SVM

Using SVM Build and Train a model using human cell records , Classifier helps to predict new one is benign or malignant.SVM works by mapping data to high dimensional feature space. A seperator can be drawn as hyperplane .

Working on dataset from Kaggle and Using SVM Predict a human cell is benign or malignant

In [28]:
```
import numpy as np
import os,  seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [29]:
```
data = pd.read_csv("data/human_cells.csv")
```

In [30]:
```
print(data.shape)
```

```
(699, 11)
```

In [31]:
```
#Print no of integers, floats and strings
data.dtypes.value_counts()
```

Out[31]:
```
int64     10
object     1
dtype: int64
```

In [32]:
```
data.head()
```

Out[32]:

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl
0	1000025	5	1	1	1	2	1	3	1
1	1002945	5	4	4	5	7	10	3	2
2	1015425	3	1	1	1	2	2	3	1
3	1016277	6	8	8	1	3	4	3	7
4	1017023	4	1	1	3	2	1	3	1

In [33]:
```
#checking which one has object
data.dtypes
```

Out[33]:
```
ID              int64
Clump           int64
UnifSize        int64
UnifShape       int64
MargAdh         int64
SingEpiSize     int64
BareNuc        object
BlandChrom      int64
NormNucl        int64
Mit             int64
Class           int64
dtype: object
```

Preprocessing Steps

1. Select Features .
2. Split the data into train and test sets.

1. Select Features .

In [34]:
```
#Dropping BareNuc has some columns which is not numerical .
data = data[pd.to_numeric(data['BareNuc'], errors='coerce').notnull()]
data['BareNuc'] = data['BareNuc'].astype('int')
data.dtypes
```

Out[34]:
```
ID              int64
Clump           int64
UnifSize        int64
UnifShape       int64
MargAdh         int64
SingEpiSize     int64
BareNuc         int64
BlandChrom      int64
NormNucl        int64
Mit             int64
Class           int64
dtype: object
```

In [38]:
```
ng ID and extracting remaining features as X and y
asarray(data[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'E

lass'] = data['Class'].astype('int')
asarray(data['Class'])
```

2. Split Data to Train and Test sets

In [39]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_st
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (512, 9) (512,)
Test set: (171, 9) (171,)
```

Modeling with SVM

Mapping to Higher dimension is called kernelling and svm uses linear, polynomial,rbf as kernel mathematical functions .

In [41]:
```
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

Out[41]:
```
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='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [44]:
```
y_hat = clf.predict(X_test)
```

Confusion Matrix and Plotting

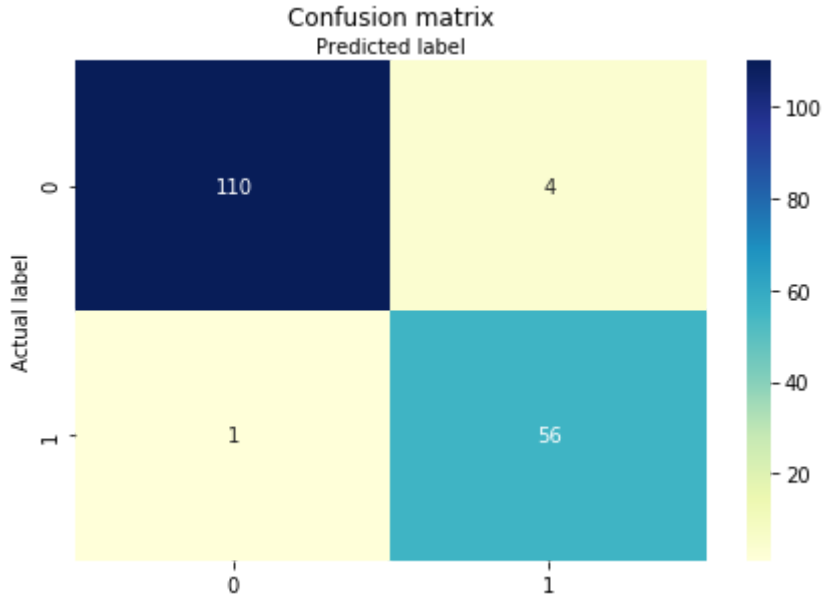
In [52]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_hat)
print(cnf_matrix)
```

```
[[110   4]
 [  1  56]]
```

Confusion Matrix shows models ability to correctly predict or seperate

In [55]:
```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
classes=[2,4] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[55]:
```
Text(0.5, 257.44, 'Predicted label')
```



Precison /Recall and F1- score

In [56]:
```
from sklearn.metrics import classification_report
print (classification_report(y_test, y_hat))
```

```
              precision    recall  f1-score   support

           2       0.99      0.96      0.98       114
           4       0.93      0.98      0.96        57

    accuracy                           0.97       171
   macro avg       0.96      0.97      0.97       171
weighted avg       0.97      0.97      0.97       171
```

Summary

SVM Classifier could predict with 97 % accuracy.

In [ ]: