60-473 Assignment 01
Zachary Rudzinski, 104064162

1a. When do you think we need to consider a tie-resolution scheme for k-NN?
We would need to consider a tie-resolution scheme for k-NN in the scenario where...

1. k>1
2. And the number of potential classes > 1

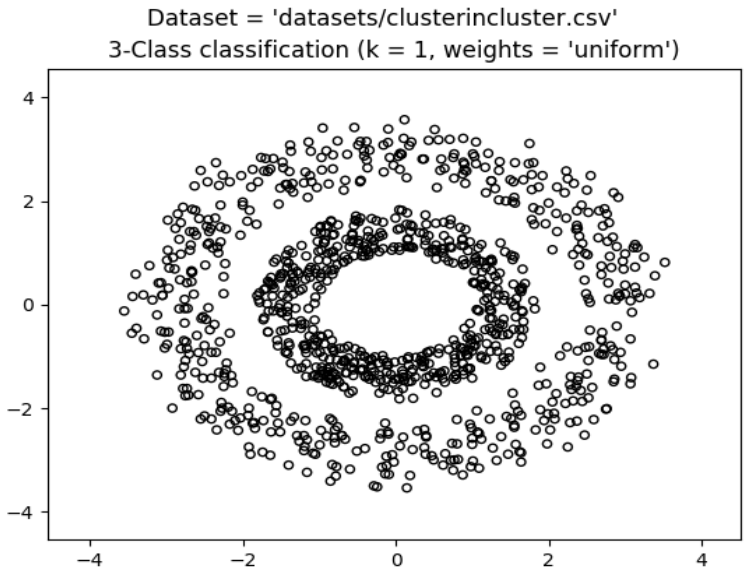
The decision rule of k-NN is essentially to poll the k nearest neighbors and classify the test sample as whatever is the most popular class in the k nearest neighbours. Therefore, if we find k nearest neighbors, where the probability of the test sample being of class C[i] is the same as class C[j], then we need a tie-breaking schema. The above pseudocode should identify situations where a tie-breaker would be possible, and therefore a tie-breaking schema necessary.

Ex: We have a test sample X, and it's 1 nearest neighbor is of class C. There is no possibility of a tie, since there is only one possible class.

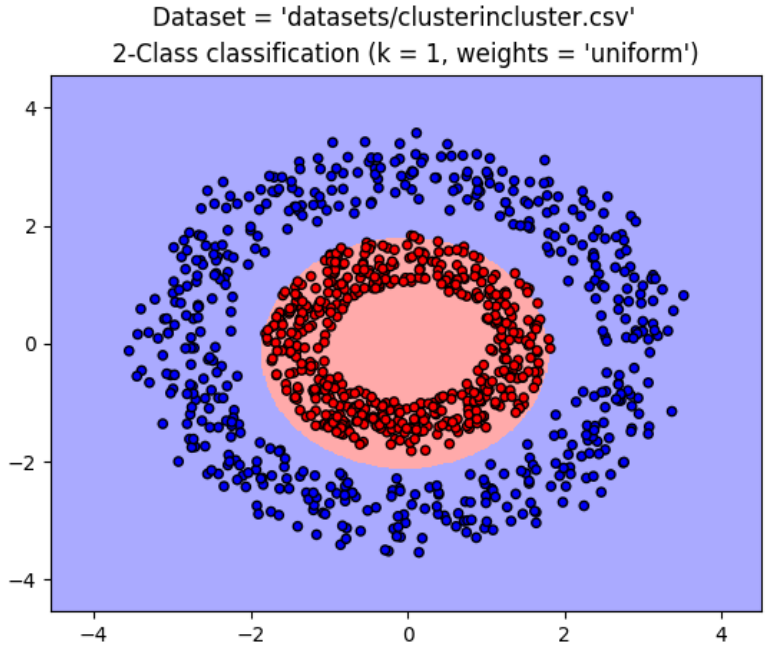
Ex: We have a test sample X, and it's 4 nearest neighbors are split between classes A and B. Since each class has a 0.50% probability of being selected, we need a tie-resolution schema.
Ex: We have a test sample X, and it's 3 nearest neighbors are of all different classes. Since each class has a 0.33% probability of being selected, we need a tie-resolution schema#.

