IMPORTING NECESSARY LIBRARIES

```
import warnings                                    # to remove warnings
warnings.filterwarnings('ignore')

import pandas as pd                                # read file
import numpy as np                                 # do operations
import os                                          # file handling
import matplotlib.pyplot as plt                    # visualize data in graph
import seaborn as sns                              # simple graph mod plot in single line
```

Importing the dataset

```
iris = pd.read_csv("Iris.csv")
iris.head()
```

	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

DROPPING UNWANTED COLUMNS

```
iris = iris.drop(columns = ['Id'])                 # del Id attribute
iris.head()
```

	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

DATA ANALYSIS

```
iris.describe()                                    # to display stat abt data
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

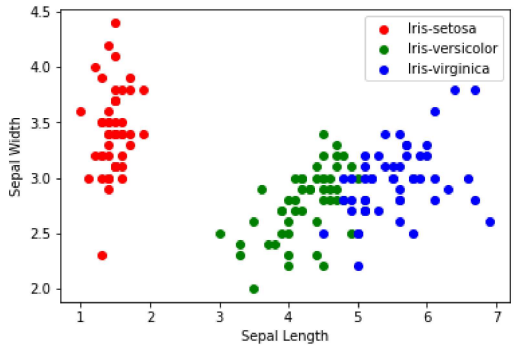
```
iris['Species'].value_counts()
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

```
clr = ['red','green','blue']
sp = ['Iris-setosa','Iris-versicolor','Iris-virginica']
```

```
for i in range(3):
    x = iris[iris['Species']==sp[i]]
    plt.scatter(x['PetalLengthCm'],x['SepalWidthCm'],c = clr[i],label = sp[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x286dac2f730>
```



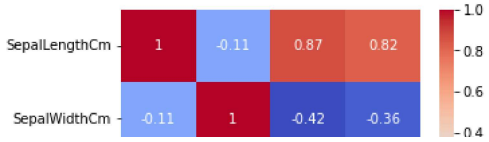
CORRELATION MATRIX

```
iris.corr()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
cor = iris.corr()
fig,ax= plt.subplots(figsize=(5,4))
sns.heatmap(cor,annot=True,ax=ax,cmap = 'coolwarm')
```

```
<AxesSubplot:>
```



LABEL ENCODER

(CHANGING CATOGERICAL VALUE TO NUMERIC VALUE)

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

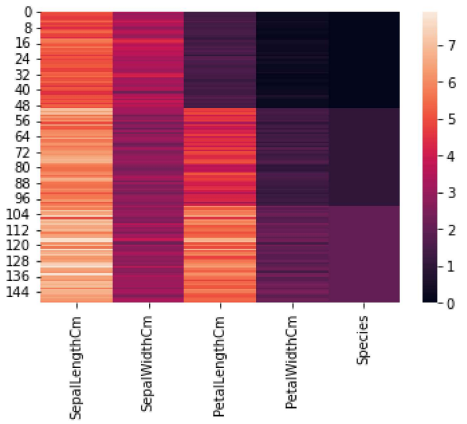
```
iris['Species'] = lb.fit_transform(iris['Species'])
iris
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

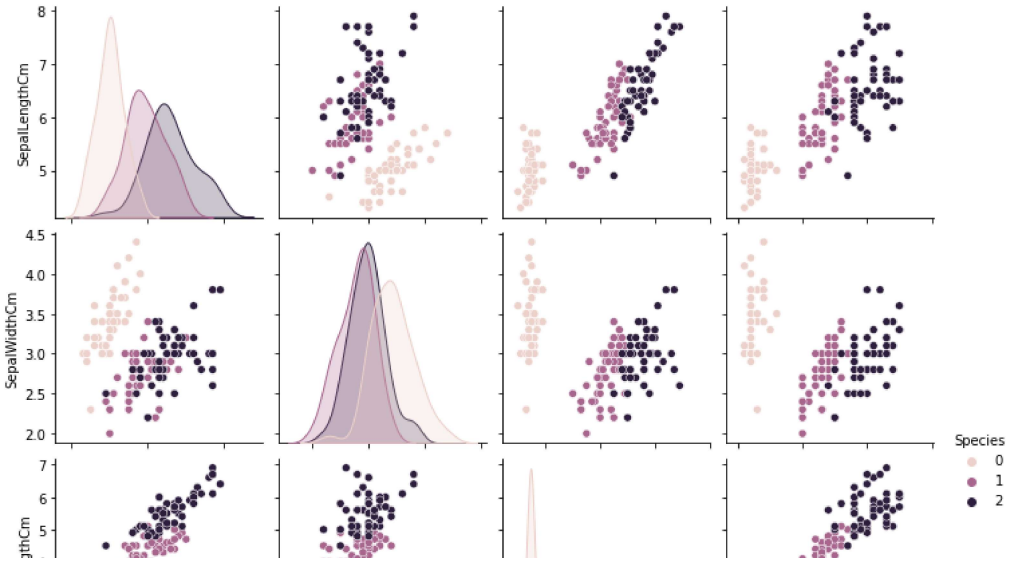
```
sns.heatmap(iris)
```

```
<AxesSubplot:>
```




```
sns.pairplot(iris,hue="Species")
```

```
<seaborn.axisgrid.PairGrid at 0x286d9a6d6a0>
```



SPLITING INTO TRAIN AND TEST DATA

```
from sklearn.model_selection import train_test_split
X = iris.drop(columns = ['Species'])
Y = iris['Species']
x_train,x_test,y_train,y_test =  train_test_split(X,Y,test_size = 0.30)
```

```
from sklearn.linear_model import LogisticRegression
mod = LogisticRegression()
```

```
mod.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
print("Accuracy:",mod.score(x_test,y_test)*100)
```

```
Accuracy: 95.55555555555556
```

```
from sklearn.neighbors import KNeighborsClassifier
mod1 = KNeighborsClassifier()
```

```
mod1.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
print("Accuracy:",mod1.score(x_test,y_test)*100)
```

```
Accuracy: 97.77777777777777
```

```
from sklearn.tree import DecisionTreeClassifier
mod2 = DecisionTreeClassifier()
```

```
mod2.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
print("Accuracy:",mod2.score(x_test,y_test)*100)
```

```
Accuracy: 91.11111111111111
```