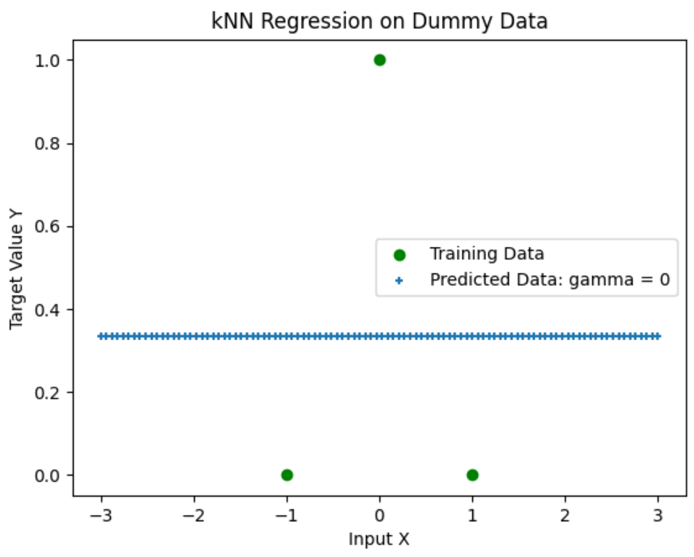
Machine Learning Assignment: Week 6

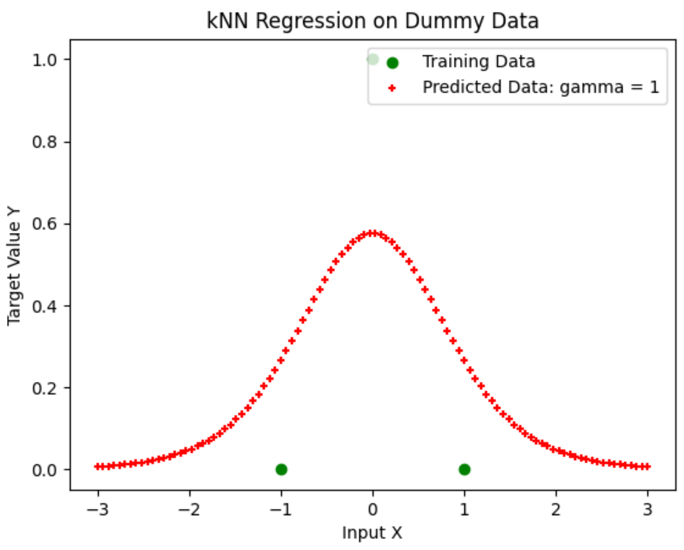
Alex Kennedy 17328638

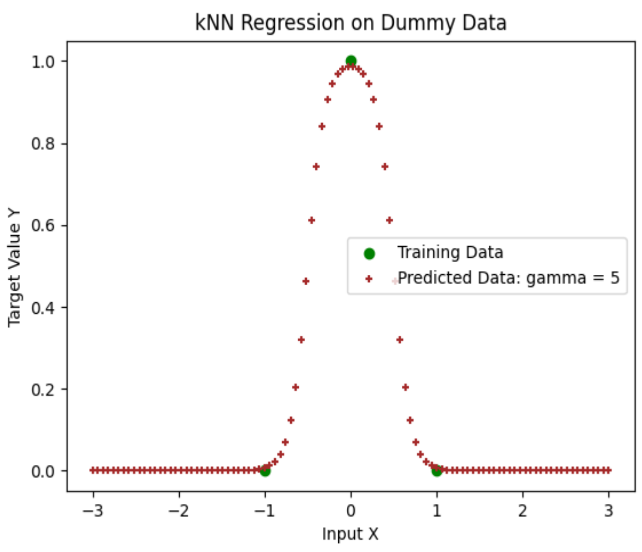
# id:11--99--11

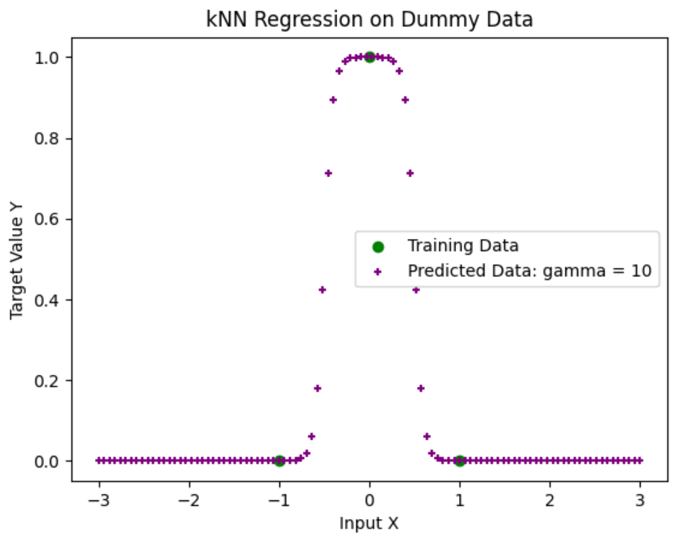
(i)(a)

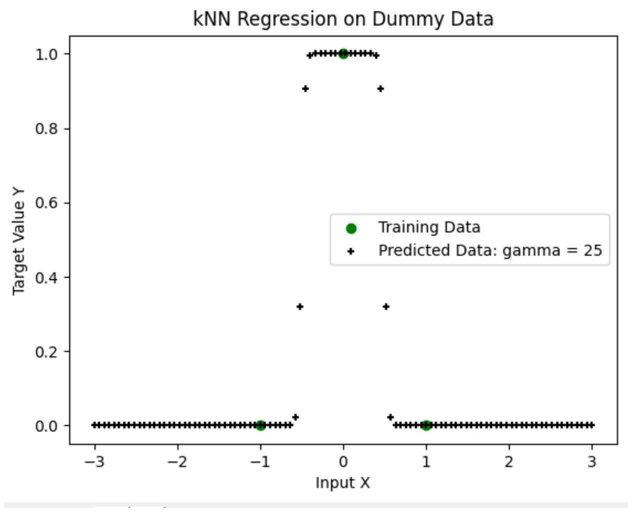
After creating the grid of feature values between -3 and 3 (100 values) and performing kNN regression with k = 3 and taking in different values of the hyper parameter γ (0, 1, 5, 10, 25); here are the plots generated.











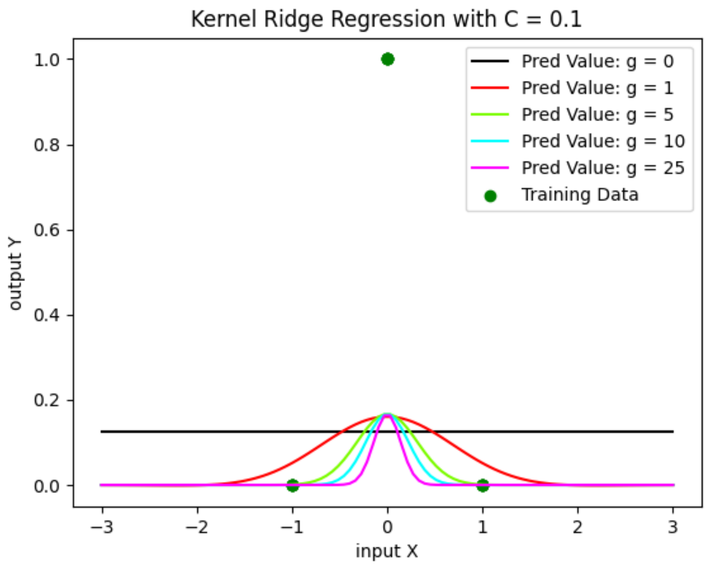
(i)(b)

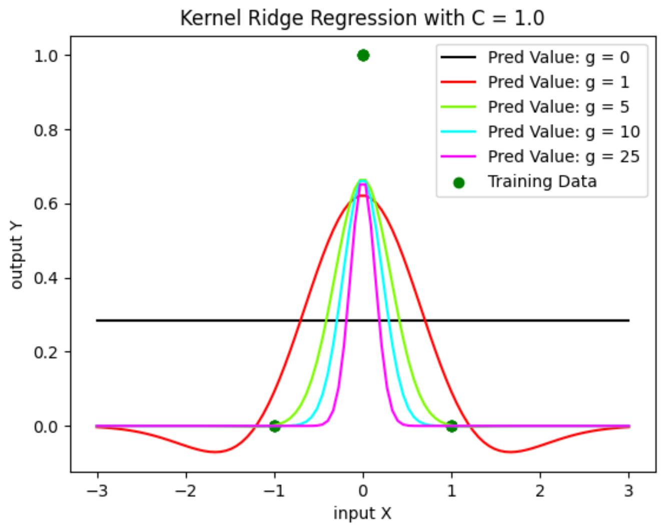
kNN predicts its values based on how similar it is to a point given in the training data. This is usually done on the distance of the point between the training data points. When using a gaussian kernel, every point is given a weight of importance, the closer a value is to a point in the training data, the more weight it is given to that point. This weight can be changed using the hyper parameter γ in the formula for a gaussian kernel $w^{(i)} = e^{-\gamma d(x^{(i)},x)^2}$. The greater the value of γ, the more weight is assigned to points. This can be seen in the different γ values I used from the plots above. If γ is 0, then the weight is uniform (i.e. w = 1), and so predicted values are all the same. When γ is 1, we can see that there is a smooth curve when the value of the input becomes closer to 0, we can see the predicted values get closer to target value 1 from the dummy data and as we move further away again it skews back down to 0. As the value of γ increases, the weight favours closer points more than points further away and this is seen as many of the predicted values become either 0 or 1.

