Support Vector Machine

In [ ]:
```
#used for both classification or regression challenge
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [5]:
```
from sklearn.datasets import load_breast_cancer
```

In [6]:
```
cancer= load_breast_cancer()
```

In [7]:
```
cancer.keys()
```

Out[7]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_mod
ule'])
```

In [8]:
```
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
```

In [9]:
```
cancer['feature_names']
```

Out[9]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [10]:
```
df_feat=pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
```

In [12]:
```
df_feat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

In [13]:
```
cancer['target']
```

Out[13]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [16]:
```
df_target=pd.DataFrame(cancer['target'],columns=['Cancer'])
```

In [17]:
```
df_target.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Cancer  569 non-null    int32
dtypes: int32(1)
memory usage: 2.3 KB
```

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
x_train,x_test,y_train,y_test=train_test_split(df_feat,np.ravel(df_target),test_size=0.30,random_state
```

In [26]:
```
from sklearn.svm import SVC
```

In [35]:
```
model=SVC(kernel='linear')
```

In [36]:
```
model.fit(x_train,y_train)
```

Out[36]:
```
SVC(kernel='linear')
```

In [37]:
```
y_pred=model.predict(x_test)
```

In [38]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [39]:
```
r = classification_report(y_test,y_pred)
print(r)
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.94        69
           1       0.95      0.97      0.96       102

    accuracy                           0.95       171
   macro avg       0.95      0.95      0.95       171
weighted avg       0.95      0.95      0.95       171
```

In [40]:
```
cm=confusion_matrix(y_test,y_pred).ravel()
```

In [41]:
```
cm
```

Out[41]:
```
array([64,  5,  3, 99], dtype=int64)
```

Example 2

In [43]:
```
ir=sns.load_dataset('iris')
ir.head()
```

Out[43]:

	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

Images

In [45]:
```
from IPython.display import Image
```

In [46]:
```
url='http://upload.wikimedia.org/wikipedia/commons/5/56/Kosaciec_szczecinkowaty_Iris_setosa.jpg'
```

In [47]:
```
Image(url,width=300,height=300)
```

Out[47]:

In [48]:
```
url1 = 'http://upload.wikimedia.org/wikipedia/commons/4/41/Iris_versicolor_3.jpg'
```

In [52]:
```
Image(url1,width=300,height=500)
```

Out[52]:

In [50]:
```
url2 = 'http://upload.wikimedia.org/wikipedia/commons/9/9f/Iris_virginica.jpg'
```

In [53]:
```
Image(url2,width=300,height=1000)
```

Out[53]:

In [57]:
```
dor = 'http://3.bp.blogspot.com/-Z7S4ovIP1lA/T78D3tX87pI/AAAAAAAAHPI/4O6WDjN-uhM/s1600/Doraemon+10.jpg'
```

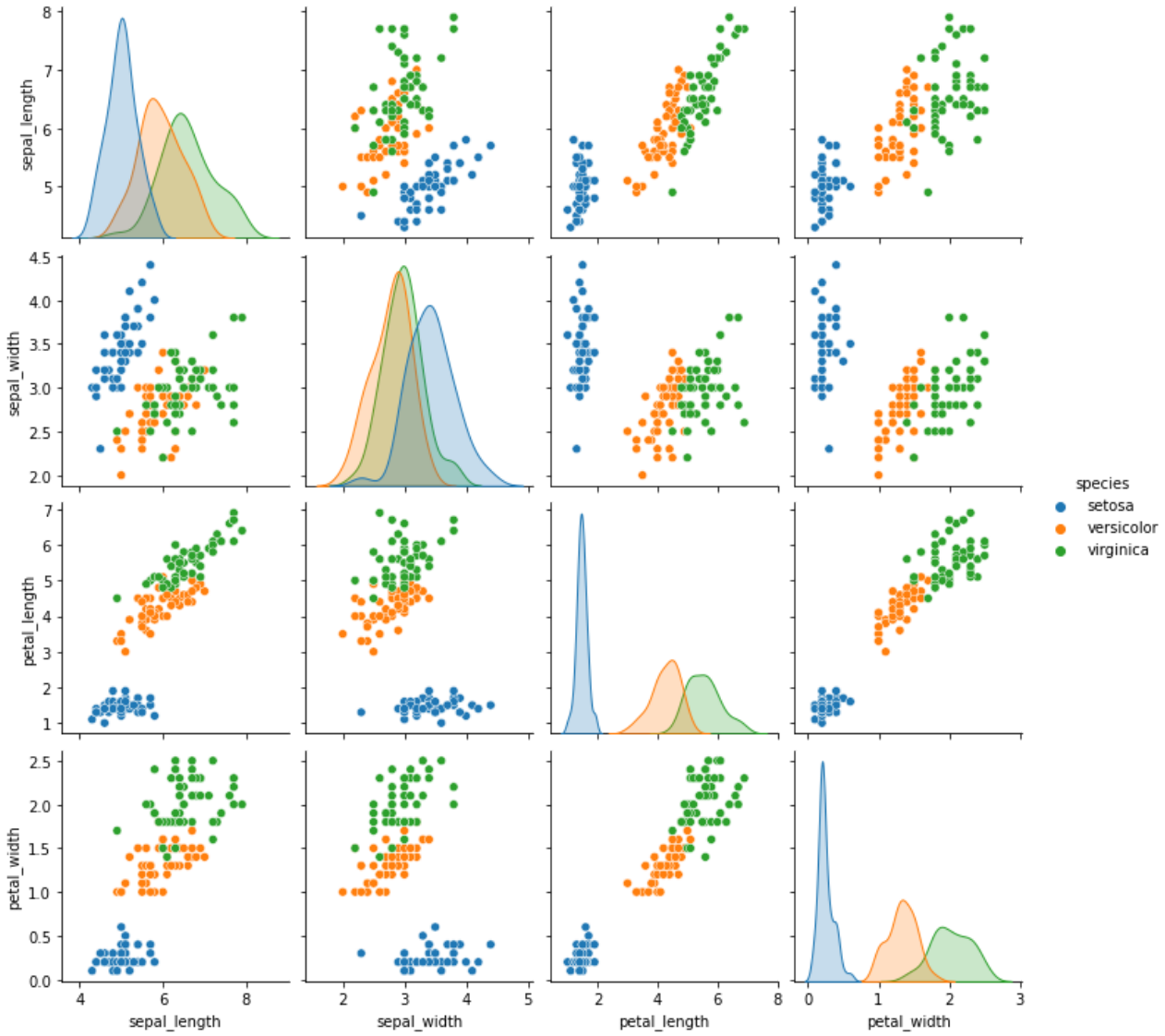
In [58]:
```
Image(dor,width=300,height=1000)
```

Out[58]:

In [ ]:
```
#pairplot to see which species can be created
```

In [61]:
```
sns.pairplot(ir,hue='species')
```

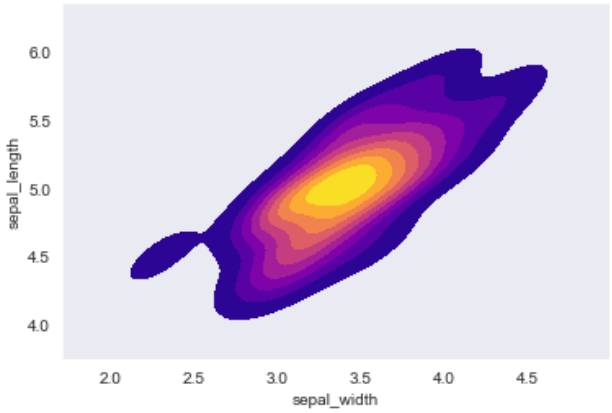
Out[61]:
```
<seaborn.axisgrid.PairGrid at 0x24c2575ad00>
```



Create kde plot of sepal_length versus sepal_width for setosa species of flower

In [64]:
```
setosa=ir[ir['species']=='setosa']
sns.kdeplot(x=setosa['sepal_width'],y=setosa['sepal_length'],cmap='plasma',shade=True,thresh=0.05)
```

Out[64]: <AxesSubplot:xlabel='sepal_width', ylabel='sepal_length'>



In [ ]:
```
#Train test split
```

In [65]:
```
from sklearn.model_selection import train_test_split
```

In [78]:
```
x=ir.drop('species',axis=1)
y=ir['species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=100)
```

In [79]:
```
from sklearn.svm import SVC
```

In [80]:
```
model=SVC(kernel='linear')
```

In [81]:
```
model.fit(x_train,y_train)
```

Out[81]: SVC(kernel='linear')

In [82]:
```
y_pred=model.predict(x_test)
```

In [83]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [84]:
```
t = classification_report(y_test,y_pred)
print(t)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       1.00      1.00      1.00        11
   virginica       1.00      1.00      1.00        18

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

In [85]:
```
#Unsupervised Learning
```

K-Means Clustring

In [ ]:
```
#works on centeroid algorithm
```

In [ ]:
```
i
```

In [ ]:
```

```