INF283 – Exercise 4

Sindre Eik de Lange

# 1. Support Vector Machines (SVMs)

1.1

1. Code to find the dimensions of a vector **X** is already defined as *X\_shape*, which, according to *make\_blobs()* DocStrings is equal to (**n\_samples**, **n\_features**).

2. Code to find the dimensions of a vector y is already defined as *y\_shape*, which, according to *make\_blobs()* DocStrings is equal to (**n\_samples**).

3. Again, according to *make\_blobs()* DocString, vector **y** contains: 'The integer labels for cluster membership of each sample', aka. each tuples' (in **X**) label.

1.2

**m** and **b** are the values in each touple, contained in the list (of touples) iterated through in the for-loop. More specifically, the for-loop iterates through a list, containing touples, consisting of values (**m\_i**, **b\_i**), where **i** = iteration number. They represent the values **a** and **b** for linear functions, given as *y=ax+b*, where **xfit** then represents the **x**.

1.3

**d** represents the distance between each vectors' line and the nearest data point on each side.

1.4

1. Code to print out the value of support vectors: *model.support\_vectors\_* (after assigning **model** as the SVM classifier).

2. There are 3 support vectors.

1.5

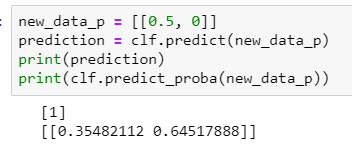
At which point did we center the **r**, while calculating it?

**r** = np.exp(-(X \*\* 2).sum(1)), hvor X = datapunktene en skal mappe til annen dimensjon. Tilsynelatende eksempel på en *radial basis function*, som er sentrert på «midtre klump».

* Hva er «midtre klump»? (aka. «middle clump»)

TODO

1.6

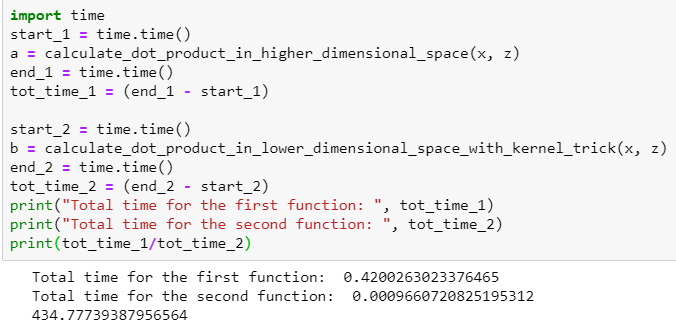


1.7

This is probably data dependent like everything else in machine learning, but I would assume on would utilize a validation set, combined with either GridSearch or RandomSearch with cross validation (depending on available resources), in order to get the optimal hyperparameter.

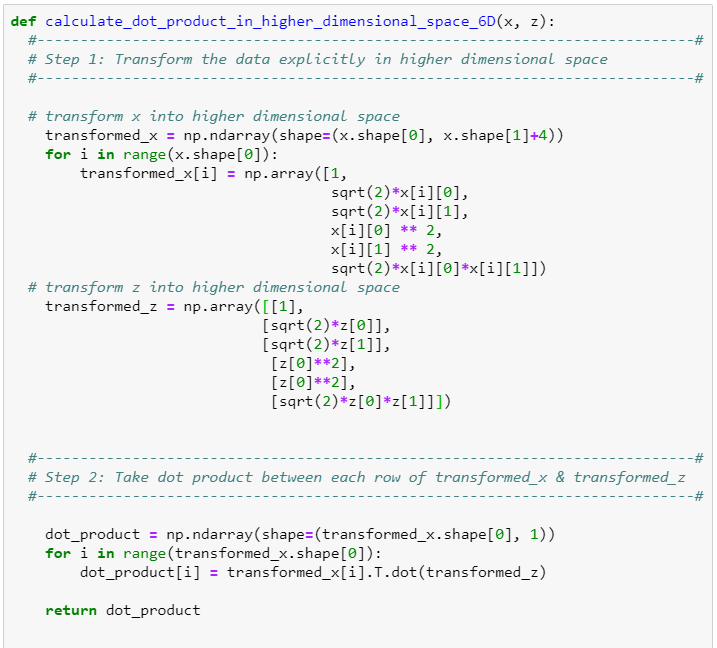
TODO: Update – Alexander.

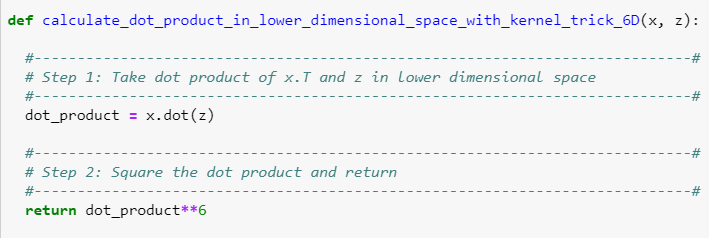
1.8

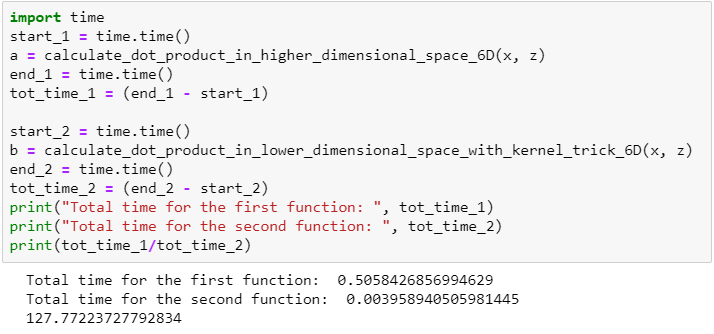


The **kernel trick** function is faster than the other one.

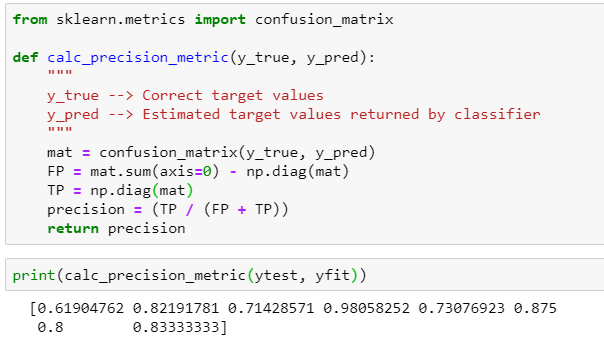
1.9







1.10

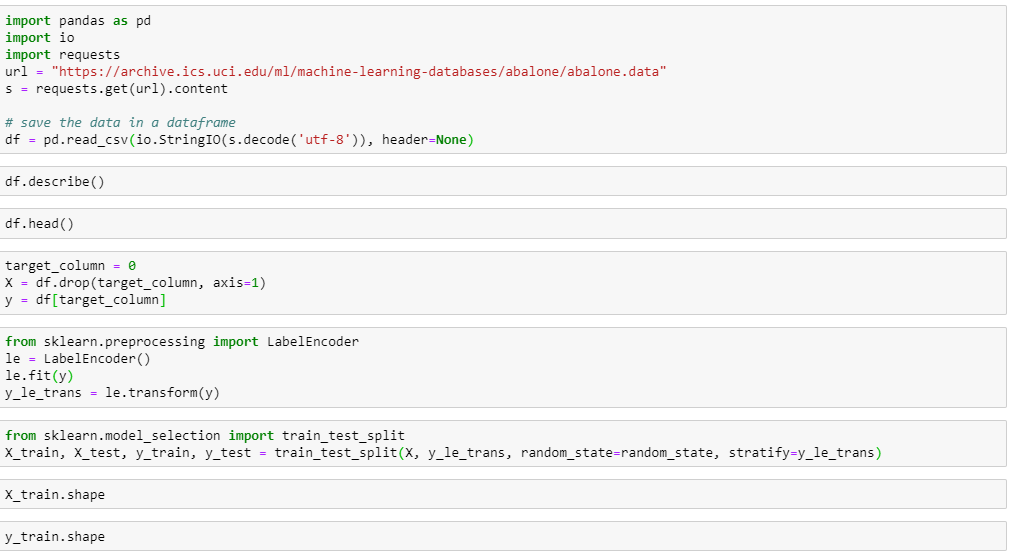


# 2. Ensemble Learning

2.1

Changing the ‘voting’ parameter from ‘hard’ to ‘soft’ slightly improves the **VotingClassifier**. This is because the ‘hard’ classifier only considers the binary input from each voter, while the ‘soft’ classifier takes into account the average of probabilities. This means that in our case, where each of the “weak” learners are fairly good (> .85), it helps to consider how certain each classifier is.

2.2







So, one can see that the **Bagging Classifier** is the best classifier. Important to note that this is based on running the models one time, and not utilizing any hyperparameter optimization tools like GridSearch(CV) or RandomSearch(CV).