

**KDD INTRUSION DATASET**

**Final Report**

Judah Drelich

**Introduction**

Cyber security is a major issue that is going to get worse. Global cyber crime costs are expected to grow 15% per year for the next 5 years reaching $10.5 Trillion dollars by 2025[[1]](#footnote-1). That would be close to a quarter of the entire world’s GDP[[2]](#footnote-2) and larger than the GDP of every country except the U.S.A., and China.2 From identity fraud to stealing financial assets to gathering important data cybersecurity is a crucial industry to protect innocent people from immense harm.

The dataset that I analyzed came from a 1998 DARPA Intrusion Detection Program managed by MIT’s Lincoln Labs.[[3]](#footnote-3) The data was created from TCP dump data from a simulated Air Force local area network (LAN). The data has 5 million connections, 41 features and a label for each observation that shows whether it was a safe connection or a certain type of intrusion.

Effective models in cyber security are valuable from the government to the private sector. By filtering out the connections that are less troublesome and giving certainty to certain observations that are clearly problematic I can give organizations a better chance of protecting themselves and potentially catching criminals.

**Data Wrangling**

The dataset did not come with column names but they were on the competition website. I assigned the list of the column names on the competition’s official website to the columns of the data frame. Next I separated the data into the feature variables (X). With X I can use pandas dummy creator method to create dummy features for all the unique values in the categorical columns. This will make it possible to analyze the entire dataset for multicollinearity instead of just the numerical columns.

To get started I used the “ProfileReport” function from the ydata-profiling package to gain an overall sense of the data. “ProfileReport” has a lot of great preliminary explorations that show the state that the dataset is in. When I ran the function on the original dataset some things stood out.

* No missing values
* Constant features
* Skewed features:
* Large Multicollinearity
* High numbers of zeros:

The first three items of the list are simple and easy to deal with. A dataset without missing values is great because I did not have to impute values or discard observations. Constant features are simple to deal with because I can drop them from the dataset since they provide no predictive ability or correlation with the target variable.

Skewed features impede models’ ability to accurately describe interactions between atypical features because the size of the highest values dwarf every part of the calculation. To fix this, I scaled the features with a standard scaler to preserve the patterns of each feature while scaling down the largest values.

**Multicollinearity**

The most glaring problem from “ProfileReport” is the number of variables that have high correlations with each other. This is because a sizable portion of the columns measure similar things. For instance, "srv\_rerror\_rate" and "dst\_host\_srv\_rerror\_rate” both measure the rate at which a server will deny a connection request. The former is recording only the last two seconds whereas the latter is using the last hundred connections. While both features have slightly different information, the information is so similar that it is impossible to leave both features in the dataset. The challenge now is to go through the dataset and try to find all instances of similar features and either combine them or discard them.

A screen shot of a computer screen

Description automatically generated with low confidenceBelow is the starting point of the multicollinearity for the entire dataset.

While it may look like most of the data does not have a multicollinearity problem, the dark part of the graph are all the dummy variables from the categorical columns. Because I dropped the first value from each of the categorical columns, it is mathematically impossible for those dummy columns to be collinear.

The heatmap shows a concerning square from features 15 to 39. While there are other problem areas, if I can clean up that square, many of the other collinearities will dissipate.

Cleaning up the collinearity is not easy. While it may seem tempting to simply drop all the problematic columns, there is too much information in those columns. Most of the features on the heatmap that have no correlation are dummy variables that do not have the same amount of information as the unchanged original features.

A screenshot of a computer screen

Description automatically generated with low confidenceThe first step was to zoom in on bright spot of the heatmap to see if there were any groupings of features that had clear real-world explanations for their high correlations. This produced the following graph:

In the heatmap above “serror” or “rerror” account for four variables each that create boxes of prominent levels of correlation. Given their clear real-world connection, I combined each grouping of the four variables into ‘Syn Error’ and ‘Rej Error’ by taking the maximum value of those four variables for each observation. That way if there is a spike in one of the four variables, then Syn and Rej Error will spike while if all four of the original variables are low, then the combined feature will be low as well.

Without an obvious next step from the heatmap above, I then created a dictionary with the features as keys and the number of correlations with other features over 0.7 as the values. The dictionary’s largest value was eight collinearities followed by six from 'Syn Error' which is the synthetic feature from the “serror” variables. By evaluating ‘Syn Error’ last, I can try to preserve it while eliminating multicollinearity.

A screenshot of a computer

Description automatically generated I then created a binary heatmap without ‘Syn Error’ where all the boxes of correlations that were greater than 0.9 were white and the rest of the graph is black:

From the graph above, I saw two groupings of features that need to combined or dropped.

* Top left: ['dst\_host\_srv\_count', 'same\_srv\_rate', 'dst\_host\_same\_srv\_rate']
* Bottom right corner: ['srv\_count','service\_ecr\_i', 'dst\_host\_same\_src\_port\_rate']

The top left group should be combined because the variables in it are measuring connections to the same server in different ways. Similar to the Syn and Rej error I took the max of 'same\_srv\_rate', 'dst\_host\_same\_srv\_rate'. Since there is no way to combine 'dst\_host\_srv\_count' and it is highly collinear I will drop it. For the bottom right there is no clear thread tying the three together. I decided to keep 'protocol\_type\_tcp' because the other two have a massive correlations with count. If I keep 'protocol\_type\_tcp' then I can keep 'count'.

I continue to iterate through the dataset with the same process where I will create a dictionary to see most collinear features. I then look to see if there is any connection between the variables that would justify combining them instead of dropping them. If there is not a clear similarity, then I drop columns in order of importance. The order is defined as:

1. Combinations of Original Columns:
   1. ‘Syn Error’, ‘Rej Error’, ‘srv\_rate'
2. Numerical Original Columns
   1. ‘count’, ‘src\_bytes’
3. Binary Original Columns
   1. ‘su\_attempted’, ‘is\_guest\_login’
4. Dummy Variables from the Categorical Variables
   1. 'protocol\_type\_tcp', ‘flag\_SF’

The more information a column contains, the higher the priority is for it to stay in the dataset. When I finally dropped all the collinear columns, the heatmap of collinearity looked like this.

A screen shot of a computer screen

Description automatically generated with low confidence

It is a significant improvement over the start of the start of the process that had a massive bright square in the top left quadrant.

To make sure that I had fully eliminated the multicollinearity I calculated every remaining feature’s Variance Inflation Factor (VIF). The formula for VIF is:

(1)

Where R2 is the R-squared value that represents correlations.[[4]](#footnote-4) The guidelines for an acceptable VIF score can range from 2.5-10 although anything over five is suspicious.[[5]](#footnote-5) With the remaining features that I had, 87 out of the 89 had a VIF of under 2.5 and 2 had a VIF < 3.03.

**Lasso Regularization**

Lasso Regularization is a technique for variable selection that uses regression to evaluate the effect that features have on a target variable. The idea is to add a penalty term that contains the coefficient or slope of the variables with respect to the target variable multiplied by parameter.

(2)

Where is the coefficient of the feature, *SSD* is the sum of squared distances of the point to the regression line and is the penalty’s parameter. If there is more than a single feature, then we can sum the coefficients:

(3)

Lasso aims to reduce the loss in equation (3). If a feature is not important, then changing its slope will not move the regression line close enough to the data points to decrease the loss function with a non-zero value. If a feature is important, the regression line will move towards the data and minimize the SSD faster than it increases the penalty term. Since only features with non-zero coefficients are meaningful, I can discard all the features with a coefficient of zero.

After running the Lasso Regularization with the target variable being a binary choice between attack and no attack, twelve out of the eighty-nine features have non-zero coefficients. By selecting out these features I have created the small dataset. The problem with the small dataset is that it is possible that it is missing some of the features that would distinguish between different attack types. To be safe, I iterated through all the different intrusion types using lasso regression to see if there were additional notable features that I had missed in smaller dataset. That process yielded sixty-one features and became the bigger data frame.

EDA

A screenshot of a computer screen

Description automatically generated with low confidence The simplest way to explore this data was to create a correlation heatmap between the twelve features and all the different intrusion types.

This graph indicates that there is some predictive power within the features but that I need to do a lot more analysis to find it. To do that I can create clusters in the data that will be able to show how multiple features can come together to create an attack.

One problem with this graph is that most of the correlations are hovering around zero. This is because the frequency of the intrusion types varies wildly. Below is a list of the most common intrusions and the percentage of observations in the dataset.

|  |  |
| --- | --- |
| Smurf. | 56.8378% |
| Neptune. | 21.6997% |
| Normal. | 19.6909% |

The most common intrusion types make up over 98% of the dataset which is why much of the heatmap has little to no correlation.

**Clustering**

The diverse types of intrusions clearly fall into separate groups. On the original challenge's [webpage](https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html) the contest makers list four distinct groups:

* DOS: denial-of-service, e.g., syn flood.
* R2L: unauthorized access from a remote machine, e.g., guessing password.
* U2R: unauthorized access to local superuser (root) privileges, e.g., various `buffer overflow’ attacks.
* probing: surveillance and other probing, e.g., port scanning.

This makes sense. For example, portsweep and ipsweep are clearly in the probing group as they are types of sweeps that search for vulnerabilities.

But we cannot assume that intrusions of the same kind should be grouped together. It could be that a hacker would use ipsweep as a first step in a ping of death (pod) attack, while rarely ever using an ipsweep and portsweep in the same attack together. Without clear information on how to group the different intrusions, we need to use cluster analysis to provide a more rigorous analysis.

I also produced my own grouping that has more specific groups than the competitions list of four. While the competition's grouping made sense, I thought that a method specific approach would be another potentially valid grouping.

[darpa.pdf (iisc.ac.in)](http://eprints.iisc.ac.in/26885/1/darpa.pdf)

1. [Cybercrime To Cost The World $10.5 Trillion Annually By 2025 (cybersecurityventures.com)](https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/) [↑](#footnote-ref-1)
2. [Gross Domestic Product (GDP) - Worldometer (worldometers.info)](https://www.worldometers.info/gdp/#:~:text=1%20U.S.A.%20%2419%2C485%2C394%2C000%2C000%202%20China%20%2412%2C237%2C700%2C479%2C375%203%20Japan,Germany%20%243%2C693%2C204%2C332%2C230%205%20India%20%242%2C650%2C725%2C335%2C364%206%20UK%20%242%2C637%2C866%2C340%2C434) [↑](#footnote-ref-2)
3. [KDD-CUP-99 Task Description (uci.edu)](https://kdd.ics.uci.edu/databases/kddcup99/task.html) [↑](#footnote-ref-3)
4. [How to Calculate VIF in Excel - Sheetaki](https://sheetaki.com/how-to-calculate-vif-in-excel/) [↑](#footnote-ref-4)
5. [Variance Inflation Factors (VIFs) - Statistics By Jim](https://statisticsbyjim.com/regression/variance-inflation-factors/) [↑](#footnote-ref-5)