Brett Luskin

Data 606 - Deliverable 4

# Malware Detection with Machine Learning

**i.** Motivation and Problem Definition

The purpose of this project is to use machine learning algorithms to identify malware attacks over network traffic based on the UNSW-NB15 dataset. This dataset is a combination of real network traffic and synthetic malware attacks that was created by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). There are 49 unique features to the dataset including the label. They consist largely of network data such as IP addresses, connection times, and number of connections.

The product of this research and successful execution would result in a Network Intrusion Detection System (NIDS). NIDS are an exciting area of research that combines the Data Science methods taught throughout the program at UMBC with cyber security. What was of particular interest to me was being able to demonstrate that these methods could be applied successfully to an area where I had no domain knowledge.

**ii.** Existing Approaches

This section could easily be titled “Emerging Approaches” as there seems to be a new paper or two published every month since I started researching the problem and dataset. When I started this project, most papers were about Machine Learning methods that had been applied to the dataset. At this point, May of 2020, there are many papers that have approached the problem with Neural Networks of different kinds.

One thing I want to note about most research papers is that they use a prepared dataset, the UNSW-NB15 training and test set CSV files. This is a subset of the original dataset where the training set contains 175,341 records and the testing set contains 82,332 records. The original dataset contains over 2.5 million records. The stratification of the data is also very different. In the prepared datasets, 68% of the data is attack data (versus 32% normal network traffic). In the original dataset attack data accounts for only 13% of the data.

**iii.** Proposed Solution and Exploratory Data Analysis

I did my initial Exploratory Data Analysis on the prepared datasets. The data quality was high, and minimal preprocessing of the data was needed. When I started working with the original large dataset, I had to redo my EDA a few times because I kept finding more issues with the data quality. This ended up being a good lesson in patience, as I was eager to start using the models I had built rather than focus on the data itself. The initial neural networks that I created

suffered in large part because of data quality, so I spent a significant amount of time trying to experiment and tune the parameters of the models rather than go back into the data and try to improve performance by cleaning it more. My initial EDA was focused on identifying important features in the data so that I could reduce the dimensionality for the machine learning algorithms. However, as I worked more with the larger dataset with the intent on focusing on the neural networks I was using, my EDA became much more about identifying quality issues.

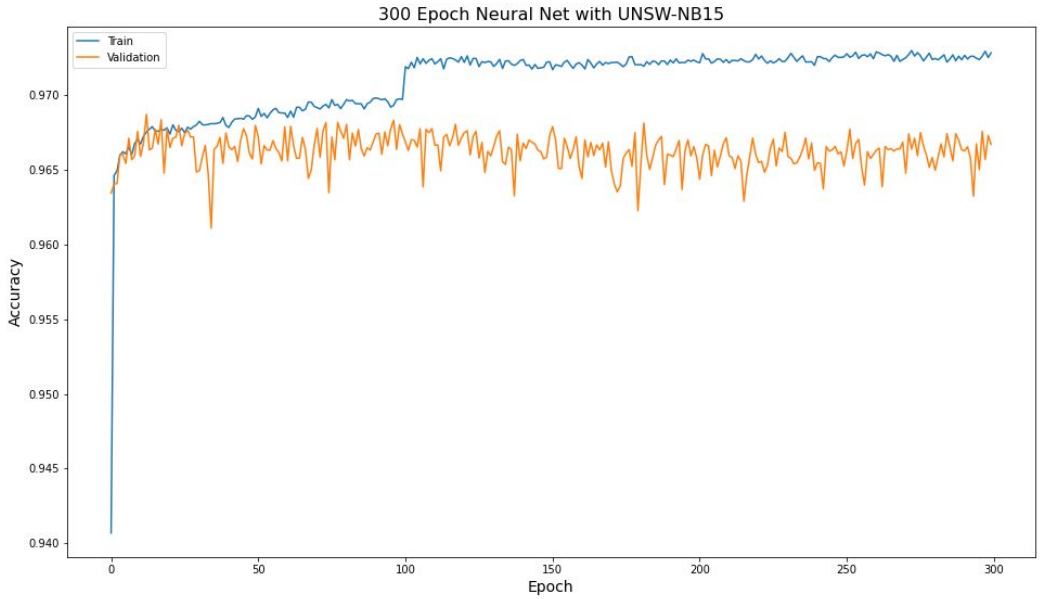
I tried five algorithms in the machine learning space of my project. Logistic Regression, SVM, PCA, Random Forests, and ADABoost. My intuition was that decision tree based models would work best because of the existence of a few information rich features in the data set, and also because the decision boundary could be unusual in a high dimensional space. I also created a few Feedforward Neural Networks which had various parameters. I believed that neural networks would vastly outperform the machine learning algorithms and would thrive by using the larger dataset. I think the neural networks are better equipped to handle the large amounts of features in this dataset.

**iv.** Implementation, Results, and Conclusion

Random Forests and ADABoost outperformed the other models by a wide margin. I benchmarked the performance against the research paper, "Important Complexity Reduction of Random Forest in Multi-Classification Problem," which achieved an accuracy score using a multiclass classifier of 75%. I achieved an accuracy score of 76%, right in line with the paper. By switching to a binary classifier, the accuracy score jumps to 85% because it is only trying to categorize by Normal or Malicious instead of into ten different categories. SVM and PCA both performed poorly, and I was surprised by SVM not doing better than a coin flip. I thought that using the kernel trick to create a non-linear decision boundary that it might be able to classify with a higher degree of accuracy. Logistic Regression was surprising in that I thought it would be terrible but ended up being between a coin flip and decision trees.

My first neural networks were created by trying to implement the methods in the paper, "Modeling Network Intrusion Detection System Using Feed-Forward Neural Network Using UNSW-NB15 Dataset". This paper claims to have achieved 99.5% accuracy using ten layers with ten neurons and only ten epochs. I was unable to reproduce those results and unfortunately, the paper did not include any code or pseudo code to help reproducibility. My first results yielded a 93.5% accuracy score. I spent quite a bit of time trying to find the best parameters for the model and spent quite a bit more time trying to get significantly longer training times (although Colaboratory can be frustrating in disconnecting long sessions). As I

stated before, I can not emphasize enough that the first intuition should be to look at the data and try to improve the data quality. By spending much less time going through and normalizing the data and cleaning some of the features better, I was able to get my accuracy score up to 96.5%, which is about where it stayed over the course of every iteration of my models. My final result was to achieve accuracy on the validation and test sets of 96.6%.




Future work on this could include trying to refine the data quality and using alternative model approaches such as semi-supervised or unsupervised learning.