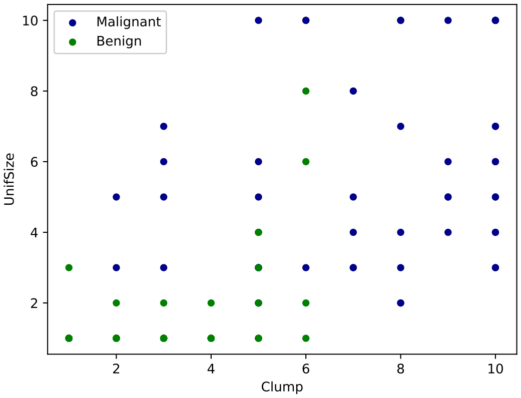
Support Vector Machine Example

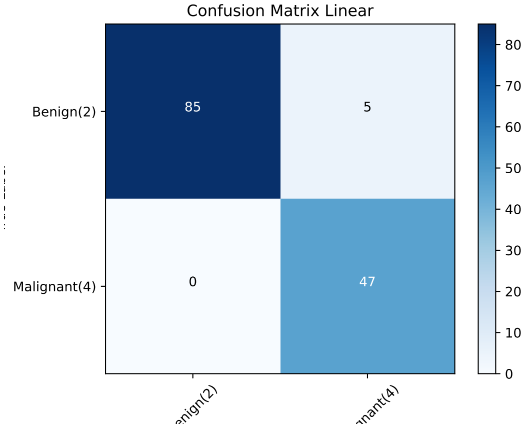
```
import numpy as np
import pandas as pd
import scipy.optimize as opt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

from matplotlib import rc,font_manager

ticks_font = font_manager.FontProperties(family='Times New Roman', style='normal',
    size=12, weight='normal', stretch='normal')

## Loading Data ##
df=pd.read_csv('D:\Python\edx\Machine Learning\Classification\cell_samples.csv')
with open('SVM.txt','a') as f:
    print(df.head(),file=f)

# Distribution of classes #
p1=df[df['Class']==4][0:50].plot(kind='scatter',x='Clump',y='UnifSize',color='DarkBlue',l
p2=df[df['Class']==2][0:50].plot(kind='scatter',x='Clump',y='UnifSize',color='Green',labe

with open('SVM.txt','a') as f:
    print(df.dtypes,file=f)

# Converting non-numeric columns to numeric #

df=df[pd.to_numeric(df['BareNuc'],errors='coerce').notnull()]
df['BareNuc']=df['BareNuc'].astype('int')

with open('SVM.txt','a') as f:
    print(df.dtypes,file=f)

# Feature variables (X) and Target field (y)

feature_df = df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc',
X = np.asarray(feature_df)
df['Class'] = df['Class'].astype('int')
y = np.asarray(df['Class'])

with open('SVM.txt','a') as f:
    print(X[0:5],file=f)
    print(y[0:5],file=f)

# Train Test Split dataset

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=4)
with open('SVM.txt','a') as f:
    print('Train set: ', X_train.shape,y_train.shape,file=f)
    print('Test set: ', X_test.shape,y_test.shape,file=f)

#Modeling SVM-Linear, RBF, Polynomial,Sigmoid

from sklearn import svm
clf_linear=svm.SVC(kernel='linear')
clf_linear.fit(X_train,y_train)
yhat_linear=clf_linear.predict(X_test)
clf_rbf=svm.SVC(kernel='rbf')
clf_rbf.fit(X_train,y_train)
yhat_rbf=clf_rbf.predict(X_test)
with open('SVM.txt','a') as f:
    print('Yhat Linear \n', yhat_linear,file=f)
    print('Yhat RBF \n', yhat_rbf,file=f)

#Evaluation usign confusion matrix

from sklearn.metrics import classification_report,confusion_matrix
import itertools

def plot_cmat(cm,classes,normalize=False,
                title='Confusion Matrix',cmap=plt.cm.Blues):
                if normalize:
                    cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
                    print('Normalized Confusion Matrix')
                else:
                    print('Confusion matrix without Normalization')

                with open('SVM.txt','a') as f:
                    print(cm,file=f)

                plt.imshow(cm,interpolation='nearest',cmap=cmap)
                plt.title(title)
                plt.colorbar()
                tick_marks=np.arange(len(classes))
                plt.xticks(tick_marks,classes,rotation=45)
                plt.yticks(tick_marks,classes)

                #For Labeling inside boxes #
                fmt='.2f' if normalize else 'd'
                threshold=cm.max()/2
                for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
                    plt.text(j,i,format(cm[i,j],fmt),
                            horizontalalignment='center',
                            color='white' if cm[i,j] > threshold else 'black')
                plt.tight_layout
                plt.ylabel('True Label')
                plt.xlabel('Predicted Label')

#Confusion matrix #
c_mat_linear=confusion_matrix(y_test,yhat_linear,labels=[2,4])
c_mat_rbf=confusion_matrix(y_test,yhat_rbf,labels=[2,4])
with open('SVM.txt','a') as f:
    print('Confusion matrix (Linear): \n ',c_mat_linear,file=f)
    print('Confusion matrix (RBF): \n ',c_mat_rbf,file=f)

#Confusion matrix plot#
plt.figure()
plot_cmat(c_mat_linear,classes=['Benign(2)','Malignant(4)'],normalize=False,title='Confu
plt.figure()
plot_cmat(c_mat_rbf,classes=['Benign(2)','Malignant(4)'],normalize=False,title='Confusio

# Compute classification report - Precision, Recall, F1Score and Support

with open('SVM.txt','a') as f:
    print('Classification Report (Linear): \n ',classification_report(y_test,yhat_linear
    print('Classification Report (RBF): \n ',classification_report(y_test,yhat_rbf),file

#Accuracy scores
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import jaccard_score
with open('SVM.txt','a') as f:
    print('F1 Score (Linear): ',f1_score(y_test, yhat_linear, average='weighted'),file=f
    print('F1 Score (RBF): ',f1_score(y_test, yhat_rbf, average='weighted'),file=f)
    print('Accuracy Score (Linear): ',accuracy_score(y_test, yhat_linear),file=f)
    print('Accuracy Score (RBF):',accuracy_score(y_test, yhat_rbf),file=f)
#Display plot
plt.show()
```

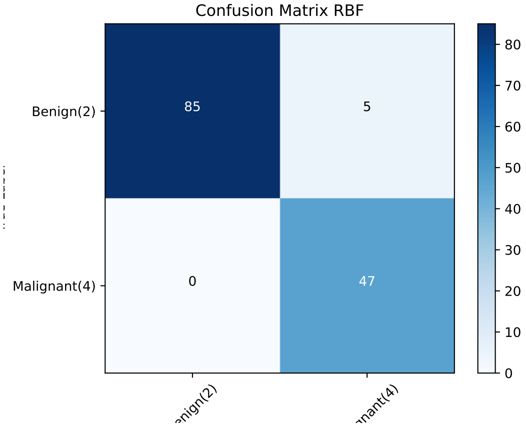
Solution:

```
        ID  Clump  UnifSize  UnifShape  MargAdh  SingEpiSize BareNuc  BlandChrom
NormNucl  Mit  Class
```

```
0  1000025      5         1          1        1            2       1           3
1    1      2
1  1002945      5         4          4        5            7      10           3
2    1      2
2  1015425      3         1          1        1            2       2           3
1    1      2
3  1016277      6         8          8        1            3       4           3
7    1      2
4  1017023      4         1          1        3            2       1           3
1    1      2
ID                int64
Clump             int64
UnifSize          int64
UnifShape         int64
MargAdh           int64
SingEpiSize       int64
BareNuc          object
BlandChrom        int64
NormNucl          int64
Mit               int64
Class             int64
dtype: object
ID               int64
Clump            int64
UnifSize         int64
UnifShape        int64
MargAdh          int64
SingEpiSize      int64
BareNuc          int32
BlandChrom       int64
NormNucl         int64
Mit              int64
Class            int64
dtype: object
[[ 5  1  1  1  2  1  3  1  1]
 [ 5  4  4  5  7 10  3  2  1]
 [ 3  1  1  1  2  2  3  1  1]
 [ 6  8  8  1  3  4  3  7  1]
 [ 4  1  1  3  2  1  3  1  1]]
[2 2 2 2 2]
Train set:  (546, 9) (546,)
Test set:  (137, 9) (137,)
Yhat Linear
 [2 4 2 4 2 2 2 2 4 2 2 4 4 4 4 2 2 2 2 2 4 2 4 4 4 4 2 2 4 4 4 2 4 2 2 2 4
 2 2 2 2 2 2 4 4 2 2 2 2 4 2 2 2 2 2 2 4 2 2 2 2 4 4 2 4 4 4 2 2 2 4 4 2 2
 2 4 2 2 4 4 2 2 2 2 4 4 2 4 2 2 4 4 2 2 2 4 2 2 2 4 2 4 2 2 4 2 4 2 2 4 2
 2 4 2 2 2 2 2 4 4 4 4 4 2 2 4 2 2 4 2 4 2 2 2 2 2 4]
Yhat RBF
 [2 4 2 4 2 2 2 2 4 2 2 4 4 4 4 2 2 2 2 2 4 2 4 4 4 4 2 2 4 4 4 2 4 2 2 2 4
 2 2 2 2 2 2 4 4 2 2 2 2 4 2 2 2 2 2 2 4 2 2 2 2 4 4 2 4 4 4 2 2 2 4 4 2 2
 2 4 2 2 4 4 2 2 2 2 4 4 2 4 2 2 4 4 2 2 2 4 2 2 2 4 2 4 2 2 4 2 4 2 2 4 2
 2 4 2 2 2 2 2 4 4 4 4 4 2 2 4 2 2 4 2 4 2 2 2 2 2 4]
Confusion matrix (Linear):
  [[85  5]
 [ 0 47]]
Confusion matrix (RBF):
  [[85  5]
 [ 0 47]]
[[85  5]
 [ 0 47]]
[[85  5]
 [ 0 47]]
Classification Report (Linear):
               precision    recall  f1-score   support

           2       1.00      0.94      0.97        90
           4       0.90      1.00      0.95        47

    accuracy                           0.96       137
   macro avg       0.95      0.97      0.96       137
weighted avg       0.97      0.96      0.96       137

Classification Report (RBF):
               precision    recall  f1-score   support

           2       1.00      0.94      0.97        90
           4       0.90      1.00      0.95        47

    accuracy                           0.96       137
   macro avg       0.95      0.97      0.96       137
weighted avg       0.97      0.96      0.96       137

F1 Score (Linear):  0.9639038982104676
F1 Score (RBF):  0.9639038982104676
Accuracy Score (Linear):  0.9635036496350365
Accuracy Score (RBF): 0.9635036496350365
```


Malignant
Benign
UnifSize
Clump
2
4
6
8
10
2
4
6
8
10


Confusion Matrix Linear
Benign(2)
Malignant(4)
85
5
0
47
enign(2)
gnant(4)
0
10
20
30
40
50
60
70
80


Confusion Matrix RBF
Benign(2)
Malignant(4)
85
5
0
47
enign(2)
nant(4)
0
10
20
30
40
50
60
70
80