Ty Smith

11-30-18

Project #2

Dr. Xu

Introduction

For this project I needed to make a program that is able to find anomalies in from a network dataset. An algorithm applied to this problem is known as a Network Intrusion Detection System (IDS). As a requirement to avoid wasting an administrator's time we would have a low false positive rate to avoid having a person check on normal network traffic.

Preprocessing

The data-set has categorical features to distinguish qualities such as UDP and TCP protocols. Since the algorithm could not take in a string, I used the provided preprocessing to turn categorical into floats, and integers into floats, to pass it into the algorithm. I did not test any other preprocessing methods this time. One method I could try is PCA.

Algorithm

For the algorithm I used 3 different scikit learn prepackaged algorithms:

1. Isolation Forest
2. One Class SVM
3. Local Outlier Factor (LOF)

I found the Isolation Forest algorithm through Scikit Learn and tested with on the data. With the first graph I made,the results were very noisy and hard to make out what it thought

would be an outlier. Since this algorithm acts like a Random Forest choosing random features and making a tree, out of an ensemble I decided to keep it in since these typically perform well.

I used a one class SVM as my second algorithm, with out labels on the dataset. Based on the paper by Markus Goldstein and Seiichi Uchida one class SVM showed good results in semi-supervised learning. I started with the RBF kernel and found it performed subpar. The RBF kernel ended having 2 classes with many anomalies (G1). I then switched to a polynomial kernel and it performed much better under these conditions giving mostly one class with some anomalies (G2). With the other setting I kept them with the default other than the gamma i set to auto choose. With limited testing I decided to keep it in the final product.

Local Outlier Factor was the first algorithm I tested. I split data and graphed the results to the Euclidean Distance Mean and seemed to give me a similar result (G3). After some closer inspections it has less noise and what looked like fewer anomalous points. This uses a nearest neighbors style function so I set the neighbors to 10 and kept most things as default. The issue with using this approach alone is it assumes that the train data has no anomalies. Scikit learn's method has a built in way to deal with this called contamination, playing with this setting I did not see a large change in results so I left it at auto (G4).

Since each of the detectors having some sort of issue, I decided to use all three and average the score value they gave me. Based on the graph it has a large spread of scores with only a few outliers.

Processing Scores

For LOF and and Isolation Forest the socres returned in a negative value, while One Class SVM gave me good values between zero and one. To correct this I use the absolute value of the outputs for both LOF and Isolation Forest. Additionally LOF gave scores that had some extreme values such as 403. I decided to try an normalize the LOF scores using a Scikit

Learn normalization package. After seeing the new graphs I realized I might have been under weighting LOF and normalized the scores other algorithms. I was not happy with the Scikit learn representation for being too hard to read in the graphs so I switched to a general normalization of $\frac{x_i - min(x)}{max(x) - min(x)}$ this made the final graph more readable.

Testing

I tested each algorithm separately with half the data as training and half the data as a validation set. With each algorithm I plotted the score it gave and found the scores that were in between the high end and low end and tried to go and label them. Though I attempted this it did not help me much in refining the algorithm or giving me a score. I ended up trying to refine the algorithms based off the produced graphs and tried to have some clear separations (G5). This makes the testing unreliable.

Practical Application

Before rolling this algorithm out, I would like to have better labeled data by an expert or an administrator for the specific network it would be on. I think simple feedback such as labeling some data points would be the most beneficial to start when taking any administrators free since each network traffic could be different. With the labels I could make the SVM portion better semi-supervised learning techniques. I think it could work with current signature based filters, and could add signatures to the dataset and retrain the algorithm.

Appendix

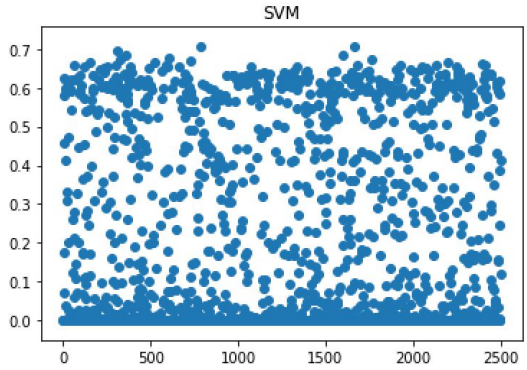
SVM - Support Vector Machine

RBF - Radial basis function

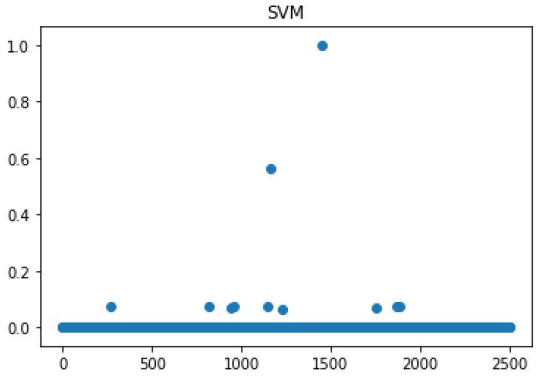
PCA - Principal component analysis

LOF - Local Outlier Factor

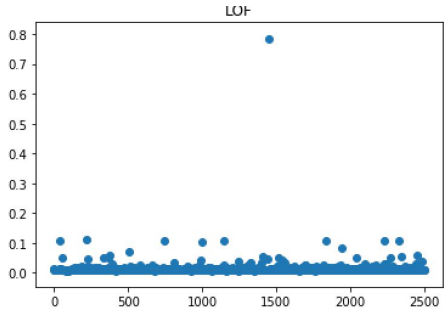
G1 - One Class SVM with RBF Kernel



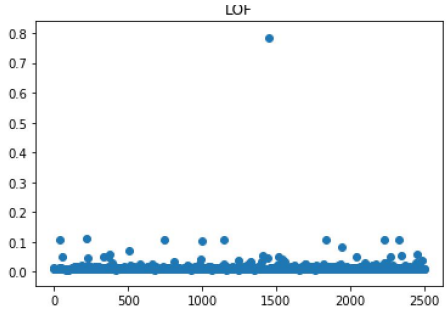
G2 - One Class SVM polynomial Kernel



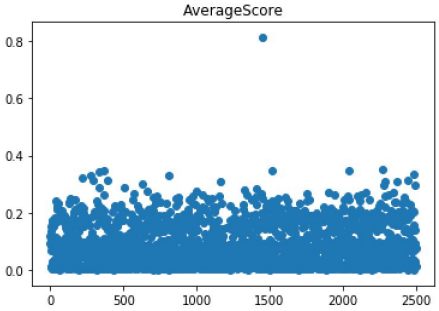
G3 - LOF Auto setting (post process)



G4 - LOF 0.2 contamination



G5 - Average Scores



Work Cited

Goldstein, Markus, and Seiichi Uchida. “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data.” *PLOS ONE*, Public Library of Science, 19 Apr. 2016, journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0152173#sec008.