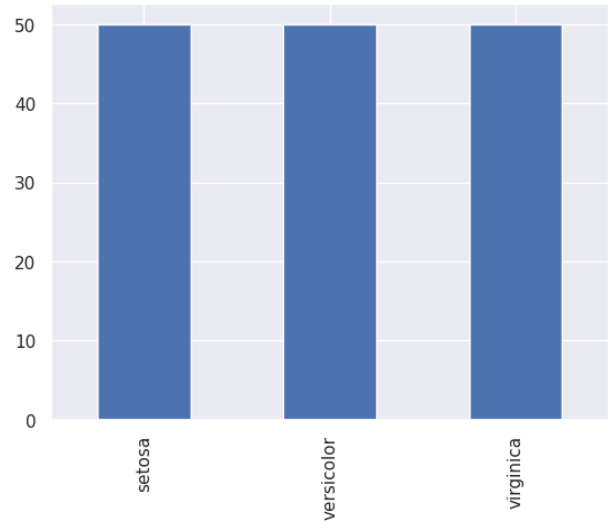
```
!pip install plotly_express
```

```
Collecting plotly_express
  Downloading plotly_express-0.4.1-py2.py3-none-any.whl (2.9 kB)
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (1.5.3)
Requirement already satisfied: plotly>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (5.15.0)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (0.14.0)
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (1.10.1)
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (0.5.3)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.10/dist-packages (from plotly_express) (1.23.5)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.20.0->plotly_express) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.20.0->plotly_express) (2023.3.post1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5->plotly_express) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=4.1.0->plotly_express) (8.2.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly>=4.1.0->plotly_express) (23.1)
Installing collected packages: plotly_express
Successfully installed plotly_express-0.4.1
```

```
# import plotly
import plotly_express as px
```

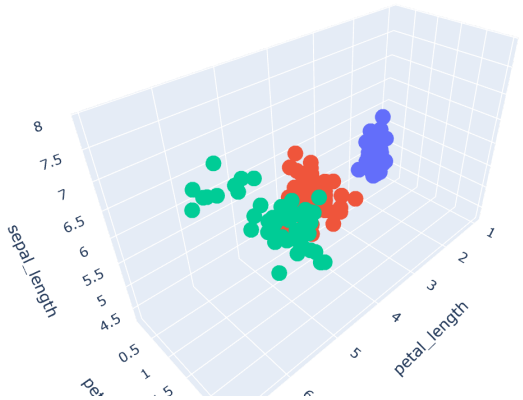
```
# Count the number of values
iris_sns['species'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
#  Scatter a 3D plot
px.scatter_3d(iris_sns, x="petal_length", y="petal_width", z="sepal_length",
```

```
                color="species")
```




```
# to create a table from Dataframe
import plotly.figure_factory as ff
# Used to plot the result
import plotly.offline as py


table = ff.create_table(iris_sns)
py.iplot(table, filename='jupyter-tabl')      #change filename as you wish, it doesn't matter
```

```
import plotly.graph_objs as go
from plotly import tools
```

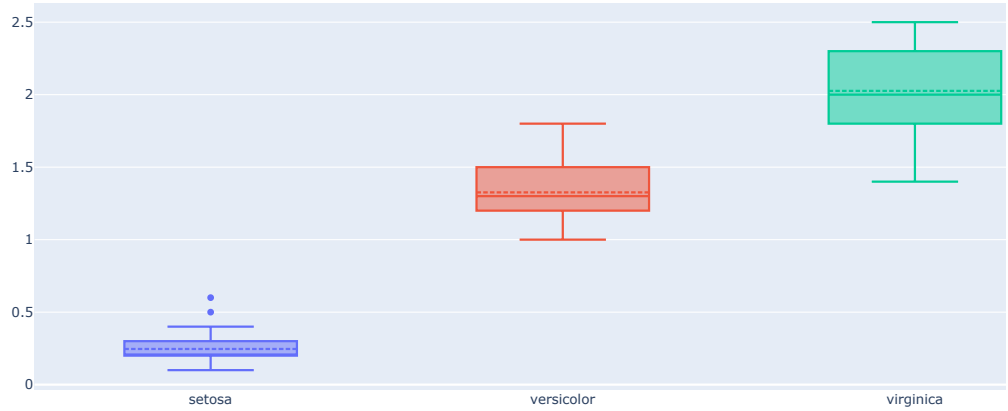
```
df
```

| | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 3750.0 | Male |
| **1** | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 3800.0 | Female |
| **2** | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 3250.0 | Female |
| **3** | Adelie | Torgersen | NaN | NaN | NaN | NaN | NaN |
| **4** | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 3450.0 | Female |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **339** | Gentoo | Biscoe | NaN | NaN | NaN | NaN | NaN |
| **340** | Gentoo | Biscoe | 46.8 | 14.3 | 215.0 | 4850.0 | Female |
| **341** | Gentoo | Biscoe | 50.4 | 15.7 | 222.0 | 5750.0 | Male |
| **342** | Gentoo | Biscoe | 45.2 | 14.8 | 212.0 | 5200.0 | Female |
| **343** | Gentoo | Biscoe | 49.9 | 16.1 | 213.0 | 5400.0 | Male |

344 rows × 7 columns

5.8 4.0 1.2 0.2 s

```
trace0 = go.Box(y=iris_sns['petal_width'][iris_sns['species'] == 'setosa'],boxmean=True, name = 'setosa')

trace1 = go.Box(y=iris_sns['petal_width'][iris_sns['species'] == 'versicolor'],boxmean=True, name = 'versicolor')

trace2 = go.Box(y=iris_sns['petal_width'][iris_sns['species'] == 'virginica'],boxmean=True, name = 'virginica')

data = [trace0, trace1, trace2]
py.iplot(data)
```




5.1 3.4 1.5 0.2 s

Practice

1 - Load a dataset from sklearn

2 - Display the data using Seaborn and Plotly

3 - Display Pie chart (labels or target values)

4 - Display Histogram

5 - Display swarmplot plot

6 - Scatter a 3D plot

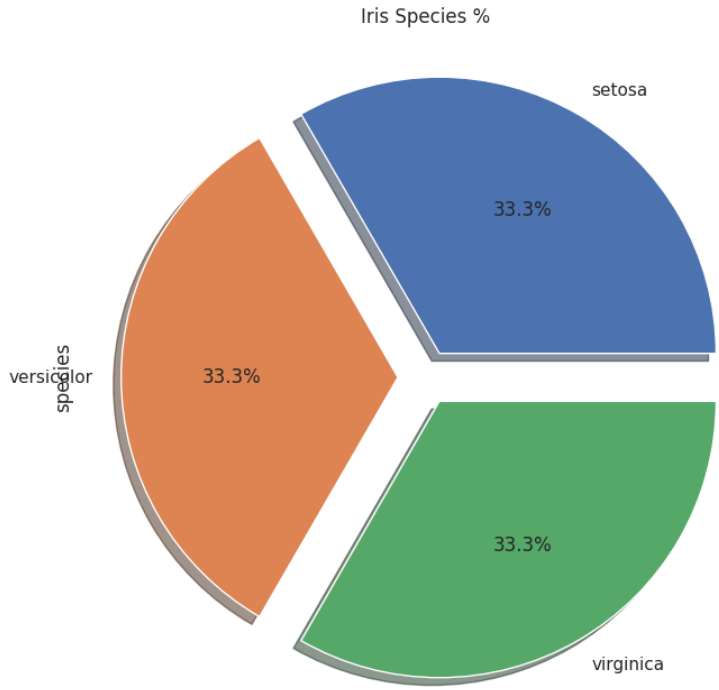
```
# 1
# Load the Iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
def_sns = sns.load_dataset("iris")
```

6.9 3.1 4.9 1.5 v

```
def_sns
```

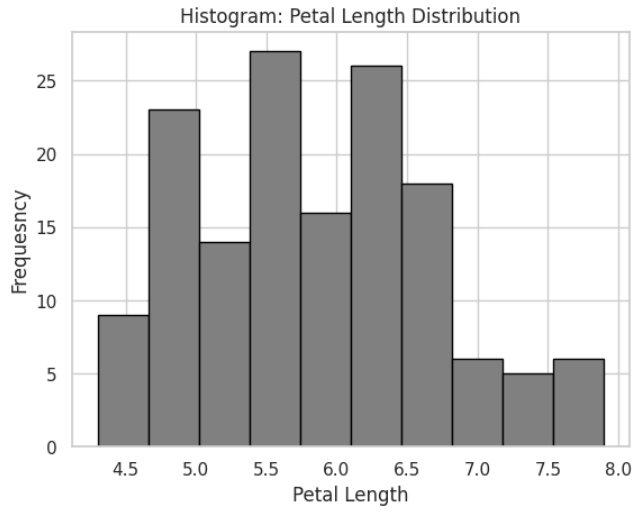
| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |

```
# 3
def_sns['species'].value_counts().plot.pie(explode=[0.1,0.1,0.1],autopct='%1.1f%%',shadow=True,figsize=(10,8))
plt.title("Iris Species %")
plt.show()
```



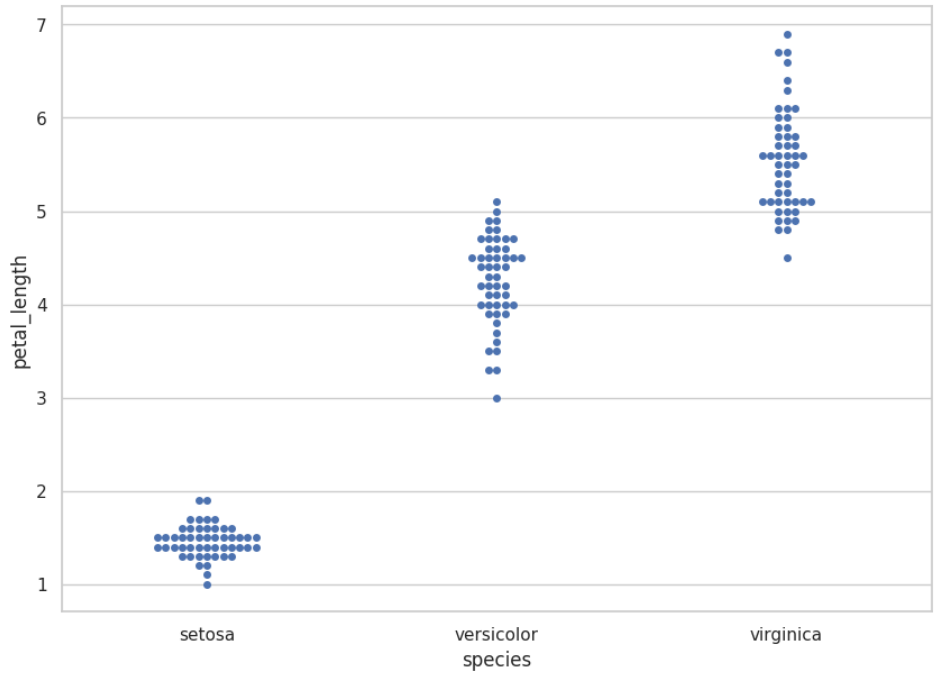

```
# 4
petal_length = X[:, 0]
plt.hist(petal_length, bins=10, edgecolor='black', color="gray")

# Add labels and title
plt.xlabel('Petal Length')
plt.ylabel('Frequesncy')
plt.title('Histogram: Petal Length Distribution')
```

```
Text(0.5, 1.0, 'Histogram: Petal Length Distribution')
```



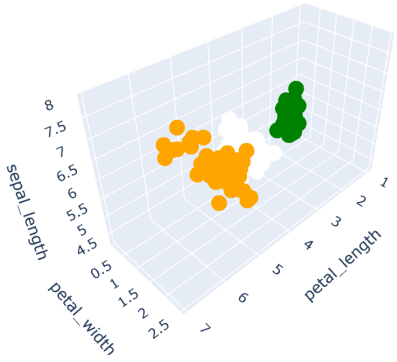

```
# 5
sns.set(style="whitegrid")
fig=plt.gcf()
```

```
fig.set_size_inches(10,7)
fig = sns.swarmplot(x="species", y="petal_length", data=def_sns)
```




```
6.7 3.3 5.7 2.5 v
```

```
# 6
px.scatter_3d(def_sns, x="petal_length", y="petal_width", z="sepal_length",
              color="species", color_discrete_map = {"setosa": "green", "versicolor": "white", "virginica":"orange"})
```




Please explain in detail understatnding of the entire activity in **atleast 200 words.**

This activity is about data visualization. We used iris data set in this activity.

## Data Visualization:

Data visualization is a method of displaying data in a way that makes its relationships and insights more understandable. Data visualization simplifies the process of understanding the insights of the relationship between variables since data is presented as graphs, scatter plots, pie charts, histograms, and other visual representations. It is an important tool for Exploratory Data Analysis(EDA). MLActivity3

### Matplotlib

Matplotlib is a library that is used to represent 2D and 3D graphics such as plots, histograms, bar charts, etc We created a Line Graph for Ya = sin(x) where x is iris data, and y is target.

The Scatter Plot shows the relation between variables

A bar graph, as the name suggests represents data in the form of rectangular bars. It is useful for comparing different categories of data. As seen in the above activity, we represented the Iris dataset class counts in the form of a bar graph.

A histogram is used for data distribution. In the activity, we represented Sepal Length Distribution in the form of a histogram.

Boxplots use the data's quartiles (or percentiles) and averages to show the distribution of numerical data and skewness. It displays the minimum score, first-quartile value, median, third-quartile value, and highest score. It is easy to see outliers in boxplots.

A pie chart is a circular shaped graph. They are frequently used to display sample data, with data points grouped into a variety of categories.

## Seaborn

Seaborn data visualization is built on top of Matplotlib, but Seaborn is more fancy in visualization. It is used to create more complex data visualization. Pair Plot Seaborn, Box Plot, HeatMap, Swarmplot, Display, and Plotly are some of the data visualization representations in Seaborn. Swamplot is used to represent categorical data where the sample size is not large. Plotly generated interactive 3D scatter plots. It doesn't come with python, we need to explicity download it using pip command.