LetsGrowMore- Data Science Intern

Author: Mary Roshini L

TASK 01 -Iris Flower Classification

ABOUT THE DATA SET: Iris is a flowering plant with showy flowers.It is a genus of around 300 species of flowering plants. It takes the name from the Greek goddess of rainbow, Iris. The Iris flower data set is also know as the Fisher's Iris data set. It is a multivariate data set introduced by Ronald Fisher in his paper. The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. The data set is available at https://archive.ics.uci.edu/ml/datasets/Iris (https://archive.ics.uci.edu/ml/datasets/Iris), it consists the following information for 150 samples,

1. sepal length(cm)
2. sepal width(cm)
3. petal length(cm)
4. petal width(cm)

Aim : The aim is to classify iris flowers among three species (Setosa, Versicolor, or Virginica) from sepals' and petals' length and width measurements.

STEPS

1)Importing the data set

2) Data Preprocessing

3) Exploratory DataAnalysis

4) Model Building

5) Interpretation

importing the iris data set

In [442]:

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd #
from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

In [443]:

```
from warnings import filterwarnings
filterwarnings('ignore')
```

In [444]:

```
data=pd.read_csv('C:/Users/Mary roshini L/Desktop/Iris/iris1.csv',header =None)
data.head()
```

Out[444]:

	0	1	2	3	4
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

In [445]:

```
header = ["Sepal_Length", "Sepal_Width","Petal_Length", "Petal_Width", "Species"]
data.columns = header
data.head()
```

Out[445]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	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

Finding the Number of rows and columns

In [446]:

```
print(data.shape)
```

```
(150, 5)
```

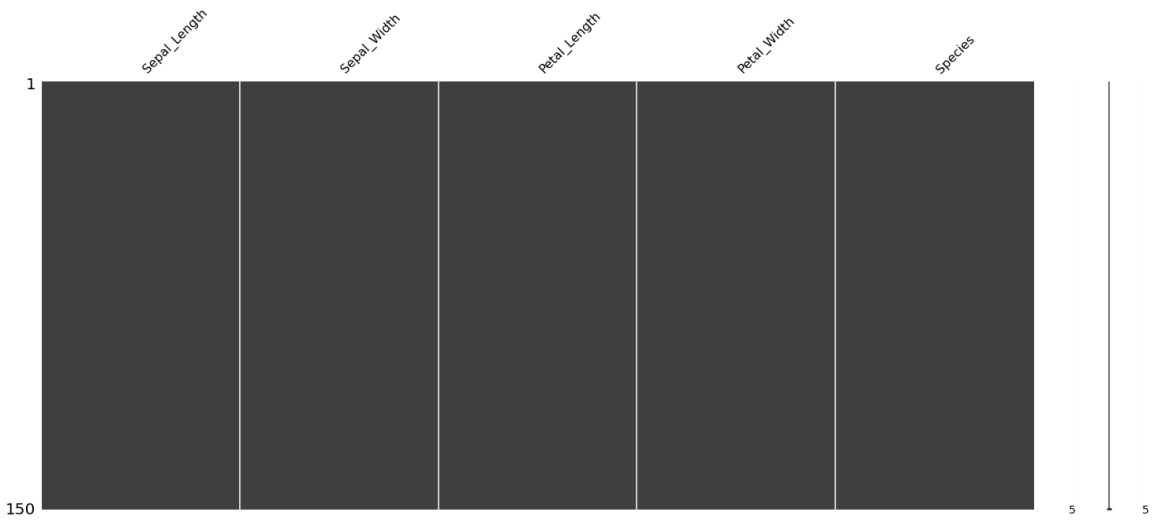
To check if there is any missing data

In [447]:

```
import seaborn as sns
import missingno as msno
%matplotlib inline
msno.matrix(data)
```

Out[447]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22329855e10>
```



Missingno package checks for missing values in the data.the above output displayes that there is no missing values

To check If there is a Null Value

In [448]:

```
data.isnull().sum()
```

Out[448]:

```
Sepal_Length    0
Sepal_Width     0
Petal_Length    0
Petal_Width     0
Species         0
dtype: int64
```

from the above output we can see that there are no null values from the data

Lets Count the number of values in each cloumn

In [449]:

```
data.count()
```

Out[449]:

```
Sepal_Length    150
Sepal_Width     150
Petal_Length    150
Petal_Width     150
Species         150
dtype: int64
```

So we can see that each coulmns contains 150 data points

Lets Group them according to the class they belong

In [450]:

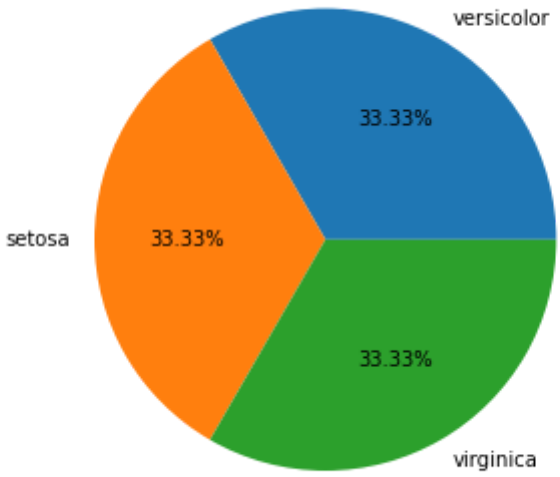
```
data.groupby('Species').size()
```

Out[450]:

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

In [451]:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
label = ['versicolor', 'setosa', 'virginica']
s = [50,50,50]
ax.pie(s, labels = label,autopct='%1.2f%%')
plt.show()
```