1c: Specify how you have used the tool for running these classifiers.
Once the database was loaded, I split the dataset into training and test sets. Using the training data, I trained the classifier:
clf.fit(xTrain, yTrain)

Then, I was able to use the remaining test data to see how well the classifier performed. The results can be found in parts 1a and 1b, for k-NN and Bayes Classifier respectively.
score = clf.score(xTest, yTest)



Plot 1: An unclassified dataset (cluster in cluster)



Plot 2: A classified dataset, using a naive bayes classifier with no cross validation

2.Explain (in 5-6 lines) how 10-fold cross validation works.
Cross validation is a method of choosing the best model to represent your data. If you split your dataset between training and test data (70-30 split, respectively), you want to make sure that the 70% subset of your dataset that you've chosen to train your model is the 70% of the dataset that will lead to best representing the test set (and hopefully new samples in the future). In other words, it helps to avoid the bias of the test set bleeding into the model.

Using 10-fold validation, you split your dataset into 10 parts. You go through all the different combinations of training set parts and test set parts to determine which 7 parts of your dataset will create the best model to classify the 3 parts of the test set. You would use an evaluation metric to determine how incorrect your classification was, for each combination, and choose the combination which lead to the fewest errors to base your model on.

Note:
I reached an interesting scenario; When using 10-fold cross validation, my test set would only contain a single class. This would lead to classifications of 100% accuracy. The result of this happening meant that the confusion matrix would end up being a single value: [[100]], where 100 is the number of samples in the test set. This is probably due to how I implimented cross validation. I think there's more built in methods for cross validation that I'll have to look into. Because of all this, I'm including the results of the 5 measures of efficiency for both the classifiers, and with/without cross validation.

-- K-nearest Neighbors, cross validation --
Dataset: datasets/clusterincluster.csv Score: 1.0
    Test set contains one class only. There is no false positive or false negative; Only the one class.
Dataset: datasets/halfkernel.csv Score: 1.0
    Test set contains one class only. There is no false positive or false negative; Only the one class.
Dataset: datasets/twogaussians.csv Score: 1.0
    tn: 6 fp: 0 fn: 0 tp: 34
    specificity: 1.0
    sensitivity: 1.0
    ppv: 1.0
    npv: 1.0
Dattaset: datasets/twospirals.csv Score: 0.95
    Test set contains one class only. There is no false positive or false negative; Only the one class.

-- K-nearest Neighbors, no cross validation --
Dataset: datasets/clusterincluster.csv Score: 1.0
    tn: 164 fp: 0 fn: 0 tp: 166
    specificity: 1.0
    sensitivity: 1.0
    ppv: 1.0
    npv: 1.0
Dataset: datasets/halfkernel.csv Score: 1.0
    tn: 157 fp: 0 fn: 0 tp: 173
    specificity: 1.0
    sensitivity: 1.0
    ppv: 1.0
    npv: 1.0
Dataset: datasets/twogaussians.csv Score: 0.9621212121212122
    tn: 71 fp: 2 fn: 3 tp: 56
    specificity: 0.9726027397260274
    sensitivity: 0.9726027397260274
    ppv: 0.9726027397260274
    npv: 0.9726027397260274
Dataset: datasets/twospirals.csv Score: 0.9212121212121213
    tn: 155 fp: 14 fn: 12 tp: 149
    specificity: 0.9171597633136095
    sensitivity: 0.9171597633136095
    ppv: 0.9171597633136095
    npv: 0.9171597633136095

-- Naive Bayes, cross validation --
Dataset: datasets/clusterincluster.csv Score: 1.0
    Test set contains one class only. There is no false positive or false negative; Only the one class.
Dataset: datasets/halfkernel.csv Score: 0.97
    Test set contains one class only. There is no false positive or false negative; Only the one class.
Dataset: datasets/twogaussians.csv Score: 1.0
    Test set contains one class only. There is no false positive or false negative; Only the one class.
Dataset: datasets/twospirals.csv Score: 0.67
    tn: 0 fp: 0 fn: 33 tp: 67
    (Divide by zero error occurs here)

-- Naive Bayes, no cross validation --

Dataset: datasets/clusterincluster.csv Score: 1.0
      tn: 161 fp: 0 fn: 0 tp: 169
      specificity: 1.0
      sensitivity: 1.0
      ppv: 1.0
      npv: 1.0
Dataset: datasets/halfkernel.csv Score: 0.9575757575757575
      tn: 155 fp: 5 fn: 9 tp: 161
      specificity: 0.96875
      sensitivity: 0.96875
      ppv: 0.96875
      npv: 0.96875
Dataset: datasets/twogaussians.csv Score: 0.9545454545454546
      tn: 67 fp: 4 fn: 2 tp: 59
      specificity: 0.9436619718309859
      sensitivity: 0.9436619718309859
      ppv: 0.9436619718309859
      npv: 0.9436619718309859
Dataset: datasets/twospirals.csv Score: 0.603030303030303
      tn: 97 fp: 63 fn: 68 tp: 102
      specificity: 0.60625
      sensitivity: 0.60625
      ppv: 0.60625
      npv: 0.60625

3.For each classifier show the results obtained for the 4 datasets.

```
-- K-nearest Neighbors, 10-fold cross validation --
Dataset: datasets/clusterincluster.csv   Score: 1.0
Dataset: datasets/halfkernel.csv   Score: 1.0
Dataset: datasets/twogaussians.csv  Score: 1.0
Dataset: datasets/twospirals.csv   Score: 0.95

-- Naive Bayes, 10-fold cross validation --
dataset: datasets/clusterincluster.csv   score: 1.0
dataset: datasets/halfkernel.csv   score: 0.97
dataset: datasets/twogaussians.csv  score: 1.0
dataset: datasets/twospirals.csv   score: 0.67
```

4. For the k-NN find the best value of k, and compare it with those of 1-NN and Naïve Bayes classifiers. What is the best value of k for each dataset? Why?

```
-- Finding best value of K for K-nearest Neighbors --
Dataset: datasets/clusterincluster.csv
      k: 1 Score: 1.0
      ...
      k: 30 Score: 1.0
      The best value of k is 1 with a score of 1.0
Dataset: datasets/halfkernel.csv
      k: 1 Score: 1.0
      ...
      k: 30 Score: 1.0
      The best value of k is 1 with a score of 1.0
Dataset: datasets/twogaussians.csv
      k: 1 Score: 0.9848484848484849
      ...
      k: 19 Score: 0.9696969696969697
      The best value of k is 1 with a score of 0.9848484848484849
Dataset: datasets/twospirals.csv
      k: 1 Score: 0.9424242424242424
      k: 2 Score: 0.9575757575757575
      k: 3 Score: 0.9545454545454546
      k: 4 Score: 0.9424242424242424
      k: 5 Score: 0.9272727272727272
      k: 6 Score: 0.9545454545454546
      k: 7 Score: 0.9636363636363636
      ...
      k: 30 Score: 0.8909090909090909
      The best value of k is 7 with a score of 0.9636363636363636
```

Note: My search for the best k value did not use cross validation, because as mentioned earlier, my solution for CV is broken.

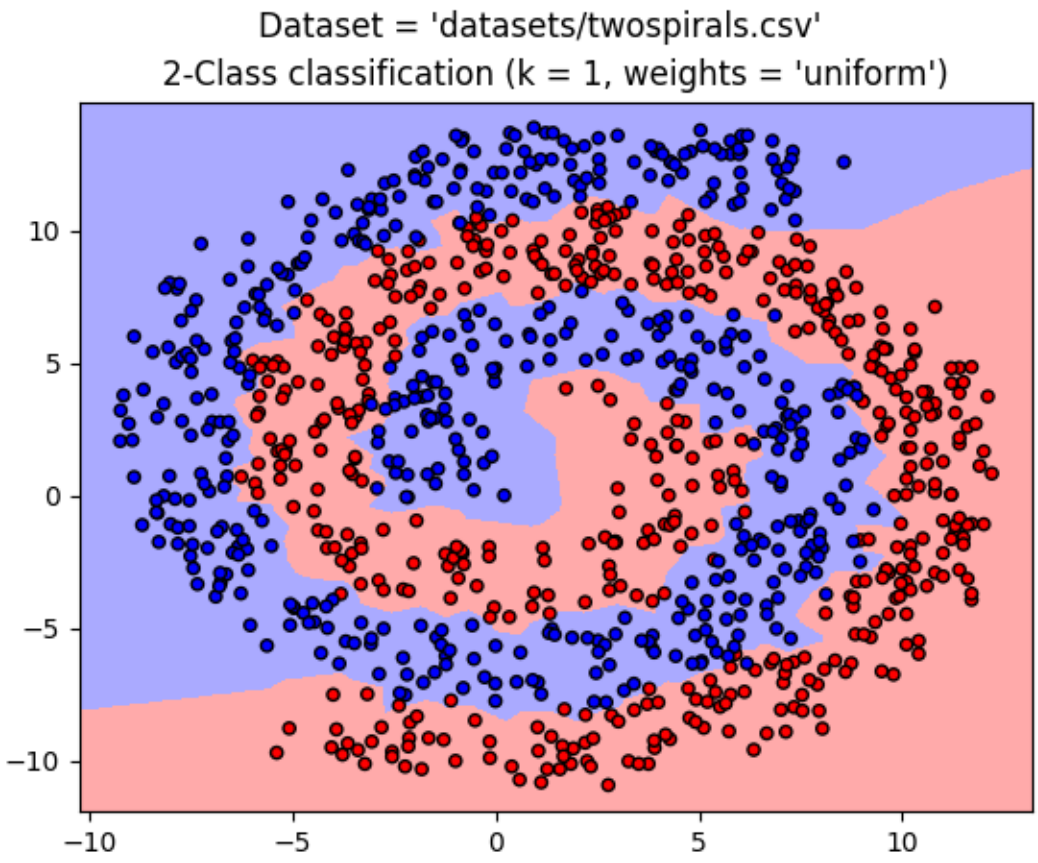
The best value of k for each dataset...

- Suprisingly, in many cases, 1 is the best option for most datasets. The two spirals dataset is the only one where this is not the case, where 7 is the best value of k.
-

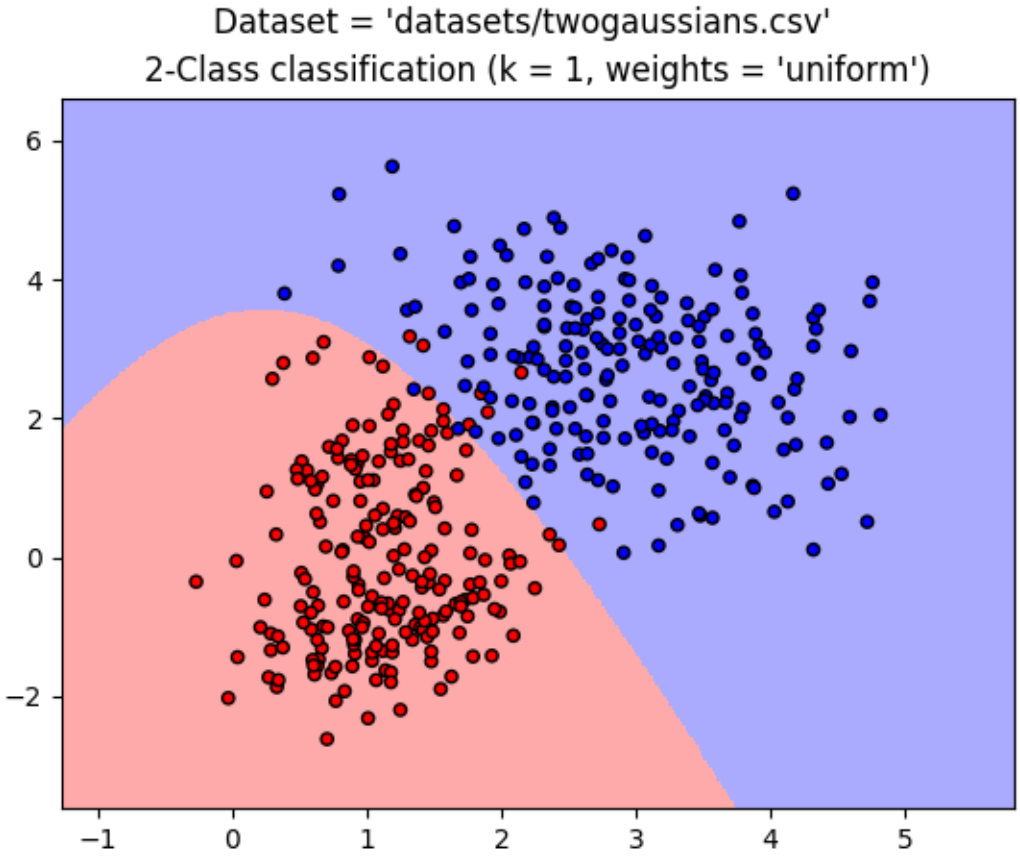
Comparing results of our classifiers...

- Even with finding the best value of k, the results were still a bit worse than the 1-NN with cross validation approach.
- For the first 3 datasets (cluster in cluster, half kernel, and two gaussians), the change in values of k didn't affect much.
  - This is probably because the data is very seperable. The only ambiguous areas might be where the two classes collide (like in the center of the two gaussians)
- However, in the two spirals dataset, we were able to go from 67% accuracy with naive bayes (10-fold cross validation) to 96% with k-NN (k=7).
  - This seems to be the case because the naive bayes approach attempted to linearly separate the data, even though it is very much non-linearly seperable.

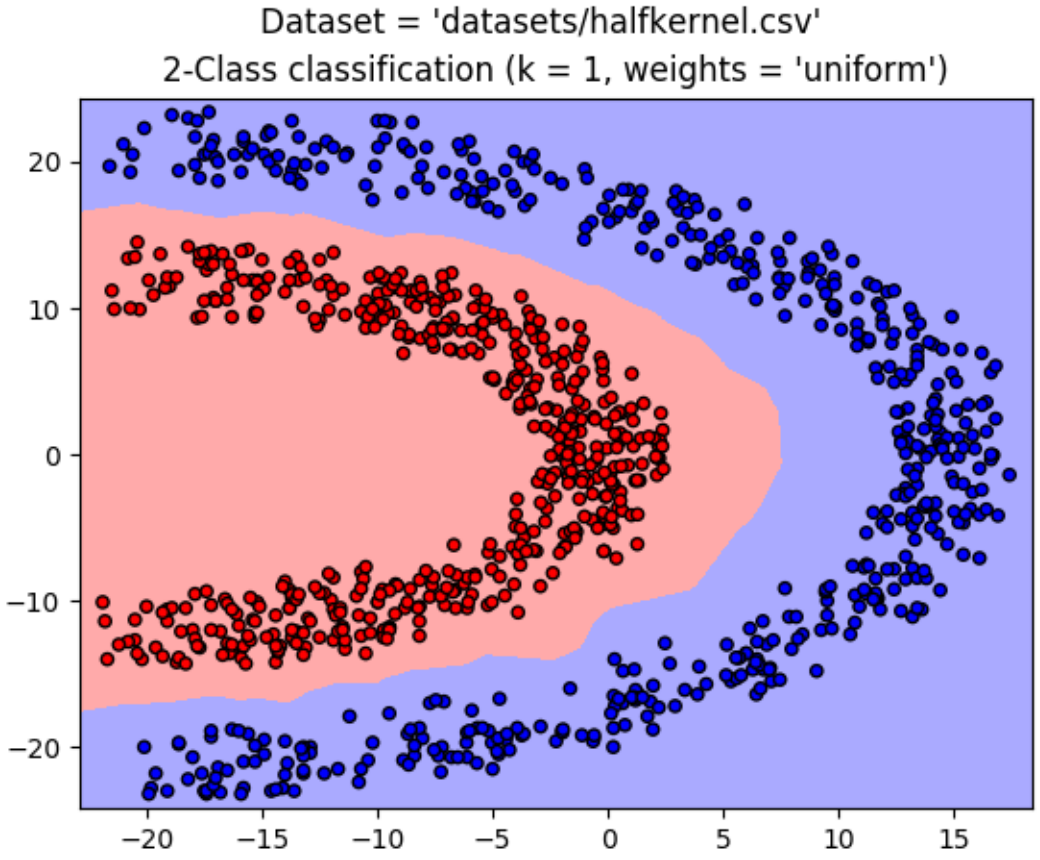
5. Plot the samples of the 4 datasets as points in the 2D space.