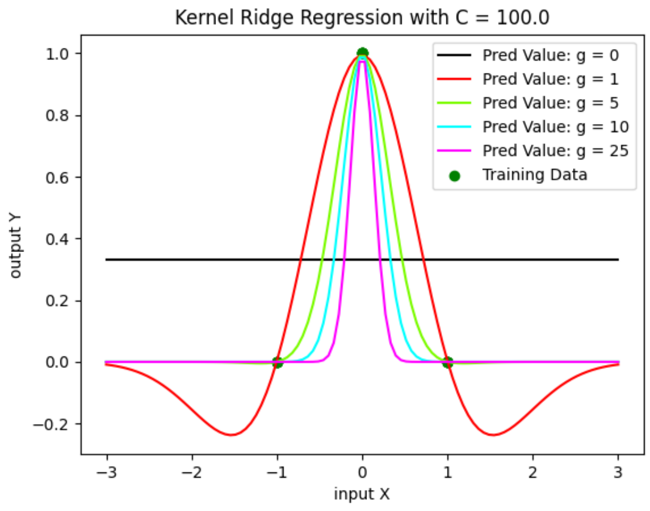
When γ = 25, pretty much all of the input values between -0.5 and 0.5 obtain a predicted value of 1 as the weight has become very high and so once so the input values become closer to 0 than either -1 or 1, their predicted value is placed at 1.

(i)(c)

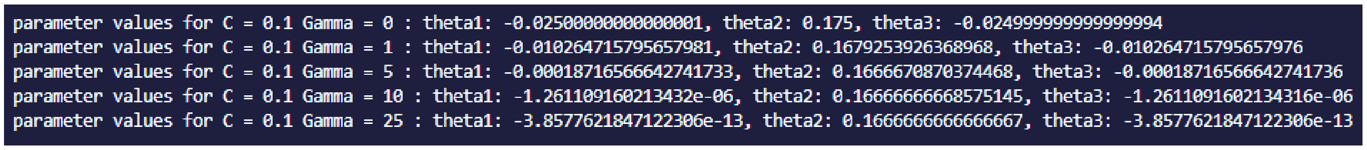
Here are three plots after performing Kernel Ridge Regression on a grid of feature values ranging from -3 to 3. Each plot uses a different C value to calculate alpha ($\alpha = (\frac{1}{2C})$) and has gamma values for weights (0, 1, 5, 10, 25).


Kernel Ridge Regression with C = 0.1
Pred Value: g = 0
Pred Value: g = 1
Pred Value: g = 5
Pred Value: g = 10
Pred Value: g = 25
Training Data
output Y
input X

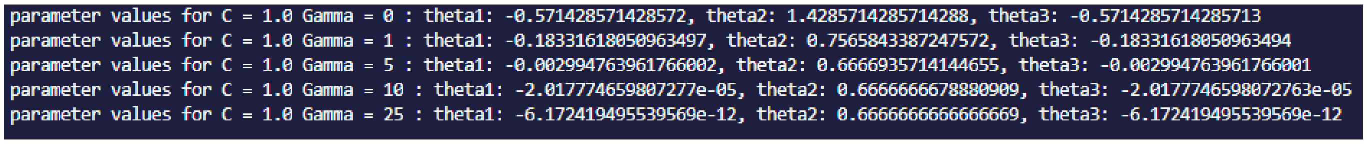

Kernel Ridge Regression with C = 1.0
Pred Value: g = 0
Pred Value: g = 1
Pred Value: g = 5
Pred Value: g = 10
Pred Value: g = 25
Training Data
output Y
input X


Kernel Ridge Regression with C = 100.0
Pred Value: g = 0
Pred Value: g = 1
Pred Value: g = 5
Pred Value: g = 10
Pred Value: g = 25
Training Data
output Y
input X

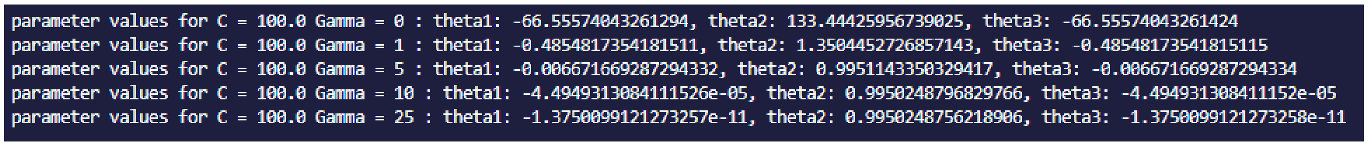
Here are the parameter values for each value of C and γ,



```
parameter values for C = 0.1 Gamma = 0 : theta1: -0.025000000000000001, theta2: 0.175, theta3: -0.024999999999999994
parameter values for C = 0.1 Gamma = 1 : theta1: -0.010264715795657981, theta2: 0.1679253926368968, theta3: -0.010264715795657976
parameter values for C = 0.1 Gamma = 5 : theta1: -0.00018716566642741733, theta2: 0.1666670870374468, theta3: -0.00018716566642741736
parameter values for C = 0.1 Gamma = 10 : theta1: -1.261109160213432e-06, theta2: 0.16666666668575145, theta3: -1.2611091602134316e-06
parameter values for C = 0.1 Gamma = 25 : theta1: -3.8577621847122306e-13, theta2: 0.1666666666666667, theta3: -3.8577621847122306e-13
```



```
parameter values for C = 1.0 Gamma = 0 : theta1: -0.571428571428572, theta2: 1.4285714285714288, theta3: -0.5714285714285713
parameter values for C = 1.0 Gamma = 1 : theta1: -0.18331618050963497, theta2: 0.7565843387247572, theta3: -0.18331618050963494
parameter values for C = 1.0 Gamma = 5 : theta1: -0.002994763961766002, theta2: 0.6666935714144655, theta3: -0.002994763961766001
parameter values for C = 1.0 Gamma = 10 : theta1: -2.017774659807277e-05, theta2: 0.6666666678880909, theta3: -2.0177746598072763e-05
parameter values for C = 1.0 Gamma = 25 : theta1: -6.172419495539569e-12, theta2: 0.6666666666666669, theta3: -6.172419495539569e-12
```



```
parameter values for C = 100.0 Gamma = 0 : theta1: -66.55574043261294, theta2: 133.44425956739025, theta3: -66.55574043261424
parameter values for C = 100.0 Gamma = 1 : theta1: -0.4854817354181511, theta2: 1.3504452726857143, theta3: -0.48548173541815115
parameter values for C = 100.0 Gamma = 5 : theta1: -0.006671669287294332, theta2: 0.9951143350329417, theta3: -0.006671669287294334
parameter values for C = 100.0 Gamma = 10 : theta1: -4.4949313084111526e-05, theta2: 0.9950248796829766, theta3: -4.494931308411152e-05
parameter values for C = 100.0 Gamma = 25 : theta1: -1.3750099121273257e-11, theta2: 0.9950248756218906, theta3: -1.3750099121273258e-11
```

It seems that in general, the parameter values increase as the value of C is increased. There are three parameter values as there are three points in the training data. There is a parameter value for each value in the training data so this can obviously become quite large quite quickly.

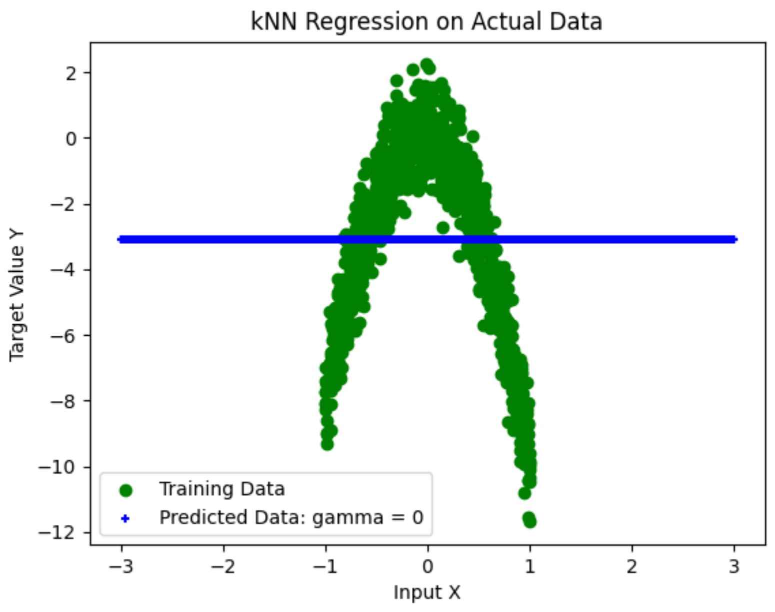
(i)(d)

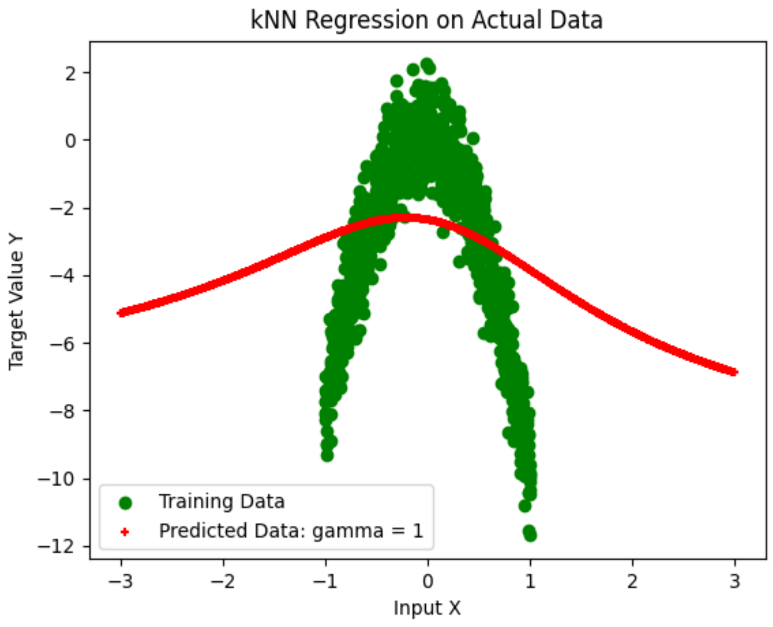
When the C value is small (C = 0.1), the model predicts that the majority of the values on the grid features are closer to 0 than to 1. This can be seen quite easily when looking at the gamma = 0 line on each of the plots. However, as the value of C continues to increase, the model stretches out more. This is due to the alpha value becoming smaller as the C value increases as large alpha values punishes outlier points that are far away from the rest of the values and causes them to be pushed closer to them. This can be seen in our parameter values If we take our parameter values for gamma = 1 for each value of C, we can see that the parameters get larger as C increases (alpha becomes smaller).

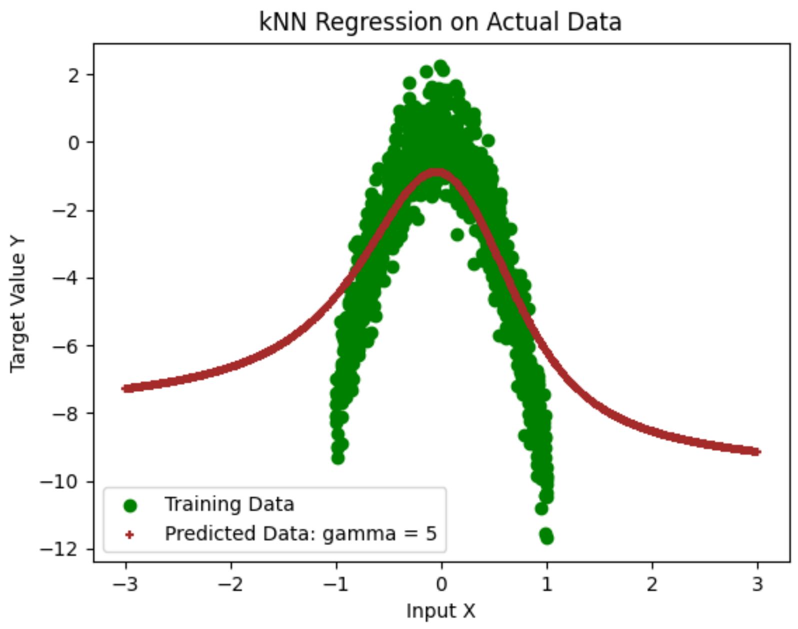
Once again when gamma = 0, the weight is uniform, so all the predictions become the same and they skew lower than y = 0.5 because there are more values = 0 than there are values = 1. As the gamma values increase, our model starts to become quite similar to our kNN model from the previous question.

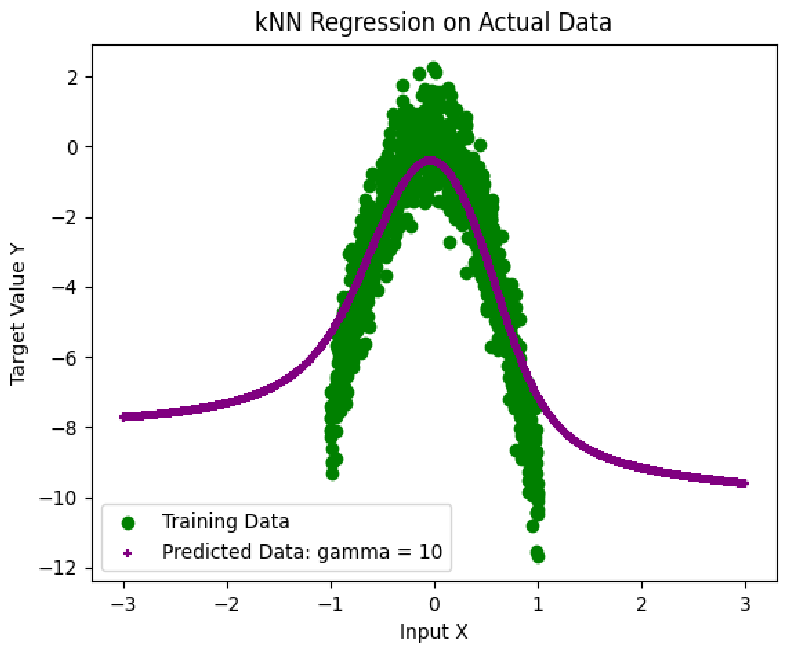
(ii)(a)

