Predict-the-class-of-flower-in-IRIS-dataset

IRIS dataset contains 3 classes of 50 instance each class reffers to a type of iris plant. The three classes in this datasets are Setosa,Versicolor,and Verginica Learn Decision Trees,CART algorithms and essemble method. Then use Random Forest classifier to make prediction.

Description

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

Id

1.SepalLengthCm

2.SepalWidthCm

3.PetalLengthCm

4.PetalWidthCm

5.Species

Sepal Width vs. Sepal Length

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
# load dataset
df = pd.read_csv('iris.csv')
df
```

Out[2]:

	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 × 5 columns

In [3]:
```
df['species'].value_counts()
```

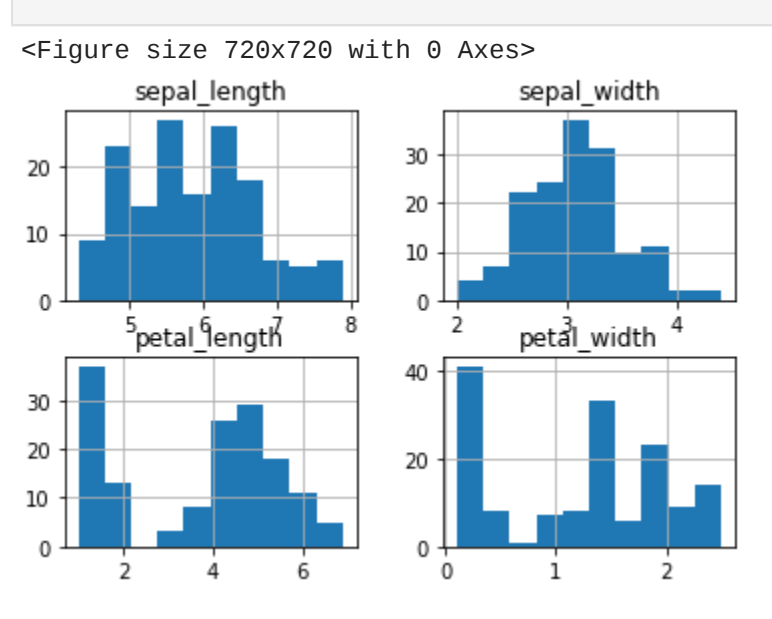
Out[3]:
```
virginica     50
versicolor    50
setosa        50
Name: species, dtype: int64
```

In [4]:
```
# checking shape of data frame
df.shape
```

Out[4]:
```
(150, 5)
```

In [40]:
```
# checking data distrubution of each columns
plt.figure(figsize=(10,10))
df.hist()
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



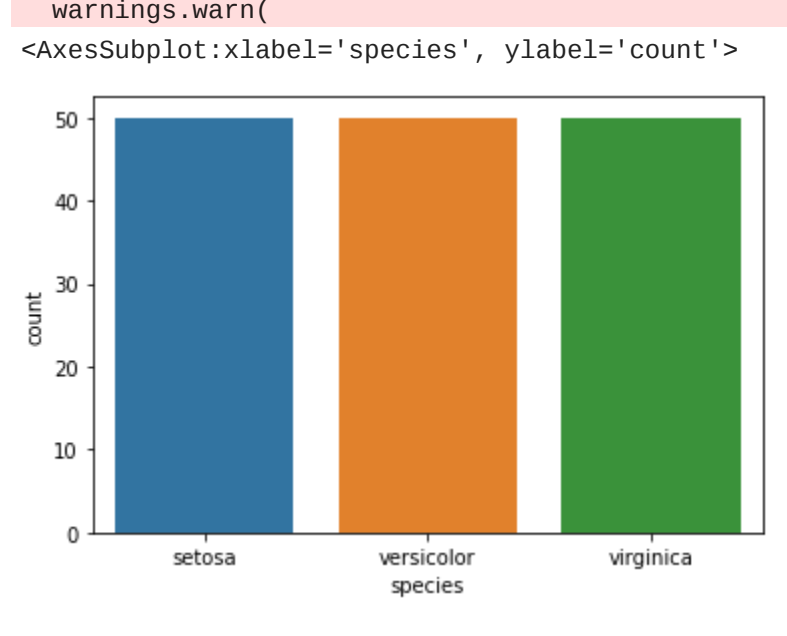
In [45]:
```
A=sns.pairplot(df, hue='species', size=4)
#g.map_upper(sns.kdeplot, levels=1, color=".2")
plt.show()
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\axisgrid.py:1969: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

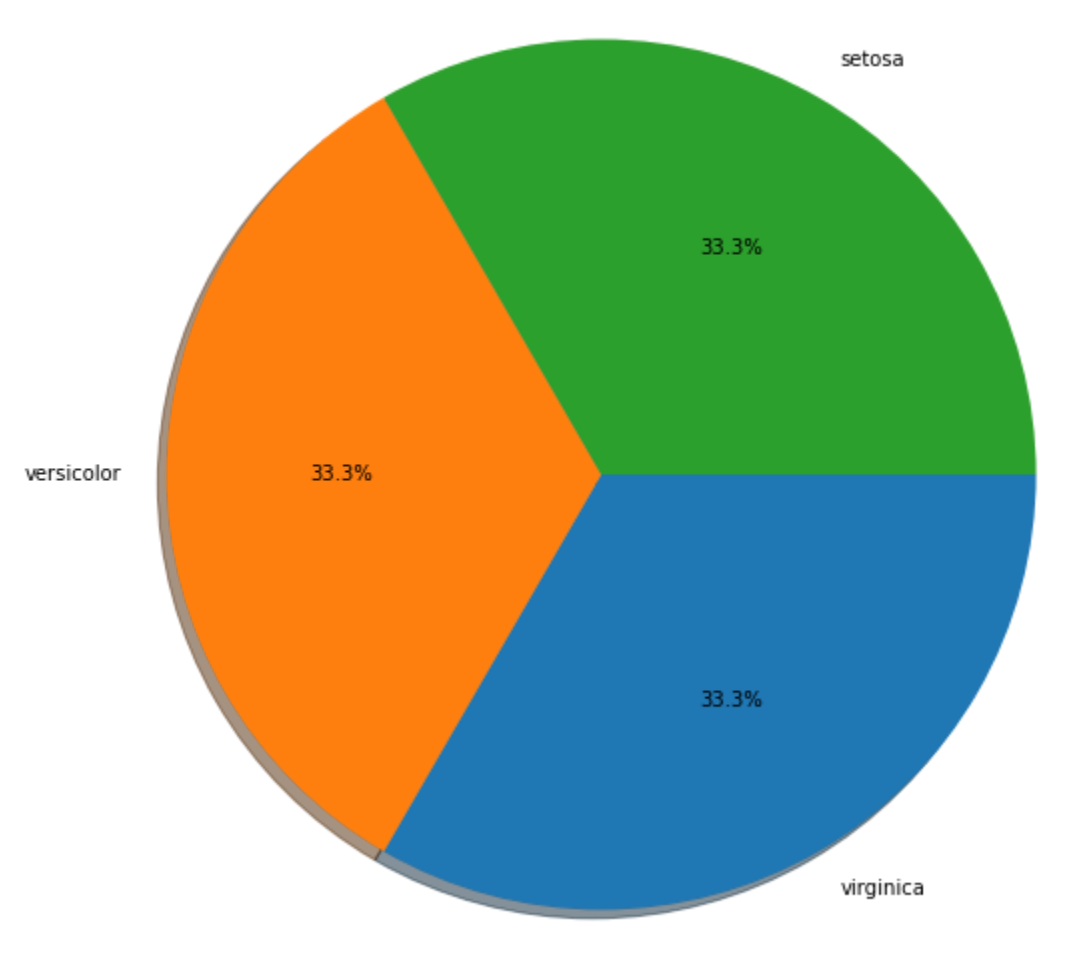
In [42]:
```
# checking target variable
sns.countplot(df['species'],data=df)
```

```
C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positi
onal argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

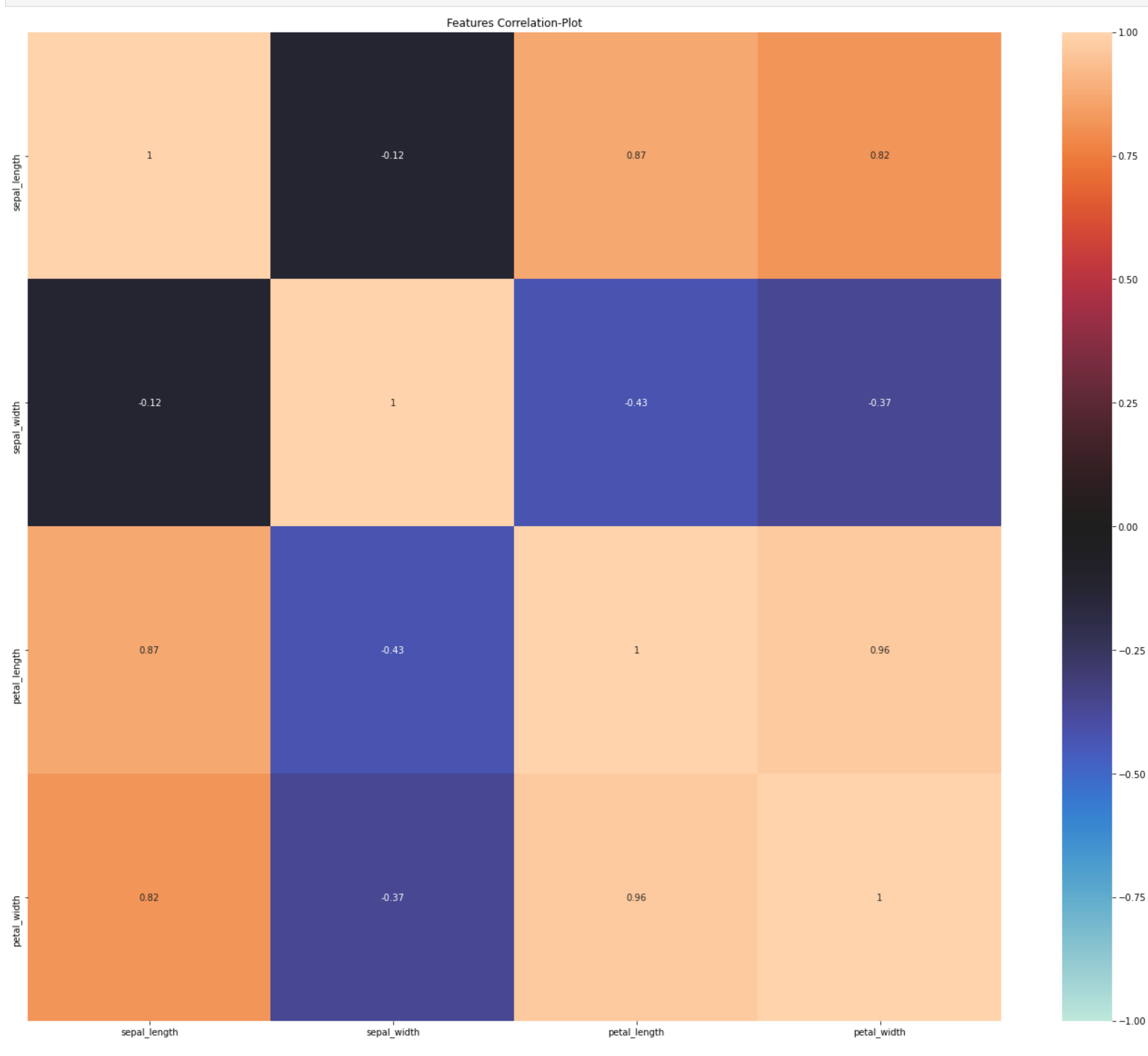
Out[42]:
```
<AxesSubplot:xlabel='species', ylabel='count'>
```



In [43]:
```
plt.figure(figsize=(10,10))
plt.pie(df['species'].value_counts(),
labels=df['species'].value_counts().index, counterclock=False,
shadow=True, autopct='%1.1f%%', radius=1, startangle=0)
plt.show()
```



In [47]:
```
B= df.corr()
plt.figure(figsize=[24,20])
plt.title('Features Correlation-Plot')
sns.heatmap(k, vmin=-1, vmax=1, center=0, annot=True)
plt.show()
```



In [5]:
```
# input
x = df.drop(columns='species')
```

In [6]:
```
x
```

Out[6]:

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

150 rows × 4 columns

In [7]:
```
#output
y = df['species']
```

In [8]:
```
Y = y.replace({'setosa':0,'virginica':1,'versicolor':2})
Y
```

Out[8]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    1
146    1
147    1
148    1
149    1
Name: species, Length: 150, dtype: int64
```

In [9]:
```
# spliting the data
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
x_train,x_test,y_train,y_test =train_test_split(x,Y,train_size=0.8)
```

In [12]:
```
#using DecisionTreeClassifier
```

In [13]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [14]:
```
dt = DecisionTreeClassifier()
dt
```

Out[14]:
```
DecisionTreeClassifier()
```

In [15]:
```
#fit the model
```

In [16]:
```
dt.fit(x_train,y_train)
```

Out[16]:
```
DecisionTreeClassifier()
```

In [17]:
```
#check score
```

In [18]:
```
dt.score(x_train,y_train)*100
```

Out[18]:
```
100.0
```

In [19]:
```
dt.score(x_test,y_test)*100
```

Out[19]:
```
96.66666666666667
```

In [20]:
```
y_pred = dt.predict(x_test)
```

In [21]:
```
y_pred
```

Out[21]:
```
array([2, 0, 0, 0, 1, 0, 0, 1, 2, 1, 0, 0, 2, 1, 1, 0, 2, 1, 2, 2, 2, 2,
       0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [23]:
```
# check the accuracy score
```

