節14 – SVM (線性可分)

執行項目：

- 資料預處理 : 分為訓練與測試data
- 建立svm模型
- 混淆矩陣 – 預測結果
- 畫出decision boundry

資料預處理 - 載入套件

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
```

修改檔案開啟目錄

```
os.chdir("C:/Users/ronald/Desktop/(程式資料)Machine-Learning-A-Z-Template-Folder/Machine Learning A-Z Template Folder/Part 3 – Classification/Section 16 - Support Vector Machine (SVM)")
```

讀入資料

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
print(dataset.head())
```

```
In [8]: print(dataset.head())
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
```

將資料切成訓練與測試DATA (25%為測試、75%訓練)

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
In [33]: X_test.shape
Out[33]: (100, 2)

In [34]: X_train.shape
Out[34]: (300, 2)

In [35]: X.shape
Out[35]: (400, 2)
```

```
In [36]: y_test.shape
Out[36]: (100,)

In [37]: y_train.shape
Out[37]: (300,)

In [38]: y.shape
Out[38]: (400,)
```

- test_size = 0.25 ，表示訓練樣本為25%
- random_state = 0，設定取樣數值，若設定同數值有同樣結果
- shape 可顯示資料維度，可看出切割大小

將資料標準化

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
In [41]: print(X_test)
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 ...,
 [ 0.97777845 -1.06066585]
 [ 0.97777845  0.59194336]
 [ 0.38358493  0.99784738]]
```

```
In [42]: print(X_train)
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 ...,
 [-0.21060859 -0.50979612]
 [-1.10189888 -0.45180983]
 [-1.20093113  1.40375139]]
```

- X_test那邊，在train部分，物件sc的fit已run過所以後面的test就不用再加上

建立SVM模型

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

```
Out[43]:
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=0, shrinking=True,
  tol=0.001, verbose=False)
```

- 參數kernel : 就是svm-kernel函數的類型，這邊做線性可分，故指定linear。
- 參數gamma : 多用於rbf函數，是對分類範圍設定。
- 參數C : 稱為懲罰函數，為誤差控制係數，其值越大錯誤分類減少(分類邊緣減小，但會有過度配適問題)，期值越小，錯誤分類多，邊緣較大。
  (建議視資料雜訊而定，資料雜訊小適合較大的C值)

混淆矩陣 - 計算估計值Y的預測值及混淆矩陣、估計準確率

```
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

<table>
<tr><td colspan="2" rowspan="2">cm</td><td colspan="2">預測值 y_pred</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td rowspan="2">實際值 y_test</td><td>0</td><td>66</td><td>2</td></tr>
<tr><td>1</td><td>8</td><td>24</td></tr>
</table>

```
In [46]: from sklearn.metrics import accuracy_score
    ...: accuracy_score(y_test, y_pred)
    ...: print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
Accuracy: 0.90
```

- 補充：使用套件accuracy_score可直接算出混淆矩陣cm的準確率

畫出DECISION BOUNDRY (1.畫出分類區塊、2.資料點)

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
```

資料範圍設定好，使用套件與指令如下：

- 使用ListedColormap套件將資料範圍上色。
- 使用np.arange建立向量 : **start** : 起始值、**stop** : 終端值、**step** : 間隔
- 使用np.mesgrid產生網格 (以X1、X2為範圍) : **np.mesgrid**會將向量回傳為陣列形態。

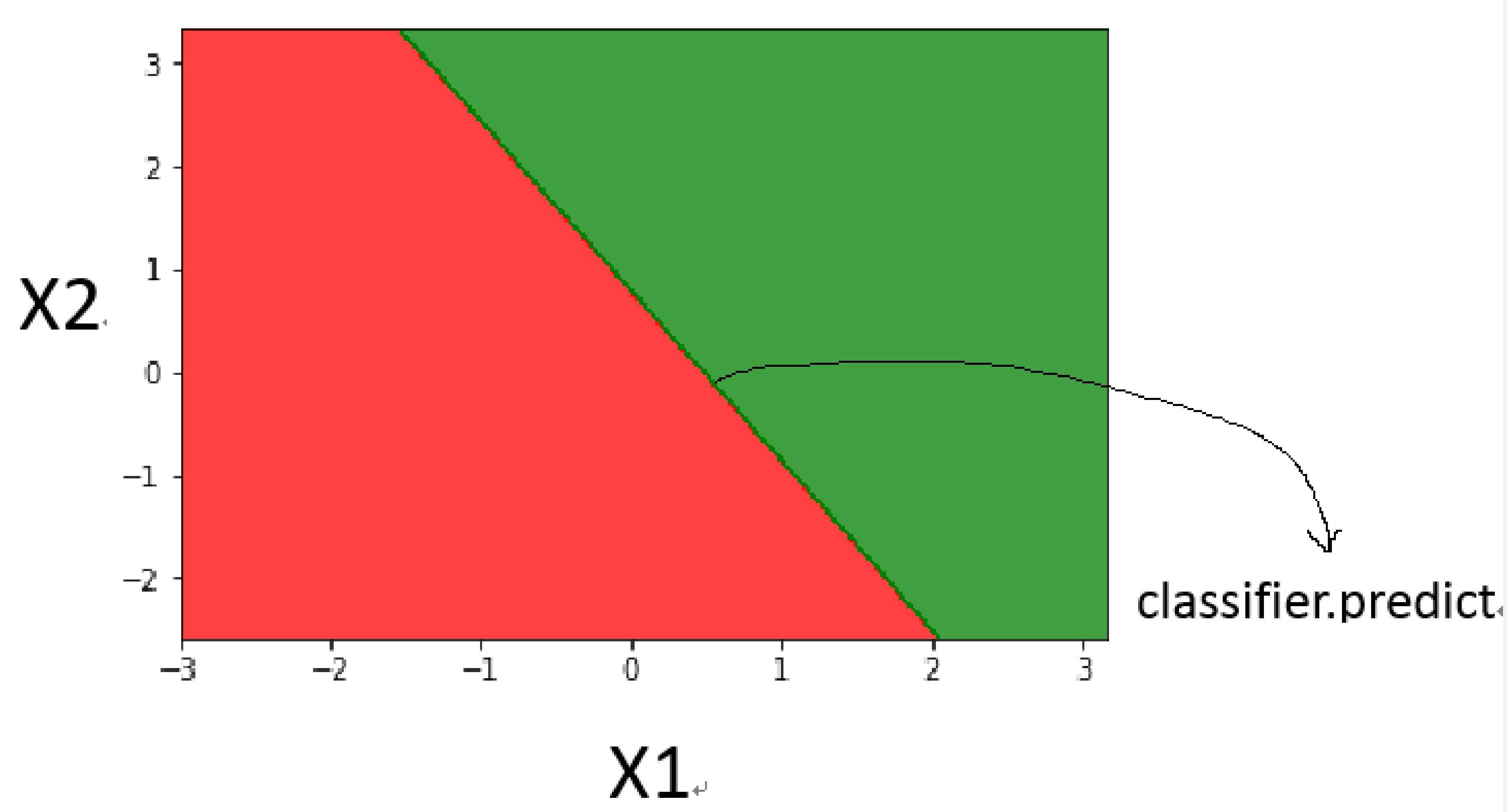
1.畫出分類區塊

```
classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape)
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

分類器形態設定，使用套件與指令如下：

- 使用.ravel()將資料點攤平
- 使用np.array建立陣列形態，再用.T轉置放入classifier.predict分類器中
- 使用reshape將預測值轉為X1的維度，即可放入等高線plt.contourf中。(用classifier.predict預測的值是向量，要轉陣列才可畫圖)
- plt.contourf維等高線圖指令，參數alpha為顏色透明度，越高顏色越深
- ListedColormap為顏色設定，為由左而右為red、green。(具順序性)
- plt.xlim、 plt.ylim為圖中x和y軸的範圍，分別為X1 和 X2

SVM預測範圍



2.畫出資料點散佈圖

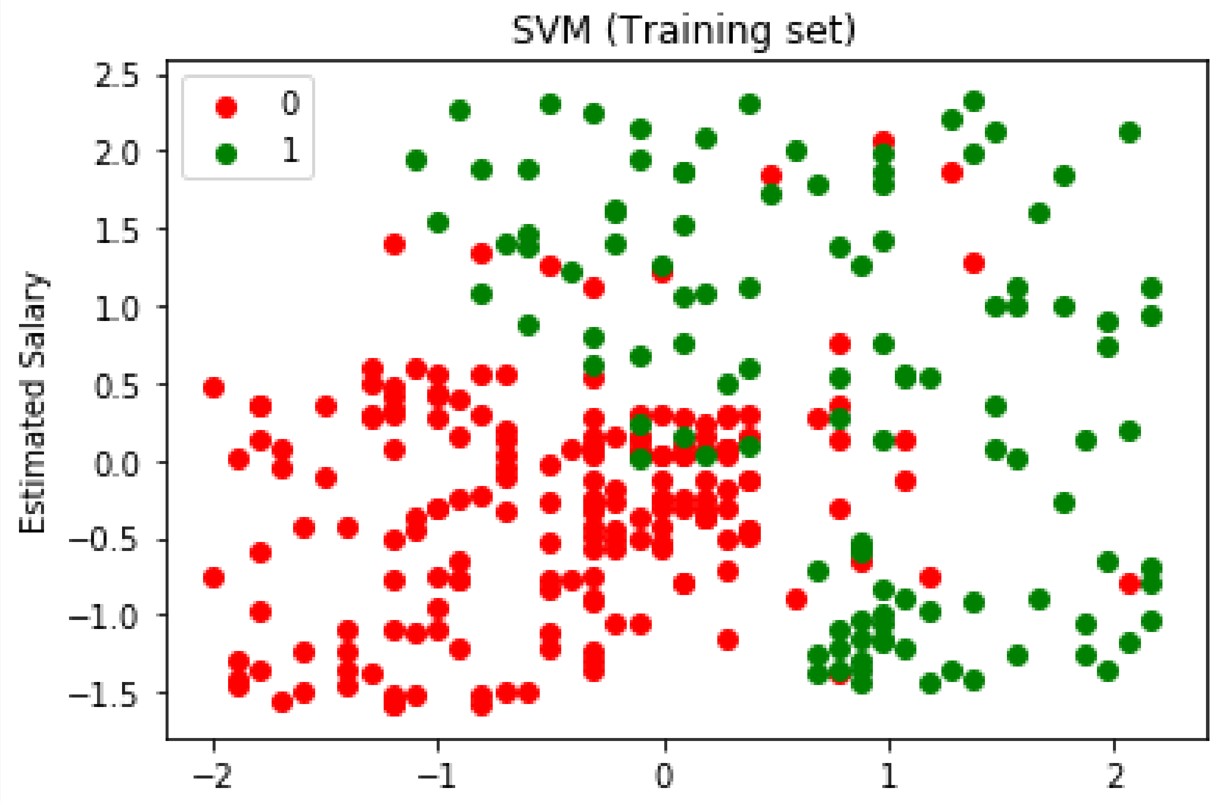
```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap((‘red’, ‘green'))(i), label = j)
```

使用enumerate迴圈，將資料點畫出:

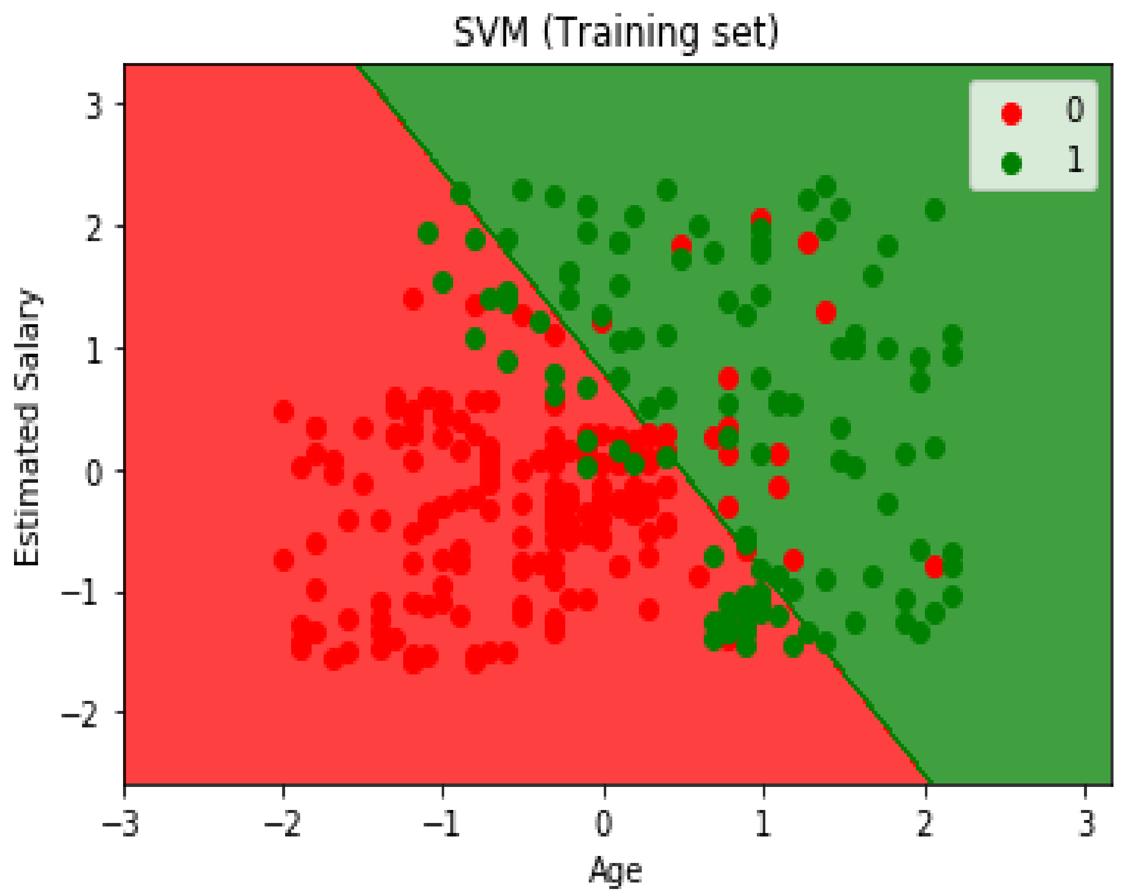
- np.unique會回傳資料的唯一值 (y_set 只有0、1)。
- enumerate迴圈 : 會回傳資料的位置及數值，所以(i,j)為(0,0)、(1,1)
- 布林邏輯式 : X_set[y_set == j, 0]，y_set==j為是否滿足y_set等於j的條件式，
  這邊是指滿足y_set為j值的所有X_set的第0列的點，同理第1列一樣。
- label : 為圖例、plt.scatter是畫出散佈圖。
- 執行方式 : 迴圈一共跑了2次，分別是i,j=0及i,j=1這2次

資料點散佈圖

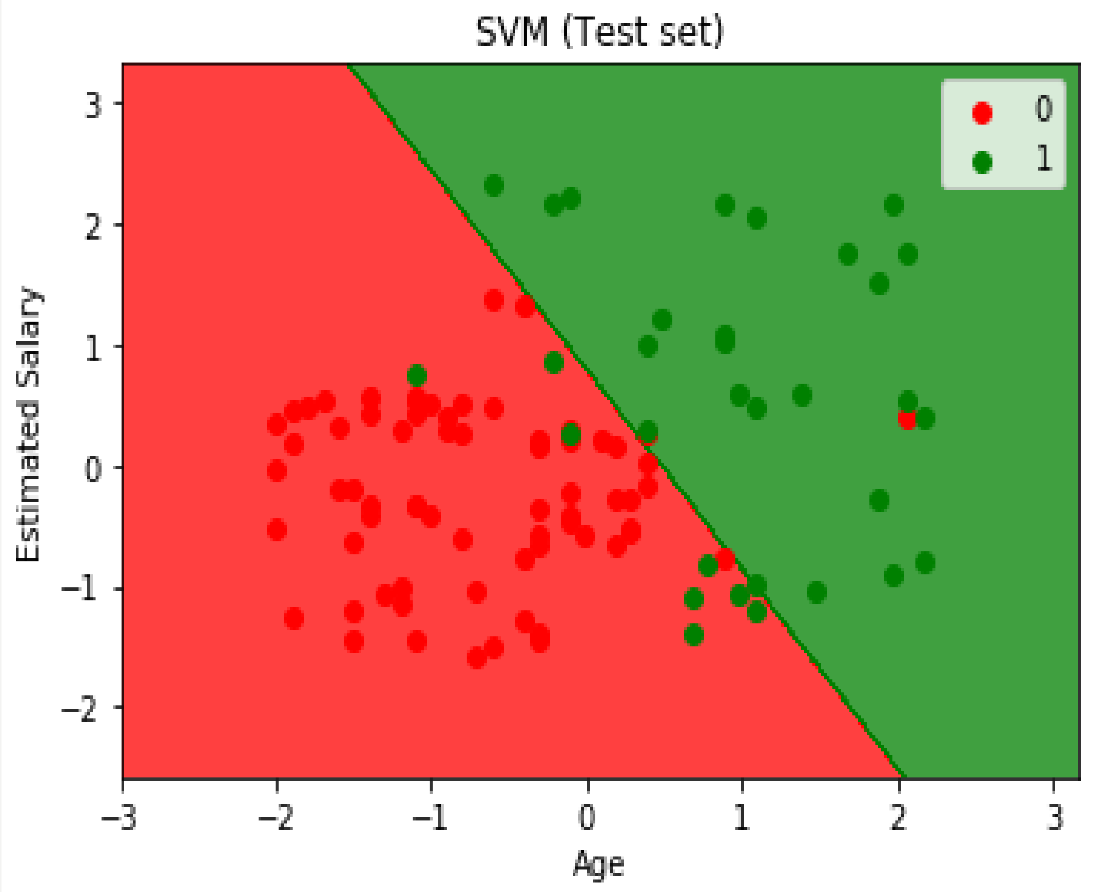
```
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



訓練及測試的預測結果



SVM (Training set)
Estimated Salary
Age
0
1



SVM (Test set)
Estimated Salary
Age
0
1

節15 – SVM KERNEL (非線性可分)

執行項目：

- 資料預處理 : 分為訓練與測試data
- 建立svm模型
- 混淆矩陣 – 預測結果
- 畫出decision boundry

KERNEL函數變為RBF高斯函數

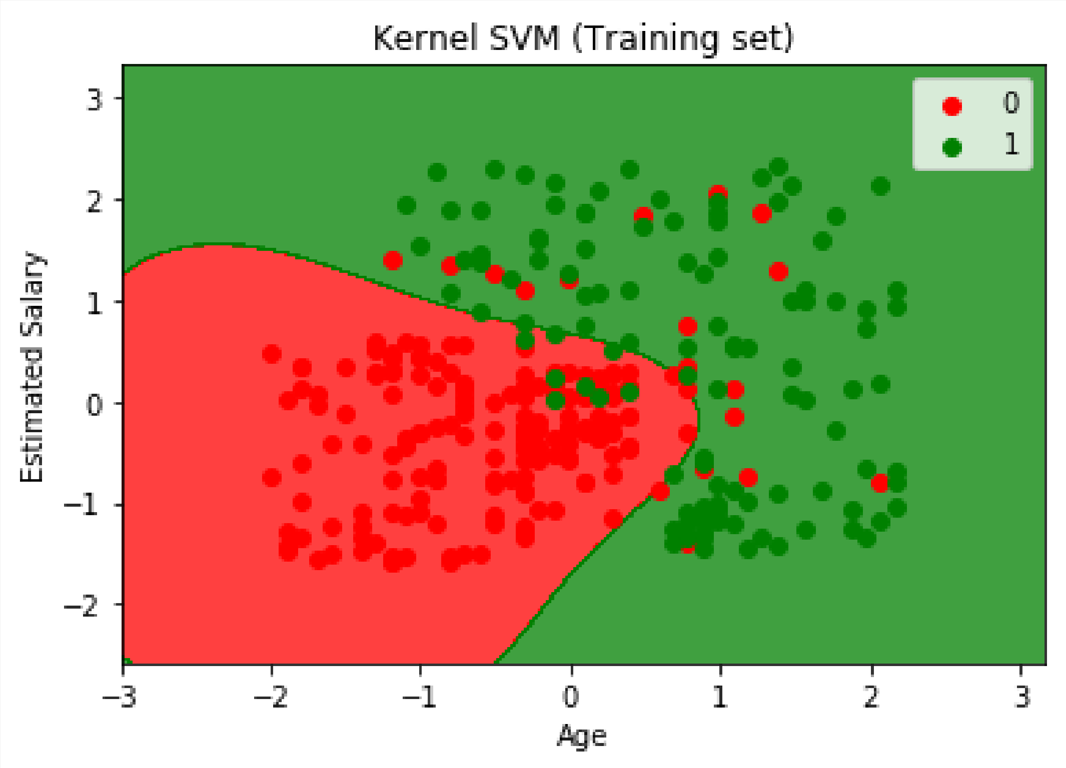
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

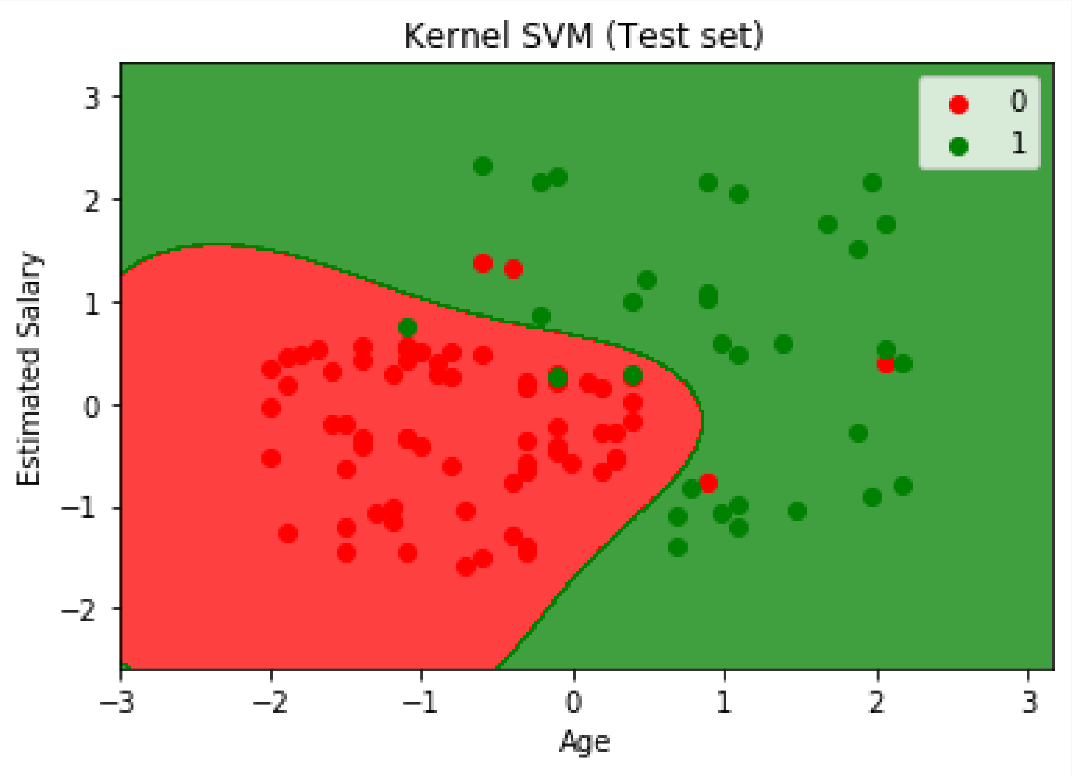
```python
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
```

rbf		預測值 y_pred	
		0	1
實際值 y_test	0	64	4
	1	3	29
準確率：93%			

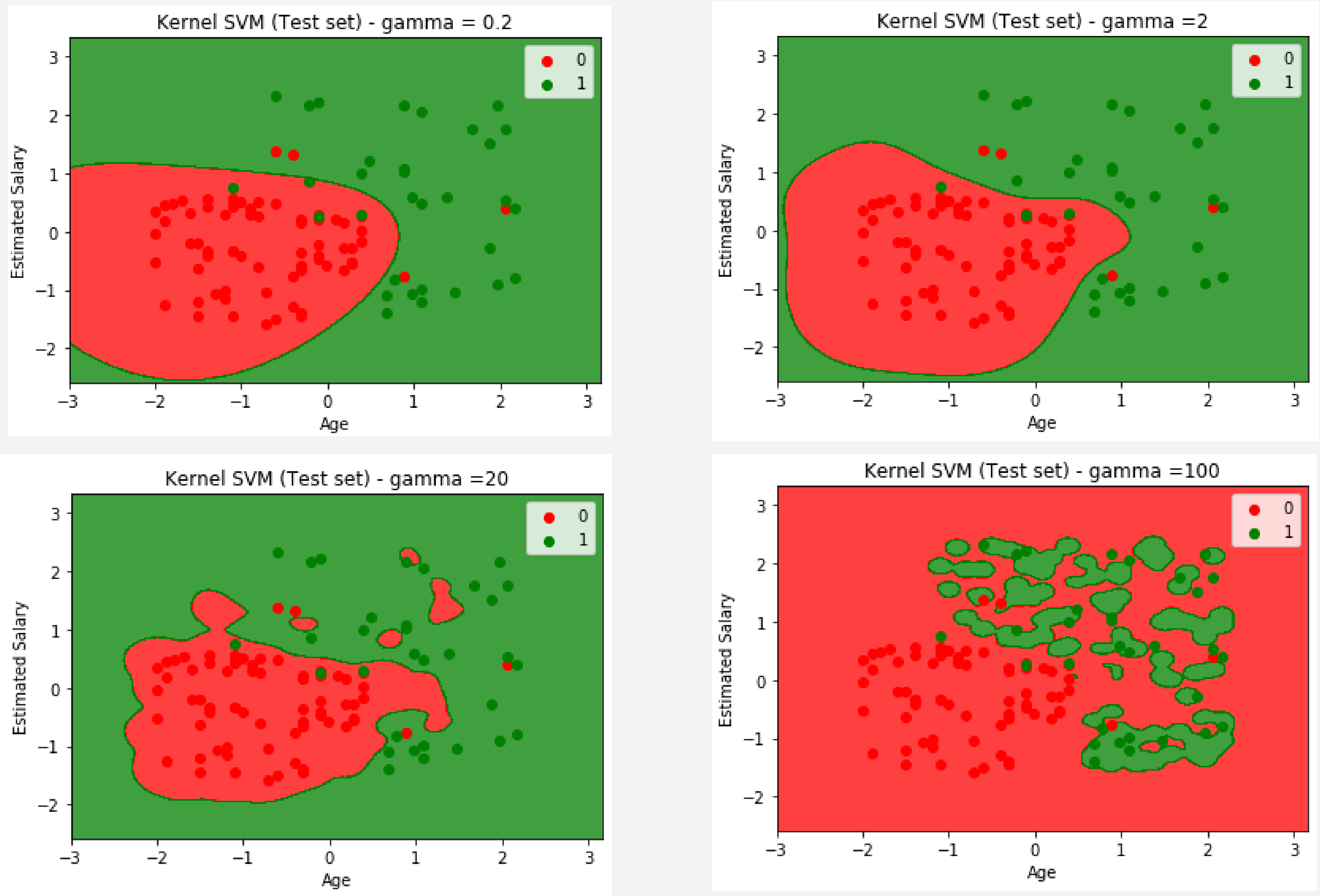
linear		預測值 y_pred	
		0	1
實際值 y_test	0	66	2
	1	8	24
準確率：90%			

RBF高斯函數預測結果 (TRAINING & TEST)





RBF高斯函數 – 各種不同GAMMA值比較



RBF高斯函數 – 精確度與結論

gamma=0.2		預測值 y_pred	
		0	1
實際值 y_test	0	64	4
	1	4	28
準確率：92%			

gamma=2		預測值 y_pred	
		0	1
實際值 y_test	0	64	4
	1	3	29
準確率：93%			

gamma=20		預測值 y_pred	
		0	1
實際值 y_test	0	64	4
	1	3	29
準確率：93%			

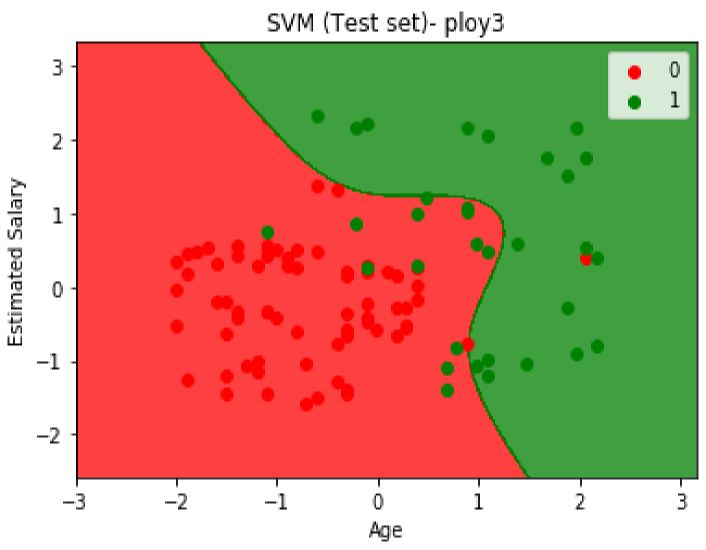
gamma=100		預測值 y_pred	
		0	1
實際值 y_test	0	66	2
	1	16	16
準確率：82%			

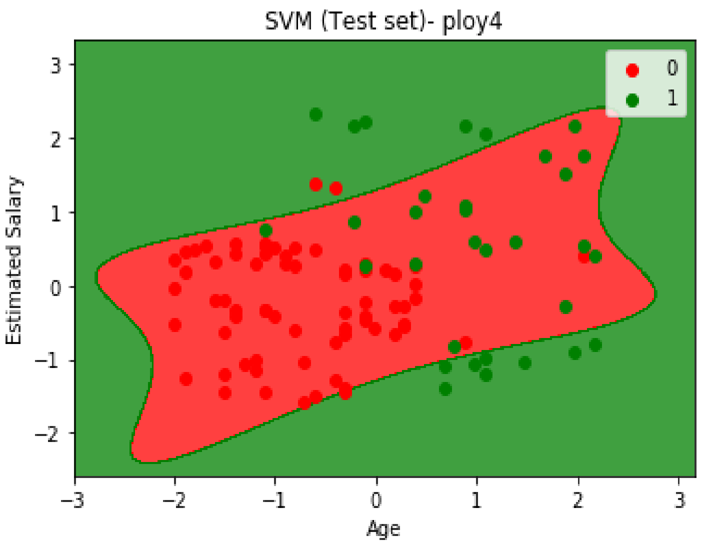
- 可以看出調高gamma值，配適較好，但過度配適，會造成準確率下滑。

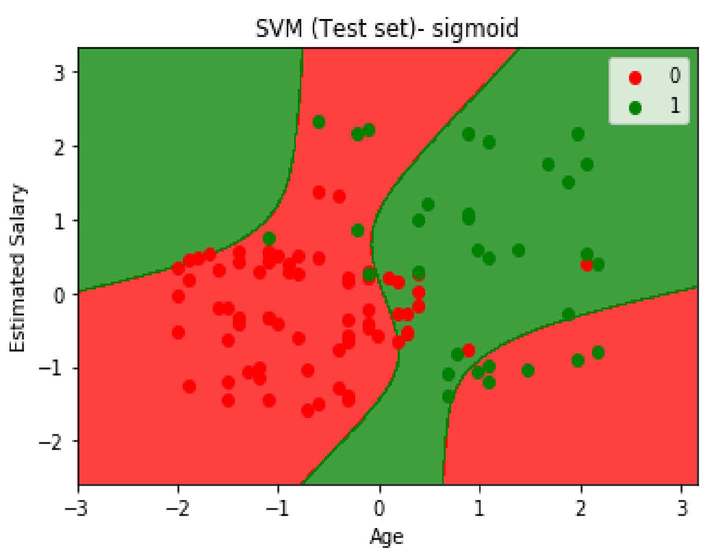
修改的GAMMA程式碼

```
from sklearn.svm import SVC
classifierg2 = SVC(kernel = 'rbf', random_state = 0 ,gamma = 2  )
classifierg2.fit(X_train, y_train)
y_pred = classifierg2.predict(X_test)
from sklearn.metrics import confusion_matrix
cm2 = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
print('Accuracy: %.2f' % accuracy_score(y_test, y_pred))
plt.contourf(X1, X2, classifierg2.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

其餘函數結果 (POLYNOMIAL、SIGMOID)







ploy3		預測值 y_pred	
		0	1
實際值 y_test	0	67	1
	1	13	19
準確率：86%			

ploy4		預測值 y_pred	
		0	1
實際值 y_test	0	66	2
	1	19	13
準確率：79%			

sigmoid		預測值 y_pred	
		0	1
實際值 y_test	0	54	14
	1	12	20
準確率：74%			