Iris Dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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

In [3]:
```
df.head()
```

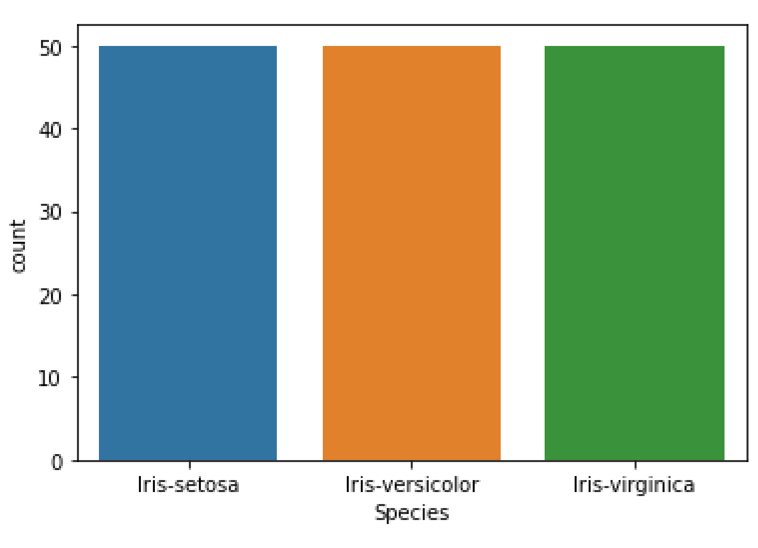
Out[3]:

	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

Data Visualization

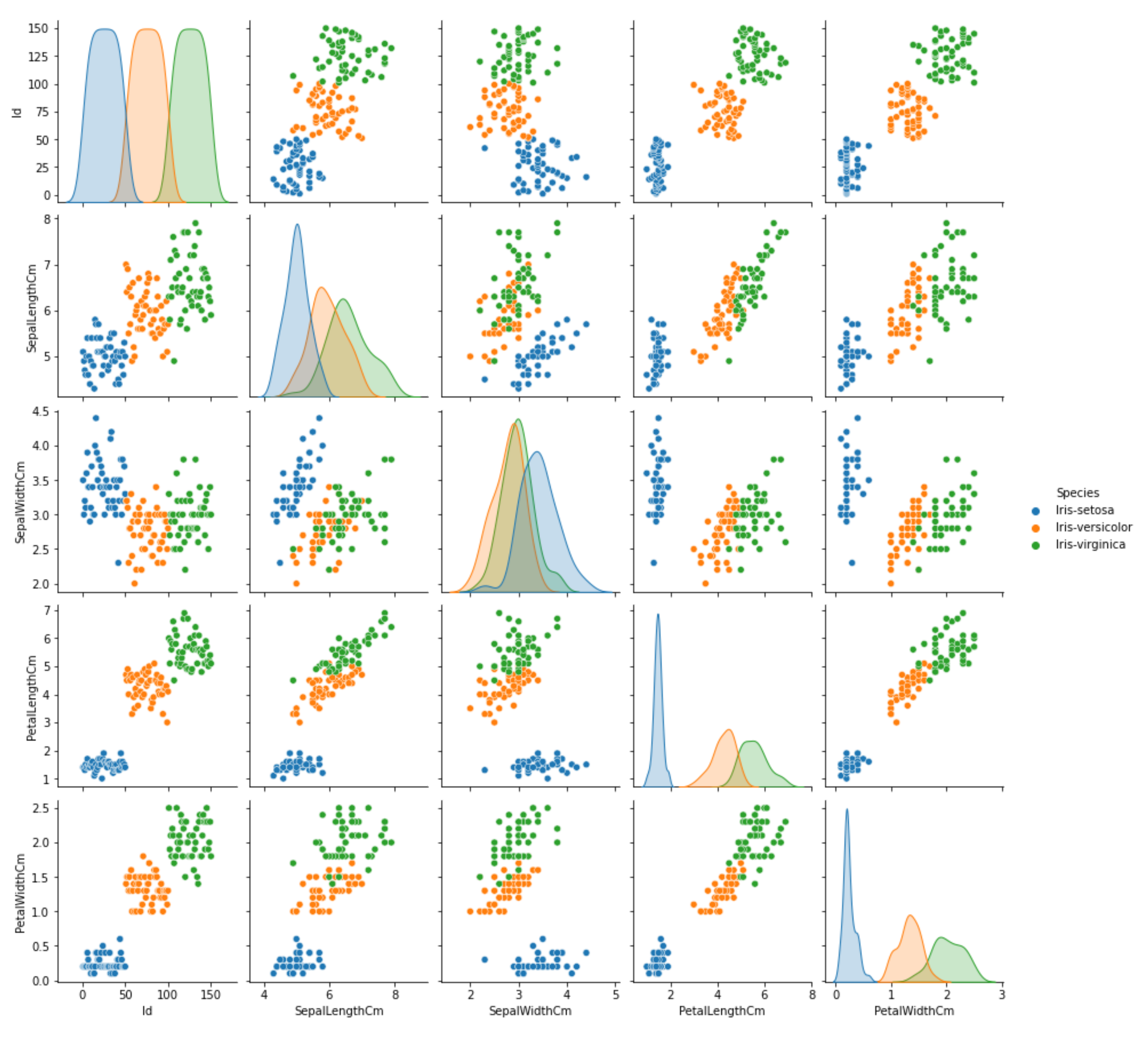
In [4]:
```
sns.countplot(x=df['Species'])
```

Out[4]: <AxesSubplot:xlabel='Species', ylabel='count'>



In [7]:
```
sns.pairplot(df,hue='Species')
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x19a3f38e7c0>



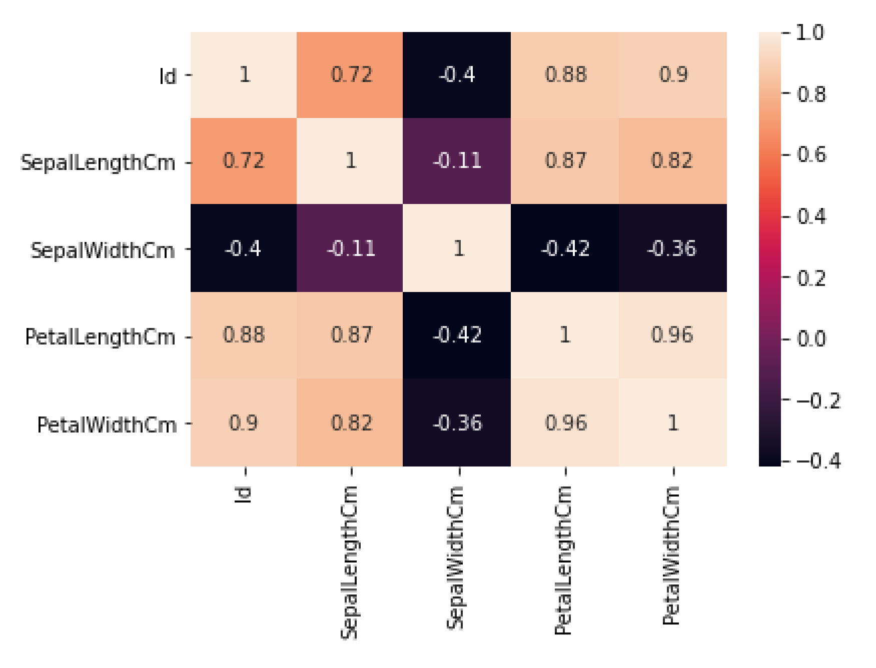
In [10]:
```
df.corr()
```

Out[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

In [12]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[12]: <AxesSubplot:>



Data Cleaning

In [13]:
```
df.isna().sum()
```

Out[13]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

no null values

Id in the table is not of any use so we will drop it

In [14]:
```
df.drop(['Id'],axis=1,inplace=True)
```

Now we need to convert the string values to neumeric by getting dummy or using label Encoder

In [16]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(df['Species'])
df['Species'] = le.transform(df['Species'])
```

In [17]:
```
df.head()
```

Out[17]:

	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

Now we have a clean data which can be used to build a model

Model Building

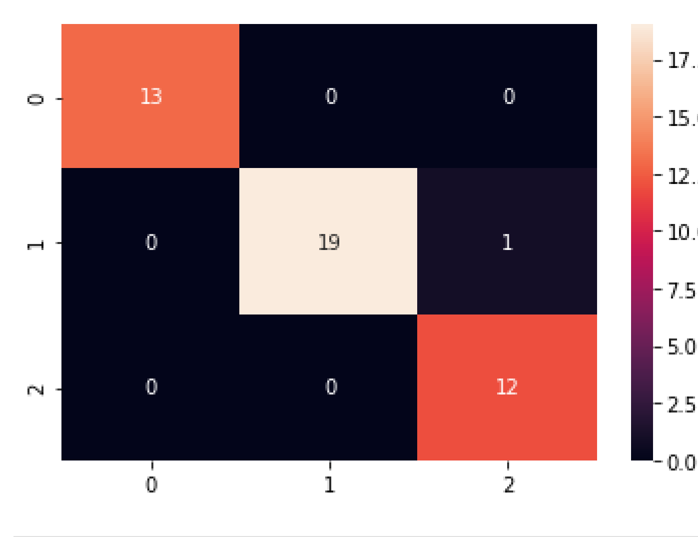
In [20]:
```
x = df.drop(['Species'],axis=1)
y = df['Species']
```

In [21]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7,random_state=101)
```

1) Logistic Regression

In [23]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix

lr = LogisticRegression()
lr.fit(x_train,y_train)
lr_predict = lr.predict(x_test)
sns.heatmap(confusion_matrix(y_test,lr_predict),annot=True)
```

Out[23]: <AxesSubplot:>



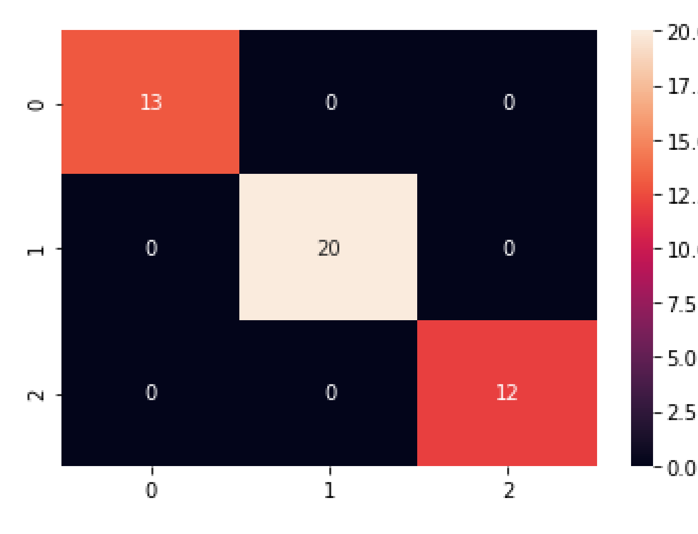
In [24]:
```
accuracy_score(y_test,lr_predict)
```

Out[24]: 0.9777777777777777

2) k-Nearest Neighbors

In [26]:
```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier(n_neighbors=10)
kn.fit(x_train,y_train)
kn_predict = kn.predict(x_test)
sns.heatmap(confusion_matrix(y_test,kn_predict),annot=True)
```

Out[26]: <AxesSubplot:>



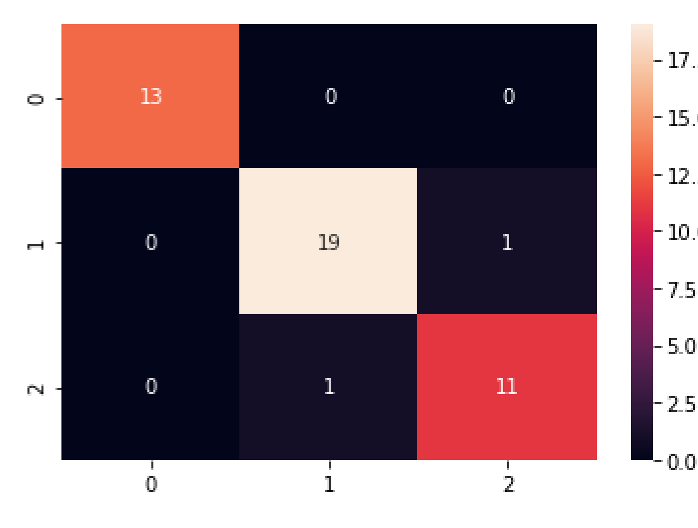
In [27]:
```
accuracy_score(y_test,kn_predict)
```

Out[27]: 1.0

3) Navie Bayes

In [30]:
```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(x_train,y_train)
nb_predict = nb.predict(x_test)
sns.heatmap(confusion_matrix(y_test,nb_predict),annot=True)
```

Out[30]: <AxesSubplot:>



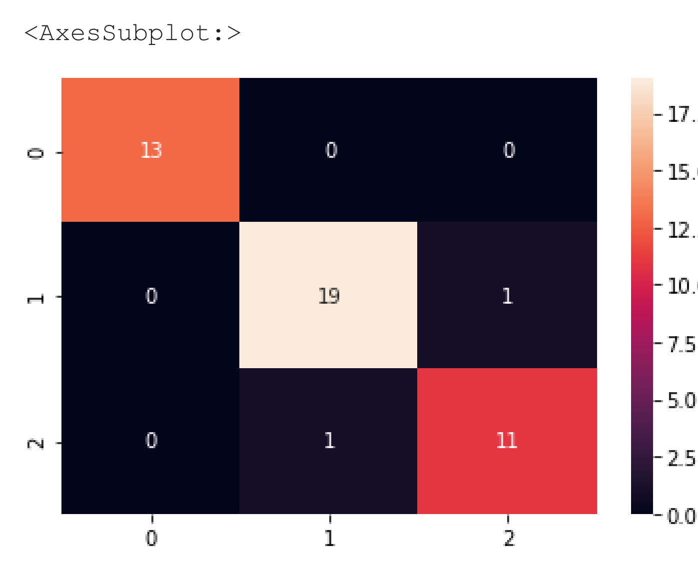
In [31]:
```
accuracy_score(y_test,nb_predict)
```

Out[31]: 0.9555555555555556

4)Decision Tree

In [32]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_predict = dt.predict(x_test)
dt_cm = confusion_matrix(y_test,dt_predict)
sns.heatmap(dt_cm,annot=True)
```

Out[32]: <AxesSubplot:>



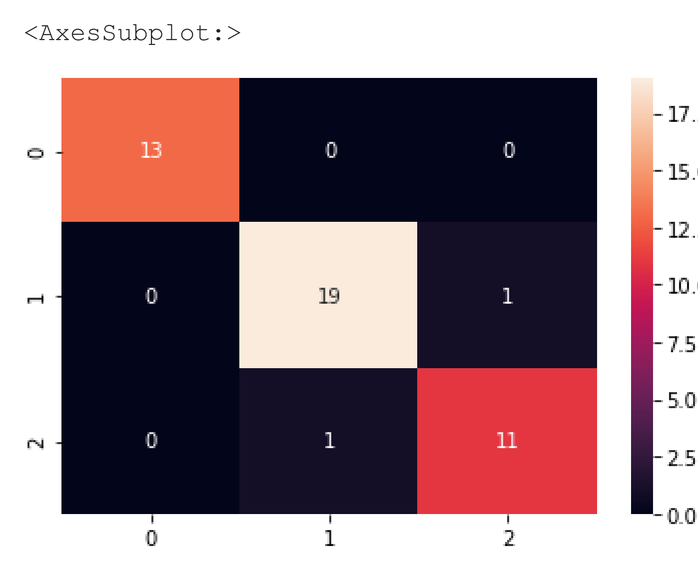
In [33]:
```
accuracy_score(y_test,dt_predict)
```

Out[33]: 0.9555555555555556

5)Random Forest

In [34]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf_predict = rf.predict(x_test)
rf_cm = confusion_matrix(y_test,rf_predict)
sns.heatmap(rf_cm,annot=True)
```

Out[34]: <AxesSubplot:>



In [35]:
```
accuracy_score(y_test,rf_predict)
```

Out[35]: 0.9555555555555556

in all the above Model we got higesht accuracy of KNN model of 100%

In [ ]: