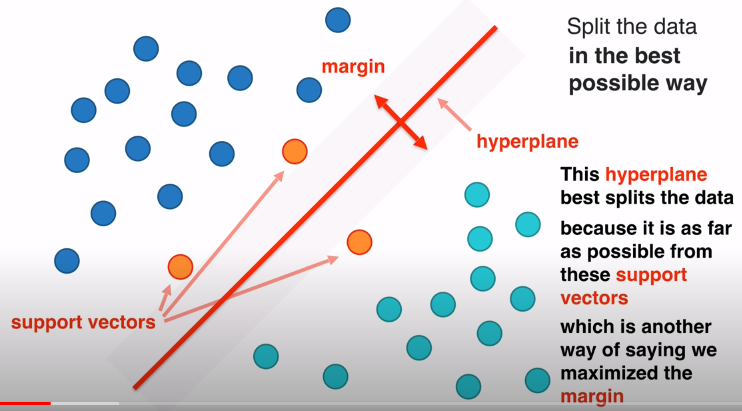
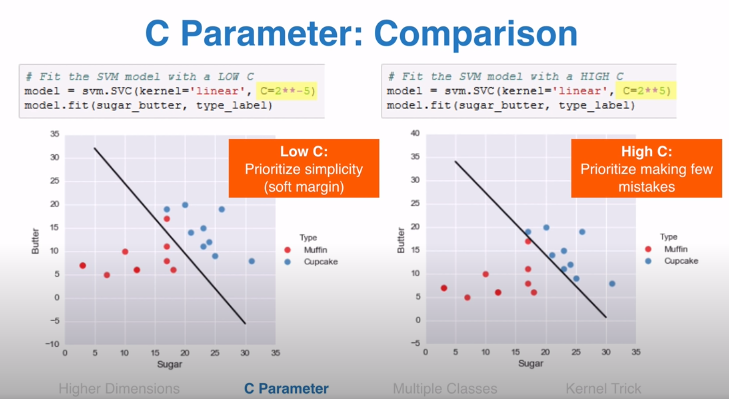
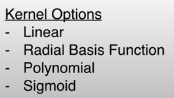
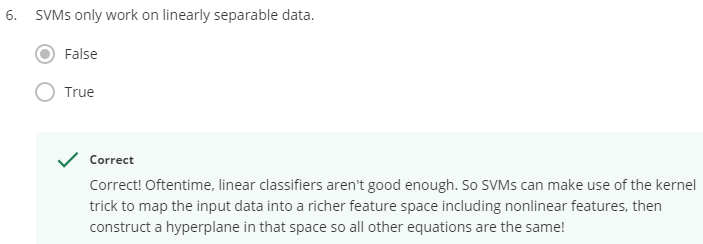


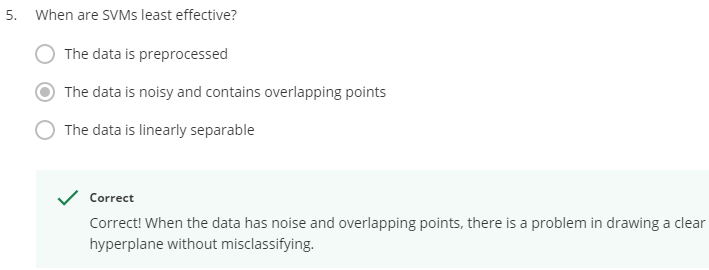
* [Tutorial - 1](https://youtu.be/N1vOgolbjSc)



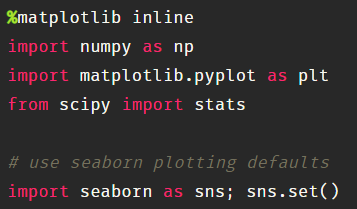


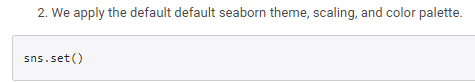




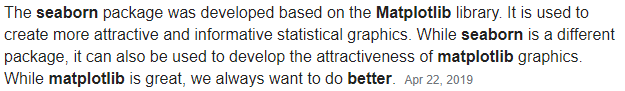


* #1

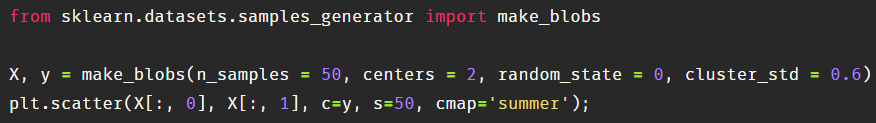


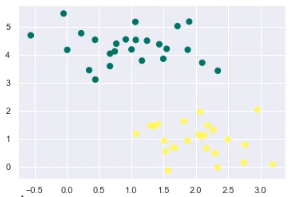
*  This will show the graph or image inside notebook, if we don’t use this line, the image may show in different pop up window.
* 



* 

* #2





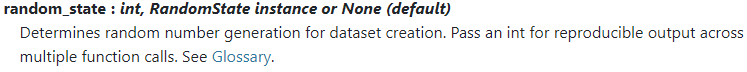
**y**

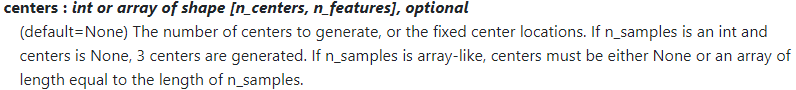
**x**

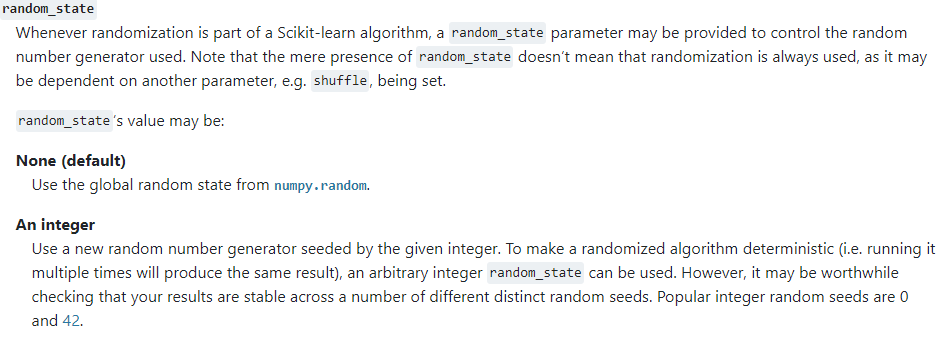
* Green = active volcanoes
* Yellow = dormant volcanoes

make\_blobs is some random Gaussian blobs

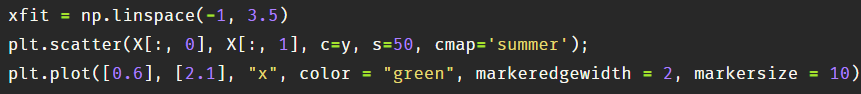


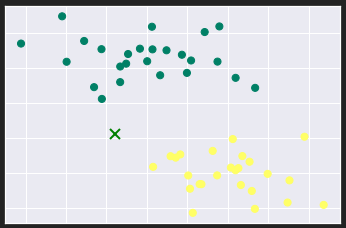


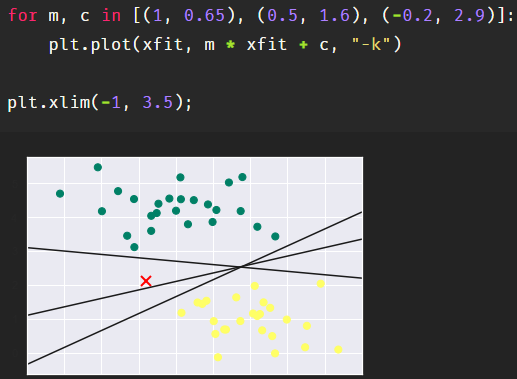




* #3







We know, y = mx + c where

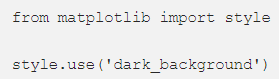
* m = slope
* c = y-intercept

 3 tuples

[(m1, c1), (m2, c2), (m3, c3)]

* “-k” is color code for black

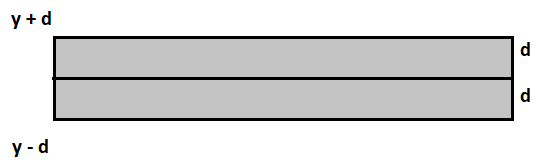
**\*\*** For dark background in the jupyter notebook, I couldn’t see the axis details. Solution

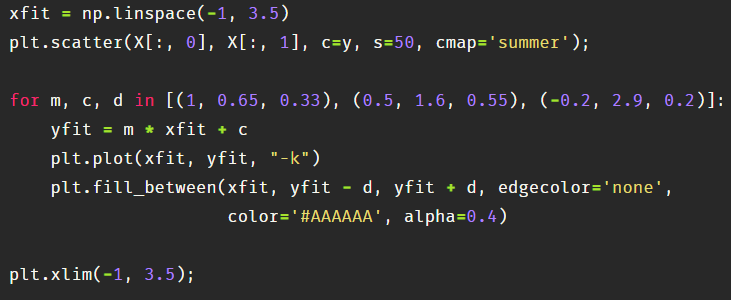
* 



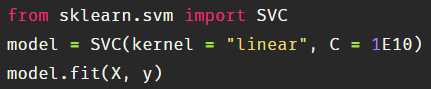
\*\* Now the problem is – there are infinite number of decision boundaries. How to select the best one from them?

* SVM use a concept named – margin
* #4

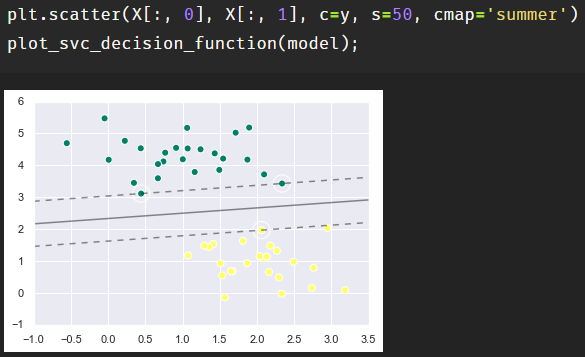




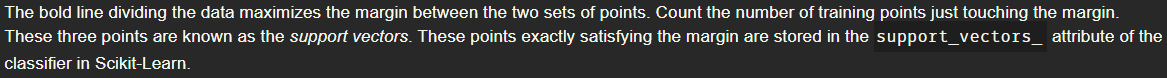
* alpha = optaity
* #5

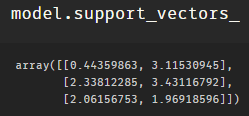


* #6



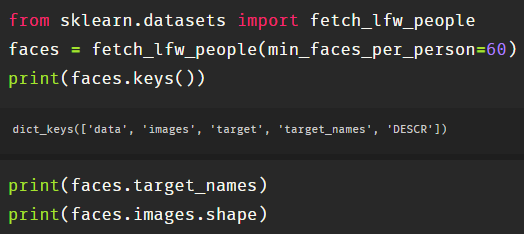
Here, there are 3 support vectors



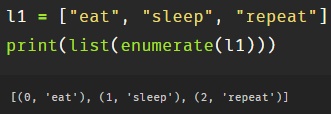
* Co-ordinates of the support vectors: 

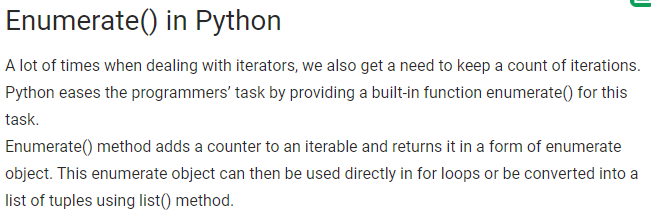


* For face recognition, we need dataset
* #1



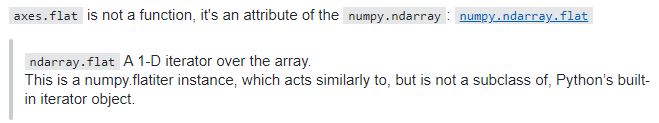
* #2





* #3

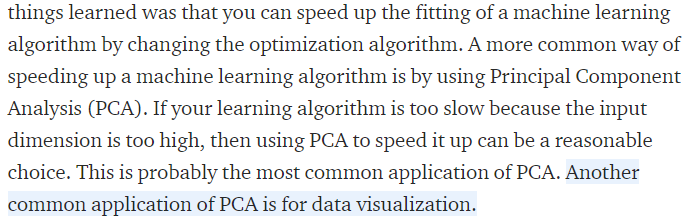




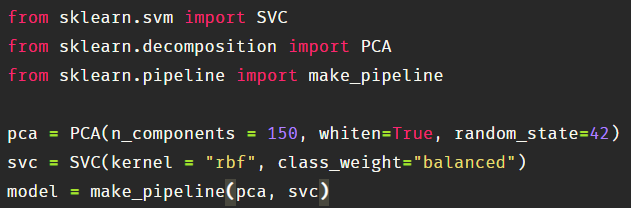
Steps:

* plotting 3 \* 5 grid
* enumerating ax.flat which iterates every cell in the grid from left to right
* every grid will contain an image which is defined by using imshow()
* every grid will contain name of the person in the image, which is set by xlabel



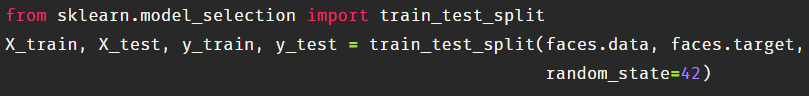


* #1



\*\* previous version of PCA is RandomizedPCA which is deprecated now

* #2

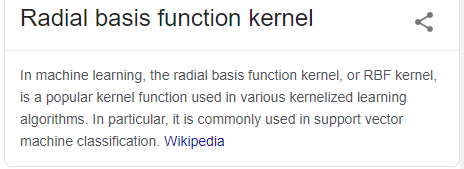


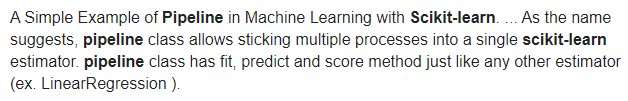
\*\* deprecated 

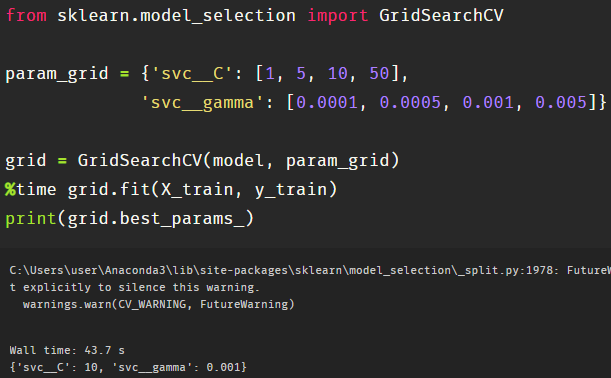
* #Grid-search model selection

c = margin hardness

gamma = size of rbf







Now with this cross-validated model, we can now predict the labels of our test data

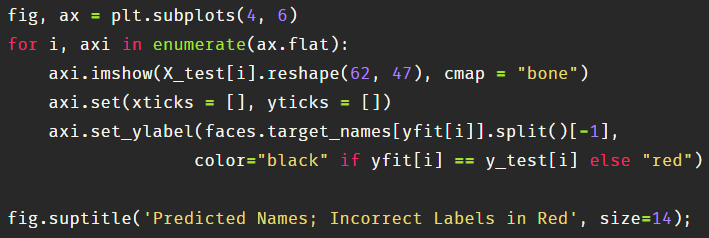




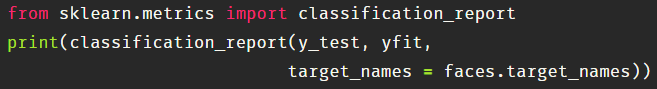
* Now again draw a 4 \* 6 subplot



* Let’s plot image iterator







* Drawing a heatmap to see where it tends to do misclassification