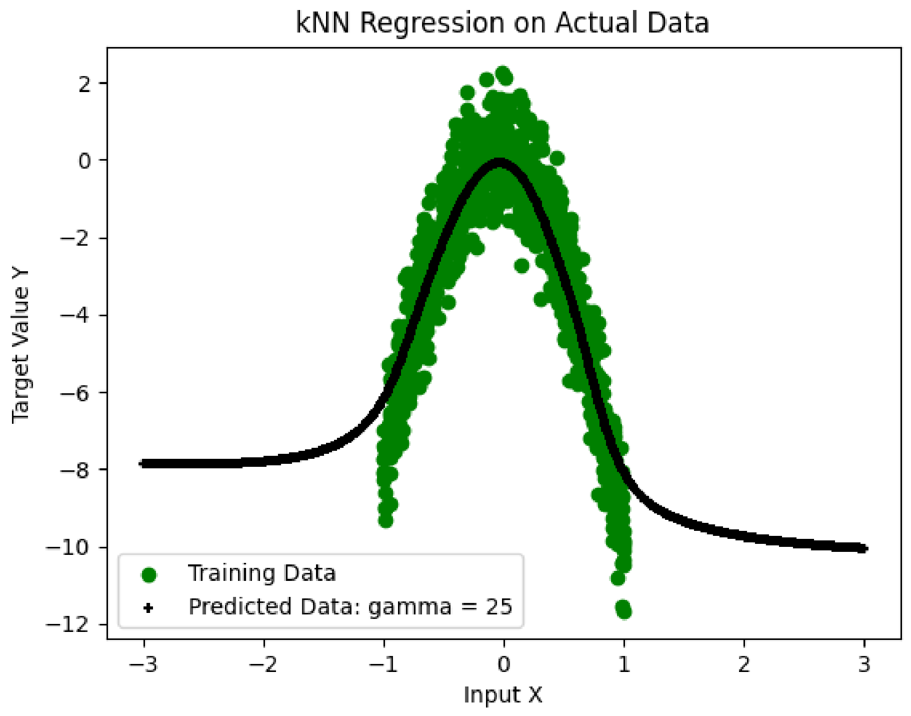
Here are the plots of the predictions of kNN using gaussian kernel where gamma = (0,1,5,10,25).









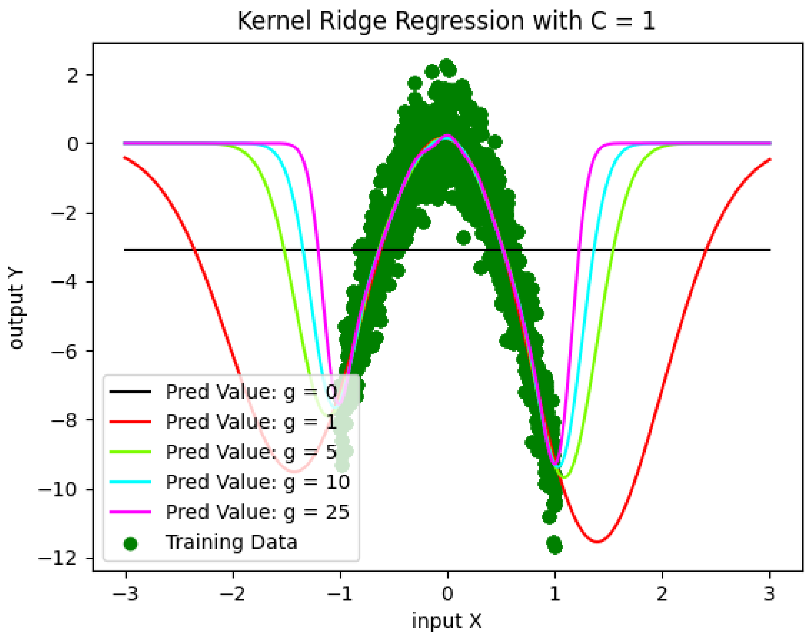


When the gamma value is zero, the weight is uniform and so we see the predictions are outputting as this straight line once again. As the gamma value increases, we see that the predictions are fitting the training data more and more. Gamma = 25 has the best fit from all of our plots. This is because the increasing gamma value gives us a higher weight for points closest to our current point than points further away and so the predictions are pushed closer to the actual training data points.

When the predictions move outside of the training data, there are no nearest neighbors and so, the regression performs quite poorly. The weights get smaller and smaller the further away we get from the training data the predictions flatten out.

(ii)(b)

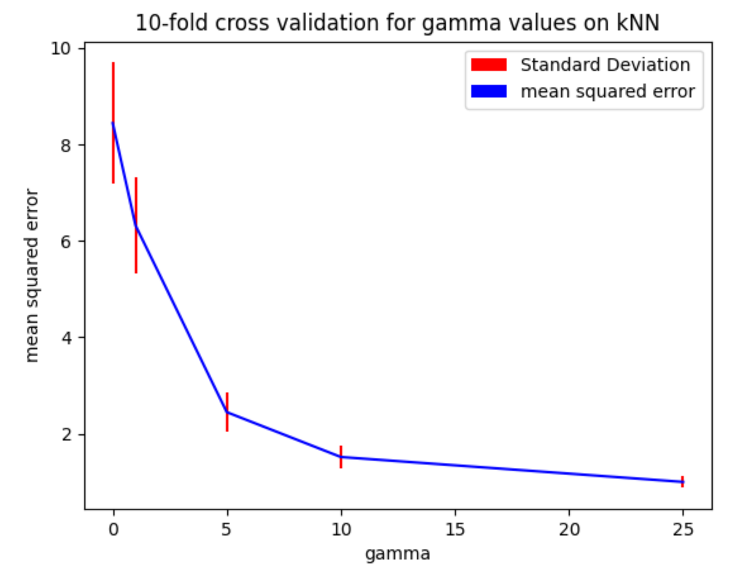
Here is the plot for the different values of gamma with C = 1 for kernel Ridge Regression



As the value of gamma changes, the predictions get tighter and tighter to the actual training data as the increase in the weights pulls these predictions closer to the actual training data. The pink line which is gamma = 25, has a little irregular movement in the middle of the data. This suggests that it is moving with the training data much better than the other values of gamma.

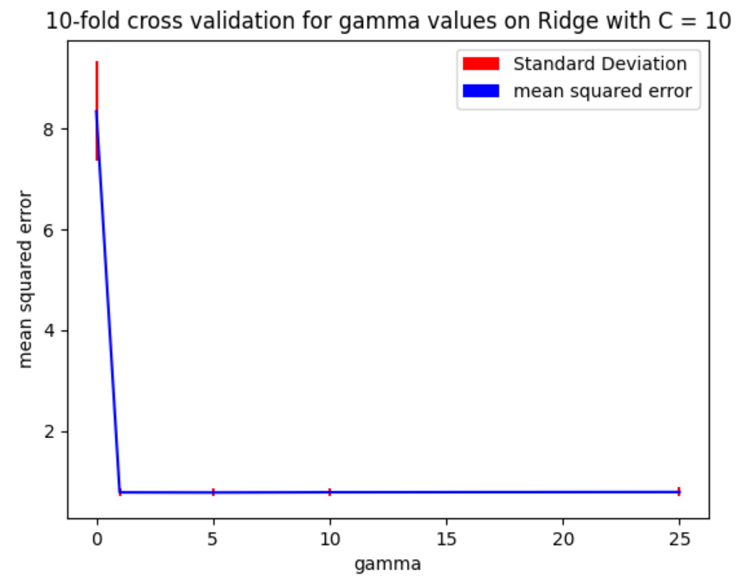
(ii)(c)

Here is the plot for performing 10-fold cross validation on a kNNRegressor with gamma values ranging from (0, 1, 5, 10, 25)

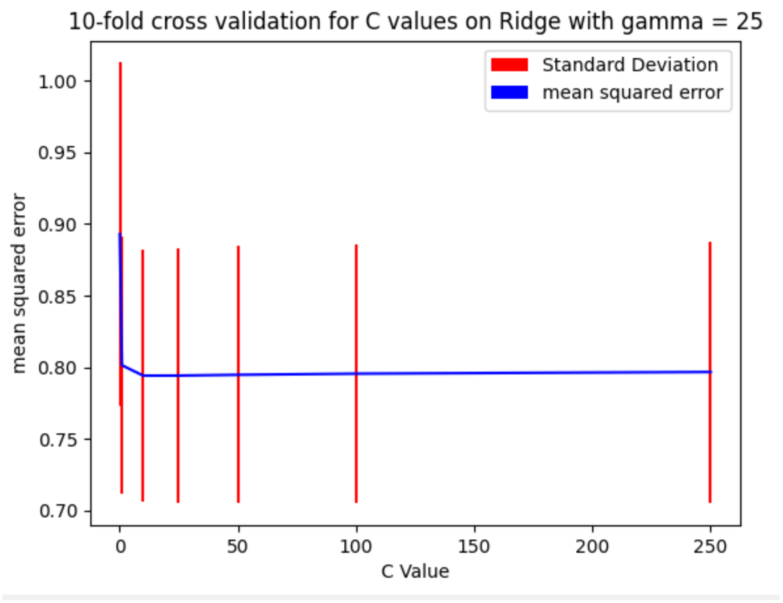


It seems that the best gamma value here is gamma value = 25. This is because we have the lowest mean squared error and also the standard deviation here is very small compared to the rest of the values of gamma. So, we choose gamma = 25 for the optimized kNN

For Ridge regression, I first decided to select the best gamma value. I chose a mid-range value of C (C = 10) to get this value then will find a good C value using the best gamma value found here.

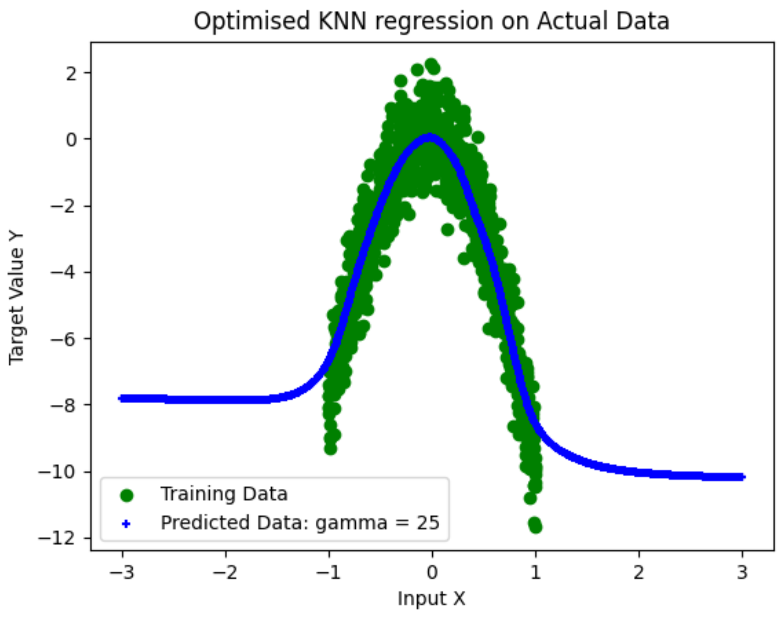


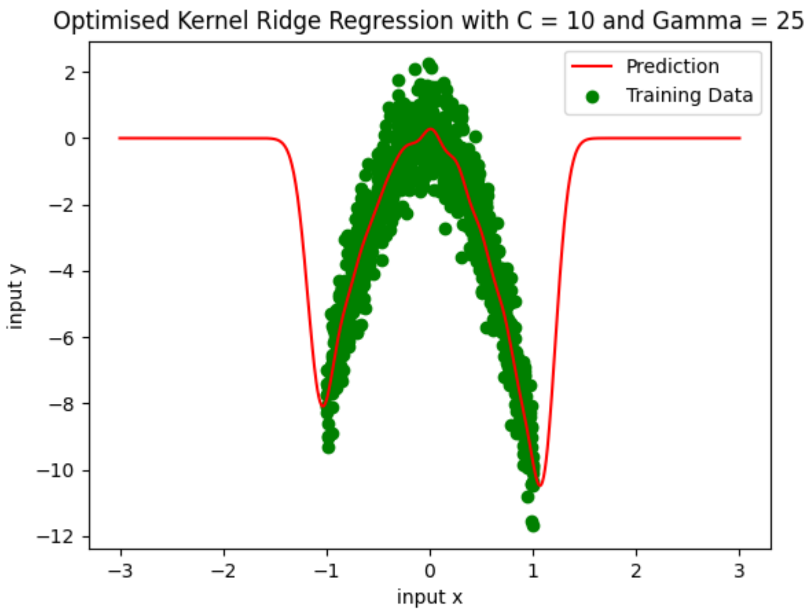
Once again it seems that gamma = 25 is the optimal value here. Although mean squared error is essentially the same throughout, the standard deviation is slightly smaller at gamma = 25. Now to choose best C value (0.1, 1, 10, 25, 50, 100, 250).



From the plot above, it seems like C = 10 is actually the optimal value due to having the lowest possible mean squared (along with the higher C values), but it has a lower standard deviation than the rest of the C values.

Here are the plots of the “optimsed” predictions using kNN and Ridge.





Both of the predictions perform very well here. They fit the training data quite nicely. It looks like Ridge fits the data a little bit better however as it reaches the very tips of the training data whereas kNN seems to miss the bottom portion on both sides. At the tip of the training data, you can see that Ridge also has a more irregular tip, which suggests that it is trying to fit the data better than kNN. Overall I would say that Ridge Regression is predicting the given training data better than kNN.