schuler_5779_ece_527_report_03

September 16, 2024

1 GMU ECE 527 - Computer Exercise #3 - Report

Stewart Schuler - G01395779
20240919

1.1 Exercise 3.1

3.1.1 Question: How does the SVC class find the maximum-margin classifier?

Answer: The SVC class applies the kernal via the kernel trick to the training data to create a higher dimentional dataset then uses the *dual problem* approach to solve the minimization probelm linearly in the higher dimentional space. When mapped back into the original feature space the decision boundary becomes non-linear (assuming a non-linear kernal was used).

Question: There are two other classes in scikit-learn that may be used to find a maximum-margin classifier: LinearSVC and SGDClassifer. How are these different from SVC, how do they find the maximum-margin classifier, and when would you use these instead of the SVC class?

Answer: *LinearSVC* solves the *primal problem* rather than the dual problem used by *SVC*. Solving this problem involves using matrix operation of size porportional to the number of features. When searching for a linear boundary it is assumed the number of features relativly low. Thus the *primal* approach will be more efficient than applying the kernel trick. The downside being for non-linear data the high dimention data transform must be applied directly which can exponentially increase the number of features making this approach slow.
The *SGDClassifer* implments the same gradient decent algorithm as the *Perceptron* class with the notible change of using the different *hinge* loss function. In practice this function allows the algorithm to work on data that isn't strictly separable. The use case for this type of algorithm would be data that is near separable in a low dimentional feature space, and when there is a relativly small ammount of training data.

3.1.2 Experiments Generating a non-linearly separable dataset and applying linear SVM with the default value of $C = 1$ yields the result shown in *Figure 1.*
The performance of the classifier on the 25% test held out set produces the following results on the training and testing tests.

Dataset	Accuracy
Training	0.946
Testing	0.960

I found it interesting that the test set had a higher accuracy than when evaluating on the training set. This is likely happening becuase with a small dataset the randomly selected test set may not be representive of the true distribution. With a dataset such as the one used for this experiment it is reasonably possible to draw a testing set that **is** linearly separable which could lead to perfect test set accuracy, even when the decision boundary doesn't perfect model the true boundary. To test this I drew 100 different train/test permutations and found the following range of results.

Dataset	Min Acc	Max Acc
Training	0.933	0.973
Testing	0.880	1.000

As expected the accuracy of the classifier on the training test has a lower range of scores because it is larger and more accuately represents the true distribution. Where as the smaller testing set can appear to be prefectly modeled by the classifier or poorly modeled depending on which samples are held out for testing.

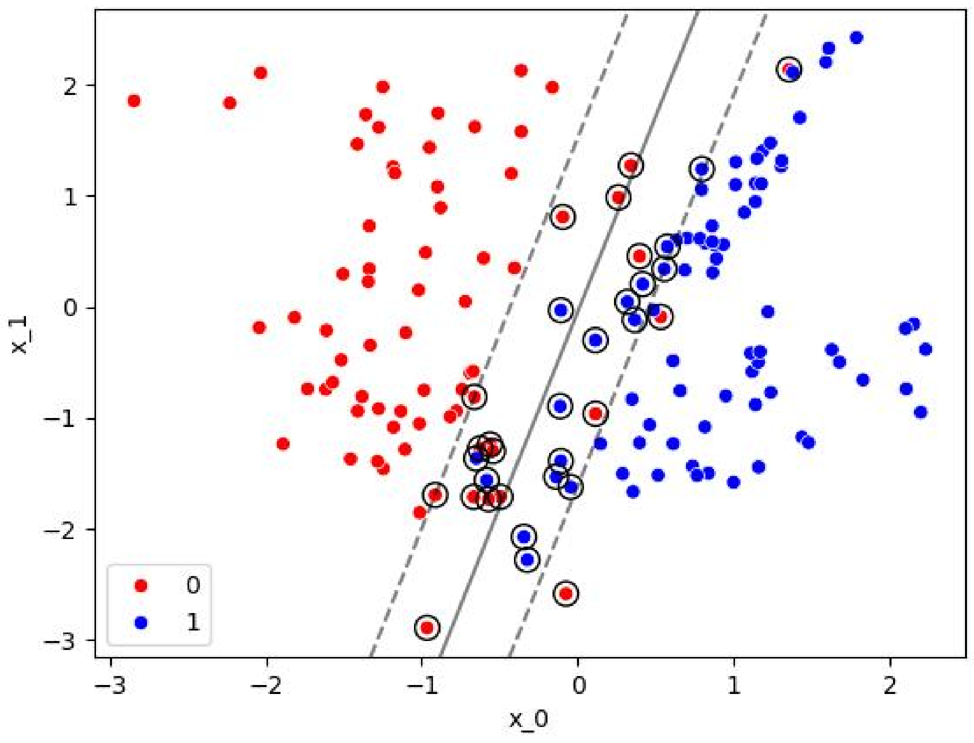

Figure 1. Linear SVM on Dataset

3.1.2 Questions For the dataset in *Figure 1* there were 17 support vectors for class 0, and 16 for class 1. If the data were linearly separable we would expect there to be as few as a single support vector for each class. There are cases where linearly separable classes can have more than one support vector, but for randomly generated data with high data percision it is unlikely to have multiple.

Figure 2 showns the effect chaning the SVM parameter C has on the classification accuracy. I can be clearly seen that some values of C produce higher accuracies for both the training and testing set, when compared to other values. That demonstrates that choosing the correct value of C is important for classification performance.

Figure 2 was created using a single specific train test split. It likely a different dataset permutation would yield a different plot.

