```
#import the useful libraries.
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
# Read the data set of "Iris" in data.
df= pd.read_csv("Iris dataset.csv")
# Printing the data
df
```

	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
...	...	...	...	...	...	...
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

150 rows × 6 columns

```
#print the head of the data frame.
df.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

```
df.shape
```

```
(150, 6)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
df.describe()
```

	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

```
df.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
data = df.drop_duplicates(subset ="Species",)
data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
100	101	6.3	3.3	6.0	2.5	Iris-virginica

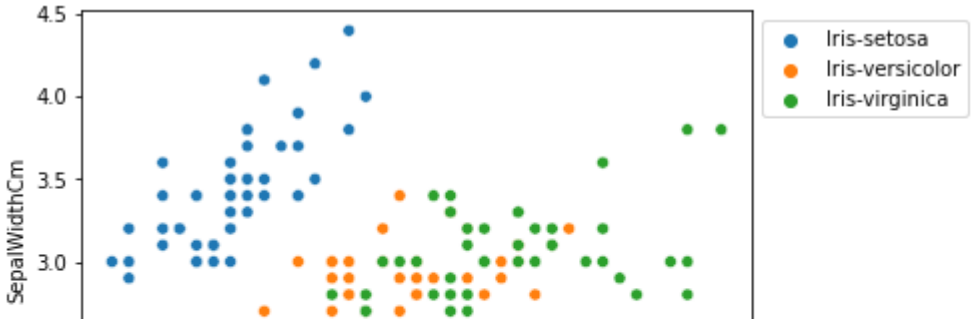
```
df.value_counts("Species")
```

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

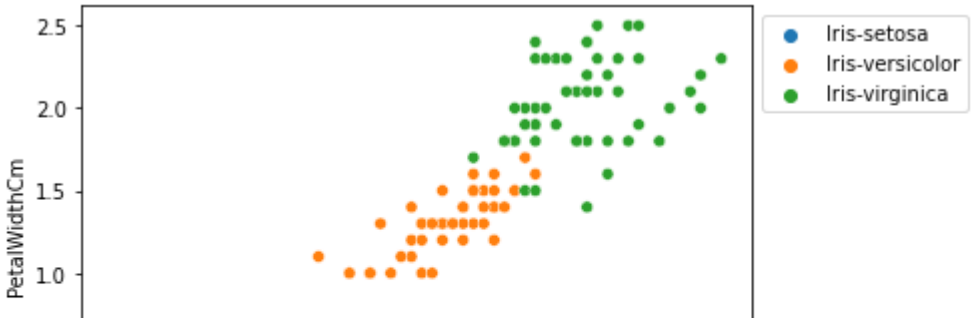
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Species', data=df, )
plt.show()
```


```
# importing packages
import seaborn as sns
```

```
import matplotlib.pyplot as plt
sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm',
                hue='Species', data=df, )
# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```



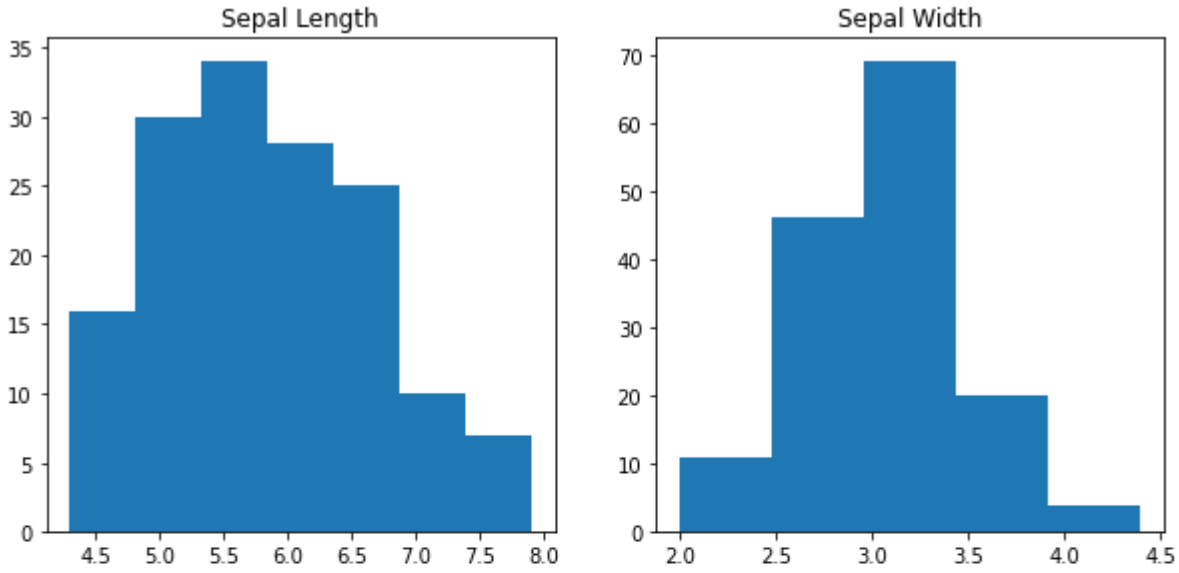

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='PetalLengthCm', y='PetalWidthCm',
                hue='Species', data=df, )
# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(df.drop(['Id'], axis = 1),
             hue='Species', height=2)
```

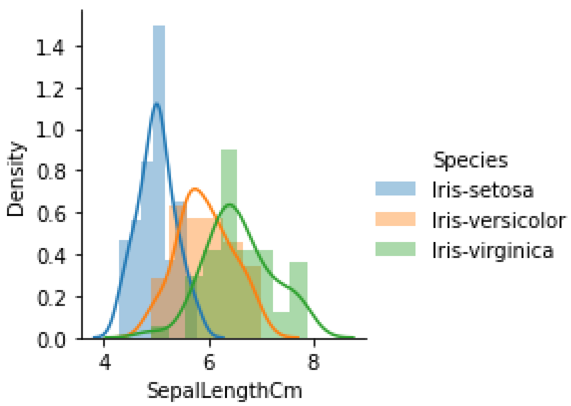
```
<seaborn.axisgrid.PairGrid at 0x7fbe7feecd90>
```

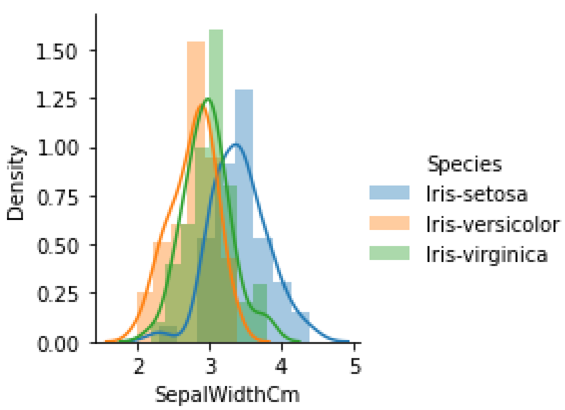
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['SepalLengthCm'], bins=7)
axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['SepalWidthCm'], bins=5);
axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['PetalLengthCm'], bins=6);
axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['PetalWidthCm'], bins=6);
```

