Semi-Supervised Learning with Support Vector Machine (SVM)

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import datasets
import matplotlib.pyplot as plt
# pd.set_option('display.max_rows', None)
```

Preparing the dataset

In [2]:
```
#Load dataset
dataset = datasets.load_wine()
```

In [15]:
```
df = pd.DataFrame(dataset.data, columns=[dataset.feature_names])
df['label'] = pd.Series(dataset.target)
df = df.sample(frac=1).reset_index(drop=True) # shuffle the dataframe in-
place and reset the index
df
```

Out[15]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_
0	13.87	1.90	2.80	19.4	107.0	2.95	2.97	
1	13.34	0.94	2.36	17.0	110.0	2.53	1.30	
2	14.38	3.59	2.28	16.0	102.0	3.25	3.17	
3	13.07	1.50	2.10	15.5	98.0	2.40	2.64	
4	12.33	1.10	2.28	16.0	101.0	2.05	1.09	
...	...	...	...	...	...	...	...	
173	12.58	1.29	2.10	20.0	103.0	1.48	0.58	
174	12.82	3.37	2.30	19.5	88.0	1.48	0.66	
175	12.43	1.53	2.29	21.5	86.0	2.74	3.15	
176	12.07	2.16	2.17	21.0	85.0	2.60	2.65	
177	12.70	3.55	2.36	21.5	106.0	1.70	1.20	

178 rows × 14 columns

Create labeled dataset

In [16]:
```
#taking half of the dataset as labled data
X = df.iloc[0:89,0:13].values
y = df.iloc[0:89,-1].values
df.iloc[0:89,0:13]
```

Out[16]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_p
0	13.87	1.90	2.80	19.4	107.0	2.95	2.97	
1	13.34	0.94	2.36	17.0	110.0	2.53	1.30	
2	14.38	3.59	2.28	16.0	102.0	3.25	3.17	
3	13.07	1.50	2.10	15.5	98.0	2.40	2.64	
4	12.33	1.10	2.28	16.0	101.0	2.05	1.09	
...	...	...	...	...	...	...	...	
84	12.33	0.99	1.95	14.8	136.0	1.90	1.85	
85	12.42	2.55	2.27	22.0	90.0	1.68	1.84	
86	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
87	13.29	1.97	2.68	16.8	102.0	3.00	3.23	
88	13.20	1.78	2.14	11.2	100.0	2.65	2.76	

89 rows × 13 columns

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7,
random_state=1)
```

In [18]:
```
X_train.shape
```

Out[18]: (26, 13)

In [19]:
```
X_test.shape
```

Out[19]: (63, 13)

Create unlabeled dataset

In [20]:
```
# taking the other half of the data as unlabeled data
X_unl_df = df.iloc[89:,0:13].reset_index(drop=True)
```

```
X_unl = X_unl_df.values
X_unl_df
```

Out[20]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_p
0	11.64	2.06	2.46	21.6	84.0	1.95	1.69	
1	14.22	3.99	2.51	13.2	128.0	3.00	3.04	
2	11.82	1.72	1.88	19.5	86.0	2.50	1.64	
3	14.19	1.59	2.48	16.5	108.0	3.30	3.93	
4	12.37	1.13	2.16	19.0	87.0	3.50	3.10	
...	...	...	...	...	...	...	...	
84	12.58	1.29	2.10	20.0	103.0	1.48	0.58	
85	12.82	3.37	2.30	19.5	88.0	1.48	0.66	
86	12.43	1.53	2.29	21.5	86.0	2.74	3.15	
87	12.07	2.16	2.17	21.0	85.0	2.60	2.65	
88	12.70	3.55	2.36	21.5	106.0	1.70	1.20	

89 rows × 13 columns

1. Training on the labeled dataset

In [21]:

```
clf = svm.SVC(kernel='linear', probability=True, C=1.0).fit(X_train,
y_train)
clf.score(X_test, y_test)
```

Out[21]: 0.7936507936507936

2. Make a prediction using the unlabeled datset (x_unl)

In [22]:

```
#find the probability of each class
clp= clf.predict_proba(X_unl)
clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2','class3'])
# predict the the label of each class
lab=clf.predict(X_unl)
clf_prob["max"] = clf_prob.max(axis = 1)
clf_prob["lab"] = lab
clf_prob
```

Out[22]:

	class1	class2	class3	max	lab

	class1	class2	class3	max	lab
0	0.154002	0.314076	0.531922	0.531922	1
1	0.292137	0.270598	0.437265	0.437265	2
2	0.023514	0.379967	0.596519	0.596519	1
3	0.998248	0.001364	0.000388	0.998248	0
4	0.024358	0.378537	0.597105	0.597105	1
...	...	...	...	...	...
84	0.119894	0.324574	0.555532	0.555532	2
85	0.160462	0.305302	0.534236	0.534236	2
86	0.015080	0.382361	0.602559	0.602559	1
87	0.018101	0.381986	0.599914	0.599914	1
88	0.090178	0.339134	0.570689	0.570689	2

89 rows × 5 columns

3. Choose the samples in X_unl with high confidence and add them into the labeled dataset

In [23]:

```
th = 0.6
clf_prob[clf_prob["max"] > th]
```

Out[23]:

	class1	class2	class3	max	lab
3	0.998248	0.001364	0.000388	0.998248	0
5	0.014084	0.383017	0.602899	0.602899	1
6	0.970301	0.019355	0.010344	0.970301	0
8	0.994399	0.004148	0.001453	0.994399	0
9	0.937449	0.038556	0.023996	0.937449	0
11	0.994186	0.004298	0.001517	0.994186	0
12	0.991134	0.006408	0.002458	0.991134	0
16	0.009069	0.387745	0.603186	0.603186	1
17	0.819020	0.094391	0.086589	0.819020	0
19	0.611770	0.171319	0.216911	0.611770	0
20	0.842308	0.085591	0.072101	0.842308	0
21	0.715648	0.137932	0.146420	0.715648	0
22	0.034904	0.364447	0.600649	0.600649	2
27	0.968139	0.020916	0.010945	0.968139	0
29	0.018317	0.379884	0.601800	0.601800	1
31	0.913269	0.050632	0.036099	0.913269	0

	class1	class2	class3	max	lab
36	0.017267	0.382011	0.600722	0.600722	1
43	0.012550	0.384893	0.602557	0.602557	1
48	0.962569	0.024327	0.013104	0.962569	0
55	0.984796	0.010579	0.004625	0.984796	0
56	0.748896	0.126724	0.124380	0.748896	0
60	0.880995	0.066579	0.052426	0.880995	0
63	0.009867	0.386900	0.603233	0.603233	1
64	0.976518	0.015492	0.007990	0.976518	0
66	0.903556	0.056237	0.040207	0.903556	0
71	0.037796	0.359127	0.603077	0.603077	2
79	0.937871	0.037880	0.024249	0.937871	0
82	0.742474	0.126368	0.131158	0.742474	0
86	0.015080	0.382361	0.602559	0.602559	1

In [24]:

```
#add the predicted labels to the training dataset
unl_size =len(X_unl[clf_prob["max"] > th])
X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
y_train_new = np.append(y_train, clf_prob['lab'][clf_prob["max"] >
th].values, axis=0)

X_train = X_train_new
y_train = y_train_new
```

In [25]:

```
#remove the added labels from the unlabled dataset
X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
#update the unlabeled set
X_unl = X_unl_df.values
# X_unl_df
```

4. Repeat

In [26]:

```
score_ls = []
while len(X_unl) != 0 and unl_size != 0: # stop when there are no more
unlabeled data or when we are no confident about the data
```

```
    #Step 1
    clf = svm.SVC(kernel='linear', probability=True,C=1).fit(X_train,
y_train)
    score_ls.append(clf.score(X_test, y_test))
    print ('Accuracy: ',clf.score(X_test, y_test))
#     print(len(X_unl))

    #Step2
    #find the probability of each class
    clp= clf.predict_proba(X_unl)
    clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2','class3'])
    # predict the the label of each class
    lab=clf.predict(X_unl)
    clf_prob["max"] = clf_prob.max(axis = 1)
    clf_prob["lab"] = lab

    #Step3
    unl_size =len(X_unl[clf_prob["max"] > th])
    X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
    y_train_new = np.append(y_train, clf_prob['lab'][clf_prob["max"] >
th].values, axis=0)
    X_train = X_train_new
    y_train = y_train_new



    X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
    X_unl = X_unl_df.values
```

```
Accuracy:  0.7936507936507936
Accuracy:  0.8095238095238095
Accuracy:  0.873015873015873
Accuracy:  0.873015873015873
```

In [ ]:

```

```

In [ ]:

```

```