Chris Murdter

11/12/19

Final Report

Cargo Ship Detection Using Satellite Imagery

Shipping is responsible for 90 percent of the worlds trade according to the International Shipping Federation and is responsible for trillions of dollars in the US economy alone. As the demand for shipping cargo increases, so does the traffic around ports. More ships increases the chances of environmental accidents such as oil spills, piracy, trafficking and other illegal cargo. Many organizations around the globe such as environmental agencies or national governments work to limit these disasters and illegal activities which costs the industry around 50 billion dollars a year. Tracking these ships using satellite images can help organizations keep tabs on the activities of ships in and around ports and can also help with supply chain analysis, improve security and limit environmental disasters. Currently, ship detection is done through AIS trackers on the ship. However ships can turn these tracking devices off going “dark”. These invisible ships are most likely to be involved in illegal activities. Using machine learning and satellite imagery we can improve the tracking method of ships and keep track of them even if they go dark. In theory, improved tracking of shipping vessels can improve traffic and ship monitoring activities potentially saving billions of dollars. My interest for this project came when reading a New York Times article on how North Korea obtains its armored limousines for Kim Jong-Un. In the article a journalist follows a cargo ship using satellite images and manages to pick it up

again even after it turns off its tracker and goes dark. I was curious about how hard that must've been if the journalist wasn't using a tracking method of their own.

My goal for this project was to use python and sklearn to build an image classifier which accurately determines the cargo ship in the image data set. I have selected a data set from Kaggle that includes a network of .png images to train the algorithm as well as a test dataset to prove whether the algorithm works or not. There are 4000 images in total in this data set. This data set is very clear and does not need cleaning as each image file is clearly labeled and has been assigned a value to determine whether there is a ship in the image or not. I kept track of how accurate my algorithm is and how fast it runs to see if I can improve upon my algorithm as I go along. Currently I am using pixel values to distinguish images with ships but I had plans to include feature selection in my algorithm to improve accuracy.

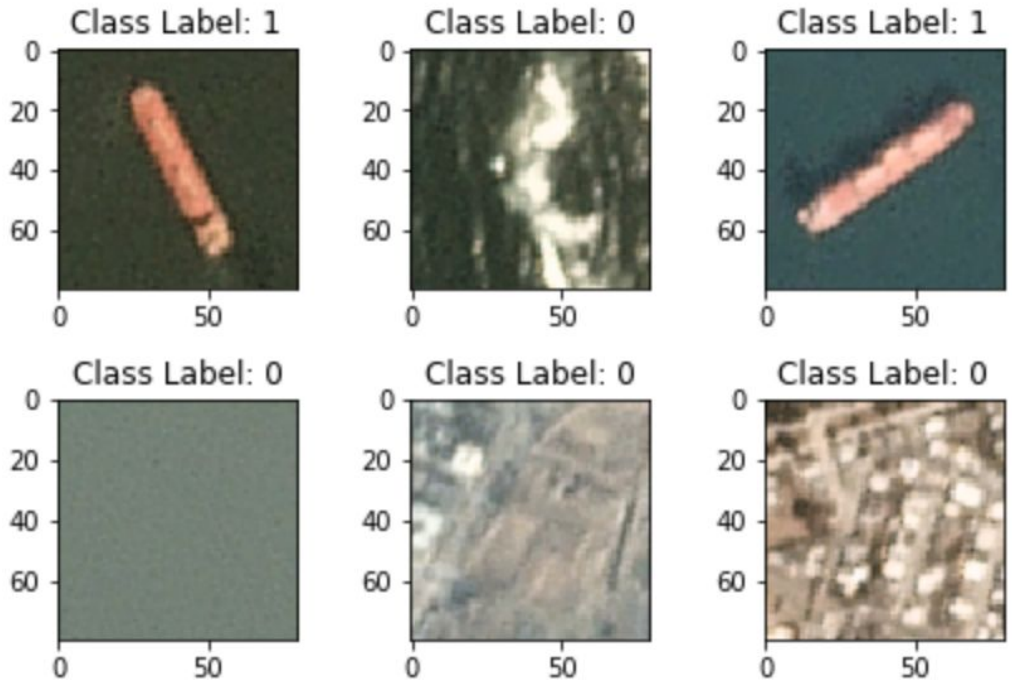
The data for my capstone project is from Kaggle, a website containing many open data sets. This data set contains images extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. Included in the data set are 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification each given a value of 1 or 0 respectively. The image files are .png files. The dataset contains four columns. The data column contains the pixel values for each image. The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue. The image is stored in row-major order, so that the first 80 entries of the array are the red channel values of the first row of the image. The labels column for each image which is a 1 or a 0 depending if the image has a ship in it or not. The locations columns has the coordinates for the image's center point and the

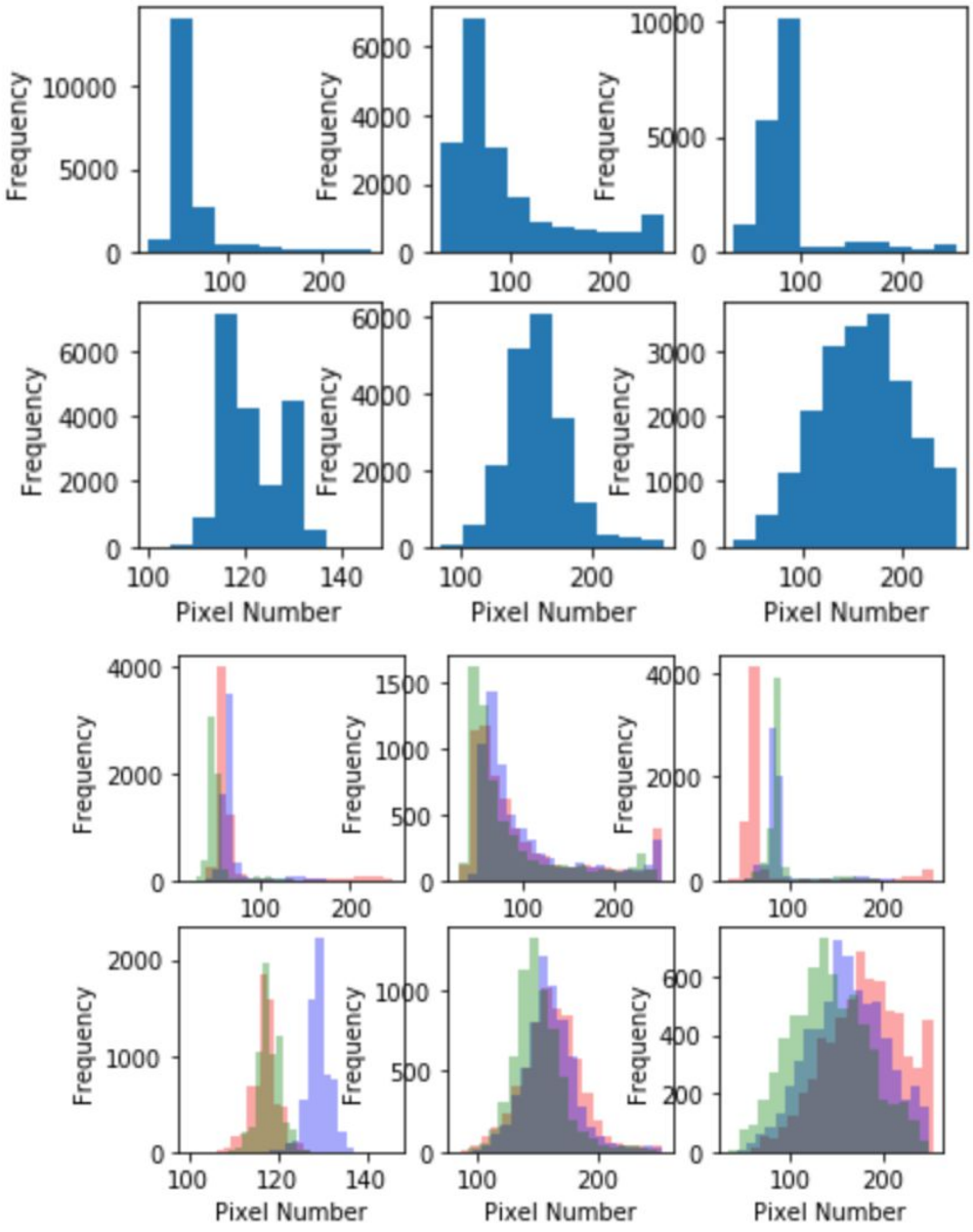
last column is the scene ID which contains the individual IDs for the images. The data columns was useful to study the differences in images.

To find any missing values I used the isna() panda function and applied it to every column. I found that there are no missing values from this data set. This is very useful and most likely because the data set has already been cleaned before being posted to Kaggle. I wasn't sure on what to look for with outliers. I wanted to include all images to have a big data set to train my machine learning algorithm as the amount of data wasn't extremely large. One challenge that came up was how to check if an image had dead/hot pixels. I was not sure how to go about this so I did not check. In the end I don't believe it had an effect on the accuracy of my classifier.

For exploratory data analysis I was looking at color pixel distribution and making comparisons based on the type of image those pixels related to. For example, a pixel distribution of an object in the water with label 0 (no ship) vs a pixel distribution of a ship in the water. Both had similar shapes when plotted as a histogram. I performed a t-test on the arrays of pixel numbers and found that the p-value was 0 or extremely close to zero. I wasn't sure what to do next since having a p-value rejects the null hypothesis. Meaning that there is a significant difference between populations. I then made boxplots of the images. But their means and shapes of boxplots were far from each other so It was hard to process what I could do with that information. Note, I did not do these tests for all images. Only the sample images that I used in the data story notebook of this capstone project. I took a group of images and compared their pixel value distributions. As I expected, images of the land were very different from images to the water. I don't believe this will be a big issue to distinguish between the two. However, I did find a non ship image that had a large wave or object in the water whose color pixel distribution

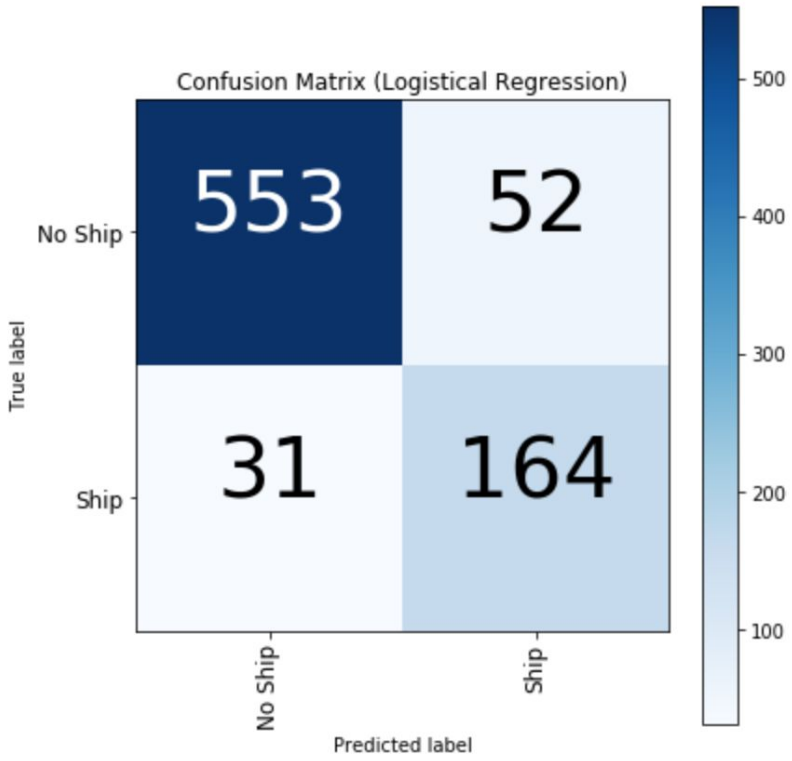
plots either looked very similar to a ship image. This led me to believe that the algorithm will have a difficult time finding false positives If i just rely on pixels/color for my algorithm. Below are the sample images that I visualized along with their pixel distributions



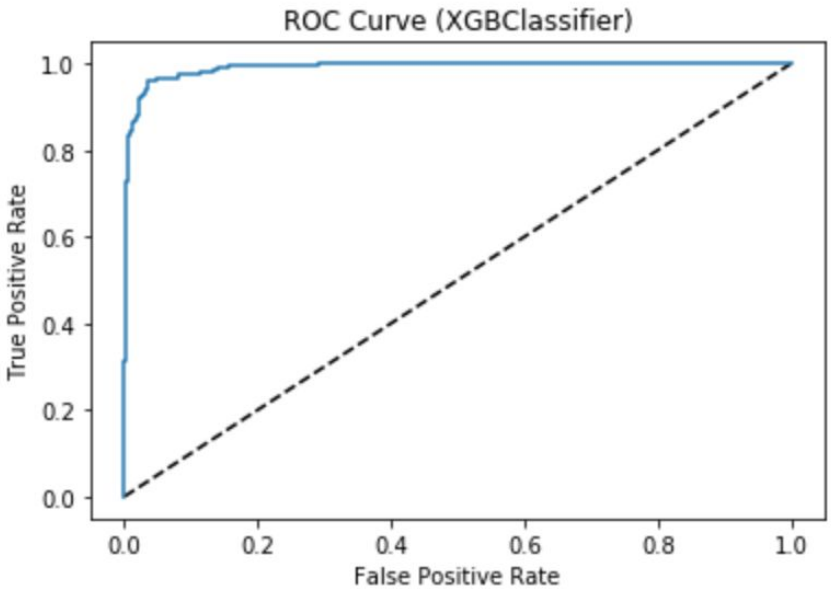


The main concern with this project was how accurately I can make my algorithm. Pixel distributions are just one aspect of understanding the data. The next step would be choosing

which model to use. I decided to focus on two different models. The first model was Sklearn's Logistic Regression model. I chose this model because logistic regression is a model that can be applied to assign observations to a discrete set of variables. logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. Applying this model was very simple using Sklearn documentation and the result was a mean accuracy score of 89.6 percent. Pretty good in my opinion given the simplicity of the model. I then made a confusion matrix to evaluate how many images the model managed to get right or wrong. The confusion matrix for the Logistic Regression Model showed that 553 "no ship" images and 164 "ship" images were labeled correctly. There were 31 false negatives meaning the model classified a "ship" image as a "no ship" and 52 false positives meaning the model classified a "no ship" image as having a ship in it. I expected there to be many false positives during my data analysis and some false negatives. Below is a confusion matrix visualization that shows the distribution of classified images.



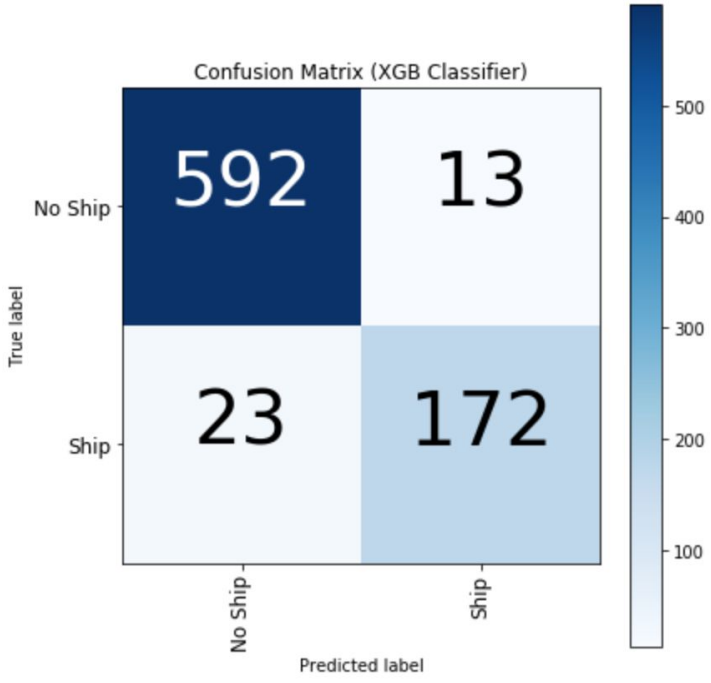
I also made a Receiver Operating Characteristic curve or ROC curve to see the true positive rate versus the false positive rate.

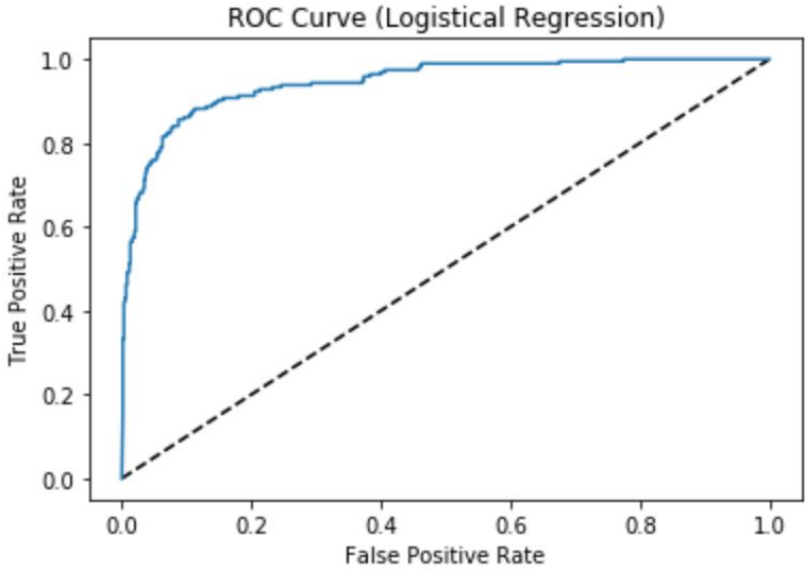


Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test

After completing the Logistic Regression model I turned to finding other models to improve upon. I found it in XGBoost's classifier model, a popular model among the Kaggle machine learning community. This model was implemented in the very same way I implemented the logistic regression model. XGBoost stands for eXtreme Gradient Boosting. It works by implementing a decision tree algorithm. This approach is when new models are created that

predict the residuals of errors of previous models. Those residuals are then added together to make the final prediction. The term ‘gradient boosting’ comes from the fact that it uses the gradient descent algorithm to minimize the lose when new models are added. The XGB classifier didn’t work as fast as I thought it would but it did have a much higher mean accuracy score than the logistic regression model. XGBoost Classifier had a mean accuracy score of 95.5. This was much higher than anticipated and a result I was very happy with. The confusion matrix of this model was much better than the logistic regression model. The confusion matrix showed that 592 “no ship” images and 172 “ship” images were classified correctly, which is an improvement. There were only 13 false positives which was a major improvement over the logistic regression model and 23 false negatives. These numbers are represented below in the confusion matrix plot.



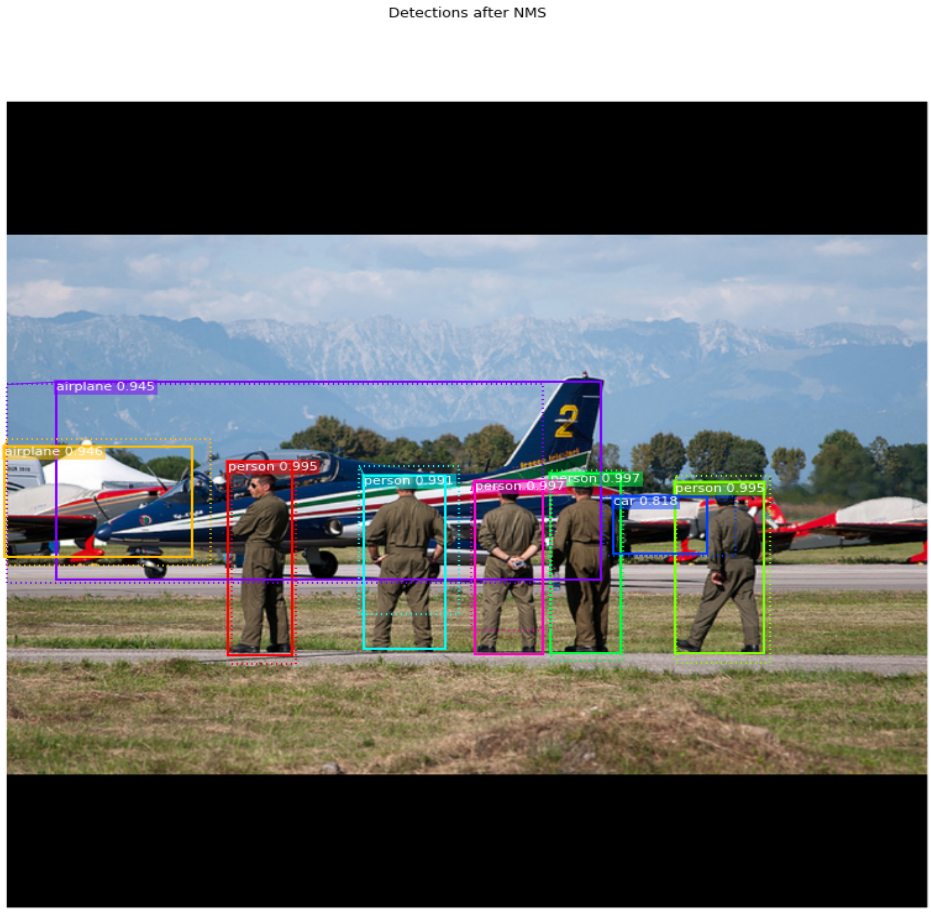


As you can see from the XGB model ROC curve, the curve is closer to the upper left corner (100% sensitivity, 100% specificity) than the logistic regression ROC curve. The XGB model is higher in overall accuracy of the test.

Overall, both models were very simple to use and easy to implement and I was very impressed with the results. My expectations of the logistic regression model were that it would perform at a mean accuracy of 80 percent or less and a more complicated model would be needed to get over 90 percent. I was very surprised to see that XGBoost model was able to complete the task with a 95.5 percent mean accuracy. Finally, I made a visualization of 100 random images with their label and prediction to get a good sense of how well the model worked and to see what images were false positives. This visualization is much too large for this paper. It can be found in the jupyter notebook uploaded to Github.

In the beginning of this project I had planned to implement a more complicated model by looking into image segmentation. Image segmentation is to Identify the object category of each

pixel for every known object within an image. In this case it would be used to segment the ships in the images to train the model to make it easier to locate ships in the test data set. With more time and knowledge I think implementing this model would increase the mean accuracy but only by a few percent over the 95.5 percent I have achieved. To implement this model, I could have looked at using a Mask-R-CNN model that would have put a box around the ship, highlighted it and gave us the accuracy of which the model predicted the object to be a ship. Below is an example of this method. You see different colored highlighted boxes around each individual segmented object with its label and its mean accuracy score.



Not only would this method have possibly improved our accuracy slightly but it can also be used to make a nice visualization. I am very happy with how the models I implemented turned out with their accuracy. I would like to find a way to improve the speed at which the model runs. PCA was one method I discovered that made the model run instantly however it caused a huge decrease in accuracy since it was reducing features. Im sure there are other models that are even more accurate than what I implemented. Possibly from Keras or elsewhere. A comparison of these models would surely be interesting but with the amount of time to work on this project I am happy with its results.

References

1. Grey, Eva. “Cargo Theft: a Billion-Dollar Problem.” *Ship Technology*, 8 Nov. 2017, www.ship-technology.com/features/featurecargo-theft-a-billion-dollar-problem-5882653/ .
2. Rhammell. “Ships in Satellite Imagery.” *Kaggle*, 29 July 2018, www.kaggle.com/rhammell/ships-in-satellite-imagery.
3. Wong, Edward, and Christoph Koettl. “How North Korea's Leader Gets His Luxury Cars.” *The New York Times*, The New York Times, 16 July 2019, www.nytimes.com/2019/07/16/world/asia/north-korea-luxury-goods-sanctions.html.