Report challenge two

Andrea Tonello

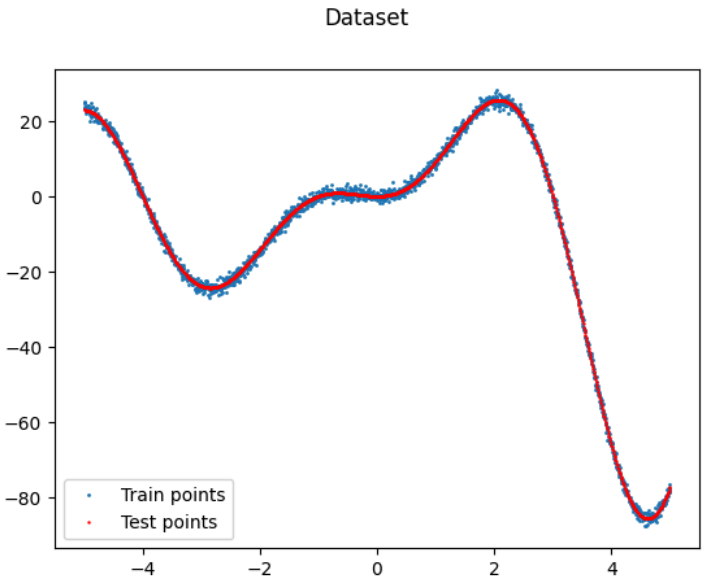
17/04/2024

Part 1: Ridge regression

We want to show the differences between a standard Linear ridge regression model, a Gaussian kernel ridge regression model, and a Polynomial kernel ridge regression model.
For this task, we generate a dataset with 3000 entries. 2000 will be used for the training set and 1000 for the test set.
Our train set is ultimately given by this function:

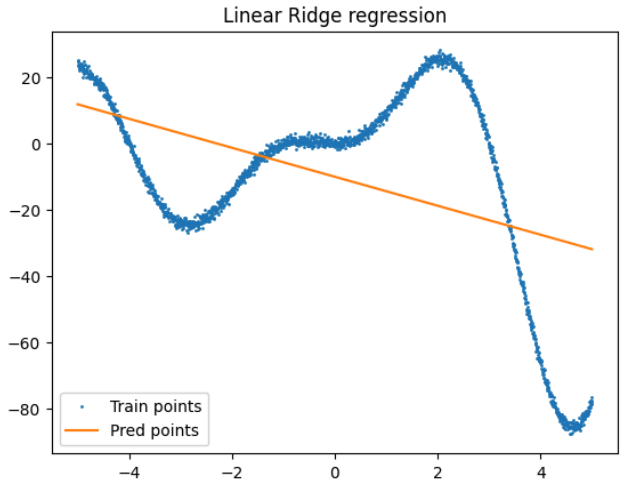
$$y = (X + 4) \cdot (X + 1) \cdot (cos(X) - 1) \cdot (X - 3) + eps$$

with eps drawn from a normal distribution $N(0, 1)$.



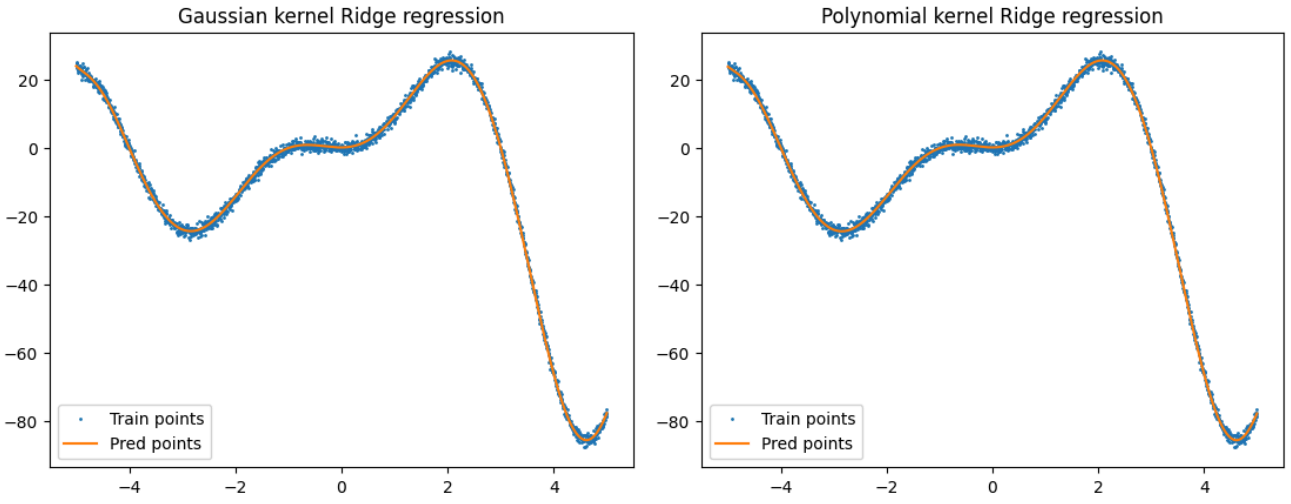
We will evaluate the models using RMSE.

