In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
iris_df = pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\ML\Amit Mishr
```

In [3]:

```
iris_df.head()
```

Out[3]:

	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

In [4]:

```
iris_df["species"].unique()
```

Out[4]:

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

Data Wrangling

In [5]:

```
iris_df.isnull().sum()
```

Out[5]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

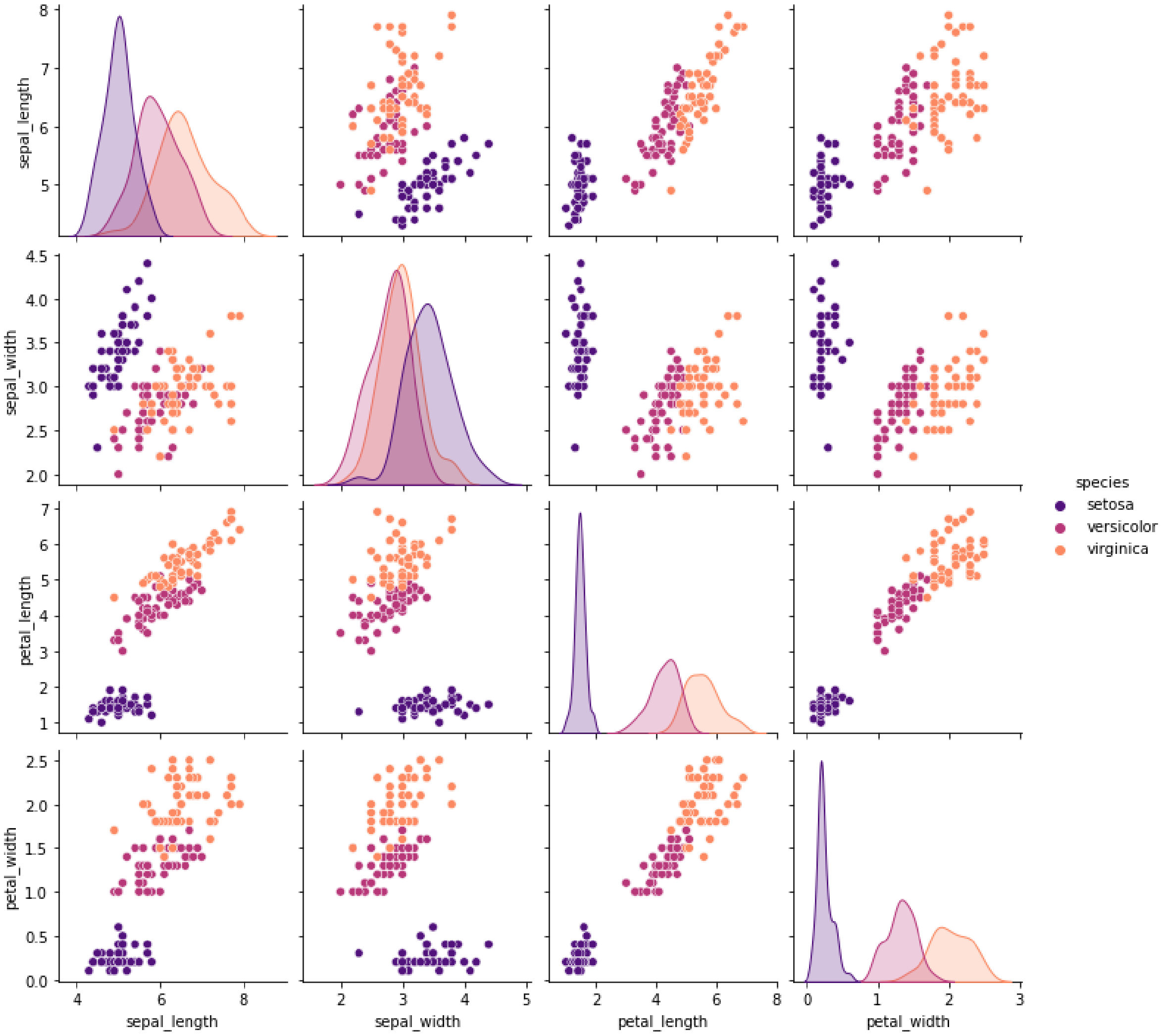
Pairplot- To find what type of relation is there between within the features, likewise kind of corelation plot

In [6]:

```
sns.pairplot(data=iris_df, hue="species",palette="magma")
```

Out[6]:

```
<seaborn.axisgrid.PairGrid at 0x21225c123a0>
```



Cross Validation

- to detect overfitting, ie, failing to generalize a pattern.

In [8]:
```
iris_df.shape
```

Out[8]:
```
(150, 5)
```

In [9]:
```
X= iris_df.drop("species",axis=1)
Y= iris_df["species"]
```

Train and Test Split

In [10]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X, Y, test_size=0.3, random_state =
```

Support Vector Classifier

Kernel Trick="Linear"

- Kernel = Linear
- C value = Higher the c value, decision boundary will be hard and vice versa

- Margins = This eg is multi class classification - no of classes bcz target species - are 3 classification, if target is 2 it's Binary Class classification
- Multivariate - Because 5 dimensions incldung species
- Margins - For multi class classification, chosse option btwn OVR (One versus rest all), and OVO (One versus another class)

In [11]:
```
from sklearn.svm import SVC
# kernel - shape of classifier
# linear trick is used for binary class classification
svc_model=SVC(C = 1e4, kernel="linear")
# by default e = value will be 1, we can increase it, 1e4= 10 to the power 4
```

In [12]:
```
svc_model.fit(x_train,y_train)
```

Out[12]:
```
SVC(C=10000.0, kernel='linear')
```

In [13]:
```
# Accuracy matrix
svc_model.score(x_test, y_test)
```

Out[13]:
```
1.0
```

In [14]:
```
# Predictions by using predict function
predictions=svc_model.predict(x_test)
```

In [15]:
```
predictions
```

Out[15]:
```
array(['setosa', 'setosa', 'virginica', 'setosa', 'setosa', 'virginica',
       'setosa', 'virginica', 'virginica', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'setosa', 'virginica', 'virginica',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'setosa',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'setosa'], dtype=object)
```

In [16]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

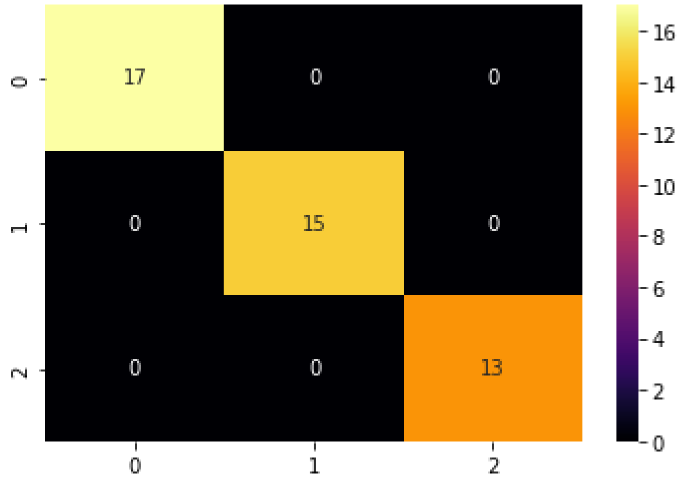
In [17]:
```
confusion_matrix(y_test,predictions)
```

Out[17]:
```
array([[17,  0,  0],
       [ 0, 15,  0],
       [ 0,  0, 13]], dtype=int64)
```

From the above fig, none of them are misclassified- diagonal side as correct prediction and non diagonal as 0

In [20]:
```
sns.heatmap(confusion_matrix(y_test,predictions),annot=True, fmt= "0.0f", cmap="in
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        17
  versicolor       1.00      1.00      1.00        15
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

kernel Trick = 'RBF'

In [25]:
```
from sklearn.svm import SVC
# kernel = shape of classifier
# decision_function_shape = 'ovr' or 'ovo'
svc_model=  SVC(C = 1e4, kernel= "rbf", decision_function_shape="ovr")
```

In [26]:
```
svc_model.fit(x_train,y_train)
```

Out[26]:
```
SVC(C=10000.0)
```

In [27]:
```
svc_model.score(x_test,y_test)
```

Out[27]:
```
0.9777777777777777
```

In [28]:
```
predictions=svc_model.predict(x_test)
```

In [29]:
```
predictions
```

Out[29]:
```
array(['setosa', 'setosa', 'virginica', 'setosa', 'setosa', 'virginica',
       'setosa', 'virginica', 'virginica', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'virginica', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'virginica', 'virginica', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'setosa'], dtype=object)
```

In [30]:
```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(y_test, predictions)
```

Out[30]:

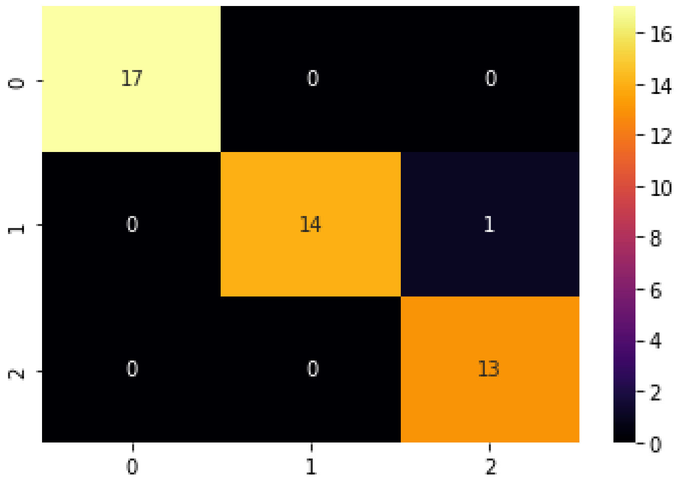
```
array([[17,  0,  0],
       [ 0, 14,  1],
       [ 0,  0, 13]], dtype=int64)
```

In [31]:

```
sns.heatmap(confusion_matrix(y_test, predictions),annot=True,cmap="inferno")
```

Out[31]:

```
<AxesSubplot:>
```



Kernel = 'Polynomial'

- Hyperplane will be non-linear

In [43]:

```
from sklearn.svm import SVC
# kernel = shape of classifier
# degree = 2, 3, 4
svc_model = SVC(C = 1e3, kernel="poly", degree =2 )
```

In [44]:

```
svc_model.fit(x_train,y_train)
```

Out[44]:

```
SVC(C=1000.0, degree=2, kernel='poly')
```

In [45]:

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

Out[45]:

```
1.0
```

In [46]:

```
predictions = svc_model.predict(x_test)
predictions
```

Out[46]:

```
array(['setosa', 'setosa', 'virginica', 'setosa', 'setosa', 'virginica',
       'setosa', 'virginica', 'virginica', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'virginica', 'versicolor', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'setosa', 'virginica', 'virginica',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'setosa',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'versicolor',
       'versicolor', 'virginica', 'versicolor', 'setosa'], dtype=object)
```

In [48]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

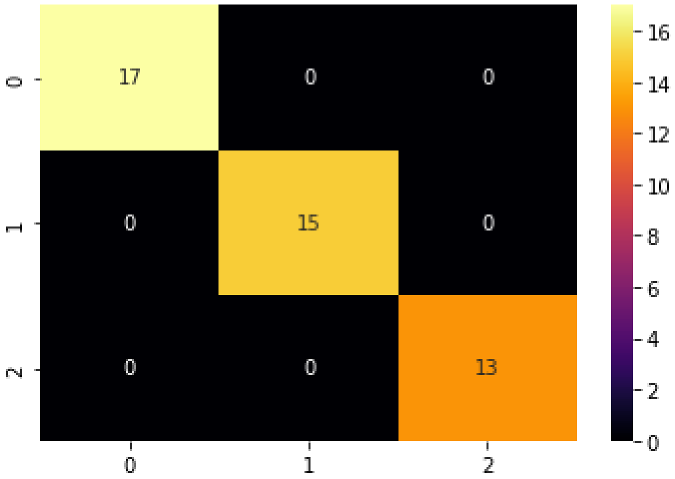
In [49]:

```
confusion_matrix(y_test, predictions)
```

```
Out[49]: array([[17,  0,  0],
                [ 0, 15,  0],
                [ 0,  0, 13]], dtype=int64)
```

```
In [50]: sns.heatmap(confusion_matrix(y_test, predictions), annot= True, fmt = '0.0f', cmap
```

```
Out[50]: <AxesSubplot:>
```



```
In [ ]:
```