There fore we can see that now the data cleaning part is done we shall proceed with the next step

From the above output we can see that there are no missing data .now the data is ready for eda

Exploratory Data Analysis

In [452]:

```
data.describe()
```

Out[452]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
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

The above output discribes the summary statistics of the data

CHECKING FOR CORRELATION

In [453]:

```
data.corr()
```

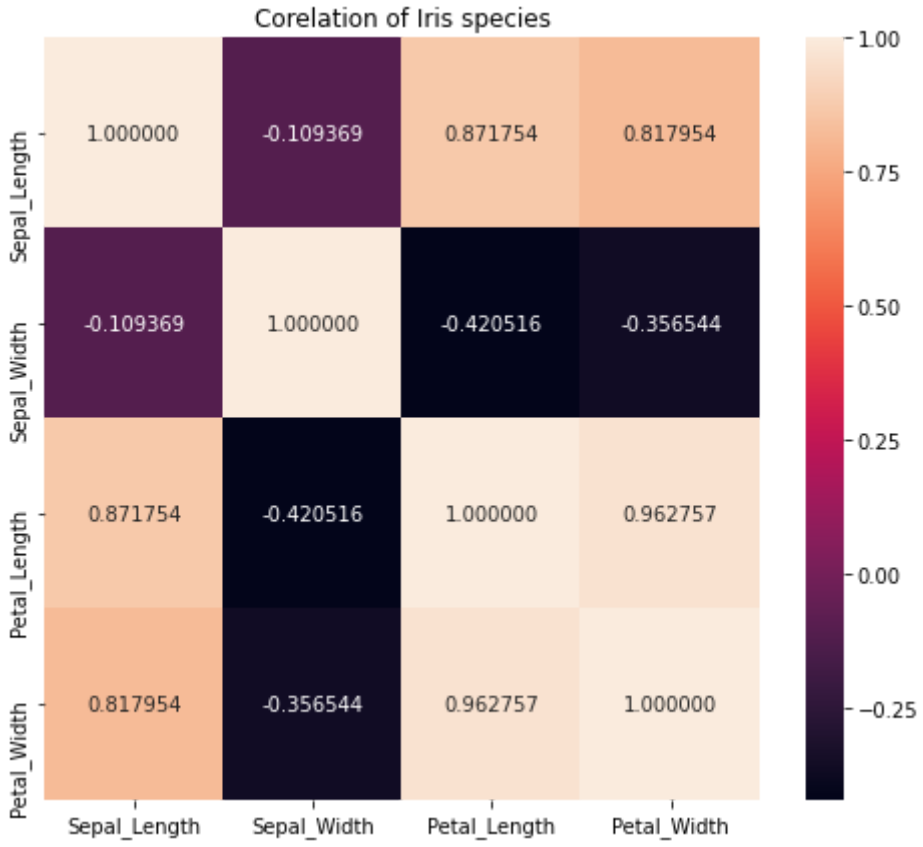
Out[453]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
Sepal_Length	1.000000	-0.109369	0.871754	0.817954
Sepal_Width	-0.109369	1.000000	-0.420516	-0.356544
Petal_Length	0.871754	-0.420516	1.000000	0.962757
Petal_Width	0.817954	-0.356544	0.962757	1.000000

Heat Map

In [454]:

```
plt.subplots(figsize = (8,7))
sns.heatmap(data.corr(),annot=True,fmt="f").set_title("Corelation of Iris species")
plt.show()
```



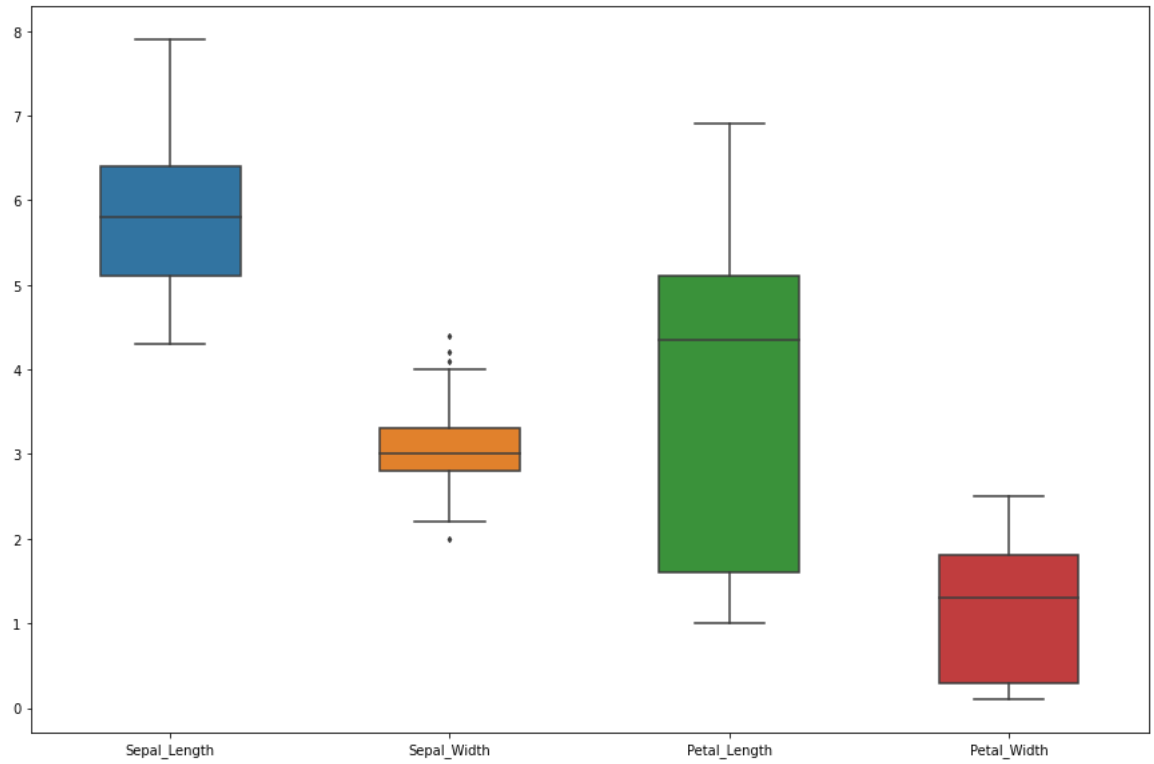
Checking for the Presence of Outlier

In [455]:

```
fig, ax = plt.subplots(figsize=(15,10))
sns.boxplot(data=data, width= 0.5,ax=ax,  fliersize=3)
```

Out[455]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232c79c0f0>
```



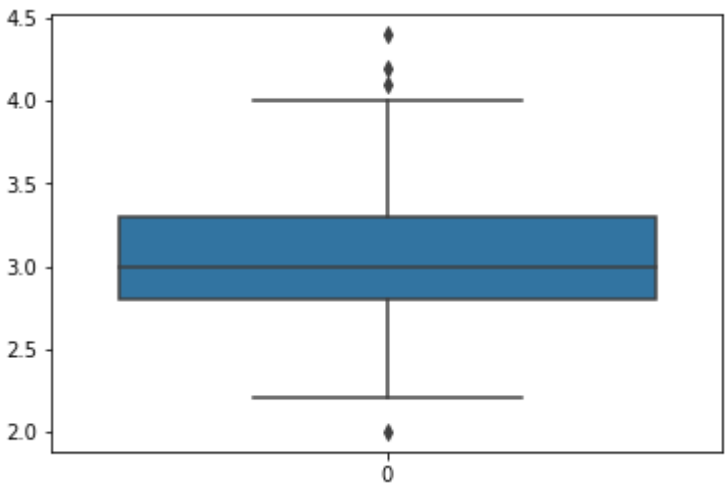
Lets look deep into the sepeal width column we noticed that there is a presence of outlier in the data

In [456]:

```
sns.boxplot(data=data['Sepal_Width'])
```

Out[456]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232c7d70f0>
```



Now lets treat the out_lier

We have two ways to remove the outliers one way is to remove it using the boxplot the second way is to replace the thrid quartile value using the discriptive statistics

In [457]:

```
data['Sepal_Width'].describe()
```

Out[457]:

```
count    150.000000
mean       3.054000
std        0.433594
min        2.000000
25%        2.800000
50%        3.000000
75%        3.300000
max        4.400000
Name: Sepal_Width, dtype: float64
```

data["Sepal_Width_Cm"]=np.where(data["Sepal_Width"]>3.30000,3.300000,data["Sepal_Width"])
data.head()

sns.boxplot(data=data['Sepal_Width_Cm'])

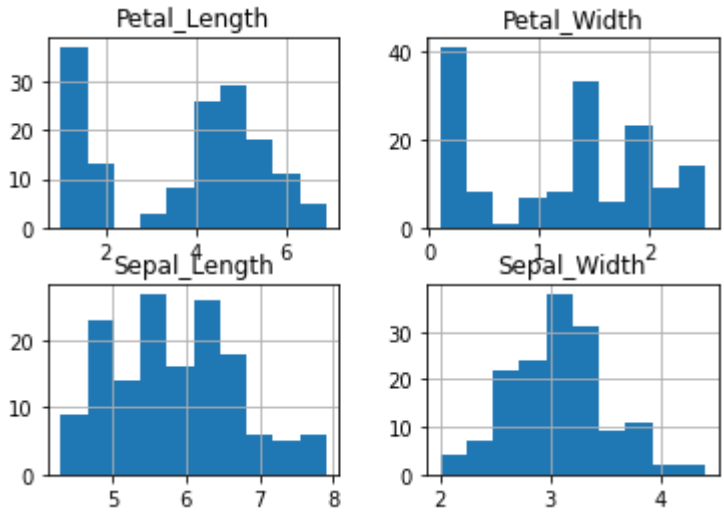
del data['Sepal_Width']

From the above graph we can see that we are not able to fully remove the outlier . Since the amou t of outlier is less we shall consider it for the futher analysis in order to avoid over fitting

