ELEN0062 - Introduction to Machine Learning
Project 1 - Classification algorithms

DOMINIKA PIECHOTA (S2505196) and PAWEL DOROSZ (S2505195)

1 Decision Trees

(1.1) (a) The shape of the decision boundary changes significantly as the tree depth increases:

- **max_depth = 1:**
  The boundary is a single straight line that roughly separates the two classes. - only one split

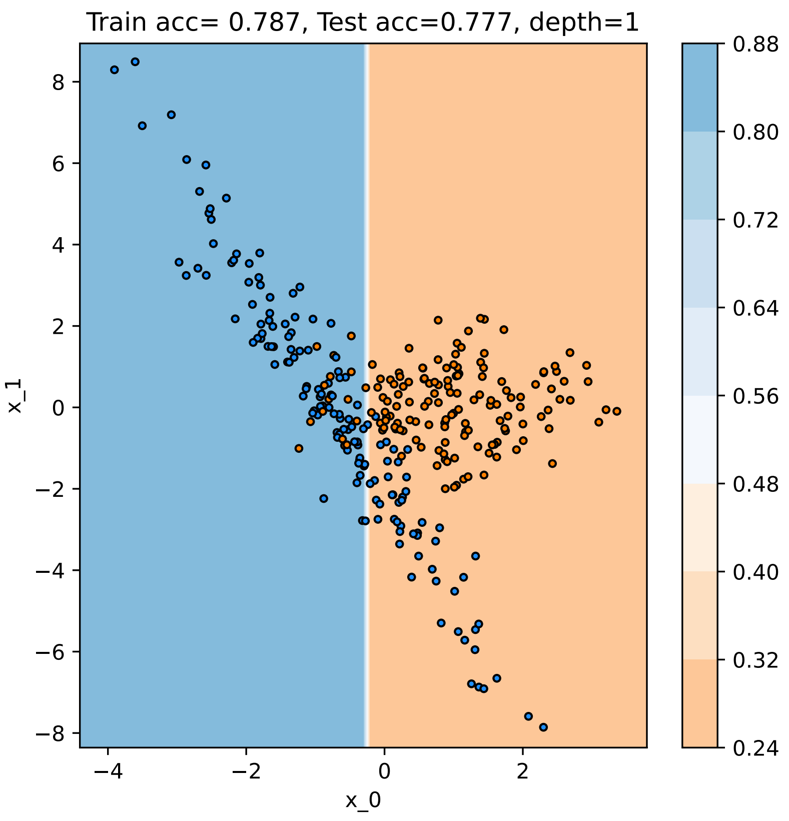

Fig. 1. Decision boundary for max_depth = 1

- **max_depth = 2:**
  The boundary becomes slightly more detailed with two possible splits. The classifier starts to capture some structure but still misses important details in the data distribution.

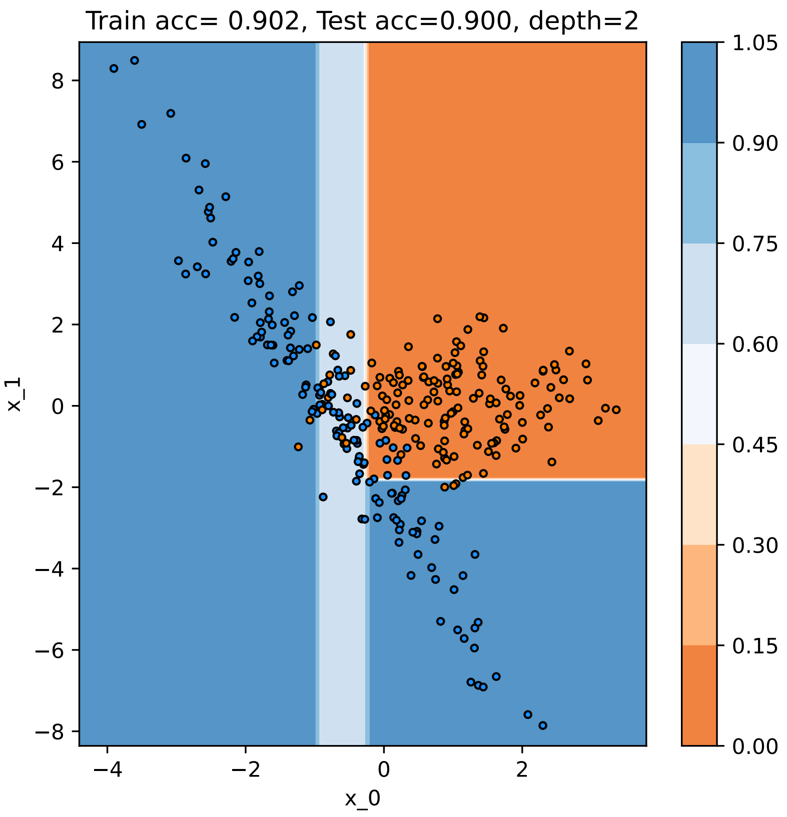

Fig. 2. Decision boundary for max_depth = 2

- **max_depth = 4:**

  The decision boundary becomes more complex, with multiple rectangular regions. - 4 possible splits

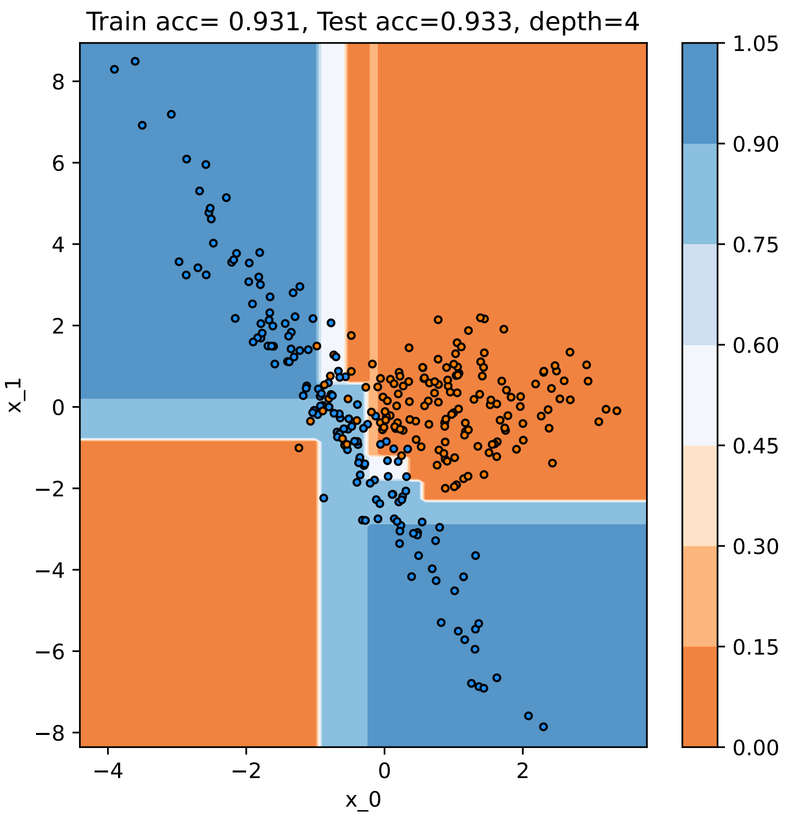

Fig. 3. Decision boundary for max_depth = 4

- **max_depth** = **6:**
  The boundary is now highly irregular and adapts closely to the training samples. The model starts to memorize small variations in the data.

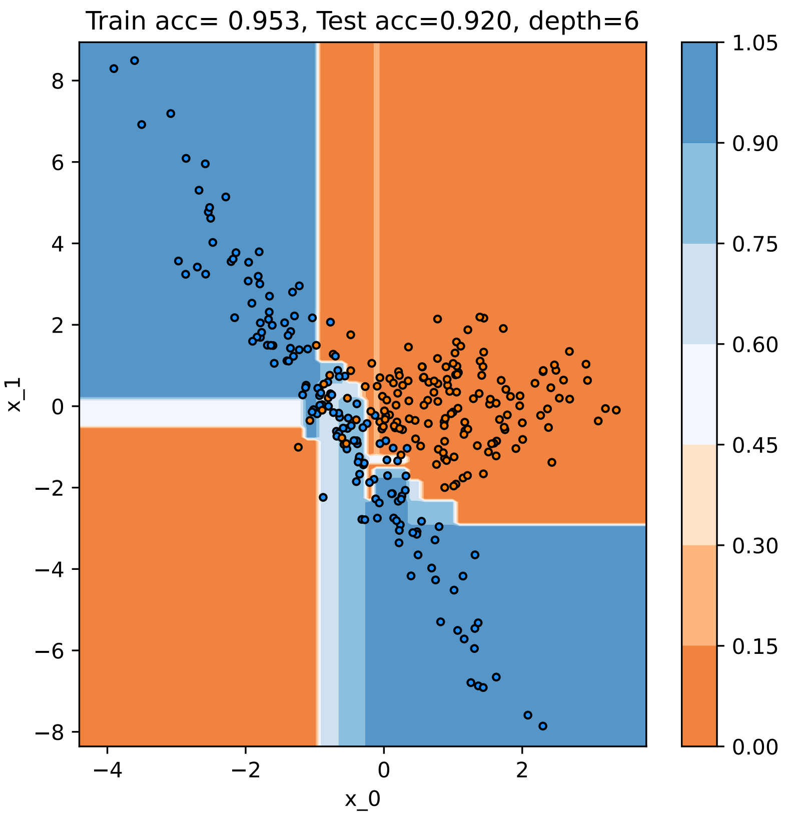

Fig. 4. Decision boundary for max_depth = 6

- **max_depth = None:**
  The decision boundary is extremely fragmented and follows almost every single training point. The model has perfectly learned the training data.

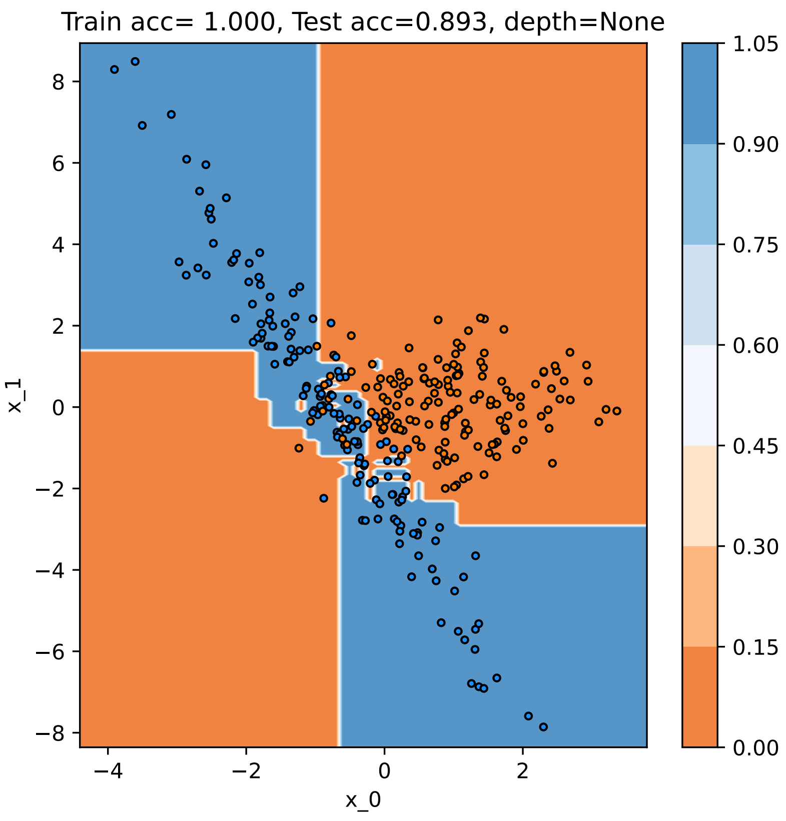

Fig. 5. Decision boundary for max_depth = None

(b)
- For **max_depth = 1 and 2**, the model clearly underfits. The boundaries are too simple, and both training and test accuracy are low.
- For **max_depth = 4**, the model shows the best generalization performance. The training and test accuracies are both high, and the boundary aligns well with the data distribution. This depth gives a good bias-variance trade-off.
- For **max_depth = 6**, the model starts to overfit. The training accuracy becomes nearly perfect, while test accuracy slightly decreases. The boundary is overly detailed and follows noise in the data.
- For **max_depth = None**, the overfitting is strongest. The model memorizes the training samples, leading to training accuracy equal to 1 but much lower test performance. The boundary becomes fragmented and unstable across different dataset generations.

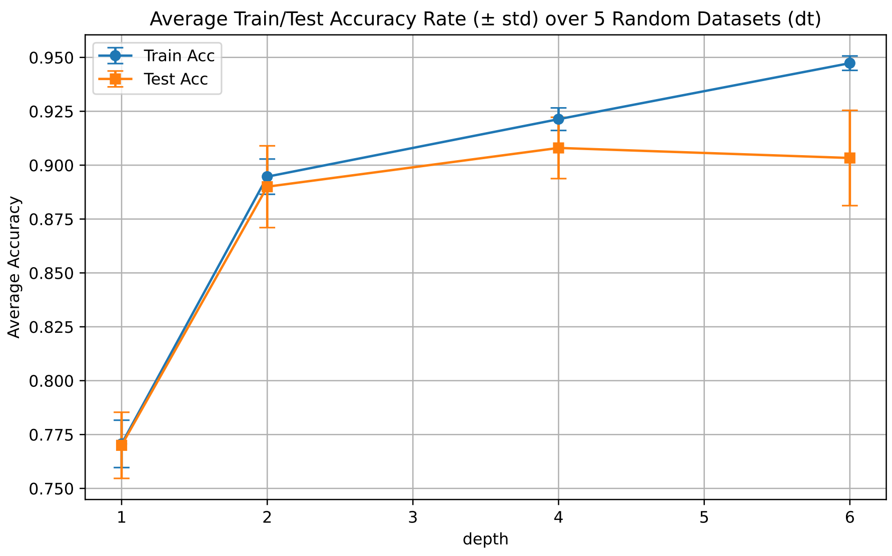

Fig. 6. Average train/test accuracy

(c) As the tree depth increases, the leaves contain fewer samples. These leaves usually contain only one class, making the model's class probabilities very close to 0 or 1 (the impurity is low). This leads to higher apparent confidence in predictions.

(1.2) Average Test Accuracies and Comments

- For **max_depth** = **1**, the mean test accuracy is around 0.77 with a standard deviation of about 0.015. The model clearly underfits and cannot separate the classes well.
- For **max_depth** = **2**, the average accuracy increases to about 0.89 (±0.019). The model improves, but it still misses local variations. The standard deviation is high so model with this depth (and max-depth = 1 also) is sensitive with respect of dataset.
- For **max_depth** = **4**, the mean accuracy reaches around 0.908 (±0.014). This depth provides the best overall performance with good generalization due to low variability.
- For **max_depth** = **6**, the average accuracy slightly decreases to around 0.903 (±0.022). The model starts to overfit, as seen by higher variability between runs.
- For **max_depth** = **None**, the average accuracy drops further to around 0.891 (±0.024). Despite perfect training accuracy, the model fails to generalize to unseen data. The std is high because it overfit to each dataset.

2 k-Nearest Neighbors

(2.1) (a) Next pages

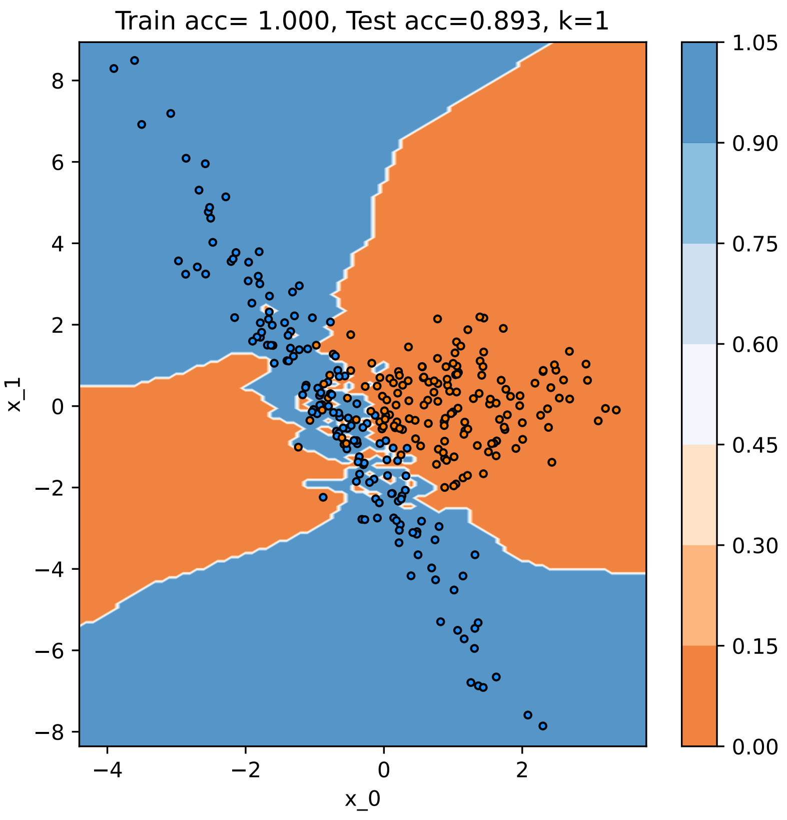
Train acc= 1.000, Test acc=0.893, k=1
x_1
x_0
8
6
4
2
0
−2
−4
−6
−8
−4
−2
0
2
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

Fig. 7. k=1

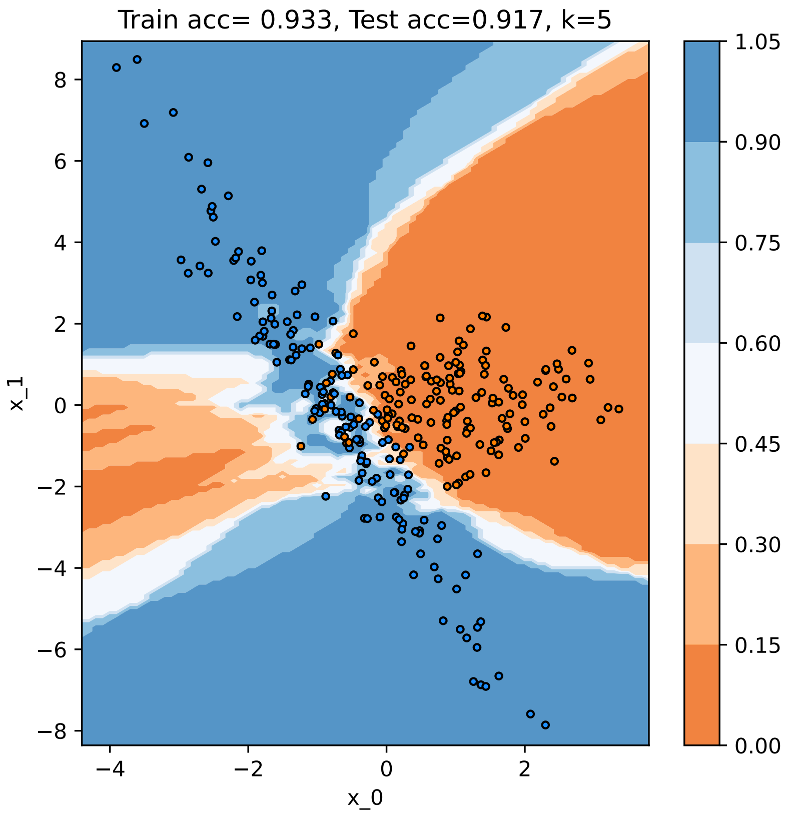
Train acc= 0.933, Test acc=0.917, k=5
x_0
x_1
−4
−2
0
2
8
6
4
2
0
−2
−4
−6
−8
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

Fig. 8. k=5

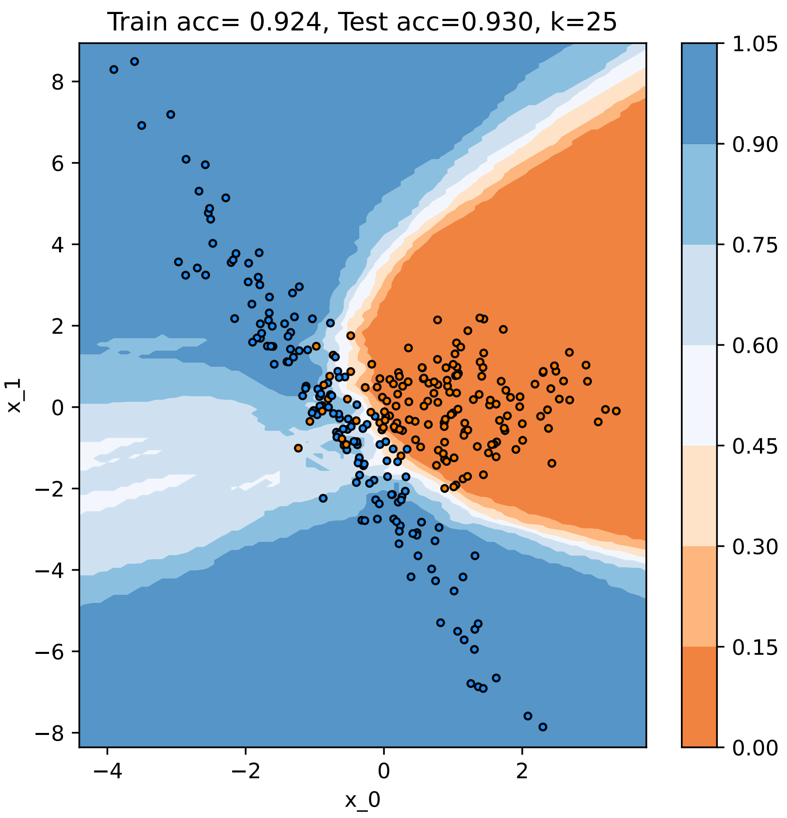
Train acc= 0.924, Test acc=0.930, k=25
x_1
x_0
8
6
4
2
0
−2
−4
−6
−8
−4
−2
0
2
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

Fig. 9. k=25

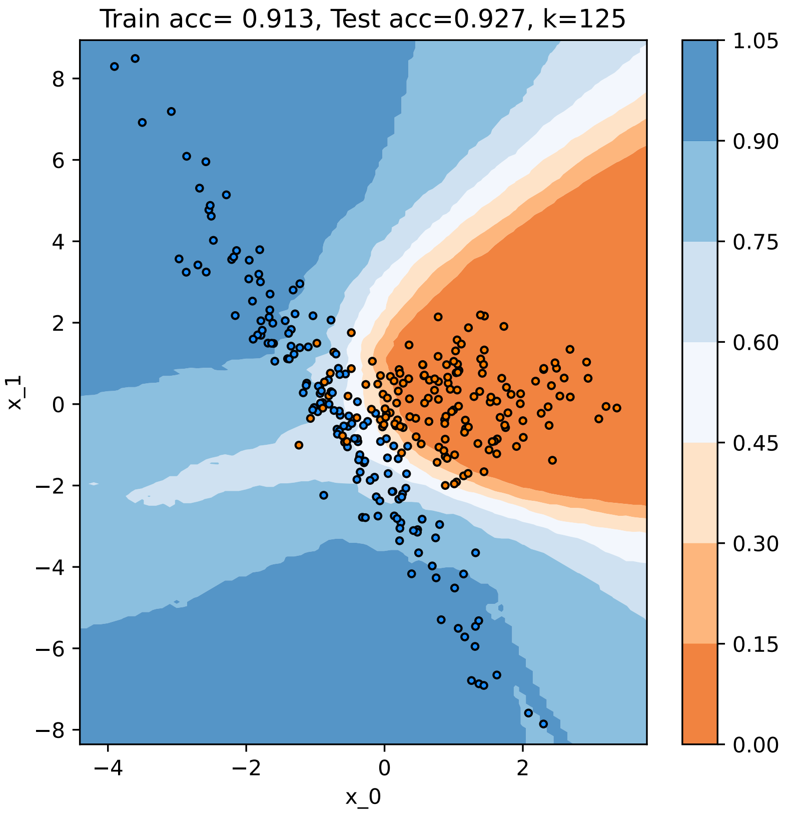
Train acc= 0.913, Test acc=0.927, k=125
x_0
x_1
−4
−2
0
2
8
6
4
2
0
−2
−4
−6
−8
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

Fig. 10. k=125

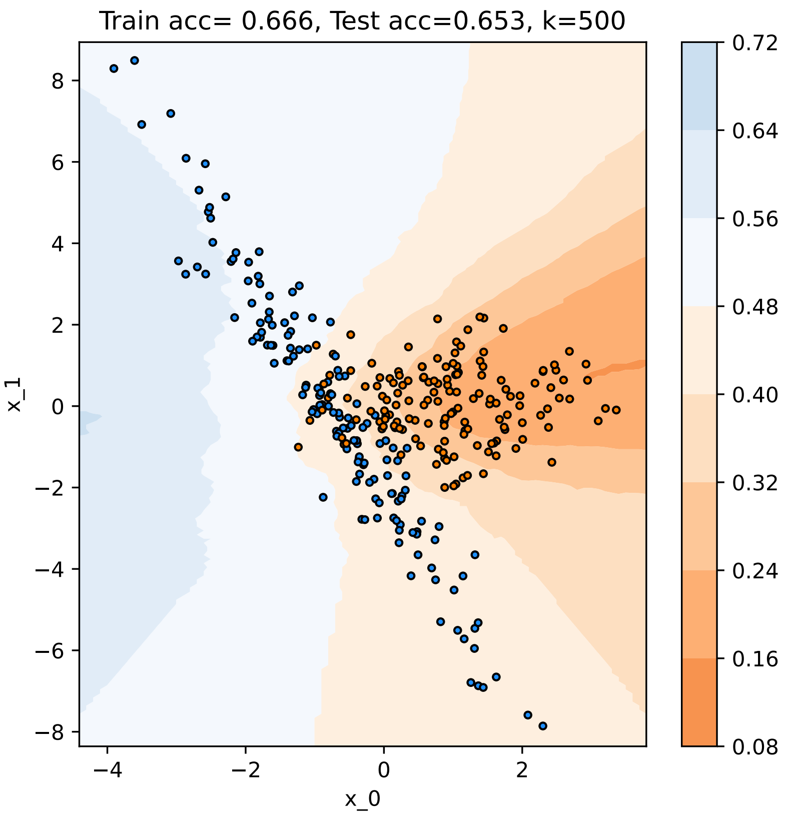
Train acc= 0.666, Test acc=0.653, k=500
x_0
x_1
−4
−2
0
2
8
6
4
2
0
−2
−4
−6
−8
0.72
0.64
0.56
0.48
0.40
0.32
0.24
0.16
0.08

Fig. 11. k=500

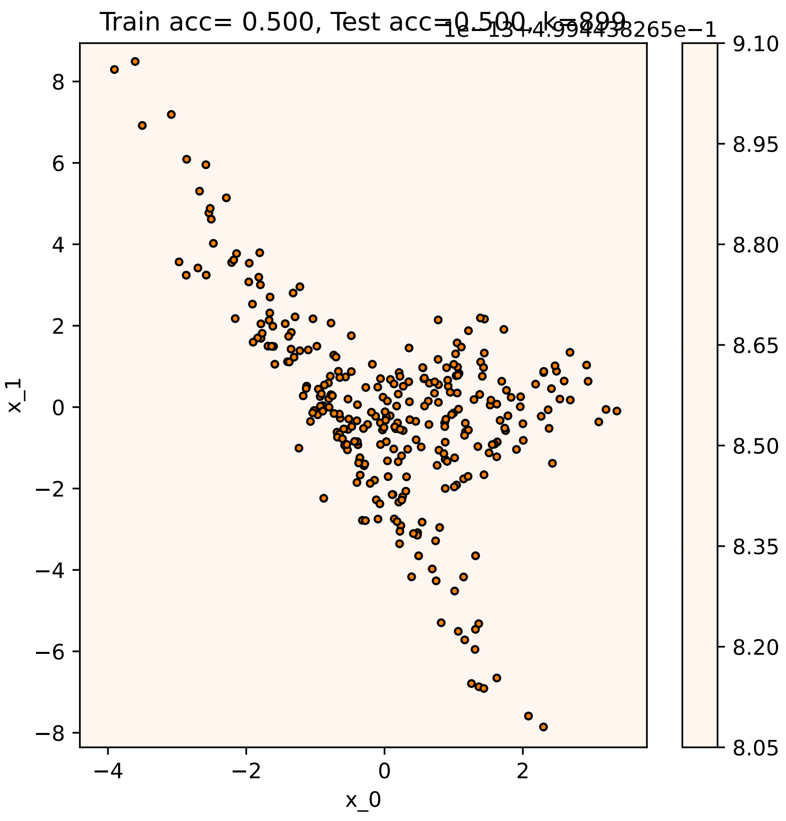
Train acc= 0.500, Test acc=0.500, k=899
1e−13+4.994438265e−1
x_0
x_1
9.10
8.95
8.80
8.65
8.50
8.35
8.20
8.05

Fig. 12. k=899

(b) For small values of k (e.g. $k = 1$), the decision boundary is very detailed and has an irregular shape. This is due to the fact that the model fits the training data very closely, which leads to overfitting – the classifier also learns the noise present in the data.
As the value of k increases (e.g. $k = 5$, $k = 25$, $k = 125$), the decision boundary becomes smoother and more blurred. In this respect, the model generalises better, is less sensitive to small fluctuations in the data and shows greater prediction stability. This means that increasing k to a certain level effectively reduces overfitting.
However, once a certain value of k is exceeded (e.g. $k = 500$ or $k = 899$), the model becomes too simple – underfitting occurs. Then the model begins to classify most points into the dominant class in the set. This is because the decision is based on averaging a very large number of neighbours, often covering almost the entire training dataset.

(2.2) Average accuracies and standard deviations
The average test accuracies (± standard deviation) obtained for five random generations of the data set show the following trend:
For small values of k, the accuracy is lower, which is due to overfitting – the model fits the training data too closely and generalises less well.
The best results are achieved for k in the range of approximately 25–125 (acc =0.917), where the model maintains a balance between complexity and generalization.
For very large values of k, accuracy decreases, indicating underfitting – the model is too simple to capture the structure of the data.
The standard deviation (maximum 0.042) remains low for almost all (exception k = 500) values of k, which means stable and repeatable results regardless of the randomness of data set generation. In other words, the model shows consistent performance, and the observed trends are representative and not random.
Slightly higher standard deviation values for extreme k (e.g., 500) suggest that when the model is misfit, its performance becomes more sensitive to the data distribution. This is probably due to the fact that with such a large k, small changes in object distribution can significantly affect the decision rule.

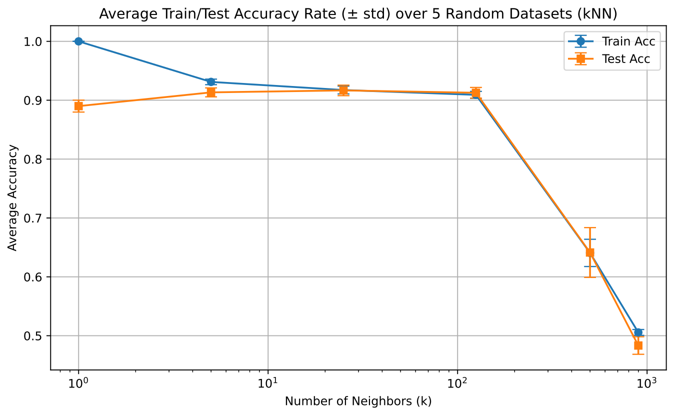

Fig. 13. Comparison of accuracies for different k

3 Quadratic/Linear discriminant analysis

(3.1) For a binary classification problem ($K = 2$), we use Bayes' rule:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{2} \pi_l f_l(x)}$$

where for QDA/LDA we assume that $f_k(x)$ follows a normal distribution:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right)$$

Discriminant Function

We classify observation x into class 1 if:

$$P(Y = 1|X = x) > P(Y = 2|X = x)$$

which is equivalent to:

$$\pi_1 f_1(x) > \pi_2 f_2(x)$$
$$\log(\pi_1 f_1(x)) > \log(\pi_2 f_2(x))$$
$$\delta_1(x) > \delta_2(x)$$

where the discriminant function $\delta_k(x)$ has the form:
For QDA (different covariance matrices):

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k$$

For LDA (common covariance matrix $\Sigma_1 = \Sigma_2 = \Sigma$):

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

Decision Boundary

For QDA
The decision boundary is defined by:

$$\delta_1(x) - \delta_2(x) = 0$$

Substituting the discriminant functions:

$$\left[-\frac{1}{2}\log|\Sigma_1| - \frac{1}{2}(x - \mu_1)^T \Sigma_1^{-1}(x - \mu_1) + \log \pi_1\right] - \left[-\frac{1}{2}\log|\Sigma_2| - \frac{1}{2}(x - \mu_2)^T \Sigma_2^{-1}(x - \mu_2) + \log \pi_2\right] = 0$$

Expanding the quadratic terms: (A covariance matrix is symetric so

$$x^T \Sigma_k^{-1} \mu_k = \mu_k^T \Sigma_k^{-1} x$$

) then:

$$(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) = x^T \Sigma_k^{-1} x - 2x^T \Sigma_k^{-1} \mu_k + \mu_k^T \Sigma_k^{-1} \mu_k$$

We obtain:

$$-\frac{1}{2}x^T(\Sigma_1^{-1}-\Sigma_2^{-1})x + x^T(\Sigma_1^{-1}\mu_1-\Sigma_2^{-1}\mu_2)$$
$$-\frac{1}{2}(\mu_1^T\Sigma_1^{-1}\mu_1-\mu_2^T\Sigma_2^{-1}\mu_2)-\frac{1}{2}\log\frac{|\Sigma_1|}{|\Sigma_2|}+\log\frac{\pi_1}{\pi_2}=0$$

Since there is a quadratic term $x^T(\Sigma_1^{-1}-\Sigma_2^{-1})x$, the decision boundary is quadratic.

For LDA , $\Sigma_1=\Sigma_2=\Sigma$, so the quadratic term disappears:

$$(\Sigma_1^{-1}-\Sigma_2^{-1})=0$$

The decision boundary simplifies to:

$$x^T\Sigma^{-1}(\mu_1-\mu_2)-\frac{1}{2}(\mu_1+\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)+\log\frac{\pi_1}{\pi_2}=0$$

This can be written as:

$$w^Tx+b=0$$

where:

$$w=\Sigma^{-1}(\mu_1-\mu_2)$$
$$b=-\frac{1}{2}(\mu_1+\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)+\log\frac{\pi_1}{\pi_2}$$

Since this is a linear equation in x, the decision boundary is linear.

Summary

- **QDA**: The decision boundary is quadratic because the covariance matrices are different for each class, introducing the quadratic term $x^T(\Sigma_1^{-1}-\Sigma_2^{-1})x$
- **LDA**: The decision boundary is linear because the common covariance matrix causes the quadratic term to vanish, leaving only linear terms

(3.2) (a) Model LDA catches only main linear pattern, with accuracy = 0.907

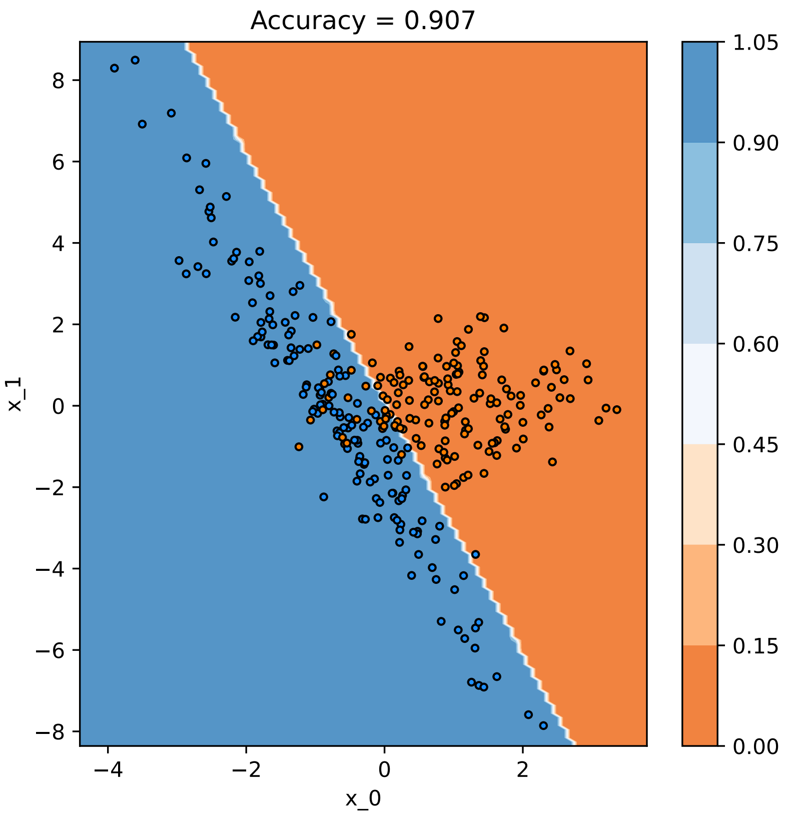

Fig. 14. Decision boundary for Model LDA

(b) Model QDA catches more advanced patterns, with accuracy = 0.923

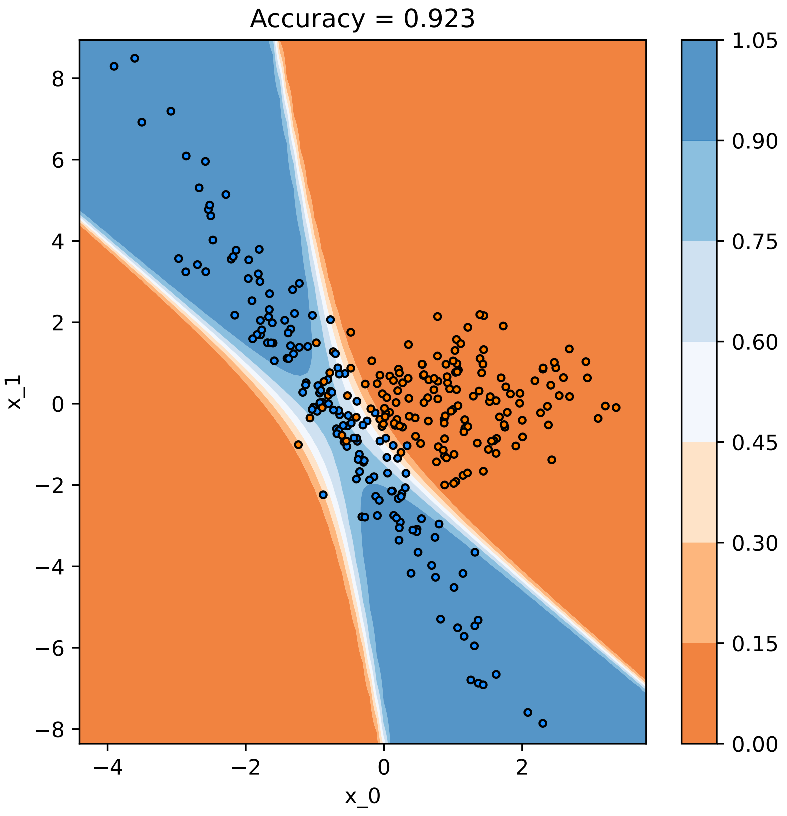

Fig. 15. Decision boundary for Model QDA

(3.3) For the synthetic dataset (make-dataset1), the Quadratic Discriminant Analysis (QDA) achieved higher performance with an average accuracy of 0.923 ± 0.013, compared to 0.901 ± 0.008 for Linear Discriminant Analysis (LDA).
On the Breast Cancer dataset, both methods performed very well, with LDA (0.961 ± 0.015) slightly outperforming QDA (0.942 ± 0.016). Here, the features are more linearly separable, which favors the simpler LDA model. Moreover, QDA showed higher variance, suggesting it may overfit due to its greater flexibility.

4 Method comparison

(4.1) (a) Split the dataset into a training set and a test set. The test set will be used only once at the end to evaluate the final model's generalization performance.

(b) Within the training set, perform k-fold cross-validation (here we used 5-fold CV => 80% for training and 20% for validating): Divide the training data into k folds.
For each candidate value of the hyperparameter (max-depth or n-neighbors), train the model on (k-1) folds and evaluate it on the remaining fold.
Repeat this process k times and compute the average validation accuracy across folds.

(c) Select the hyperparameter value that yields the highest average cross-validation accuracy on the training set.
For Decision Trees, we select the max-depth with the best CV accuracy.
For K-Nearest Neighbors, we do the same for n-neighbors.

(d) Retrain the final model using the entire training set and the tuned hyperparameter. This model is then evaluated once on the test set to estimate its generalization performance.

(4.2) For the synthetic dataset (make-dataset1), the cross-validation tuning procedure indicates that the optimal tree depth is most frequently 4, while the optimal number of neighbors for kNN is typically 25. These configurations consistently yield the highest validation accuracy.
Across the five dataset generations, the tuned decision tree demonstrated stable performance, with an average test accuracy of approximately 0.908 (the average cross-validation accuracy also about 0.9) and a standard deviation of about 0.014. The kNN method produced very similar results, often performing slightly better, with a mean accuracy of around 0.917 and a standard deviation of about 0.009 (the average cross-validation accuracy about 0.91).
Overall, both methods achieved comparable scores and exhibited low variability between runs, indicating a high degree of stability.

For the Breast Cancer dataset, the optimal decision tree depth varied across runs, with the best values being [6, 6, 4, 2, 6]. The kNN results were somewhat more consistent and showed greater variability than in the synthetic dataset, with the best numbers of neighbors being [1, 25, 5, 5, 5]. This pattern reflects a higher sensitivity of the models to data sampling.
The tuned decision tree achieved a mean test accuracy of 0.92 with a standard deviation of 0.01, while the tuned kNN reached a mean test accuracy of 0.918 with a standard deviation of 0.019.
These results indicate strong and consistent generalization performance for both models on this real-world dataset, with the Decision Trees method showing slightly better accuracy.

Below we present performance of tuned methods on the same dataset:

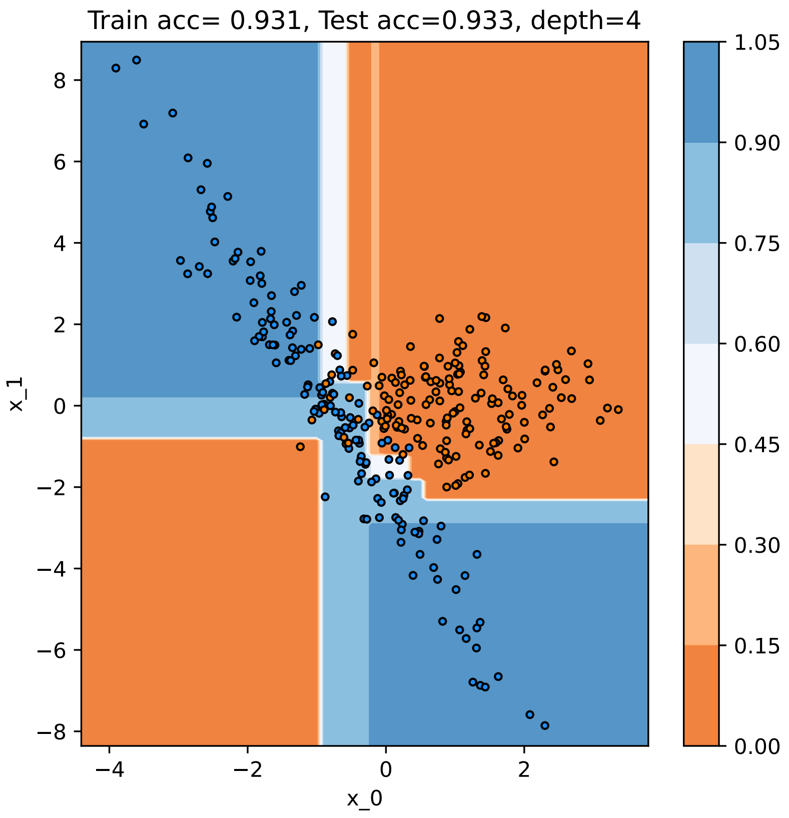

Fig. 16. Decision boundary for dt depth = 4

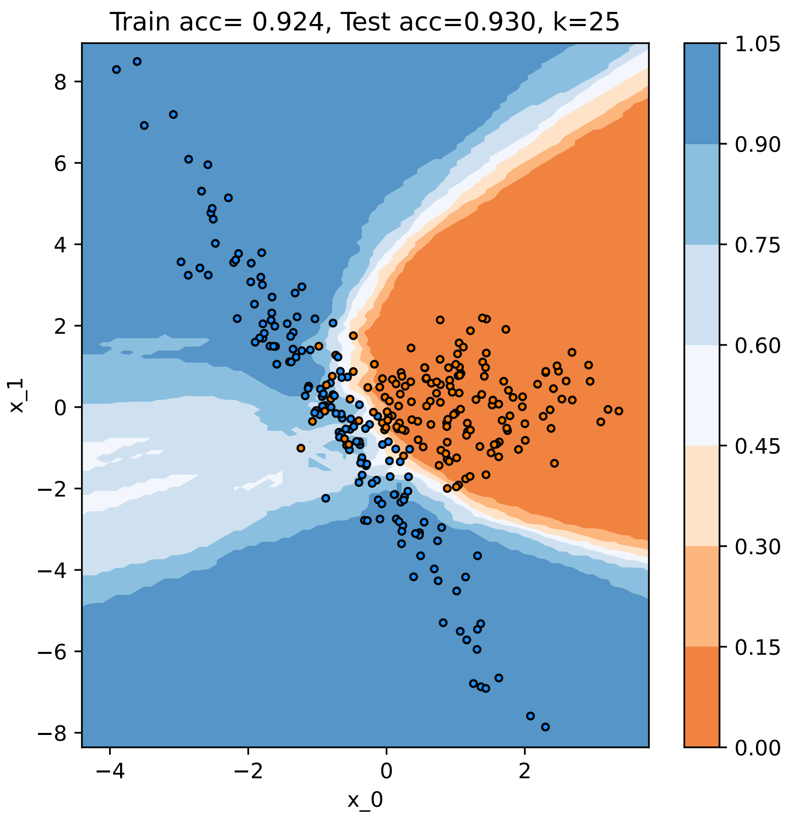

Fig. 17. Decision boundary for kNN k = 25

(4.3) Then we have comparison of the methods over five datasets along with the standard deviation on both datasets.

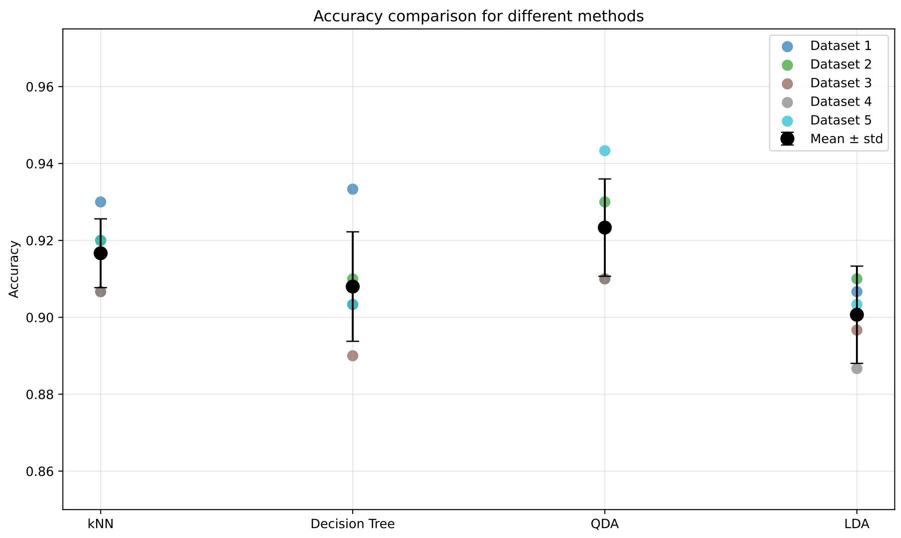

Fig. 18. Method Comparison on synthetic data

The QDA model achieved the best performance with an accuracy of 0.923 ± 0.013, closely followed by the tuned kNN model (0.917 ± 0.009). The tuned Decision Tree performed slightly worse (0.908 ± 0.014), while LDA showed the lowest accuracy (0.901 ± 0.013). Overall, all methods seem to perform well.

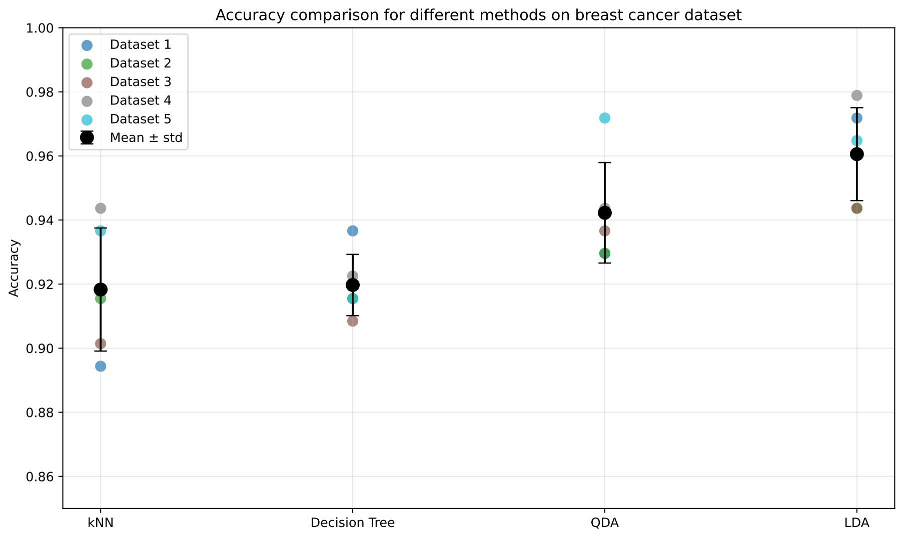

Fig. 19. Method Comparison on real data

Among the evaluated models, LDA achieved the highest performance with an accuracy of 0.961 ± 0.015, followed by QDA (0.942 ± 0.016). The tuned Decision Tree performed slightly lower (0.92 ± 0.01), while the tuned kNN model obtained the lowest accuracy (0.918 ± 0.019). Overall, non-parametric models (LDA, QDA) outperformed the parametric ones (kNN, DT).