

```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

## Step 1: Data Set Reading and Studying

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

	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 x 5 columns]
```

## Step 2: Data Set Cleaning and Analysis

```
#Data Cleaning
#checking for the presence of any null values in the iris dataset
iris.isnull().sum()
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
#checking the datatypes of columns of the dataset:
iris.dtypes
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species         object
dtype: object
```

```
#to list all the unique values in each column:
for i in iris.columns:
    print(f"{i}: \n {iris[i].unique()}")
```

```

sepal_length:
[5.1 4.9 4.7 4.6 5.  5.4 4.4 4.8 4.3 5.8 5.7 5.2 5.5 4.5 5.3 7.  6.4
6.9
6.5 6.3 6.6 5.9 6.  6.1 5.6 6.7 6.2 6.8 7.1 7.6 7.3 7.2 7.7 7.4 7.9]
sepal_width:
[3.5 3.  3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.  4.4 3.8 3.3 4.1 4.2 2.3 2.8
2.4
2.7 2.  2.2 2.5 2.6]
petal_length:
[1.4 1.3 1.5 1.7 1.6 1.1 1.2 1.  1.9 4.7 4.5 4.9 4.  4.6 3.3 3.9 3.5
4.2
3.6 4.4 4.1 4.8 4.3 5.  3.8 3.7 5.1 3.  6.  5.9 5.6 5.8 6.6 6.3 6.1
5.3
5.5 6.7 6.9 5.7 6.4 5.4 5.2]
petal_width:
[0.2 0.4 0.3 0.1 0.5 0.6 1.4 1.5 1.3 1.6 1.  1.1 1.8 1.2 1.7 2.5 1.9
2.1
2.2 2.  2.4 2.3]
species:
['setosa' 'versicolor' 'virginica']

```

```
iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```

import warnings
warnings.filterwarnings("ignore")

```

### Step 3 : Data Visualization / Exploratory Data Analysis (EDA)

```

count = iris["species"].value_counts()  #counts the different types
of species it has in the dataset
count

```

```

species
setosa      50
versicolor  50
virginica   50
Name: count, dtype: int64

```

```

# Plotting the columns through heatmap which is a 2D color-coded
matrix that allows relationships or patterns between two variables.
#Heatmap is mainly used to show correlation between features and

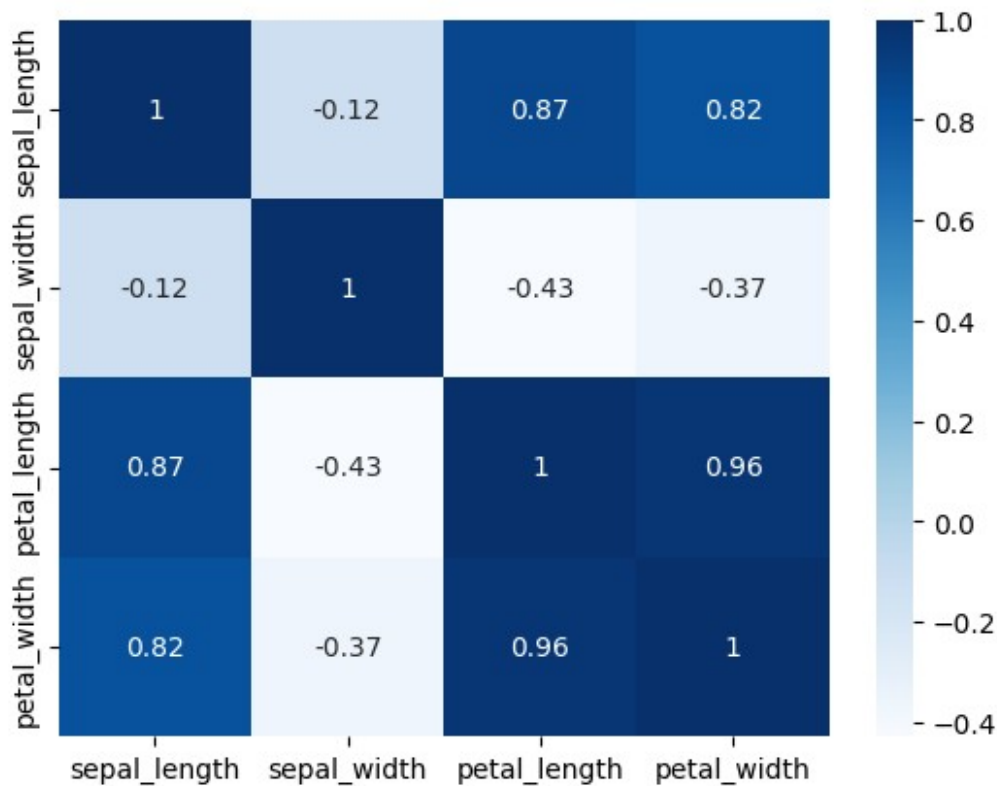
```

```
visualize confusion matrices.
#This mainly works on numerical data columns
columns = ["sepal_length", "sepal_width", "petal_length", "petal_width"]
iris[columns].corr()
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

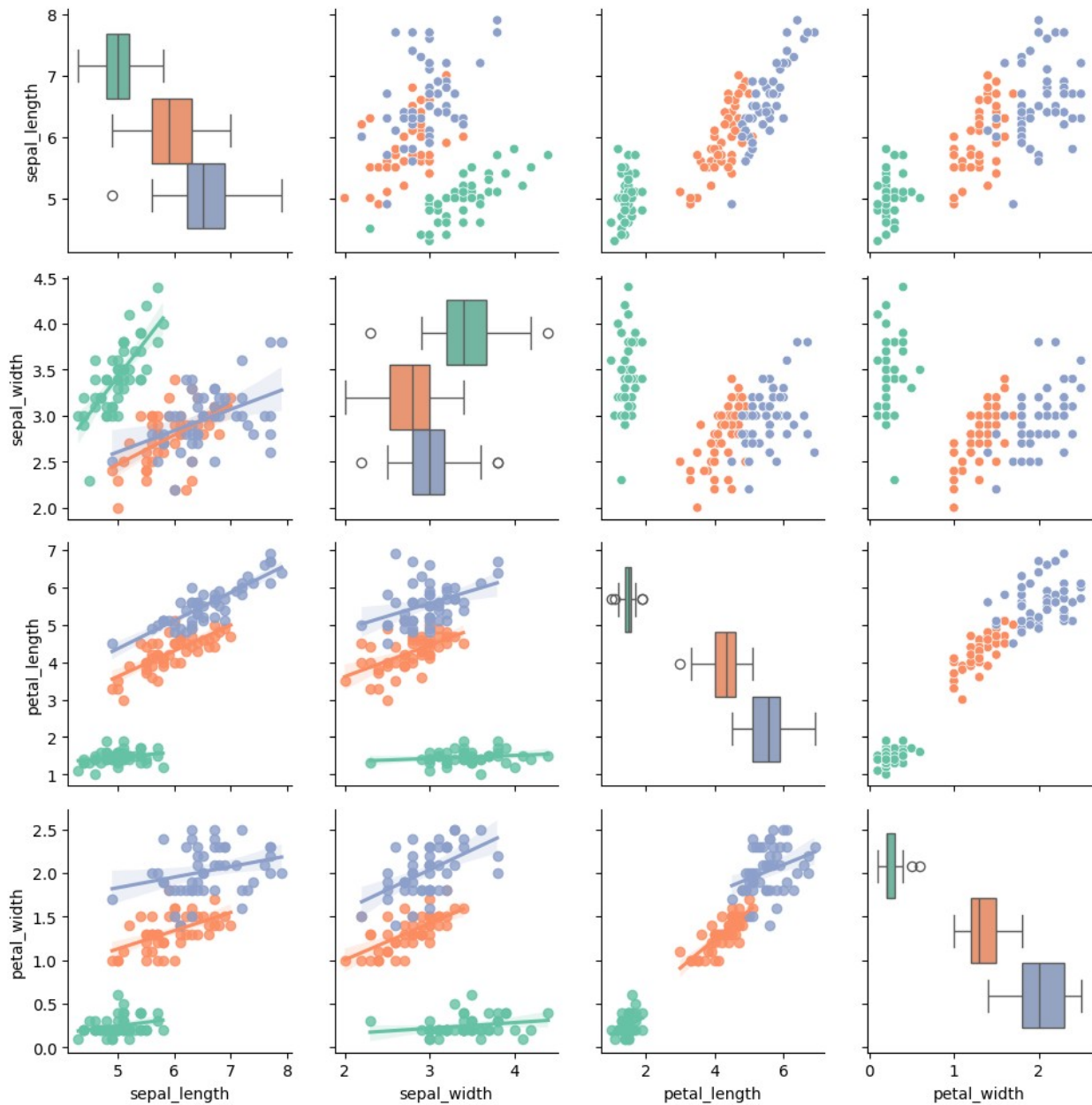
```
sns.heatmap(data = iris[columns].corr(),annot = True, cmap = "Blues")
#annot helps to labels the boxes
```

