Case Study 5: Cybersecurity & Firewall Traffic

Matt Farrow

July 3, 2022

1 Introduction

In this study, the goal to take a [data set](https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data) of internet traffic records from Firat University in Turkey, and through the use of Support Vector Classifier (SVC) and Stochastic Gradient Descent (SGD) models, make decisions about whether to allow, deny, drop, and reset-both the traffic.

2 Methods

## 2.1 Data Examination

An initial examination of the data revealed 65,532 observations and 12 features, including ‘Action’, the multiclass response variable of interest (Table 1).

|  |
| --- |
| Feature |
| Source |
| Port |
| Destination Port |
| NAT Source Port |
| NAT Destination Port |
| Action |
| Bytes |
| Bytes Sent |
| Bytes Received |
| Packets |
| Elapsed Time (sec) |
| pkts\_sent |
| pkts\_received |

Table : Feature List

In taking a closer look at the response variable, ‘Action’, I noticed that the data is heavily imbalanced, with allow being the primary action, and reset-both being almost non-existent in the data. As a result, that value was dropped from the data set in order to make a small correction to the imbalanced data.

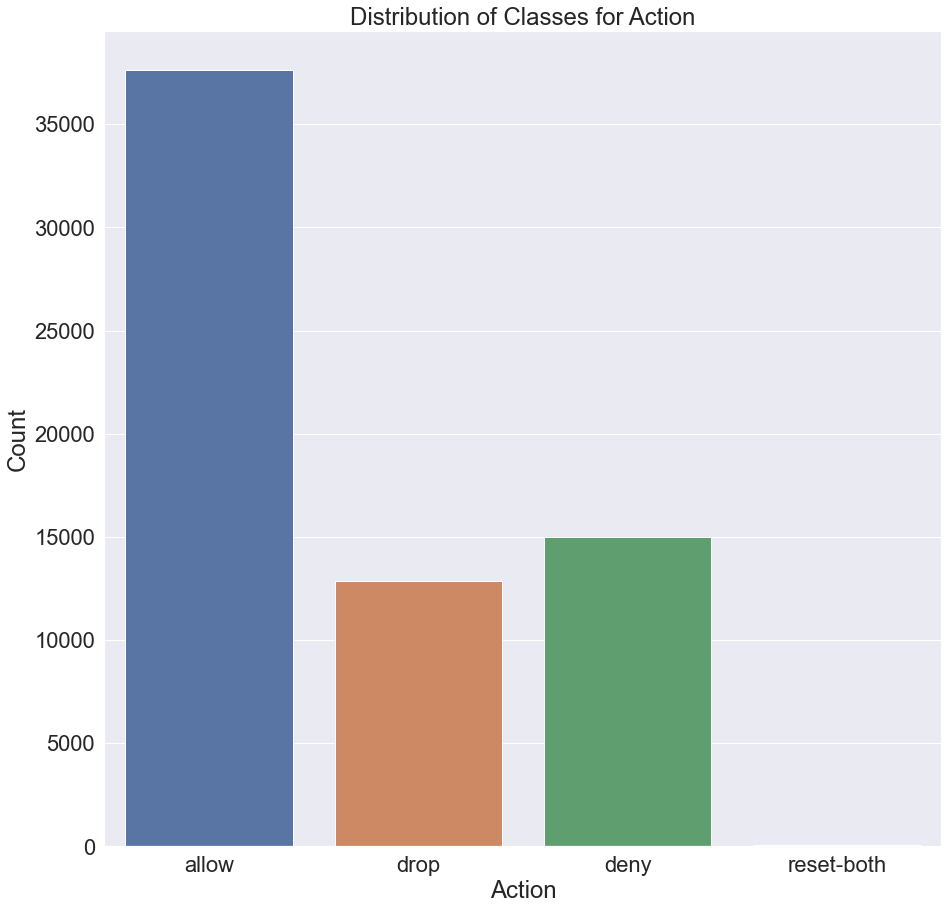


Figure : Distribution of Response Variable

The one piece of feature engineering that I did was to create an attribute to note, through either a 0 or 1, whether the source or destination port required network address translation (NAT). I then examined the pairwise correlations of the numeric features in the data. Unsurprisingly, bytes is closey related to packets and pkts\_sent, but I ended up deciding to leave those features in the data.

Chart, treemap chart

Description automatically generated

Figure : Pairwise Correlation of Numeric Variables

## 2.2 Model Preparation & Execution

The data was split into test and training data sets using an 80/20 split with a stratified shuffle and columns that needed to be one-hot encoded were identified (Table 2).

|  |
| --- |
| Feature |
| Source Port |
| Destination Port |
| NAT Source Port |
| NAT Destination Port |
| Source Need NAT  Destination Need NAT |

Table : Features to One-Hot Encode

A pipeline was built to define the model and handle the one-hot encoding and scaling of the numeric features before running everything through a randomized cross-validation with a hyperparameter search for the SVC (Table 3) and SGD (Table 4) models.

|  |  |
| --- | --- |
| Attribute | Values to Search |
| C | 0.001, 0.01, 0.1, 1, 10, 100 |
| Gamma | ‘scale’, ‘auto’, 1, 0.1, 0.01, 0.001, 0.0001 |
| Kernel | ‘linear’, ‘poly’, ‘rbf’ |

Table : SVC Hyperparameter Search

|  |  |
| --- | --- |
| Attribute | Values to Search |
| Alpha | 0.000001, 0.00001, 0.0001, 0.001, 0.01 |
| Eta0 (learning rate ‘optimal’) | 0 |
| Eta0 (learning rate ‘constant’) | 0.01, 0.1, 1.0, 5, 10 |
| Class Weight | ‘balanced |

Table : SGD Hyperparameter Search

Additionally, in order to try and visualize the work that the SVC model was doing, the response variable was collapsed to only two potential actions – allow and deny. That allowed me to plot the results and see the impact of the hyperparameter choices, rather than simply visualize them through confusion matrices. In that model, the hyperparameter list was also simplified (Table 5).

|  |  |
| --- | --- |
| Attribute | Values to Search |
| Kernel | ‘linear’, ‘rbf’ |
| C | 0.001, 0.01, 0.1, 1, 10, 100 |

Table : SVC Model (Allow/Deny) Hyperparameter Search

3 Results

## 3.1 Support Vector Classifier Model

The SVC model performed with an extremely high score across all measures (Table 6). Additionally, the confusion matrix (Figure 1) details the small number of erros made in the model with deny being the most frequent incorrect classification (as allow).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| allow | 0.999334 | 0.997078 | 0.998205 | 7528 |
| deny | 0.992027 | 0.995997 | 0.994008 | 2998 |
| drop | 0.997287 | 0.999222 | 0.998251 | 2570 |
|  |  |  |  |  |
| accuracy |  |  | 0.997251 | 13096 |
| macro avg | 0.996214 | 0.997432 | 0.996821 | 13096 |
| weighted avg | 0.997259 | 0.997251 | 0.997253 | 13096 |

Table : SVC Model Performance

Chart, treemap chart

Description automatically generated

Figure : SVC Confusion Matrix

The linear kernel shows an extremely high accuracy score, followed by rbf, and finally poly has a significant drop off in accuracy (Figure 4). Examining the accuracy performance of the ‘C’ parameter, I note that the performance of all values below 1 is 57.5% accuracy, but at C = 1, the accuracy jumps to 91.9% (Figure 5).

|  |  |
| --- | --- |
| Kernel | Accuracy Score |
| linear | 0.9928 |
| poly | 0.5748 |
| rbf | 0.9187 |

Figure : SVC Kernel Accuracy

|  |  |
| --- | --- |
| C | Accuracy Score |
| 0.001 | 0.5748 |
| 0.01 | 0.5748 |
| 0.1 | 0.5748 |
| 1 | 0.9187 |
| 10 | 0.9193 |
| 100 | 0.9497 |

Figure : SVC 'C' Accuracy

## 3.2 Stochastic Gradient Descent Model

For all intents and purposes, the model performance of the SGD model is almost identical to that of the SVC model (Table 7).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| allow | 0.999334 | 0.996148 | 0.997738 | 7528 |
| deny | 0.990381 | 0.995997 | 0.993181 | 2998 |
| drop | 0.997284 | 1.000000 | 0.998640 | 2570 |
|  |  |  |  |  |
| accuracy |  |  | 0.996869 | 13096 |
| macro avg | 0.995666 | 0.997382 | 0.996520 | 13096 |
| weighted avg | 0.996882 | 0.996869 | 0.996872 | 13096 |

Table : SGD Model Performance

Chart, treemap chart

Description automatically generated

Figure : SGD Confusion Matrix

## 3.3 Two-Class SVC Model

For the two-class SVC model, the training and testing accuracy were both 99.8% and the confusion matrix shows the small number of misclassifications that the model made (Figure 7).

Chart, treemap chart

Description automatically generated

Figure : Two-Class SVC Model Confusion Matrix

In terms of kernel performance, the ‘linear’ kernel returned a 99.9% accuracy whereas the ‘rbf’ kernel was only able to return 77.9% accuracy. This is able to be visualized in the plots which show the distribution of points and the fitted hyperplane (Figure 8).

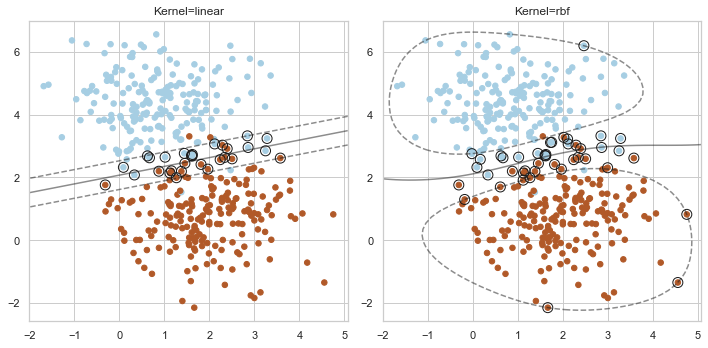


Figure : Kernel Hyperplane Performance and Margin

Visualizing the performance of the linear (Figure 9) and rbf (Figure 10) kernels, I was able to see how the different levels of ‘C’ affected the accuracy results in the original models. When , it is possible to see the points that the model did not correctly capture through the margin and hyperplane.

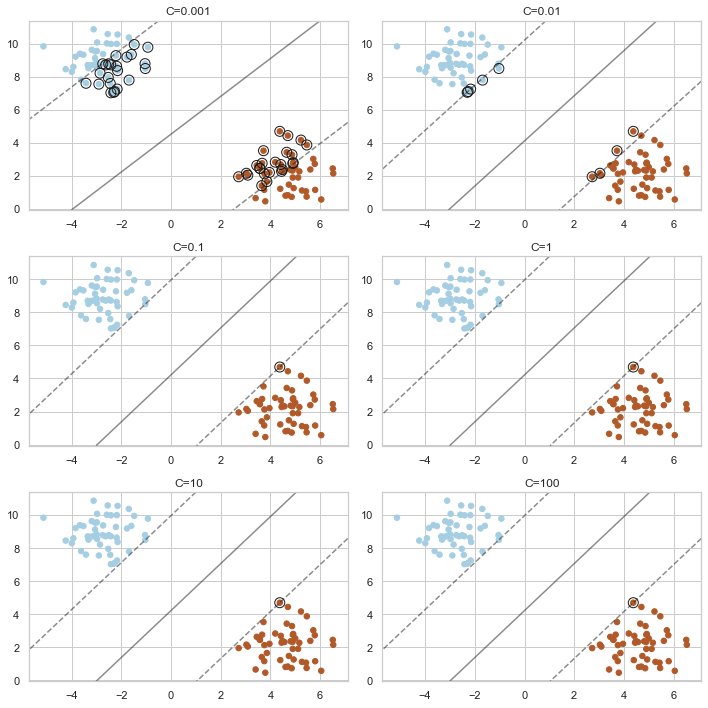


Figure : Linear Kernel Performance at Different Levels of 'C'

Chart, scatter chart

Description automatically generated

Figure : RBF Kernel Performance at Different Levels of 'C'

4 Conclusion

In conclusion, all three models were able to produce incredibly accurate accuracy scores which does give me pause to think that more work should have been done to take multicollinearity into account. In terms of model preference, I’m partial to the two-class SVC model due to the ability to visualize the hyperplane, margin, and model performance.

# Appendix

## Code

Code begins on the following page.