ML LAB 9

Implement Support Vector Machine algorithm for classification in a given business environment and comment on its efficiency and performance.

Support Vector Machine Tutorial Using Python Sklearn

In [1]:

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

In [2]:

```
iris.feature_names
```

Out[2]:

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

In [3]:

```
iris.target_names
```

Out[3]:

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

In [6]:
```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[6]:

	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 [8]:
```
df['target'] = iris.target
df.head()
```

Out[8]:

	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 [9]:
```
df[df.target==1].head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

In [10]:
```
df[df.target==2].head()
```

Out[10]:

	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 [11]:

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

Out[11]:

	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
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 [13]:

```
df[45:55]
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

In [15]:

```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

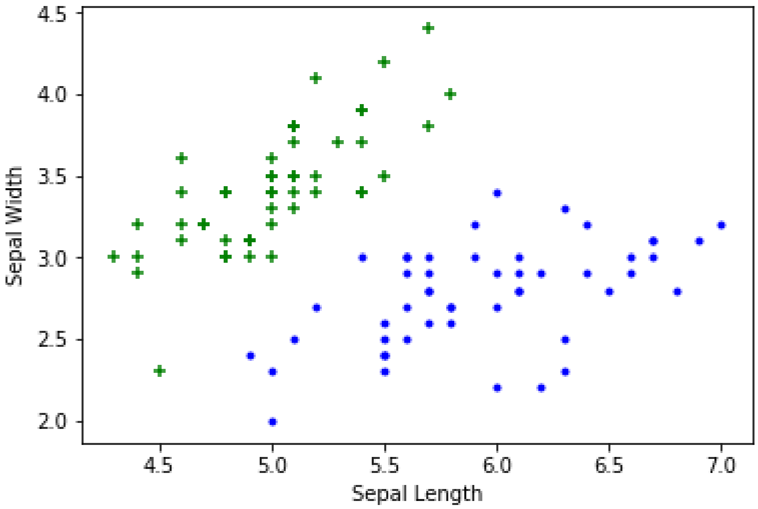
In [14]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Sepal length vs Sepal Width (Setosa vs Versicolor)

In [17]:
```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marke
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1f1b16976a0>
```



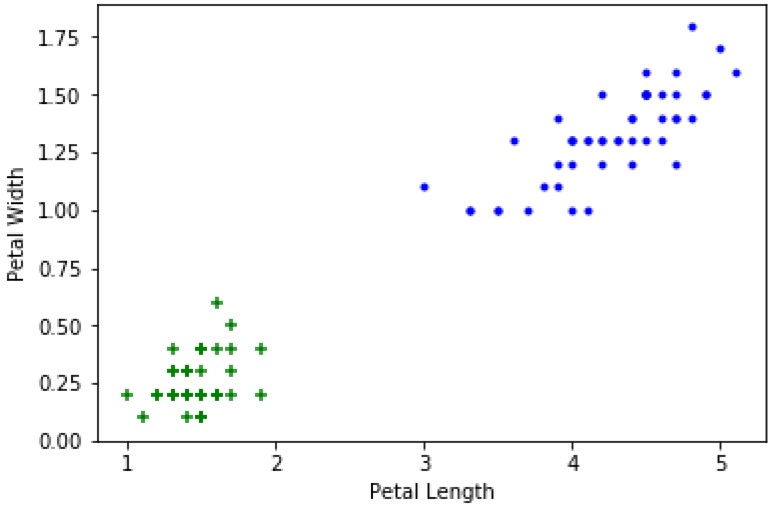
Petal length vs Pepal Width (Setosa vs Versicolor)

In [18]:

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marke
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x1f1b2018390>
```



Train Using Support Vector Machine (SVM)

In [49]:

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

In [50]:

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

In [51]:

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

In [52]:

```
len(X_train)
```

Out[52]:

```
120
```

In [53]:

```
len(X_test)
```

Out[53]:

```
30
```

In [75]:

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

In [76]:

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

Out[76]:

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

In [77]:

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

Out[77]:

```
0.9333333333333335
```

In [78]:

```
model.predict([[4.8,3.0,1.5,0.3]])
```

Out[78]:

```
array([0])
```

Tune parameters

1. Regularization (C)

In [97]:

```
model_C = SVC(C=1)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

Out[97]:

```
0.9333333333333335
```

In [106]:

```
model_C = SVC(C=10)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

Out[106]:

```
0.9666666666666667
```

2. Gamma

In [103]:

```
model_g = SVC(gamma=10)
model_g.fit(X_train, y_train)
model_g.score(X_test, y_test)
```

Out[103]:

```
0.9000000000000002
```

3. Kernel

In [104]:

```
model_linear_kernal = SVC(kernel='linear')
model_linear_kernal.fit(X_train, y_train)
```

Out[104]:

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

In [105]:

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

Out[105]:

```
0.9666666666666667
```

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