Support Vector Machine Tutorial Using Python Sklearn

In [8]:

```python
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

In [9]:

```python
dir(iris)
```

Out[9]:

```
['DESCR', 'data', 'feature_names', 'filename', 'target', 'target_name
s']
```

In [10]:

```python
iris.feature_names
```

Out[10]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [12]:

```python
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df.head()
```

Out[12]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [13]:

```
df['target'] = iris.target
df.head()
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
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

In [14]:

```
iris.target_names
```

Out[14]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [19]:

```
df[df.target==2].head()
```

Out[19]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

In [20]:

```
df['flower_name']=df.target.apply(lambda x: iris.target_names[x])
df.head()
```

Out[20]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

In [21]:

```
from matplotlib import pyplot as plt
```

In [22]:

```
%matplotlib inline
```

In [23]:

```
df0 = df[df.target==0]
df1 = df[df.target==1]
df2 = df[df.target==2]
```

In [24]:

```
df2.head()
```

Out[24]:

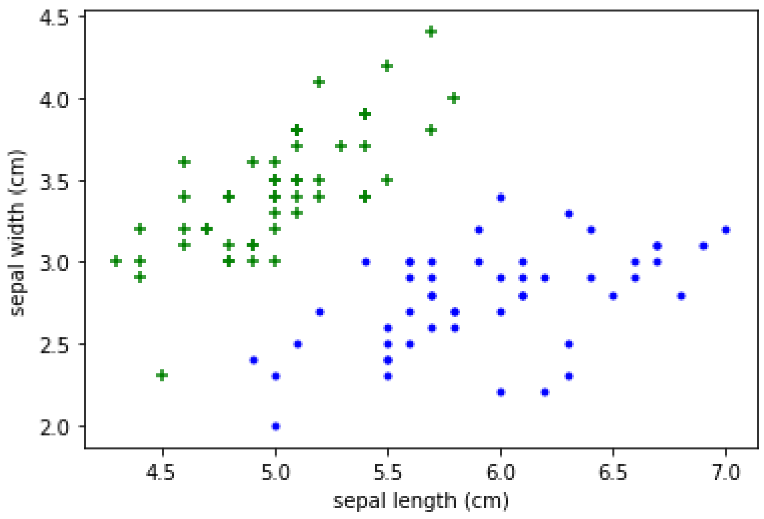
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
100	6.3	3.3	6.0	2.5	2	virginica
101	5.8	2.7	5.1	1.9	2	virginica
102	7.1	3.0	5.9	2.1	2	virginica
103	6.3	2.9	5.6	1.8	2	virginica
104	6.5	3.0	5.8	2.2	2	virginica

In [29]:

```
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.scatter(df0['sepal length (cm)'],df0['sepal width (cm)'],color='green',marker='+')
plt.scatter(df1['sepal length (cm)'],df1['sepal width (cm)'],color='blue',marker='.')
```

Out[29]:

```
<matplotlib.collections.PathCollection at 0x1721a60a108>
```

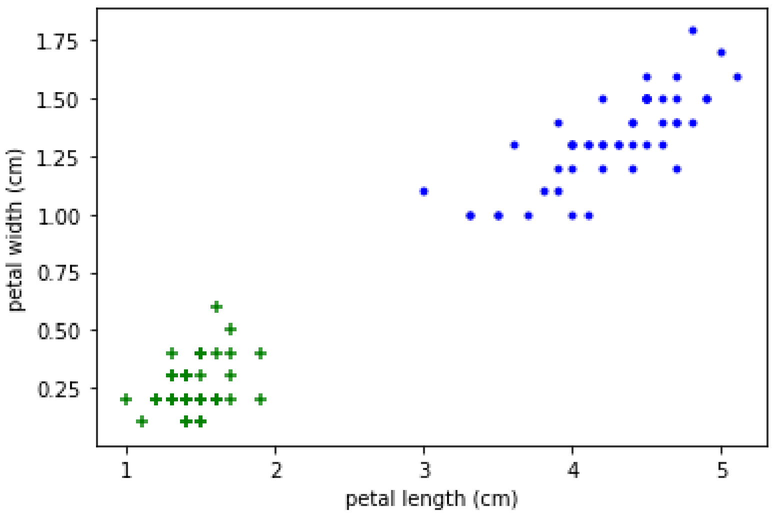


In [30]:

```
plt.xlabel('petal length (cm)')
plt.ylabel('petal width (cm)')
plt.scatter(df0['petal length (cm)'],df0['petal width (cm)'],color='green',marker='+')
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='blue',marker='.')
```

Out[30]:

```
<matplotlib.collections.PathCollection at 0x1721bf14a48>
```



In [31]:

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

In [34]:

```
X = df.drop(['target','flower_name'],axis='columns')
X.head()
```

Out[34]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [35]:

```
y = df.target
```

In [49]:

```
X_train,X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [38]:

```
len(X_train)
```

Out[38]:

```
120
```

In [39]:

```
len(X_test)
```

Out[39]:

```
30
```

In [56]:

```
from sklearn.svm import SVC
model = SVC(kernel='linear')
```

In [57]:

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

Out[57]:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=
0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='lin
ear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [58]:

```
model.score(X_test,y_test)
```

Out[58]:

```
1.0
```

Exercise

Train SVM classifier using sklearn digits dataset (i.e. from sklearn.datasets import load_digits) and then,

1. Measure accuracy of your model using different kernels such as rbf and linear.
2. Tune your model further using regularization and gamma parameters and try to come up with highest accurancy score
3. Use 80% of samples as training data size