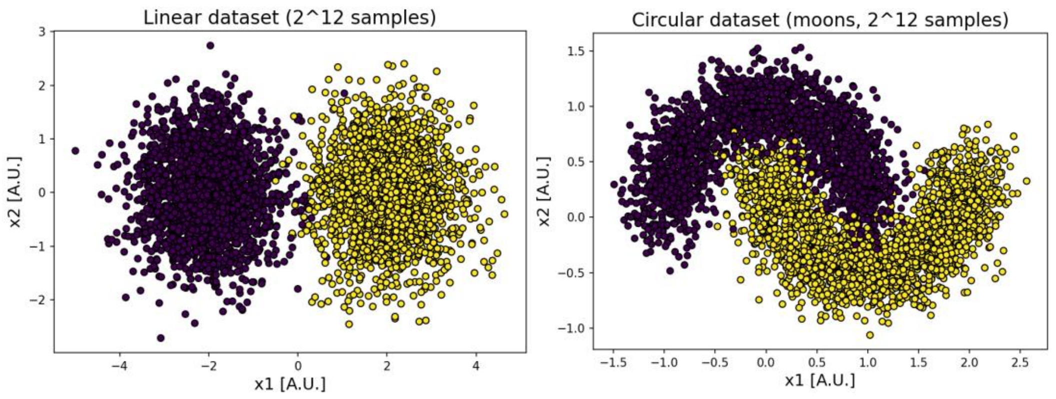
Name : Yael Tsuk ran
Date: 4.12.25

Machine Learning & Neural Networks for Neuroscience -HM2

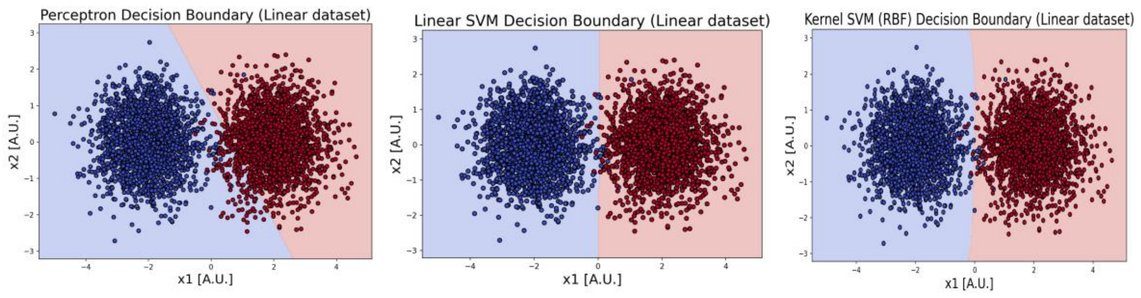
Question 1 & 2:



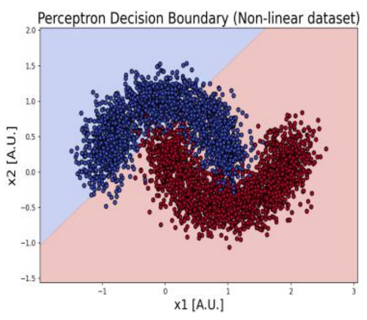
Question 3:

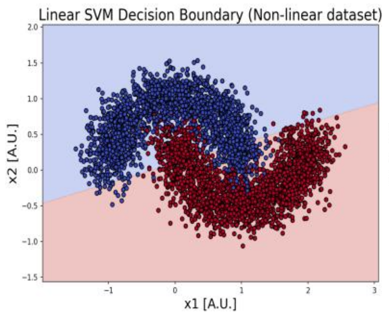
	Linear data	Circular data
Perceptron model	0.9622	0.7817
Linear SVM	0.9963	0.8524
Kernel SVM (RBF kernel)	0.9951	0.9683

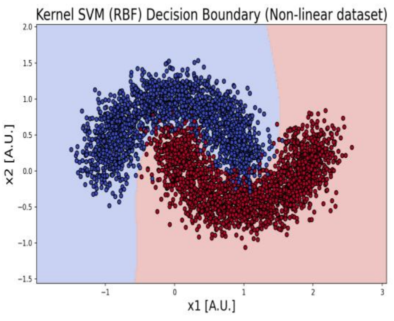
In the linear dataset, all models perform well because the classes are linearly separable. The Perceptron and Linear SVM reach very high accuracy, and the Kernel SVM offers no real advantage.
In the non-linear moons dataset, the Perceptron and Linear SVM struggle because they can only learn straight boundaries, while the Kernel SVM performs much better since it can model the curved structure of the data. Even so, the linear models still achieve moderate accuracy because parts of the moons are roughly separable.



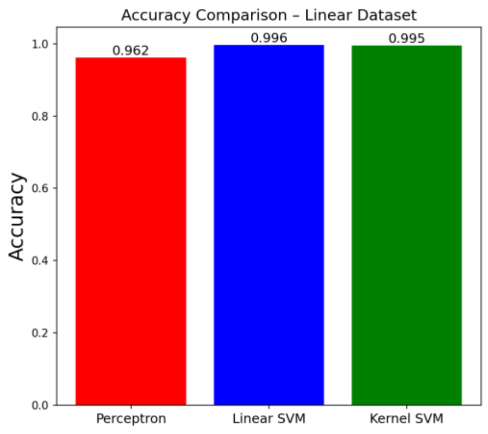
Name : Yael Tsuk ran
Date: 4.12.25

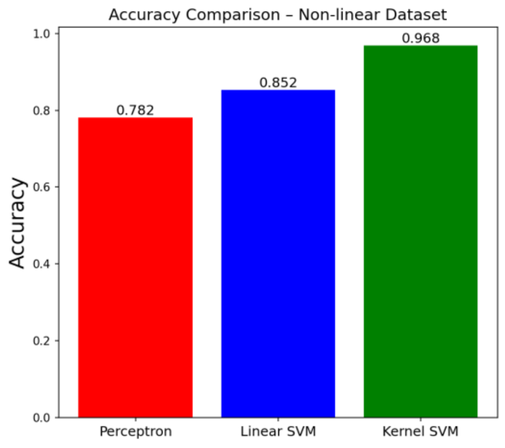




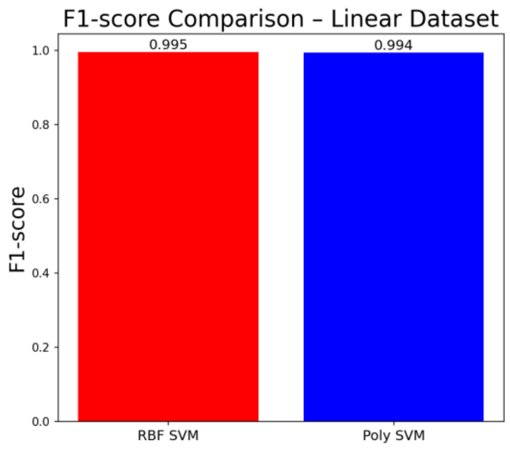


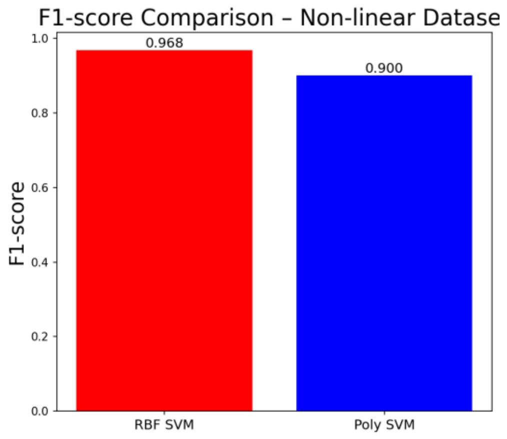
The decision-boundary plots show the same pattern as the accuracies.
On the linear dataset, all three models learn a straight boundary that separates the classes well. On the non-linear moons dataset, the Perceptron and Linear SVM still produce a straight line and misclassify many points, while the Kernel SVM learns a curved boundary that matches the moon shape, explaining its much higher accuracy.



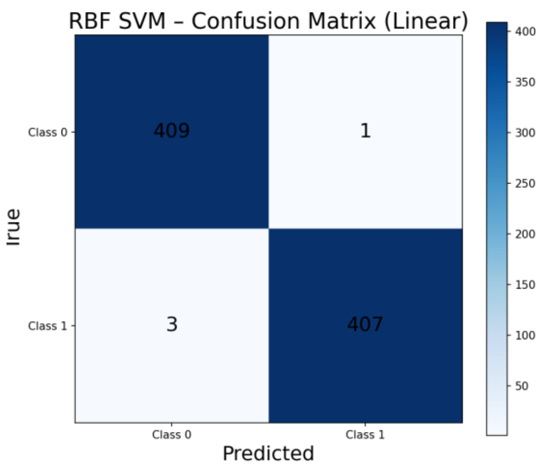


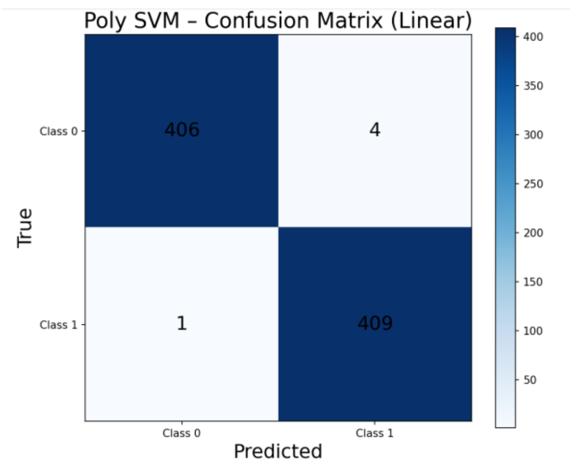
<u>Question 4:</u>

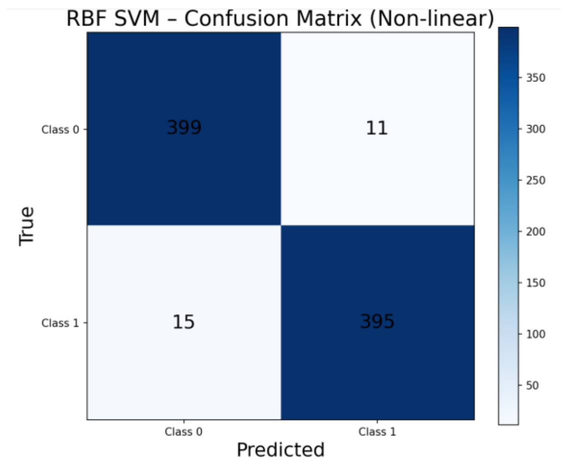


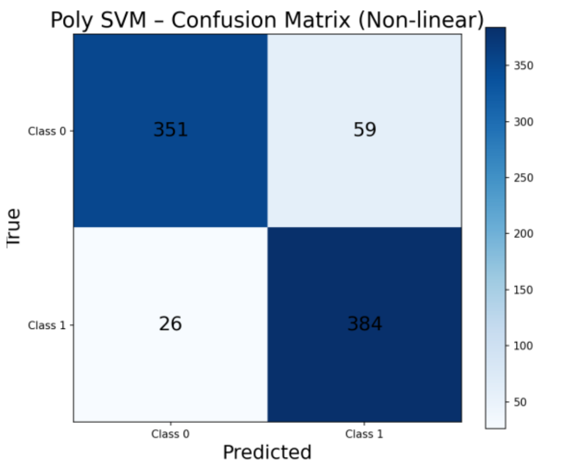


Name : Yael Tsuk ran
Date: 4.12.25

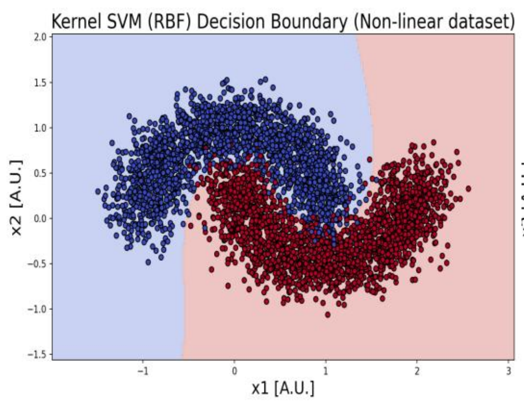


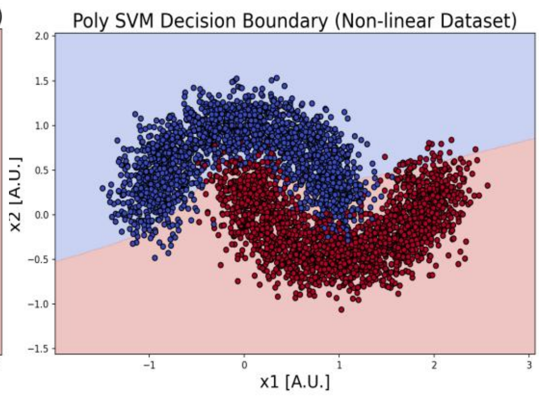






The results show that RBF and Polynomial kernels behave similarly on linear data, both achieving near-perfect F1-scores and almost identical confusion matrices. However, on the non-linear moons dataset, the RBF kernel outperforms the Polynomial kernel, achieving a higher F1-score (0.968 vs. 0.900) and fewer misclassifications. My suggestive explanation to that is because it creates flexible, local decision boundaries that can follow the curved "moons" structure. In contrast, the Polynomial kernel forms more global shapes that cannot adapt well to the complex, intertwined class boundaries. Therefore, RBF is the more suitable kernel for non-linear data such as the moons data.

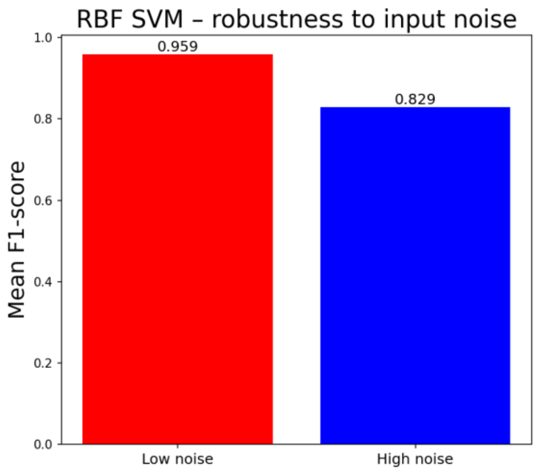




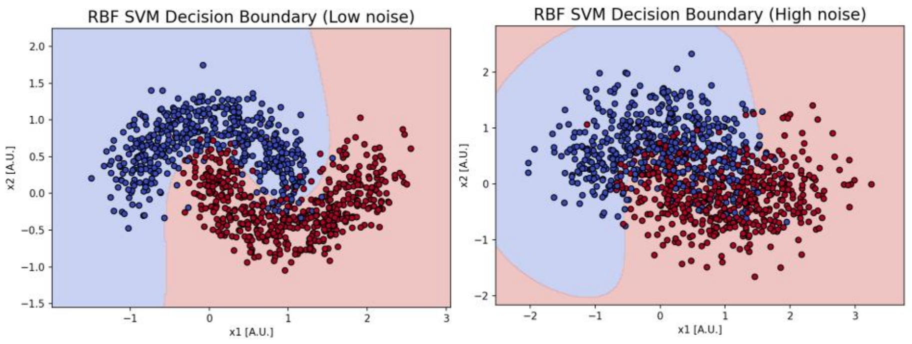
Name : Yael Tsuk ran
Date: 4.12.25

Question 5:

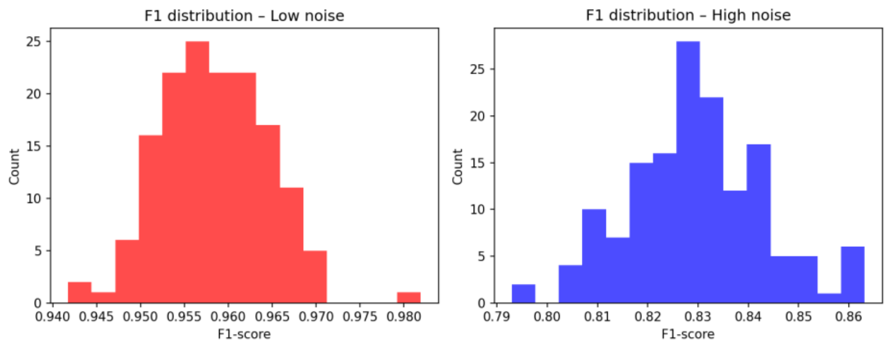
A :



With low input noise, the RBF SVM stayed highly accurate (mean F1 = 0.959) because small distortions do not change the local structure the kernel relies on. Under high noise, the F1 dropped to 0.829, since the noise shifts points far enough to blur the class boundaries, making the model less reliable.



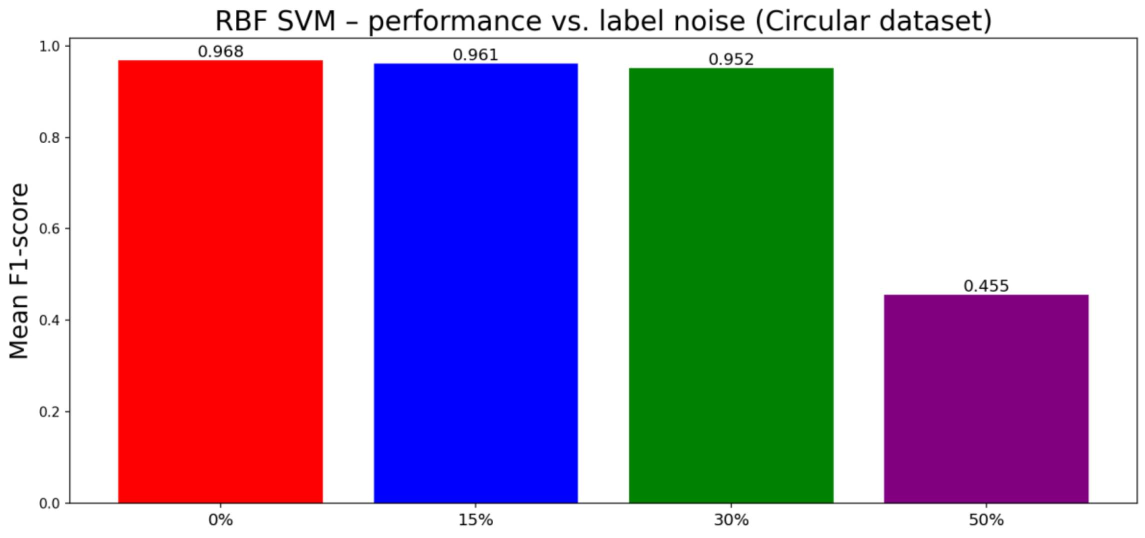
Low noise: The decision boundary stays smooth and closely follows the shape of the two moons, since the noise barely distorts the data.
High noise: The boundary becomes more irregular and less precise because the strong noise mixes the classes and makes separation harder.



Name : Yael Tsuk ran
Date: 4.12.25

Low noise: The F1-scores cluster tightly around ~0.95, showing that the model remains stable and accurate under small perturbations.
High noise: The distribution is wider and shifted lower (~0.82), indicating reduced stability and accuracy when the input is heavily corrupted.

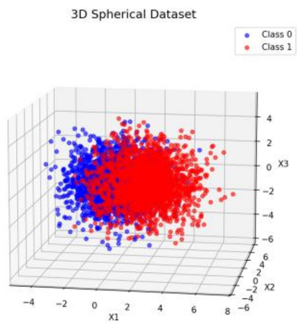
B:

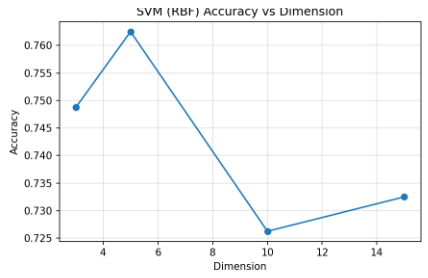


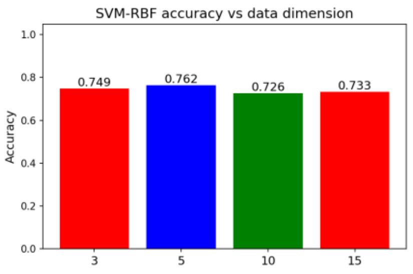
To test the RBF SVM's robustness to label noise, randomly flipped 15%, 30%, and 50% of the labels in the training set and trained a new model each time (repeating the process 50 times to average out randomness). The results show that the classifier is very robust to mild label corruption: the F1-score drops only slightly from 0.968 at 0% noise to 0.961 at 15% and 0.952 at 30%. However, at 50% label noise, the structure of the classes is essentially destroyed, and the performance collapses to an F1 of 0.455.
Overall, the RBF SVM can tolerate moderate label noise, but once too many labels are incorrect, the model can no longer learn a meaningful boundary.

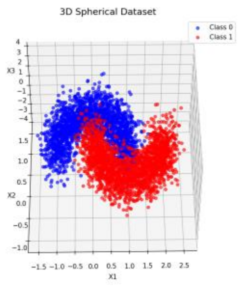
Name : Yael Tsuk ran
Date: 4.12.25

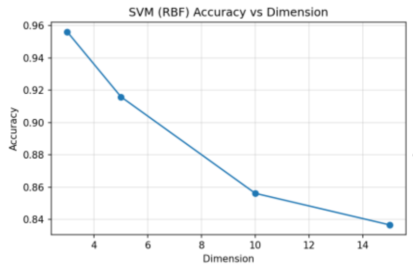
<u>C:</u>

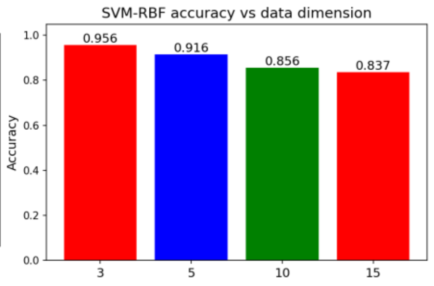












In my first attempt I generated two Gaussian ball clusters in different dimensions. The accuracy stayed almost the same because increasing the dimension did not really change the problem: the two classes differed only in one coordinate, and all extra dimensions had the exact same distribution for both classes. Therefore, the model had no reason to perform worse as dimension increased.
the next thing I tried was to creating 2D moon shaped data and then adding extra random dimensions. In this case only the first two dimensions contain real structure, and the rest make the problem harder. With this dataset the SVM accuracy decreased as the dimension grew. The top figures show that for Gaussian “ball” data, accuracy stays almost constant because higher dimensions do not change the structure of the two clusters.
The bottom figures show that for moon data with added dimensions, accuracy decreases as dimension increases.