Fall 12-16-2016

A Machine Learning Approach to Determine Oyster Vessel Behavior

Devin Frey
University of New Orleans, dfrey@uno.edu

A Machine Learning Approach to Determine Oyster Vessel Behavior

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Devin Frey

B.S. University of New Orleans, 2013

December, 2016

Acknowledgements

I would like to thank the following professors for their assistance in the completion of this project: Dr. Mahdi Abdelguerfi, for providing me the opportunity to work on this project; Dr. Thomas Soniat, for entrusting the project to me and providing insight on the biology-related aspects of this paper; and Dr. Md Tamjidul Hoque, for providing much guidance on the machine learning portions of this project. I would also like to thank the Louisiana Department of Wildlife and Fisheries for collecting the vessel data used in this project, as well as the University of New Orleans for providing the education that allowed me to produce this thesis.

Thanks also to David Gallegos, for providing the groundwork that this project is built upon, and Nathan Cooper, Dustin Peabody, and Daniel Ward for providing feedback on some of the visualizations produced for this project.

Lastly, I thank my parents and other family members who pushed me to pursue a higher education.

Table of Contents

List of Figures

Abstract

A support vector machine (SVM) classifier was designed to replace a previous classifier which predicted oyster vessel behavior in the public oyster grounds of Louisiana. The SVM classifier predicts vessel behavior (docked, poling, fishing, or traveling) based on each vessel's speed and either net speed or movement angle. The data from these vessels was recorded by a Vessel Monitoring System (VMS), and stored in a PostgreSQL database. The SVM classifier was written in Python, using the scikit-learn library, and was trained by using predictions from the previous classifier. Several validation and parameter optimization techniques were used to improve the SVM classifier's accuracy. The previous classifier could classify about 93% of points from July 2013 to August 2014, but the SVM classifier can classify about 99.7% of those points. This new classifier can easily be expanded with additional features to further improve its predictive capabilities.

Machine Learning, SVM, VMS, Fisheries

1. Introduction

The Louisiana Department of Wildlife and Fisheries (LDWF) manages recreational and commercial fishing in the inland and coastal waters of Louisiana. One of the most important species fished in these waters is the eastern oyster, *Crassostrea virginica.* From 1997 to 2012, Louisiana accounted for an average of 34% of the nation's oyster landings, as well as 55% of all oysters landed among the gulf coast states in 2012 [*2014 Oyster Stock Assessment Report,* 2014]. Over 700 permits are issued each year, giving vessels access to state oyster grounds, which total 1.6 million acres. In 2012, it became law that vessel's fishing in Louisiana's public oyster grounds were required to have a Vessel Monitoring System (VMS) installed to monitor vessel activity in public grounds ["Louisiana Wildlife and Fisheries Commission Considers", 2012]. The monitors periodically record the vessel's latitude and longitude at a given timestamp. While the modules allowed the LDWF to monitor vessels' locations in public waters, they do not specify which behavior (docked/anchored, poling, fishing, or traveling) the vessel is performing at the time of the ping.

Gallegos [2014] predicted the behavior of a vessel using its data from the installed VMS module. By running comparisons of speed versus net speed of a vessel, the vessel's behavior at most times was predicted. This method could not account for every ping, however. About twelve percent of the points from the 2012-2013 fishing season and about seven percent of points from the 2013-2014 fishing season fell into uncertain ranges, and were marked as exhibiting unknown behavior. It was decided that a machine learning approach would be attempted to classify these points.

Machine learning is a branch of computer science concerned with training a model to produce an outcome measurement from a set of features. A set of training data is produced where the outcome and feature measurements are known for a group of objects for building a supervised machine learning model. The prediction model is built using this data as well as cross-validated, which can be used to predict the outcome for new data. Ideally, a model produced in this fashion will make accurate predictions by mapping the feature space into higher dimensions, and then separate the classes using efficiently built decision boundaries [Hastie, 2009]. This is unlike the previously designed classifier, which used a deterministic approach to exhaustively map each data-point via if-then-else conditions. To handle the identification of points with unknown behavior, a method of machine learning was implemented via a support vector machine, or SVM, with probability estimates, which would allow it to accurately determine which behavior a previously unknown point was exhibiting. The SVM implementation used was from the scikit-learn library. Scikit-learn is a Python module for machine learning built on top of NumPy, SciPy, and matplotlib, and is available for personal and commercial use under the BSD license. [Scikit-learn, 2016].

The purpose of this project was to produce a more robust classifier, that could more accurately predict vessel behavior from position-at-time data transmitted from the on-board VMS modules. The previous project by Gallegos [2014] developed a deterministic classifier, which classified all but about 7% of vessel behaviors from July 2013 to August 2014. The goal of this project was to use a support vector machine classifier, a machine learning based approach which can predict the unknown behavior by efficiently mapping the sample behaviors in higher

dimensional space, to overcome the limitations that were present in the previously developed deterministic approach.

Oyster fishing in Louisiana is generally by dredging over subtidal oyster reefs. Classifiers distinguishing fishing from other types of vessel behaviors could be used to discover oyster reefs that were unknown to the LDWF. By using data from vessel's fishing, poling, and movement patterns, one could thus map reefs that were missed by the traditional Side Scan Sonar methods used by the LDWF, as well as determine the most utilized areas of each reef. With greater understanding of fishing behavior over shorter periods of time, one could determine catch per unit effort in sacks per hour fished as opposed to sacks per day.

An automated system to determine fishing effort can bypass more error prone methods, such as physical logging by the captain of a fishing vessel. By having accurate measurements of fishing effort on-hand, the LDWF will be more prepared to protect the oysters and their habitat. Knowing what reefs are most frequently fished would allow the LDWF to respond to protect them, such as by closing the grounds to fishing vessels, or seeding the grounds with oyster cultch.

2. Materials and Methods

Overview of Methods

Figure 1 below demonstrates how the problem was approached. First, the VMS module onboard an oyster vessel produces a ping while the vessel is in the public oyster grounds of Louisiana. These pings contain the coordinates and ID of the vessel, as well as a timestamp. Pings with the same vessel ID and timestamps within 20 minutes of each other are concatenated together into a path, and the vessel’s speed and net speed at a given point are calculated from the coordinates and timestamps of the other points in the path. Points with speeds over 50 m/s are discarded as erroneous data, and the rest of the points have their speeds and movement angles calculated. The path is then passed to the previous classifier, which assigns each point to a behavior based on the point’s speed and net speed. These behaviors consist of docked, poling, fishing, traveling, and unknown. Unknown points are put aside, while points exhibiting the other four behaviors are used as training data for an SVM. Once the SVM is trained, points with unknown behavior are passed to it, and the SVM attempts to reclassify these points to the appropriate behavior, based on the combination of either the point’s speed and net speed or speed and movement angle.

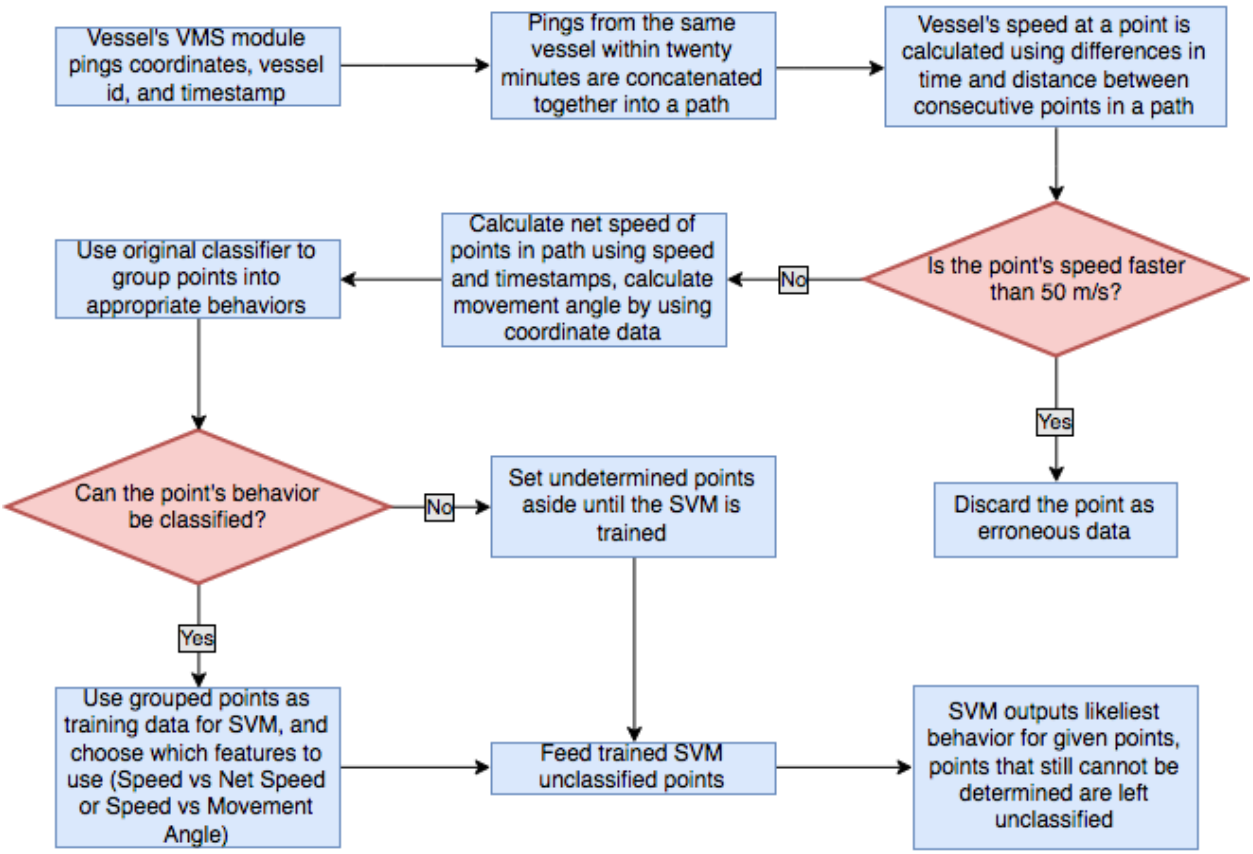

Figure 1: Flowchart of project methods

Machine Specifications

All experiments were performed on a 64-bit Windows 7 machine, with a 2.40 GHz AMD Athlon II X4 610e Processor, and 6 GB of RAM.

Study Area

Oyster growing areas of Louisiana include private leases and public grounds. About 155,800 hectares are privately leased. The public grounds encompass 667,732 hectares, of which about 24,000 hectares are oyster reef (Louisiana Department of Wildlife and Fisheries, 2012). The study area for this project is the Louisiana public oyster grounds, primarily consisting of Sabine Lake, Calcasieu Lake, Vermillion Bay, Atchafalaya Bay, Sister Lake (also known as

Caillou Lake), Barataria Bay, Lake Borgne, and Breton Sound. The seed grounds of each are outlined in purple in Figure 2 and Figure 3 below.

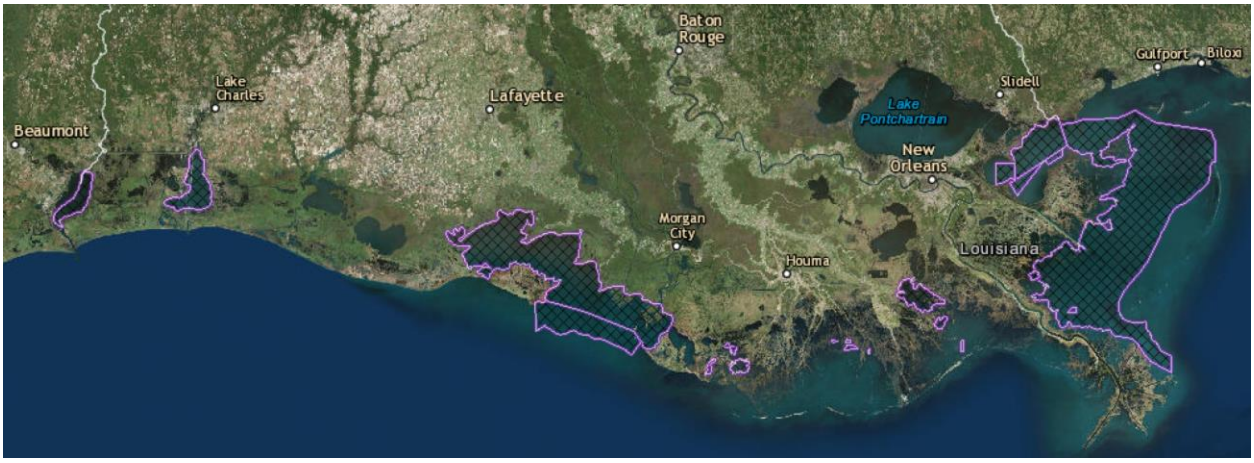

Figure 2: Public oyster grounds of Louisiana

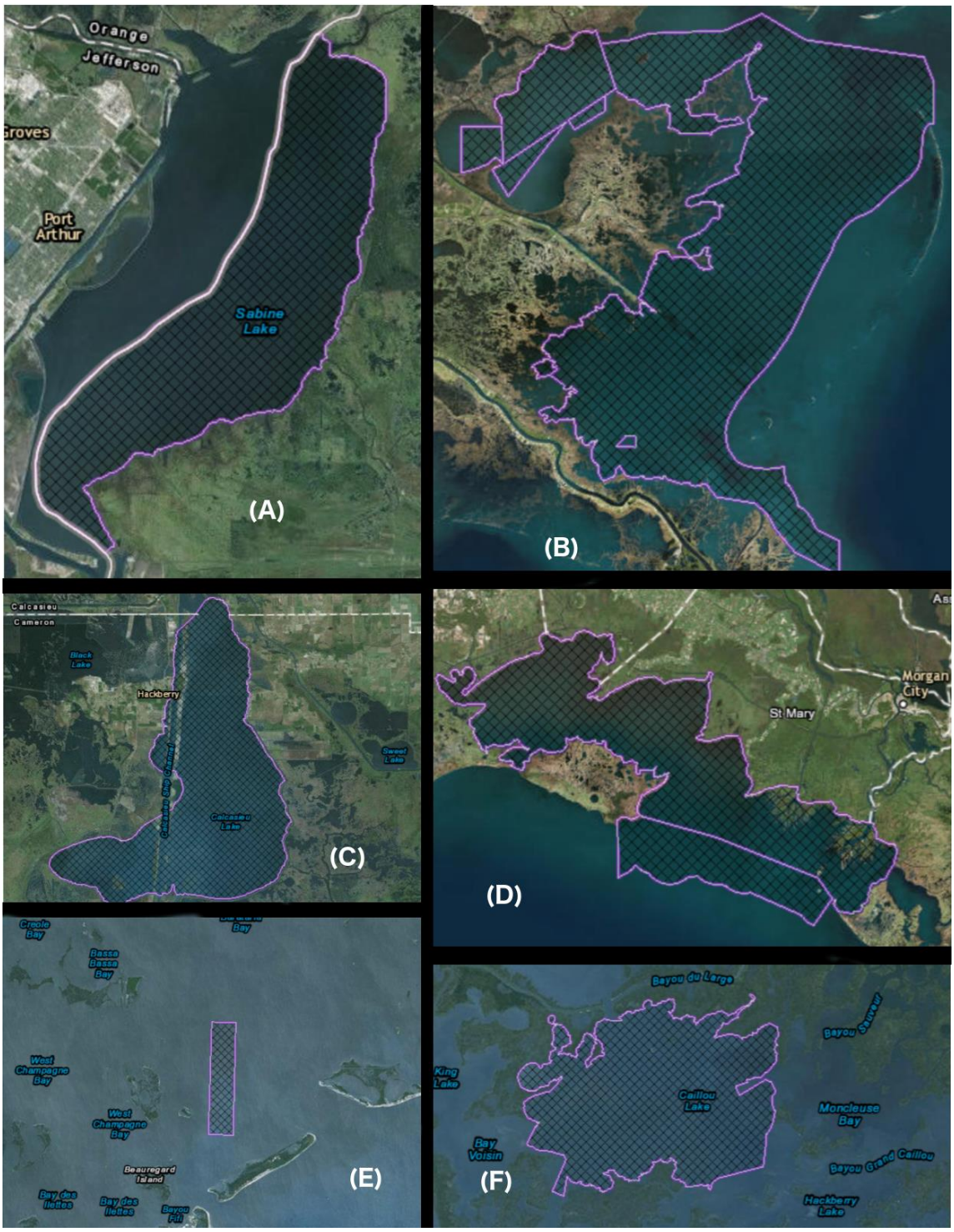

Figure 3: Detail shots of primary oyster grounds. In order from left to right, top to bottom: Sabine Lake (A), Breton Sound and Lake Borgne (B), Calcasieu Lake (C), Vermillion and Atchafalaya Bay (D), Barataria Bay (E), and Sister Lake (F).

VMS Modules

The data for the project was provided by the on-board VMS modules. These modules were provided free-of-charge to vessels fishing in the public oyster grounds of Louisiana, with the ship's operator only needing to provide minimal maintenance to the module [“Louisiana Wildlife and Fisheries Commission Considers”, 2012]. Three features were used to determine the behavior of a vessel: the vessel's speed, net speed, and movement angle. A vessel's VMS module pings approximately once a minute, giving a vessel ID, a timestamp, and a set of coordinates.

Vessel Paths

Pings from the same vessel that are less than twenty minutes apart are concatenated together into a trip, or path. Vessel paths show the movement of a vessel over time, and can provide insight into what the vessel was doing at a given point. Figure 4 below shows two idealized movement patterns of oyster vessels, consisting of five points each. The top path (A) shows linear movement, where the last point in the path is far away from the origin. The bottom path (B) shows circular movement, where the first and last points are close to each other. Linear movement is associated with vessels that are traveling. Circular movement is associated with vessels that are poling and docked. Fishing can occur during either movement pattern, but is more common during circular motion. Determining a vessel’s movement pattern was based on several factors: the vessel’s speed, the vessel’s net speed, and the vessel’s movement angle.

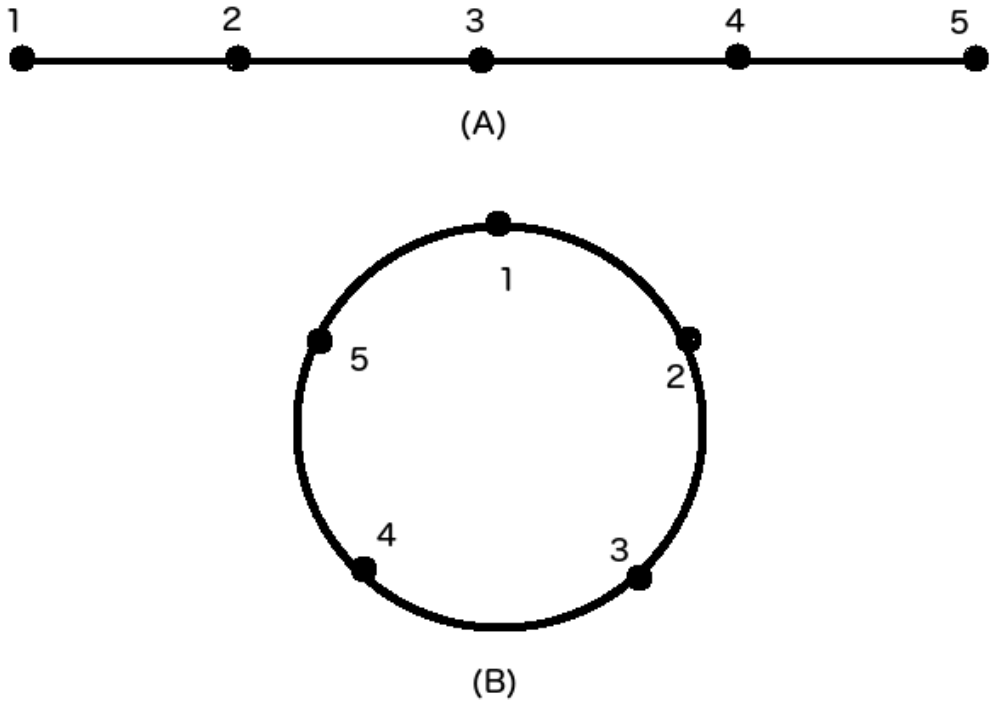

Figure 4: Two idealized vessel movement paths, linear (A) and circular (B).

Speed

A vessel's speed was the first factor used to identify vessel behavior patterns. Speed was calculated by measuring the distance between two consecutive points in a path, over the difference in seconds between the last point's VMS ping and the first point's VMS ping. The speed of vessel at a point helps to determine if the vessel was performing an activity or traveling.

Data for vessel speeds during certain behaviors was gathered via interviews with ship captains [Gallegos, 2014]. If a point has a speed under 0.1 m/s, the vessel is docked. If a point has a speed between 0.1 m/s and 0.5 m/s, the vessel is probably poling for an area to fish. If a

point has a speed greater than 0.5 m/s, the vessel could either be fishing or traveling. If a point has a speed greater than 2 m/s, then the vessel is likely traveling.

Net Speed

A vessel's net speed was the second factor used to identify a vessel's behavior at a point. Net speed was calculated using a sliding window of ten points. The distance between the last and first points of the window over the difference in seconds between those two points would be a single net speed value. Each point in the window has this net speed value appended to its list of net speeds, and the window would slide forward by one point. This process was repeated until the sliding window reached the last point in a path. For each point in the path, the net speed was calculated by averaging its list of net speeds.

There can be significant overlap between the net speeds associated with different behaviors, so speed must also be taken into account when using net speed to identify behaviors. Points with speeds and net speeds below 0.1 m/s likely indicate docked or anchored behavior. Points with speeds of less than 0.5 m/s and net speeds less than or equal to their speed plus 0.125 m/s are probably poling. Points with speeds between 0.5 and 1.75 m/s, and net speeds less or equal to their speed minus 0.125 m/s, are probably fishing. Points with either speeds greater than or equal to 0.5 m/s and net speeds greater than their speed plus 0.125 m/s, or speeds greater than 2 m/s and net speeds greater or equal to their speed minus 2 m/s, are probably traveling.

Movement Angles

While the original classifier used speed and net speed for its behavior calculations, the new classifier was implemented using movement angles in place of net speed. The reasoning behind this decision was that movement angles were easily calculated, and the delineation between circular and straight movement would be clear.

The points in a path derive their movement angles from their adjacent points. In a path consisting of 5 points, A, B, C, D, and E, we calculate B's movement angle by treating A, B, and C as the corners of a triangle. Below (Figure 5) is a visualization of this triangle, where A, B, and C are the sequential points of the path, and a, b, and c are the distances between those three points.

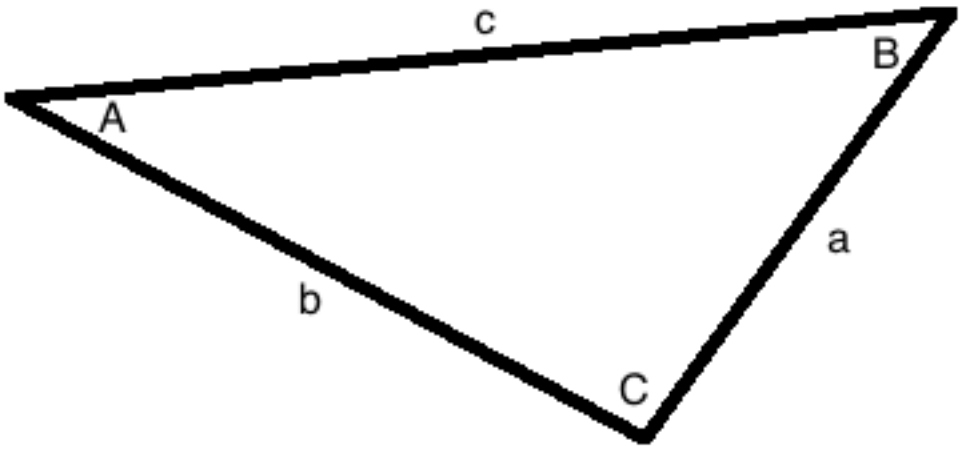

Figure 5: Triangle representing three sequential points in a theoretical vessel path.

The coordinate positions of these three points are known, so the distance from each point to each other point can be calculated. With these distances as the sides of the triangle, the Law of Cosines ["Law of Cosines", 2016] is used to calculate B:

$$B = \cos^{-1}(c^2 + a^2 - b^2 / 2ca) \quad (1)$$

This process is repeated for points C and D in the example path. Because point A has no previous point, and point E has no next point, their movement angles cannot be calculated. Instead, A's angle is set to the angle of the next point, B, and E's angle is set to the angle of the previous point, D.

Like net speed, movement angle alone cannot determine a vessels behavior, as each behavior can exhibit movement angles between 0° and 360°. However, vessels that are traveling should generally have movement angles between 150° and 210°, making the delineation between those two behaviors more clear. If a point has a speed either greater than 2 m/s and a movement angle around 180°, it is most likely that the vessel is traveling. If a point has a speed between 0.1 m/s and 0.5 m/s and any movement angle, the vessel is probably poling for an area to fish. If a point has a speed between 0.5 and 1.75 m/s and any movement angle, the vessel is likely going in circles over an oyster reef. If a point has a speed under 0.1 m/s and any movement angle, the vessel is docked or anchored.

Features

Points in the dataset have six features: latitude, longitude, timestamp, speed, net speed, and movement angle. Latitude, longitude, and the timestamp are provided by the VMS modules, while speed, net speed, and movement angle are calculated from those values. These six features were provided to the original classifier, in order to generate the behaviors for the training and testing sets of data for the SVM classifier.

Previously Built Classifier

In the previous system, a simple classifier determined a vessel's behavior. This classifier used a deterministic if-then-else block to group points into the likeliest behavior, based on the combination of a point's speed and net speed. The results from this classifier were used as training data for the SVM classifier later in the project. A vessel's speed at a point was determined by measuring its distance from its previous point by using the haversine formula. The haversine formula is a method to compute the distance between two points on the surface of a sphere. For two points on a sphere, the distance between them is represented by the formula:

$$hav(d/r) = hav(lat2 - lat1) + cos(lat1)cos(lat2)hav(lon2 - lon1) \quad (2)$$

where ***hav*** is the haversine function:

$$hav(theta) = sin^2(theta/2) = (1 - cos(theta))/2 \quad (3)$$

d is the distance between the two points, and r is the radius of the sphere. In this case, d is the distance between two consecutive pings of a VMS module, and r is the radius of the earth, approximately 6,371 kilometers.

Support Vector Machines

The classification of data point behavior was handled by using a support vector network, also known as a support vector machine, or SVM. An SVM uses maximum-margin affine hyperplanes to separate data into distinct groupings, allowing it to predict the behavior of new data based on which grouping that data falls into. Methods to classify via support vectors had existed before, but the current incarnation of support vector networks were introduced by Cortes and Vapnick [1995].

In a data set of two linearly separable classes, there are hyperplanes, which are the planes separating the classes. The optimal hyperplane in a support vector machine is defined as the linear decision function with the maximum margin between the vectors of two classes. For this reason, the optimal hyperplane is also known as the maximum-margin hyperplane. The points that determine the size of the margin from the maximum-margin hyperplane are known as support vectors. Support vectors are significant because they determine the size and position of the optimal hyperplane, while other vectors have no effect on the optimal hyperplane. Having the largest possible margin prevents noise and random error from reducing the accuracy of the classifier.

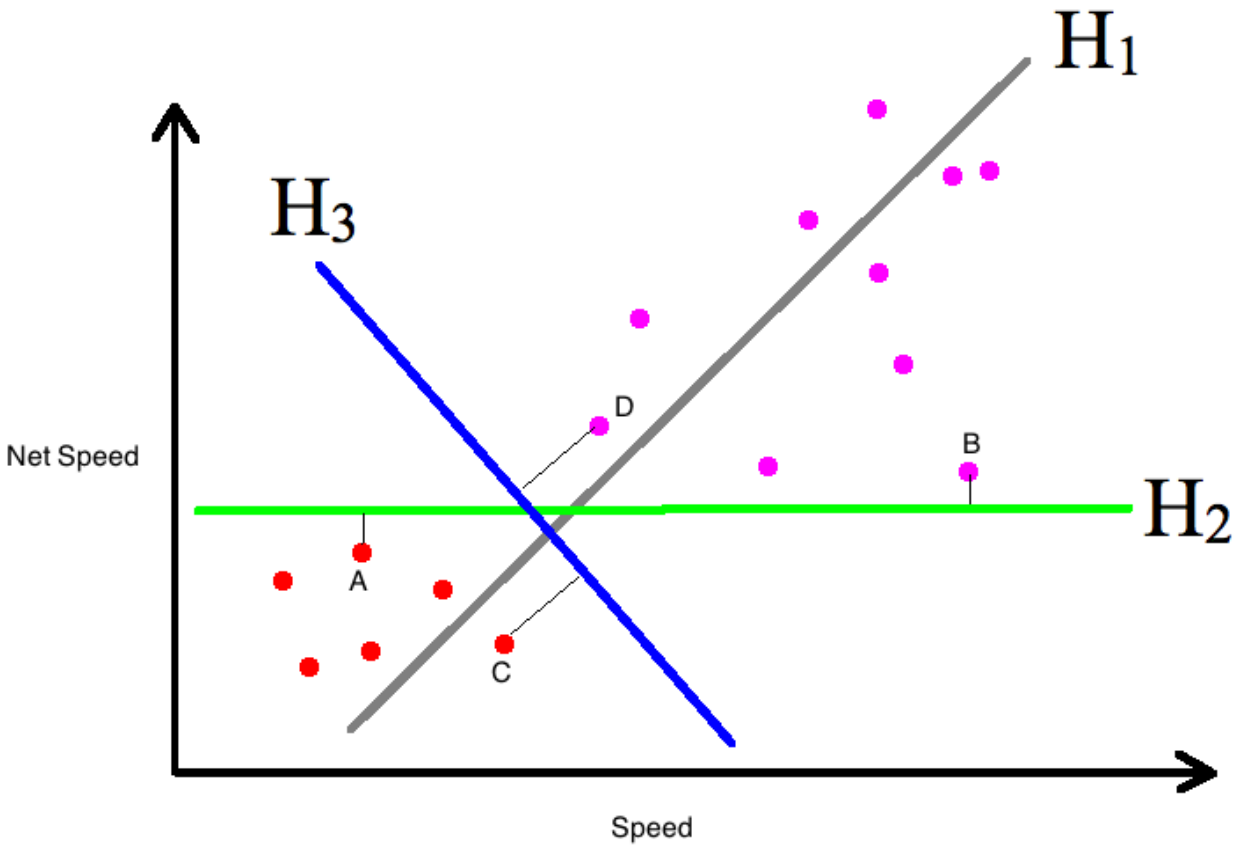

Figure 6: Graph depicting the linearly separable case of two classes of vectors

Figure 6 is a simplified graph showing poling and traveling points, and demonstrates a linearly separable case. There are two classes, poling, labeled with red points, and traveling, labeled with magenta points, as well as three planes, H_1, H_2, and H_3. The plane H_1 does not separate the classes, but H_2 and H_3 do. Points A and B are the support vectors of H_2, and C and D are the support vectors of H_3. While H_2 separates the classes, H_3 separates the classes with the largest possible margin, thus H_3 is the maximum-margin hyperplane.

The best hyperplane is one that has the largest margin and correctly separates all classes. However, there are cases where a linear hyperplane cannot do both optimally. In this case, there is a way to relax the constraints so a maximum-margin hyperplane can be found. The cost parameter, C, which affects whether our SVM features a "hard" margin or a "soft" margin, controls the trade-off between complexity of the decision rule and frequency of errors. Greater values of C increase complexity of the decision rule but reduce the frequency of errors. Smaller values of C create larger margins, while large values of C create smaller margins [Hastie, 2009]. In short, higher C values cause the classifier to prefer hyperplanes which more correctly separate the classes, while lower C values cause the classifier to prefer hyperplanes with larger margins between the two classes. In the case that two classes cannot be completely separated, both vectors on the margins as well as vectors on the wrong side of the maximum-margin hyperplane are support vectors.

Additionally, if two classes are not linearly separable, a radial basis function, or RBF, kernel can be used to alleviate the situation. The value of a radial basis function depends on a point's distance from a center. In this case, a "center" is a point in the training data, while a "point" is a point in the testing data. A point's proximity to a center is what determines

which class it will be classified as. In addition to the cost parameter C, they possess an attribute, gamma, which affects the area of influence of each training example. Larger values of gamma reduce the radius of influence of a single vector. If the value of gamma is too high, the radius will only include the support vector itself, leading to overfitting. If the value of gamma is too small, the radius of influence will encompass all the training data, and will not reveal any pattern in vector behavior. Using an RBF kernel, input data that is not separable can be mapped into a higher-dimensional feature space where the data is separable [Hastie, 2009].

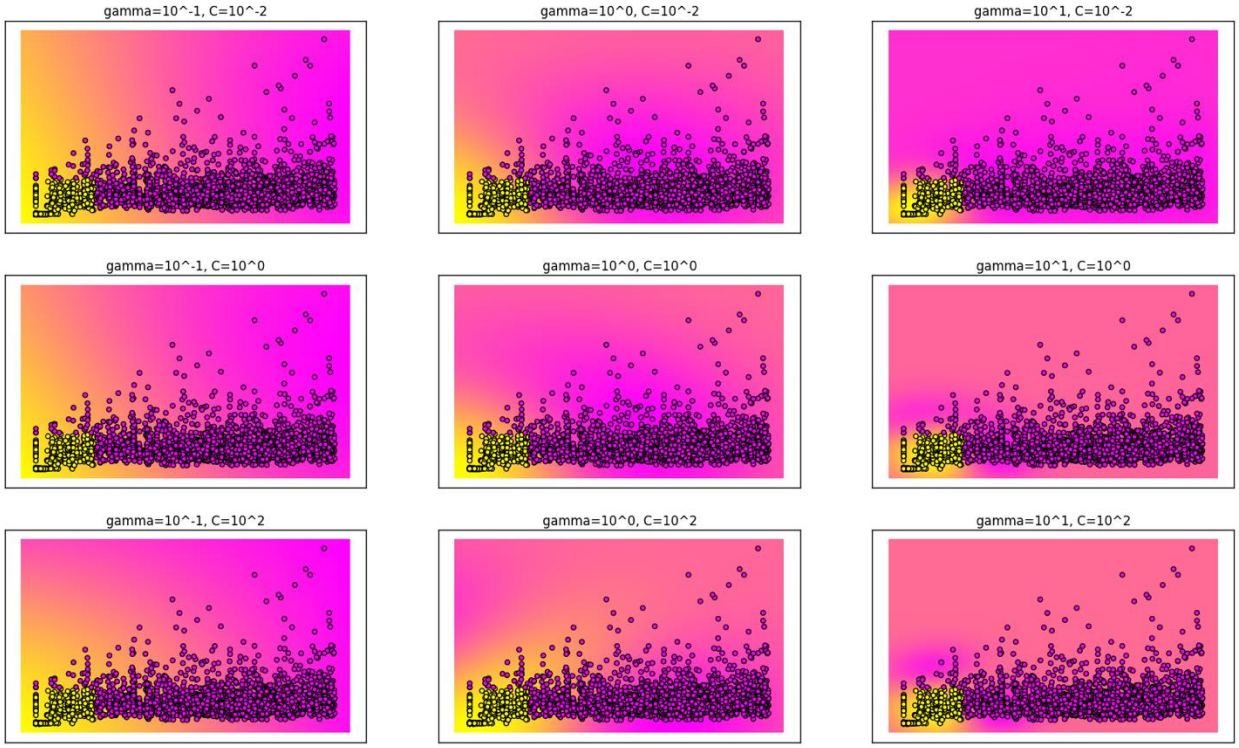

Figure 7: Effect of gamma and C on RBF SVMs

The above image (Figure 7) was made using the scikit-learn library and demonstrates how C and gamma affect the decision function of an RBF SVM. These SVMs feature two

classes, fishing and traveling points, marked by yellow points and magenta points, respectively. Each point's area of influence shown by an aura of the corresponding color. Low gammas produce large areas of influence, while large gammas create small areas of influence. Low values of C use few support vectors and make the model's decision surface smooth, while large values of C increase the complexity of the model by allowing it to select more support vectors ["Scikit-learn", 2016].

In the case where more than two classes are being considered, an SVM can divide a problem into a series of two-class problems. There are several approaches to this binary classification system, including one-vs-all and one-vs-one. In one-vs-all, a classifier is produced for each class, where the first class is the class in question, and the other class is a combination of every other class. New data is tested against each classifier, and the one with the greatest output determines which class the new input is sorted into. In one-vs-one, a classifier is made for each combination of two classes. Each classifier casts a vote as to which class the new input belongs in, and the class with the most votes wins [Hastie, 2009]. Scikit features implementations of both methods, but we use a one-vs-one scheme for multi-class classification, as it is the default implementation for the SVC library ["Scikit-learn", 2016].

For linearly separable classes, a linear kernel SVM should calculate faster and more accurately than an RBF kernel SVM. For non-linearly separable classes, RBF kernels should function better. Planes of speed and net speed are not linearly separable, so an RBF kernel should perform better in that case. Planes of speed and movement angle are separable, at least between all behaviors except fishing and movement, so a linear kernel should perform better, at least in separating docked, poling, and fishing behaviors from each other.

Cross-Validation

Estimating the prediction error of the classifier was done by using cross-validation, a testing method that is both simple and widely used. In cross-validation, a portion of the data is set aside to be used for testing while the rest of the data is used to train the classifier. This way, the classifier can be tested on data that it has never before encountered. Cross-validation is performed in folds, where the data is segmented into K parts which are each about equal size. When a fold is set aside for validation, the other K – 1 folds are used to train the validator. This process repeats until each fold has been set aside once for validation, and then the K estimates of error are combined to find the average [Hastie, 2009].

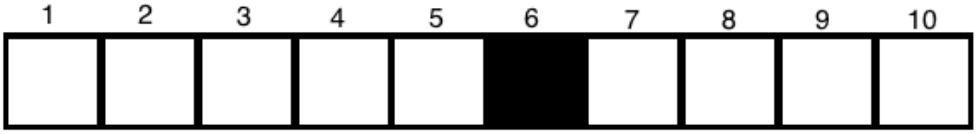

Figure 8: Figure demonstrating data set partitioned into 10 folds

Figure 8 demonstrates 10-fold cross-validation. Say each of the ten squares above represents one-tenth of our data set. For training, we will use squares one through five, and seven through ten. Once the SVM is trained, we will take our data from square six, filled in with black in the image above, and use it as testing data. Once testing is complete for this iteration, the SVM's accuracy is recorded for this configuration, another square is set aside as testing data, and square six is used as training data in all further iterations. This process will repeat until each of our ten squares has been used exactly once as testing data.

In our data set, there are five distinct behaviors to consider: unknown, docked, poling, fishing, and traveling. Points with unknown behavior are not inputted into our validator at this stage, as we do not want the validator to produce unknown points. Of the other four behaviors,

we have an abundance of fishing and traveling points, a smaller amount of poling points, and comparably few docked points. There were concerns that due to the comparably large volume of poling points in comparison to docked points, our classifier would discriminate against docked points and label them as poling, since the ranges of their behaviors were so similar.

To counteract this unwanted behavior, when the folds are prepared, only paths that have at least a single docked point are gathered. Docked points are still a minority, but now they make up a larger portion of the data, and are more likely to be correctly classified. However, if the folds happened to be grouped in such a way that all the docked points ended up in a single fold, it could greatly impact the classifier's ability to correctly predict docked behavior.

To circumvent this occurrence, stratified K-folds were used to partition the data into about even sets. In regular K-folds, there is no concern for what data points are grouped into which fold. Therefore, it is possible that an entire class could end up within a single fold, which could skew the accuracy of the classifier if it has never encountered the segregated class. Stratified K-folds split the data such that each fold has about an equal number of each class as every other fold. This way, the classifier is guaranteed to have encountered every class before it ever has to predict them.

With the folds stratified, a way to determine what value of K to choose was needed. If K is equal to our number of samples, a case known as leave-one-out cross-validation, the error will be very low, but the variance will be great. In addition, leave-one-out is computationally intensive, especially with a large dataset to test. For our data set, we chose to use 10 folds, which is widely accepted as a reasonable compromise for number of folds [Hastie, 2009].

Windowing

Windowing was also used to improve the accuracy of the SVM. Instead of single points, a "window" of points was fed to the SVM. For example, if there is a path P with points A, B, C, D, and E, and the window size is three, the SVM would be given the following windows: [A, B, C], [B, C, D], [C, D, E]. The window sizes used in testing ranged from one to fifty. Paths that had fewer points than the window size were not fed to the SVM.

Using windows as opposed to individual points rapidly increases the complexity of our testing. For example, if we have 10,000 points and a window size of one, we have 10,000 points to consider. If the window size is increased to three, we now have (10,000 – (3 – 1)) * 3 = 29,994 points. At window size 15, we have (10,000 – (15 – 1)) * 15 = 149,790 points. At 45, we have (10,000 – (45 – 1)) * 45 = 448,020 points. Increasing the window size by one increases the number of points to consider by an amount about equal to the number of points in the dataset. The formula for the number of points to be fitted by the SVM is

$$\mathbf{(p - (w - 1)) * w} \qquad \mathbf{(4)}$$

where p is the total number of points in the dataset, and w is the window size.

Grid Searching

There are a number of parameters which affect the SVM's behavior. In the case of a linear kernel, there is one value, C, which affects how harshly the SVM penalizes miscalculated data. An RBF kernel has a gamma value, which affects the area of influence each data point exhibits, as well as a C value. To find the best values for these parameters, a method called grid searching was used.

In grid searching, an array of values is searched for those which give the best accuracy. In the event there are multiple parameters, such as when using a RBF kernel, the parameters are grouped together in every possible combination. For example, if there are values C = [a, b] and gamma = [x, y], a grid search would produce the sets [a,x], [a,y], [b,x], and [b,y]. These sets would be tested with 3-fold cross-validation on a subset of the available data. After testing all combinations, the algorithm returns the set that produced the greatest accuracy. Because each set is independent of the others, this process is easy to run in parallel, which is especially efficient as the exhaustive searching of values is time-consuming.

Smoothing

If a vessel's behavior is near the threshold of two different behaviors, there is a chance it could cross from one into the other, and back again. To handle situations like these, we turn to the method known as smoothing, where the classes of the previous and next points affect the class of the current point. If we have classes A and B, and a series of points are attributed to the classes AABAA, then the center point is not likely to actually be class B. Rather, it is likely some change in values caused the classifier to misclassify the central point. To amend the situation, we "smooth" the middle point, by assigning it to the same class as its neighbors, in this case A. If our series was AABBB, then we would require no smoothing, as there is no "wiggling" between the classes.

The longest amount of time separating the timestamps of two points in a path is twenty minutes. If a ship that is exhibiting some behavior, it is unlikely that that ship will switch from one behavior to another, and then back again. For instance, if the classifier thinks a ship is

fishing, then poling, then fishing again within a brief period, it is unlikely that the ship was exhibiting different behavior for a brief period, but more likely that the ship was adjusting its course while still exhibiting the same behavior.

Smoothing was performed on the output after the SVM had predicted the behavior of points in the path. If smoothing was done before hand, the classifier would be less reliable, as points would be incorrectly attributed to the wrong behavior, due to smoothing the points. In the context of the SVM, the behavior of the previous and next points is irrelevant, as the only attributes of note are speed, net speed, and movement angle.

Probability Estimates

Once the SVM has been trained on the known data, we still have to handle the points that the original classifier labelled as unknown. If we gave these points to the SVM, it would allocate each to its mostly likely class. However, in the event that a point's likelihood of belonging to multiple classes is equally unlikely, say if a point has a 34% chance of belonging to traveling, 33% chance to belong to fishing, 32% chance of belonging to poling, and a 1% chance to be docked, we should not mark it as traveling, because our confidence in the behavior is so low.

To handle this problem, we use SVM probability estimates to ensure the likelihood of a behavior is at least as likely as a confidence threshold we provide. The SVC class has a Boolean value “probability” which, when set to true, provides probability estimates of each class, at the expense of running slightly slower. The SVM calculates the probabilities of each class, and compares the top probability against a confidence threshold value of 0.5. If the most likely class

has a probability less than our confidence value, then we discard that point, as our SVM is not confident enough to label it as any of the classes.

Scaling the Data

If the data of an SVM is not scaled, larger values can overshadow smaller ones and skew the hyperplanes in their favor. Also, larger values can make computations more intensive, causing calculations to take more time. It is important to note that if an SVM is trained on scaled data, it would be unable to accurately predict the behavior of new data, unless the new data is also scaled in the same range as the data the classifier was trained on.

For the data points, movement angles measured between 0° and 360°, net speeds ranged from 0 m/s to about 25 m/s, and speeds ranged from 0 m/s to about 50 m/s. A scale from -1 to 1 was used for movement angles, with -1 representing 0°, and 1 representing 360°. Speeds and net speeds were likewise confined between -1 and 1, with -1 representing the slowest speed in our data, and 1 representing the greatest speed.

3. Results

Movement Angle Versus Net Speed

After extensive testing, it was found that movement angles were not as robust as originally believed. For docked, poling, and fishing behaviors, the separation between the three based on speed was clear, but all three featured movement angles ranging from 0 to almost 360. Fishing and traveling typically had overlap between their speeds, however, traveling points generally had movement angles closer to 180, providing some division between the two points. There was significant overlap between the two behaviors between about 150° and 210°, however, the SVM classifier could still reach an accuracy of 98.5%, even with contamination between the two behaviors. Below is an image showing two graphs (Figure 9) of speed vs. movement angle of vessel points. The left image shows a number of points that were classified by the original classifier, the right graph is the trained SVM's prediction of the same points. The x-axis is the speed of the vessel, and the y-axis is the movement angle. Both are scaled between -1 and 1.

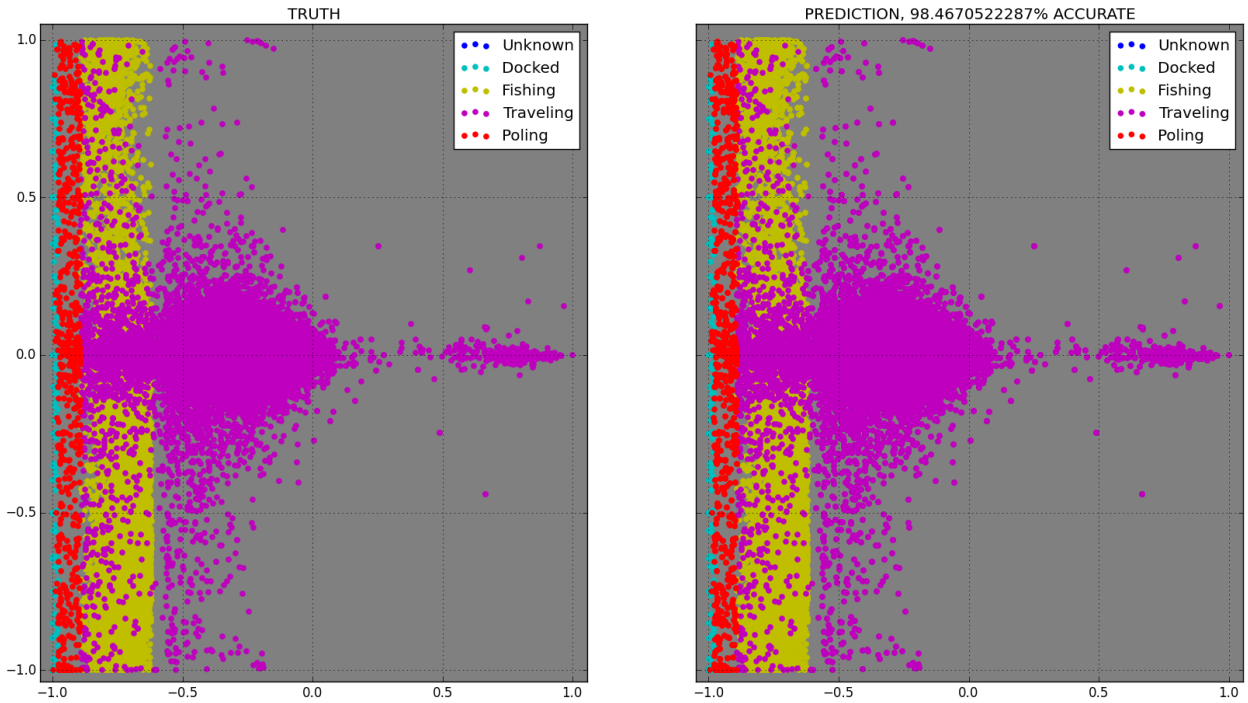

Figure 9: Graphs showing the speed, movement angle, and predicted behavior of vessel points.

Upon reverting to using speed vs net speed in the behavior calculations, it was found that the classifier reached accuracies greater than 99%, and the graphs were also cleaner, with virtually no contamination between the behaviors. The image below (Figure 10) shows two graphs of speed vs. net speed. The left graph shows a number of points that were classified by the original classifier, the right graph is the trained SVM's prediction of the same points. The x-axis is the speed of the vessel, and the y-axis is the net speed. Both are scaled between -1 and 1. Excess Traveling points have been cropped from the image, so the boundaries between the four behaviors are visible. Because of this, movement angles were discarded, and speed and net speed were chosen as the attributes to determine vessel behavior.

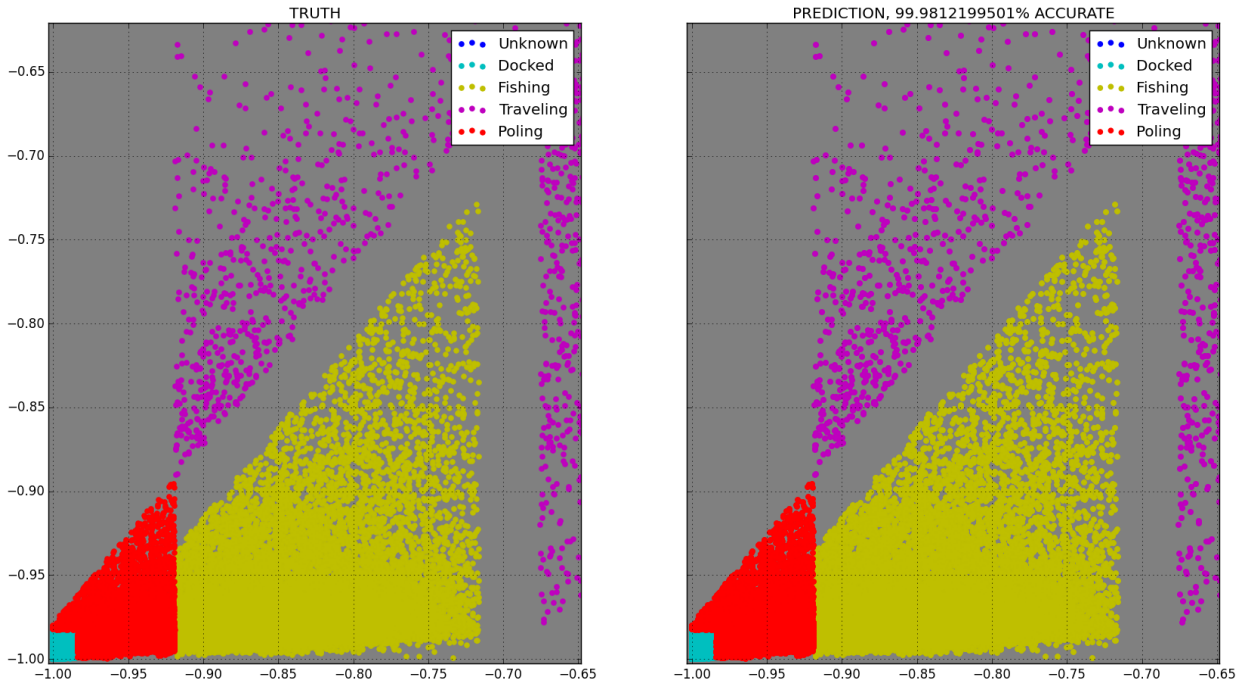

Figure 10: Graphs showing speed, net speed, and predicted behavior of vessel points

Below are graphs (Figures 11-14) of each behavior's receiver-operating characteristic curve, also known as a ROC curve, demonstrating the ratio between the detection of true positives and false positives. The y-axis functions as sensitivity, while the x-axis functions as specificity. Typically, as sensitivity increases, specificity drops, as more true and false positives are detected. The diagonal, dotted line demonstrates the classifier of a coin flip, which is a useless classifier [Ebell, 2016]. The classifiers used to make these graphs were each trained on over 1000 paths each, with C values of 1,000, window sizes of 1, and the RBF classifier had a gamma of 10. The key in the bottom right shows the area under the curve for each behavior. A slight dip can be seen for the “Docked” and “Poling” curves in each kernel’s graph, but each curve still possesses an area of approximately 1.0, meaning the classifier is almost perfectly accurate.

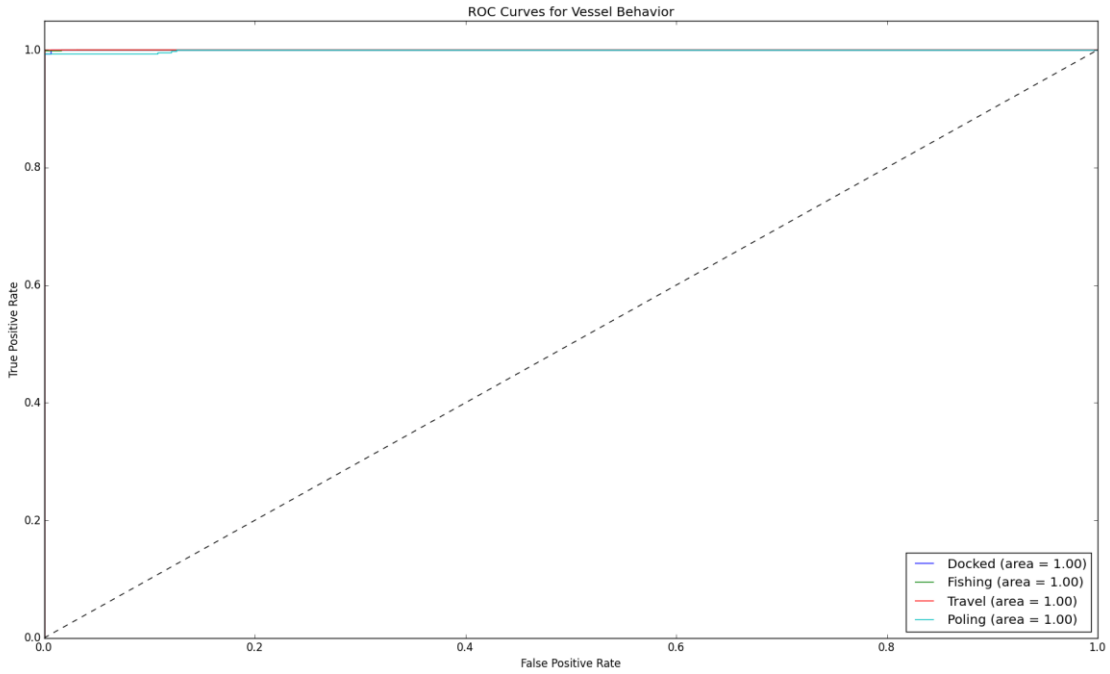

Figure 11: ROC Curve for a linear kernel

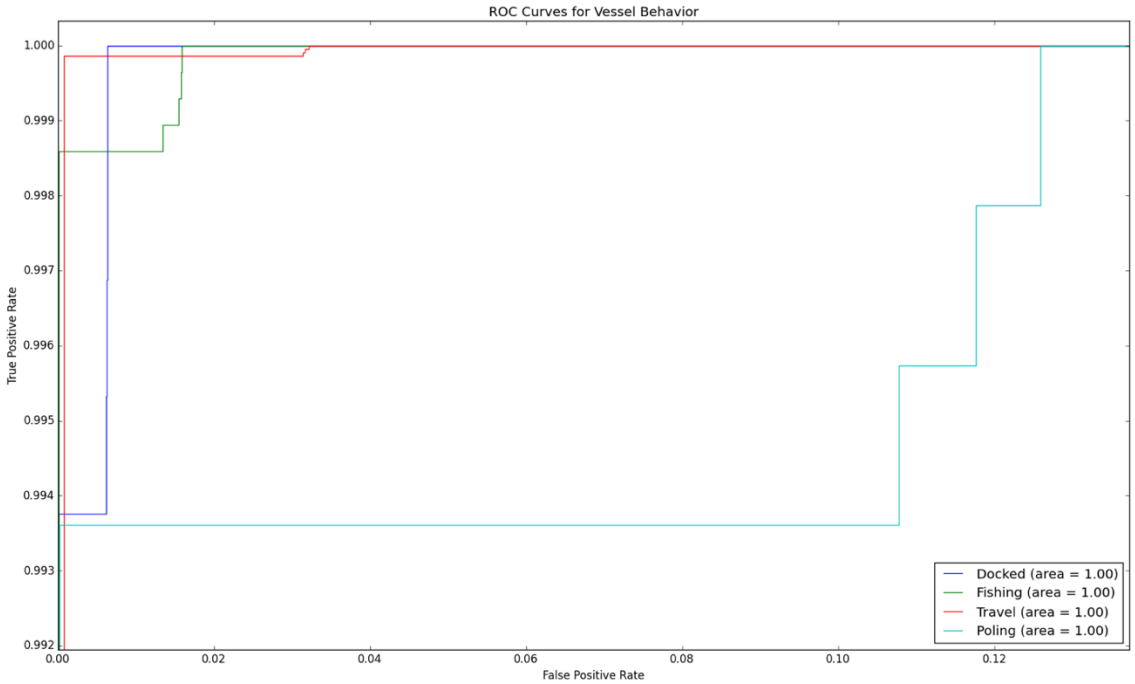

Figure 12: Close-up of ROC Curve for a linear kernel

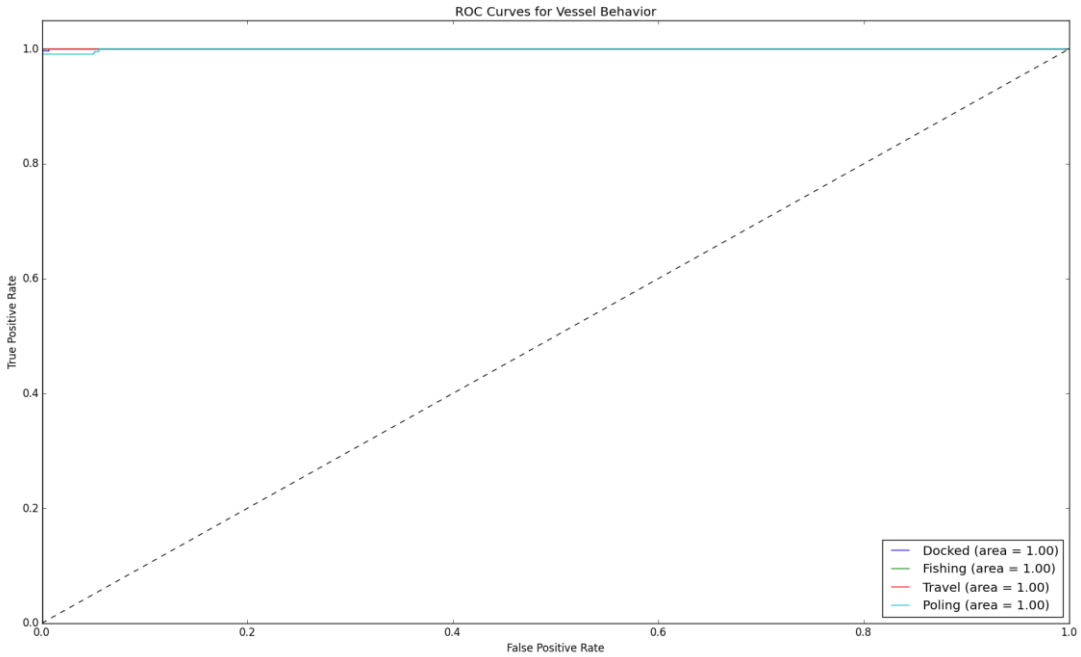

Figure 13: ROC curve for an RBF kernel

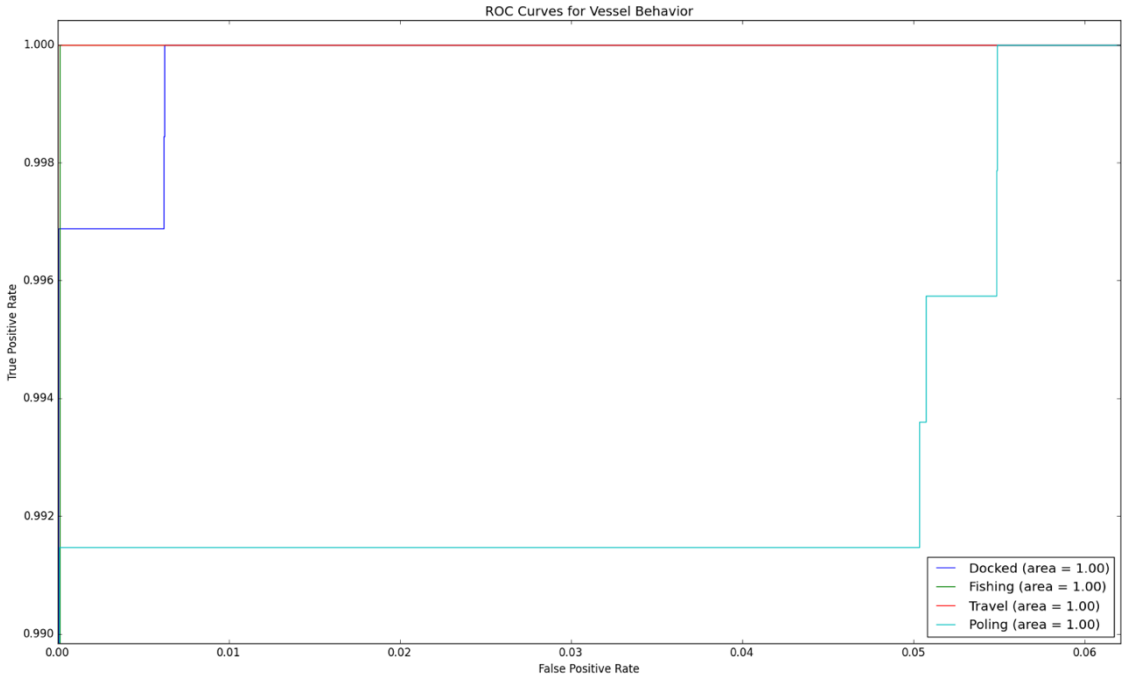

Figure 14: Close-up of ROC curve for an RBF kernel

Examining the close-ups of the ROC curves, we can see that poling and docked points have less area under their curves than fishing or traveling points, meaning the classifier is less accurate at determining their behavior than other points. This is likely because fishing and traveling points make up the majority of the training data, allowing the SVM to more accurately predict their boundaries. Still, even with comparably large dips in accuracy for those behaviors, both classifiers still predict the correct behavior of each point over 99% of the time.

Using speed, net speed, and movement angle together could possibly improve the accuracy of the classifier. However, with accuracies in excess of 99.9%, it is unlikely the inclusion of movement angles would make any significant improvement. Additionally, adding movement angles would more than likely slow down the training of the SVM, as it would need to calculate the relations between three parameters instead of just two.

Window Size vs. SVM Accuracy

Window size and accuracy have a positive correlation. As window size increases, accuracy undergoes an upward trend. Using a linear kernel with a C value of 1,000, with a dataset of only the first 250 paths consisting of 50 or more points, we see that accuracy generally increases with window size, starting at about 99.92% accuracy for window size 1, and rising steadily to 99.95% at window size 35. Below is a graph (Figure 15) showing the results.

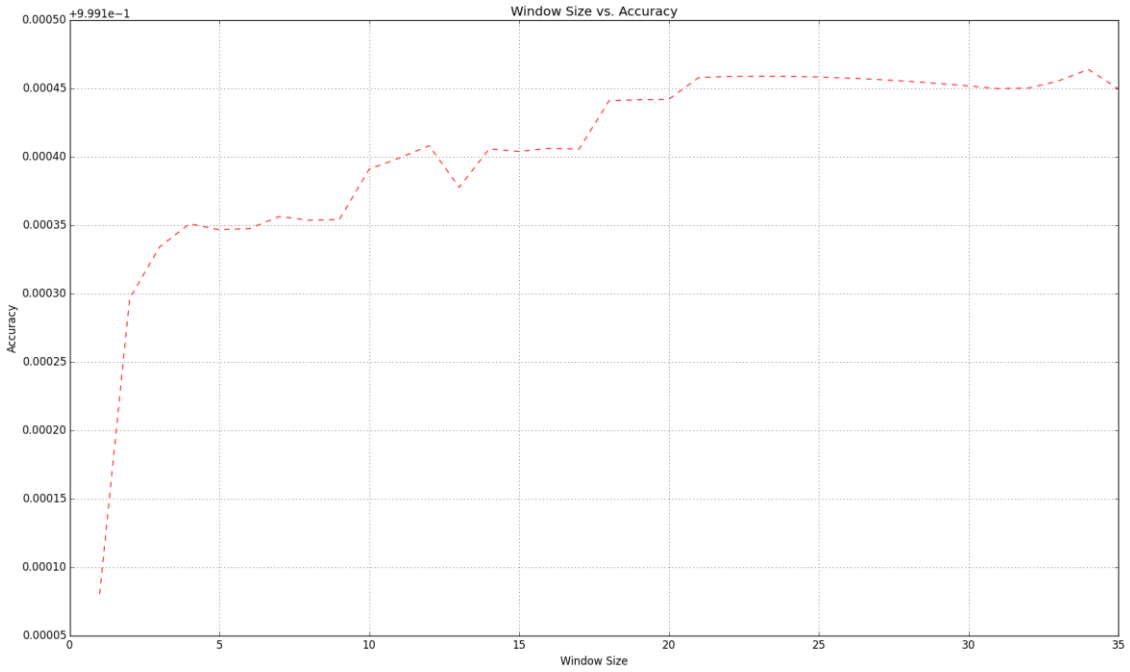

Figure 15: Window Size vs. Accuracy for a Linear SVM

With an RBF kernel with a C value of 1,000 and a gamma value of 10, using the same dataset, we see a similar trend, with accuracy starting at about 99.97% accuracy at window size 1 and rising to about 99.98% at window size 35. This data is shown in Figure 16 below.

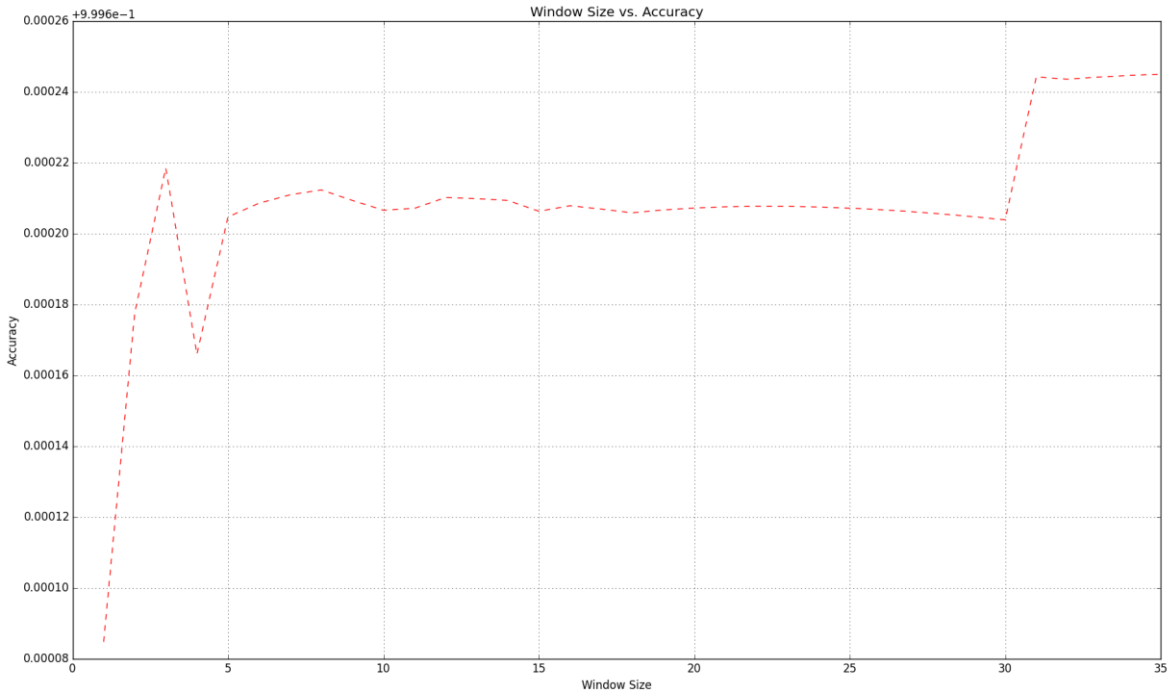

Figure 16: Window Size versus Accuracy for an RBF SVM

It can be observed that increasing window size has a generally positive effect on accuracy. However, the classifiers' accuracies only increased in amounts well under 1%, and training times increased dramatically as window size rose.

Grid Searching

Using a range from 1 to 100, It was found that the classifier was generally more accurate with larger values of C, preferring values around 98. However, accuracy gains were slight, generally only showing improvements of under 1%, and calculation times increased considerably. Adjusting the range, we attempted a grid search with the C values [0.01, 0.1, 1, 10, 100, 1000] using net speed as our feature of choice. Generally, the gap in accuracy between a C of 0.01 and 0.1 was large, over 20% for window size 1 and shrinking closer to 10% as window size grew. Between a C of 0.1 and 1, there was a 12% accuracy discrepancy at window size 1, shrinking to

about 3% as the window size grew. C's of 1 and 10 were about 1% different, and there was less than 1% difference between 10, 100, and 1000.

This process was repeated for gamma, with the range adjusted to [0.1, 1, 10]. In nearly every case, the grid search found a value of 10 to be optimal, though it was only slightly more accurate than the alternatives. Grid searching for gamma alone increased run times, but grid searching for both gamma and C simultaneously dramatically extended execution time.

Grid searching revealed that larger values of both C and gamma produced the most accurate results. Normally, having large C and gamma values can lead to overfitting of the model, where it performs well on test data but is unable to accurately predict new data. However, because the boundaries of the data were well defined, and because the SVM was trained using 10-fold cross-validation, overfitting was unlikely to be a concern.

Scaling

If an SVM is trained on unscaled data, larger values can overpower smaller ones, producing favoritism in the classifier and thus reducing its accuracy. Additionally, if values grow large, the calculations performed on them can take longer to complete.

An SVM using a linear kernel with a C of 1,000 was trained on 250 paths with over 50 unscaled points each. With window sizes from 1 to 35, it had a mean calculation time of over 11 minutes per window, and an average accuracy of about 99.95%. Using the same parameters with scaled data, the classifier had a mean calculation time of about 1.95 minutes per window, and had an average accuracy of about 99.94%. This process was repeated for an SVM using an RBF kernel with a gamma value of 10. For window sizes 1 to 35 with unscaled data, the RBF

classifier had a mean accuracy of 99.98% and an average calculation time of about 2.77 minutes. For scaled data, it had a mean accuracy of 99.96% and an average calculation time of 0.35 minutes.

It can be observed that, in the case of a linear and RBF kernel SVMs with our dataset, scaling does not significantly affect accuracy, but does greatly reduce the amount of time spent training the SVM.

Maps

Below are several maps (Figures 17-19) demonstrating the predictive capabilities of the SVM. The top map shows the previous classifier's prediction of the vessel's behavior at each point, while the bottom map is the SVM classifier's prediction of the behaviors of those same points. The maps are nearly identical, as the SVM classifier has over 99% accuracy, in relation to the previous classifier's prediction.

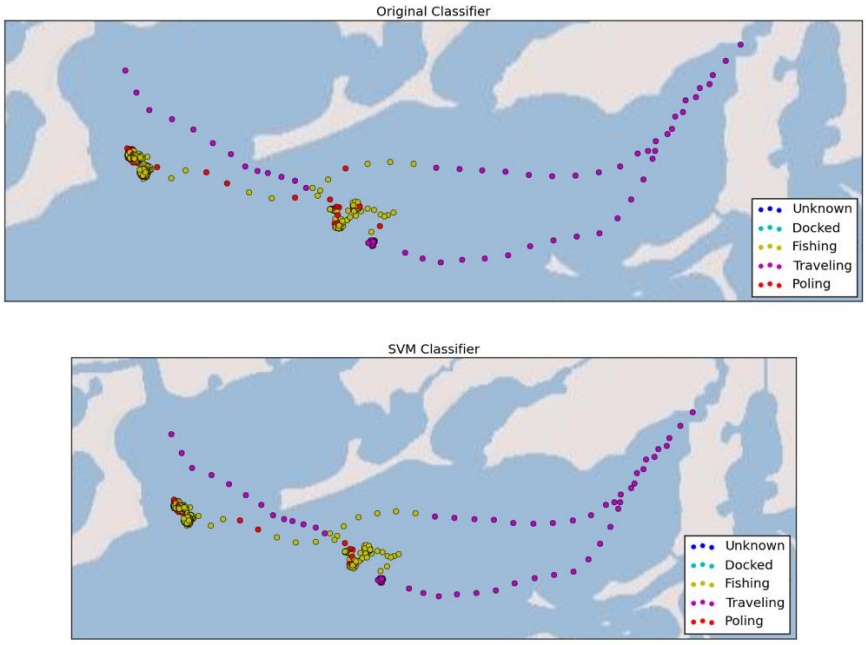

Figure 17: Previous classifier's prediction of vessel behavior versus SVM classifier's prediction of vessel behavior

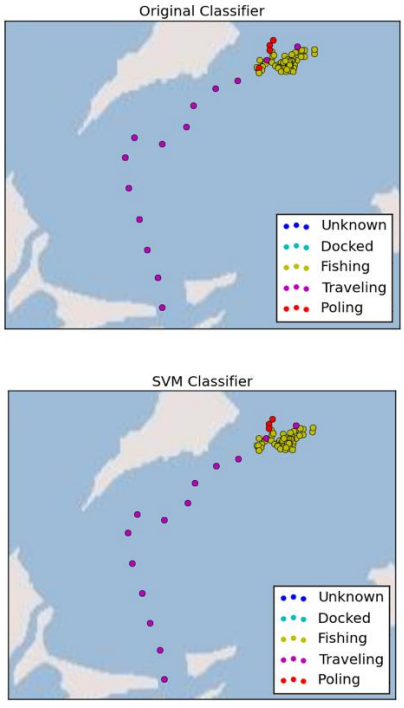

Figure 18: Previous classifier's prediction of vessel behavior versus SVM classifier's prediction of vessel behavior

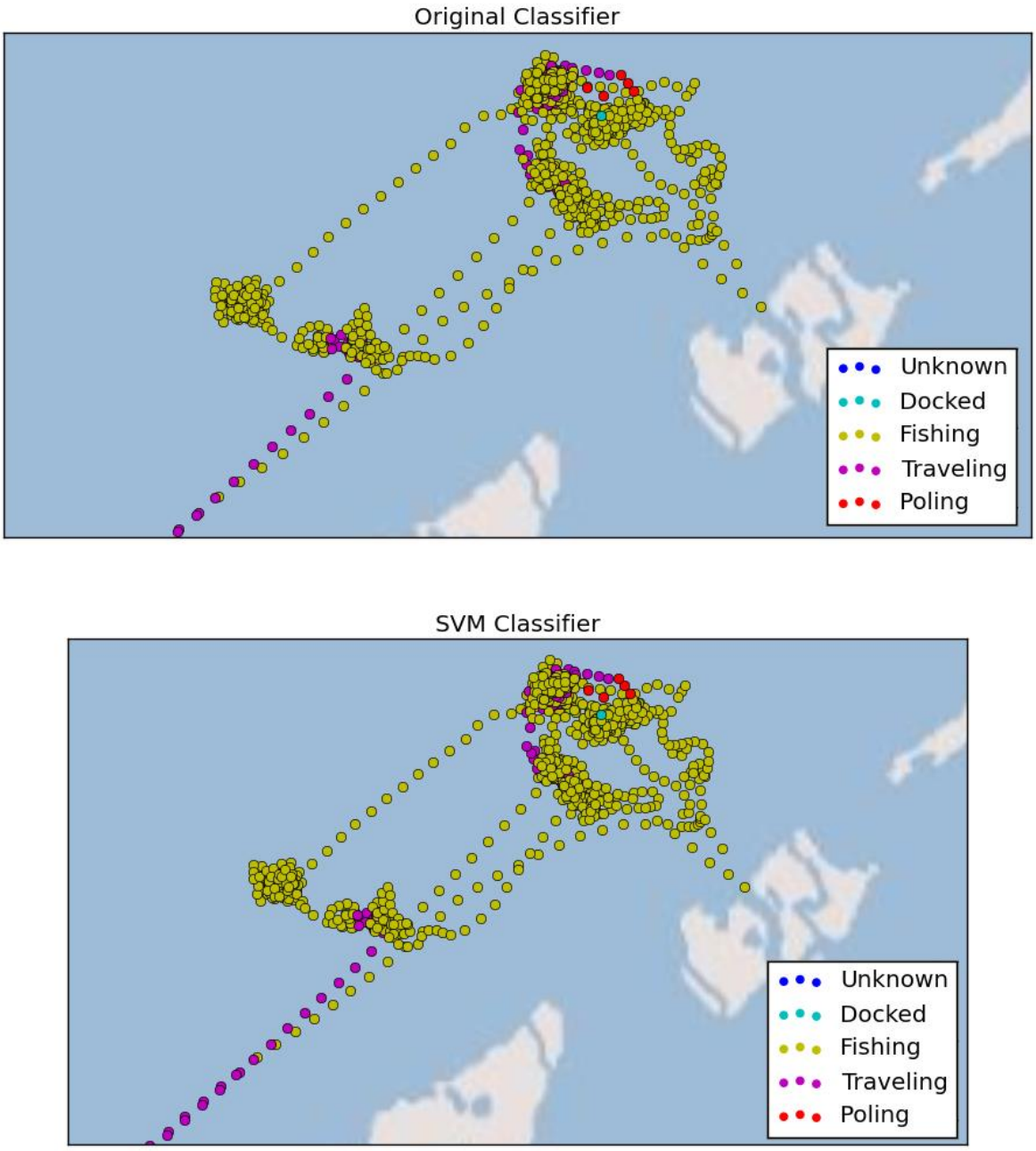

Figure 19: Previous classifier's prediction of vessel behavior versus SVM classifier's prediction of vessel behavior

Below are several maps (Figures 20-23) of unknown behavior points in the Louisiana public oyster grounds. Again, the top map shows the original classifier's prediction of the vessel's behavior, while the bottom map shows the SVM classifier's prediction. The unknown points have been reclassified to the likeliest behavior, based on a classifier with an accuracy of about 99.98%. Additionally, each of these points has to breach a 50% confidence threshold in order to be reclassified to its new behavior. These predictions are not completely indicative of the vessel's actual behavior, they are simply the classifier's best guess as to which behavior each point belongs in. As only speed and net speed are used to determine behavior, the classifier can only make an estimate of a vessel's behavior based on those features. Additional factors that could affect behavior, such as the presence of an oyster reef or distance from a dock, are not taken into account.

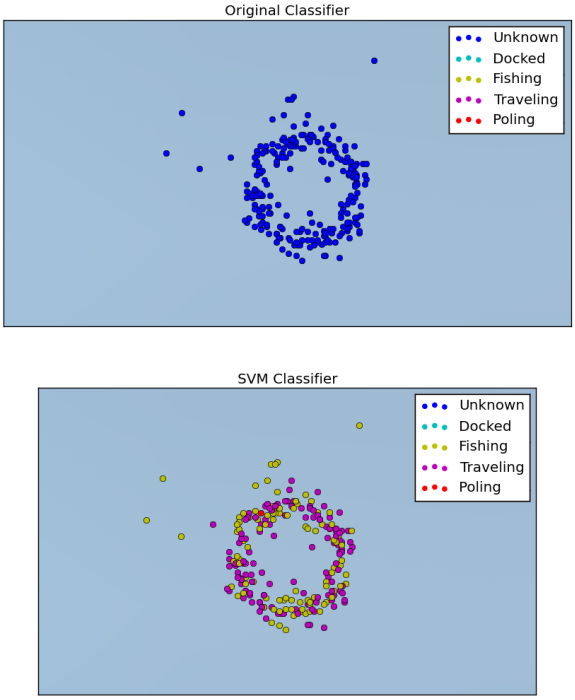

Figure 20: Unknown points for vessel operating in Sister Lake reclassified by SVM classifier

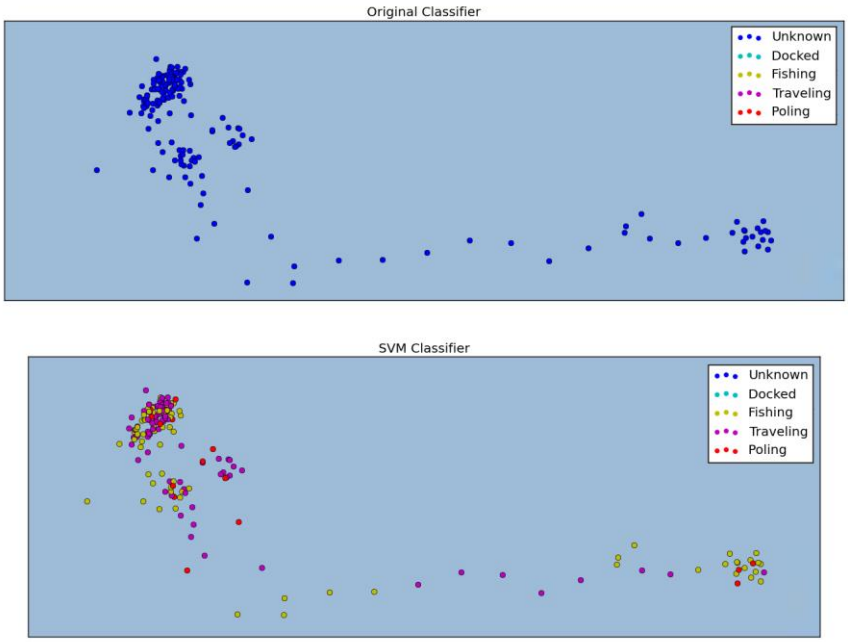

Figure 21: Unknown points for vessel operating in Lake Mechant reclassified by SVM classifier

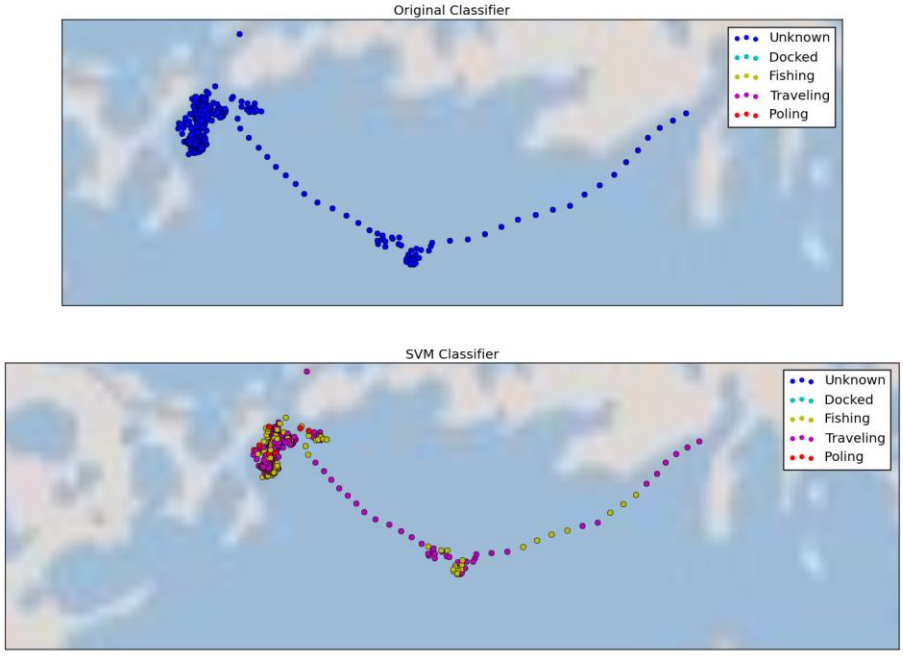

Figure 22: Unknown points for a vessel operating in Sister Lake reclassified by SVM classifier

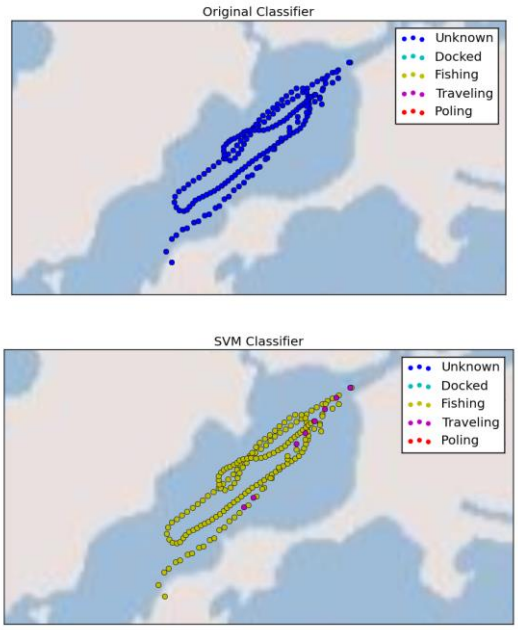

Figure 23: Unknown points for a vessel operating in Bay Junop reclassified by SVM classifier

4. Discussion

The SVM classifier is found to perform significantly better at determining behavior than the previous classifier. While the previous classifier left about 7% of behaviors from July 2013 to August 2014 undetermined, the SVM left about 0.3% of the behaviors unidentified. However, the classifier still had a number of uncertainties. For example, when reclassifying unknown points in a path, one can observe trends where a ship travels for a time, then shifts to fishing, while still maintaining its route as though it was still in motion. Without a ground truth for each vessel we cannot be completely certain in the predictions of the SVM classifier, however, its predictions are the most accurate metric we have with our current understanding of vessel behavior.

Larger window sizes produced classifiers that are more accurate, however, accuracy gains were under 1%, and large windows made training the SVM take a longer amount of time. Scaling sped up the training process, while degrading accuracy by a negligible amount. Both RBF and linear kernel SVMs performed better when using speed and net speed as attributes. In both cases, the SVMs trained faster and were more accurate when using net speed as a feature rather than movement angle. Using optimal setups, linear SVMs reached about 99.95% accuracy, while RBF SVMs reached about 99.98% accuracy.

There are a number of additions and adjustments that could improve the project. Adding a graphical user interface to the tools would make them easier to use. Using additional nautical data, such as location of docks, waterways, and oyster reefs, would improve vessel behavior predictions. If more parameters are used to determine vessel behavior, updating the classifier to use them would be easy. It would just be another dimension for the SVM, such as speed and net

speed are. The inclusion of more parameters would make training take more time, but would likely improve accuracy. A ground truth would be necessary to train a classifier to use this new data, however. Data from vessels about the value of the parameter in question during each behavior would need to be acquired.

Without adding more features to the classifier, the greatest way to improve the system would be to acquire data on each vessel's speed for each behavior. The current system uses ranges gathered from interviews with several ship captains. However, if there is variance in the speeds at which each vessel can fish or pole, then ships that do not match the data from the interviews will be grouped incorrectly. While this data could make the classifier more accurate, establishing a ground truth would also have several downsides. The database would become more complicated, as each vessel ID would need to be correlated with some set of behavior ranges. These ranges would also need to be closely checked, as an error would cascade and invalidate the classifier's predictions of an entire vessel's data. Finally, if a vessel's operating speeds are known in the system, they could be correlated with data on actual vessels to determine which vessel has which vessel ID, reducing the anonymity of the vessels.

References

Gallegos, David Xavier. *A GIS-Centric Approach for Modeling Vessel Management Behavior System Data to Determine Oyster Vessel Behavior on Public Oyster Grounds in Louisiana*. Thesis. University of New Orleans, 2014. N.p.: n.p., n.d. Print.

"Louisiana Wildlife and Fisheries Commission Considers Establishing Vessel Monitoring for the Harvesting of Oysters on Public Seed Grounds." *Louisiana Department of Wildlife and Fisheries*. N.p., 5 Jan. 2012. Web. 15 May 2016.

Hastie, Trevor, Robert Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2009. Print.

"Law of Cosines." -- *from Wolfram MathWorld*. N.p., n.d. Web. 14 Apr. 2016.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Mach Learn Machine Learning, 20*(3), 273-297. doi:10.1007/bf00994018

"Scikit-learn." *: Machine Learning in Python — 0.17.1 Documentation*. N.p., n.d. Web. 25 Aug. 2016.

2014 Oyster Stock Assessment Report. N.p.: Louisiana Department of Wildlife and Fisheries, Sept. 2014. PDF.

Ebell, Mark H., MD, MS. "Chapter 4 - Diagnostic Tests - ROC Curves." *Chapter 4 - Diagnostic Tests - ROC Curves*. N.p., n.d. Web. 13 Sept. 2016.

"Support Vector Machine." *Wikipedia*. Wikimedia Foundation, n.d. Web. 1 Aug. 2016.

Vita

The author was born in Chalmette, Louisiana. He obtained his Bachelor's degree in Computer Science from the University of New Orleans in 2013. He was employed as a graduate assistant at UNO while working on his thesis.