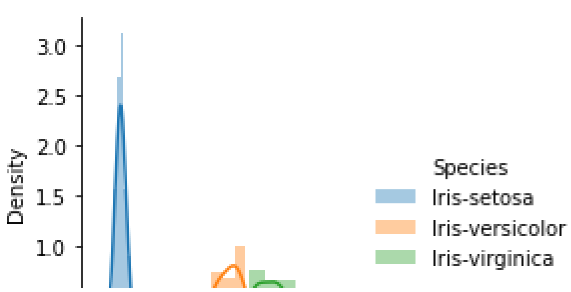


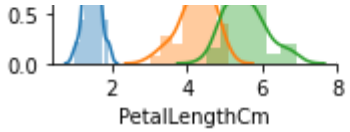

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalWidthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalLengthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalWidthCm").add_legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
```





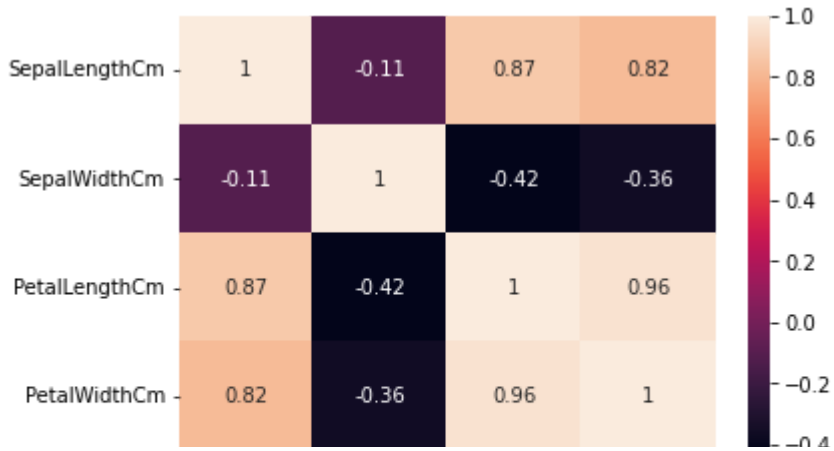





```
data.corr(method='pearson')
```

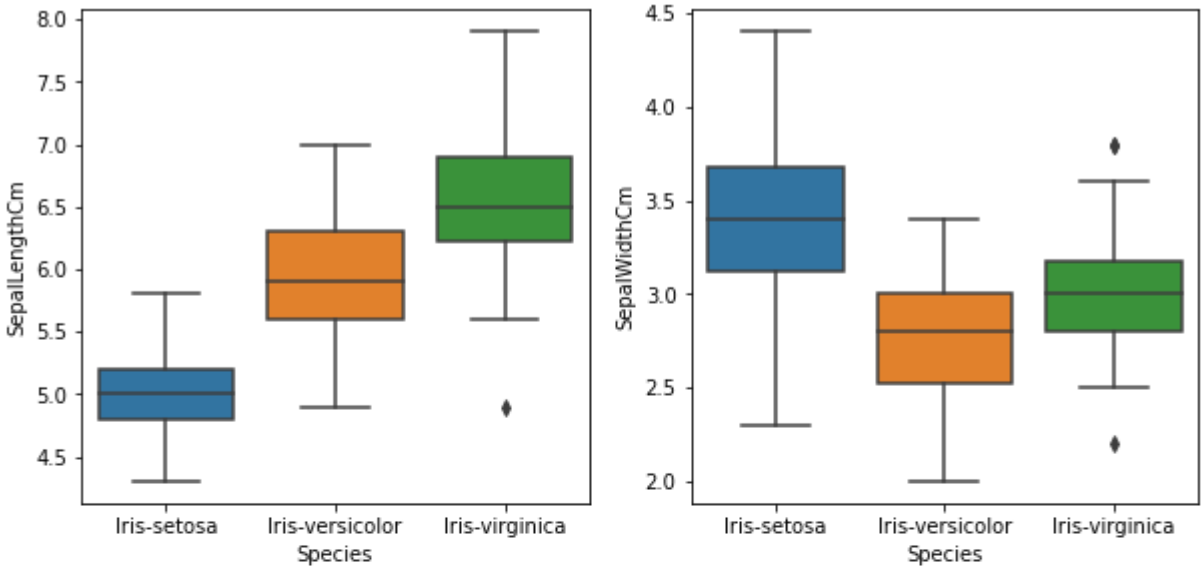
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.624413	-0.654654	0.969909	0.999685
SepalLengthCm	0.624413	1.000000	-0.999226	0.795795	0.643817
SepalWidthCm	-0.654654	-0.999226	1.000000	-0.818999	-0.673417
PetalLengthCm	0.969909	0.795795	-0.818999	1.000000	0.975713
PetalWidthCm	0.999685	0.643817	-0.673417	0.975713	1.000000

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.corr(method='pearson').drop(
  ['Id'], axis=1).drop(['Id'], axis=0),
            annot = True);
plt.show()
```



