The data we used was provided to us as a csv file named log2.csv. This file contained information on internet traffic that was passed through a firewall. The features that describes the data in the row where Source Port, Destination Port, NAT Source Port, NAT Destination Port, Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time (sec), pkts\_sent, pkts\_received and action. Action is the target we are building the model to predict. The data was read in using pythons built in csv library. From there the data was saved into a dictionary where the data and target column were stored as numpy arrays.

It was decided to One hot encode the Destination Port and the NAT Destination Port columns using Sklearn’s OneHotEncoder package (Source Port and NAT Source Port were removed and will be discussed why further on.). Before one hot encoding these 2 categorical columns we took the approach to compress the data size by including only the ports that made up 97% of the Destination and NAT destination ports and coding the remaining 3% as an other category. This resulted in a reduction of Destination Port values from 3,336 down to 587. For NAT destination ports the number of ports went from 2,540 down to 95. All together one hot encoding those two columns resulted in an additional 678 columns to our training set.

The continuous data (Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time (sec), pkts\_sent, pkts\_received) were all scaled using the Sklearn StandardScaler package. Scaling is important because in models like SVM (SVC in Sklearn) and SGD the model is finding the decision boundary between two points and the distance between those two points will be different between data that is scaled and data that is not scaled.

In the end we decided to remove (drop) the Source Port, and NAT source port features. Our reasoning behind this was because source port numbers are randomly generated when connections are established to act session ID’s (see <https://en.wikipedia.org/wiki/Ephemeral_port> for more information). If the source ports were to be included. and one hot encoded, the source port feature’s randomly generated numbers would add thousands of columns to our dataset that in turn would add noise and ultimately interfere with our results on top of adding an exponentially longer time to fit models.

The default data type of our continuous data was float64, and in order to reduce the amount of memory used during model fitting, we converted it to float32. This also saved us time. We used the float data type because scaling our data introduced decimal values.

Our final dataset contained 65,532 rows with 687 feature columns and the target (action) column.

—-------

First, we identified the continuous (Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time (sec), pkts\_sent, pkts\_received) and categorical variables (Source Port, Destination Port NAT, Source Port, NAT Destination Port) in our dataset.

Next, we scaled the continuous variables using sklearn’s StandardScaler(). We scaled our continuous data because SVM and SGD models are both sensitive to the scale of their input features. If the input features are not scaled, the models may assign too much or too little weight to certain features which could affect their performance.

The default data type of our continuous data was float64, and in order to reduce the amount of memory used during model fitting, we converted it to float32. This also saved us time. We used the float data type because scaling our data introduced decimal values.

Next, we one-hot encoded the categorical variables using pandas’ get\_dummies() function with drop\_first=True. We set drop\_first to True because typically when you one-hot encode a column, the first new one-hot encoded column will have a very high correlation with the other columns. This could introduce performance issues in our models, so we chose to remove that risk.

Our categorical variables had 10+ thousand unique variables which made it not plausible for us to  to one-hot encode all of them. So, we took the top 5% most occurring values in each categorical column, one-hot encoded those, and put the rest of the 95% of that column’s data into an “other” column. We did this for each of our four categorical “port” columns in our data. Since we separated out 5% of the data in each port column for one-hot encoding, this left holes in the port’s “other” columns. We dealt with this by filling in those missing values with zeros because that information was already captured in the one-hot encoded columns and therefore did not think we needed to make up bogus data or have duplicate data in our dataset.

 Our final preprocessed dataset contained 65532 rows and 2894 columns.

—---------