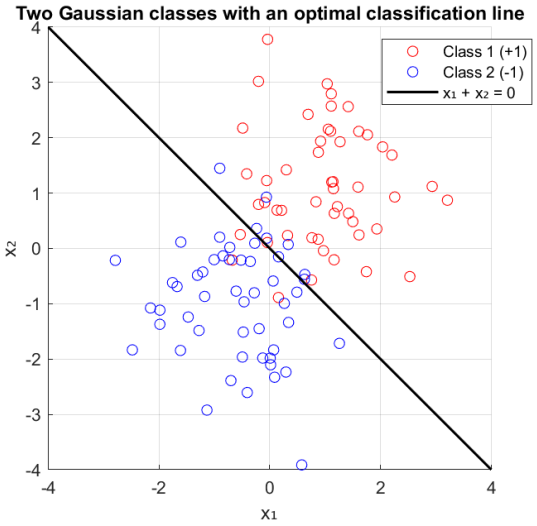
Bartosz Bogucki r1032166

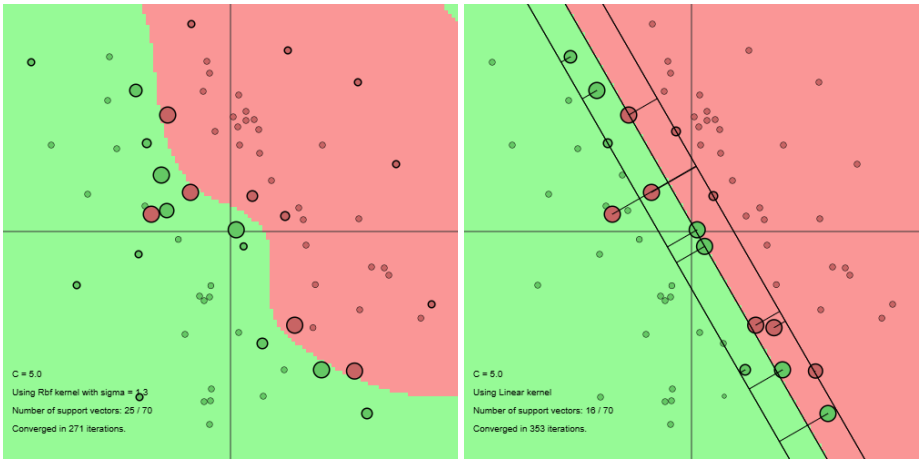
Exercise 1

1.1

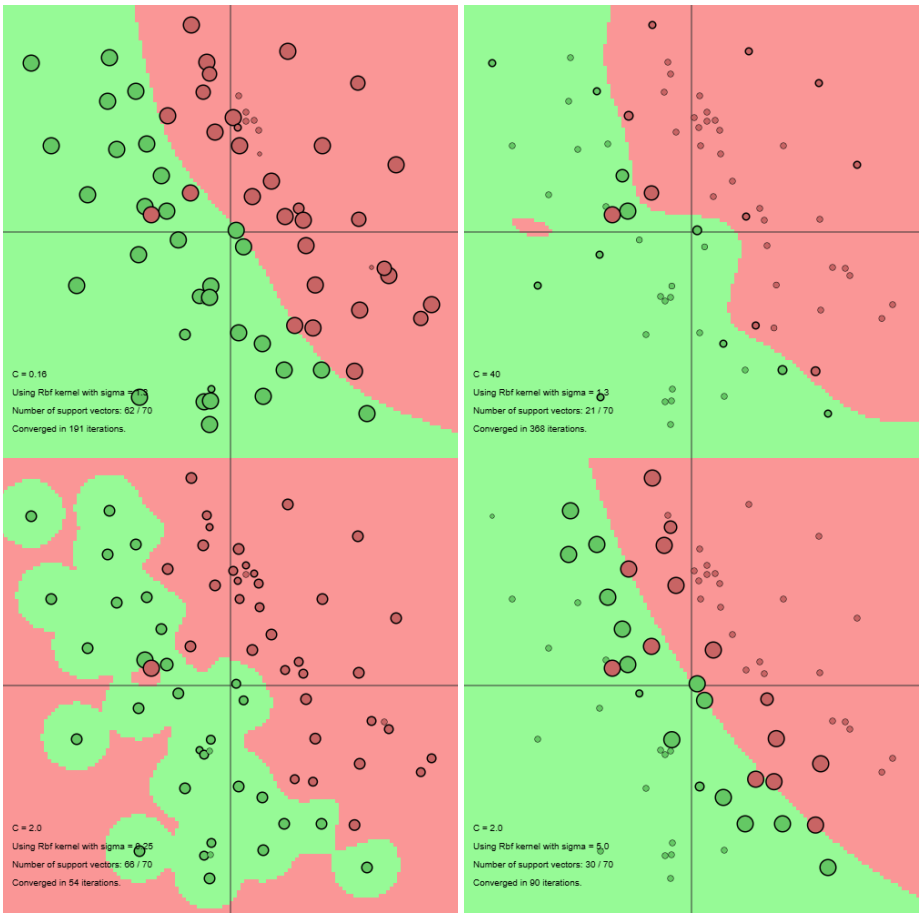


Given that both classes are of Gaussians with the same covariance matrices, the optimal separating line to classify the data is the one perpendicular to the line connecting the two class means and passing through their midpoint, which in this case is (0,0). Thus, the line is x1+x2=0. This line is optimal in the Bayes sense as it minimizes the probability of misclassification for two Gaussian distributions with equal covariance matrices. It corresponds to the maximum likelihood decision boundary.

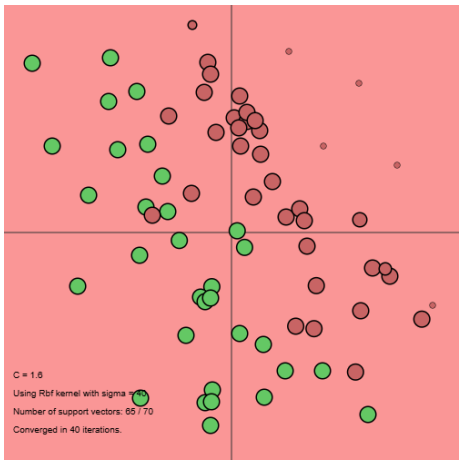
1.2



In the example above, the left plot uses an RBF kernel with c=5 and σ=1.3. Out of a total of 70 data points, 35 were identified as support vectors and are marked with bold circles. On the right, the linear kernel is used with c=5, where out of 70 points, 16 are support vectors. Note that a support vector is indicated by a bold-lined circumference and that its importance is proportional to the size. Support vectors are the data points that lie closest to the decision boundary, hyperplane, in a support vector machine. They are the most challenging points to classify and play a critical role in determining the optimal position of the hyperplane. The harder a data point is to classify correctly, the more important it tends to be as a support vector. Removing, moving or adding a support vector change the decision boundary. The decision function of SVM is fully specified by this typically small subset of samples of support vectors. The SVM algorithm relies on the support vectors to define the optimal hyperplane, aiming to maximize the margin of the distance between the hyperplane and the nearest data points from each class. In the case of RBF, the number of hidden units corresponds to the number of support vectors.



The parameter c is a regularization parameter that controls the trade-off between margin size and classification errors. A large c, as seen in the top-right plot, forces the model to classify all training examples correctly by heavily penalizing misclassifications. This results in a more complex and less smooth boundary and may lead to overfitting. In contrast, a small c, as in the top-left plot, allows some misclassifications, leading to a wider margin and a simpler, more generalized boundary, but often involves more support vectors with more evenly distributed influence. The role of the c parameter is the same regardless of the kernel (whether RBF or linear). The kernel parameter σ determines the influence range of each data point in the RBF kernel. There is no σ for the linear kernel. A small σ, as seen in the bottom-left plot, makes the boundary highly sensitive to nearby points, with nearly all points becoming support vectors. This creates tightly curved boundaries and increases the risk of overfitting. A large σ, as in the bottom-right plot above, causes the decision boundary to become smoother and more linear, reducing flexibility and potentially leading to underfitting if too large.



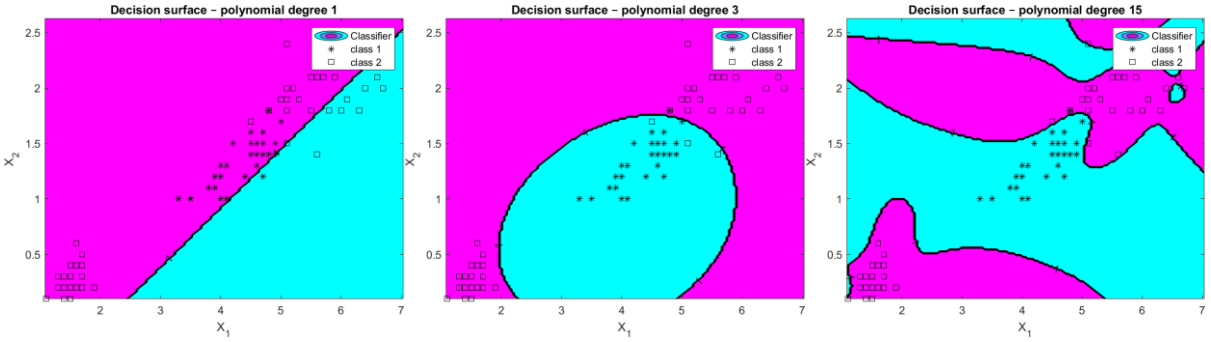
When σ is extremely high, the RBF kernel becomes too wide, causing the model to treat all data points as almost identical in terms of similarity, regardless of their actual distance. The decision boundary

becomes almost linear and unresponsive to the underlying class structure. Therefore, as we have more red points, the model predicts all the points to be in the majority class of the red class, leading to the misclassification of all the green points of the minority class.

1.3

1.3.1

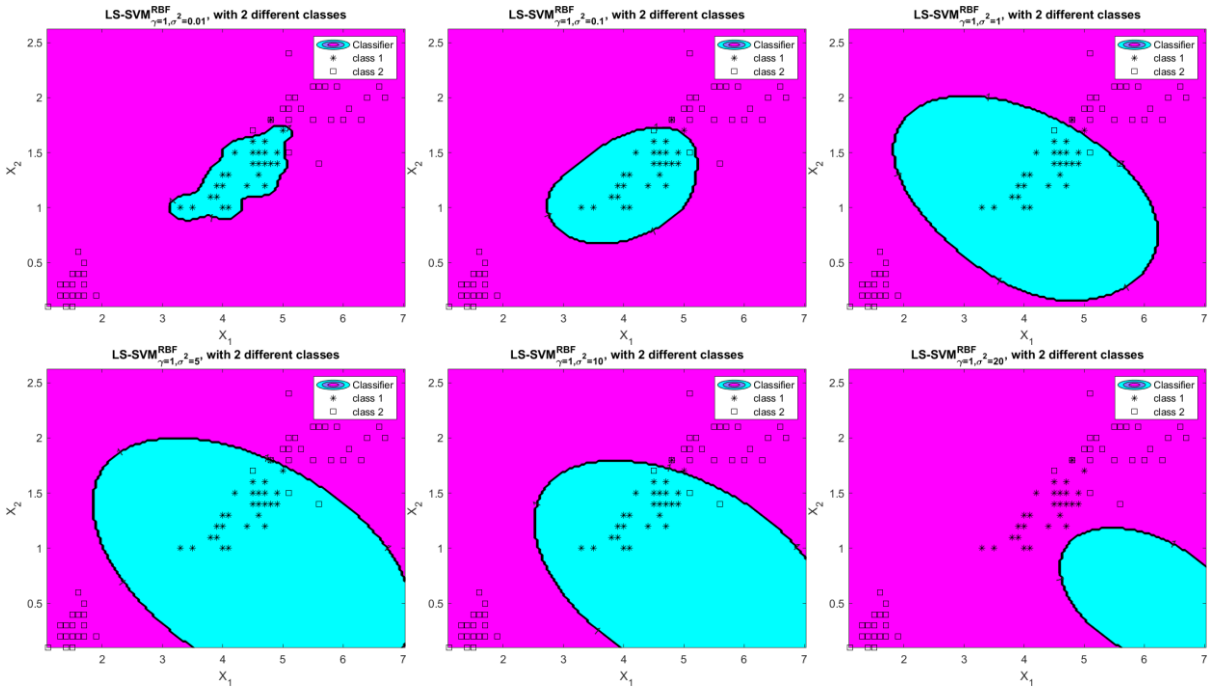
A polynomial kernel with degrees ranging from 1 to 15 is evaluated on both the training and test sets of the Iris dataset, using a fixed regularization parameter γ=1. Additionally, the decision surfaces obtained from the training process are visualized to observe the complexity of the decision boundaries:



Degree	Error count on train	Error rate on train	Error count on test	Error rate on test
1	42	42%	11	55%
2	6	6%	1	5%
3	4	4%	0	0%
4	4	4%	0	0%
5	3	3%	0	0%
6	4	4%	0	0%
7	4	4%	0	0%
8	3	3%	0	0%
9	3	3%	0	0%
10	3	3%	0	0%
11	2	2%	0	0%
12	2	2%	1	5%
13	1	1%	1	5%
14	1	1%	1	5%
15	1	1%	1	5%

Degree 1 corresponds to a linear kernel and performs poorly on both the training and test sets, as it fails to capture the nonlinear boundaries present in the data. Increasing the degree to 2 yields a substantial improvement, and degree 3 achieves perfect classification on the test set. As the degree increases further, training error continues to drop slightly, indicating increased fit to the training data thanks to increased model complexity. However, test performance begins to degrade from around degree 12 onward, suggesting the onset of overfitting. Therefore, the optimal degree lies in the range of 3 to 5 - high enough to capture the nonlinearity but not so high as to compromise generalization.

The RBF kernel is also evaluated with varying kernel bandwidths σ^2, using a fixed γ=1. Decision surfaces are again plotted to visualize the boundary complexity.



σ²	Error count on test	Error rate on test
0.01	2	10%
0.1	0	0%
1	0	0%
5	0	0%
10	0	0%
20	10	50%

A very small σ² value of 0.01 leads to highly localized decision boundaries around individual training points, which causes overfitting and errors on unseen data of test set. In contrast, extremely large σ² values of 20 overly smooth the decision boundary, resulting in underfitting and a high test error. The best performance is observed for σ² values between 0.1 and 10, where the model effectively balances bias and variance and achieves 0% test error.

To evaluate the influence of γ, the regularization parameter, performance is assessed using a reasonable fixed σ² of 0.01 and varying γ values:.

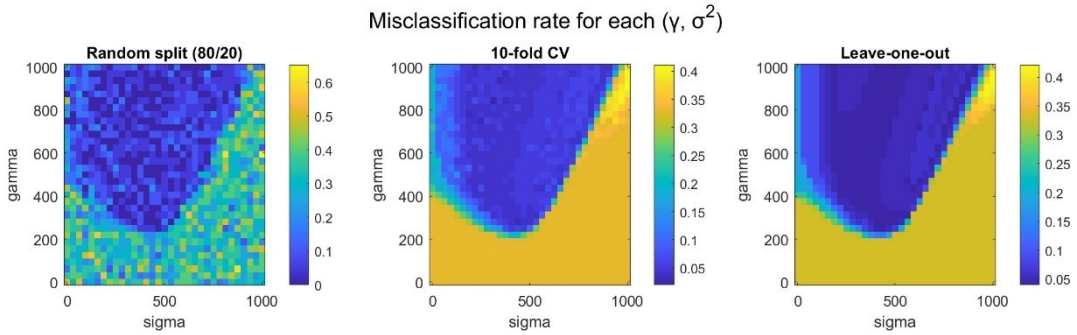
γ	Error count on test	Error rate on test
0.01	10	50%
0.1	2	10%
1	0	0%
5	0	0%
10	0	0%
1000	1	5%

Very small γ values, e.g. 0.01 or 0.1, lead to strong regularisation, which prevents the model from fitting well to the data, leading to underfitting and misclassifications also on the test set. Very large γ values, e.g. 1000, can lead to overfitting or numerical instability, increasing the test error. The optimal performance is achieved for γ values between 1 and 10, where the model generalizes well and is not underfitting.

The script of SampleScript_iris.m shows similar behavior, using very comparable ranges of parameters as above.

1.3.2

Each plot below shows the misclassification rate on validation sets over a range of values for γ (vertical axis) and σ² (horizontal axis), using different strategies to split the training data into training and validation subsets. Dark blue areas represent low error rates, while yellow indicates higher misclassification.



The comparison shows significant differences in the structure of the error surfaces between random split and cross-validation methods, 10-fold and leave-one-out. Random split produces a noisier and more irregular misclassification map, with error rates fluctuating over γ and σ² with some inconsistencies. This randomness is due to the arbitrary way in which the data is split in each run, so that only one validation sample is used, which is different each time. Using only one validation makes the identification of optimal hyperparameters less stable and less reliable. However, some low error regions are visible and comparable to those of k-fold validation methods. The 10-fold cross-validation produces a much more stable and consistent error surface. The transitions between parameters are more consistent, indicating a more reliable result of model tuning. This method uses 10 different validation sets, improving the certainty of the final results. The leave-one-out method, which is a special case of k-fold cross-validation where k is the number of training samples, further enhances this reliability. It produces the smoothest and most sharply defined low error region. Although computationally expensive, it provides the most accurate and stable results. Based on the results, a distinct region of low misclassification rates is visible for around moderate values of γ and σ² with the best performing region being around σ² 250-400 and γ around 400-700. This means that these parameter values provide a good balance between model complexity and generalisation, achieving the smallest misclassification errors during validation.

Cross-validation is preferred to simple random split validation because of its lower variance, greater consistency and full use of available training data. While random split uses only part of the training data, cross-validation uses the entire training data by rotating the training and validation sets across all folds. Among cross-validation strategies, 10-fold or 5-fold is the standard choice, offering a solid bias-variance trade-off. A smaller k (such as 5) is faster but may result in slightly noisier estimates, while larger k (e.g. leave-one-out) provides highly accurate estimates at a higher computational cost. Depending on the number of observations, we can use leave-one-out for very small datasets, and as a rule of thumb use k of 10 for a moderate dataset and k of 5 for a very large dataset.

1.3.3

The tuning of the parameters is conducted in two steps. First, global optimization technique of Coupled Simulated Annealing (CSA) determines suitable initial parameters. Second, these parameters are then given to a second optimization procedure (simplex or gridsearch) to perform a fine-tuning step. Therefore, the obtained hyperparameters can differ significantly across different runs for both the simplex method and gridsearch.

The simplex method searches for a local minimum of the cost function starting from the given initial point. The local minimum is found using the Nelder-Mead algorithm, a heuristic search technique that does not require gradient information. If the initial starting point of CSA lies in a relatively good region, simplex typically converges quickly by making small local moves and may remain close to the starting point without exploring the broader parameter space. Thus, in the simplex method, the optimization tends to stay close to the initial point in our case of this exercise.

The brute-force gridsearch method performs an exhaustive search over a limited range around the initial starting point. The algorithm systematically tries all possible parameter combinations within a fixed grid, evaluates each one independently, and selects the one with the best result.

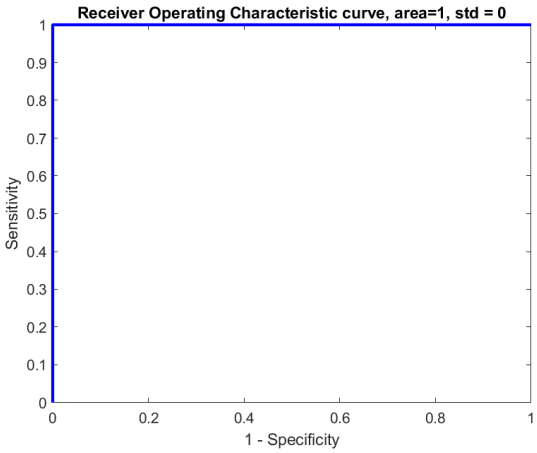
Over 20 runs, the variance in the selected hyperparameters and cost was high for both methods due to different initial points determined by the stochastic nature of Coupled Simulated Annealing. Gridsearch generally requires longer computational time and more iterations because it exhaustively evaluates a larger number of combinations within the range. However, this comprehensive exploration often leads to a very slightly lower cost compared to simplex. On the other hand, the simplex method, by often staying near the initial point, is faster but may achieve slightly higher cost values. The following table summarizes the average cost with its standard deviation (in brackets) and the average computational time measured over 20 runs:

Method	**Cost**	**Computional speed**
Simplex	3.55 % (± 0.76 %)	0.47 s
Bruteforce	3.40 % (± 0.75 %)	0.82 s

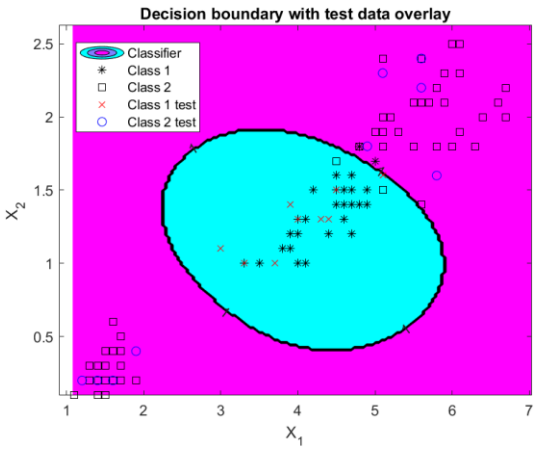
1.3.4

The ROC curve is computed on the test set rather than on the training set because the goal is to evaluate the performance of the model on unseen data that was not used during training. Evaluating on the training set would primarily measure how well the model memorized the data, often leading to overly optimistic results and potentially hiding issues like overfitting. Testing on the test set provides a measure of the model's ability to generalize, which is critical in practice, as we aim to apply models to new, unknown data.

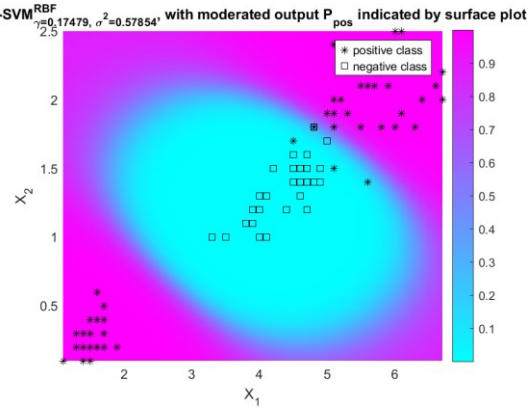
The ROC curve generated on the test set of the iris dataset, using the LS-SVM model trained with tuned hyperparameters (γ = 0.17479 and σ^2 = 0.57854), is shown below:



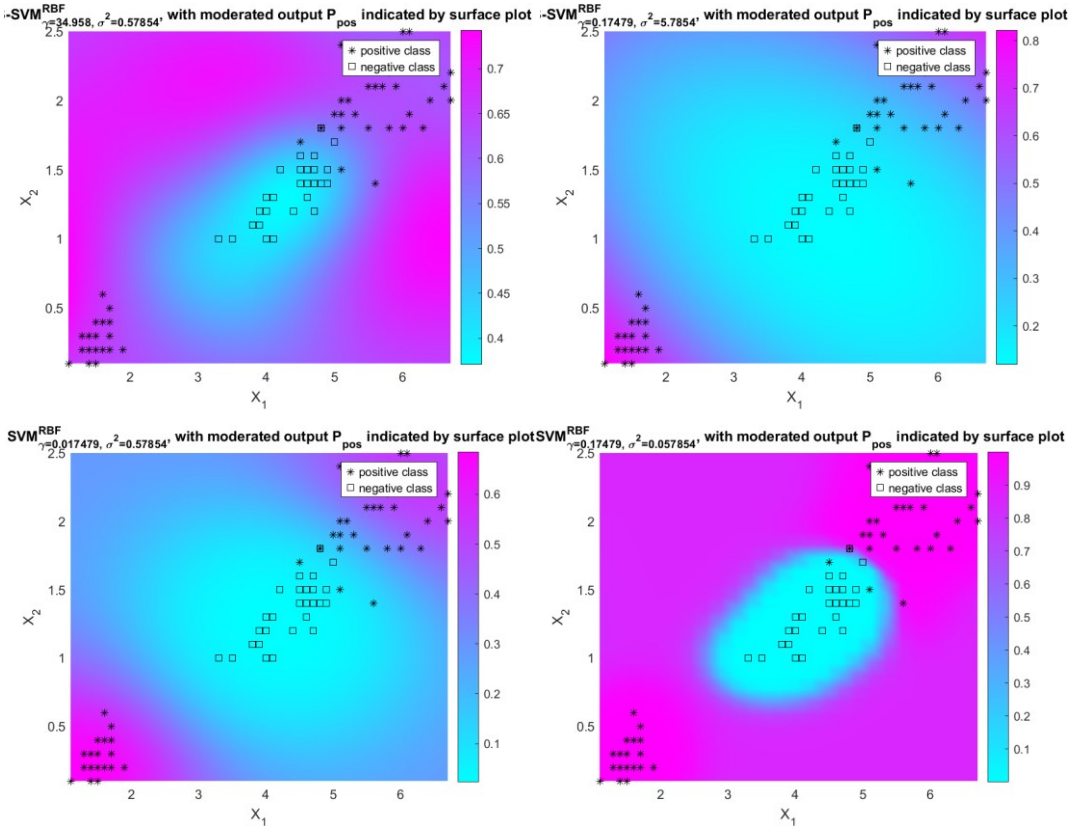
The area under the ROC curve (AUC) is equal to 1, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives), indicating a perfect classifier. This result shows that the LS-SVM classifier with the tuned hyperparameters achieves excellent performance on the test set, correctly classifying all the test instances of our binary classification problem of iris dataset to its true labels without any errors:



1.3.5

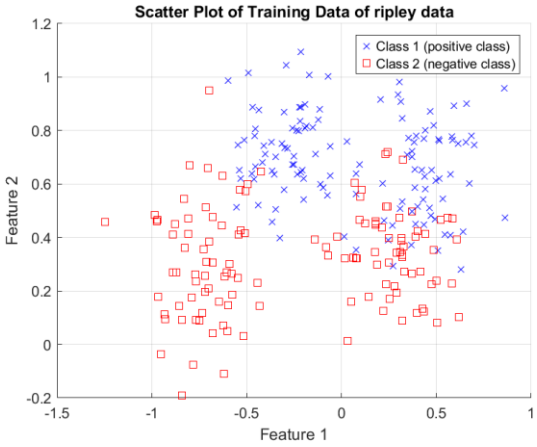


The method calculates the posterior class probabilities of a binary classifier using Bayesian inference. The colors of the plot represent the posterior probability that a data point belongs to the positive class. Values close to 0 of blue colour correspond to high confidence that the point belongs to the negative class (Class 1). Values close to 1 of purple colour correspond to high confidence that the point belongs to the positive class (Class 2). It can be seen that the classifier of γ = 0.17479 and σ² = 0.57854 performs well by confidently assigning high probabilities to true positive class points and low probabilities to true negative class points. Additionally, the decision boundaries are smooth, indicating that the model generalizes well without overfitting to very specific training samples.



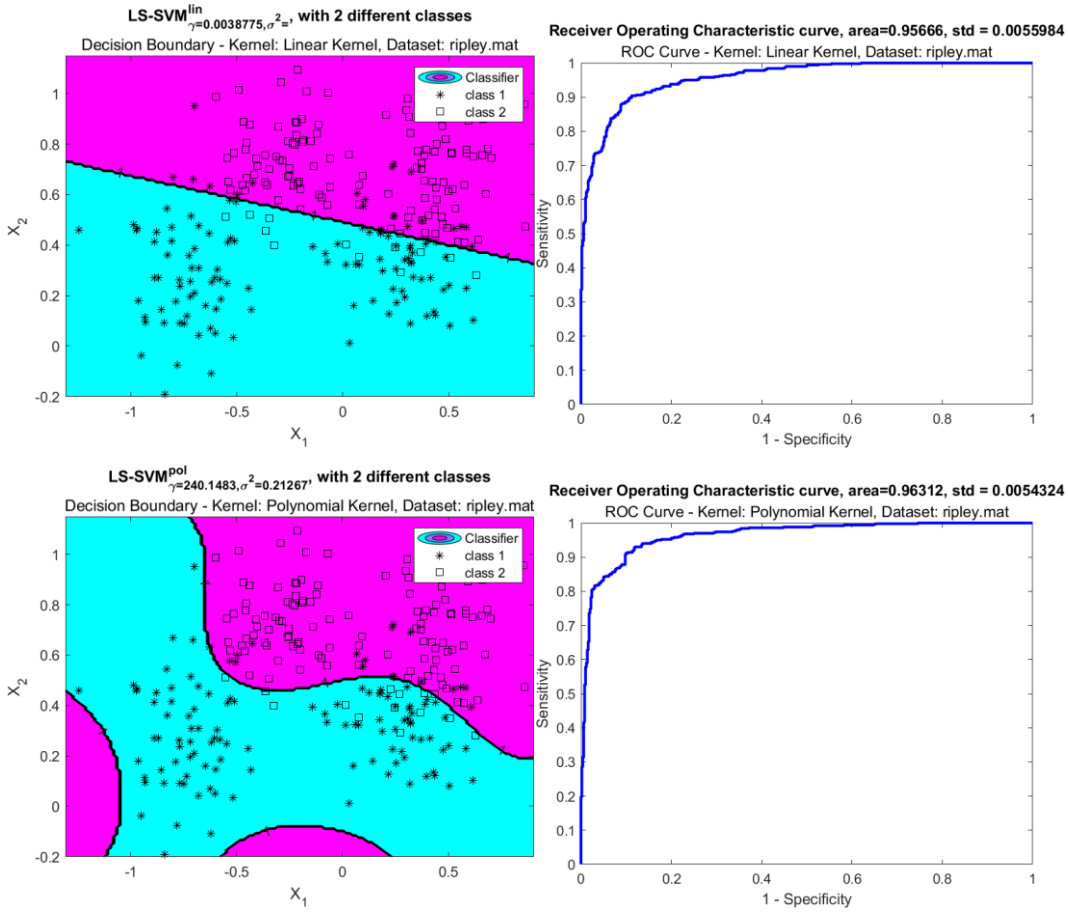
Changing the value of γ and σ² significantly influences the classifier's behavior. Lower σ² values create sharper decision boundaries with the probability surface highly focused around the training points, increasing the risk of overfitting. Higher γ values can also cause the model to overemphasize fitting the training data, which may reduce the certainty of probability estimates across broader regions. Higher σ² and lower γ values produce smoother probability transitions, leading to underfitting and sometimes misclassifications on training data. Therefore, proper tuning of both parameters is crucial to balance the trade-off between underfitting and overfitting, and to achieve good performance on both training and test data.

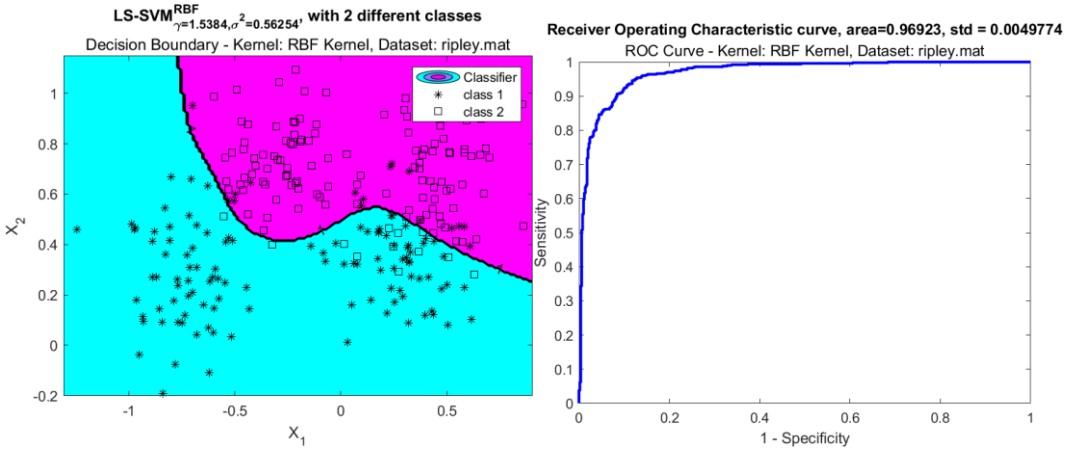
2



The Ripley dataset consists of two features and binary class labels, which are visualized in the feature space. From the scatter plot, we can observe overlapping between the two classes in several regions, indicating potential challenges in clear separation, especially they are not linearly separable. The dataset is balanced, with 250 training samples and 1000 test samples, evenly distributed between the two classes. Given the visible non-linear structure of the data, a linear classifier is expected to perform the worst, while non-linear kernels such as polynomial and radial basis function (RBF) kernels are expected to be more effective at capturing the complex decision boundaries.

Decision boundaries derived on training data and Receiver Operating Characteristic (ROC) curves on test data for different kernels applied to the Ripley dataset are shown below:





The decision boundary of linear kernel cannot fully separate the two classes due to the overlap. The ROC AUC on test data is 95.67%, showing reasonable but limited performance. For polynomial kernel the model fits the decision boundary more flexibly, achieving an AUC of 96.31% on test set. However, based on the decision boundary, it appears the model has a higher tendency to overfit. The highest AUC of 96.92% on test data is achieved by the model of RBF kernel. It balances flexibility and generalization, handling non-linear relations effectively.

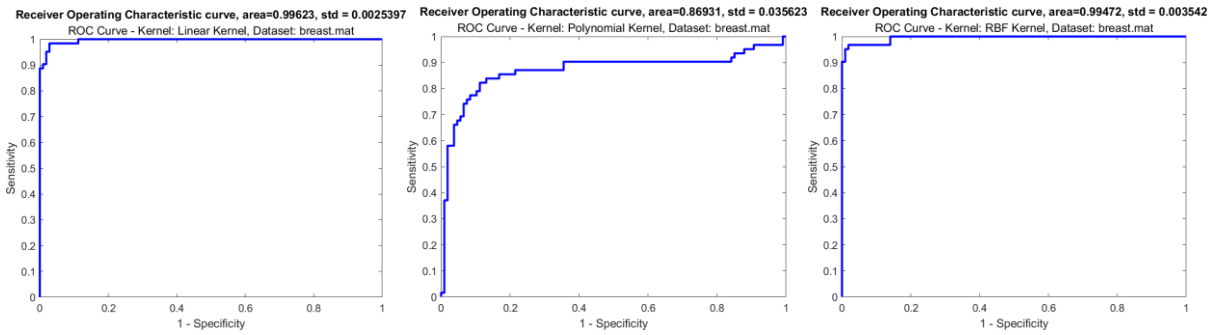
Therefore, the best model is achieved using the RBF kernel, achieving ROC AUC of 96.92%. It can also be seen that decision boundaries handled the non-linear relations of the data well. However, the other models with different kernels performed only slightly worse, with 96.31% for the polynomial kernel and 95.67% for the linear kernel. Taking into account the decision boundaries, the polynomial kernel has a slight potential for overfitting, and the linear kernel may underfit. The results are satisfactory, as AUC is high and above 95% for every kernel, with the best performance from the RBF kernel and not much space left for further improvement.

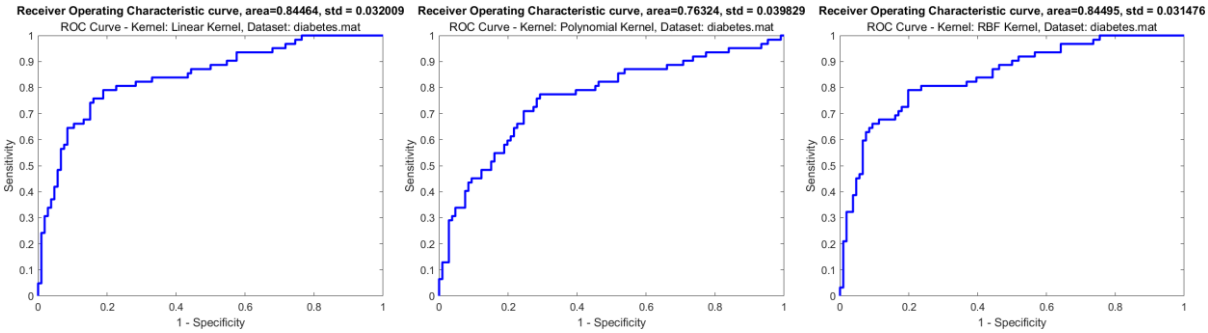
The data of breast and diabetes has more features than 2, so visualizing it would require dimensionality reduction, e.g., PCA analysis. However, here our main objective is to predict the target label and test LS-SVM classification, so we focus on this. Still, it is important to mention that:

- Breast training data has 30 features of 400 entries, and the test set has 169 entries. The dataset is rather balanced, with 150 positive and 250 negative labels in training, and 62 positive and 107 negative in the test set.
- Diabetes training data has 8 features of 300 entries, and the test set has 168 entries. It is also balanced, with 95 positive and 205 negative labels in training, and 62 positive and 106 negative in the test set.

Since the number of features is relatively high while the number of training samples is moderate, the polynomial kernel risks overfitting by modeling complex and potentially spurious relationships. Meanwhile, linear and RBF kernels are expected to perform better, as they generalize more effectively under these conditions. Note that the data is normalized as done by default by LS-SVMlab during preprocessing.

For both datasets of breast and diabetes, ROC curves were computed on test data to evaluate which model performs best:





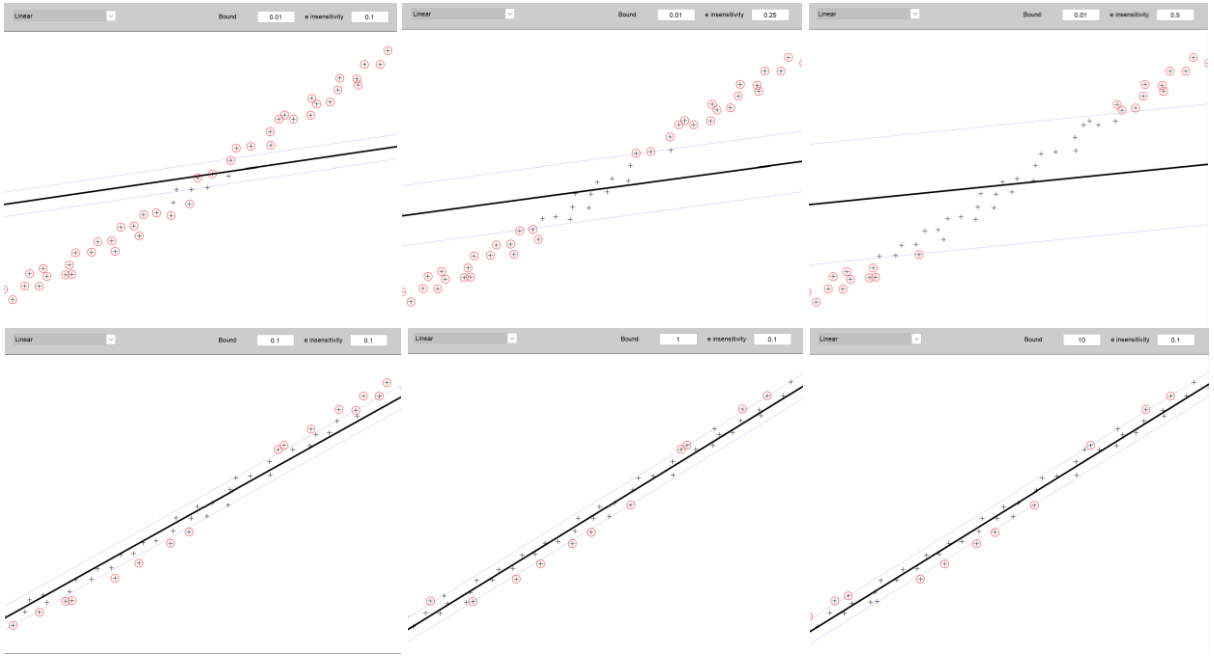
The polynomial kernel is the worst among kernels for both datasets, potentially overfitting the data and not performing well on test data. Linear and RBF kernels achieve very comparable results, with ROC AUC of 99.62% for linear kernel as the highest for breast data, and ROC AUC of 84.5% for RBF kernel as the highest for diabetes data. The results of linear and RBF kernels are very satisfactory for the breast dataset, as both are above 99%.

For the diabetes dataset, although the performance is reasonable, the results suggest that there is room for further improvement. The RBF kernel achieved an AUC of around 84.5% on test set, which is a solid performance considering the complexity and inherent noise of the diabetes dataset. However, applying feature engineering or dimensionality reduction prior to training the LS-SVM may improve the results. Other loss functions than squared and weighting of data points can also be further tested for better robustness. In addition, ensemble methods can be considered.

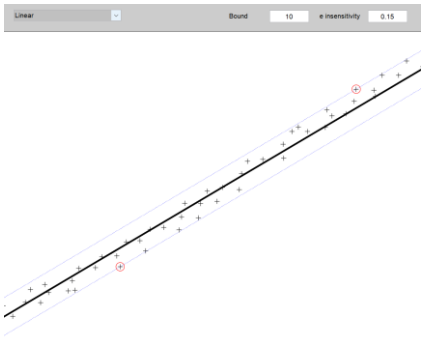
Exercise 2

1.1

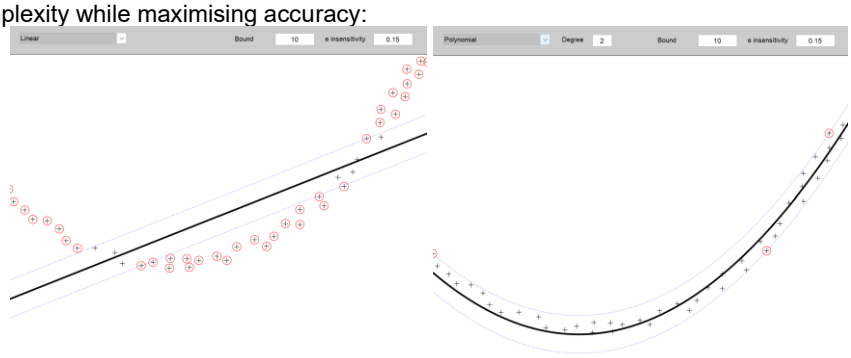
In the first experiment, we generated roughly twenty noisy points clustered around a straight-line trend. As expected, the linear kernel outperforms any nonlinear alternative: it matches the data's underlying linear structure without introducing unnecessary complexity. When tuning the ε-insensitive tube and the Bound parameter, we observed that widening ε simply admits more points as inliers - shrinking the set of support vectors and smoothing the fit - while tightening ε draws more points into the loss and makes the slope more sensitive to noise. The bound parameter, denoted also as regularization parameter C, measure deviations beyond the ε-tube. Very small C tolerates large slack violations and collapses the fit toward a horizontal line, whereas large C heavily penalizes deviations, pulling the regression line closer to outliers and steepening its slope.



In our tests, ε = 0.15 and C = 10 achieve the best balance: the tube is wide enough to ignore minor noise yet narrow enough to capture the true trend, and the penalty was strong enough to enforce that trend without overfitting.



For the second, more challenging data set, we generate about twenty points from a noisy quadratic function. Here, the linear kernel fails completely, leaving large systematic errors as it is unable to model non-linearities. A second degree polynomial kernel provides the ideal fit: it naturally models the underlying parabola. Although an RBF kernel could also capture the non-linearity, the degree 2 polynomial exactly matches the true generating process of the data, minimising complexity while maximising accuracy:



SVM regression differs from classical least-squares fitting in multiple ways, given the formula:

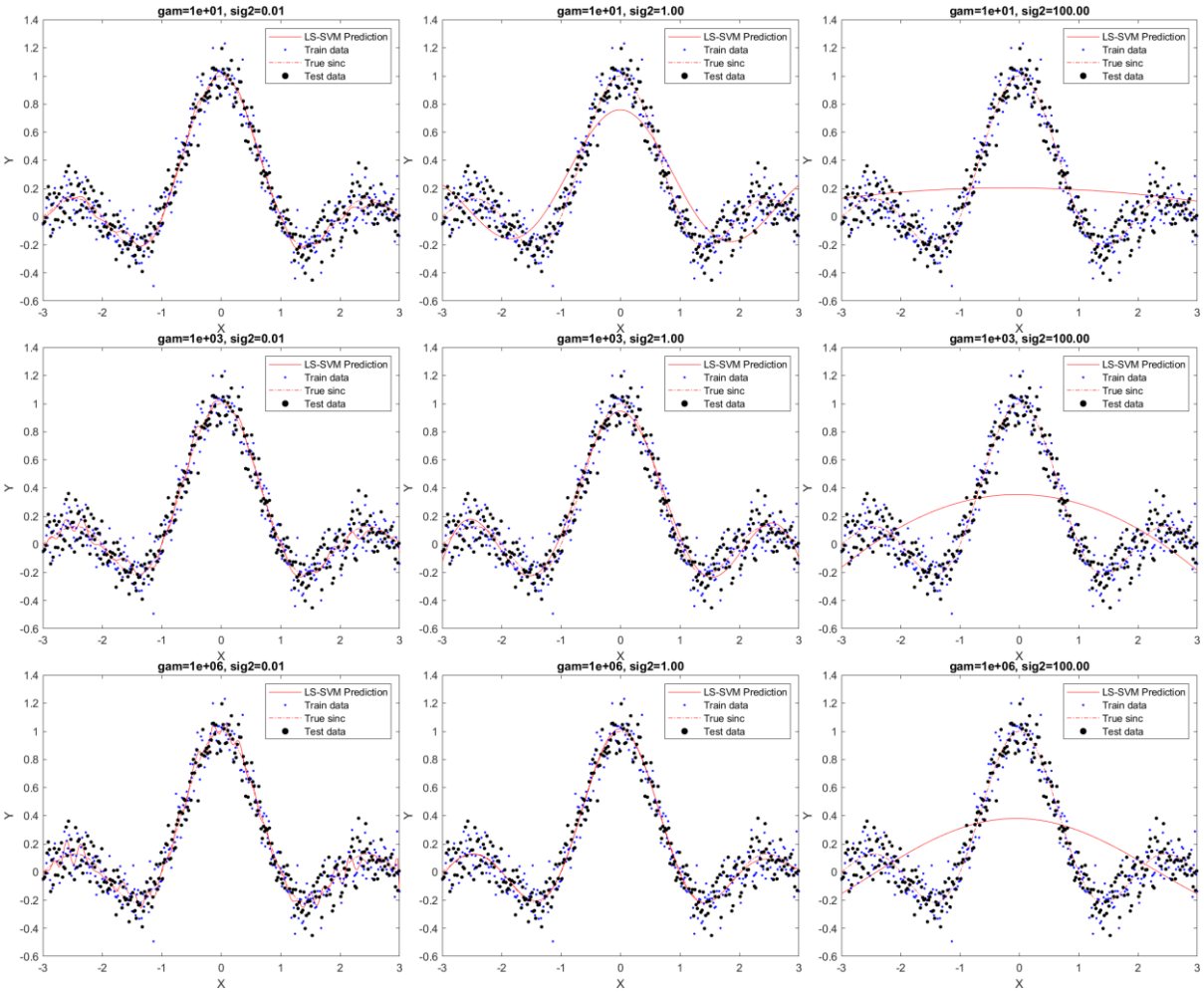
$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2} w^T w + c \sum_{i=1}^{N} (\xi_i + \xi_i^*)$$

First, its ε-insensitive parameter ignores errors smaller than ε, producing a tube within which deviations incur zero penalty. It can therefore control sensitivity to noise and outliers. Second, the constrained QP formulation with slack variables yields a sparse solution as only points outside the ε-tube become support vectors with nonzero dual coefficients. Third, this approach allows for dual representation and the kernel trick, which allows nonlinear regression without explicitly mapping inputs to high-dimensional feature spaces - features that standard least squares cannot provide.

1.2.1

Below is a concise summary of our experiments on the noisy sinc-function regression with RBF kernel:

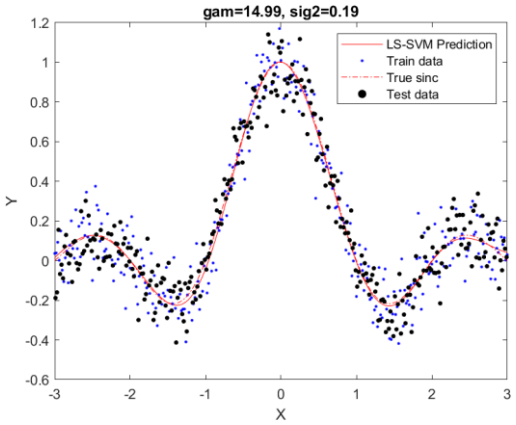
gam	sig2	MSE on test set
10	0.01	0.0106
10	1	0.0332
10	100	0.1297
10^3	0.01	0.0108
10^3	1	0.0114
10^3	100	0.1109
10^6	0.01	0.0114
10^6	1	**0.0099**
10^6	100	0.1054



Very low σ² of 0.01 produces a very narrow RBF, which overfits the noise and gives poor generalisation. Very high σ² of 100 oversmooths the function and underfits the underlying function, as the MSE is above 0.11 for any γ. In addition, γ controls the slackness penalty relative to the width of the margin: a very large γ forces the model to stick tightly to the training points risking overfitting, whereas a very small γ produces an overly flat function that underfits. A moderate σ² of 1 with a γ of 10^6 achieves the lowest MSE on the test set of 0.0099, which is the best performance on our fixed grid. Although with an RBF kernel and σ² of 1, the results with γ of 10^6 can be considered satisfactory, as this was an arbitrary grid, this may be not a true optimum. Therefore, it is essential to use an optimisation method to find the hyperparameters that provide optima.

Automated tuning (simplex or grid) efficiently smooths out the arbitrary discreteness of a manual grid. Three iterations each of the simplex and grid search tuning methods are run using leave-one-out validation under the RBF kernel. All runs converged to roughly the same performance of MSE on the test set of 0.0098, but chose slightly different hyperparameters, but very close to each other - σ^2 around 0.2 and γ between 10-50. We plot one of the chosen models.

Method	gamma	Sig2	MSE on test set
Simplex – iteration 1	17.6699	0.195398	0.0098
Simplex – iteration 2	12.3008	0.182788	0.0098
Simplex – iteration 3	44.014	0.219948	0.0098
Gridsearch – iteration 1	18.3824	0.200828	0.0098
Gridsearch – iteration 2	44.6014	0.218156	0.0098
Gridsearch – iteration 3	14.9898	0.193429	0.0098



1.2.2

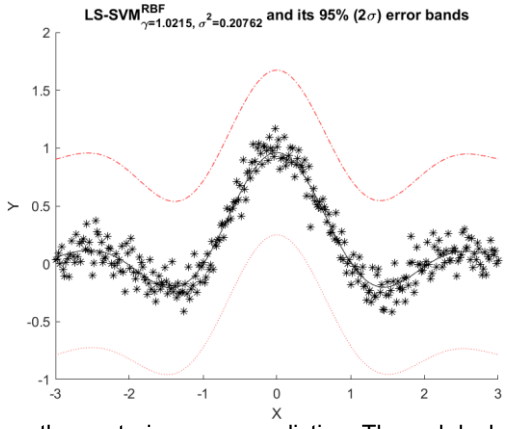
In a previous setting in 1.2.1, a validation set must be withheld and these points are never used during training, and then an external grid or simplex search over hyperparameters is performed to minimise the withheld validation error. The Bayesian framework overcome this as an alternative by treating γ and σ^2 as random variables and performing hierarchical inference, so that making full use of the dataset:

$$p(w,b|D,\mu,\zeta,H_\sigma) = \frac{p(D|w,b,\mu,\zeta,H_\sigma)}{p(D|\,\mu,\zeta,H_\sigma)} * p(w,b|\mu,\zeta,H_\sigma)$$

$$p(\mu,\zeta|D,\ H_\sigma) = \frac{p(D|\,\mu,\zeta,H_\sigma)}{p(D|\,H_\sigma)} * p(\mu,\zeta|\,H_\sigma)$$

$$p(H_\sigma|D) = \frac{p(D|\,H_\sigma)}{p(D)} * p(H_\sigma)$$

The resulting model is of γ=1.0215 and σ^2=0.20762:

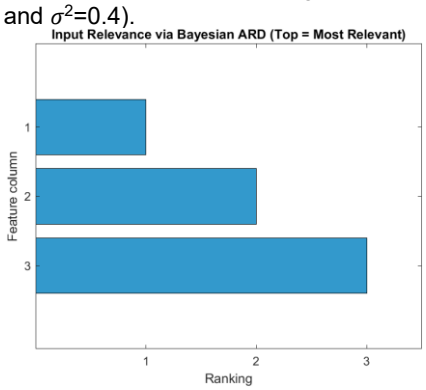


In the plot, the solid black line traces the posterior mean prediction. The red dashed lines curves around it mark the ±2 σ bounds of the posterior predictive distribution, giving the 95% credible interval - they quantify uncertainty by marking two standard deviations around the posterior mean. Because the Bayesian hierarchy infers γ and σ^2 from all available data, there is no need for a separate validation set, and the model automatically provides honest error bars on its predictions.

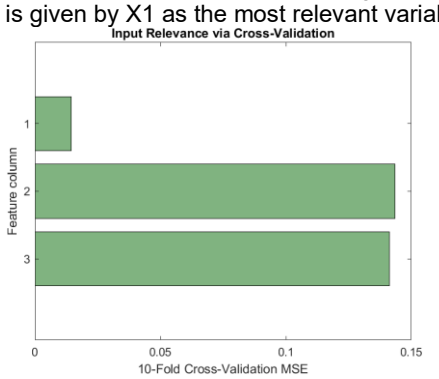
1.3

The Bayesian framework can also be used to select the most relevant inputs by Automatic Relevance Determination (ARD). In a backward variable selection, the third level of inference of the Bayesian framework is used to infer the most relevant inputs of the problem. It is applied to a three-dimensional input selection task constructed in such a

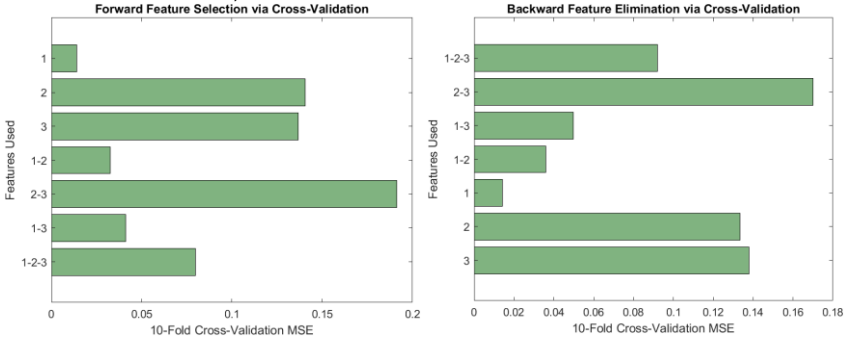
way that Y depends only on X1 and not on X2, X3, which are all generated random variables (the model selected is the one with RBF kernel and γ=10 and σ^2=0.4).



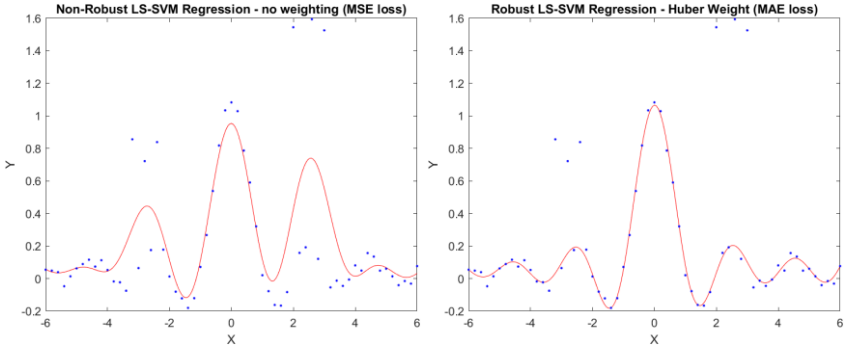
The resulting output is a ranking of the feature and a final selected subset. In our case, the ARD ranking is [1;2;3], indicating that the feature variable X1 is the most relevant, as indicated by the top bar. Furthermore, the model considering only feature X1 is selected, pruning the remaining X2, X3 as expected. An analogous result without Bayes can be obtained by using 10-fold cross-validation to evaluate single feature models and calculating the MSE for cross-validation. The lowest MSE is given by X1 as the most relevant variable.



Alternatively, we can add new features sequentially (forward selection), or start with all three and drop one at a time (backward elimination), each time evaluating the cross-validation MSE. Plotting these MSEs - first for singletons, then for pairs, then for the full trio - shows that the model with the lowest cross-validation error is the one that considers only X1 as a feature variable, and so should be the one selected.



1.4

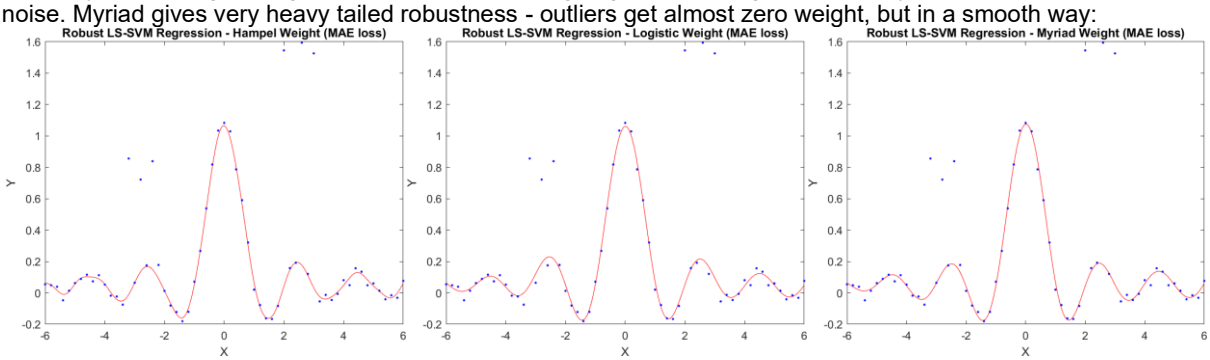


Non-robust LS-SVM (using MSE loss) produces the red curve in the left plot. It can be seen that extreme outliers pull the fit upwards, especially the three high points around -3 and 2.5. These outliers force the RBF network to bend away from the sinc shape in order to reduce their squared errors. In contrast, the Huber-weighted robust LS-SVM (using the MAE loss) is resistant to the same outliers. In the right plot, the central peak and side lobes of the sinc are recovered almost exactly.

MSE penalises large residuals quadratically, so outliers impose an excessive cost and force the model to distort itself. By switching to MAE (absolute error), the loss of each point grows linearly, preventing a single extreme residual from dominating the objective. This is why MAE is the preferred loss for the robust LS-SVM.
In addition to Huber, there are other weighting schemes:

	Huber	Hampel	Logistic	Myriad
Weight function W(r)	$\begin{cases} 1 \; if \; \lvert r\rvert < \beta \\ \frac{\beta}{r} \; if \; \lvert r\rvert > \beta \end{cases}$	$\begin{cases} 1 \; if \; \lvert r\rvert < b_1 \\ \frac{b_2 - \lvert r\rvert}{b_1 - b_2} \; if \; b_1 \le \lvert r\rvert \le b_2 \\ 0 \; if \; \lvert r\rvert > b_2 \end{cases}$	$\frac{\tanh(r)}{r}$	$\frac{\delta^2}{\delta^2 + r^2}$

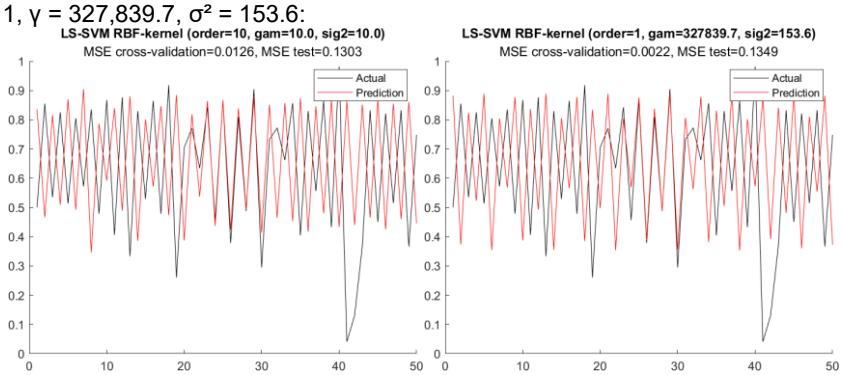
For Huber, small errors $|r| < \beta$ get full weight = 1. Beyond β, the weight decays as $\frac{\beta}{r}$, so large outliers are down-weighted but not completely ignored. For Hampel, errors up to b_1 are also given full weight of 1. Between b_1 and b_2 the weight tapers linearly to zero, and beyond b_2 outliers are completely ignored as the weight is 0. Logistic smoothly down-weights large residuals without ever going to zero, so good for heavy-tailed but not quite extreme noise. Myriad gives very heavy tailed robustness - outliers get almost zero weight, but in a smooth way:



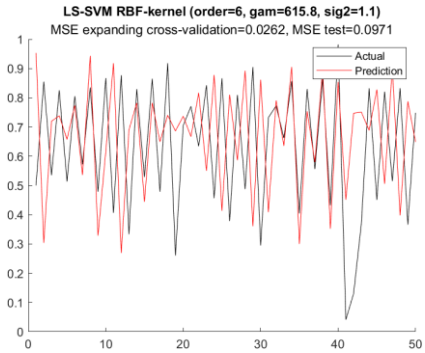
All four robust schemes recover the sinc curve accurately in our case. Hampel gives the strongest outlier rejection as the weights are set to 0. This is ideal when we are sure that the points are spurious, which is the case here. Logistic and Myriad offer a softer down-weighting - outliers still contribute slightly, but much less than with MSE. Similarly, Huber is somewhere in between, down-weighting the influence without completely discarding outliers. The choice between these methods depends on how aggressively we want to ignore extreme residuals: Hampel zeroes them out, Huber caps them, and Logistic/Myriad provides smooth, bounded downweighting.

2.2

The initial prediction used an order of 10, with both γ and σ² set to 10. For hyperparameter tuning, we iterated over orders 1–30 and, for each, applied simplex optimization to find the best γ and σ based on the standard 10-fold cross-validation. We then selected the combination with the lowest validation MSE across all orders, which turned out to be order = 1, γ = 327,839.7, σ² = 153.6:



Both results – the initial one and tuned one - are not satisfactory. The tuned model produced an even higher MSE on the test set than our intial settings. The reason is that the standard k-fold CV introduce data leakage for time-series data, using future data to predict the past, causing overfitting. Therefore, an alternative approach is considered. As an alternative, we adopted expanding window cross-validation, which starts with an initial training sample of 100 observations, splits the remaining data into 5 sequential chunks, for each chunk: test on that chunk, then append it to the training set before moving on to the next. We searched orders 1, 6, 11, 16, 21, 26, σ² in [0.01, 1000] (log-spaced), and γ in [0.01, 10^6] (log-spaced). The configuration with the lowest average MSE over the 5 chunks is: order = 6, γ = 615.8, σ² = 1.1:



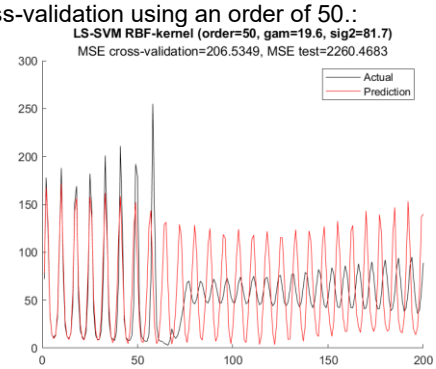
On the test set, this model achieved an MSE of 0.0971, which is significantly lower than the above 0.13 we saw both initially and after tuning on regular k-fold cross-validation, indicating that the performance is improved. Further improvements could come from embedding simplex or grid-search optimization methods based on this expanding-window cross-validation to fine-tune γ and σ^2 in a more proper manner.

2.3

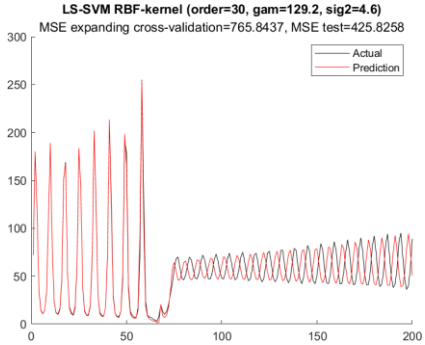
An initial order of 50 may be justified if the process of laser's intensity exhibits correlations spanning that many time steps. However, such a high order greatly increases the feature dimension and thus the number of support vectors - potentially causing overfitting, higher computational cost, and the modeling of spurious long-range dependencies. Therefore, although an order of 50 is not inherently unreasonable, its value should be confirmed through appropriate time-series validation.

Hyperparameters must be optimized on validation-set performance while respecting the sequential nature of time series. As depicted in 2.2, standard k-fold CV breaks temporal order - training on future observations to predict past ones - leading to data leakage, overly optimistic error estimates, and overfitting. A more appropriate strategy is expanding-window cross-validation, which begins by training on an initial portion of the series and then testing on the immediately following block. That block is subsequently added to the training set, and the cycle repeats until the entire series has been covered. This scheme preserves chronological order and yields much more realistic validation scores.

For an initial benchmark, and to demonstrate the shortcomings of naive cross-validation, we first tuned γ and σ^2 via simplex under standard 10-fold cross-validation using an order of 50.:



To more appropriately handle the time-series structure, we then switched to expanding-window cross-validation: the first 400 samples formed the initial training set, and the remaining data were split into five sequential chunks. We searched orders from 5 to 50 (in steps of 5), σ^2 from 0.01 to 100 (log-spaced), and γ from 0.01 to 1,000 (log-spaced). The combination minimizing the average validation MSE across all chunks was order of 30, γ of 129.2, and σ^2 of 4.6:

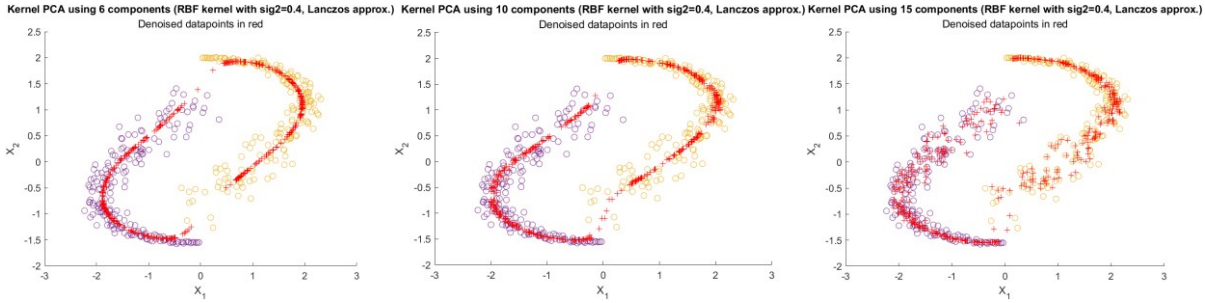


When retrained on the full training set and evaluated on the test set, this model achieved an MSE of 426, which is significantly better than the baseline's 2 260. The predicted values align closely with the true laser intensities on test data, confirming good performance and the benefit of time-aware validation. Further improvements could be done by embedding simplex or grid-search routines directly into the expanding-window scheme to fine-tune γ and σ^2 even more precisely.

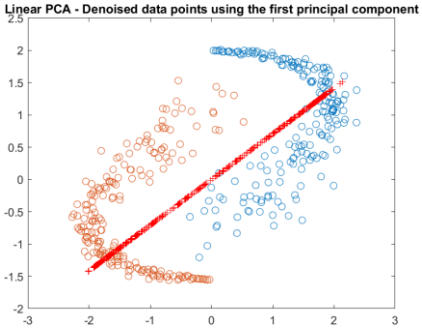
Exercise 3

1.1

Principal component analysis (PCA) reduces the number of dimensions in datasets to principal components that retain most of the original information. Denoising by PCA is possible because high-variance components tend to contain the true signal, whereas low-variance components are often dominated by noise. By reducing the data to include only the components with the highest variance, the low-variance components are discarded, thereby suppressing the noise while still capturing the important structure of the input data. As the number of principal components increases, more of the original variance is recovered, but noise from the lower-variance components is also reintroduced. This is evident in the kernel PCA (RBF kernel, σ = 0.4, Lanczos approximation) on a dataset of two spirals represented by 400 points with a dispersion of 0.3 below using 6, 10 and 15 principal components. The higher the number of components, the more noise is retained.

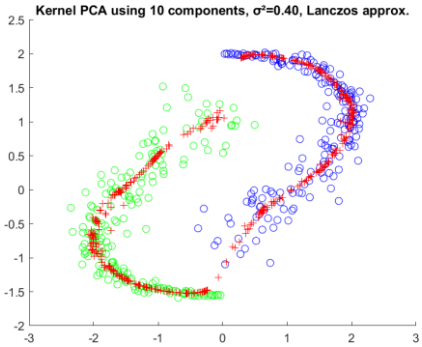


The fundamental difference between Linear PCA and Kernel PCA lies in how they extract patterns from data. Linear PCA identifies the directions of maximum variance that are orthogonal to each other by performing straightforward linear transformations on the original feature space. In contrast, kernel PCA uses the kernel trick to implicitly map the data into a higher-dimensional feature space before finding the principal components. This allows kernel PCA to capture nonlinear relationships that linear PCA would miss, as demonstrated in our example.



Regarding the number of obtainable principal components, linear PCA is limited to a maximum of min(n−1, d) components, where n is the number of observations and d is the dimensionality. However, Kernel PCA can extract up to n components, regardless of the original data's dimensionality. Thus, when working with datasets having few original dimensions, Kernel PCA can extract a higher number of components than the original dimensionality, reflecting complex, nonlinear patterns in the transformed space.

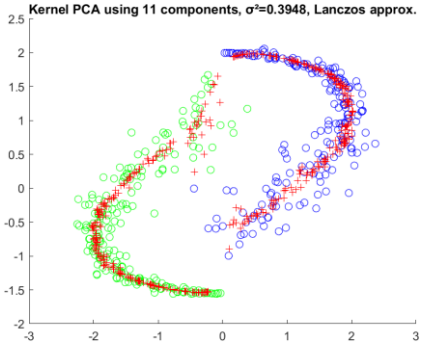
To tune the number of components, the kernel width, and the approximation method, we partitioned the noisy data into four folds and carried out 4-fold cross-validation. An exhaustive grid search is run over number of components between 2 to 15, σ of 0.1, 0.4 or 1, and both Lanczos and Nystrom approximations, selecting the combination that minimized the average reconstruction error on validation folds against the clean ground truth.



The optimal setting turned out to be 10 components, $\sigma^2 = 0.4$ with Lanczos, achieving the lowest mean squared error across validation sets.

To further enhance efficiency and coverage of the parameter space, Bayesian optimization is also applied, which adaptively explores promising regions based on prior evaluations. This was run for 30 iterations, tuning the number

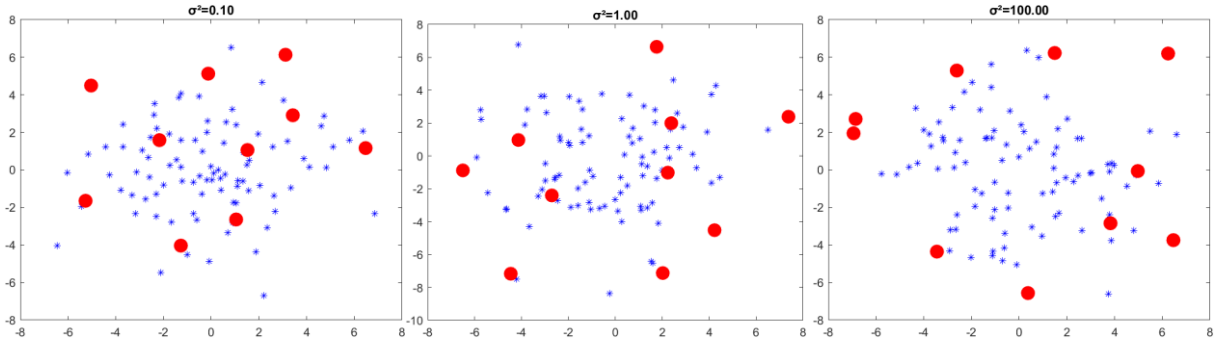
of components between 2 and 15, σ continuously between 0.1 and 10, and selecting between Lanczos and Nystrom approximations. The resulting optimal setting is similar: 11 components, σ² = 0.3948, with Lanczos method:



1.3

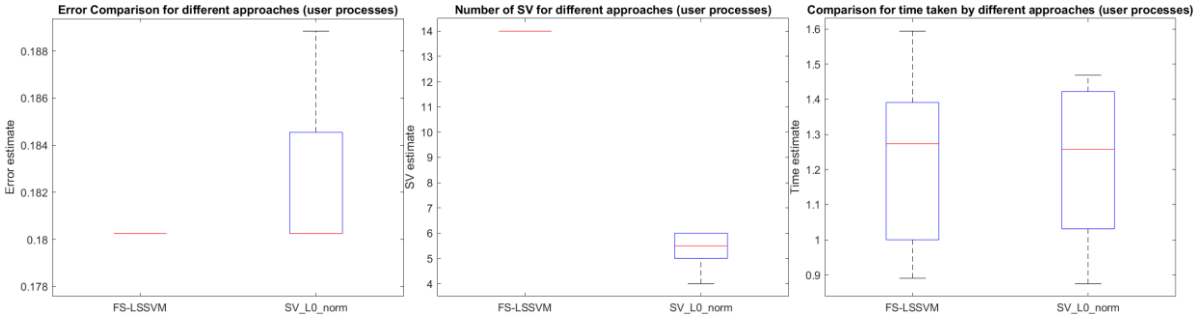
When the dataset is so large that constructing or storing the full kernel matrix becomes infeasible, solving the LS-SVM in the primal is preferred. In the primal, we optimize directly over the weight vector in a (possibly approximate) feature space of dimension $m \ll n$, leading to an $m \times m$ linear system rather than an $n \times n$ one. Conversely, when the input dimensionality d is high but n is moderate, the dual formulation of using kernels directly can be more efficient, since disappears and only the kernel matrix $n \times n$ is needed.

To approximate the unknown feature map we select a small subset of support vectors of the input data of size $M \ll N$, M equals 10 in our case, and then uses it to approximate the feature map such that a parametric solution can be found. For fixed-size LS-SVMs the subsample starts randomly and is then updated iteratively as long as the Renyi entropy improves.



Based on the above, a larger value for σ² leads to a selection of data points that are more widely dispersed. This is because for small σ² kernel affinities are very local, so maximizing entropy picks landmarks clustered around high-density or boundary regions. As the bandwidth increases, the reach of each data point grows, the similarity with distant points increases and maximising entropy, which involves prioritising dissimilar points, causes points maximally distanced from each other to be selected as support vectors. Once the iterative entropy maximization stabilizes, the support vectors converge to the most diverse points in feature space - those that are maximally dissimilar under the chosen kernel.

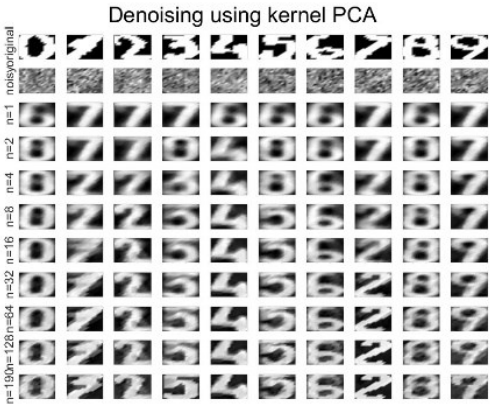
The comparison between fixed-size LS-SVM and ℓ_0-approximation is done on the shuttle dataset using Coupled Simulated Annealing (CSA) global tuning, a linear kernel, and a scaling factor of choosing representative points for the entropy estimation of 4. All runs are evaluated in terms of test errors, number of support vectors, and computational time:

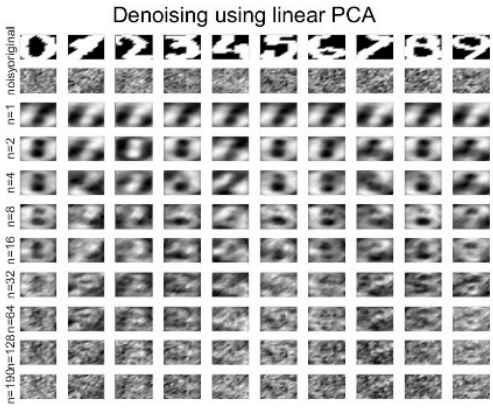


The ℓ_0-approximation post-processes the fixed-size LS-SVM solution with an ℓ_0 penalty to drive extra sparsity. This adds an O(M³) refinement step, but it is negligible compared to the main solve's cost. The main fixed-size LS-SVM solves an O(N*M²) problem, and ℓ_0 post-processing adds O(M³) per iteration. In practice, total computational times are nearly identical, though the ℓ_0 version occasionally spikes slightly higher because of its iterative pruning. With ℓ_0 post-processing, the result is much more sparse due to the lower number of support vectors. Since the ℓ_0 norm counts the number of non-zero values aiming for sparse representations, the model retains only about 5 support

vectors compared to 14 for the fixed-size LS-SVM. The increased sparsity may come at a slightly higher error, but it does not raise the test error significantly - from roughly 0.18 to about a range around 0.182.

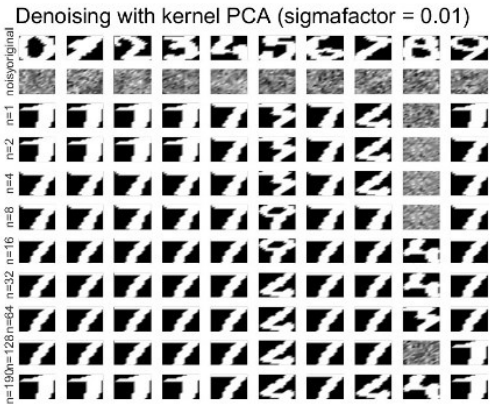
2.

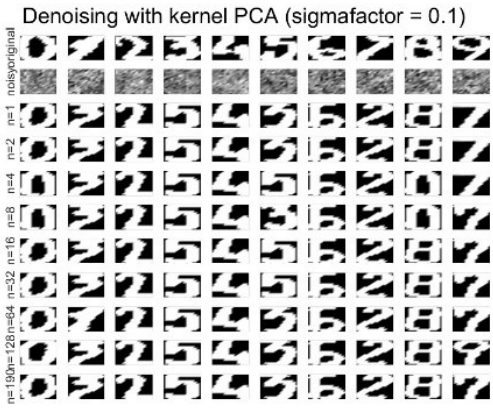


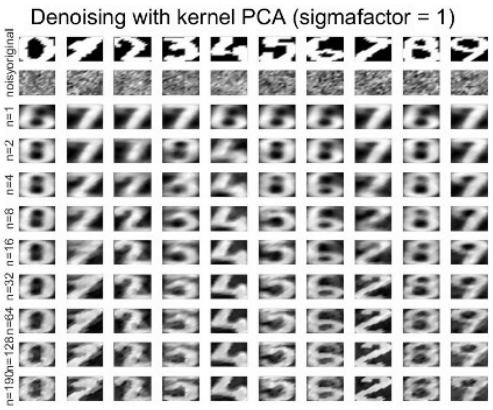


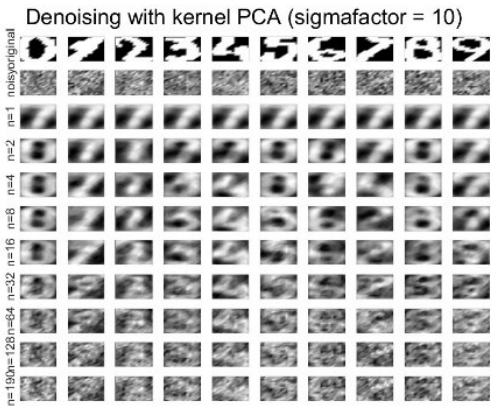
Linear PCA reconstructs noisy digits only along global, linear directions and thus leaves substantial residual noise and blurs digit-specific features. On the other hand, Kernel PCA with an RBF kernel can capture the nonlinear contours of each digit class. In practice, with around 64 principal components, Kernel PCA quite cleanly recovers digit shapes, whereas linear PCA reconstructions remain ambiguous. Beyond a certain number of components, both methods begin to reintroduce noise - but Kernel PCA retains much sharper, more digit-like structure throughout.

The influence of σ^2 parameter is investigated by changing sigmafactor parameter over equispaced values in logarithmic scale between 0.01 and 10:





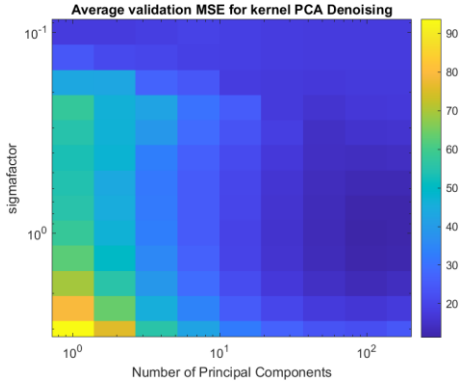




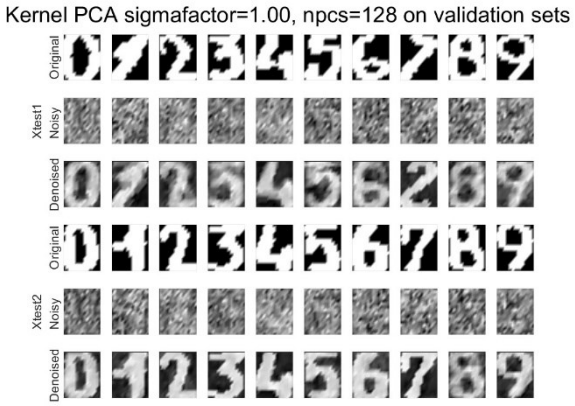
When σ^2 is much smaller than the suggested estimate, the RBF kernel becomes too narrow and each sample is effectively isolated from its neighbours. In this regime, the principal directions mostly capture high-frequency noise rather than the true digit manifold. While the reconstructions appear clean, they may represent different digits; for instance, a noisy 9 could be reconstructed as a smooth 7. In contrast, when σ^2 is much larger than the suggested estimate, the RBF kernel becomes overly broad, treating almost every point as equally similar. Kernel PCA then collapses into global variation modes and can no longer discriminate between digit-specific structures and

background noise. Consequently, noise is mitigated slowly and to a limited extent, resulting in blurred reconstructions that lose the shapes that define each digit. The right balance between locality and global structure is achieved for a moderate value of σ² – around the suggested estimate (sigmafactor = 1). Nearby pixel patterns cluster tightly enough together to reject noise, yet the kernel remains broad enough to capture the overall manifold of each digit class. In our experiments, this setting produced the most accurate reconstructions with only a minimal amount of noise.

The reconstruction error on training and validation sets is evaluated using kernel PCA with different denoising parameters. It is calculated as the mean squared error between the reconstructed and original images. To select the optimal parameter values such that the error on the validation sets is minimal, an exhaustive search is conducted over different numbers of components: 1, 2, 4, 8, 16, 32, 64, 128, or 190, and different σ² values using sigmafactor values equally spaced on a logarithmic scale between 0.01 and 10^(0.5):



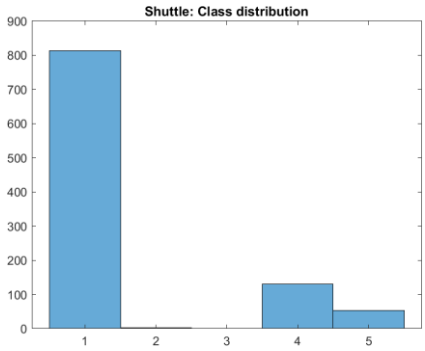
The lowest average MSE is achieved by using a sigmafactor of 1 and a number of components of 128.



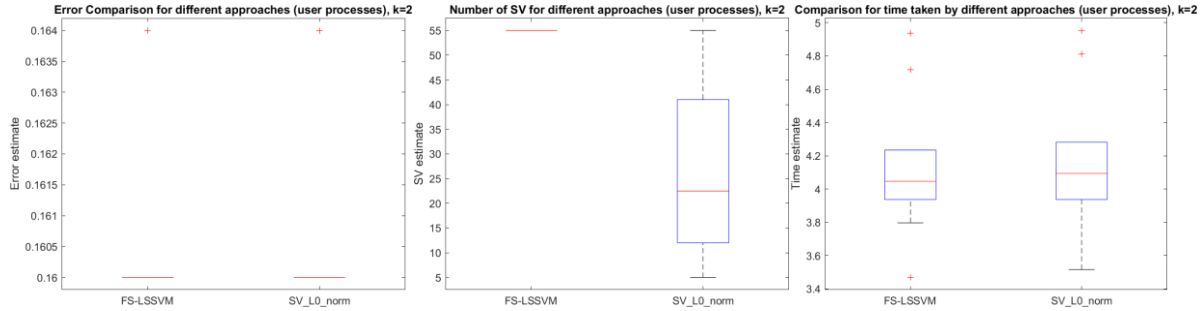
With this configuration, most of the validation digits are reconstructed accurately and quite cleanly, especially in Xtest2. There remains a small failure on Xtest1, for example, where a noisy 7 is reconstructed looking like a 2. However, the visual quality is still noticeably better than with non-tuned settings, with denoising being improved.

2.2.1

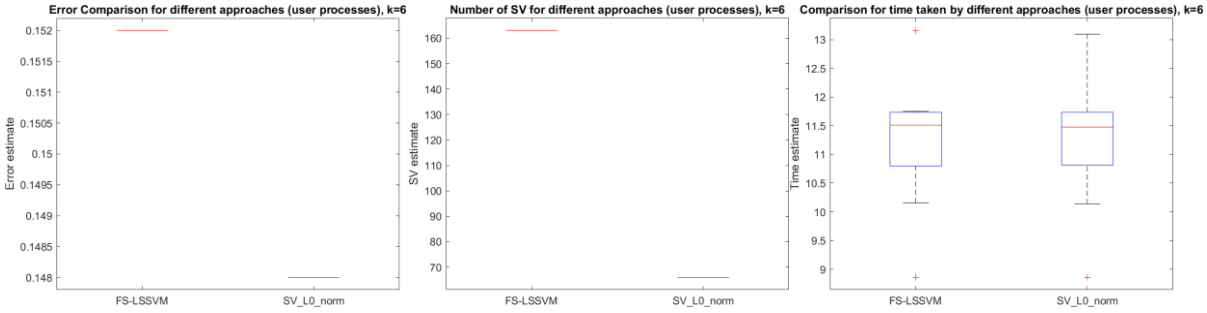
The Shuttle dataset contains 58 000 observations and 10 variables: 9 numeric features plus a target class label. There are 7 possible classes, but the data are highly imbalanced - about 80% of all samples belong to class 1. Therefore the default accuracy is about 80%, i.e. it is expected to beat the 80% baseline accuracy. For faster experimentation, we used only the first 1 000 samples - in that subset, only 5 classes actually appear.



We ran both Fixed-size LS-SVM and the ℓ_0-approximation methods on all 9 features. We varied the entropy-subset scaling factor k over {2, 4, 6}, used Coupled Simulated Annealing (CSA) for global tuning, and compared two kernels in turn: RBF and polynomial. A stratified 25% hold-out of these 1 000 samples was used for final testing. For RBF kernel the lowest median test error occurred at k=2:



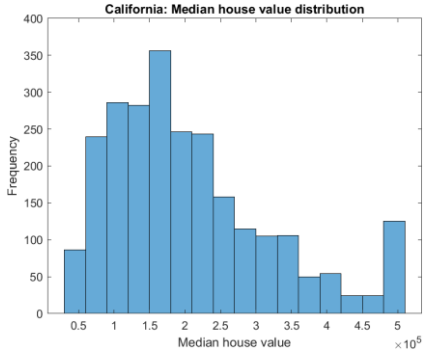
For polynomial the lowest error was at k=6, though at the cost of longer computation:



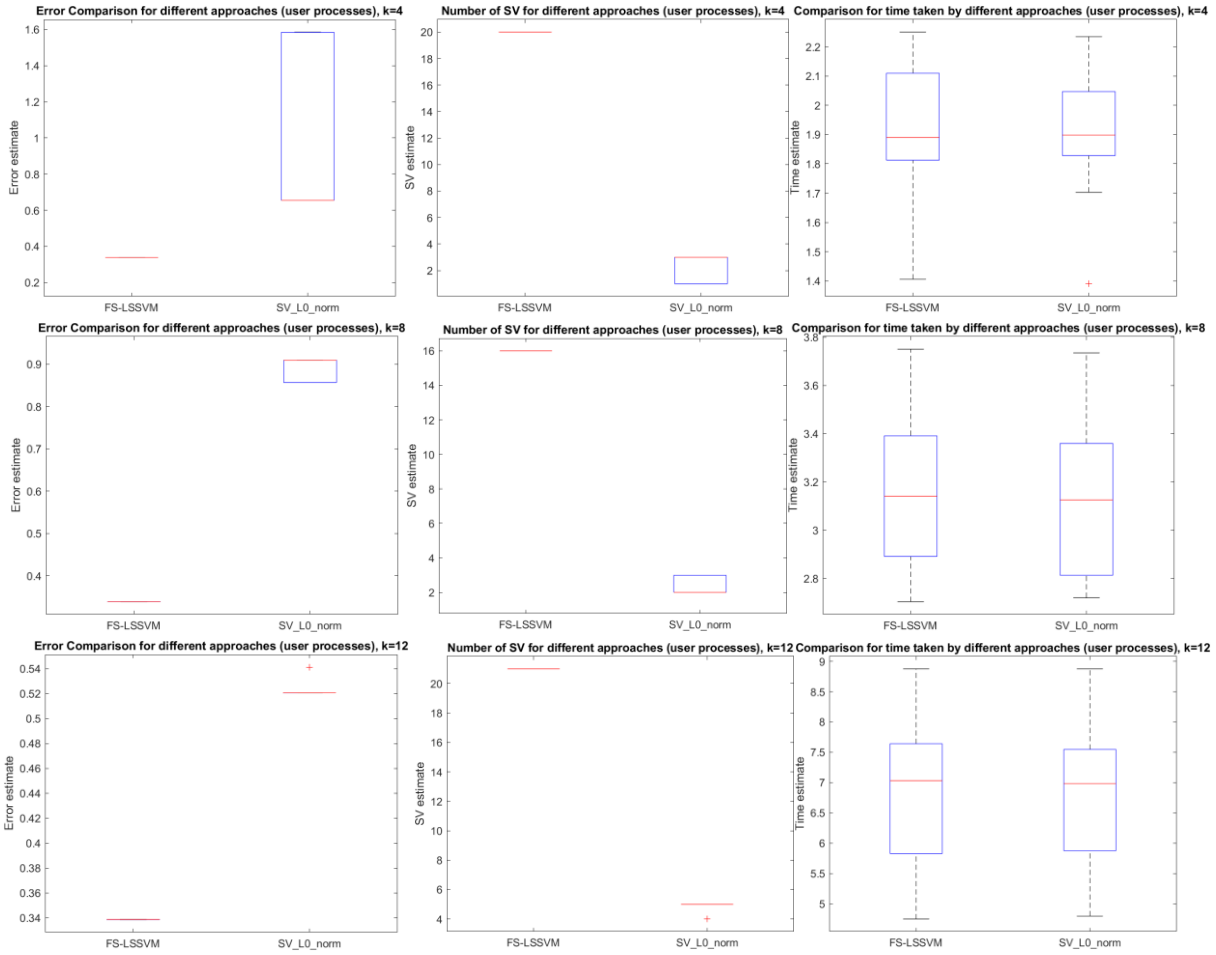
Overall, the polynomial kernel gave a marginally better best-case error than RBF. Kernel choice interacts with k - here the polynomial kernel needed a larger k to match or exceed RBF's best performance. The factor k trades off between representation and complexity - small k runs fastest but may underrepresent rare classes and large k runs slower and risks overfitting. The 80% majority-class baseline accuracy is surpassed for both cases. Further work should scale up to the full dataset and tune hyperparameters more extensively.

2.2.2

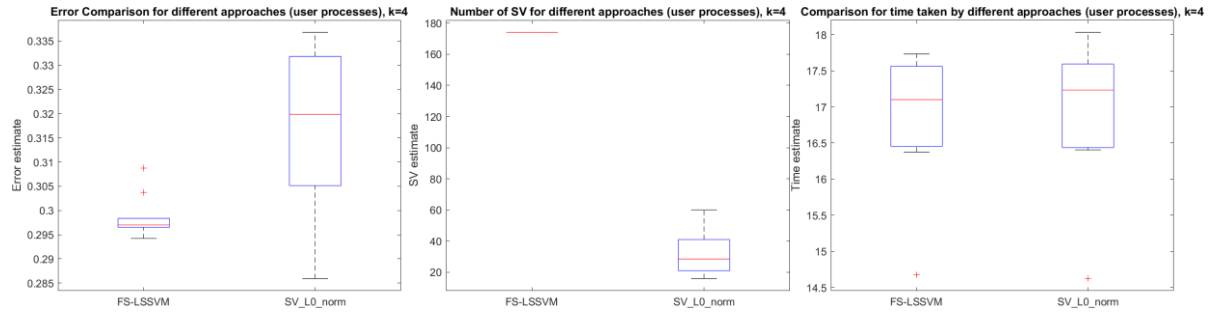
The California dataset consists of 20 640 observations and 9 variables - 8 features (median income, housing median age, total rooms, total bedrooms, population, households, latitude, and longitude) and a target variable of median house value. For faster processing, we select the first 2 500 observations as our final dataset:



The distribution shows that most of the median house values are around 100 000 to 250 000. There are also some very expensive values on the right tail, around 500 000. However, we do not consider these as outliers but rather as reflecting wealthier districts. Moreover, Bayes Automatic Relevance Determination (ARD) using RBF kernel and 5-fold cross-validation shows that no features are pruned, so all are considered significant for the target variable. The most important one, as shown in the ranking, is median income. Both fixed-size LS-SVM and ℓ_0-approximation are therefore run on all variables using different scaling factor values for choosing representative points for the entropy estimation, with k set to 4, 8, and 12, using CSA global tuning and a linear kernel. A stratified sample of 25% of the whole dataset is held out as the test set.



Comparing both methods, we obtain conclusions similar to those in 1.3. We further observe that increasing the scaling factor k reduces test set error at the cost of higher computational effort. A higher k leads to more representative points and potentially better accuracy, because the entropy-based subset better captures the structure and variability of the full dataset. However, increasing k also adds redundant information and, beyond a certain point, may not significantly improve accuracy but greatly increases computational cost. A small k risks underfitting, while a large k can increase the risk of overfitting. Furthermore, the RBF kernel is used to compare the results with those of the linear kernel:



Using k = 4 achieves lower errors than any other tested k value for the linear kernel. Therefore, despite taking more computational time, the RBF kernel is preferred. RBF kernel also results in more support vectors. Further work on function estimation for this dataset could include testing different k values for the RBF kernel, a polynomial kernel, as well as incorporating ensemble methods such as combining LS-SVM models in committee networks or exploring robust methods.

GenAI code of conduct for students (2024-2025)

Generative AI (GenAI) assistance tools can be used to generate text, image, code, video, music or combinations of these. It includes typical tools like (but this list is not limited to): ChatGPT, Google Gemini, MS Copilot, Midjourney, Claude.ai, Perplexity.ai, Dall-E, …

Student name: Bartosz Bogucki

Student number: r1032166

Please indicate with "X" whether it relates to a course assignment or to the master thesis:

X This form is related to a **course assignment**.

Course name: Support Vector Machines: Methods and Applications

Course number: H02D3a

O This form is related to **my Master thesis**.

Title Master thesis: .....

Promoter: .....

Daily supervisor: .....

Please indicate with "X":

O **I did not use** any GenAI assistance tool.

X **I did use** GenAI Assistance. In this case **specify which ones** (e.g. ChatGPT, ...):

	Name of the GenAI tool used *Please indicate with "X" (possibly multiple times)* *in which way you were using GenAI:*				
GenAI assistance used as/for:	ChatGPT	DeepL Write			
Language assistance	X	X			
Search engine					
Literature search					
Short-form input assistance					
Generating programming code	X				
Generating new research ideas					
Generating blocks of text					
Other (specify):					

X **As a language assistant for reviewing or improving texts I wrote myself**

- *Code of conduct*: This use is similar to using spelling and grammar check tools, you do not have to refer to use of GenAI in the text.
  Be careful:
  - Using GenAI tools on texts you did not write yourself to cover up plagiarism (by paraphrasing original texts) is not allowed.

O **As a search engine to get information on a topic or to search for existing research on the topic**

- *Code of conduct*: This use is similar to e.g. a Google search or checking Wikipedia. If you then write your own text based on this information, you do not have to refer to use of GenAI in the text.
  Be careful:
  - Be aware that the output of the GenAI tool cannot be guaranteed as a 100% reliable source of information. The output may not be entirely correct and be limited due to the databases it uses. Knowledge evolves and may change over time, it may be that the database of the GenAI tool is not up to date.

O **For literature search**

- *Code of conduct*: This use is comparable to e.g. a Google Scholar search.
  Be careful:
  - Be aware that the output is restricted to the database it is built on. After this initial search, look for scientific sources and conduct your own analysis of the source documents. Interpret, analyse and process the information you obtained; verify it and don't just copy-paste it.
  - Be aware that some GenAI tools (like ChatGPT) may output no or wrong references. As a student you are responsible for further checking and verifying the absence or correctness of references; don't just copy-paste it.

  If you then write your own text based on this information, you do not have to refer to use of GenAI in the text.

O **For short-form input assistance**

- *Code of conduct*: This use is similar to e.g. Google docs powered by generative language models

X **To generate programming code**

- *Code of conduct*: Use of GenAI for coding should be explicitly allowed by the teacher. If used for coding, correctly mention the use of GenAI assistance and cite it.

O **To generate new research ideas**

- *Code of conduct*: Further verify in this case whether the idea is novel or not. It is likely that it is related to existing work, which should be referenced then.

O **To generate blocks of text**

- *Code of conduct*: Inserting blocks of text without quotes and a reference to GenAI assistance in your report or thesis is not allowed. According to Article 84 of the exam regulations in evaluating your work one should be able to correctly judge on your own knowledge, understanding and skills. In case it is really needed to insert a block of text from a GenAI tool, mention it as a citation by using quotes. But this should be kept to an absolute minimum. When you literally copy elements from a conversation with a GenAI tool: Quote between quotation marks and refer according to the specified reference style or as a personal communication within the text itself. Describe the use of the GenAI-tool (tool name, version, date, ...) in the method section (if there is one) and optionally add the (link to the) full conversation as an attachment.

O **Other**

- *Code of conduct*: Contact the professor of the course or the promoter of the thesis. Inform also the program director. Motivate how you comply with Article 84 of the exam regulations. Explain the use and the added value of ChatGPT or another AI tool: ….

Further important guidelines and remarks

- GenAI assistance cannot be used related **to data or subjects under Non-Disclosure Agreement.**
- GenAI assistance cannot be used related **to sensitive or personal data due to privacy issues**.
- **Take a scientific and critical attitude** when interacting with GenAI assistance and interpreting its output. Don't become emotionally connected to AI tools.
- As a student you are responsible for complying with Article 84 of the exam regulations: your report or thesis should reflect your own knowledge, understanding and skills. Be aware that plagiarism rules also apply to (work that is the result of) the use of GenAI assistance tools.
- **Exam regulations Article 84**: "Every conduct individual students display with which they (partially) inhibit or attempt to inhibit a correct judgement of their own knowledge, understanding and/or skills or those of other students, is considered an irregularity which may result in a suitable penalty. A special type of irregularity is plagiarism, i.e. copying the work (ideas, texts, structures, designs, images, plans, codes , ...) of others or prior personal work in an exact or slightly modified way without adequately acknowledging the sources. Every possession of prohibited resources during an examination (see article 65) is considered an irregularity."
- In order to maintain academic integrity and avoid plagiarism **more information about being transparent on the use of GenAI assistance and about citing and referencing GenAI** can be found on this website for students (Dutch/English).
- **Additional reading : KU Leuven guidelines on responsible use of Generative AI tools, and other information** (Dutch/English)

A few final words

If you are uncertain whether or not you should declare your use of AI tools, we suggest that you discuss the matter with your teacher or promoter. It is safer to declare AI use when it is not needed than to withhold that declaration when it is required.

Finally, remember that advanced AI tools are new and that they can do things they could not do, up until recently – so we do not have all the answers about how to use them responsibly yet. It is important to follow-up on most recent evolutions in AI technologies, to be a bit cautious, to communicate with teachers, teachings assistants, supervisors, promoters and peers with open minds, to be as transparent as we can, and to learn together as we move along.

This code of conduct can be used as a framework for academic year 2024-2025, changes will be made based on new evolutions.