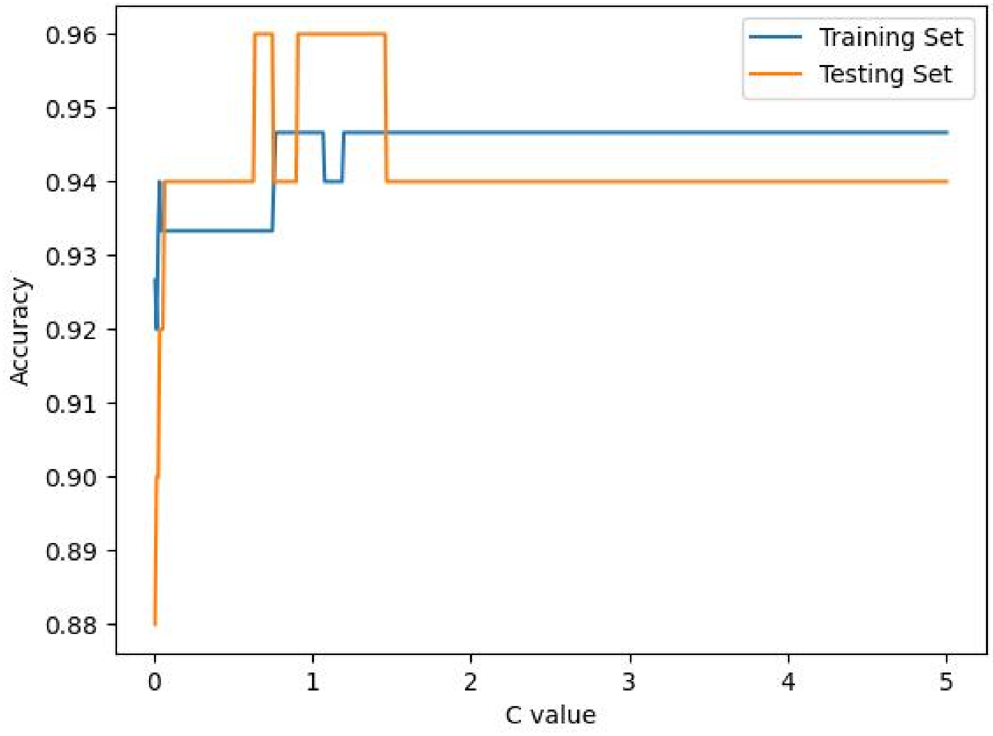

Figure 2. SVM C value

The SVM returns *dual coefficients*, one for each support vector which range between values of -1 and 1. A dual coefficient with a value of $|\alpha_n| < 1$ indicates the corresponding support vector x_n is a *marginal* support vector, that is it lies directly on the margin. A value $|\alpha_n| = 1$ indicates the corresponding support vector x_n is a *non-marginal* support vector. That is it lies off the margin at some distance in the direction of the decision boundary. The dual vectors are bounded by -1 and 1 becuase in the soft margin SVM formulation $|\alpha_n| < C$ and in this case C is chosen as the default value 1.

With the *dual coefficients* being known they can be used to create the decision boundary $g(x)$ via the following relationship.

$$g(x) = \mathbf{w}^T\mathbf{x} + b$$

$$\mathbf{w} = \sum_{SV} y_n \alpha_n \mathbf{x_n}$$

3.1.3 Experiments Next consider a SVM with a polynomial kernel function. The plots of the decision boundaries with a value of $C = 1$ can be seen in *Figure 3*. The results of the classifiers with varying polynomial orders are recorded in the following table.

Order	# y_0 SV	# y_1 SV	Training Acc	Testing Acc
2	70	70	0.593	0.540
3	30	30	0.913	0.920
4	64	66	0.586	0.560
5	40	40	0.826	0.840

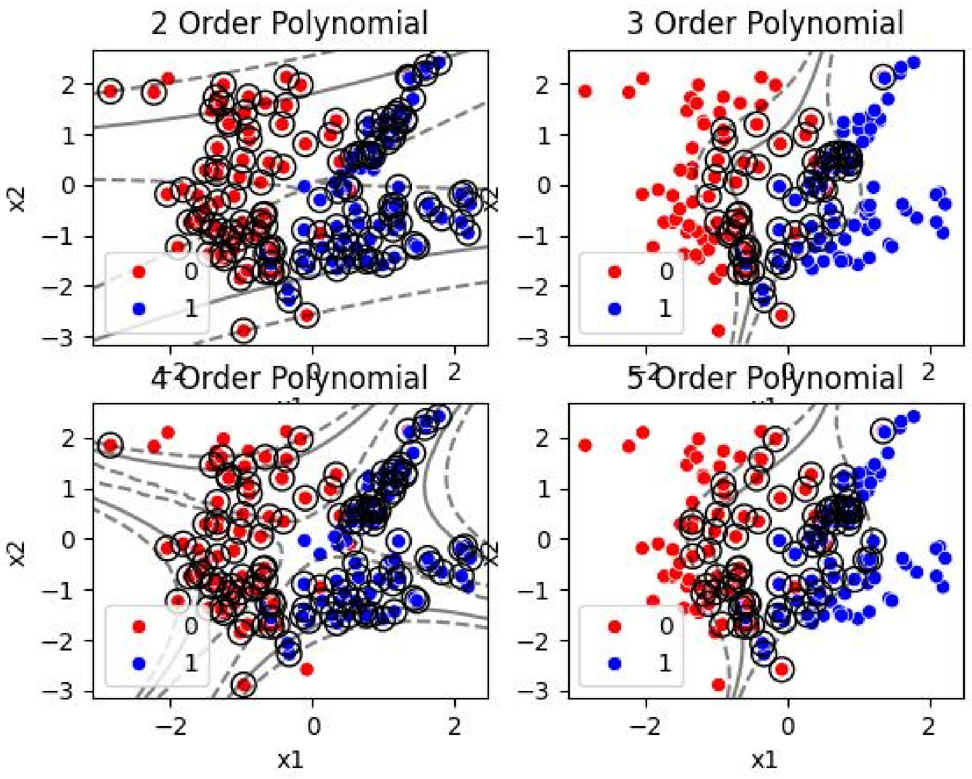

Figure 3. Differing Polynomial Orders

Clearly from the classifier accuracy the 3^{rd} order polynomial performs the best. The next investigation was to see if 3^{rd} order polynomial classifier out performing the other three was a result of the default C value. *Figure 4* shows the results of chaning the C value for each order. While there is some change in the result accuracy, the changes coming from the C value are nowhere near the change in accuracy as a result of order.

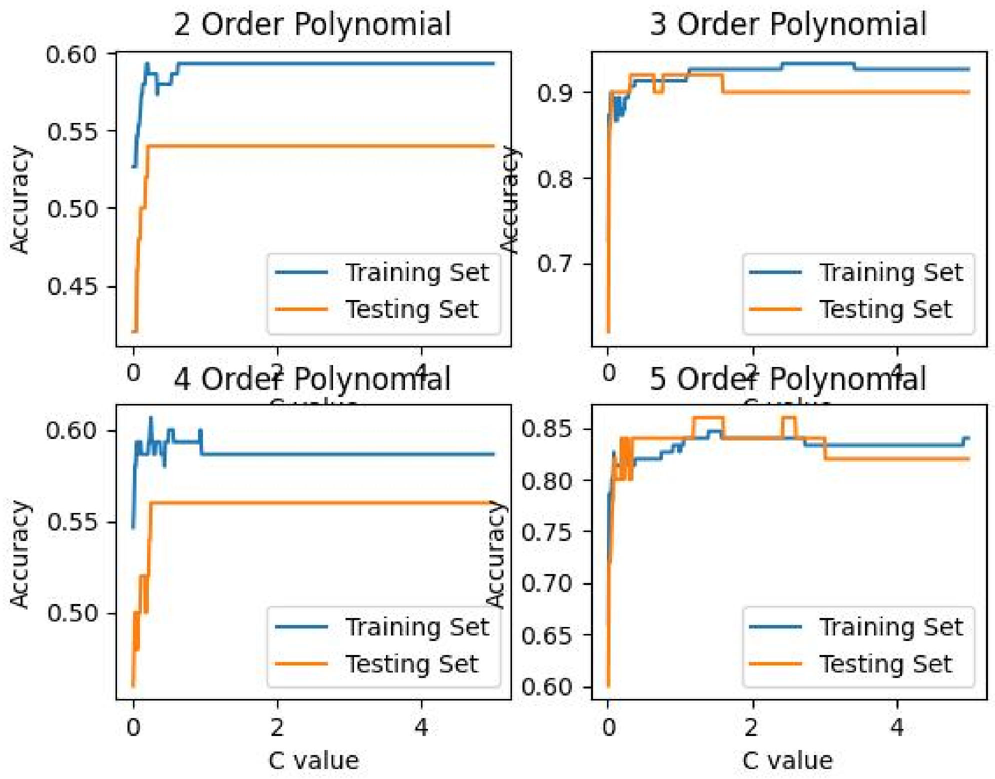

Figure 4. Differing Polynomial Orders and C Value

Even the best performing polynomial classifier performs worse than the previously discussed linear classifier. It isn't too suprising given the randomly generated dataset being used for these experiments. While it is non-linearly separable it is very close to being which is why it would make sense the linear classifier does such a good job.

Lastly we consider the RBF kernel function. This kernel is parameterized by γ. The SVC class has two default values for γ, *auto* and *scale* computed by the following equations.

$$\gamma_{auto} = \frac{1}{d}$$
$$\gamma_{scale} = \frac{1}{d * var(\mathbf{x})}$$

Decision boundary plots for the γ_{auto} and γ_{scale} approaches are shown in *Figures 5* and *6* respectively. It can be seen that they produce very similar decision boundaries. And perform with an accuracy nearly equivliant to that of the linear classifer.

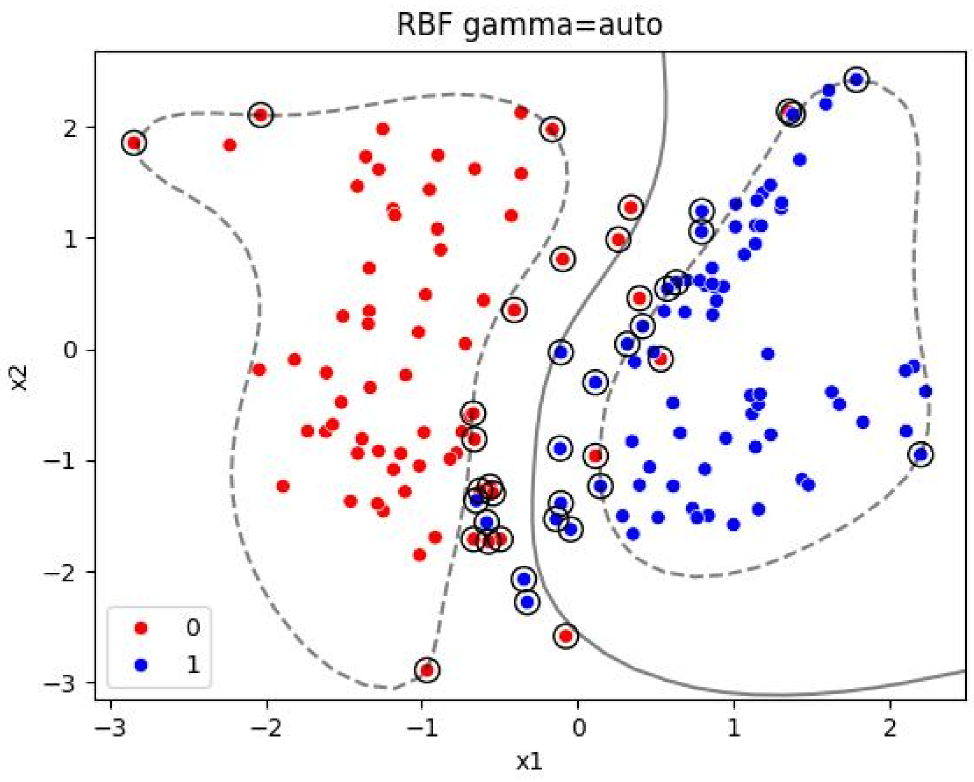

Figure 5. Decision Boundary RBF auto

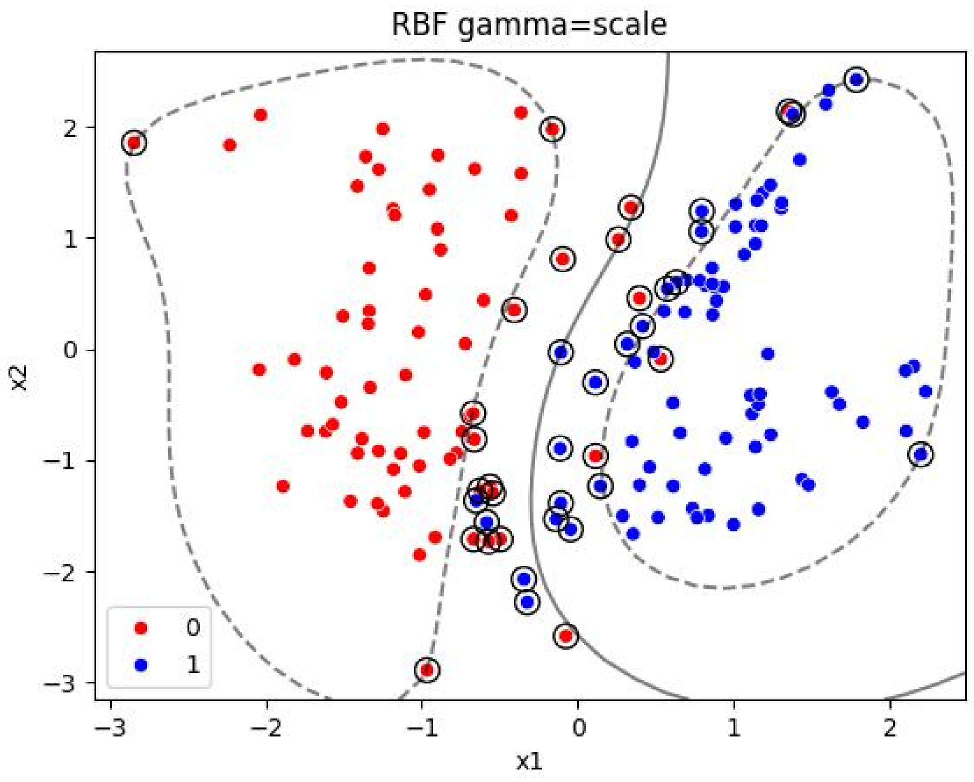

Figure 6. Decision Boundary RBF scale

Next to consider the effect of γ on the accuracy of the classifier we again plot training and testing accuracy as a function of the hyperparameter, in the case of RBF, γ. It can be seen that the *auto* and *scale* values of γ don't produce the best result accuracy result. Being out performed by $\gamma = 3.5$ on both the training and testing set.

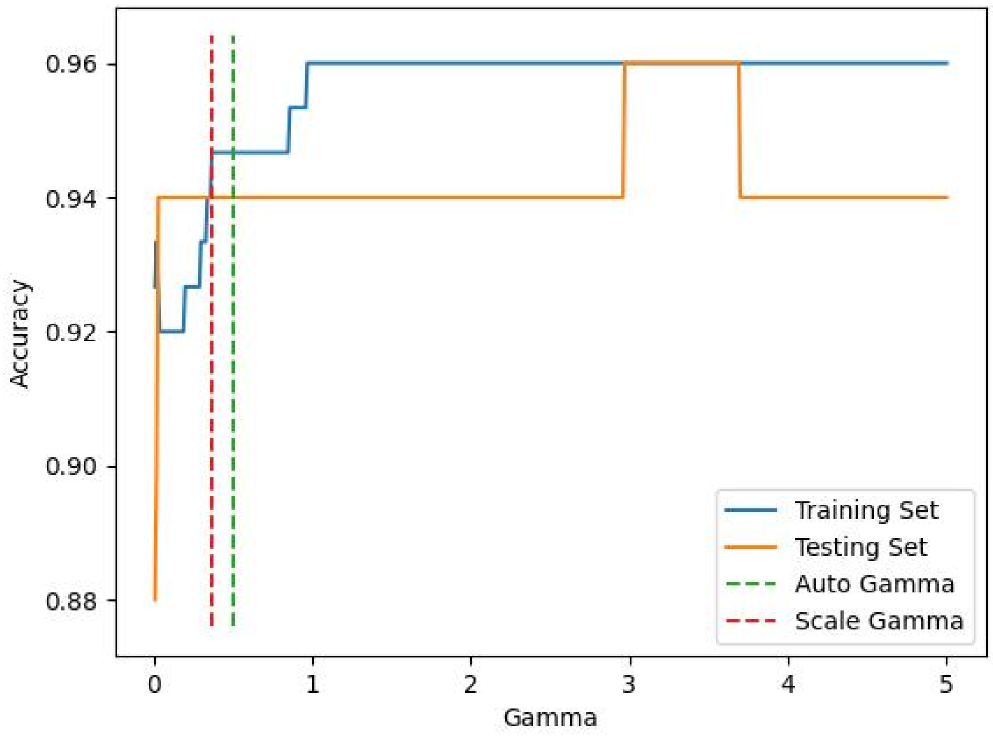

Figure 7. Decision Boundary RBF scale

If we consider the $\gamma = 3.5$ case in *Figure 8* it can be seen that even though the accuracy is the highest for both training and testing, the boundary is rough and the number of support vectors drastically increases. This is an indication that this classifier is overfit to the data, and while it performs well on this specific testing set it will likely not acheive the same performance on a different test set permutation.

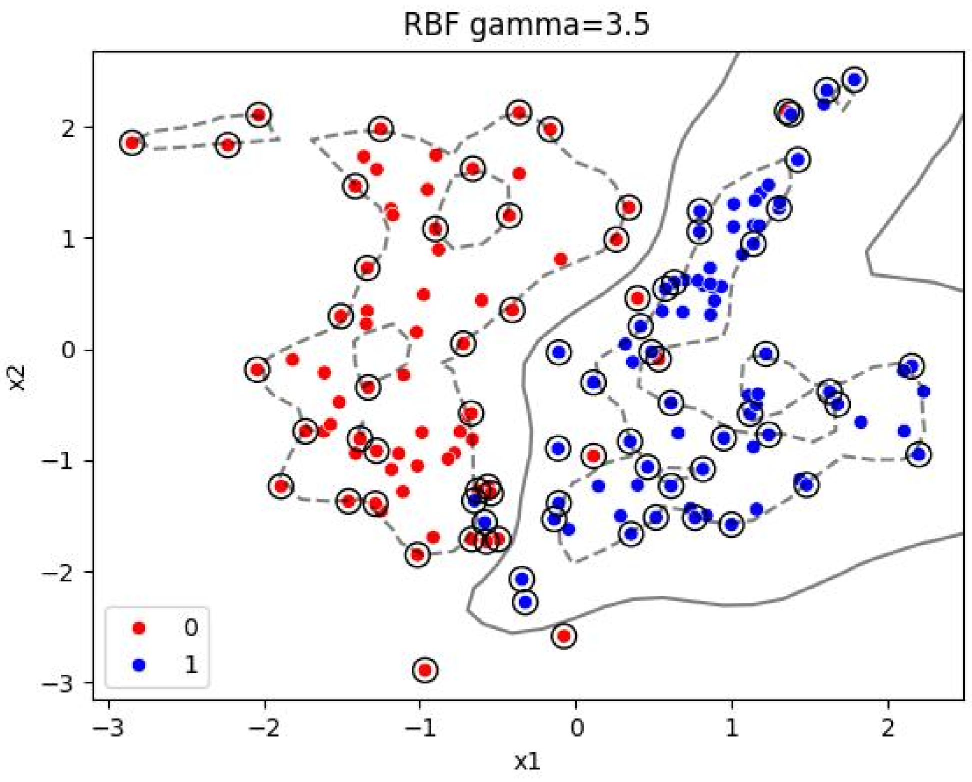

Figure 8. Decision Boundary RBF scale

The results of the RBF investigation are summarized in tabular form below.

Gamma	# y_0 SV	# y_1 SV	Training Acc	Testing Acc
auto	21	20	0.946	0.940
scale	20	20	0.946	0.940
3.5	64	66	0.960	0.960

1.2 Exercise 3.2

The next dataset underconsideration is the *Wisconsin Breast Cancer* dataset. For this dataset each feature is on an independent scale. As can be seen by *Figure 9* when plotting the *log10* mean and variance of each feature the features can vary by multiple orders of magnitude.

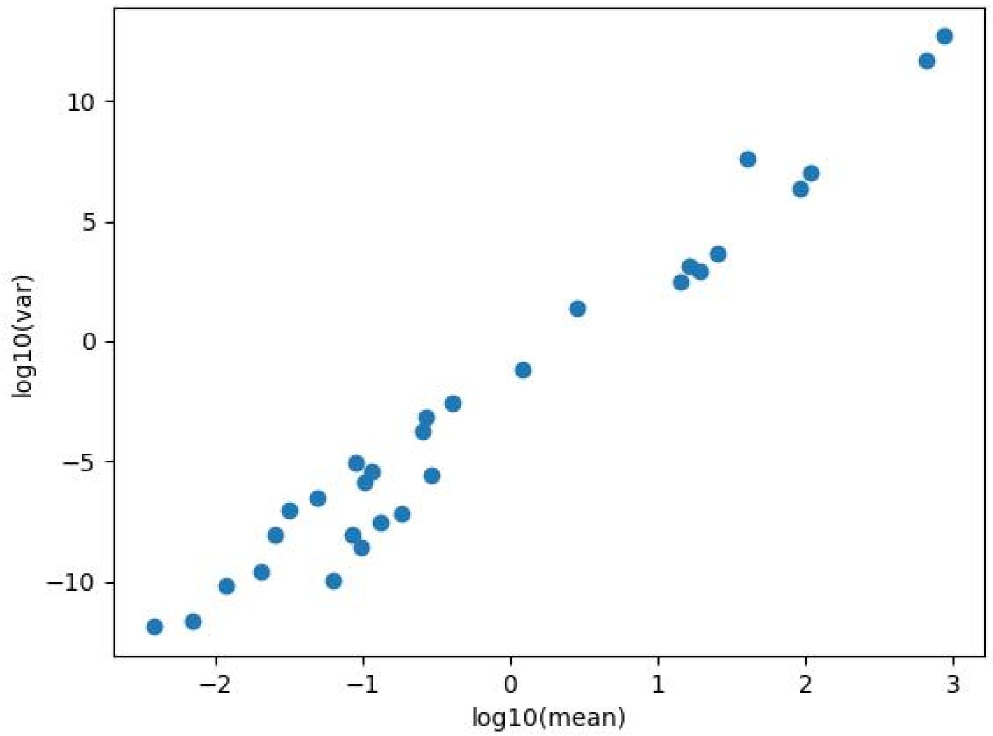

Figure 9. Feature Log Mean Var

SVM performs better when all the features use the same scale. For this reason we will apply gausian normalization to each feature, which transforms each feature from the set to have a mean of 0 and a variance of 1. The transform is applied to each feature using the following equation.

$$\mathbf{x_{norm}} = \frac{\mathbf{x} - \mathbf{mean}(\mathbf{x})}{\mathbf{var}(\mathbf{x})}$$

The computed mean and variance are derived only from the training data as the held-out testing data should have no influence on how the classifier is designed. Before running classification on the testing set it should be transformed using the same mean and variance values computed on the training data.

Figure 10 shows the result of this normalization transform on the training and testing sets. As expected all the training features have a mean of 0 and variance of 1. The testing features do not, however if we note the linear scale of this plot verses the log scale of *Figure 9*, all the features are much closer in scale than before the transform.

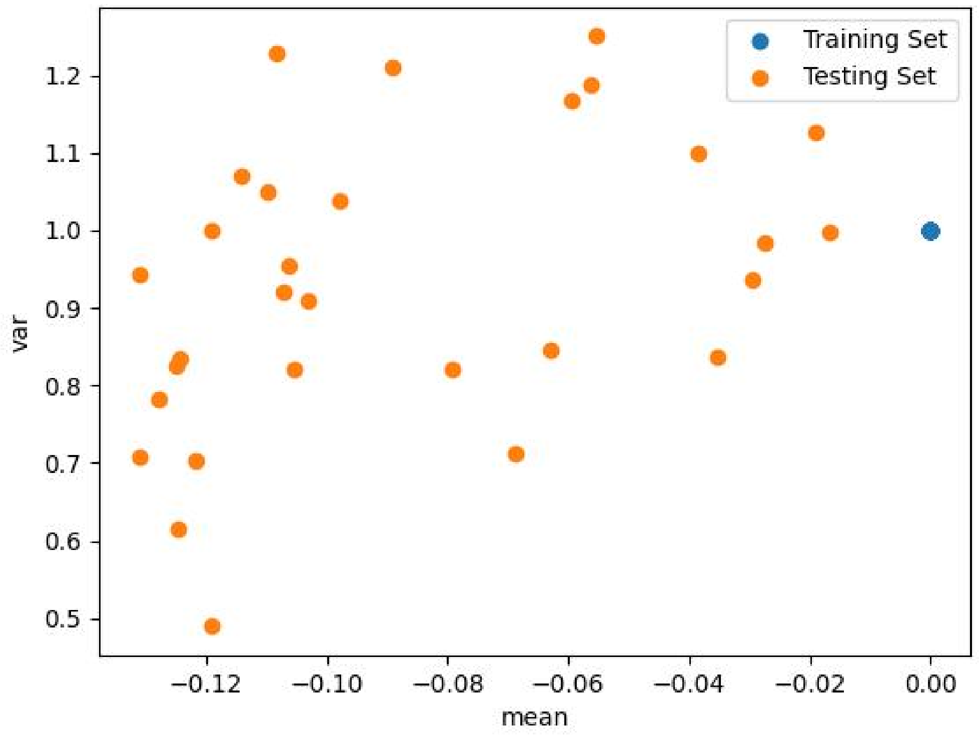

Figure 10. Feature Mean Var Post Transform

3.2.1 Experiments Applying a linear SVM classifier to the dataset with various values of C produces the results shown in the below table. It can be seen that changing C has a large impact on the number of support vectors, but little impact on the accuracy of the classifer on the held-out test set.

C	# y_0 SV	# y_1 SV	Training Acc	Testing Acc
0.01	51	50	0.974	0.972
0.1	24	25	0.988	0.972
1	16	17	0.992	0.979
10	13	14	0.992	0.979
100	13	12	0.997	0.979

3.2.2 Experiments Next we applied the RBF kernel to the SVM classifier. In this case since the training data is normalized to have a variance of 1 there is no difference between γ_{auto} and γ_{scale}. With the default values of $\gamma = \gamma_{scale} = \frac{1}{30}$ and $C = 1$, the classifier produces the results shown in the below table.

C	γ	# y_0 SV	# y_1 SV	Training Acc	Testing Acc
1	0.033	55	54	0.988	0.979

To visualize the effect of changing γ on the classifier *Figure 11* plots accuracy vs γ. From *Figure 11* it looks like the default values of RBF do the best job. Even though training accuracy get higher and γ increases the divergence between training and testing accuracy means that data is being overfit to the training set. The default values produces the highest accuracy on the test set while performing similarly on both sets. As such the default combination produces the best version of the RBF kerneled classifier.

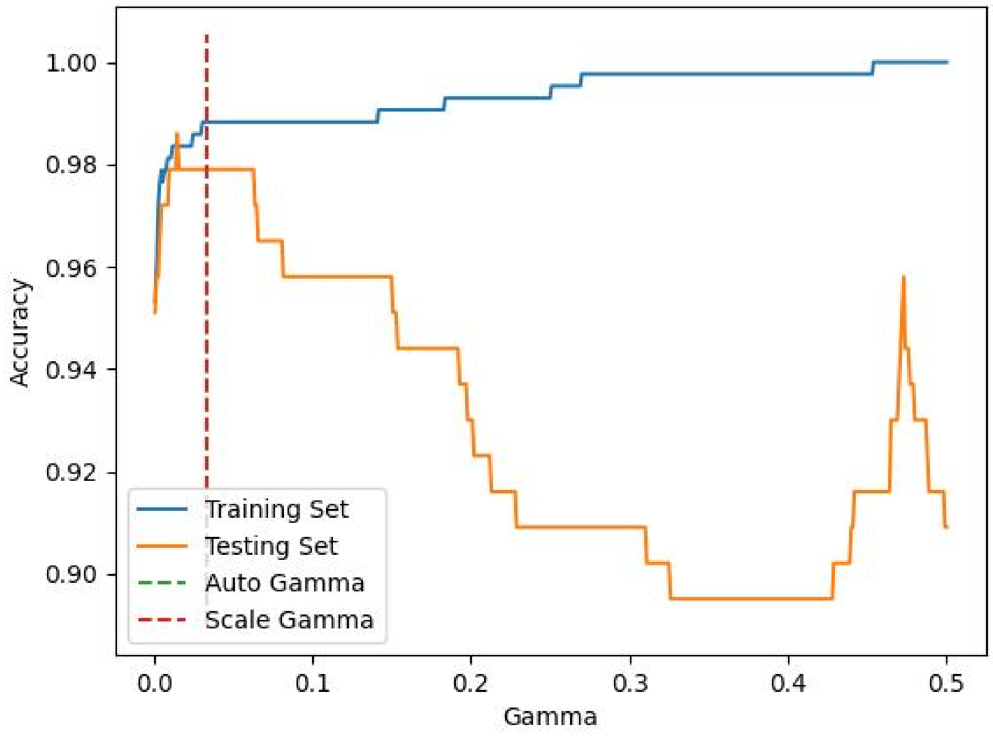

Figure 11. Accuracy vs γ

3.2.3 Experiments Finally we reduce the dataset to only two features deemed important. Using only these two features we apply the three types of kernels we've been looking at previously and produce the following results for the linear, 3^{rd} order polynomial, and RBF kernels. Shown in *Figures 12-14* respectively.

Type	# y_0 SV	# y_1 SV	Training Acc	Testing Acc
Linear	33	33	0.955	0.958
3rd Poly	54	54	0.908	0.860
RBF	39	34	0.962	0.937

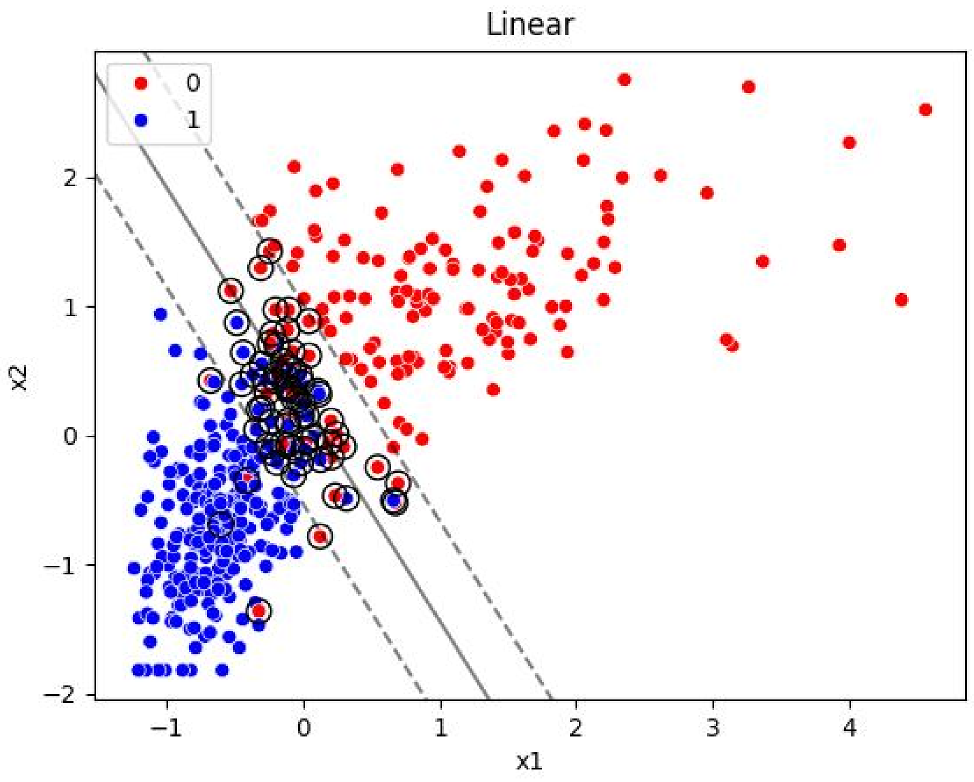

Figure 12. Two Feature Linear

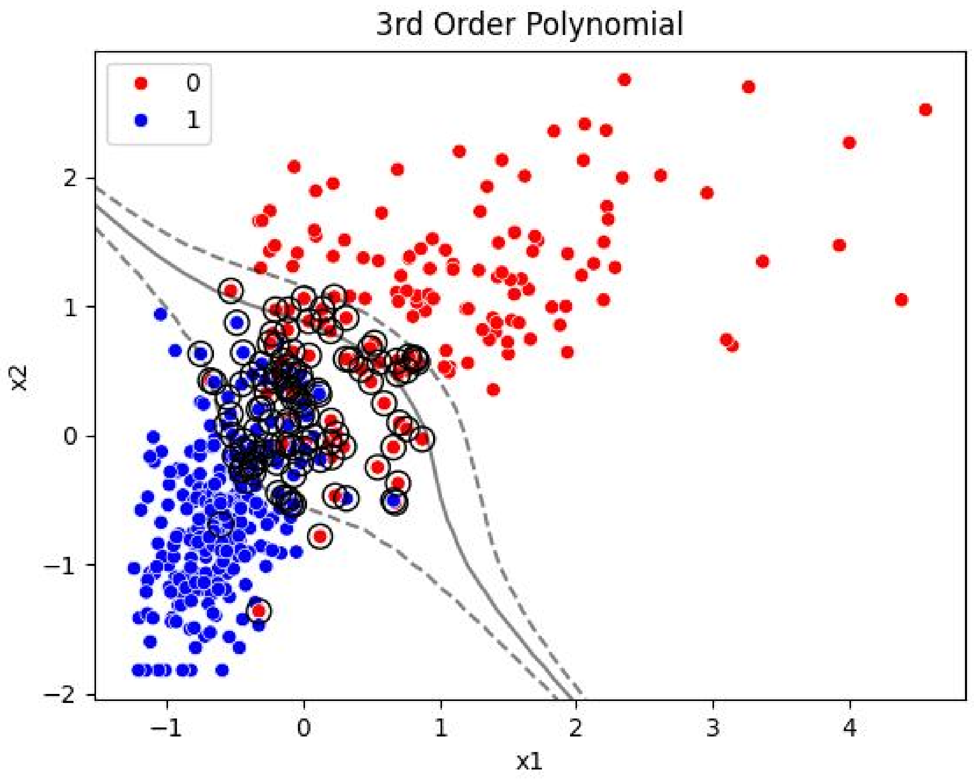

Figure 13. Two Feature Polynomial

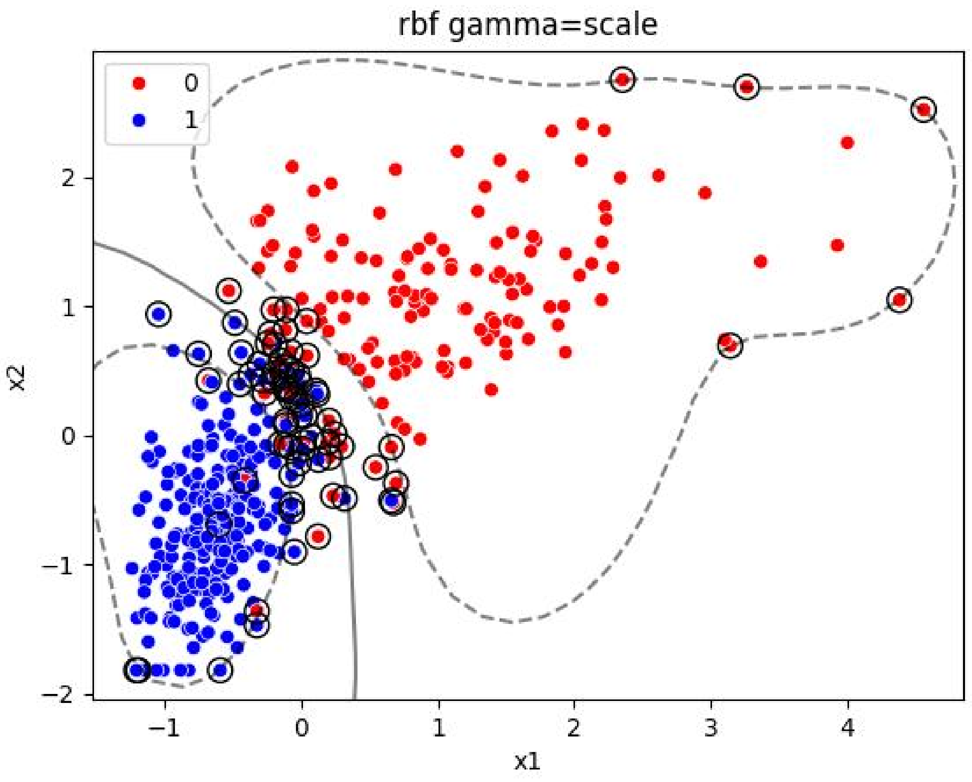

Figure 14. Two Feature RBF

3.2 Final Analysis Finally of all the classifiers considered during this Experiment I would suggest that the 30 featured RBF kerneled classifier is the best. I believe this to be the case for two reasons. The first being that when considering the two feature space visualizations the decision boundary computed by the RBF classifier appeared to be more natural to the probable unknown truth boundary. As we have learned rarely do datasets truely have an optimal linear boundary. Even though on the two feature dataset the linear slightly outperformed the rbf classifier that was only on a specific test permutation and only considering a relativly small sample size. I think it is reasonable to make the assumption that the decision boundary between most features tends to be more realistically modeled by the RBF curve than a linear boundary.
The second reason is that for all tested classifiers the equivilant classifier considering 30 features outperformed the two feature counterpart. Even though the two feature classifiers performed well, because they were consistantly out performed by the 30 feature equivilant it leads me to believe that there is some information of value contained in the 28 other features, and they shouldn't be discounted even if they only add a slight improvement.