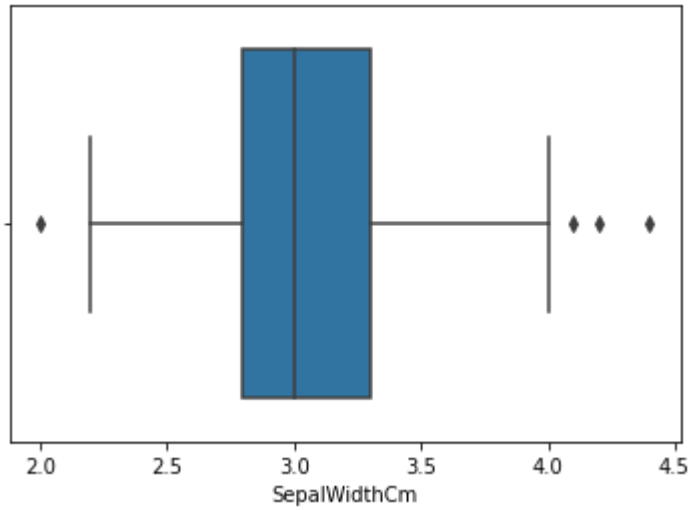

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
def graph(y):
    sns.boxplot(x="Species", y=y, data=df)
plt.figure(figsize=(10,10))
# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')
plt.subplot(222)
graph('SepalWidthCm')
plt.subplot(223)
graph('PetalLengthCm')
```

```
plt.subplot(224)
graph('PetalWidthCm')
plt.show()
```




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
# Load the dataset
df = pd.read_csv('Iris dataset.csv')
sns.boxplot(x='SepalWidthCm', data=df)
```

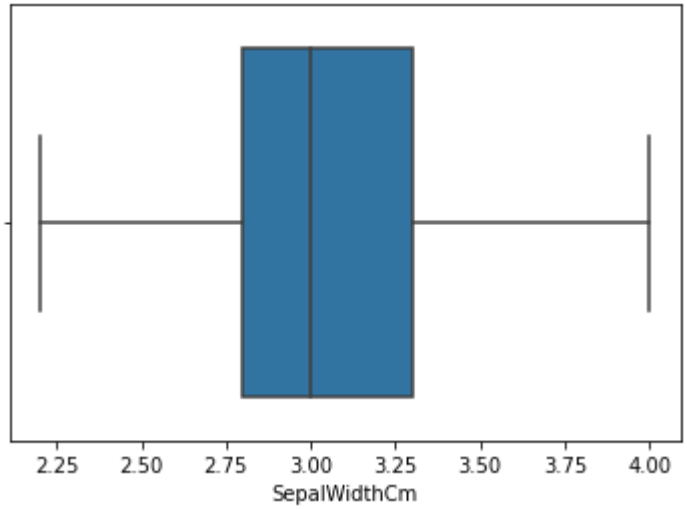
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe7b07f090>
```




```
# Importing
import sklearn
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns
# Load the dataset
df = pd.read_csv('Iris dataset.csv')
# IQR
Q1 = np.percentile(df['SepalWidthCm'], 25,
                   interpolation = 'midpoint')
Q3 = np.percentile(df['SepalWidthCm'], 75,
                   interpolation = 'midpoint')
```

```
IQR = Q3 - Q1
print("Old Shape: ", df.shape)
# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))
# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)
print("New Shape: ", df.shape)
sns.boxplot(x='SepalWidthCm', data=df)
```

```
Old Shape:  (150, 6)
New Shape:  (146, 6)
<matplotlib.axes._subplots.AxesSubplot at 0x7fbe7a088790>
```



REGRESSION MODEL

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
from sklearn.datasets import load_iris
data=load_iris()
dir(data)
```

```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data.data,data.target,test_size=0.2)
len(x_train),len(x_test)
```

```
(120, 30)
```

```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
reg.predict(x_test),y_test
```

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

ACCURACY MODEL

```
reg.score(x_test,y_test)
```

```
1.0
```

CLUSTERING MODEL

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MinMaxScaler
```

```
iris = pd.read_csv("Iris dataset.csv")
x = iris.iloc[:, [0, 1, 2, 3]].values
iris.info()
iris[0:10]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

	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

```
iris_outcome = pd.crosstab(index=iris["Species"],
                           columns="count")
iris_outcome
```

col_0	count
Species	
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

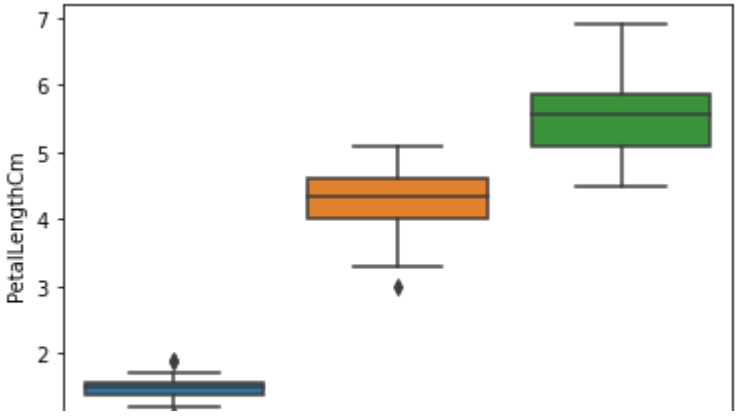
```
iris_setosa=iris.loc[iris["Species"]=="Iris-setosa"]
iris_virginica=iris.loc[iris["Species"]=="Iris-virginica"]
iris_versicolor=iris.loc[iris["Species"]=="Iris-versicolor"]
```

```
sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"PetalLengthCm").add_legend()
sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"PetalWidthCm").add_legend()
```

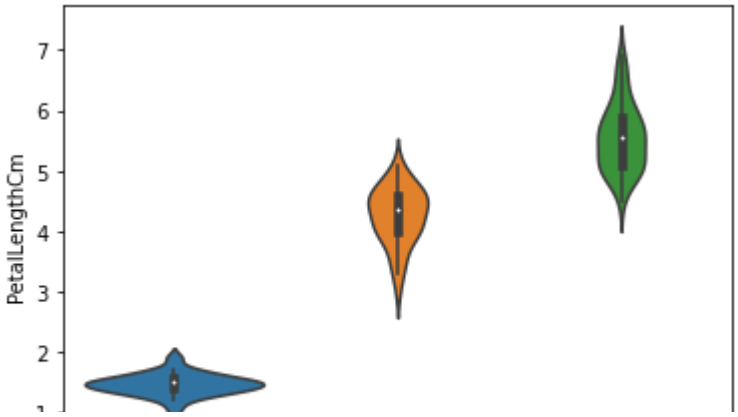
```
sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"SepalLengthCm").add_legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning: Th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
  warnings.warn(msg, FutureWarning)
```

```
sns.boxplot(x="Species",y="PetalLengthCm",data=iris)
plt.show()
```

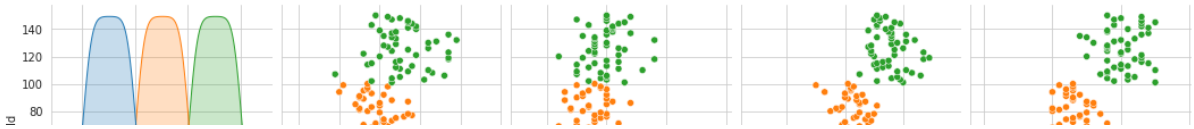



```
sns.violinplot(x="Species",y="PetalLengthCm",data=iris)
plt.show()
```

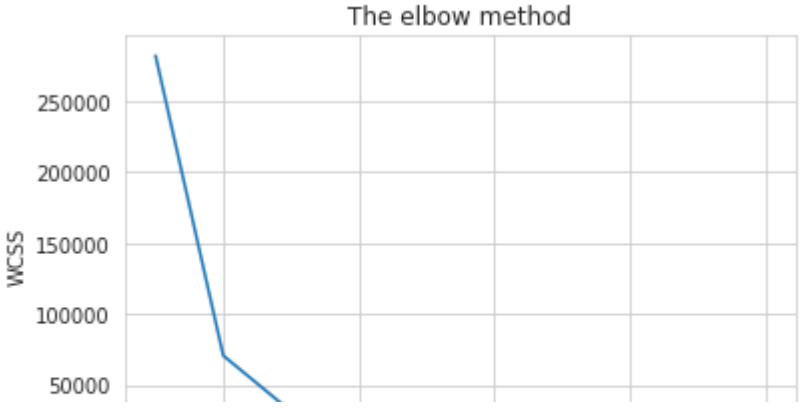



```
sns.set_style("whitegrid")
sns.pairplot(iris,hue="Species",size=3);
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarning: Th
  warnings.warn(msg, UserWarning)
```

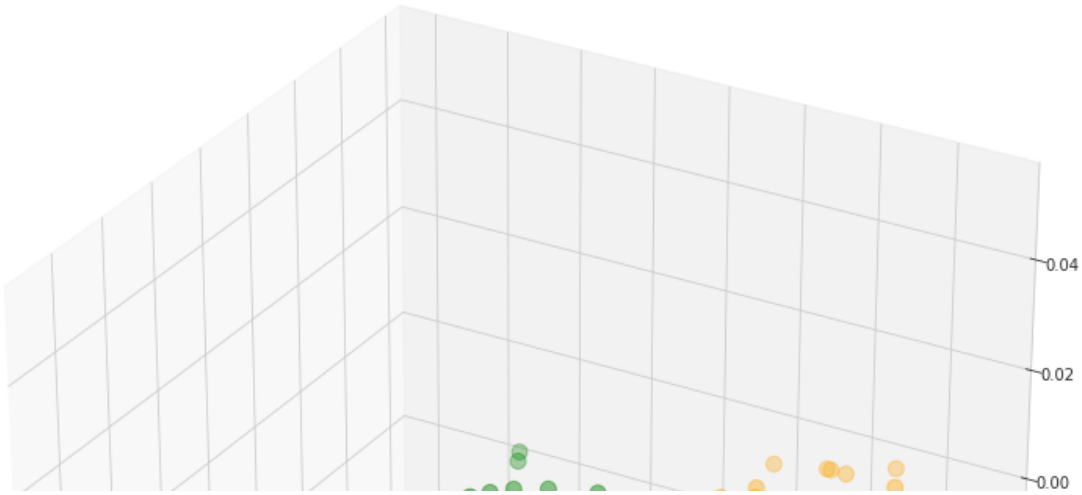



```
#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, rando
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)


plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```




```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_st
y_kmeans = kmeans.fit_predict(x)
#Visualising the clusters
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Iris
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label = 'Iris
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-
#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'red
plt.legend()
```

```
# 3d scatterplot using matplotlib

fig = plt.figure(figsize = (15,15))
ax = fig.add_subplot(111, projection='3d')
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Iris
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label = 'Iris
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iris-
#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s = 100, c = 'red
plt.show()
```



PERORMING CLASSIFICATION

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

```
from sklearn.model_selection import train_test_split
# Droping the target and species since we only need the measurements
X = iris.drop(['Id','Species'], axis=1)
# converting into numpy array and assigning petal length and petal width
X = X.to_numpy()[:, (2,3)]
y = iris['Id']
# Splitting into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X,y)
```

```
     LogisticRegression()
```

```
training_prediction = log_reg.predict(X_train)
training_prediction
```

```
array([ 77, 110,  60,  23,  99, 101,  23,  23,  63, 130,  14, 145,  23,
        23, 115,  99, 119, 135, 108, 119,  60,  23,  23, 135, 119,  23,
        23,  23, 135, 110,  23, 110, 119,  23, 135, 120, 115, 135, 145,
        14, 110, 107, 145,  80,  60,  74,  23,  74,  74,  23,  68, 119,
       119,  23,  99, 110, 123,  14, 110,  23, 135, 119, 123,  68, 123,
        74,  74, 115, 137,  14,  60, 115,  23,  68, 119])
```

```
test_prediction = log_reg.predict(X_test)
test_prediction
```

```
array([135,  23, 119,  77, 135,  23,  65, 145,  77,  63, 145,  14,  23,
        14,  23, 147, 119,  68,  74, 110,  14, 115,  23, 119, 119, 145,
       123, 119,  23,  14,  23,  23,  74,  14,  23, 115,  77,  23,  23,
        14, 145, 122, 134,  23,  23,  63, 135, 123,  74, 119,  91, 119,
        77,  23, 119,  68,  23,  23,  14,  80, 115,  13,  14,  23,  91,
        23,  68, 119,  23,  74, 115,  23, 115, 145,  68])
```

```
from sklearn import metrics
print("Precision, Recall, Confusion matrix, in training\n")
# Precision Recall scores
print(metrics.classification_report(y_train, training_prediction, digits=3))
# Confusion matrix
print(metrics.confusion_matrix(y_train, training_prediction))
```

```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           2      0.000     0.000     0.000         1
           3      0.000     0.000     0.000         1
           4      0.000     0.000     0.000         1
           6      0.000     0.000     0.000         1
           7      0.000     0.000     0.000         1
           8      0.000     0.000     0.000         1
           9      0.000     0.000     0.000         1
          14      0.000     0.000     0.000         1
          15      0.000     0.000     0.000         1
          18      0.000     0.000     0.000         1
          21      0.000     0.000     0.000         1
          22      0.000     0.000     0.000         1
          23      0.000     0.000     0.000         0
          35      0.000     0.000     0.000         1
          36      0.000     0.000     0.000         1
          38      0.000     0.000     0.000         1
          39      0.000     0.000     0.000         1
          42      0.000     0.000     0.000         1
          44      0.000     0.000     0.000         1
          47      0.000     0.000     0.000         1
          49      0.000     0.000     0.000         1
          50      0.000     0.000     0.000         1
          51      0.000     0.000     0.000         1
          53      0.000     0.000     0.000         1
```

54	0.000	0.000	0.000	1
55	0.000	0.000	0.000	1
58	0.000	0.000	0.000	1
59	0.000	0.000	0.000	1
60	0.250	1.000	0.400	1
62	0.000	0.000	0.000	1
63	0.000	0.000	0.000	0
64	0.000	0.000	0.000	1
68	0.000	0.000	0.000	0
71	0.000	0.000	0.000	1
72	0.000	0.000	0.000	1
73	0.000	0.000	0.000	1
74	0.000	0.000	0.000	0
75	0.000	0.000	0.000	1
77	0.000	0.000	0.000	0
78	0.000	0.000	0.000	1
80	1.000	1.000	1.000	1
81	0.000	0.000	0.000	1
84	0.000	0.000	0.000	1
85	0.000	0.000	0.000	1
88	0.000	0.000	0.000	1
89	0.000	0.000	0.000	1
90	0.000	0.000	0.000	1
91	0.000	0.000	0.000	1
92	0.000	0.000	0.000	1
93	0.000	0.000	0.000	1
94	0.000	0.000	0.000	1
99	0.333	1.000	0.500	1
100	0.000	0.000	0.000	1
101	0.000	0.000	0.000	1

```
print("Precision, Recall, Confusion matrix, in testing\n")
# Precision Recall scores
print(metrics.classification_report(y_test, test_prediction, digits=3))
# Confusion matrix
print(metrics.confusion_matrix(y_test, test_prediction))
```

```
      Precision, Recall, Confusion matrix, in testing

                   precision    recall  f1-score   support

                1      0.000     0.000     0.000         1
                5      0.000     0.000     0.000         1
               10      0.000     0.000     0.000         1
               11      0.000     0.000     0.000         1
               12      0.000     0.000     0.000         1
               13      0.000     0.000     0.000         1
               14      0.000     0.000     0.000         0
               16      0.000     0.000     0.000         1
               17      0.000     0.000     0.000         1
               19      0.000     0.000     0.000         1
               20      0.000     0.000     0.000         1
               23      0.050     1.000     0.095         1
               24      0.000     0.000     0.000         1
               25      0.000     0.000     0.000         1
               26      0.000     0.000     0.000         1
```

27	0.000	0.000	0.000	1
28	0.000	0.000	0.000	1
29	0.000	0.000	0.000	1
30	0.000	0.000	0.000	1
31	0.000	0.000	0.000	1
32	0.000	0.000	0.000	1
33	0.000	0.000	0.000	1
34	0.000	0.000	0.000	1
37	0.000	0.000	0.000	1
40	0.000	0.000	0.000	1
41	0.000	0.000	0.000	1
43	0.000	0.000	0.000	1
45	0.000	0.000	0.000	1
46	0.000	0.000	0.000	1
48	0.000	0.000	0.000	1
52	0.000	0.000	0.000	1
56	0.000	0.000	0.000	1
57	0.000	0.000	0.000	1
61	0.000	0.000	0.000	1
63	0.000	0.000	0.000	1
65	1.000	1.000	1.000	1
66	0.000	0.000	0.000	1
67	0.000	0.000	0.000	1
68	0.250	1.000	0.400	1
69	0.000	0.000	0.000	1
70	0.000	0.000	0.000	1
74	0.000	0.000	0.000	1
76	0.000	0.000	0.000	1
77	0.000	0.000	0.000	1
79	0.000	0.000	0.000	1
80	0.000	0.000	0.000	0
82	0.000	0.000	0.000	1
83	0.000	0.000	0.000	1
86	0.000	0.000	0.000	1
87	0.000	0.000	0.000	1
91	0.000	0.000	0.000	0
95	0.000	0.000	0.000	1
96	0.000	0.000	0.000	1
97	0.000	0.000	0.000	1