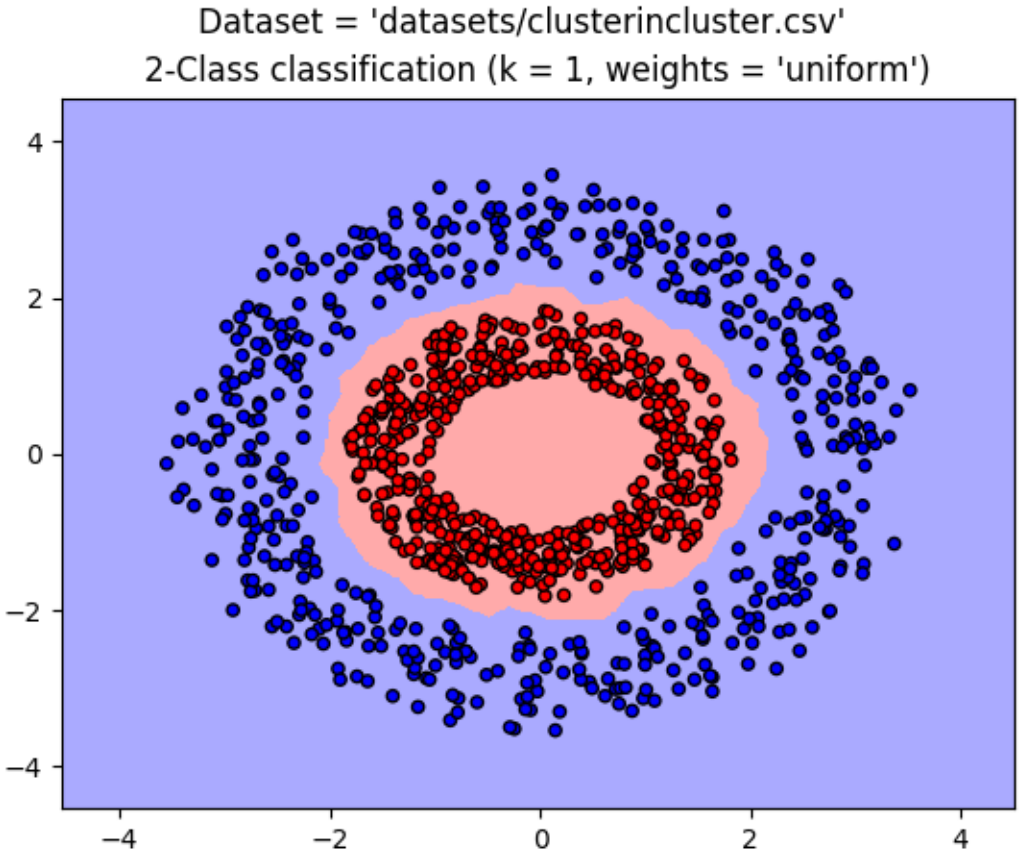
Plot 3: 1-NN classifier on the two spirals dataset



Plot 4: Naive bayes classifier on the two gaussians dataset



Plot 5: k-NN (k=1) classifier on the half kernel dataset



Plot 6: k-NN (k=1) classifer on the cluster in cluster dataset

6.Explain the behavior of each classifier. Why do you think it is good/poor?

- The k-NN classifier tends to be more accurate, while the bayes classifier tends to be more readable.
- Looking at the plots for each classifier, you can see that the k-NN classifier, while reaching better accuracy, is shakier.
- The bayes classifier on the other hand is worse, but has a smoother shape to it’s decision boundary, which is easier to read visually.
- This in mind, the readability gained by the naive bayes approach is negligible in this case; The k-NN classifier is clearly better in this scenario.

6b.In your opinion, which classifier is the best for that particular dataset, and why?

clusterincluster.csv

- Both classifiers have 100% acuracy after cross validation. The k-NN algorithm seems to be more sensitive to noise and the training set, as the positive region has more room to accomadate noise.
- I'd say that the k-NN version is better for this reason, as the bayes classifier would probably lead to some false negative classifications (especially near the coordinates of (1,1).

halfkernel.csv

- The k-NN classifier outdoes the bayes classifier here. The k-NN classifier takes on more of an elipse shape, properly classifying all datapoints. The bayes classifier however, takes on a spherical shape, misclassifying quite a bit of points.
- The k-NN algorithm is clearly better for this dataset, as further points would be properly classified.

twogaussians.csv

- In this case, both classifiers seem to have 100% accuracy. However, there is quite a bit of noise in the center of the two gaussians, which I feel would be mis-classified easily. You can see that the k-NN classifier bleeds into the negative gaussian.
- This makes me think that the bayes classifier would be more effective in this case. It succeeds in isolating positive from negative, while ignoring some of the noise in the center.

twospirals.csv

- Once again, the k-NN is the best classifier here. it properly forms the shape of the interweiving spirals, without listening too closely to the noise. The bayes classifier however, attempts to linearly separate a non-linearly seperable dataset, causing it's accuracy to be a low 67%.
- For these reasons, the k-NN algorithm is better.