In [24]:
```
from sklearn.metrics import accuracy_score
```

In [25]:
```
accuracy_score(y_test,y_pred)*100
```

Out[25]:
```
96.66666666666667
```

In [26]:
```
#create an object of random forest algorithm
```

In [27]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [28]:
```
# fit the model
```

In [29]:
```
model = RandomForestClassifier()
model.fit(x_train,y_train)
```

Out[29]:
```
RandomForestClassifier()
```

In [30]:
```
#check the score
```

In [31]:
```
model.score(x_train,y_train)*100
```

Out[31]:
```
100.0
```

In [32]:
```
model.score(x_test,y_test)*100
```

Out[32]:
```
96.66666666666667
```

In [33]:
```
y_pred = model.predict(x_test)
y_pred
```

Out[33]:
```
array([2, 0, 0, 0, 1, 0, 0, 1, 2, 1, 0, 0, 2, 1, 1, 0, 2, 1, 2, 2, 2, 2,
       0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [34]:
```
# check the accuracy score
```

In [35]:
```
from sklearn.metrics import accuracy_score
```

In [36]:
```
accuracy_score(y_test,y_pred)*100
```

Out[36]:
```
96.66666666666667
```

In [37]:
```
#checking precission ,accuracy ,recall
from sklearn.metrics import classification_report
```

In [49]:
```
print(classification_report(y_test,y_pred))
```

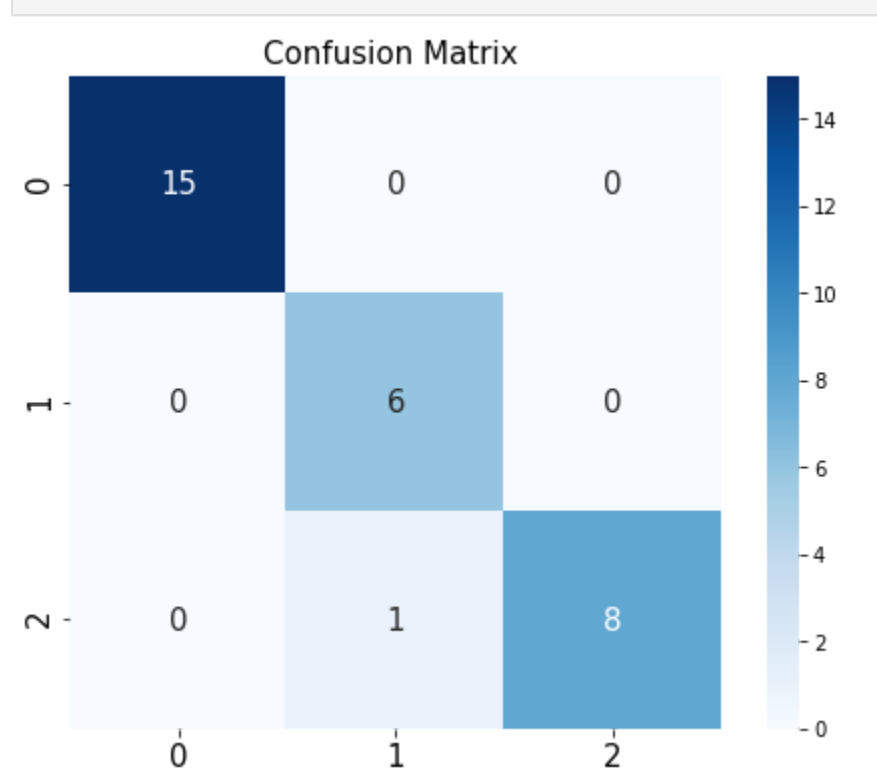
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.86      1.00      0.92         6
           2       1.00      0.89      0.94         9

    accuracy                           0.97        30
   macro avg       0.95      0.96      0.95        30
weighted avg       0.97      0.97      0.97        30
```

In [48]:
```
#confusion matrix
from sklearn.metrics import confusion_matrix
```

In [51]:
```
model=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(model, annot = True, cmap = 'Blues', annot_kws = {'size': 15}, square = True)
plt.title('Confusion Matrix ', size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.show()
```



In [ ]: