TASK -1 IRIS FLOWER CLASSIFICATION

By - Anupreksha Shukla

Here I use Three Machine Learning Model to compare there accuracy Score

1)Logistic Regression

2)Decision Tree Classifier

3)Random Forest Classifier

1- Import the Libraries

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

2 -Read The Dataset

In [2]:
```
df_iris = pd.read_csv("C://Users//HP//Downloads//Iris1.csv")
```

In [3]:
```
df_iris
```

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

Analyse the Dataset

Classification Models

Info Method Prints information about the dataframe

In [4]:
```
df_iris.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
```

In [5]:
```
df_iris['Species'].value_counts()
```

Out[5]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [6]:
```
df_iris.describe()
```

Out[6]:

	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

In [7]:
```
df_iris.nunique()
```

Out[7]:
```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

In [8]:
```
df_iris.head()
```

Out[8]:

	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

In [9]:
```
df_iris.tail()
```

Out[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [10]:
```
df_iris.isnull().sum()
```

Out[10]:
```
Id              0
SepalLengthCm   0
SepalWidthCm    0
PetalLengthCm   0
PetalWidthCm    0
Species         0
dtype: int64
```

In [11]:
```
df_iris
```

Out[11]:

	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

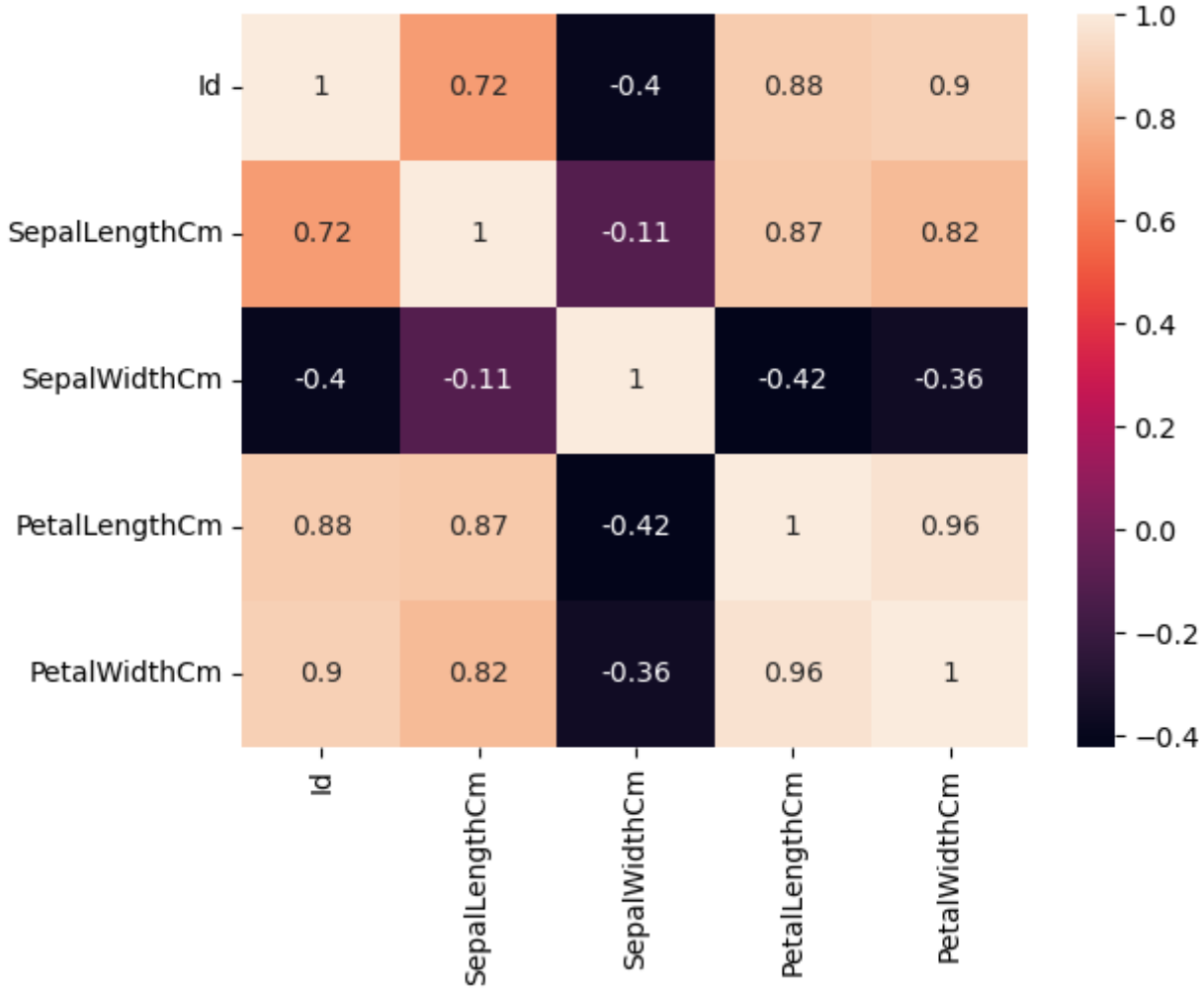
In [12]:
```
corr=df_iris.corr()
corr
```

Out[12]:

	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 [13]:
```
sns.heatmap(corr,annot=True)
```

Out[13]:
```
<AxesSubplot:>
```



Defining dependent and independent variable

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df_iris['Species']=le.fit_transform(df_iris['Species'])
df_iris['Species'].value_counts()
```

Out[14]:
```
0    50
1    50
2    50
Name: Species, dtype: int64
```

In [15]:
```
df_iris.head(2)
```

Out[15]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0

In [16]:
```
x=df_iris.iloc[:,1:5]
y=df_iris['Species']
x.shape,y.shape
```

Out[16]:
```
((150, 4), (150,))
```

splitting x and y in to train and test dataset

In [17]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[17]:
```
((112, 4), (38, 4), (112,), (38,))
```

Train Test Split

In [18]:
```
x_train.head(2)
```

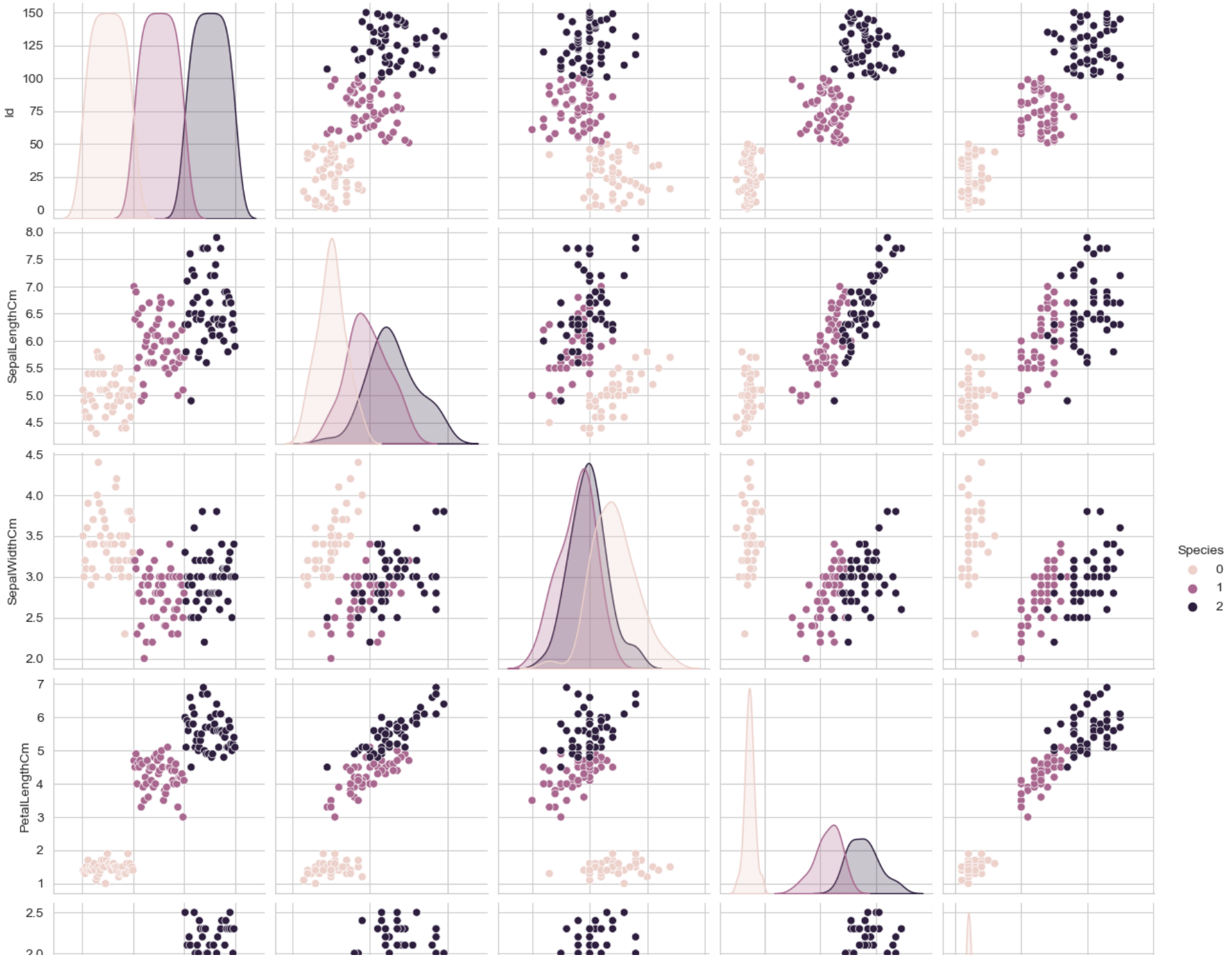
Out[18]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
85	6.0	3.4	4.5	1.6
81	5.5	2.4	3.7	1.0

In [19]:
```
y_test.head(2)
```

Out[19]:
```
12     0
115    2
Name: Species, dtype: int32
```

In [20]:
```
sns.set_style("whitegrid")
sns.pairplot(df_iris,hue="Species");
plt.show()
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
Species
0
1
2

Import the model/Algorithm

LogisticRegression

In [21]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

In [22]:
```
#Train model with x_train and y_train
lr.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression()
```

Predict with x_train and y_train

In [23]:
```
y_pred_lr=lr.predict(x_test)
y_pred_lr[:5],y_test.values[:5]
```

Out[23]:
```
(array([0, 2, 0, 0, 0]), array([0, 2, 0, 0, 0]))
```

Evaluate the Model

In [24]:
```
print(lr.score(x_train,y_train))
print(lr.score(x_test,y_test))
```

```
0.9642857142857143
1.0
```

In [25]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
print(confusion_matrix(y_pred_lr,y_test))
print(classification_report(y_pred_lr,y_test))
print(f'model_score- {lr.score(x_test,y_test)}')
print(f'accuracy_score- { accuracy_score(y_pred_lr,y_test)}')
```

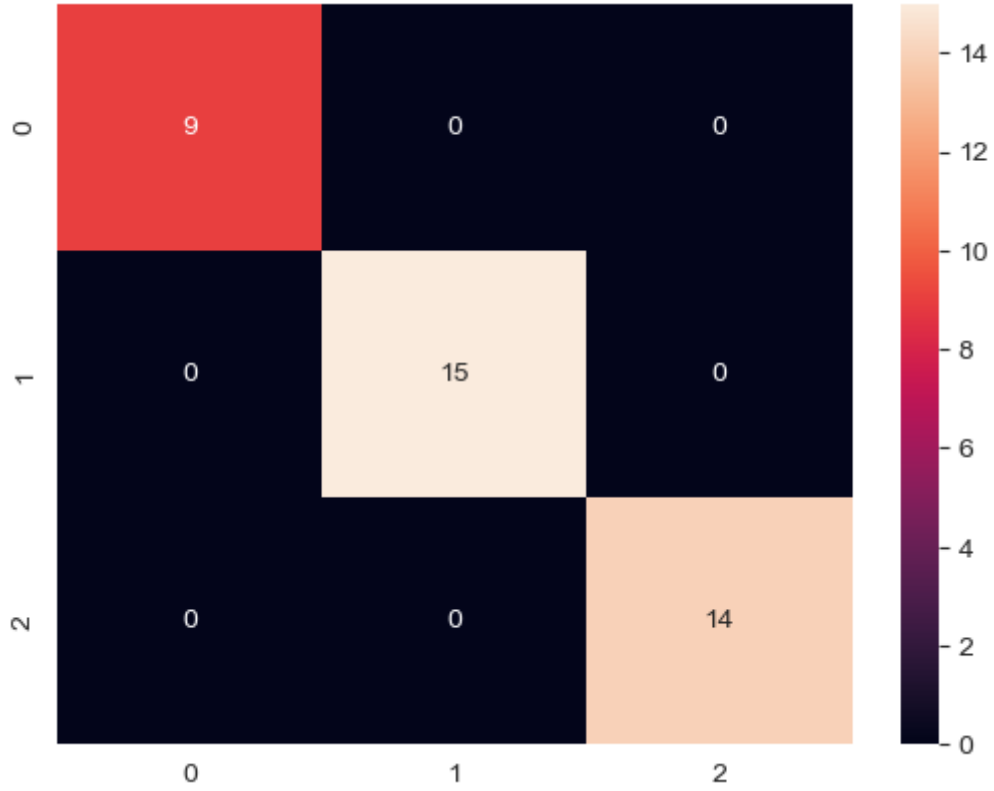
```
[[ 9  0  0]
 [ 0 15  0]
 [ 0  0 14]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        15
           2       1.00      1.00      1.00        14

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38

model_score- 1.0
accuracy_score- 1.0
```

In [26]:
```
cm = confusion_matrix(y_pred_lr,y_test)
sns.heatmap(cm,annot=True)
plt.show()
```

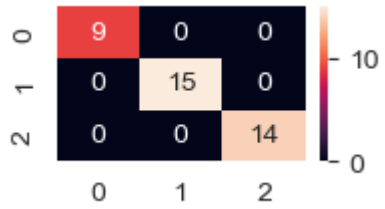


DECISION TREE CLASSIFIER

In [27]:
```
#-----------------USING DECISION TREE CLASSIFIER --------------------------------
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred_dtc=dtc.predict(x_test)
print(f'predicted y is:{y_pred_dtc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_dtc,y_test))
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
cm=confusion_matrix(y_pred_dtc,y_test)
plt.figure(figsize=(2,1))
print(sns.heatmap(cm,annot=True))
plt.show()
print(classification_report(y_pred_dtc,y_test))
```

```
print(f'model_score-{dtc.score(x_test,y_test)}')
print(f'accuracy_score - { accuracy_score(y_pred_dtc,y_test)}')
ax=plt.axes(projection = '3d')
ax.scatter3D(x_test['PetalLengthCm'],x_test['PetalWidthCm'],y_test)
ax.scatter3D(x_test['PetalLengthCm'],x_test['PetalWidthCm'],y_pred_dtc,'black')
plt.show()
```

```
predicted y is:[0 2 0 0 0] Actual_y[0 2 0 0 0]
[[ 9  0  0]
 [ 0 15  0]
 [ 0  0 14]]
AxesSubplot(0.125,0.11;0.62x0.77)
```

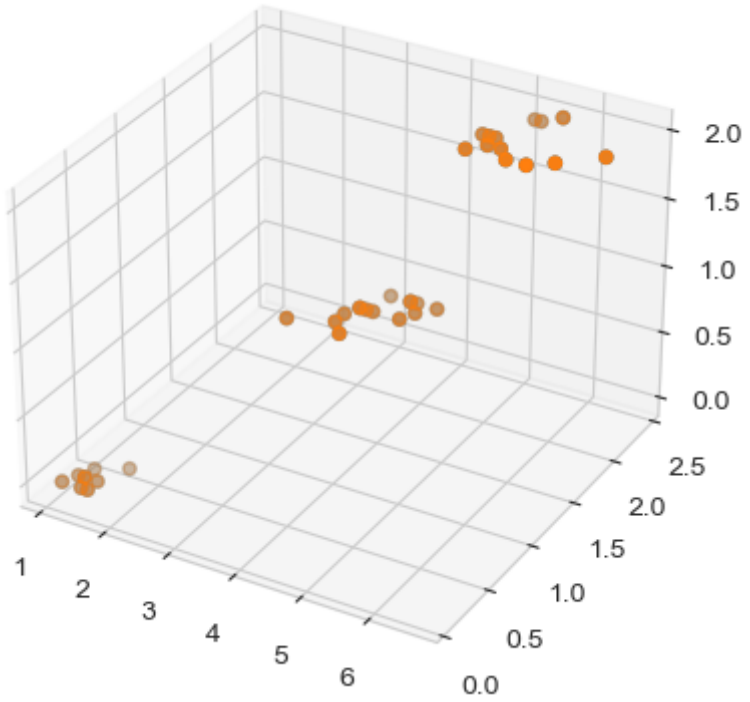



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        15
           2       1.00      1.00      1.00        14

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38

model_score-1.0
accuracy_score - 1.0
```



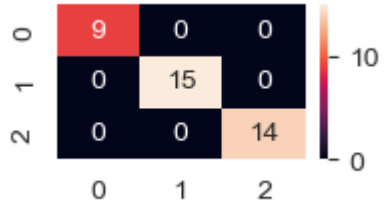
RANDOM FOREST CLASSIFIER

In [28]:
```
# -----------------USING RANDOM FOREST CLSASSIFIER----------------


from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred_rfc=rfc.predict(x_test)
print(f'predicted y is:{y_pred_dtc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_rfc,y_test))
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
#cm=confusion_matrix(y_pred_dtc,y_test)
plt.figure(figsize=(2,1))
print(sns.heatmap(cm,annot=True))
plt.show()
```

```
print(classification_report(y_pred_dtc,y_test))
print(f'model_score-{dtc.score(x_test,y_test)}')
print(f'accuracy_score - { accuracy_score(y_pred_rfc,y_test)}')
ax=plt.axes(projection = '3d')
ax.scatter3D(x_test['PetalLengthCm'],x_test['PetalWidthCm'],y_test)
ax.scatter3D(x_test['PetalLengthCm'],x_test['PetalWidthCm'],y_pred_dtc,'black')
plt.show()
```

```
predicted y is:[0 2 0 0 0] Actual_y[0 2 0 0 0]
[[ 9  0  0]
 [ 0 15  0]
 [ 0  0 14]]
AxesSubplot(0.125,0.11;0.62x0.77)
```

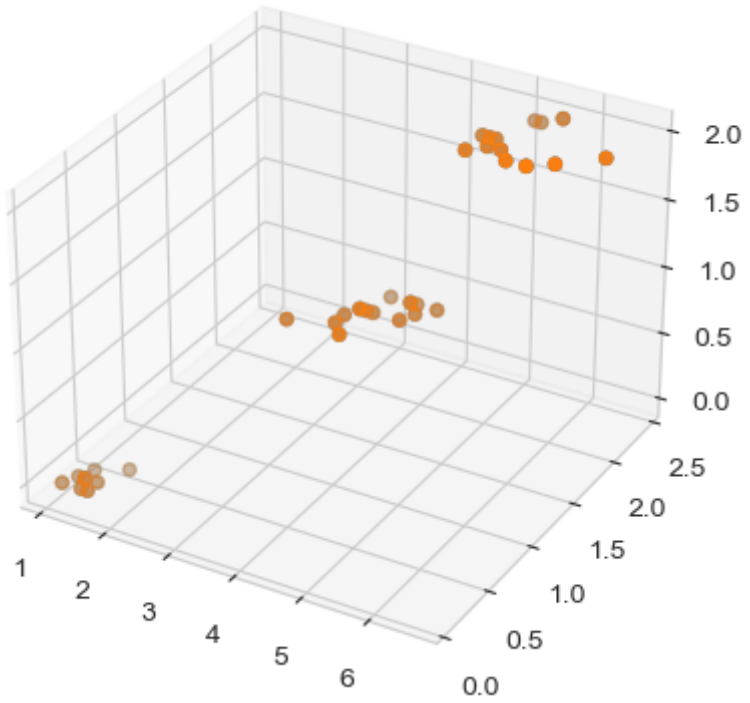



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        15
           2       1.00      1.00      1.00        14

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38

model_score-1.0
accuracy_score - 1.0
```


2.0
1.5
1.0
0.5
0.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6