Training the standard Linear ridge regression model, we find the expected:



A straight line cannot properly model our data, and the reported RMSE is 26.75

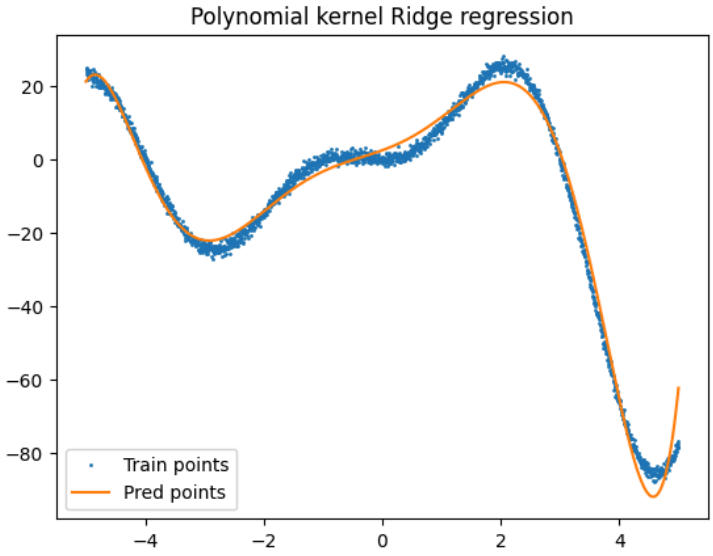
To tune the hyperparameters of the two kernel-based models, we perform a grid search with 5-fold cross validation.
Gaussian kernel hyperparameter search (gamma): 0.001, 0.01, 0.1, 1, 1.5, 3, 10
Polynomial kernel hyperparameter search (degree): 7, 8, 9, 10, 11, 12, 13



These graphs are produced with the best hyperparameters found during the grid search, which are gamma = 0.1 for the gaussian model and degree = 13 for the polynomial model.

The RMSEs are 0.1184 and 0.1118 respectively, so the two models are fairly comparable in terms of performance.

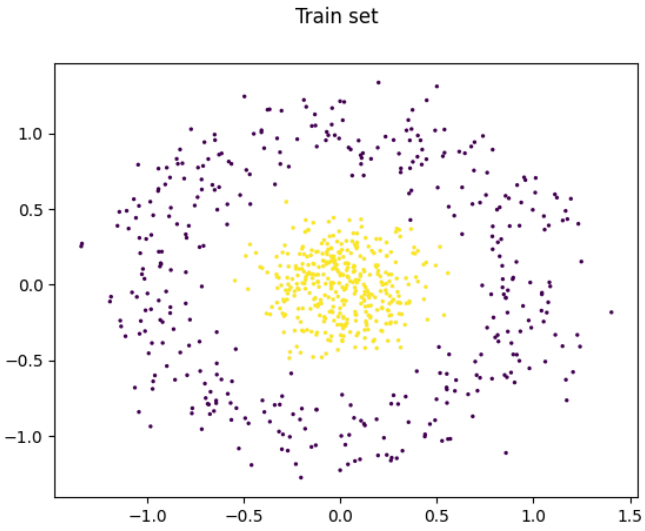
However, we observe significant performance degradation in the polynomial model if we were to choose a lower degree, for example 7.



