2022

Case study 5

INTERNET FIREWALL

BALAJI AVVARU, APURV MITTAL, RAVI SIVARAMAN

Abstract

This study is for the Classification of firewall log files using support vector machine (SVM) and stochastic gradient descent (SDG) modeling techniques. Based on the various parameters of the incoming traffic, the system needs to decide if it should be allowed or denied. In this study, logs obtained with the Firewall Device used at Firat University are classified using SVM and SDG classifier.

Introduction

In this study, firewall log data is used with the defined action if the traffic was allowed to go through or not. In this data set 65,532 instances are using 11 features. The Action attribute is the target variable with class values of “allow”, “deny”, “drop”, and “reset-both”.

SVM and SDG models are used in this study with various tuning parameters to get the maximum recall and precision values. Receiver Operating Characteristic (ROC) curves are also created for each model.

Here is the list of features available in the dataset:

#	Feature	Description
1	Source Port	Client Source Port
2	Destination Port	Client Destination Port
3	NAT Source Port	Network Address Translation Source Port
4	NAT Destination Port	Network Address Translation Destination Port
5	Bytes	Total Bytes
6	Bytes Sent	Bytes Sent
7	Bytes Received	Bytes Received
8	Packets	Total Packets
9	Elapsed Time	Elapsed Time for flow in seconds
10	pkts_sent	Packets Sent
11	pkts_received	Packets Received
12	Action	Class (allow, deny, drop, reset-both)

Table 1: Feature Description

Reference: https://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data

Data Analysis

There were 65,532 values against 11 features. There are no missing values. This dataset doesn't require imputation for the features.

Correlation Analysis

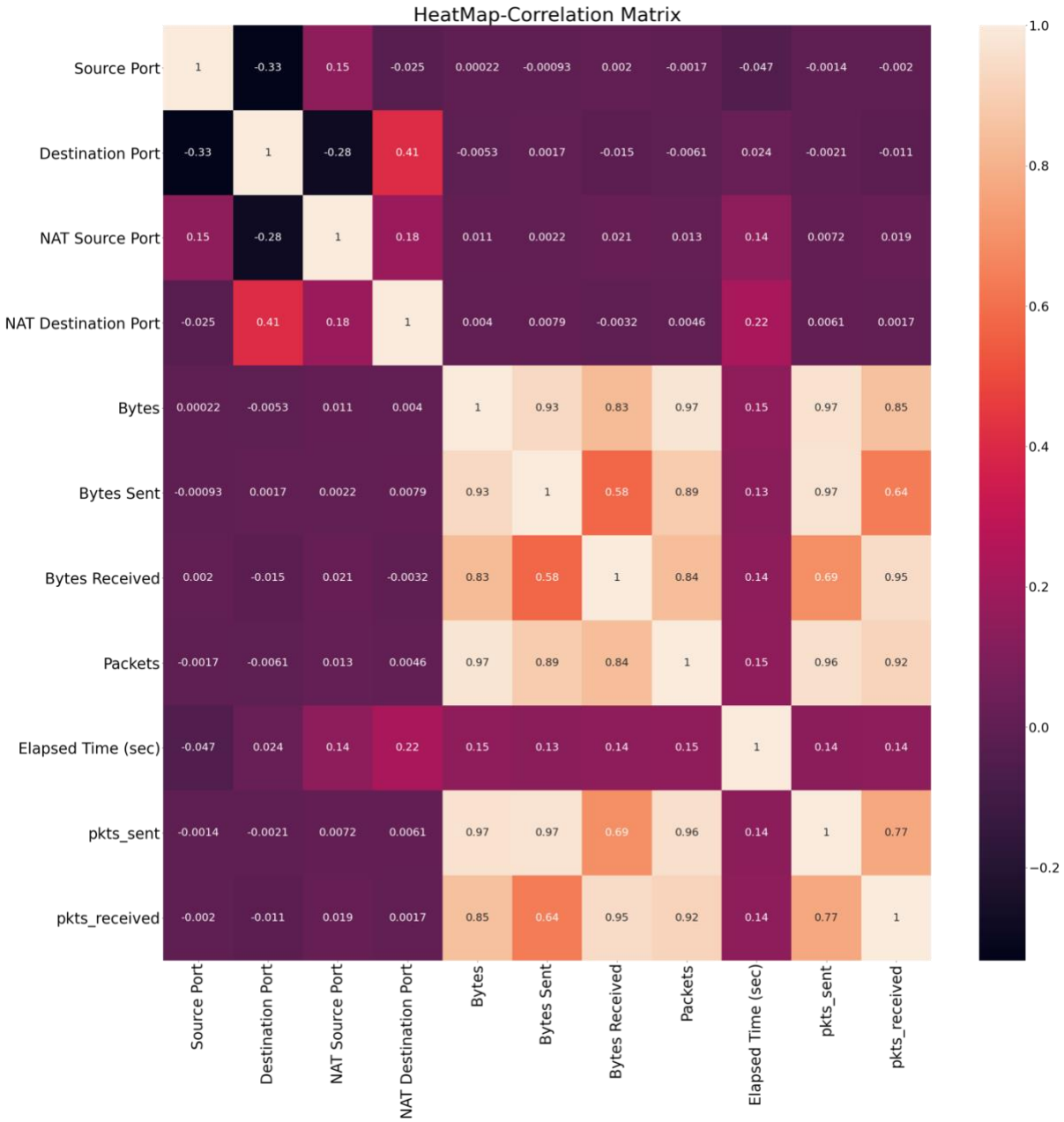


Figure 1: Collinearity heatmap of Attributes

Data shows there is high collinearity for a few of the attributes, and this is expected, as many of the features are closely coupled like: *Bytes and Bytes Sent* or *Packets and Packets sent* etc.

This study considered all the features with lower than 99% correlation for this analysis. There are features with up to 97% of correlation. Since none of the features showed 99% or above collinearity, all features are considered for modeling, and none are dropped.

All the features are normalized using the `Standard Scaler` technique.

Target

Target is a multi-class variable that defines the result of the incoming request. The target variable "Action" has four classes: Allow, Deny, Drop, and Reset-both.

Action	Description
Allow	Allows the internet traffic.
Deny	Blocks traffic and enforces the default Deny Action defined for the application that is being denied
Drop	Silently drops the traffic; for an application, it overrides the default deny action. A TCP reset is not sent to the host/application.
Reset-Both	Sends a TCP reset to both the client-side and server-side devices.

Table 2: Action (target) Class Description

Reference: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8355382*

Distribution of "Target "class variables:

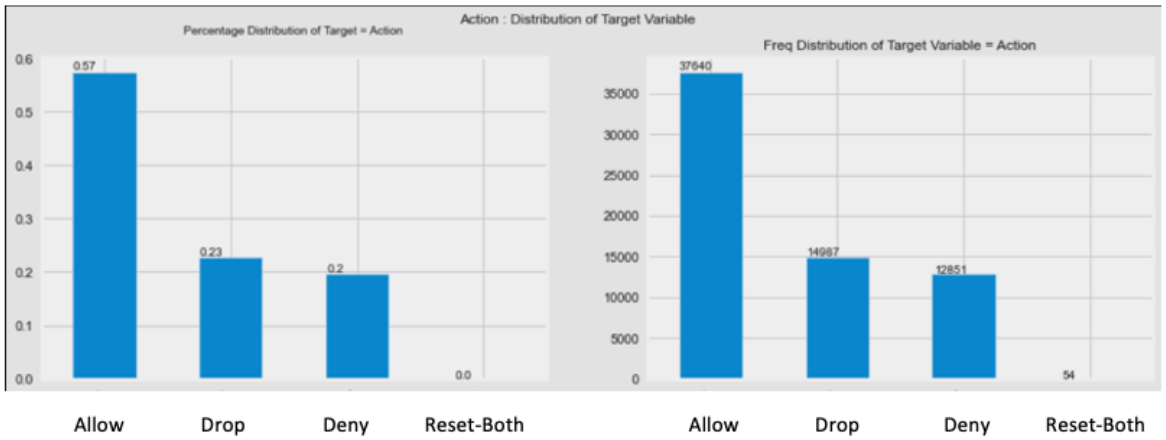


Figure 2: Target - Class Distribution

Class values of “drop”, “deny” and “reset-both” are combined as “drop”. After this change the target variable appears well balanced.

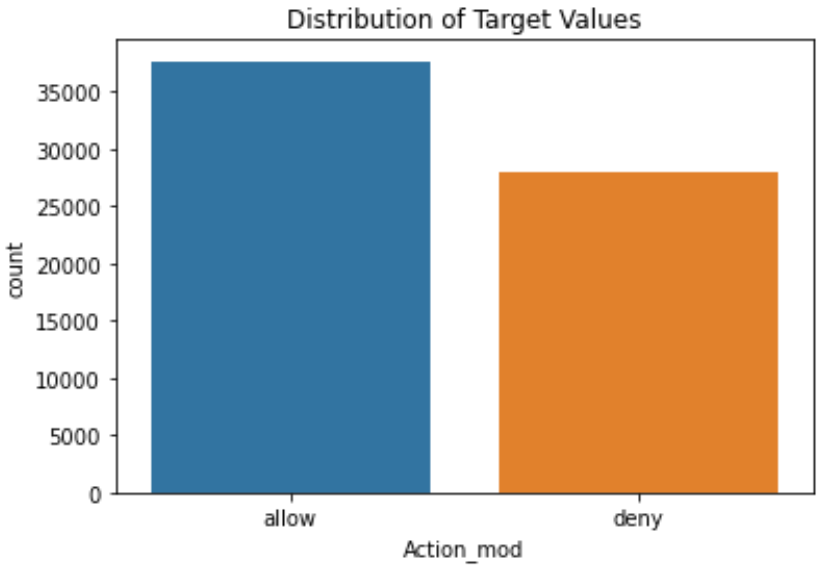


Figure 3: Action (target) distribution after imputation

For modeling purposes, the target variable is converted to Boolean with `allow = 1` and `deny = 0`.

2. Methods

Classification Models

This study uses support vector machine (SVM) and stochastic gradient descent (SDG) modeling techniques. Also, compares the model with various hyper-tuning parameters to determine the best set of parameters for the model with the highest accuracy, precision, recall, and f-1 score.

Two model evaluation techniques were used:

1. Modeling Technique 1: Use the features dataset as provided which all features are continuous variables including *Source Port, Destination Port, NAT Source Port, and NAT Destination Port.*
2. Modeling Technique 2: Convert the following features as categorical: Source Port, Destination Port, NAT Source Port, and NAT Destination Port.

a. Converting features to categorical generates the need for one-hot encoding leading to more than 57,000 additional columns. This makes the model computation process intensive.
b. To avoid long computational times vowpal wabbit optimization technique is used.

Support Vector Machine

`Support Vector Machine (SVM)` is a supervised learning classification model which uses non-probabilistic binary linear classification method. SVM maps training examples to points in space to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

Reference: https://en.wikipedia.org/wiki/Support-vector_machine

Parameters:

- `C:` Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

- *`Kernel:`* Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' is defaulted. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples)

- *`class_weight`*: If not given, all classes are defaulted to have weight as one. The *`balanced`* mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as

```
a. n_samples / (n_classes * np.bincount(y))
```

Stochastic Gradient Descent

Stochastic gradient descent (SGD) can be regarded as a stochastic approximation of gradient descent optimization since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems, this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate.

Reference: https://en.wikipedia.org/wiki/Stochastic_gradient_descent

The hyper-parameters (tunable parameters) are:

Parameters:

loss: The possible options are 'hinge', 'log', 'modified_huber', 'squared_hinge', 'perceptron', or a regression loss: 'squared_loss', 'huber', 'epsilon_insensitive', or 'squared_epsilon_insensitive'

penalty: The penalty (aka regularization term), Defaults to '*l2*' which is the standard regularizer for linear SVM models. '*l1*' and '*elasticnet*' might bring sparsity to the model

alpha: Constant that multiplies the regularization term, The higher the value, the stronger the regularization. Also used to compute the learning rate when set to *learning_rate* is set to '*optimal*'

max_iter: The maximum number of passes over the training data (aka *epochs*).

class_weight: Weights associated with classes. If not given, all classes are default to have weight of one. The *balanced* mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as

```
b. n_samples / (n_classes * np.bincount(y))
```

GridSearch

GridSearch method is used to find the most optimum model of the dataset.

SVM:

For *SVM,* below hyper tuning parameters were used:

```
C = [0.1, 1, 5, 10]
class_weight = ['balanced', None]
kernel = ['linear', 'poly', 'rbf']
```

SGD:

For *SGD,* below hyper tuning parameters were used:

```
loss = ['modified_huber', 'hinge']
penalty = ['l2']
alpha = [0.00001,0.00005, 0.0001, 0.0005, 0.001, 0.03, 0.01, 0.1, 1]
class_weight = ['balanced', None]
max_iter = [100, 200, 300]
```

3. Results

Modeling Technique 1:

Use the features dataset as provided which all features are continuous variables including *Source Port, Destination Port, NAT Source Port, and NAT Destination Port.*

The two models *SVM* and *SGD* are computed on the given data. Both models produce high precision results. *SVM* and *SGD* both produces the highest accuracy of 99.8%.

Best Support Vector Machine Model:

Below parameters produced the best *SVM* model:

`C` = [10]
`class_weight` = [None]
`kernel` = ['rbf']

Best Stochastic Gradient Descent Model:

Below parameters produced the best *SGD* model:

`loss` = ['hinge']
`penalty` = ['l2']
`alpha` = [0.0001]
`class_weight` = [None]
`max_iter` = [100]

Both the models with the parameters listed above were executed with Cross Validation of 10.

The detailed results from both the models are listed below:

Metric	**SVM**	**SGD**
Best Accuracy	0.998	0.998
F-Score	0.998	0.998
Precision	0.999	0.999
Recall	0.997	0.997
Runtime (grid search)	28 min 17 sec	25.7 sec

Table 3: Method 1 - Classification Matrix comparison of SVM and SGD

The ROC curve for the SVM and SGD models showed a mean area under the curve AUC of 1.00. The ROC curve also indicates there are no false positives compared to the true positive rate. The area under the curve indicates the results cover the entire dataset.

The Precision-Recall Curve for the SVM and SGD models are shown below (second plots) indicates both SGD and SVM are performing precisely to allow/ deny network traffic for this dataset.

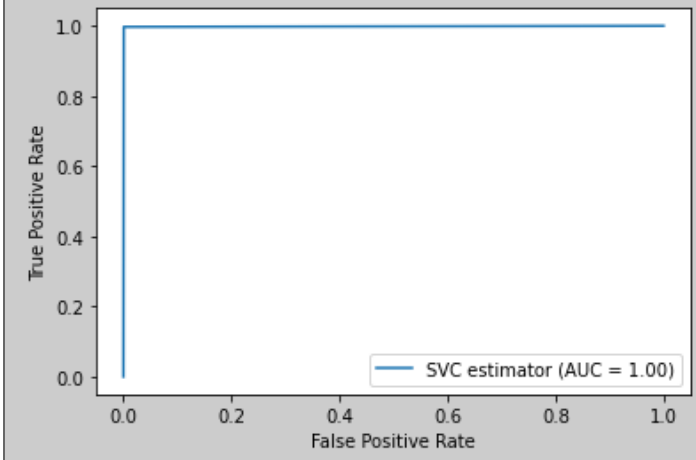
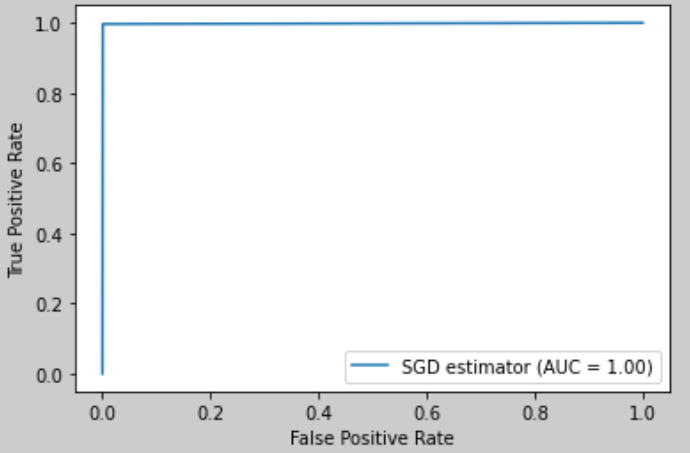
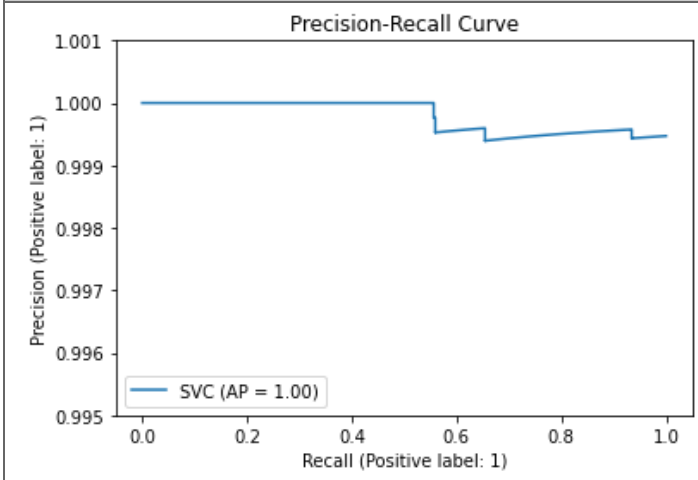
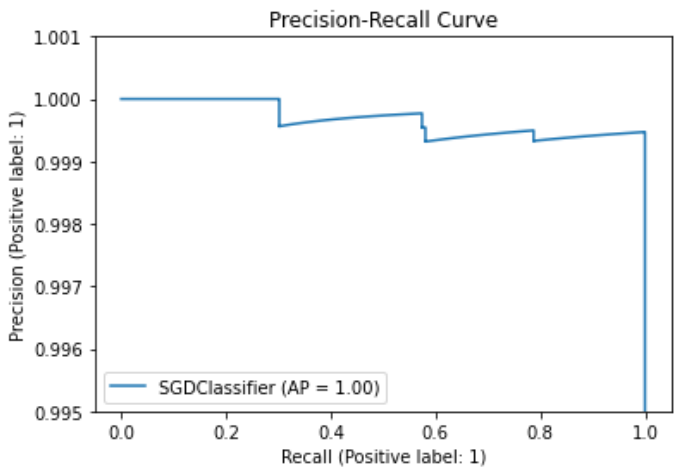
SVM	**SGD**
	
	

Table 4: Comparison of SVM and SGD

Modeling Technique 2:

Convert the following features as categorical: Source Port, Destination Port, NAT Source Port, and NAT Destination Port.

Vowpal Wabbit

The performance of Python is slower, due to being a dynamic typed general-purpose language and doesn't allow direct handling of memory. On the other hand, `Vowpal Wabbit (VW)` is a powerful command-line program that handles memory directly to perform calculations faster compared to Python.

`VW` can intrinsically handle missing data. On the other hand, it assumes that that missing data is `0`. Targets are not `0` and `1` (like coded for SVM and SGD model in method 1, discussed above). In VW the target is coded as `-1` and `1`.

In the current dataset, "`allow`" is converted to 1, and all other classes are converted to -1.

In this dataset, the "`ports`" – even though they are numbers – are considered as categorical features, as the port number is just a name, not a numeral, so the port numbers must be considered as categorical.

The categorical features are added as strings by appending the feature name and the value. For example, in a row of `Source Port` value is `443`, this data is converted to `SourcePort_443`, which becomes a categorical feature.

For other numerical features, it is retained as a number by appending with a feature name and "`:`". For example, the `Bytes` feature has `117`, then it is converted to `Bytes:117`. Any value that has "`:`" is considered numeric, and _ is a string, so it becomes categorical.

In `VW`, the data is present if it is 1, but none are present if data is missing, reducing the size of `VW` matrix to be very small. The missing data is constructed in memory as `0` in successive registers; thus, `VW` runtime can access them extremely faster, improving performance by several orders.

SVM

Below parameters produced the best *SVM* model:

`Passes` = [100]
`Loss function` = [hinge]
`kernel` = ['rbf']
`Regularization` = [L2]

```
--quiet --cache --passes 100 --kernel rbf --loss_function hinge --l2 0.0001
```

SGD

Below parameters produced the best *SGD* model:

`Passes` = [100]
`Loss function` = [hinge]
`Regularization` = [L2]

```
--quiet --cache --passes 100 --sgd --loss_function hinge --l2 0.001
```

The detailed results from both the models with their runtimes are listed below:

Metric	SVM	SGD
Accuracy	0.998	0.997
F-Score	0.998	0.997
Precision	0.998	0.997
Recall	0.998	0.997
Runtime	30.7 sec	26.6 sec

Table 5: Method 2 - Classification Matrix comparison of SVM and SGD

4. Conclusion

Both modeling techniques SVM and SGD produces models of very high accuracy, precision, and recall. There is no significant difference between the output from the models for the given dataset. However, the processing time varies significantly. SGD processes much faster (25.7 sec) compared to SVM (28 min 17 sec). With shorter processing time also, SGD produces comparable results to that of the SVM.

By using vowpal wabbit processing technique the SVM processing time is also greatly reduced and obtained in about 30 seconds. This method brings great efficiencies in running the model and the accuracy of results is not compromised.

Based on the above models and results its can be concluded that for this dataset both SVM and SGD produces almost equivalent high accuracy results. Vowpal wabbit technique can be used for improving the processing efficiencies in real-time as network traffic decisions need to be decided in milliseconds.

Appendix – Code

NB Viewer Link:

http://nbviewer.org/github/apurvmittal-MSDS/QuantifyingTheWorld-DS/blob/main/Case%20Study%20-%205%20Vowpal%20Wabbit.ipynb