Case Study 5 | Firewall Classification

Joey Hernandez | DAnny Chang

DS 7333 | Quantifying the World

2023

**Introduction**

**Background**

In today's technologically advanced and interconnected world, the management of data traffic and ensuring the security of a network infrastructure is paramount. Large corporations, especially those heavily reliant on online platforms, witness a significant amount of firewall interactions daily. These firewalls act as gatekeepers, determining which requests should be granted access and which should be denied, to protect the integrity and confidentiality of a company's data and resources.

The company for our project is one large-scale entity that faces these challenges. Historically, their approach to managing these interactions has been manual, necessitating considerable manpower and resources. This traditional method not only leads to increased operational costs but also leaves room for human error, potentially compromising the security of the entire system. As the volume of requests and data interactions continues to rise, it is crucial to rethink and reimagine the process to ensure scalability and efficiency.

## Objective and Scope

The primary objective of this project is to develop an intelligent, autonomous system that can accurately and efficiently classify incoming firewall requests and decide whether to allow, deny, drop, or reset both connections. This system is intended to replace the current manual method, which is resource-intensive and prone to human error, with a more advanced, automated, and reliable solution. To achieve this, the project will employ machine learning algorithms, particularly Support Vector Machines (SVC) and Stochastic Gradient Descent (SGD) Classifier, to learn from the historical data and make informed decisions in real-time.

## Scope

**Data Utilization:**

* Utilize the available dataset of approximately 60,000 rows, which contains historical decisions on firewall requests.
* Address the challenge of class imbalance, particularly with the "reset-both" class, which has only about 50 occurrences, thereby posing a risk of model overfitting towards other classes.

**Model Development and Tuning:**

* Implement Support Vector Machine (SVC) models, renowned for their effectiveness in high-dimensional spaces and adaptability in creating complex decision boundaries.
  + Focus on tuning the 'C' parameter in SVC, which controls the trade-off between achieving a low training error and a low testing error that is crucial for the model's effectiveness in generalizing unseen data.
* Employ the Stochastic Gradient Descent (SGD) Classifier, known for its efficiency in large-scale data handling and versatility in optimizing different loss functions and penalties.
  + Fine-tune the 'alpha' parameter, which is a constant that multiplies the regularization term and inversely scales with the learning rate, influencing model convergence and accuracy.

**Classification Strategy:**

* Develop a model capable of classifying requests into one of the following categories: "allow," "deny," "drop," and "reset-both."
* Implement strategies to handle class imbalance, such as resampling the dataset or applying different weights during the training process, ensuring that the "reset-both" category is treated accurately.

**Performance Evaluation:**

* Upon development, the model will undergo rigorous testing to evaluate its accuracy, speed, and reliability in classifying and processing requests.
* Performance metrics will not only focus on overall accuracy but also on the precision, recall, and F1-score for each class, especially for the underrepresented "reset-both" category.

## Data Inspection

Upon preliminary inspection of the dataset provided for our use case, it's evident that the data quality is fair. Some key observations include:

1. **Duplicates:** The dataset does not contain any duplicate entries, ensuring that each data point is unique and will contribute to the robustness of the model.
2. **Missing Values:** There are no missing values present within the dataset, suggesting that the data collection process has been thorough and systematic.
3. **Integrity:** The data appears to be intact, with no evident signs of corruption or anomalies that might jeopardize the model-building process.
4. **Class Imbalance:** One noteworthy aspect of the dataset is the class imbalance observed in the target variable, specifically in the "reset-both" category, which only registers about 50 occurrences. While the data is of high quality, this imbalance poses a challenge for model training, as it may lead to overfitting or underrepresentation of this specific class.

**Feature Analysis and Treatment:**

While the dataset encompasses various features, attention is required for a specific subset of features: Source port, destination port, NAT source port, and NAT Destination port. At a passing glance, these features appear numerical. However, based on domain knowledge and the context in which these ports operate, it's more apt to treat them as categorical variables.

To efficiently manage these features without inflating the feature space, a binning strategy has been employed. The binning criteria adopted are as follows:

* **Bins:** [0, 1023, 49151, 65535]

Corresponding to these bins, the data points are categorized into:

* **Labels:** ['well-known', 'registered', 'dynamic/private']

This binning strategy allows for meaningful categorization while keeping the dataset compact and interpretable. For instance, ports ranging from 0 to 1023 are typically "well-known" ports designated for standard services, whereas the ones from 1023 to 49151 are "registered" ports. Lastly, ports from 49151 to 65535 fall under the "dynamic/private" category.

## Target Variable Inspection

As mentioned above, in our analysis spanning the entire dataset, we meticulously examined the distribution of our target variable, differentiating between "allow", "deny", "drop", and "reset-both" categories. The table highlights a pronounced class imbalance, with "reset-both" instances forming a minuscule fraction of the total data. To break it down: "allow" commands a dominant 62.73%, "deny" stands at 24.98%, "drop" at 21.42%, while "reset-both" barely touches 0.09% of the dataset. Such a stark disparity, especially in the "reset-both" category, emphasizes the need to craft our modeling approaches with precision to adeptly address this imbalance.

***Figure 1:*** *Count plot illustrating the distribution of Classification targets.*

*A graph of a number of bars

Description automatically generated with medium confidence*

***Description****: The count plot displays the class distribution for our target variable, highlighting the disparity between each of the four instances. This visualization aids in identifying class imbalances present in the dataset.*

***Table 1:*** *Table detailing the distribution of target classes.*

|  |  |  |
| --- | --- | --- |
| Distribution of Target Variable | | |
| Class | **Class Counts** | **Class Percentages** |
| Allow | 37640 | 62.73% |
| Drop | 14987 | 24.98% |
| Deny | 12851 | 21.42% |
| Reset-Both | 54 | 0.09% |

***Description****:* The table presents the distribution of occurrences of the data target variable. It succinctly highlights the number of instances for each class and the corresponding percentages.

## Modeling

## Support Vector Machines (SVC)

The Support Vector Machine (SVM) is a powerful and versatile supervised machine learning model capable of performing classification tasks. SVM works by finding the hyperplane that best divides the dataset into classes. In high dimensional spaces, this hyperplane can be thought of as a "line" that optimally separates data into respective categories.

For our analysis, we specifically employed the Linear Support Vector Classification (Linear SVC). This SVM variant is suitable for large datasets and performs linear classification, which means it seeks to classify data by finding the best linear separator.

**Data Preparation and Splitting**

Prior to model training, we partitioned our dataset using a train-test split, ensuring a training set that constitutes 80% of the entire dataset. Given the known class imbalances in our target variable, it was paramount to ensure that both training and test sets have proportional representations of each class. To achieve this, we utilized the stratify argument, which guarantees that the distribution of target classes is consistent across both training and test datasets.

**Hyperparameter Tuning**

A key hyperparameter for the SVM is the regularization parameter, commonly denoted as 'C'. The role of 'C' is to determine the trade-off between achieving a low classification error on the training dataset and maintaining a small margin. A smaller value of 'C' will look for a larger margin separating hyperplane, even if it misclassifies more points, whereas a larger 'C' prioritizes classifying all training examples correctly, potentially at the expense of a smaller margin.

For the optimal selection of the 'C' parameter, we employed a Randomized Search CV process. Instead of exhaustively exploring the entire parameter space, a randomized search samples a fixed number of parameter settings from the specified distributions. This approach offers a more efficient way to navigate the hyperparameter space, especially when dealing with a large dataset. For our model, we conducted 60 iterations to hone in on the most effective 'C' value.

**Specific Configurations**

Our implementation of the Linear SVC came with two additional noteworthy configurations:

1. **Dual = False**: By setting the dual parameter to False, we solved the primal optimization problem, which is more efficient for larger datasets since the number of samples often exceeds the number of features. This was selected based on the recommendations from the sklearn website: *If n\_samples < n\_features and optimizer supports chosen loss, multi\_class and penalty, then dual will be set to True, otherwise it will be set to False*.
2. **Class Weight = 'Balanced'**: Due to the presence of class imbalances in our target variable, it was essential to weigh the classes such that the model does not exhibit bias toward the majority class. The 'balanced' option automatically adjusts weights inversely proportional to class frequencies, ensuring that each class contributes equally to the model's learning process.

In conclusion, our SVM-based approach, combined with meticulous data handling and hyperparameter tuning, aimed to derive meaningful patterns from the data while also addressing challenges such as class imbalance. The choice of Linear SVC, in particular, catered to our need for efficiency and accuracy in handling a dataset of substantial size.

**Results:**

**Confusion Matrix Analysis:**

The confusion matrix is an essential tool for understanding the classification performance, as it presents a clear picture of true positives, false positives, true negatives, and false negatives for each class:

* **Allow**: Out of 7528 instances, the model correctly classified 7511 as 'allow' and misclassified 17 as 'deny'. This indicates a near-perfect prediction for this class.
* **Deny**: Of the 2998 'deny' instances, the model correctly predicted 2942. However, there were 48 instances wrongly classified as 'drop' and 6 as 'allow', along with 2 misclassifications as 'reset-both'.
* **Drop**: The model showcased excellent performance for the 'drop' category, correctly classifying all 2570 instances.
* **Reset-Both**: This was the most challenging class for our model, given its sparse representation in the dataset. Out of 11 instances, the model only correctly classified one, with 10 instances being erroneously labeled as 'deny'.

Classification Report:

The classification report provides a comprehensive overview of key metrics, including precision, recall, and F1-score for each class:

* **Allow**: The model achieved a perfect precision, recall, and F1-score of 1.00, demonstrating impeccable performance for this category.
* **Deny**: With a precision of 0.99, recall of 0.98, and an F1-score of 0.99, the model displayed robust performance, correctly identifying the vast majority of 'deny' instances.
* **Drop**: The model attained a precision of 0.98, matched with a perfect recall and an F1-score of 0.99, indicating a near-perfect classification.
* **Reset-Both**: Here, the model faced challenges, achieving a precision of 0.33, a recall of 0.09, and a subsequent F1-score of 0.14. This underscores the difficulty posed by the class imbalance inherent to the 'reset-both' category.

Furthermore, the model achieved an overall accuracy of 0.99, indicating its excellent ability to generalize. The weighted average precision, recall, and F1-score were also 0.99, reiterating the model's strong overall performance. However, the macro averages, which give equal weight to each class, highlight the challenges with the 'reset-both' category, with values of 0.83 for precision, 0.77 for recall, and 0.78 for the F1-score.

***Figure 4:*** *A visual of the Linear SVC prediction matrix.*

A screenshot of a graph

Description automatically generated

***Description****: The plot above illustrates the confusion matrix for the predictions made by the Linear SVC model. On the x-axis, we have the predicted labels, and on the y-axis, we have the true labels.*

***Figure 5 :*** *Classification Report for the Linear SVC Classification Model*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Report** | | | | |
|  | **Precision** | **Recall** | **F1- Score** | **Support** |
| **Allow** | 1.00 | 1.00 | 1.00 | 7528 |
| **Deny** | 0.99 | 0.98 | 0.99 | 2998 |
| **Drop** | 0.98 | 1.00 | 0.99 | 2570 |
| **Reset-Both** | 0.33 | 0.09 | 0.14 | 11 |
| **Accuracy** |  |  | 0.99 | 13107 |
| **Macro Avg** | 0.83 | 0.77 | 0.78 | 13107 |
| **Weighted Avg** | 0.99 | 0.99 | 0.99 | 13107 |

***Description****:* *The classification report highlights the model's strong performance across most categories, with precision, recall, and F1-score exceeding 0.95 for "Allow", "Deny", and "Drop".*

## Stochastic Gradient Descent Classifier (SGD)

The Stochastic Gradient Descent (SGD) classifier is a potent and adaptable supervised machine learning algorithm capable of executing classification tasks. At its core, SGD updates parameters iteratively in response to each training example, making it particularly efficient for large datasets. The goal of SGD is to minimize the loss function by adjusting weights based on the gradient of the loss with respect to each weight.

Data Preparation and Splitting

Prior to beginning the modeling process, we split our dataset via a train-test partitioning scheme, using 80% for training and the remaining 20% for validation. Given the inherent class imbalances in our dataset, maintaining a consistent representation of classes in both subsets was vital. To ensure this, the stratify argument was leveraged, preserving the distribution of the target classes across both the training and test datasets.

Hyperparameter Tuning

A crucial hyperparameter for the SGD Classifier is the learning rate, often denoted as alpha. This parameter regulates the step size at each iteration while moving toward a minimum of the loss function. The correct balance is imperative; too large an alpha might overshoot the minimum, while too small could lead to slow convergence or getting stuck in local minimums.

To optimize the selection of the alpha parameter, we deployed a Randomized Search CV strategy. Instead of a comprehensive exploration of all possible values, this approach samples from a predefined distribution of possible alpha values, providing a balance between accuracy and efficiency. Our tuning procedure involved 5-fold cross-validation, ensuring robustness by testing the model on various sub-samples of the training data.

Specific Configurations

Our instantiation of the SGD Classifier was characterized by a few distinct configurations:

1. **Loss = 'Log'**: Opting for the logarithmic loss function (or log loss) transforms our SGD Classifier into a logistic regression, allowing the model to estimate probabilities, thus furnishing more nuanced decisions.
2. **Penalty = 'L2'**: This selection applies L2 regularization to the model. Regularization penalizes complex models, providing a countermeasure against overfitting. The L2 penalty specifically targets the squared magnitudes of the coefficient, promoting feature weights to be small and smooth.
3. **Class Weight = 'Balanced'**: Given the uneven distribution of classes in our target variable, it was of paramount importance to address this disparity. The 'balanced' setting adjusts the weights automatically, in an inversely proportional manner relative to class frequencies. This ensures that the model does not inherently favor the majority class, promoting a more unbiased learning experience.

**Results:**

**Confusion Matrix Analysis:** The confusion matrix provides critical insights into classification performance by detailing true positives, false positives, true negatives, and false negatives for each class:

* **Allow**: Out of 7,528 instances, the model accurately classified 7,414 as 'allow'. It misclassified 53 as 'deny', 58 as 'drop', and 3 as 'reset-both', showcasing an impressive prediction accuracy for this class.
* **Deny**: From the 2,998 'deny' instances, the model rightly predicted 2,923. There were 67 instances mistakenly classified as 'drop', 8 as 'allow', and none as 'reset-both'.
* **Drop**: For the 'drop' category, the model exhibited outstanding performance by correctly classifying all 2,570 instances without any misclassifications.
* **Reset-Both**: The 'reset-both' category remained challenging for the model. Out of 11 instances, none were accurately classified, with 1 instance mislabeled as 'allow' and 10 as 'deny'.

**Classification Report:** The classification report provides a comprehensive analysis of key metrics, including precision, recall, and F1-score for each class:

* **Allow**: The model achieved a precision of 1.00, a recall of 0.98, and an F1-score of 0.99, indicating stellar performance for this category.
* **Deny**: With a precision of 0.98, recall of 0.97, and an F1-score of 0.98, the model displayed strong precision in identifying 'deny' instances.
* **Drop**: The model posted a precision of 0.95, matched with a flawless recall of 1.00 and an F1-score of 0.98, underscoring its proficiency in this classification.
* **Reset-Both**: This class posed difficulties for the model, resulting in a precision, recall, and F1-score of 0.00, highlighting the significant challenges linked with this sparsely represented category.

Furthermore, the model achieved a commendable overall accuracy of 0.98. The weighted average metrics for precision, recall, and F1-score were also at 0.98, emphasizing the model's overall efficacy. However, the macro averages, which treat each class with equal weight, revealed the model's challenges with the 'reset-both' class, yielding values of 0.73 for precision, 0.74 for recall, and 0.74 for the F1-score.

***Figure 7:*** *A visual of the SGD Classification prediction matrix.*

A screenshot of a graph

Description automatically generated

***Description****: The plot above illustrates the confusion matrix for the predictions made by the Linear SVC model. On the x-axis, we have the predicted labels, and on the y-axis, we have the true labels.*

***Figure 6:*** *Classification Report for the SGD Classification Model*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Report - SGD** | | | | |
|  | **Precision** | **Recall** | **F1- Score** | **Support** |
| **Allow** | 1.00 | 0.98 | 0.99 | 7528 |
| **Deny** | 0.98 | 0.97 | 0.98 | 2998 |
| **Drop** | 0.95 | 1.00 | 0.98 | 2570 |
| **Reset-Both** | 0.00 | 0.00 | 0.00 | 11 |
| **Accuracy** |  |  | 0.98 | 13107 |
| **Macro Avg** | 0.73 | 0.74 | 0.74 | 13107 |
| **Weighted Avg** | 0.98 | 0.98 | 0.98 | 13107 |

***Description****:* *The classification report highlights the model's strong performance across most categories, with precision, recall, and F1-score exceeding 0.95 for "Allow", "Deny", and "Drop".*

**Conclusion**

In conclusion, as the digital landscape continues to expand, efficient and secure data traffic management is becoming ever more essential. Our project's intent was to transform a manual system into an intelligent, autonomous solution to meet the growing demands of today's interconnected world.

Employing both the Support Vector Machines (SVC) and the Stochastic Gradient Descent (SGD) Classifier, our endeavor yielded promising results. The SGD Classifier, for instance, showcased a remarkable accuracy rate, correctly classifying 7,414 out of 7,528 'allow' instances and all 2,570 'drop' instances. However, it struggled with the 'reset-both' category, misclassifying all 11 instances. On the other hand, the SVM-based approach displayed near-perfect predictions for the 'allow' class, with 7,511 correct out of 7,528. While it handled the 'deny' and 'drop' categories commendably, it faced challenges with the 'reset-both' category, accurately predicting just one out of 11 instances.

These results underscore both the potential and the areas needing refinement. While our machine learning models exhibited significant advancements in automating firewall request classifications, the 'reset-both' category highlighted the necessity for ongoing model improvements.

Looking ahead, it is crucial for the company to engage in regular model evaluations and updates, ensuring that the system remains responsive to evolving data patterns. By successfully integrating machine learning into their network management strategy, the company is not only poised to realize operational efficiencies but also enhance the overall robustness and security of its digital infrastructure.

**Recommendations**

1. **Frequent Model Retraining:** As firewall interaction patterns and types of requests may evolve, it is advisable to retrain the models periodically. This ensures that the models remain updated with recent patterns and can maintain high accuracy levels.
2. **Ensemble Techniques:** Given that both SGD and SVM have their strengths and weaknesses, the company might consider deploying ensemble techniques. Combining the predictions of both models could harness the strengths of each and potentially yield a more robust overall performance.
3. **Hyperparameter Retuning:** As new data becomes available, consider running regular hyperparameter tuning sessions. Given that the optimal parameters may change with different datasets, this ensures that the models are always tuned to their best possible configurations.
4. **Invest in Anomaly Detection:** Beyond classification, incorporating an anomaly detection system can provide an added layer of security. This would flag any unusual firewall request patterns, allowing for timely intervention.
5. **Scalability Concerns:** As the company grows and the volume of firewall interactions increases, it's vital to ensure that the infrastructure supporting these machine learning models can handle the increased load. This might involve scaling up computational resources or adopting cloud-based solutions.
6. **Address Class Imbalance:** Continuous monitoring of class distributions in incoming data is crucial. If significant shifts are noticed, the company might need to readdress the class weighting or even consider techniques like resampling to maintain model performance.
7. **Continuous Evaluation Metrics:** Beyond accuracy, other metrics like precision, recall, and the F1 score for each class should be continuously monitored. These metrics provide a more holistic view of model performance, especially when dealing with imbalanced datasets.

**Appendix**