Lets Try Finding the distribution of the data

In [461]:

```
data.hist()
plt.show()
```

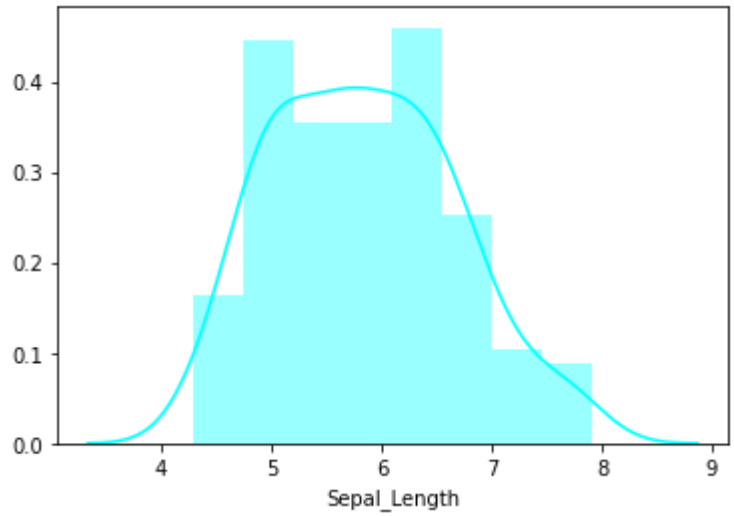


In [462]:

```
sns.distplot(data['Sepal_Length'], color = 'aqua')
```

Out[462]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232c7d7f98>
```

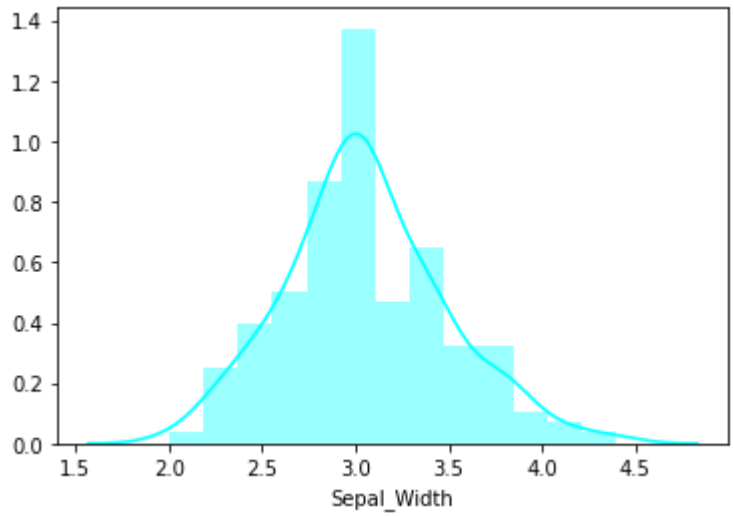


In [463]:

```
sns.distplot(data['Sepal_Width'], color = 'aqua')
```

Out[463]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232c89a320>
```

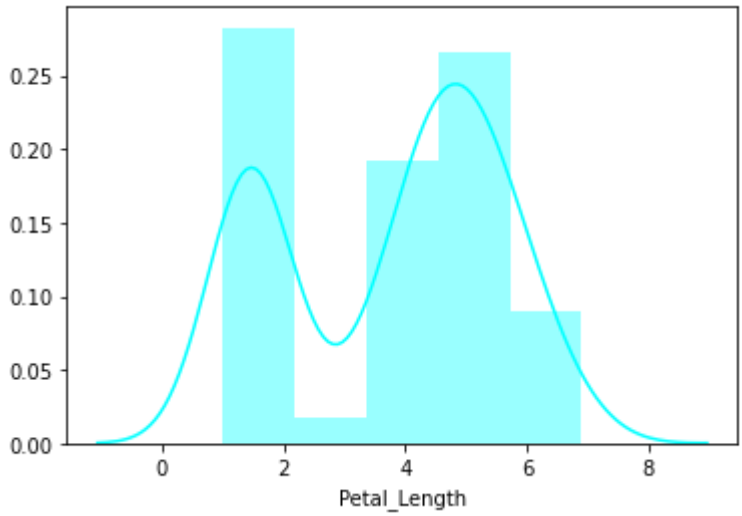


In [464]:

```
sns.distplot(data['Petal_Length'], color = 'aqua')
```

Out[464]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232c7c4c88>
```

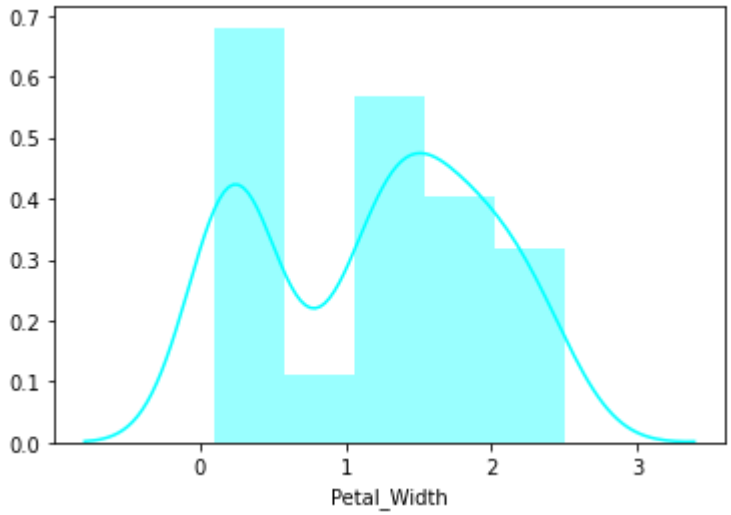


In [465]:

```
sns.distplot(data['Petal_Width'], color = 'aqua')
```

Out[465]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2232e219ba8>
```



The above graph suggest that it is very important for us to proceed with the normalising the data . Since the data is not normally distributed

SPLITTING THE DATA FOR MODELLING

In [466]:

```
### INPUT VARIABLES
x = data[['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']].values
x[0:4]
```

Out[466]:

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2]])
```

In [467]:

```
### OUTPUT VARIABLES
y = data['Species'].values
y[0:4]
```

Out[467]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa'],
      dtype=object)
```

In [468]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.2,
                                                    random_state = 1)
```

In [469]:

```
print ('Trainingset:', x_train.shape,  y_train.shape)
print ('Testingset :', x_test.shape,  y_test.shape)
```

```
Trainingset: (120, 4) (120,)
Testingset : (30, 4) (30,)
```

Lets Normalize the data

In [470]:

```
x_train = preprocessing.StandardScaler().fit(x_train).transform(x_train.astype(float))
x_train[0:5]
```

Out[470]:

```
array([[ 0.31553662, -0.03612186,  0.44748582,  0.2345312 ],
       [ 2.2449325 , -0.03612186,  1.29803965,  1.39642889],
       [-0.2873996 , -1.240184  ,  0.0505607 , -0.15276803],
       [ 0.67729835, -0.51774672,  1.01452171,  1.13822941],
       [-0.04622511, -0.51774672,  0.73100376,  1.52552864]])
```

In [471]:

```
x_test = preprocessing.StandardScaler().fit(x_test).transform(x_test.astype(float))
```

Model Building

In [482]:

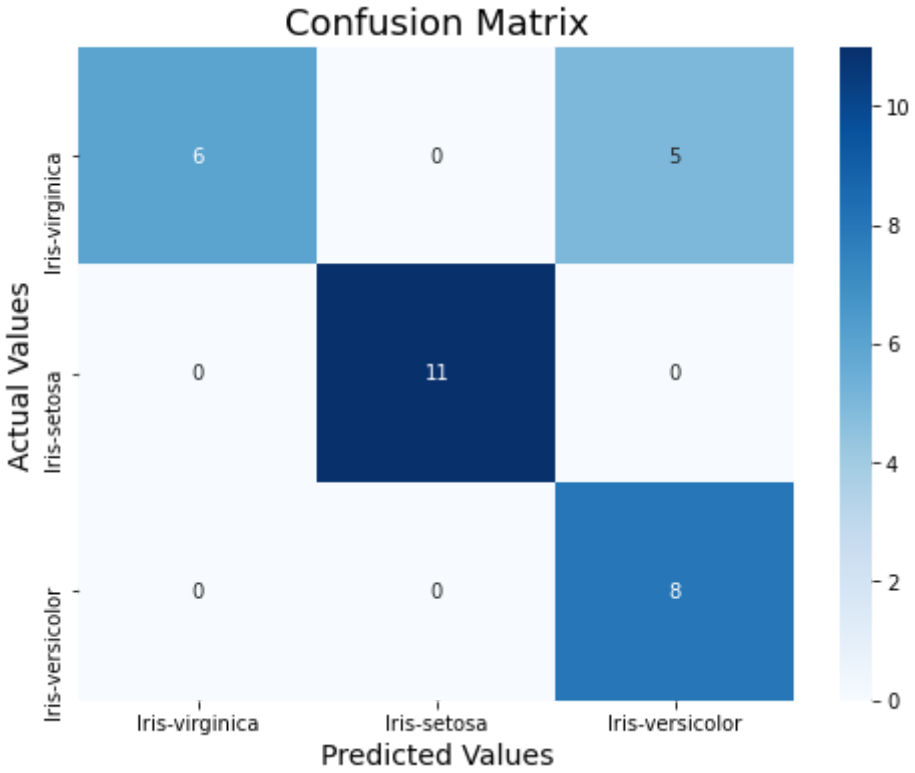
```
def confusion_matrix_plot(pred):
    labels = ['Iris-virginica', 'Iris-setosa', 'Iris-versicolor']
    matrix = confusion_matrix(pred, y_test, labels=labels)

    fig, ax = plt.subplots(figsize=(8,6))
    ax = sns.heatmap(matrix, annot = True, xticklabels = labels, yticklabels = labels,
cmap = "Blues")
    ax.set_title("Confusion Matrix", fontsize=18)
    ax.set_xlabel("Predicted Values", fontsize=14)
    ax.set_ylabel("Actual Values", fontsize=14)
```

1) KNN

In [492]:

```
from sklearn.neighbors import KNeighborsClassifier

start = time.time()
K = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
K.fit(x_train, y_train)
end = time.time()
final = end - start
final_K = round(final,3)

y_pred_K = K.predict(x_test)
accuracy_K=accuracy_score(y_test,y_pred_K)*100
accuracy_K=round(accuracy_K,2)

print("Accuracy is                  :", round(accuracy_K,2))
print("Computation time                    : {} - Sec".format(final_K))
print("\nClassification Report:\n",classification_report(y_test, y_pred_K))
confusion_matrix_plot(y_pred_K)
```

