In [1]:
```
"""Task 1: To train a machine learning model that can learn from the measurements of the
            iris species and classify them into three different species(setosa, versicolor, and virginica)""


import numpy as np   #numpy is used for linear algebra
import pandas as pd   #pandas is used for operations on csv files
import matplotlib as plt  #matplotlib is used for creating pie charts, histogram, etc
import seaborn as sns   #seaborn is used for visualisation of data. It is build on top of matplotlib
```

In [2]:
```
df=pd.read_csv('/Users/nitish/internship_task/Iris.csv')   #reading or importing the file
```

In [3]:
```
df.head(15)  #Here we are displaying first 15 rows of our Iris.csv file to check if file has successfully re
```

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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa

In [4]:
```
df=df.drop(columns=['Id'])  #we will drop 'Id' column as we already have default index for our data
```

In [5]:
```
df
```

Out[5]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [6]:
```
df.count()   #here we calculated total flowers in each columns
```

Out[6]:
```
SepalLengthCm    150
SepalWidthCm     150
PetalLengthCm    150
PetalWidthCm     150
Species          150
dtype: int64
```

In [7]:
```
df.Species.value_counts()   #here we found out the number of flowers in each species
```

Out[7]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [8]:
```
df.shape  #using .shape we can find number of rows and columns in the given dataset
```

Out[8]:
```
(150, 5)
```

In [9]:
```
df["Species"].replace({"Iris-setosa": 2, "Iris-versicolor": 3, "Iris-virginica": 4}, inplace=True)

#We will replace our three unique species with numbers to make calculations and representation easy
```

In [10]:
```
df.head(10)   #we are checking if values are replaced
```

Out[10]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	2
1	4.9	3.0	1.4	0.2	2
2	4.7	3.2	1.3	0.2	2
3	4.6	3.1	1.5	0.2	2
4	5.0	3.6	1.4	0.2	2
5	5.4	3.9	1.7	0.4	2
6	4.6	3.4	1.4	0.3	2
7	5.0	3.4	1.5	0.2	2
8	4.4	2.9	1.4	0.2	2
9	4.9	3.1	1.5	0.1	2

In [11]:
```
x=pd.DataFrame(df,columns=["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]).values
y=df.Species.values.reshape(150,1)   #Here we are creating two arrays 'x' for flower dimensions and 'y' for
```

In [12]:
```
x
```

```
[4.8, 3. , 1.4, 0.1],
[4.3, 3. , 1.1, 0.1],
[5.8, 4. , 1.2, 0.2],
[5.7, 4.4, 1.5, 0.4],
[5.4, 3.9, 1.3, 0.4],
[5.1, 3.5, 1.4, 0.3],
[5.7, 3.8, 1.7, 0.3],
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
```

In [13]:
```
y
```

```
Out[13]: array([[2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [2],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
```

```
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [3],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4],
       [4]])
```

In [19]:

```
#Now we will train a model with above data

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import warnings
import matplotlib.pyplot as plt      #warnings package is used for control of warnings(ie if to ignore them,
```

In [15]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

In [16]:

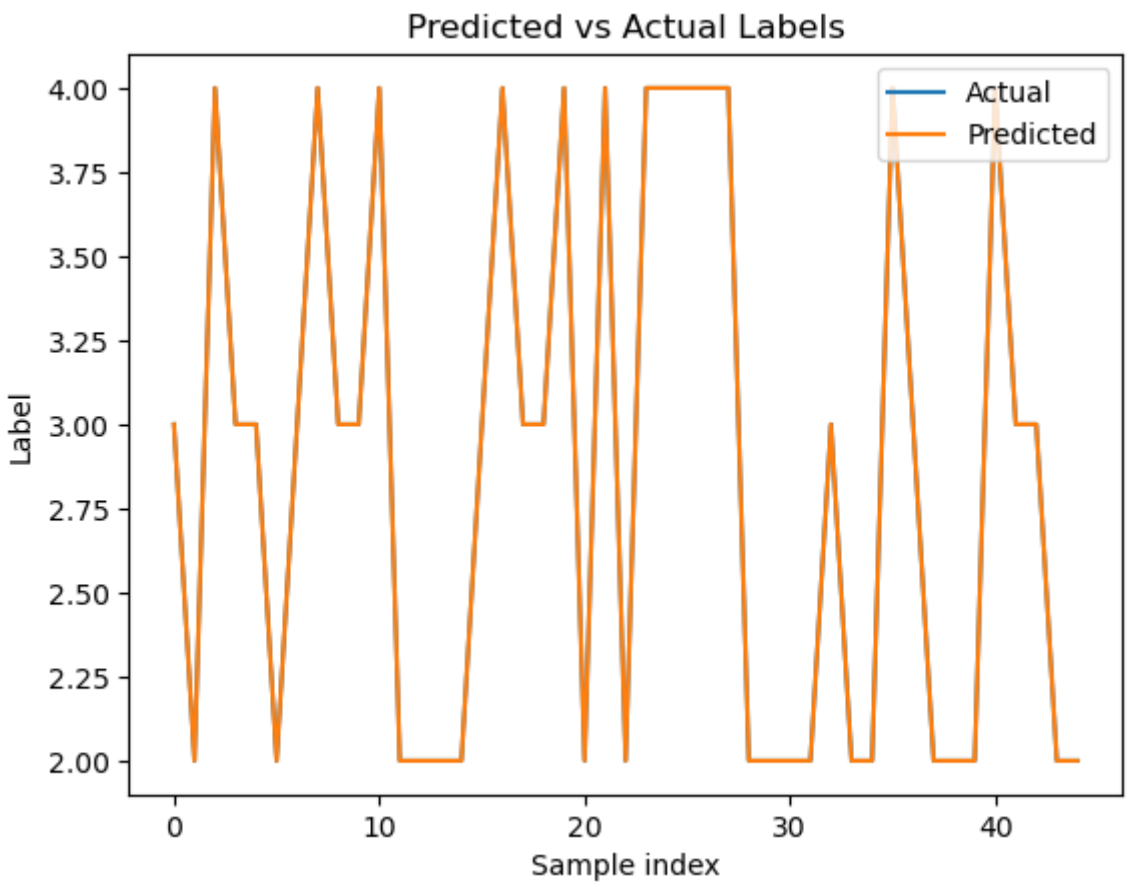
```
k=7
clf=KNeighborsClassifier(k)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
```

```
/Users/nitish/internship_task/env/lib/python3.8/site-packages/sklearn/neighbors/_classification.py:228: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

In [17]:
```
metrics.accuracy_score(y_test,y_pred)*100
```

Out[17]: 100.0

In [21]:
```
import matplotlib.pyplot as plt

# Create a figure and axis
fig = plt.figure()
ax = fig.add_subplot(111)

# Plot the predicted labels
ax.plot(y_test, label='Actual')

# Plot the actual labels
ax.plot(y_pred, label='Predicted')

# Set labels and title
ax.set_xlabel('Sample index')
ax.set_ylabel('Label')
ax.set_title('Predicted vs Actual Labels')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```



In [ ]: