Support Vector Machines with Python

Import Libraries

In [1]:

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

Get the Data

In [2]:

```
IRIS = pd.read_csv('IRIS.csv')
```

In [3]:

```
IRIS.keys()
```

Out[3]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [4]:

```
IRIS.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:

```
print(IRIS['petal_width'])
```

```
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petal_width, Length: 150, dtype: float64
```

In [6]:

```
print(IRIS['species'])
```

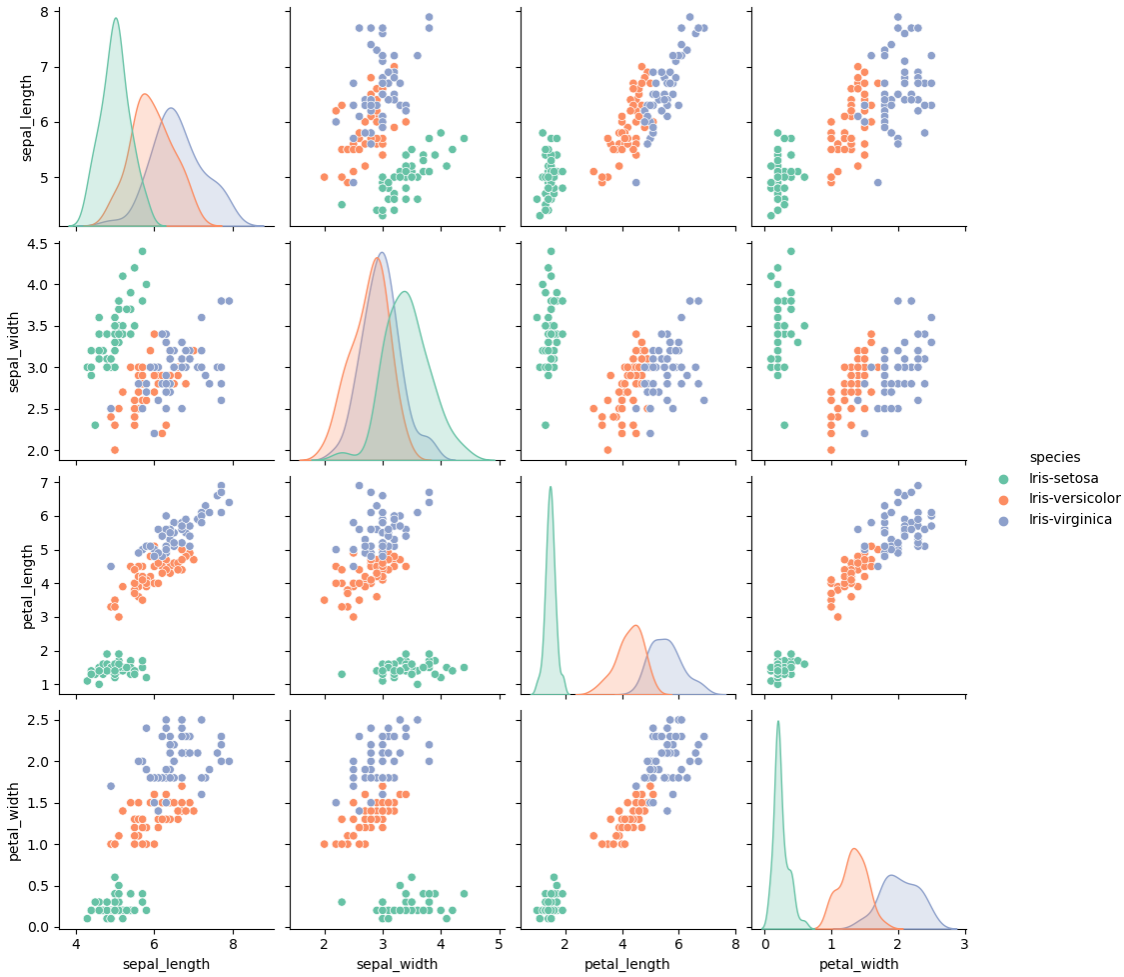
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: species, Length: 150, dtype: object
```

In [7]:

```
sns.pairplot(data=IRIS, hue='species', palette='Set2')
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1ef17977df0>
```



Set up DataFrame

In [8]:

```
df_IRIS = pd.DataFrame(IRIS['sepal_length'],columns=IRIS['species'])
df_IRIS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Columns: 150 entries, Iris-setosa to Iris-virginica
dtypes: object(150)
memory usage: 0.0+ bytes
```

In [9]:

```
IRIS['petal_width']
```

Out[9]:

```
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: petal_width, Length: 150, dtype: float64
```

In [10]:

```
df_petal_width = pd.DataFrame(IRIS['petal_width'],columns=['IRIS'])
```

Train Test Split

In [11]:

```
from sklearn.model_selection import train_test_split
```

In [12]:

```
x=IRIS.iloc[:,:-1]
y=IRIS.iloc[:,4]
x_train,x_test, y_train, y_test=train_test_split(x,y,test_size=0.30)
```

In [13]:

```
from sklearn.svm import SVC
model=SVC()
```

In [14]:

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

Out[14]:

SVC

SVC()

In [15]:

```
pred=model.predict(x_test)
```

Model Evaluation

In [16]:

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

In [17]:

```
print(confusion_matrix(y_test,pred))
```

```
[[20  0  0]
 [ 0 12  0]
 [ 0  2 11]]
```

In [18]:

```
print(classification_report(y_test, pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        20
Iris-versicolor       0.86      1.00      0.92        12
 Iris-virginica       1.00      0.85      0.92        13

       accuracy                           0.96        45
      macro avg       0.95      0.95      0.95        45
   weighted avg       0.96      0.96      0.96        45
```

Gridsearch

In [19]:

```
param_grid = {'C': [0.1,1,5, 10, 50, 100, 1000], 'gamma': [10,1,0.1,0.01,0.001,0.0001],
```

In [20]:

```
from sklearn.model_selection import GridSearchCV
```

In [21]:

```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=5)
```

In [22]:

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

```
me=   0.0s
[CV 1/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total ti
me=   0.0s
[CV 2/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.952 total ti
me=   0.0s
[CV 3/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total ti
me=   0.0s
[CV 4/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.905 total ti
me=   0.0s
[CV 5/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total ti
me=   0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.952 total ti
me=   0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.952 total ti
me=   0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total ti
me=   0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.905 total ti
me=   0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total ti
```

In [23]:

```
grid.best_params_
```

Out[23]:

```
{'C': 5, 'gamma': 10, 'kernel': 'linear'}
```

In [24]:

```
grid.best_estimator_
```

Out[24]:

```
SVC(C=5, gamma=10, kernel='linear')
```

In [25]:

```
grid_predictions = grid.predict(x_test)
```

In [26]:

```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[20  0  0]
 [ 0 11  1]
 [ 0  0 13]]
```

In [27]:

```
print(classification_report(y_test,grid_predictions))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        20
Iris-versicolor       1.00      0.92      0.96        12
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.98        45
      macro avg       0.98      0.97      0.97        45
   weighted avg       0.98      0.98      0.98        45
```

In [ ]:

```

```