```
Accuracy is                  : 83.33
Computation time                    : 0.002 - Sec

Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.62      0.76        13
 Iris-virginica       0.55      1.00      0.71         6

       accuracy                           0.83        30
      macro avg       0.85      0.87      0.82        30
   weighted avg       0.91      0.83      0.84        30
```



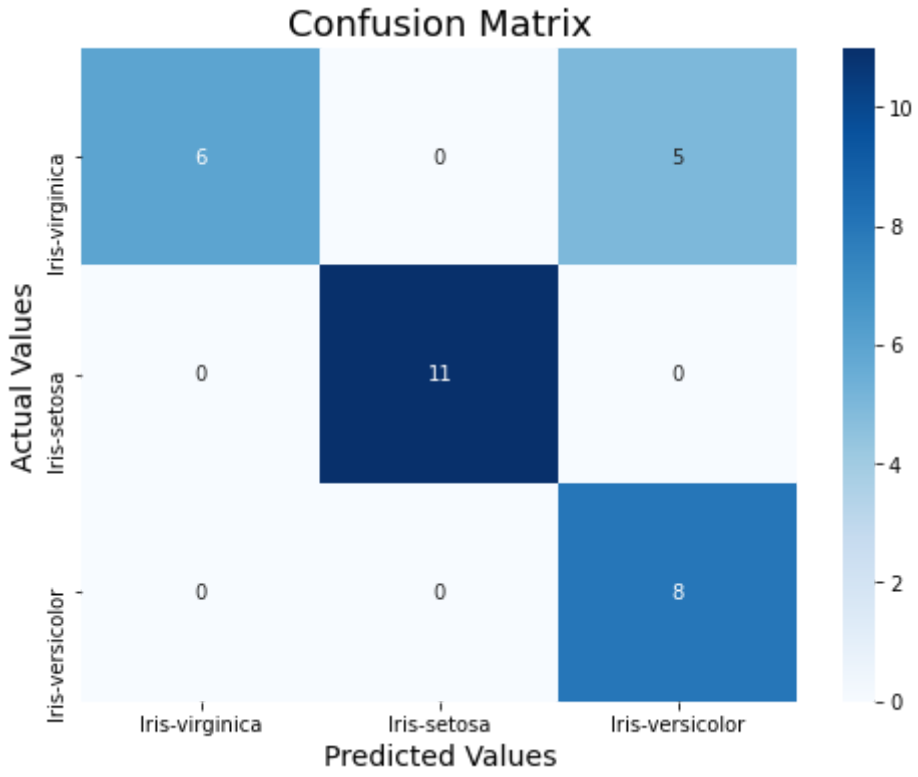
The above output suggest that out of 30 samples 25 samples is been predicted correctly.which can viewed using the confusion matrix

2) LOGISTIC REGRESSION

In [504]:

```
from sklearn.linear_model import LogisticRegression

c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space}

start = time.time()
L = GridSearchCV(LogisticRegression(),param_grid,cv = 5)
L.fit(x_train, y_train)
end = time.time()
final = end - start
final_L = round(final,3)

y_pred_L = L.predict(x_test)
accuracy=L.best_score_*100
accuracy_L=round(accuracy_L,2)

print("Tuned Logistic Regression Parameters: {}".format(L.best_params_))
print("Accuracy of Logistic Regression is  :", round(accuracy_L,2))
print("Computation time                    : {} - Sec".format(final_L))
print("\nClassification Report:\n",classification_report(y_test, y_pred_L))
confusion_matrix_plot(y_pred_L)
```

```
Tuned Logistic Regression Parameters: {'C': 3.727593720314938}
Accuracy of Logistic Regression is  : 95.83
Computation time                    : 1.464 - Sec

Classification Report:
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        11
 Iris-versicolor       1.00      0.62      0.76        13
  Iris-virginica       0.55      1.00      0.71         6

        accuracy                           0.83        30
       macro avg       0.85      0.87      0.82        30
    weighted avg       0.91      0.83      0.84        30
```



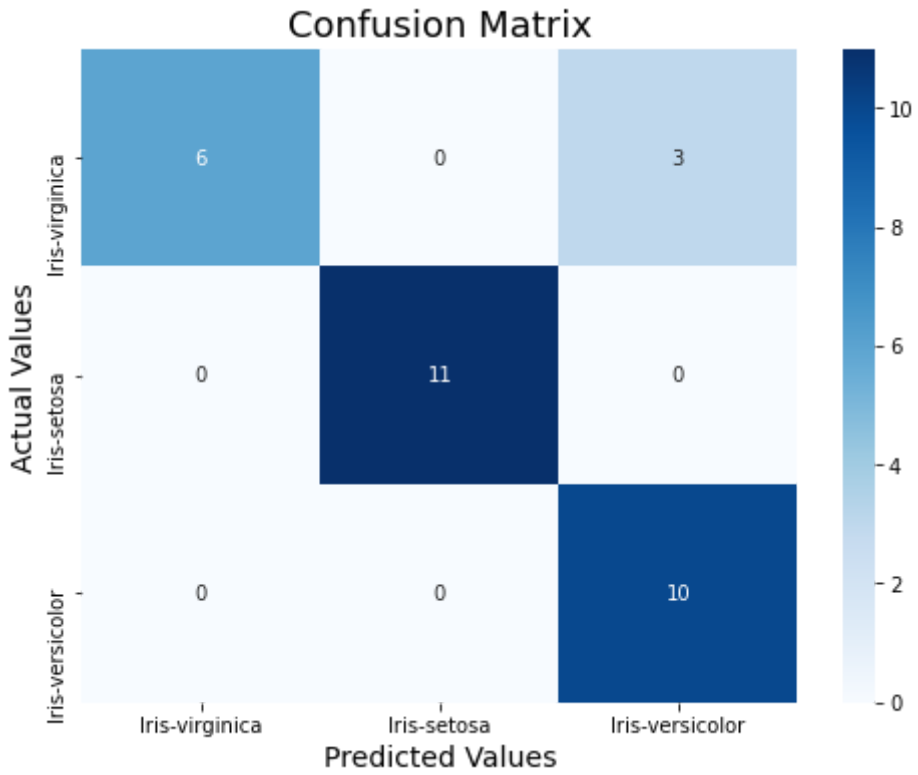
The accuracy given by logistics model is 95% which is quite a good value . Also its has correlctly predicted 25 out of 30

3) Desicion Tree

In [498]:

