Breast Cancer Predictions using SVM

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import preprocessing
import pandas as pd
import random
import itertools
import seaborn as sns

sns.set(style = 'darkgrid')
% matplotlib inline
```

In [2]:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    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)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [3]:

```
bc = pd.read_csv('../input/data.csv')
bc.head(1)
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.8	1001.0	(

1 rows × 33 columns

Scale the data to chart it and allow better predictive power

In [4]:

```
bcs = pd.DataFrame(preprocessing.scale(bc.ix[:,2:32]))
bcs.columns = list(bc.ix[:,2:32].columns)
bcs['diagnosis'] = bc['diagnosis']
```

Let's build a SVM to predict malignant or benign tumors

In [9]:

```
X = bcs.ix[:,0:30]

y = bcs['diagnosis']
class_names = list(y.unique())
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=
42)
```

Model scores very well. Very slight over-fitting.

In [11]:

```
svc = SVC(kernel = 'linear',C=.1, gamma=10, probability = True)
svc.fit(X,y)
y_pred = svc.fit(X_train, y_train).predict(X_test)
t = pd.DataFrame(svc.predict_proba(X_test))
svc.score(X_train,y_train), svc.score(X_test, y_test)
```

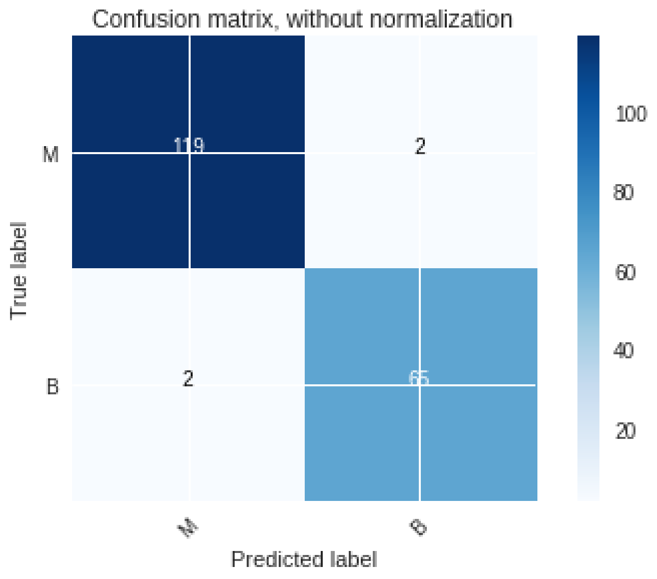
Out[11]:

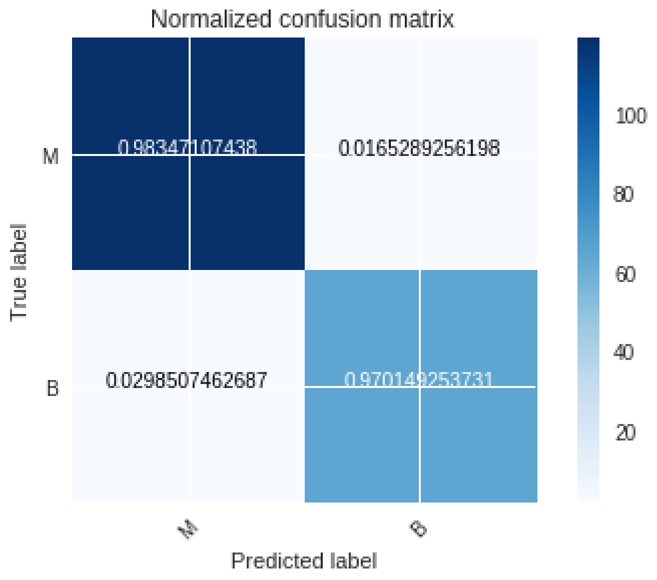
```
(0.98425196850393704, 0.97872340425531912)
```

In [12]:

```
mtrx = confusion_matrix(y_test,y_pred)
np.set_printoptions(precision = 2)

plt.figure()
plot_confusion_matrix(mtrx,classes=class_names,title='Confusion matrix, without normali
zation')

plt.figure()
plot_confusion_matrix(mtrx, classes=class_names, normalize = True, title='Normalized co
nfusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[119   2]
 [  2  65]]
Normalized confusion matrix
[[ 0.98  0.02]
 [ 0.03  0.97]]
```





In [13]: