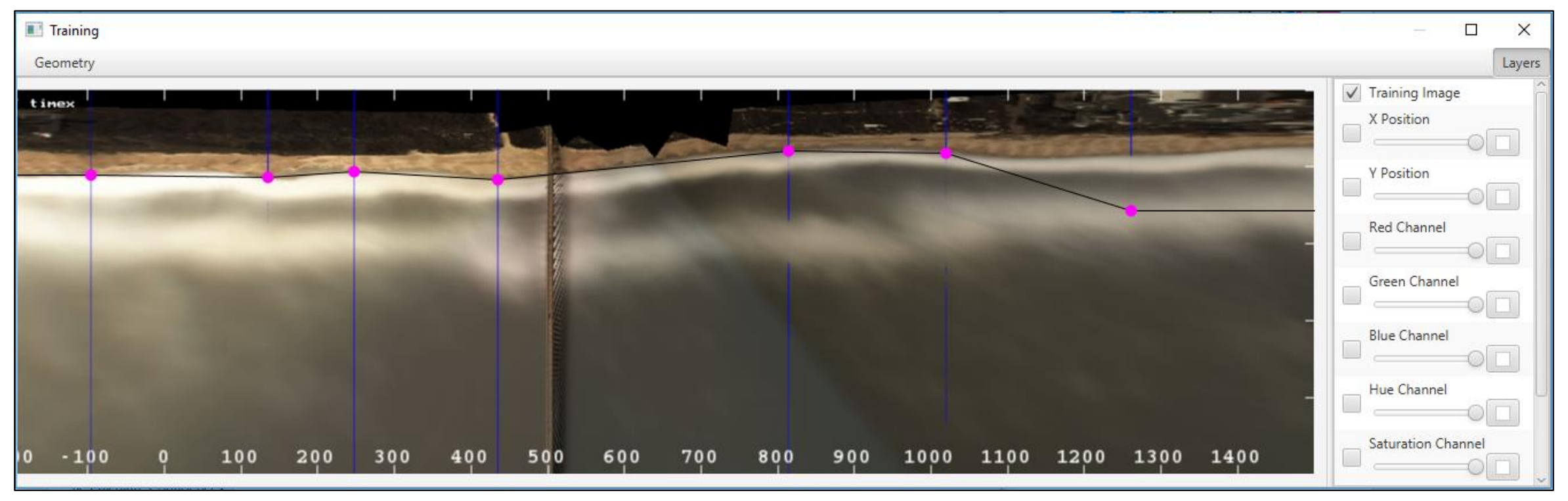
Visualization-Driven Boosting for High-Dimensional K Nearest Neighbors

Laura Smith – SEAP – U.S. Naval Research Laboratory, Stennis MS

ABSTRACT

Performing image region annotation on coastal datasets is a time consuming task, but very helpful for studying coastal erosion and the sea floor. The Naval Research Laboratory currently studies ways of applying artificial intelligence, or machine learning, to help scientists with annotation. However, machine learning implementations are typically imperfect and may be optimized, especially when there isn't a vast amount of training data or the implementation must be very fast. Although this is typically done with optimization algorithms, this project explores the benefits of using data visualization. Data visualization allows scientists to gain familiarity with their data, and make intuitive adjustments to their machine learning implementation. I created and used a visualization tool to boost the accuracy of the machine learning implementation in an image region annotation program. I was able to optimize the learner to reduce incorrect classifications by 11%. Although the tool was only used for image region annotation, it has the potential to work with any machine learner that uses an instance based classifier.



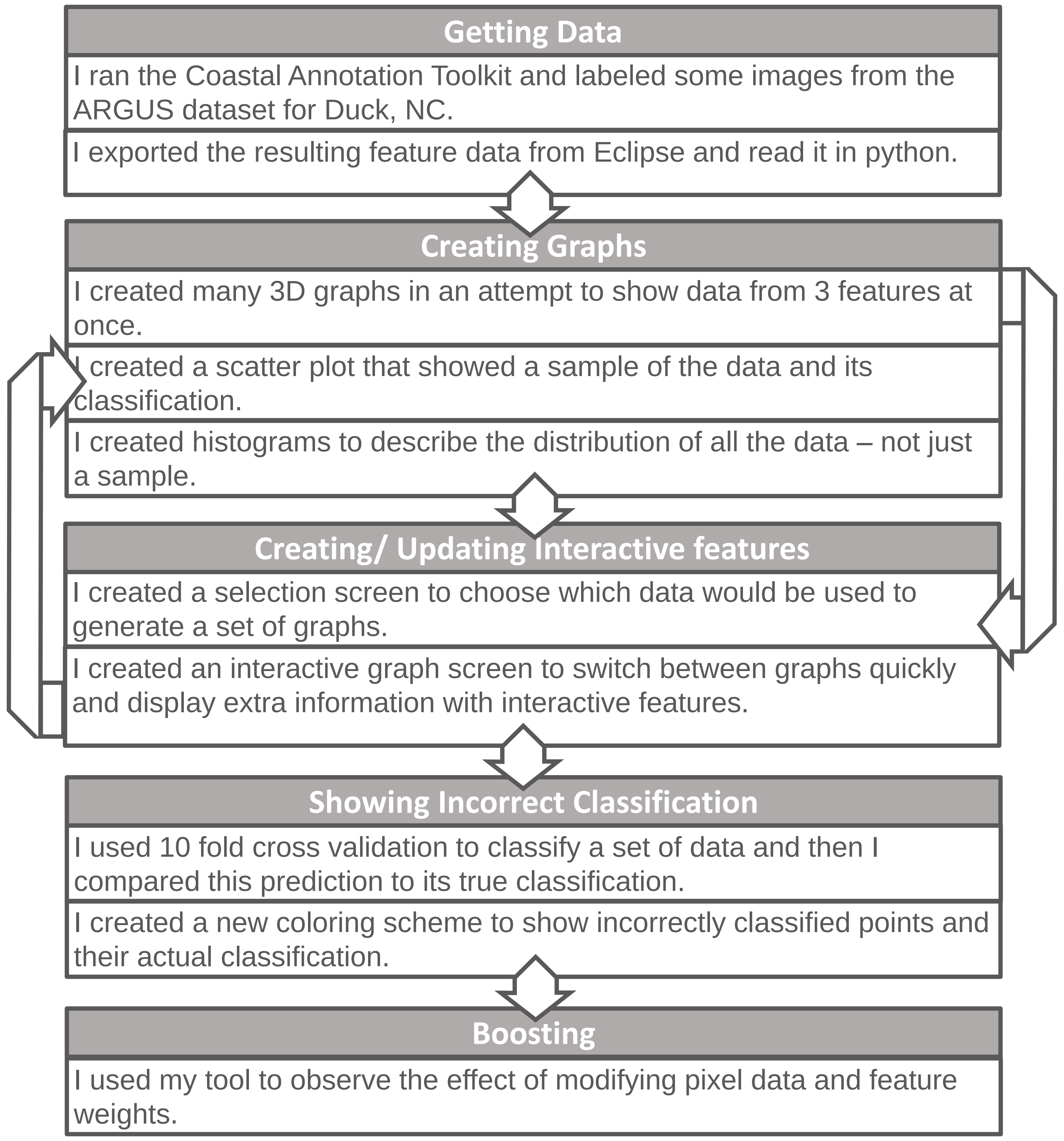
BACKGROUND

The Coastal Annotation Toolkit is a program that helps users label different regions, such as sand and vegetation, in images of shorelines. It uses machine learning to predict where these regions are located to aid the user in annotation. In order to learn, the toolkit first describes each pixel in a given image as a feature point with many dimensions of data, such as position, hue, and brightness. The user begins by hand classifying some of these points as either part of the region or not. Then, the toolkit predicts the classification of new points by comparing their feature data to that of the classified ones. This is done by an instance based classifier called K Nearest Neighbors (KNN). Though the current implementation of the Coastal Annotation Toolkit reduces the amount of time for annotation, the KNN implementation still suffers from a significant amount of errors that this project seeks to improve.

HYPOTHESIS

After initial observation, I hypothesized that the KNN classifier can be optimized with my visualization tool by addressing two problems. First, not all image features are equally important. For example, when annotating a horizontal shoreline, y position is a better predictor of where the shore is located than x position. Despite this, KNN weighs each feature equally. Second, the distribution of a feature's data set can affect its impact on the classification of new points. Heavily clustered data that is not of the same class will give less accurate results than evenly distributed data. The accuracy of KNN's predictions can be increased by weighting the features differently and by performing operations on the data to distribute it more evenly. This project intends to determine which modifications will result in the most accurate prediction of shorelines. This was done by creating a tool to visualize feature data.

PROCESS



Getting Data

I ran the Coastal Annotation Toolkit and labeled some images from the ARGUS dataset for Duck, NC.

I exported the resulting feature data from Eclipse and read it in python.

Creating Graphs

I created many 3D graphs in an attempt to show data from 3 features at once.

I created a scatter plot that showed a sample of the data and its classification.

I created histograms to describe the distribution of all the data – not just a sample.

Creating/ Updating Interactive features

I created a selection screen to choose which data would be used to generate a set of graphs.

I created an interactive graph screen to switch between graphs quickly and display extra information with interactive features.

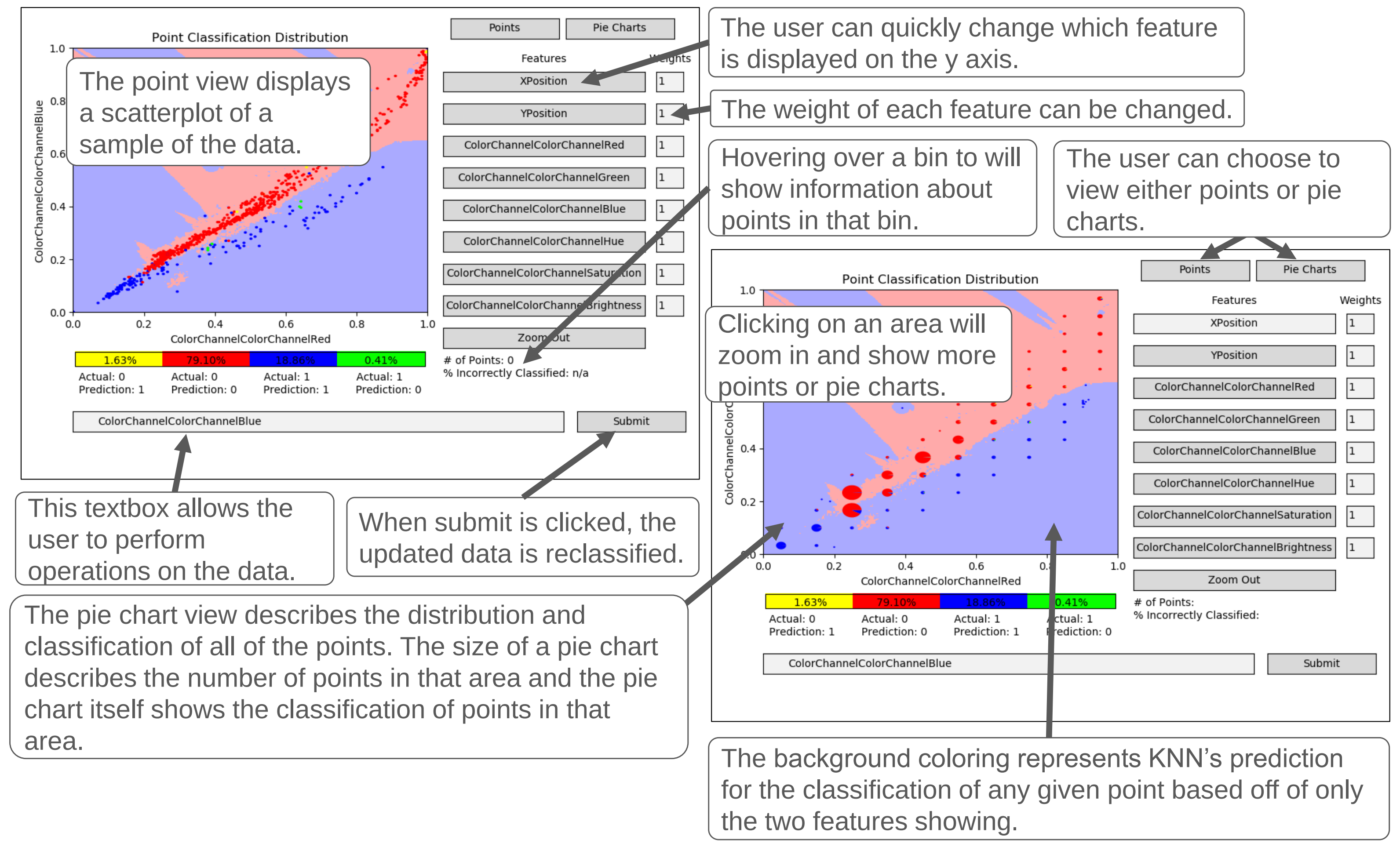
Showing Incorrect Classification

I used 10 fold cross validation to classify a set of data and then I compared this prediction to its true classification.

I created a new coloring scheme to show incorrectly classified points and their actual classification.

Boosting

I used my tool to observe the effect of modifying pixel data and feature weights.



The point view displays a scatterplot of a sample of the data.

The user can quickly change which feature is displayed on the y axis.

The weight of each feature can be changed.

Hovering over a bin to will show information about points in that bin.

The user can choose to view either points or pie charts.

Clicking on an area will zoom in and show more points or pie charts.

This textbox allows the user to perform operations on the data.

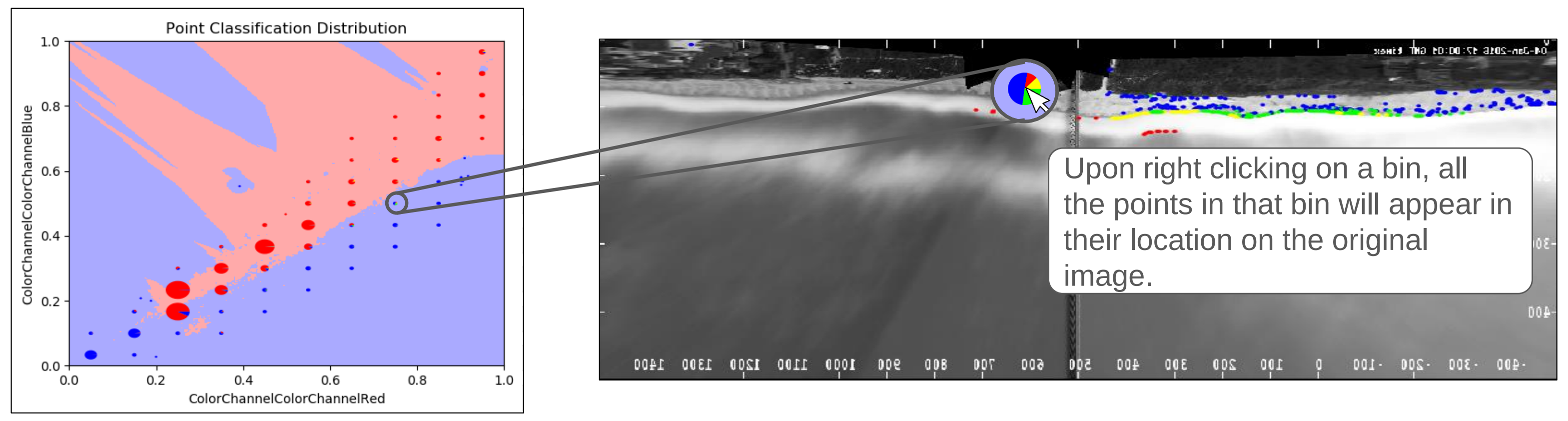
When submit is clicked, the updated data is reclassified.

The pie chart view describes the distribution and classification of all of the points. The size of a pie chart describes the number of points in that area and the pie chart itself shows the classification of points in that area.

The background coloring represents KNN's prediction for the classification of any given point based off of only the two features showing.

RESULTS

To help me make predictions, I often used my tool's right click functionality. By right clicking on an area of a graph, I could see where all of the points in that area were located on the original image. I found that points that were clustered in my graph were also often clustered in the image. These clusters might follow the shoreline or the outline of a cloud. Seeing this helped me understand what role each feature played in classifying the data.



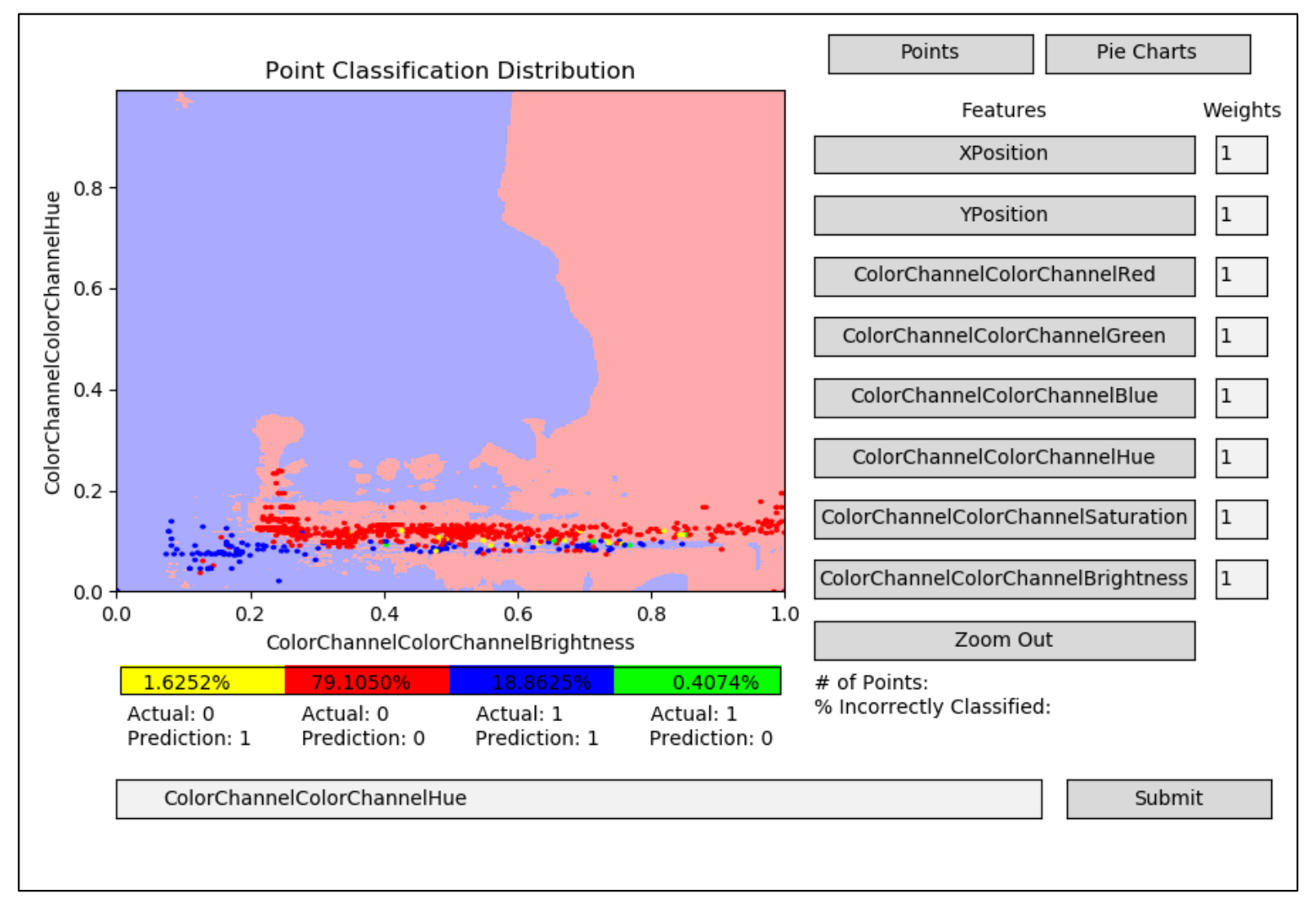
Upon right clicking on a bin, all the points in that bin will appear in their location on the original image.

Understanding this, I was able to weight the features differently. The set of images I worked with contained elements that could throw off the classifier's predictions, such as a pier jutting out from the shore, numbers bordering the image, and black splotches which were artifacts of orthorectification. To increase the accuracy of the classifier, I decreased the weight of the saturation values. Although saturation was a strong predictor of the location of the shoreline, it also tended to group the pier or the black locations into this prediction. Instead, I increased the weight of the hue values, which were more resistant to these elements. I also reduced the weight of the y positon values to 0. Y position threw off the accuracy of the classifier because although it could describe a general band that the shoreline was likely in, it was irrelevant when determining the exact location of the shoreline at a point. By making these adjustments, I greatly reduced misclassification.

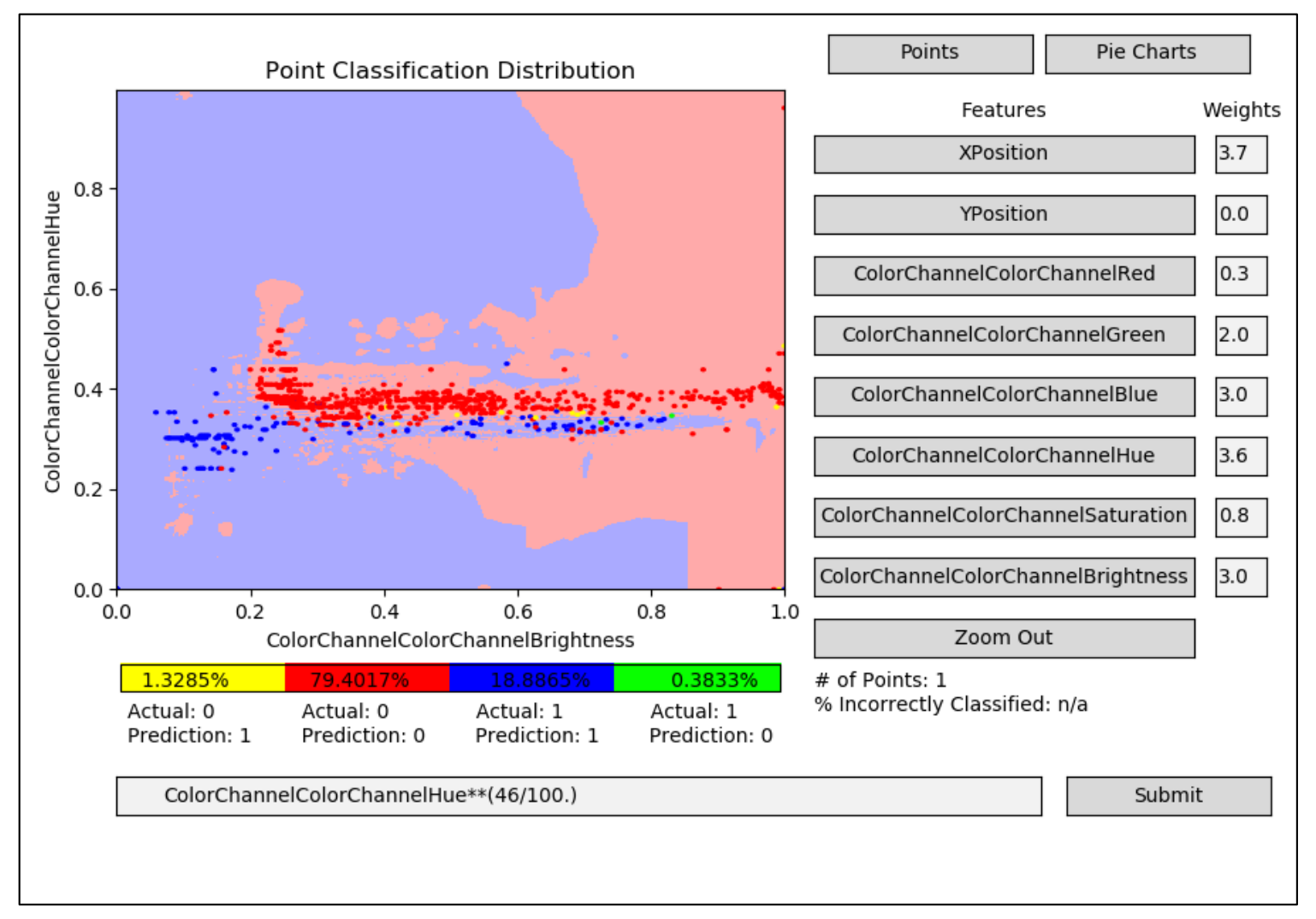
Once the features were weighted correctly, I performed modifications on the data to spread it out. After observing that most of the hue and saturation values were low, I performed the following transforms: hue = hue ^0.46 and saturation = saturation ^0.65. These changes combined with weighting the features differently reduced misclassification by an average of 11%.

	X Pos	Y Pos	Red	Green	Blue	Hue	Saturation	Brightness
Weights	3.7	0.0	0.3	2.0	3.0	3.6	0.8	3.0
Operatons	none	none	none	none	none	x ^ 0.46	x ^ 0.65	none

Before



After



ACKNOWLEDGEMENTS

I would like to thank Dr. Chris Michael and Mrs. Shannon Mensi for making this possible.