```
from sklearn.tree import DecisionTreeClassifier

start = time.time()
D = DecisionTreeClassifier()
D_model = D.fit(x_train, y_train)
end = time.time()
final = end - start
final_D = round(final,3)

y_pred_D = D_model.predict(x_test)
accuracy=accuracy_score(y_test, y_pred_D)*100
accuracy_D=round(accuracy,2)

print("Accuracy of Decision Tree is        :", round(accuracy_D,2))
print("Computation time                    : {} - Sec".format(final_D))
print("\nClassification Report:\n",classification_report(y_test, y_pred_D))
confusion_matrix_plot(y_pred_D)
```

```
Accuracy of Decision Tree is        : 90.0
Computation time                    : 0.002 - Sec

Classification Report:
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        11
 Iris-versicolor       1.00      0.77      0.87        13
  Iris-virginica       0.67      1.00      0.80         6

        accuracy                           0.90        30
       macro avg       0.89      0.92      0.89        30
    weighted avg       0.93      0.90      0.90        30
```



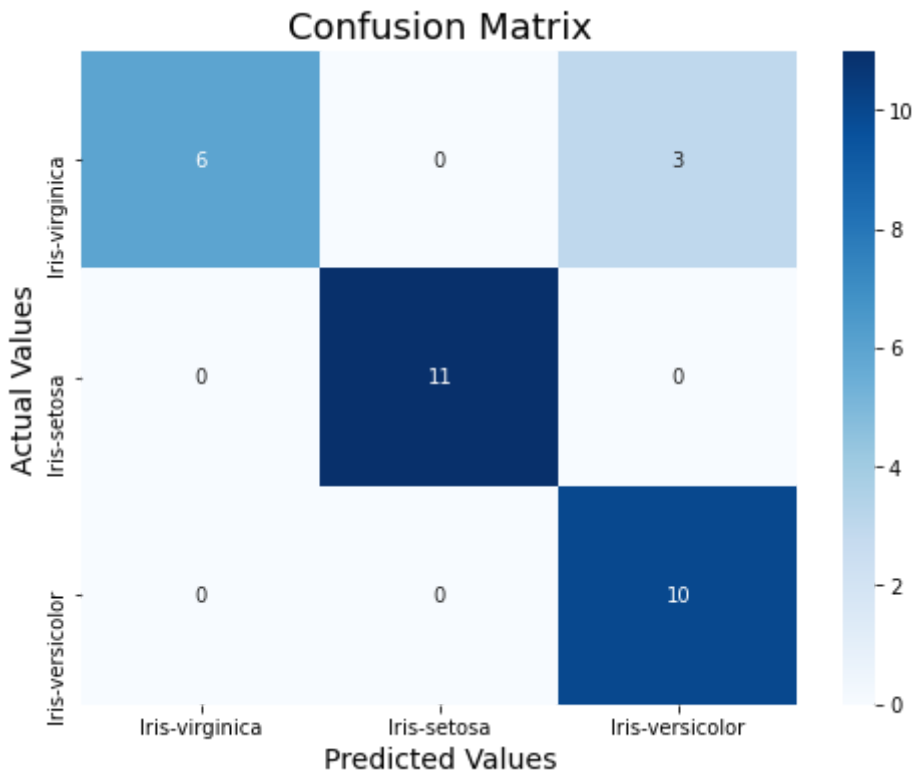
The above output has interpreted 27 out of 30 correctly which is quite a good improvement and the accuracy of the model is 90 %

4) RANDOM FOREST

In [497]:

```
from sklearn.ensemble import RandomForestClassifier

start = time.time()
R = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5,
                           n_estimators=100, oob_score=True)
R.fit(x_train, y_train)
end = time.time()
final = end - start
final_R = round(final,3)

y_pred_R = R.predict(x_test)
accuracy=accuracy_score(y_test,y_pred_R)*100
accuracy_R=round(accuracy,2)

print("Accuracy of  is        :", round(accuracy_R,2))
print("Computation time                    : {} - Sec".format(final_R))
print("\nClassification Report:\n",classification_report(y_test, y_pred_R))
confusion_matrix_plot(y_pred_R)
```

```
Accuracy of  is        : 90.0
Computation time                    : 0.362 - Sec

Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.77      0.87        13
 Iris-virginica       0.67      1.00      0.80         6

       accuracy                           0.90        30
      macro avg       0.89      0.92      0.89        30
   weighted avg       0.93      0.90      0.90        30
```



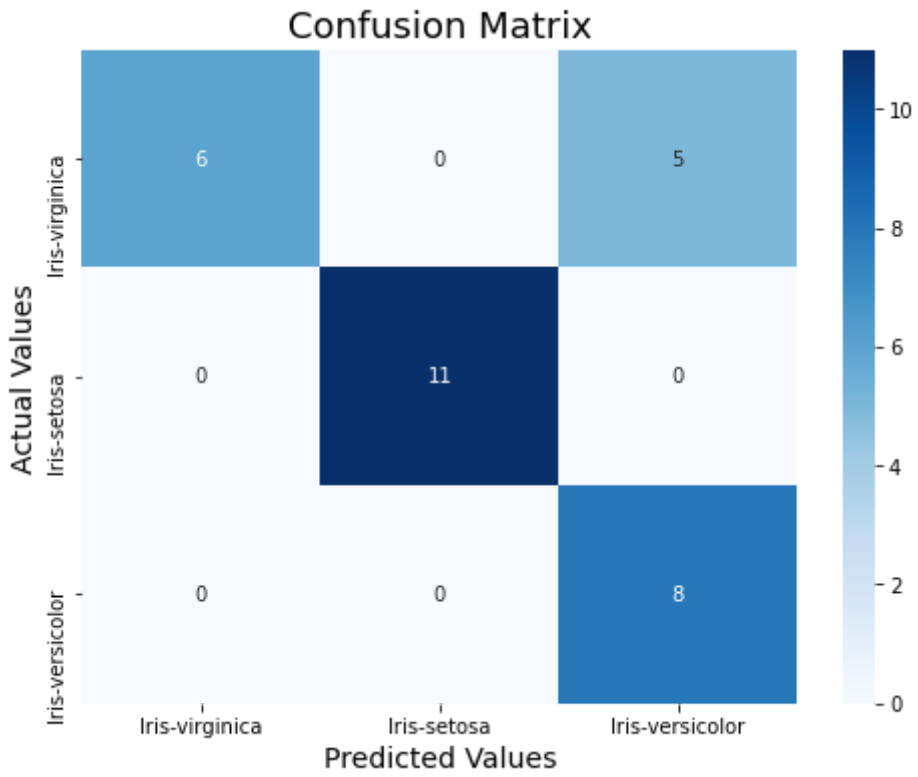
The above output has interpreted 27 out of 30 correctly which is quite a good improvement and the accuracy of the model is 90 %

5) Support Vector Machine

In [495]:

```
from sklearn.svm import SVC

start = time.time()
S = SVC()
S_model = S.fit(x_train, y_train)
end = time.time()
final = end - start
final_S = round(final,3)

y_pred_S = S_model.predict(x_test)
accuracy=accuracy_score(y_test, y_pred_S)*100
accuracy_S=round(accuracy,2)

print("Accuracy  is        :", accuracy_S)
print("Computation time                    : {} - Sec".format(final_S))
print("\nClassification Report:\n",classification_report(y_test, y_pred_S))
confusion_matrix_plot(y_pred_S)
```

```
Accuracy  is        : 83.33
Computation time                  : 0.003 - Sec

Classification Report:
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        11
 Iris-versicolor       1.00      0.62      0.76        13
  Iris-virginica       0.55      1.00      0.71         6

        accuracy                           0.83        30
       macro avg       0.85      0.87      0.82        30
    weighted avg       0.91      0.83      0.84        30
```



Out of 30 support Vector Classifier has perdicted 25 correcltly with an accuracy of 83%

FINAL RESULT

In [500]:

```
results = pd.DataFrame({
    'Algorithm': ['Logistic Regression','K Nearest Neighbours', 'Random Forest','Decisi
on Tree','SVC'],
    'Accuracy Score': [accuracy_L,accuracy_K,accuracy_R,accuracy_D,accuracy_S],
    'Execution Time in Secs': [final_L,final_K,final_R,final_D,final_S]})
results.head(9)
```

Out[500]:

	Algorithm	Accuracy Score	Execution Time in Secs
0	Logistic Regression	95.83	1.329
1	K Nearest Neighbours	83.33	0.003
2	Random Forest	90.00	0.362
3	Decision Tree	90.00	0.002
4	SVC	83.33	0.003

INTERPRETATION

From the above output we can see that the model which performs best classification task is Logistics Regression also considering the fact that the time taken by this model is quite high .AND accuracy given by logistics model is 95% which is quite a good value . Also its has correlctly predicted 25 out of 3 Where as KNN that out of 30 samples 25 samples is been predicted correctly.which can viewed using the confusion matrix with accuray of 83%.Random Forest has interpreted 27 out of 30 correctly which is quite a good improvement and the accuracy of the model is 90 %. Decision Tree The above output has interpreted 27 out of 30 correctly which is quite a good improvement and the accuracy of the model is 90 %. Support vector Machine Classifier has perdicted 25 correclty with an accuracy of 83%