```
<Axes: >
```



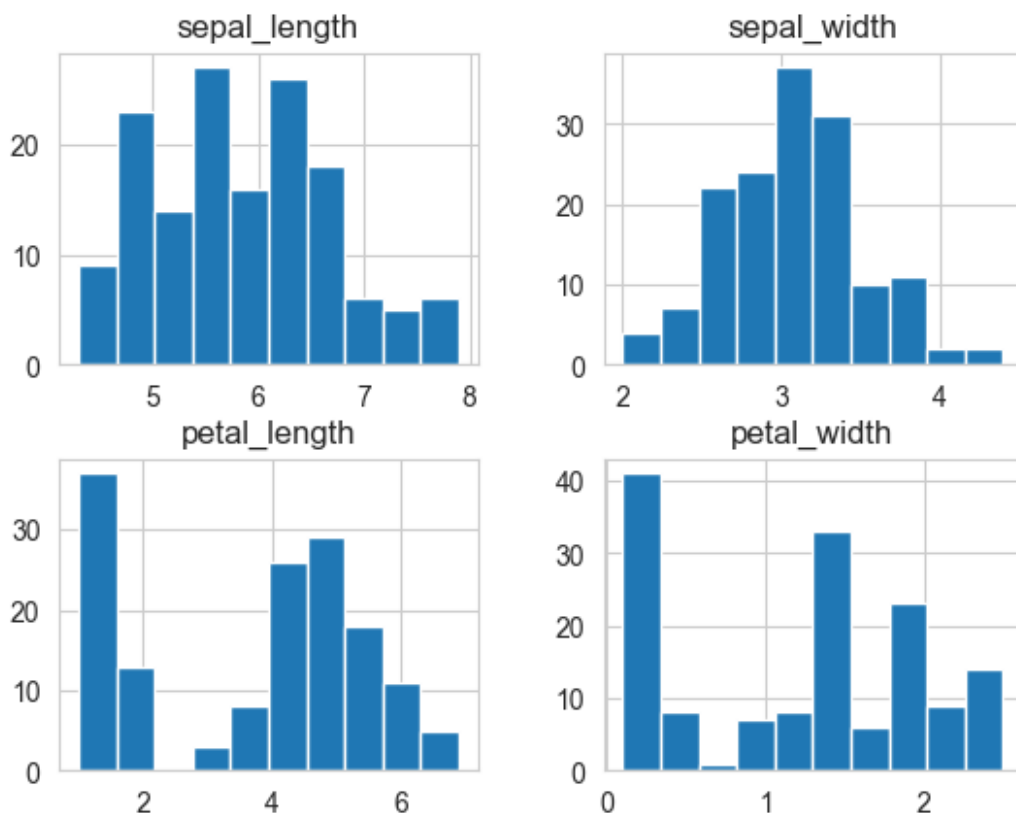
```
#Pair grid for plotting pairwise relationships in a dataset
graph = sns.PairGrid(data = iris, hue = "species", palette = "Set2")
graph.map_upper(sns.scatterplot) #upper graph
graph.map_lower(sns.regplot)    #lower graph
graph.map_diag(sns.boxplot)     #diagonal graph
```

```
<seaborn.axisgrid.PairGrid at 0x17aa12942f0>
```



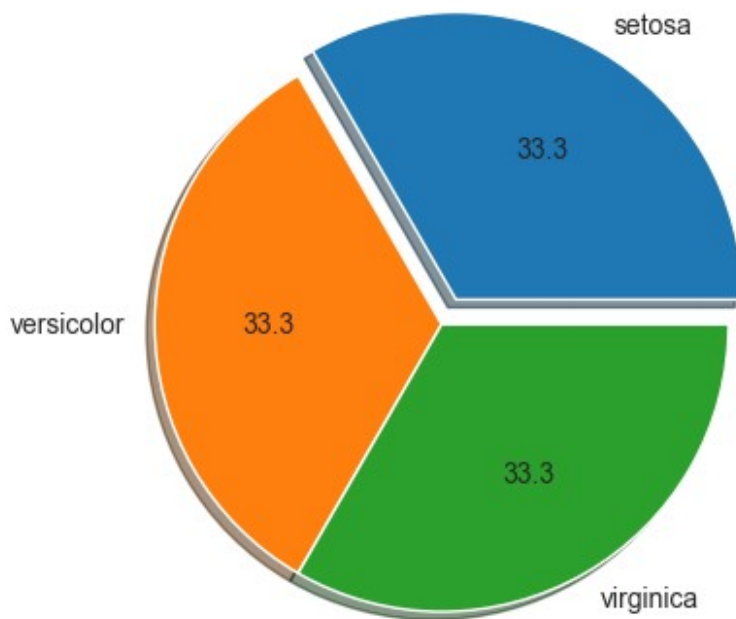
```
sns.set_style("whitegrid")
iris.hist()

array([[<Axes: title={'center': 'sepal_length'}>,
        <Axes: title={'center': 'sepal_width'}>],
       [<Axes: title={'center': 'petal_length'}>,
        <Axes: title={'center': 'petal_width'}>]], dtype=object)
```



```
plt.pie(count, labels = count.index, autopct = "%1.1f", shadow = True,
explode = [0.1, 0, 0])
```

```
([<matplotlib.patches.Wedge at 0x17ad5ba57f0>,
<matplotlib.patches.Wedge at 0x17ad6f0c690>,
<matplotlib.patches.Wedge at 0x17ad6f0cb90>],
[Text(0.5999999697158604, 1.039230502025882, 'setosa'),
Text(-1.0999999999999959, -9.616505800409723e-08, 'versicolor'),
Text(0.5500003659264656, -0.9526277328950455, 'virginica')],
[Text(0.3499999823342519, 0.6062177928484311, '33.3'),
Text(-0.5999999999999978, -5.2453668002234845e-08, '33.3'),
Text(0.3000001995962539, -0.5196151270336611, '33.3')])
```



#### Step 4 : Encoding

*#Label Encoding -> providing unique integer values based on alphabetical order*

```
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()
```

*#changing species column:*

```
iris.species = le.fit_transform(iris.species)  
iris.species.unique()
```

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

```
le.inverse_transform([0,1,2])
```

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

#### Step 5: IP/OP Creation

```
ip = iris.drop("species", axis = 1)  
ip.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```

op = iris.species
op
0      0
1      0
2      0
3      0
4      0
..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64

```

#### Step 6 : Train Test Split:

```

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(ip,op, test_size = 0.2)

print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)

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

xtrain.head()

```

	sepal_length	sepal_width	petal_length	petal_width
121	5.6	2.8	4.9	2.0
43	5.0	3.5	1.6	0.6
127	6.1	3.0	4.9	1.8
62	6.0	2.2	4.0	1.0
6	4.6	3.4	1.4	0.3

```

xtest.head()

```

	sepal_length	sepal_width	petal_length	petal_width
67	5.8	2.7	4.1	1.0
111	6.4	2.7	5.3	1.9
146	6.3	2.5	5.0	1.9
102	7.1	3.0	5.9	2.1
37	4.9	3.6	1.4	0.1

```

ytrain.head()

```

```

121    2
43     0
127    2
62     1
6      0

```

Name: species, dtype: int64

```
ytest.head()
```

```

67     1
111    2
146    2
102    2
37     0

```

Name: species, dtype: int64

Step 7 : Standardization : to bring all the ip to a ranges of -inf to +inf

```

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

```

```
#fit_transform
```

```
xtrain = sc.fit_transform(xtrain)
```

```
xtrain
```

```

array([[ -0.24490525,  -0.59447112,   0.70067165,   1.11674667],
       [ -0.95306501,   0.96601557,  -1.15528804,  -0.72260079],
       [  0.34522788,  -0.14861778,   0.70067165,   0.85398275],
       [  0.22720126,  -1.93203114,   0.19450083,  -0.19707294],
       [ -1.42517152,   0.7430889 ,  -1.26777044,  -1.11674667],
       [  0.58128114,  -1.70910447,   0.41946564,   0.19707294],
       [ -0.12687862,   2.97235561,  -1.21152924,  -0.98536471],
       [  0.69930777,  -0.37154445,   0.36322443,   0.19707294],
       [ -0.36293188,  -1.04032446,   0.41946564,   0.06569098],
       [ -0.48095851,   0.7430889 ,  -1.21152924,  -0.98536471],
       [ -1.18911827,  -0.14861778,  -1.26777044,  -1.3795106 ],
       [  0.81733439,  -0.14861778,   0.86939526,   1.11674667],
       [ -0.48095851,  -0.14861778,   0.47570684,   0.45983687],
       [ -1.18911827,   0.7430889 ,  -1.15528804,  -1.24812863],
       [  0.22720126,  -0.81739779,   0.81315406,   0.59121883],
       [ -0.008852 ,  -0.81739779,   0.13825962,   0.06569098],
       [  0.34522788,  -0.37154445,   0.58818924,   0.3284549 ],
       [  2.23365392,   1.63479558,   1.7130133 ,   1.3795106 ],
       [ -0.83503839,  -1.26325113,  -0.3679112 ,  -0.06569098],
       [  1.05338765,   0.07430889,   0.41946564,   0.3284549 ],
       [  1.05338765,   0.07430889,   1.09436007,   1.64227452],
       [  1.87957404,  -0.59447112,   1.37556608,   0.98536471],
       [  1.05338765,   0.52016223,   1.15060127,   1.77365648],
       [  2.23365392,  -1.04032446,   1.8254957 ,   1.51089256],
       [  1.17141428,  -0.14861778,   1.03811887,   1.24812863],

```

[ -0.83503839, 0.96601557, -1.26777044, -1.24812863 ],  
[ 0.69930777, 0.29723556, 0.47570684, 0.45983687 ],  
[ 0.58128114, -0.37154445, 1.09436007, 0.85398275 ],  
[ -0.71701176, 0.7430889, -1.26777044, -1.24812863 ],  
[ 0.81733439, 0.29723556, 0.81315406, 1.11674667 ],  
[ -0.48095851, 1.41186891, -1.21152924, -1.24812863 ],  
[ 0.81733439, -0.14861778, 1.03811887, 0.85398275 ],  
[ -1.42517152, 0.29723556, -1.26777044, -1.24812863 ],  
[ -0.95306501, 0.7430889, -1.21152924, -1.24812863 ],  
[ -1.07109164, -1.26325113, 0.47570684, 0.72260079 ],  
[ 1.40746753, 0.29723556, 0.58818924, 0.3284549 ],  
[ -0.12687862, -0.59447112, 0.25074203, 0.19707294 ],  
[ 0.10917463, 0.29723556, 0.64443045, 0.85398275 ],  
[ 0.58128114, 0.7430889, 1.09436007, 1.64227452 ],  
[ -0.83503839, 1.63479558, -0.98656443, -0.98536471 ],  
[ -0.83503839, 0.96601557, -1.26777044, -1.11674667 ],  
[ 0.69930777, -0.59447112, 1.09436007, 1.3795106 ],  
[ -0.48095851, 1.85772225, -1.32401165, -0.98536471 ],  
[ 0.46325451, -1.93203114, 0.47570684, 0.45983687 ],  
[ 0.22720126, -1.93203114, 0.75691285, 0.45983687 ],  
[ 0.69930777, -0.59447112, 1.09436007, 1.24812863 ],  
[ -1.18911827, 0.7430889, -0.98656443, -1.24812863 ],  
[ -0.12687862, 1.63479558, -1.09904684, -1.11674667 ],  
[ 1.05338765, 0.52016223, 1.15060127, 1.24812863 ],  
[ -0.36293188, 2.52650226, -1.26777044, -1.24812863 ],  
[ -0.48095851, 0.7430889, -1.09904684, -1.24812863 ],  
[ -1.3071449, 0.29723556, -1.15528804, -1.24812863 ],  
[ 0.58128114, -0.59447112, 0.81315406, 0.45983687 ],  
[ 0.93536102, -0.37154445, 0.53194804, 0.19707294 ],  
[ -0.12687862, -0.59447112, 0.47570684, 0.19707294 ],  
[ -0.83503839, 0.7430889, -1.21152924, -1.24812863 ],  
[ 0.22720126, 0.7430889, 0.47570684, 0.59121883 ],  
[ -0.12687862, -0.14861778, 0.30698323, 0.06569098 ],  
[ 0.81733439, -0.59447112, 0.53194804, 0.45983687 ],  
[ 1.2894409, 0.07430889, 0.81315406, 1.51089256 ],  
[ -0.008852, 2.08064892, -1.38025285, -1.24812863 ],  
[ -0.95306501, -2.37788448, -0.08670519, -0.19707294 ],  
[ -0.36293188, -1.4861778, 0.08201842, -0.06569098 ],  
[ -0.12687862, -1.04032446, -0.08670519, -0.19707294 ],  
[ 1.76154741, -0.37154445, 1.48804849, 0.85398275 ],  
[ -1.54319815, -1.70910447, -1.32401165, -1.11674667 ],  
[ -1.66122478, -0.37154445, -1.26777044, -1.24812863 ],  
[ -0.95306501, 0.96601557, -1.32401165, -1.11674667 ],  
[ -0.36293188, -1.26325113, 0.19450083, 0.19707294 ],  
[ 0.10917463, -0.14861778, 0.81315406, 0.85398275 ],  
[ 1.05338765, -0.14861778, 0.86939526, 1.51089256 ],  
[ -0.95306501, 0.29723556, -1.38025285, -1.24812863 ],  
[ -0.95306501, -1.70910447, -0.19918759, -0.19707294 ],  
[ -1.66122478, -0.14861778, -1.32401165, -1.24812863 ],

```
[ -0.71701176, 2.30357559, -1.21152924, -1.3795106 ],
[ 0.34522788, -0.59447112, 0.19450083, 0.19707294],
[ 0.46325451, -0.37154445, 0.36322443, 0.19707294],
[ -1.07109164, 0.07430889, -1.21152924, -1.3795106 ],
[ 0.69930777, 0.29723556, 0.92563646, 1.51089256],
[ -0.12687862, -0.37154445, 0.30698323, 0.19707294],
[ -1.07109164, -1.4861778 , -0.19918759, -0.19707294],
[ -0.59898513, 1.41186891, -1.21152924, -1.24812863],
[ 1.05338765, -0.14861778, 0.75691285, 0.72260079],
[ 1.64352078, 0.29723556, 1.31932488, 0.85398275],
[ 0.93536102, -0.14861778, 0.41946564, 0.3284549 ],
[ -0.95306501, -0.14861778, -1.15528804, -1.24812863],
[ 2.23365392, -0.59447112, 1.7130133 , 1.11674667],
[ -1.42517152, 0.07430889, -1.21152924, -1.24812863],
[ -0.36293188, -1.70910447, 0.19450083, 0.19707294],
[ -0.008852 , -0.59447112, 0.81315406, 1.64227452],
[ 0.34522788, -1.04032446, 1.09436007, 0.3284549 ],
[ -0.24490525, -1.26325113, 0.13825962, -0.06569098],
[ -1.42517152, 1.18894224, -1.49273526, -1.24812863],
[ 1.64352078, 1.18894224, 1.37556608, 1.77365648],
[ -0.83503839, 1.41186891, -1.21152924, -0.98536471],
[ 1.17141428, 0.29723556, 1.26308368, 1.51089256],
[ -0.36293188, -1.4861778 , 0.02577722, -0.19707294],
[ -0.83503839, 0.52016223, -1.09904684, -0.85398275],
[ -1.07109164, 0.07430889, -1.21152924, -1.24812863],
[ -0.12687862, -1.26325113, 0.75691285, 1.11674667],
[ -1.3071449 , 0.29723556, -1.32401165, -1.24812863],
[ 0.69930777, 0.07430889, 1.03811887, 0.85398275],
[ -0.95306501, 0.52016223, -1.26777044, -1.24812863],
[ 1.2894409 , 0.29723556, 1.15060127, 1.51089256],
[ 1.2894409 , 0.07430889, 0.98187766, 1.24812863],
[ 2.46970717, 1.63479558, 1.54428969, 1.11674667],
[ -0.83503839, 1.63479558, -1.15528804, -1.24812863],
[ -0.95306501, 1.18894224, -1.26777044, -1.24812863],
[ -1.18911827, -0.14861778, -1.26777044, -1.11674667],
[ 0.58128114, 0.52016223, 0.58818924, 0.59121883],
[ 1.17141428, -0.59447112, 0.64443045, 0.3284549 ],
[ 1.64352078, -0.14861778, 1.20684247, 0.59121883],
[ -1.66122478, 0.29723556, -1.32401165, -1.24812863],
[ -1.77925141, -0.14861778, -1.43649405, -1.3795106 ],
[ -0.24490525, -0.14861778, 0.47570684, 0.45983687],
[ 1.2894409 , 0.07430889, 0.70067165, 0.45983687],
[ -0.24490525, -0.14861778, 0.25074203, 0.19707294],
[ -1.07109164, -0.14861778, -1.26777044, -1.24812863],
[ 0.22720126, -0.37154445, 0.47570684, 0.45983687],
[ 0.58128114, -1.26325113, 0.70067165, 0.45983687]])
```

```
#transform
```

```
xtest = sc.transform(xtest)
```

xtest

```
array([[ -0.008852, -0.81739779,  0.25074203, -0.19707294],
       [  0.69930777, -0.81739779,  0.92563646,  0.98536471],
       [  0.58128114, -1.26325113,  0.75691285,  0.98536471],
       [  1.52549416, -0.14861778,  1.26308368,  1.24812863],
       [ -1.07109164,  1.18894224, -1.26777044, -1.3795106 ],
       [  0.81733439, -0.14861778,  1.20684247,  1.3795106 ],
       [  0.34522788, -0.14861778,  0.53194804,  0.3284549 ],
       [ -1.18911827,  0.07430889, -1.15528804, -1.24812863],
       [  0.10917463, -0.14861778,  0.30698323,  0.45983687],
       [ -0.71701176, -0.81739779,  0.13825962,  0.3284549 ],
       [  1.05338765, -1.26325113,  1.20684247,  0.85398275],
       [  1.05338765,  0.07430889,  0.58818924,  0.45983687],
       [  2.23365392, -0.14861778,  1.37556608,  1.51089256],
       [ -0.83503839,  1.63479558, -1.21152924, -1.11674667],
       [  0.58128114,  0.52016223,  1.31932488,  1.77365648],
       [ -0.24490525, -0.81739779,  0.30698323,  0.19707294],
       [  0.34522788, -0.59447112,  0.58818924,  0.06569098],
       [ -0.008852, -1.04032446,  0.19450083,  0.06569098],
       [  2.11562729, -0.14861778,  1.6567721,  1.24812863],
       [  0.46325451, -0.59447112,  0.64443045,  0.85398275],
       [ -0.36293188,  0.96601557, -1.32401165, -1.24812863],
       [ -0.48095851,  1.85772225, -1.09904684, -0.98536471],
       [  0.22720126, -0.14861778,  0.64443045,  0.85398275],
       [ -0.71701176,  0.96601557, -1.21152924, -1.24812863],
       [ -0.008852, -0.81739779,  0.81315406,  0.98536471],
       [ -0.008852, -0.81739779,  0.81315406,  0.98536471],
       [  0.46325451,  0.7430889,  0.98187766,  1.51089256],
       [ -0.95306501,  0.7430889, -1.15528804, -0.98536471],
       [  0.58128114, -0.81739779,  0.70067165,  0.85398275],
       [ -0.24490525, -0.37154445, -0.03046398,  0.19707294]])
```

Step 8 : Applying ML algorithm

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(xtrain,ytrain)

LinearRegression()
```

Step 9 :Testing the data

```
pred = lr.predict(xtest)
pred

array([ 0.98230431,  1.70420223,  1.63514632,  1.87611811, -
 0.10079844,
        1.98869168,  1.29125458,  0.01831651,  1.26395436,
 1.22750079,
```

```

1.74307987, 1.28602864, 1.95092026, 0.00573473,
2.23436036,
1.19855232, 1.21060193, 1.06091269, 1.99557221,
1.54354082,
-0.16214003, 0.07119363, 1.57377595, -0.06407879,
1.73606629,
1.73606629, 1.9785533 , 0.09815469, 1.55579965,
1.03565674])

```

ytest

```

67      1
111     2
146     2
102     2
37      0
104     2
91      1
30      0
61      1
59      1
108     2
86      1
135     2
19      0
100     2
94      1
73      1
92      1
105     2
126     2
36      0
5       0
138     2
27      0
142     2
101     2
148     2
26      0
123     2
64      1
Name: species, dtype: int64

```

MSE (Mean Squared Error) : to help find error

```

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(ytest,pred)
mse

```

```
0.05064454715177477
```

R2 Score : to find accuracy

```
from sklearn.metrics import r2_score  
mse = r2_score(ytest, pred)  
mse
```

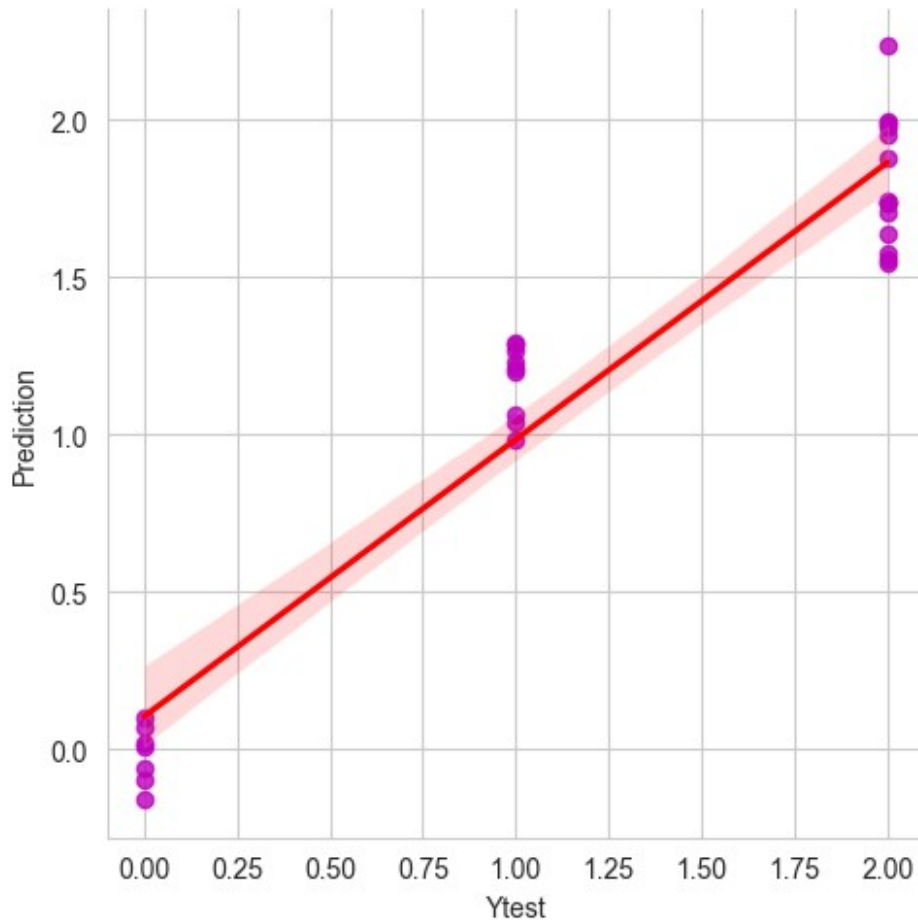
0.9215488942571475

Plotting Best Fit Line

```
df = pd.DataFrame({'Ytest' : list(ytest), 'Prediction' : pred})  
df.head()
```

	Ytest	Prediction
0	1	0.982304
1	2	1.704202
2	2	1.635146
3	2	1.876118
4	0	-0.100798

```
sns.lmplot(x = 'Ytest', y = 'Prediction', data = df, scatter_kws =  
{ 'color' : 'm' }, line_kws = { 'color' : 'r' })  
plt.show()
```



```
#KNN Algorithm
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(xtrain,ytrain)

KNeighborsClassifier()

#testing:
ypred1 = knn.predict(xtest)
ypred1

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

#### Step 10 : Checking Model's Performance

```
from sklearn.metrics import accuracy_score, recall_score,
precision_score, f1_score
accl = accuracy_score(ytest, ypred1)
recl = recall_score(ytest, ypred1, average='weighted')
prel = precision_score(ytest, ypred1, average='weighted')
```

```
f11 = f1_score(ytest, ypred1, average='weighted')
```

```
print("Accuracy:", acc1)
```

```
print("Recall(weighted):", rec1)
```

```
print("Precision(weighted):", pre1)
```

```
print("F1 Score(weighted):", f11)
```

```
Accuracy: 0.9333333333333333
```

```
Recall(weighted): 0.9333333333333333
```

```
Precision(weighted): 0.9454545454545454
```

```
F1 Score(weighted): 0.9341025641025641
```

```
#plotting confusion matrix
```

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(ytest, ypred1)
```

```
cm
```

```
array([[ 7,  0,  0],  
       [ 0,  9,  0],  
       [ 0,  2, 12]])
```

```
from sklearn.metrics import ConfusionMatrixDisplay
```

```
cmd = ConfusionMatrixDisplay(cm)
```

```
cmd.plot()
```

```
plt.show()
```

