AI-02-MLB14

1)SUMAN J 2)KARTIK KASHID 3)GASTEENA PESS

IRIS Data Analysis

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
iris = pd.read_csv("Iris.csv")          #importing dataset into dataframe "iris"
```

In [3]:
```
iris.head()
```

Out[3]:

	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 [4]:
```
iris.tail()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
iris.columns
```

Out[5]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [6]:
```
iris.axes
```

Out[6]:
```
[RangeIndex(start=0, stop=150, step=1),
 Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
        'Species'],
       dtype='object')]
```

In [7]:
```
iris.describe()
```

Out[7]:

	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

In [8]:
```
iris.info
```

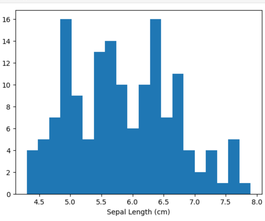
Out[8]:
```
<bound method DataFrame.info of         Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0        1            5.1           3.5            1.4           0.2
1        2            4.9           3.0            1.4           0.2
2        3            4.7           3.2            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]>
```

Data Pre - Processing

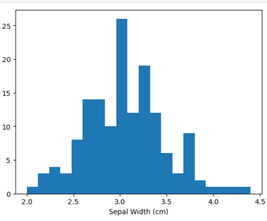
In [9]:
```
iris.isnull().sum()                #to check whether there's any null values
```

Out[9]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

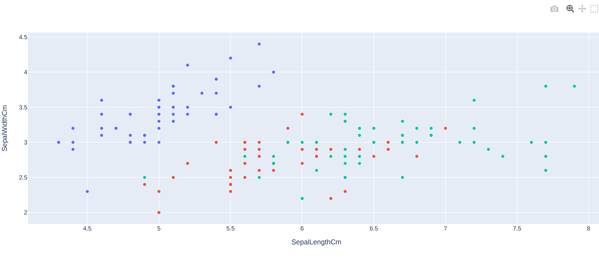
In [10]:
```
#Histogram 1

plt.hist(iris["SepalLengthCm"],bins=20)
plt.xlabel("Sepal Length (cm)")
plt.ylabel("frequency")
plt.show()
```



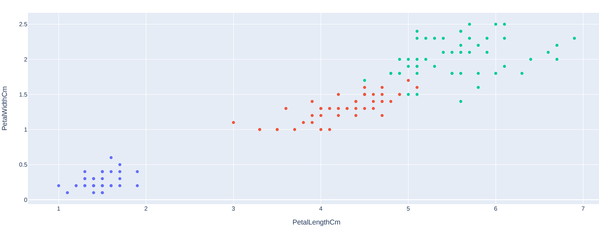
In [11]:
```
#Histogram 2#

plt.hist(iris["SepalWidthCm"],bins=20)
plt.xlabel("Sepal Width (cm)")
plt.ylabel("frequency")
plt.show()
```



In [12]:
```
import plotly.express as px
figure1 = px.scatter(iris,x="SepalLengthCm",y="SepalWidthCm", color = "Species")
figure1.show()
```



In [13]:
```
figure2 = px.scatter(iris,x="PetalLengthCm",y="PetalWidthCm", color = "Species")
figure2.show()
```



Iris Classification Model

In [14]:
```
#Splitting data into independant and dependant variables

x = iris.drop("Species",axis=1)
y = iris["Species"]
print(x.shape)
print(y.shape)
```

```
(150, 5)
(150,)
```

In [15]:
```
# Training the data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state = 42 )
```

In [16]:
```
# applying KNN classifier

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(x_train, y_train)
```

Out[16]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [17]:
```
abc=knn.predict(x_test)
print(abc)
```

```
['Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa']
C:\Users\kartik\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning:

Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: t
he default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to
True or False to avoid this warning.
```

In [18]:
```
from sklearn.metrics import accuracy_score        # ACCURACY OF MACHINE LEARNING MODEL

accuracy = accuracy_score(y_test,abc)
print(accuracy)
```

```
1.0
```

In [ ]:
```

```