Fadhla Mohamed Mutua

SM3201434

1. Introduction

This study investigates three machine learning parts with simulated data from random numbers. The random number generation comes from the use of numpy `np.linspace(-5, 5, train_points)` for linear ridge and kernel ridge (where `train_points` is 20 for traning and 1000 for testing) regression and sklearn `sklearn.datasets.make_circles` for PCA. The first part compares linear ridge regression with kernel ridge regression (using both Gaussian and polynomial kernels) to assess regression performance under different model assumptions and kernel formulations. The second part compares traditional Principal Component Analysis (PCA) with Kernel PCA (KPCA) to determine how both methods perform in reducing dimensionality on structured data. A third "experimental" section repeats the PCA versus KPCA comparison on a dataset made using `sklearn.datasets.make_classification`, highlighting the sensitivity of kernel-based methods.

2. Ridge Regression vs. Kernel Ridge Regression

2.1 Objective and Methodology

- **Objective**: The goal was to determine whether kernelizing the ridge regression approach (with either Gaussian or polynomial kernels) could capture non-linear relationships in data better than standard linear ridge regression.
- **Methodology**:
  - Split of training dataset: The dataset consists of 20 training data values in a range from -5 to 5 and whose y value is gotten from the formula:
    - $y_{train} = (X_{train} + 4) * (X_{train} + 1) * (np.cos(X_{train}) - 1) * (X_{train} - 3) + eps$
    - With $eps =$ `np.random.normal(0, 1, train_points)` an array of random movements (obtained from the normal distribution) from the true value of y
  - Split of testing dataset: The dataset consists of 1000 testing data valus in a range from -5 to 5 and whose y value is gotten from the formula:
    - $y_{test} = (X_{test} + 4) * (X_{test} + 1) * (np.cos(X_{test}) - 1) * (X_{test} - 3)$

2.2 Linear Ridge Regression:

A regularized linear model that imposes penalties on the coefficient sizes to avoid overfitting.

- **Formula used**: Given the formula for linear regression $Y = X * \omega + b$, by including the bias term as a feature (in this case all 1), we can express the model as
  - $Y = X_{training\_bias} * \omega$ from which
  - $\omega = (X^T_{training\_bias} * X_{training\_bias} + \lambda * I)^{-1} * X^T_{training\_bias} * Y$
  - The prediction is then formulated as the line $Y_{predict} = X_{training\_bias} * \omega$
- **Observation**: From the data, we observe that the prediction line $y_{prediction}$ does not accurately represent the data. In fact, the high RMSE suggests significant underfitting.

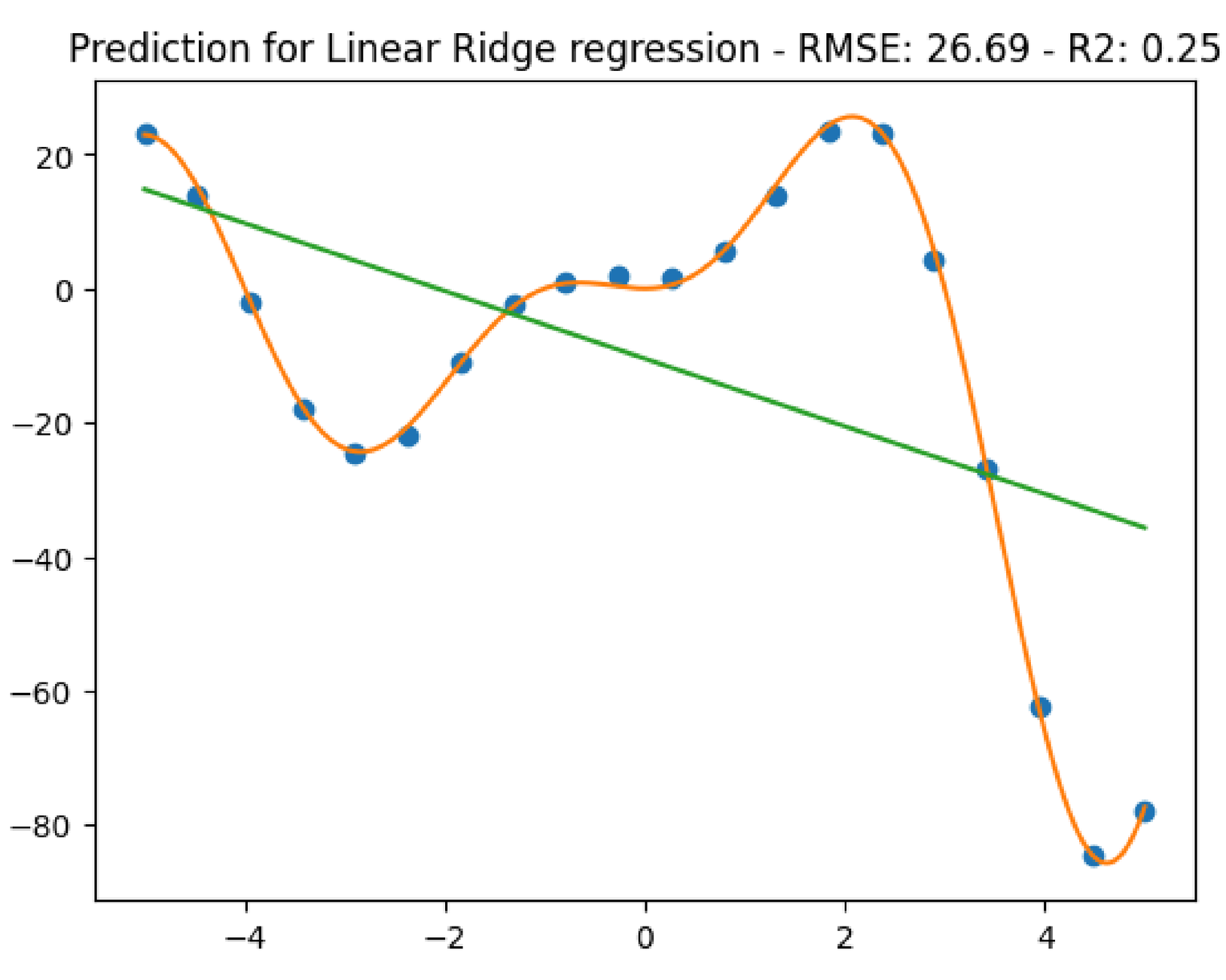

Figure 1: Plot for Linear Ridge Regression with RMSE = 26.69

2.3 Kernel Ridge Regression (KRR):

Extends ridge regression by mapping data to a higher-dimensional space.

- **Formula used**: To make our regression non linear, we utilize kernels where kernel $k(x_1, x_2) = \phi(x_1)^T * \phi(x_2)$. By having

$$\Phi(X) = \begin{pmatrix} \phi(x_1) \\ . \\ . \\ . \\ \phi(x_n) \end{pmatrix}$$

- And replacing all X in the previous linear ridge regression with $\Phi(X)$ we get:

$$\omega = \Phi(X)^T * (\Phi(X)^T * \Phi(X) + \lambda * I)^{-1} * Y$$

- Then we get

$$Y_{predict} = \phi(x)^T * \omega = \phi(x)^T * \Phi(X)^T * (\Phi(X)^T * \Phi(X) + \lambda * I)^{-1} * Y$$

- We can then replace:

$$\Phi(X)^T * \Phi(X) = K$$
$$(K)_{i,j} = k(x_i, x_j)$$
$$\phi(x)^T * \Phi(X)^T = \sum_{i=1}^n \phi(x)^T * \phi(x_i)$$
$$\sum_{i=1}^n \phi(x)^T * \phi(x_i) = \sum_{i=1}^n k(x, x_i)$$

- From which:

$$\alpha = (K + \lambda * I)^{-1} * Y$$
$$Y_{predict} = \sum_{i=1}^n \alpha_i * k(x, x_i)$$

Two kernels were tested:

2.3.1 Gaussian (RBF) Kernel

- **Formula used**: Applying the Gaussian Kernel we have:
  - $k(x, x') = e^{-\|x-x'\|^2/(2*\sigma^2)}$ with $\sigma > 0$
- **Observation**: By applying a grid search on sigma_grid = [0.01, 0.1, 1, 5, 10] and lambda_grid = [0.01, 0.1, 1, 5, 10]

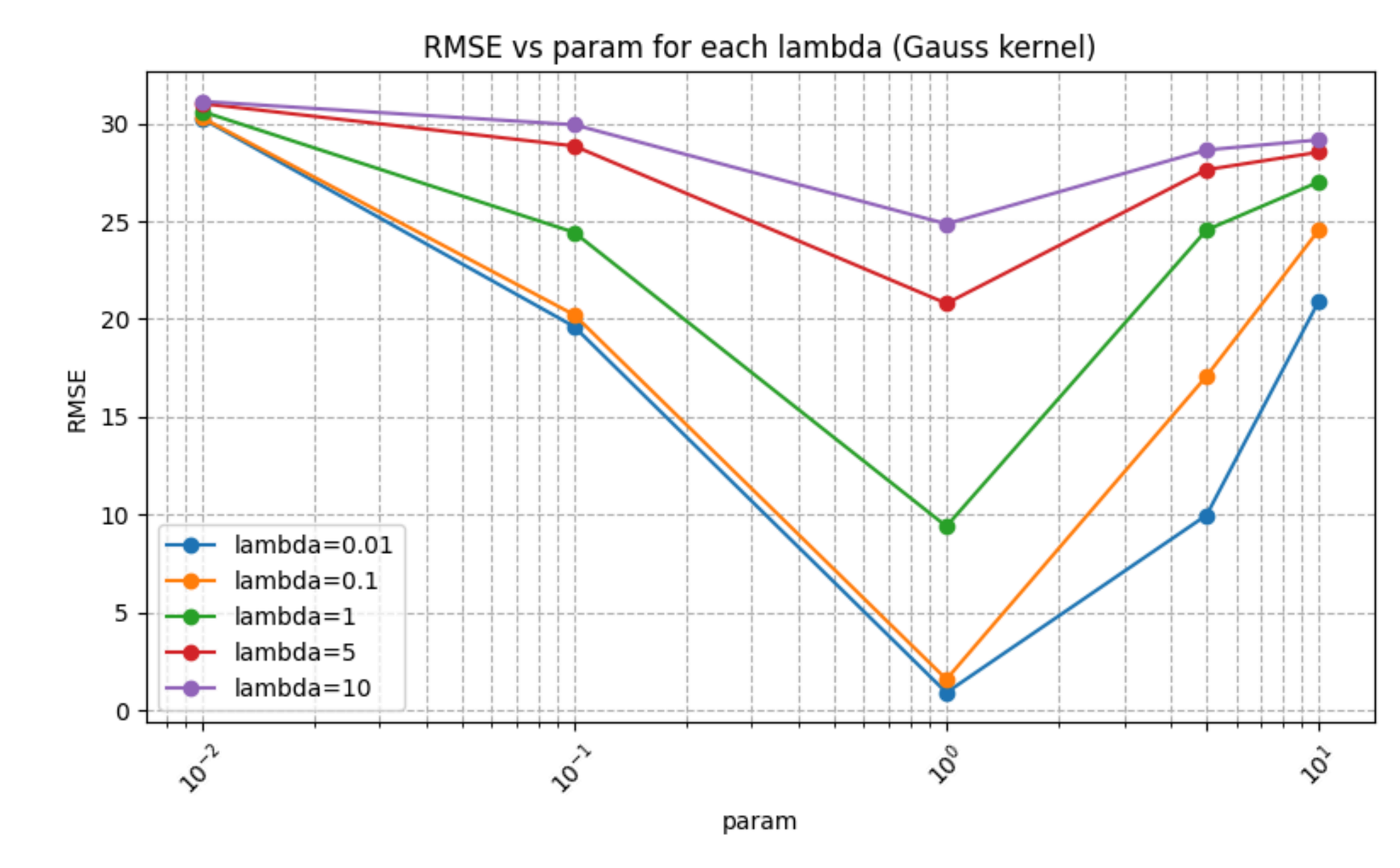

Figure 2: Plot for RMSE vs Sigma Gauss

- We find that the best fit curve has $\sigma = 1$ and $\lambda = 0.01$

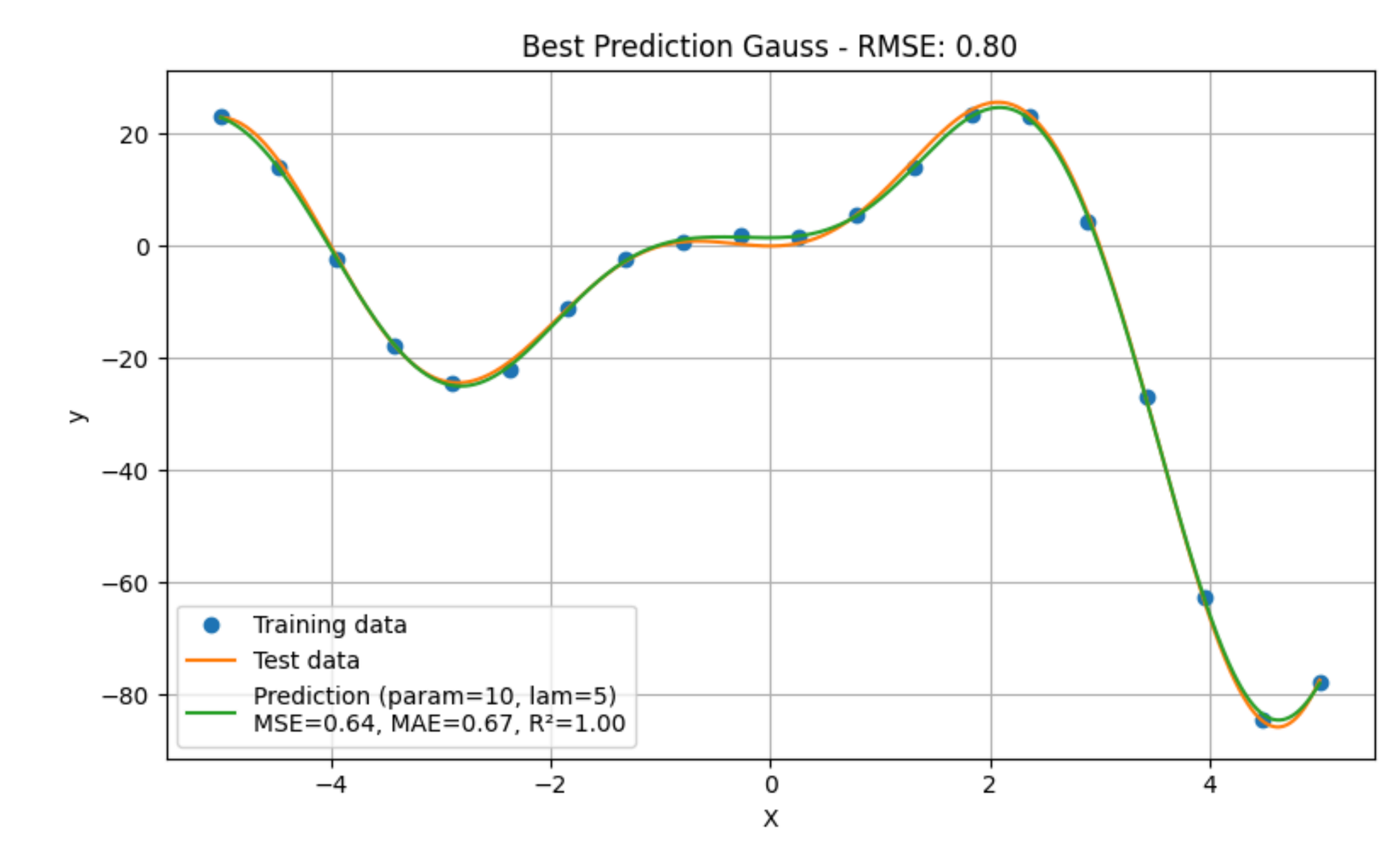

Figure 3: Plot for Kernel Ridge Rigression (Gauss) with RMSE = 0.80

2.3.2 Polynomial Kernel

- **Formula used**: Applying the Polynomial Kernel we have:
  - $k(x, x') = (x^T x' + 1)^\sigma$
- **Observation**: By applying a grid search on sigma_grid = [0.01, 0.1, 1, 5, 10] and lambda_grid = [0.01, 0.1, 1, 5, 10]

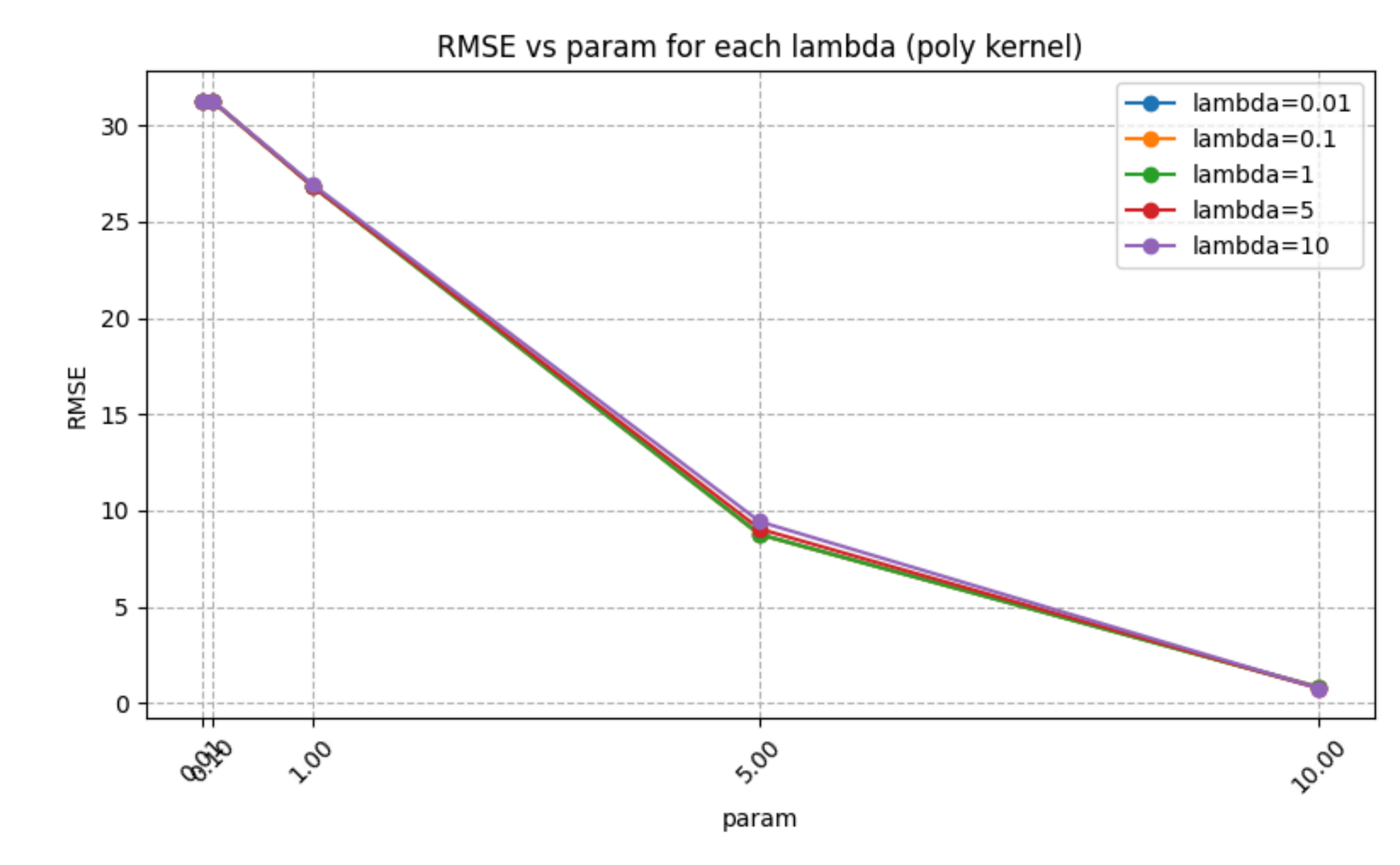

Figure 4: Plot for RMSE vs Sigma

- We find that the best fit curve has $\sigma = 10$ and $\lambda = 5$

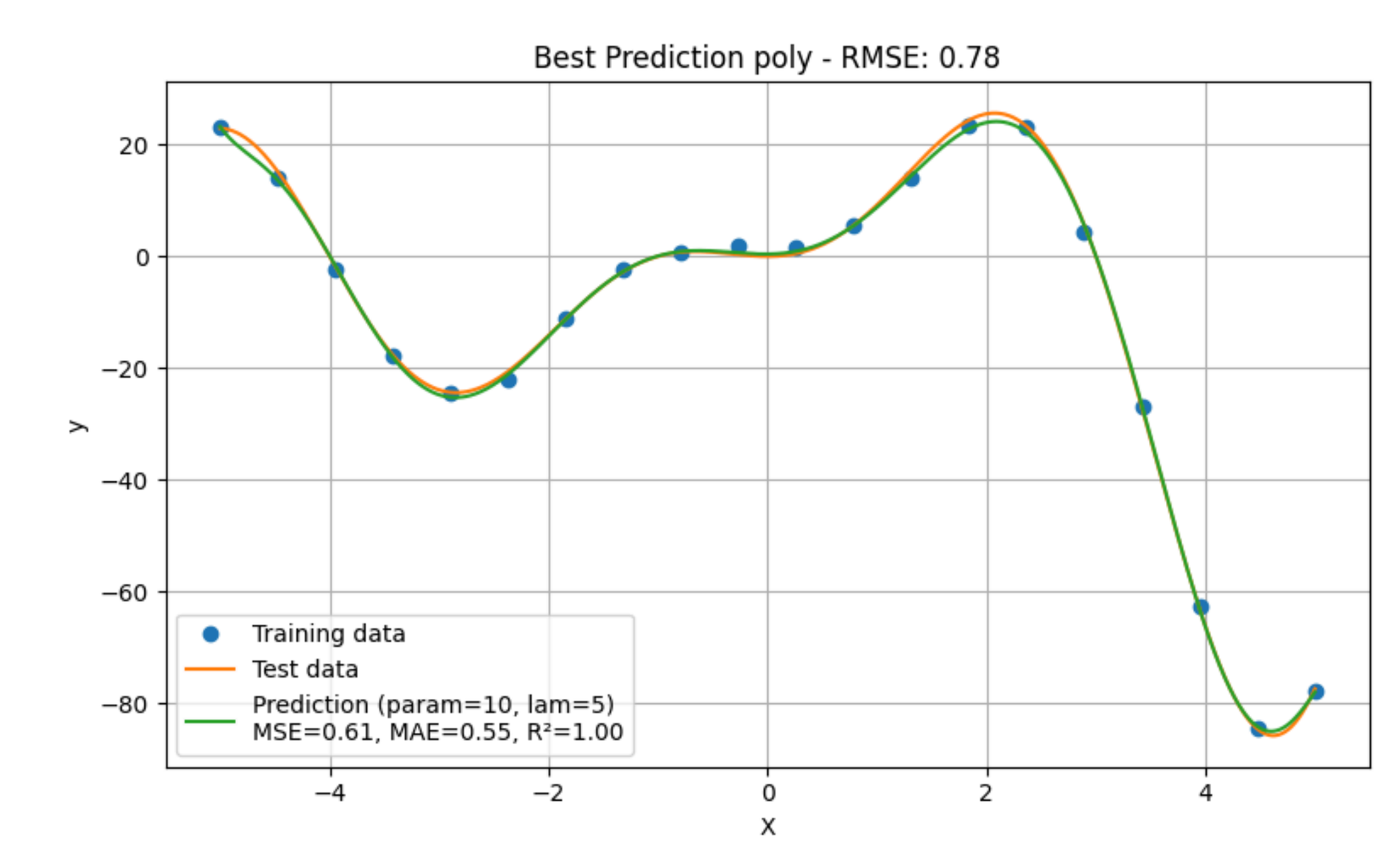

Figure 5: Plot for Kernel Ridge Rigression (Poly) with RMSE = 0.78

2.4 Key Findings

When non-linearity was introduced or present in the dataset, both kernel methods outperformed the linear approach, with the Gaussian kernel providing a slight edge over the polynomial kernel.

3. PCA vs. Kernel PCA

3.1 Objective and Methodology

- **Objective**: To compare the effectiveness of standard PCA against KPCA in terms of capturing the variance and revealing underlying structure in the data for linear SVM.
- **Methodology**:
  - Gets the dataset from `sklearn.datasets.make_circles` with 1000 samples and divides it into 20% training and 80% testing with 15% of them being noise. For consistency the sklearn seed is set to 0

3.2 PCA

A linear dimensionality reduction technique that projects data onto a subspace spanned by the principal components (directions of maximal variance).

- PCA Performance:
  - `Using sklearn.decomposition.PCA`, we perform dimensionality reduction on `X_train` and visualize the result. Although the dataset is correctly labeled, the projected shape reveals two concentric circles; one enclosed within the other. This circular structure highlights a key limitation: the data is not linearly separable in the reduced space, which significantly reduces the effectiveness of a linear SVM in classifying the points correctly as proven with the classification report on accuracy:

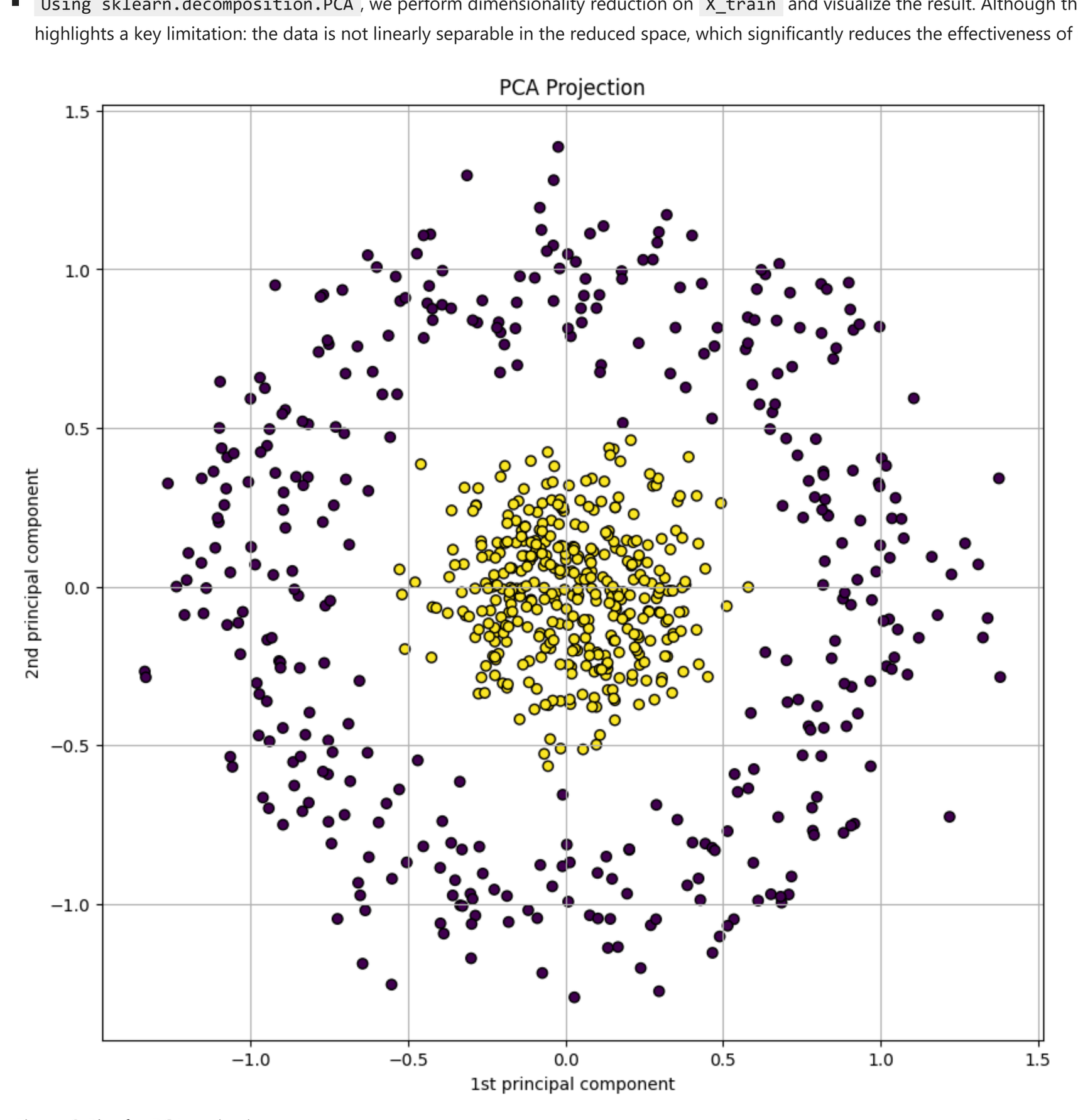

Figure 6: Plot for PCA projection

	Precision	Recall	F1-score	Support
Class 0	0.90	0.30	0.45	125
Class 1	0.58	0.97	0.72	125
Accuracy			0.63	250
Macro avg	0.74	0.63	0.59	250
Weighted avg	0.74	0.63	0.59	250

3.2 KPCA

A non-linear extension of PCA that first applies a kernel transformation. The Gaussian or polynomial kernel can help uncover non-linear manifolds hidden in the data.

- KPCA Performance:
  - Using `sklearn.decomposition.KernelPCA` with `gamma=5`, we perform dimensionality reduction on `X_train` and visualize the result. The projected data is well-separated and nearly linearly separable, which significantly improves the performance of a linear SVM, as confirmed by the classification report accuracy.

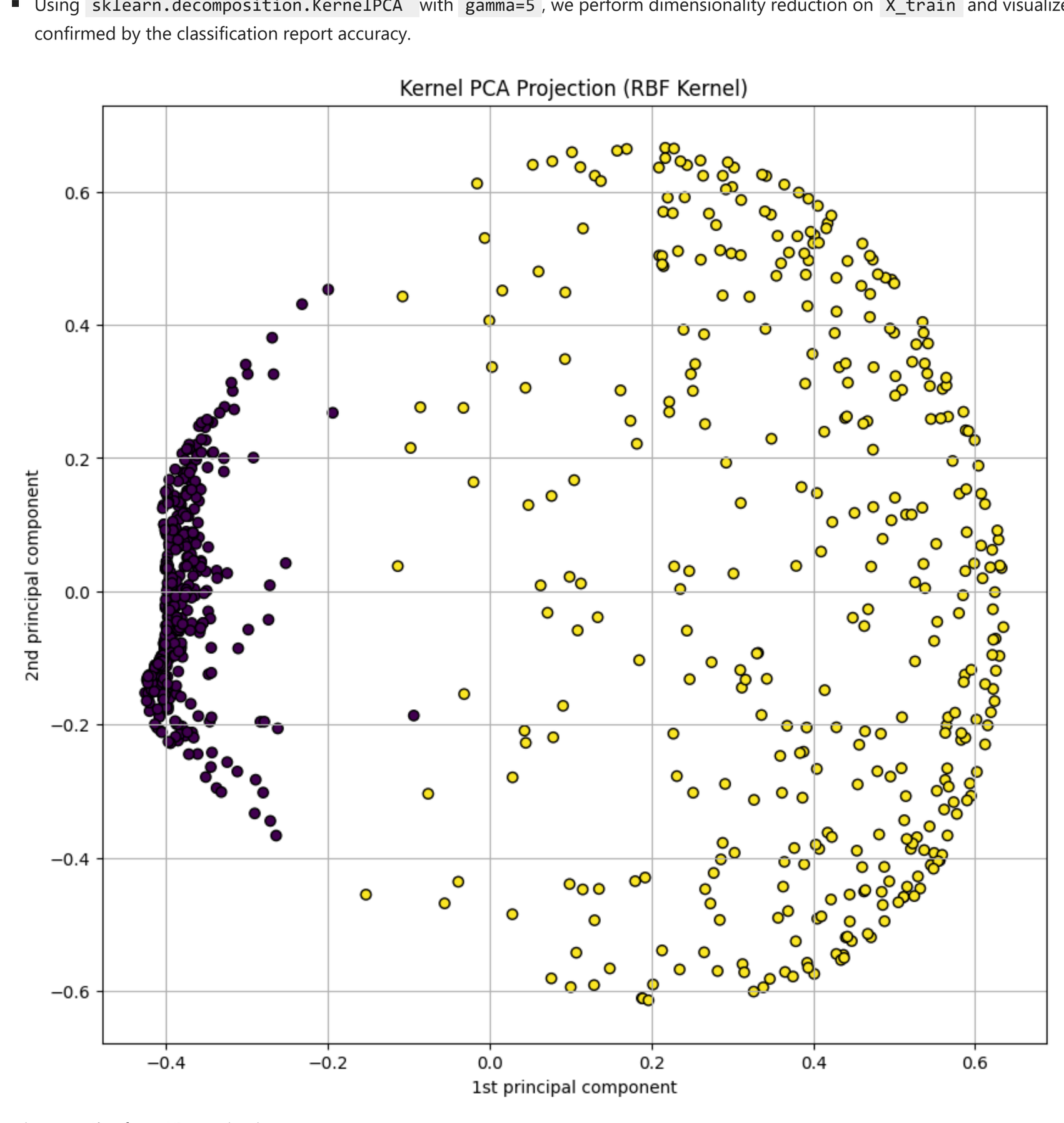

Figure 7: Plot for KPCA projection

	Precision	Recall	F1-score	Support
Class 0	0.98	1.00	0.99	125
Class 1	1.00	0.98	0.99	125
Accuracy			0.99	250
Macro avg	0.99	0.99	0.99	250
Weighted avg	0.99	0.99	0.99	250

3.3 Key Findings

In scenarios where the data had a non-linear structure, KPCA delivered more insightful lower-dimensional representations.

4. PCA vs. Kernel PCA Part 2

4.1 Objective and Methodology

- **Methodology**:
  - We generate a dataset using `sklearn.datasets.make_classification` with 1000 samples and 15% label noise. The data is split into 80% testing and 20% training, with `random_state=0` set for consistency.

4.2 PCA

- PCA Performance:
  - Using `sklearn.decomposition.PCA`, we perform dimensionality reduction on `X_train` and visualize the result. We observe that the dataset becomes almost linearly separable, as confirmed by the SVM classification report.

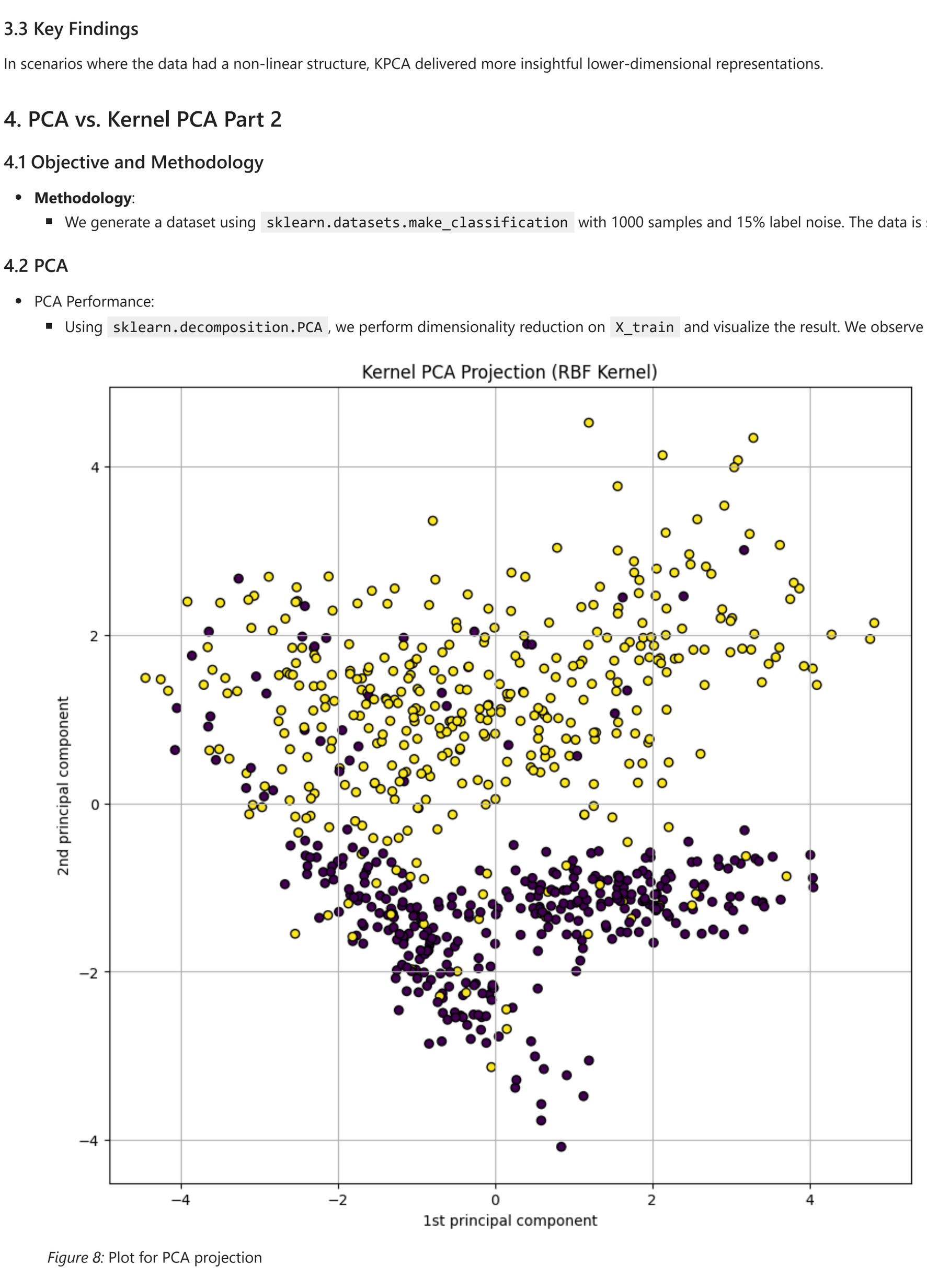

Figure 8: Plot for PCA projection

	Precision	Recall	F1-score	Support
Class 0	0.98	1.00	0.99	124
Class 1	1.00	0.98	0.99	126
Accuracy			0.99	250
Macro avg	0.99	0.99	0.99	250
Weighted avg	0.99	0.99	0.99	250

4.3 PCA

- PCA Performance:
  - Using `sklearn.decomposition.KernelPCA` with gamma=0.01, we perform dimensionality reduction on `X_train` and visualize the result. We observe that the dataset becomes nearly linearly separable. However, the classification accuracy does not improve compared to standard PCA; in fact, it slightly decreases, as confirmed by the SVM classification report.

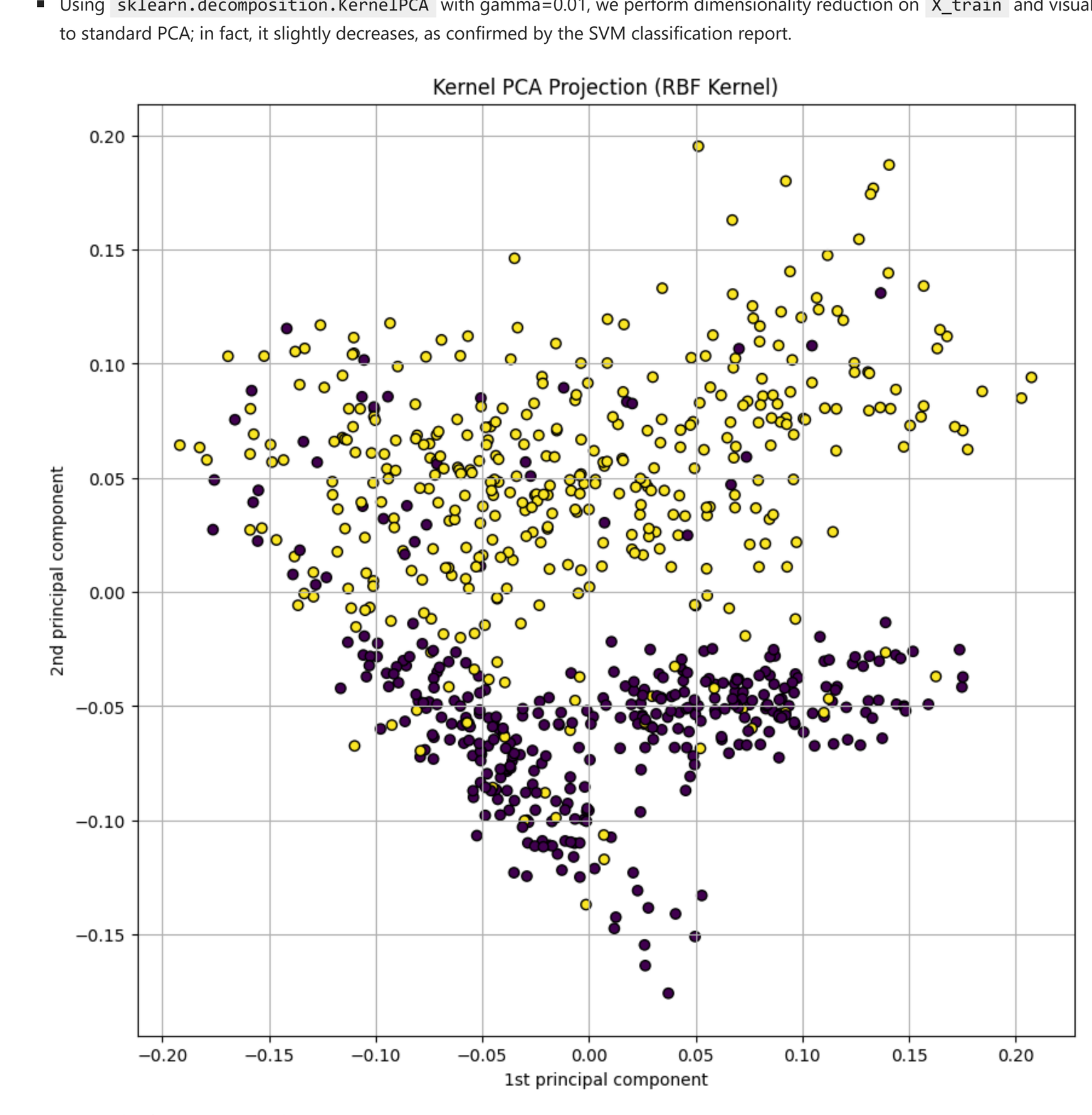

Figure 9: Plot for KPCA projection

	Precision	Recall	F1-score	Support
Class 0	0.84	0.89	0.86	124
Class 1	0.88	0.83	0.86	126
Accuracy			0.86	250
Macro avg	0.86	0.86	0.86	250
Weighted avg	0.86	0.86	0.86	250

4.4 Key Findings

These results highlight that while KPCA can be highly effective, its benefits are conditional on the data characteristics.

In datasets where the non-linear structure is weak or obscured by noise, traditional PCA's stability and simplicity can make it a more reliable choice.