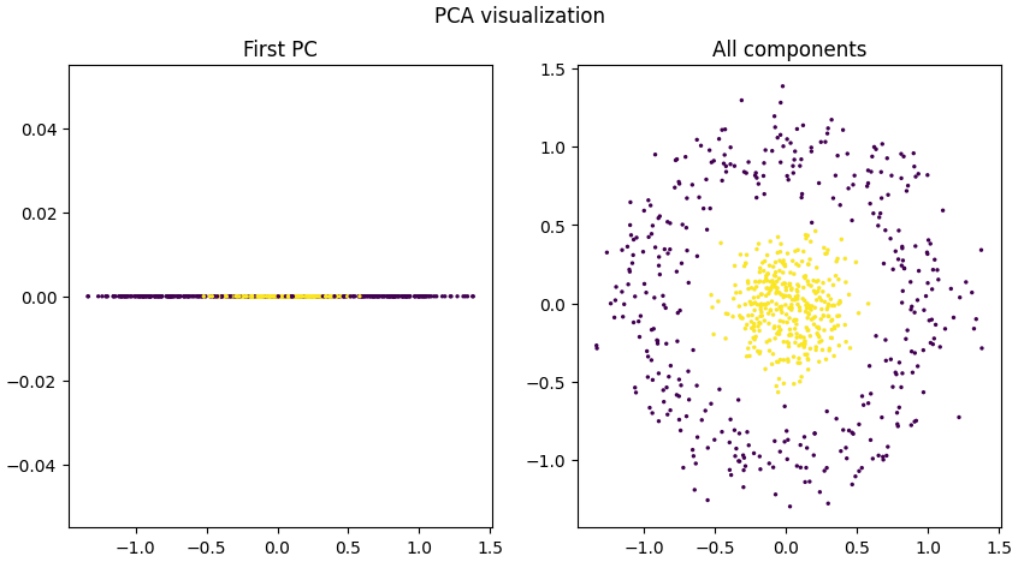
The same performance hit is not as hard on the gaussian kernel if we were to change the gamma value; overall, we can say that the Gaussian kernel ridge regression model has a “broader” application for this particular dataset and may be preferable.

Part 2: PCA on a circular dataset

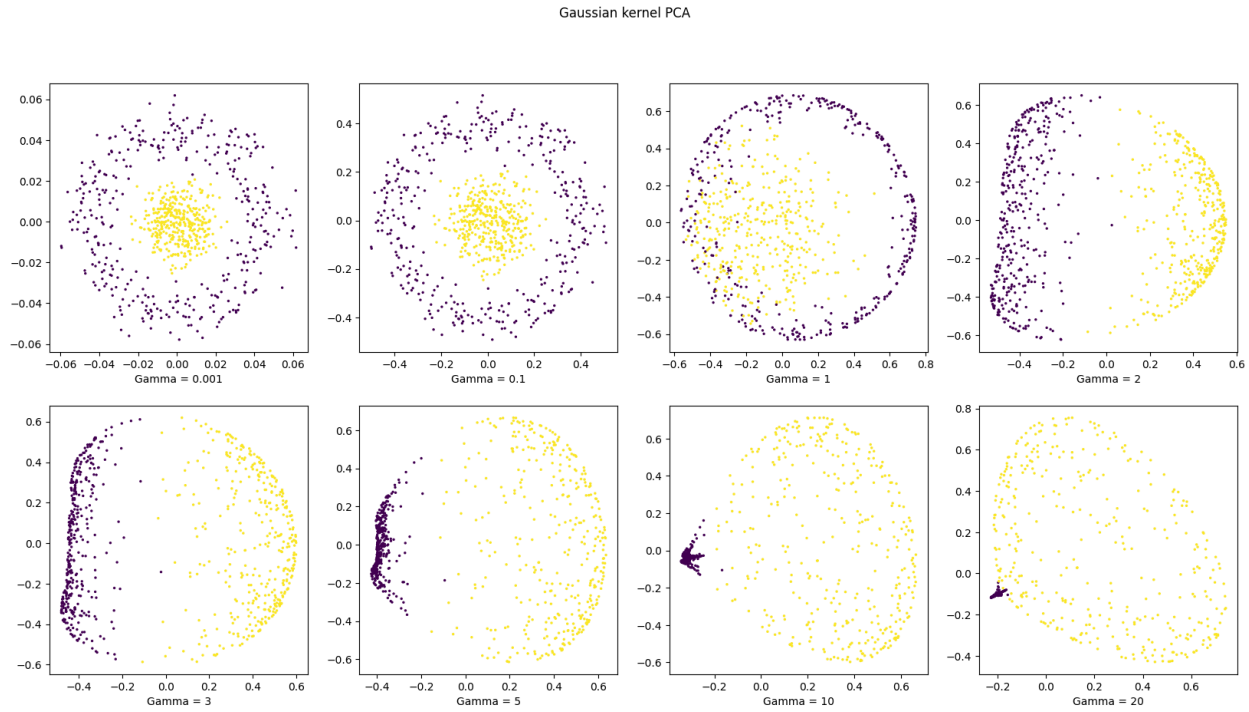
Now we take a look at a classification problem, which we want to solve using PCA and its gaussian kernel variant.
The generated dataset has 1000 entries (750 for the train set, 250 for the test set) and two circular clusters.



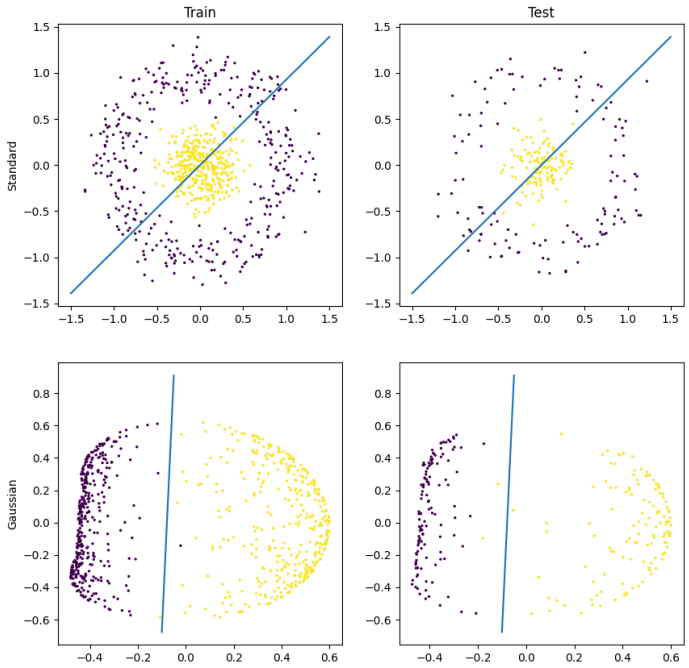
Just by looking at the dataset, we can already assume that standard PCA won't be able to do very much, since our data is definitely not linear; this is confirmed by computing PCA:



Let’s resort to kernelization by using Gaussian kernel PCA to see if our data becomes linearly separable in higher dimensions.
Testing various gamma values through grid search yields the following results:



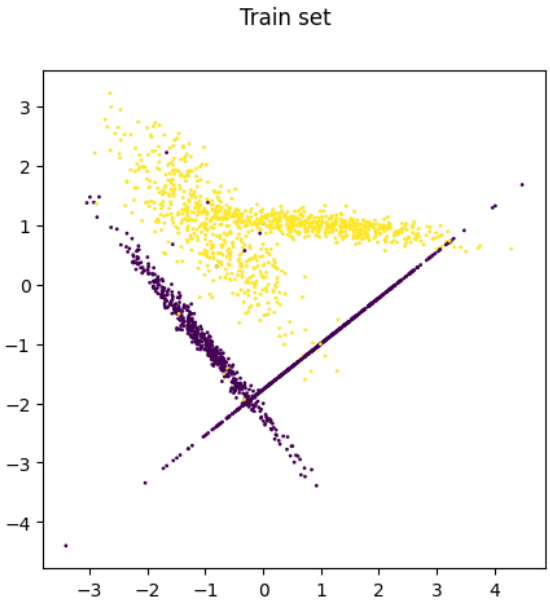
We start to see linearity from gamma >= 2. Let’s compute SVM in the gamma = 3 instance:



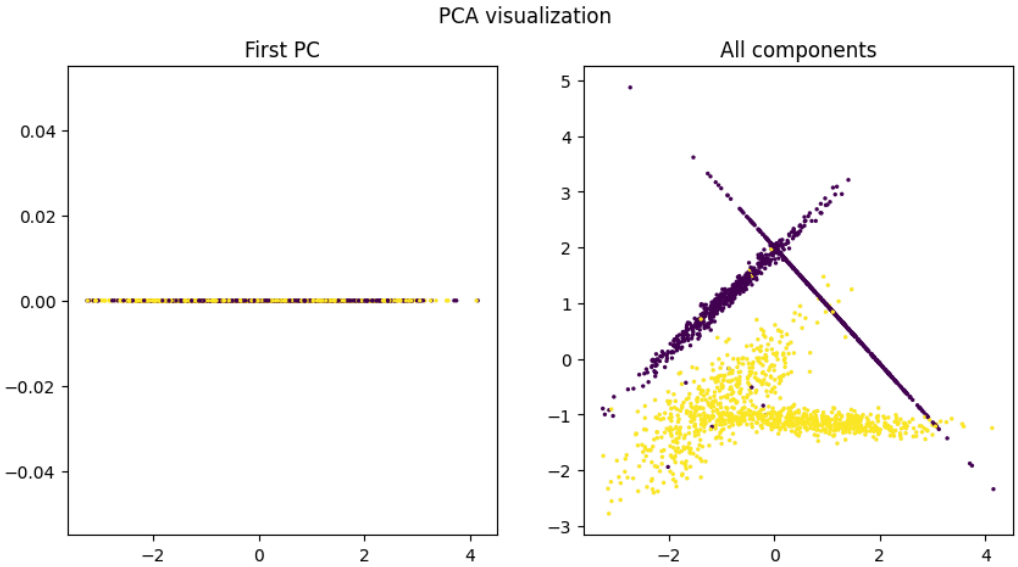
As assumed, the standard PCA cannot properly separate the two classes, resulting in an accuracy of 0.496, which can also be seen by looking at how the linear separator cuts right in the middle of the circles.
Gaussian PCA on the other hand is able to linearly separate the data much better, resulting in an accuracy score of 0.992.

Part 3: PCA on a random dataset

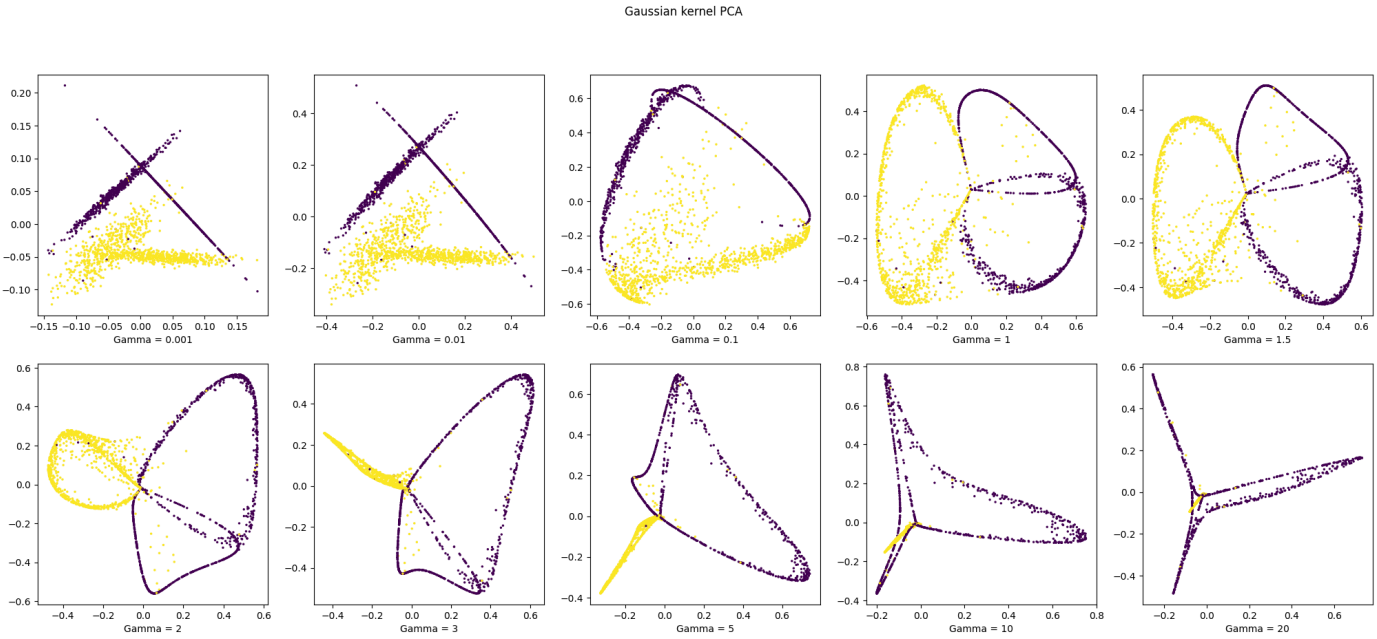
In this part we make use of a dataset generated via `make_classification()`, with 3000 entries (2000 for training, 1000 for testing).



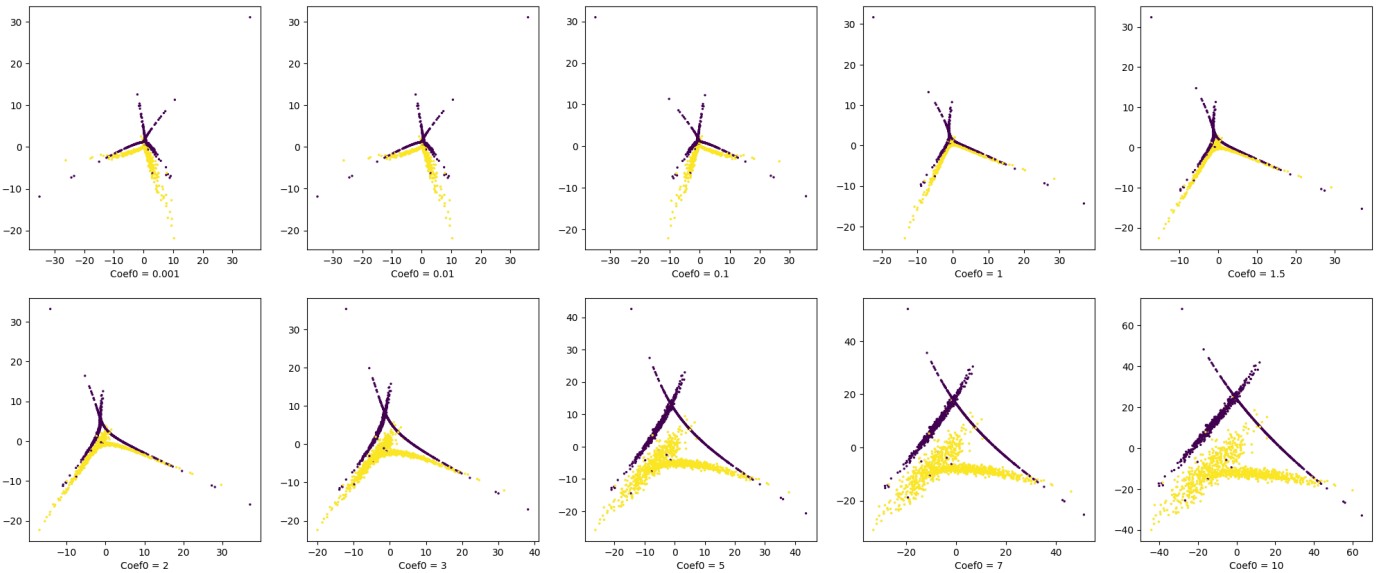
The same steps as in Part 2 are applied; this time, however, we also involve Polynomial kernel PCA.
Standard PCA yields:



Gaussian kernel PCA yields:
(gamma hyperparameter range: 0.001, 0.01, 0.1, 1, 1.5, 2, 3, 5, 10, 20)

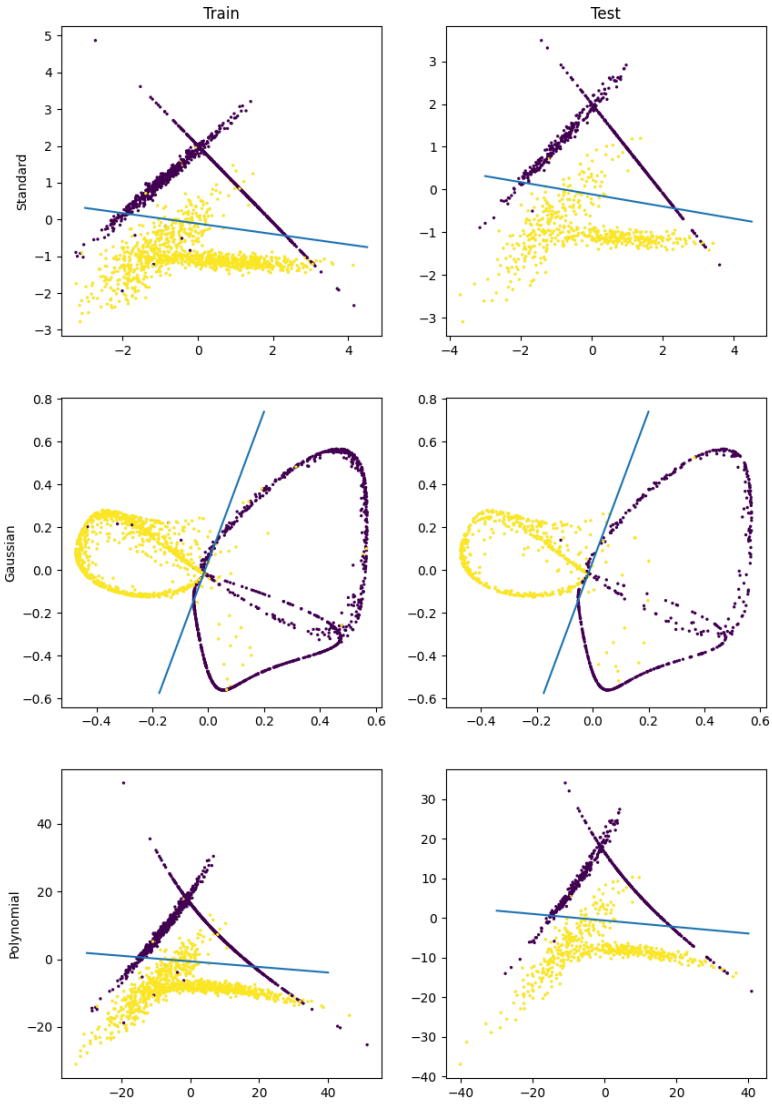


Polynomial kernel PCA yields:
(polynomial degree =3 , coef0 hyperparameter range: 0.001, 0.01, 0.1, 1, 1.5, 2, 3, 5, 7, 10)



The polynomial degree did not seem to play a crucial role: two major points dispositions were found, one given by even degrees and one given by odd degrees. Odd degrees produced more clearly separable data, and so 3 was chosen as the degree of choice for this problem.

Having chosen gamma = 2 for Gaussian PCA and coef0 = 7 for Polynomial PCA, SVM yields:



with accuracy scores of 0.917, 0.932 and 0.892 respectively.
Polynomial kernel PCA is in fact performing worse than standard PCA, probably due to the fact that the data was already fairly linearly separable from the start.