An Experiment on Feature Selection using Logistic Regression

Raisa Islam[1], Subhasish Mazumdar[2], and Rakibul Islam[3]
Dept of Computer Science & Engineering
New Mexico Institute of Mining and Technology
Socorro, NM 87801 USA
[1]raisa.islam@student.nmt.edu, [2]subhasish.mazumdar@nmt.edu, [3]mdrakibul.islam@student.nmt.edu

Abstract

- Investigated a feature selection method using L1 and L2 regularization in logistic regression.
- Used the CIC-IDS2018 dataset to test the method, chosen for its size and challenging class separability.
- Ranked features using L1 and L2, then synthesized a feature set common to both rankings.
- Compared logistic regression models with L1 (LR+L1) and L2 (LR+L2) regularization.
- Found no significant accuracy difference between L1 and L2 once features were selected.
- Applied the synthesized feature set to complex models like Decision Trees and Random Forests.
- Despite the reduced feature set, accuracy remained high across models.
- Reported performance metrics: accuracy, precision, recall, and F1-score.

Experiment 1: Excluding one of the problematic classes.

- This experiment simplifies the classification task by removing a challenging class.
- The goal is to assess the model's performance on a less complex dataset.

Experiment 2: Including both problematic classes.

- This experiment introduces the full complexity of the dataset.
- The goal is to evaluate the model's ability to handle challenging classification scenarios.

Introduction

Method Overview: *Combines L1 and L2 regularization by ranking features separately* using *logistic regression* and *selecting features common to both sets.*
Dataset: A large, complex real-world dataset, with a particularly challenging class that is hard to distinguish from another.

Model Choice: Decision Tree for its explainability and Random Forest for higher accuracy, though less interpretable.
Results: The proposed method *reduced feature size by 72%, with only a 0.8% and 0.6% accuracy loss in Decision Trees and Random Forests*, respectively, even with the difficult class included.
Paper Structure: Includes background information (Section II), experiment details (Section III), results and analysis (Section IV), and conclusions with future research (Section V).

III. EXPERIMENT

A. Dataset and Pre-processing

1. **Dataset**:
    - **CIC-IDS2018**: A comprehensive dataset *representing network traffic observations over ten days.*
    - **Volume**: Contains *16,233,002 samples*, *distributed* across *10 CSV files.*
    - **Features**: Each sample includes *79 features* and one of *15 target classes.*
2. **Target Classes**:
    - The dataset comprises 15 distinct classes.
    - Sample sizes for classes vary significantly; for instance, the **DDOS Attack LOIC UDP** class has only **1,730 samples**.
    - Other classes with smaller sample sizes include **Brute Force XSS**, **Brute Force Web**, and **SQL Injection**.

The **CIC-IDS2018 dataset** consists of **15 classes**.

The **DDOS Attack LOIC UDP** class has **1,730 samples**, which is relatively small.

The classes for **Brute Force XSS**, **Brute Force Web**, and **SQL Injection** have significantly fewer samples: **230**, **611**, and **87** respectively.

Class imbalance is prevalent, as some classes have drastically fewer samples than others.

Most machine learning (ML) algorithms assume an **even distribution** of data across classes, making this imbalance a critical issue.

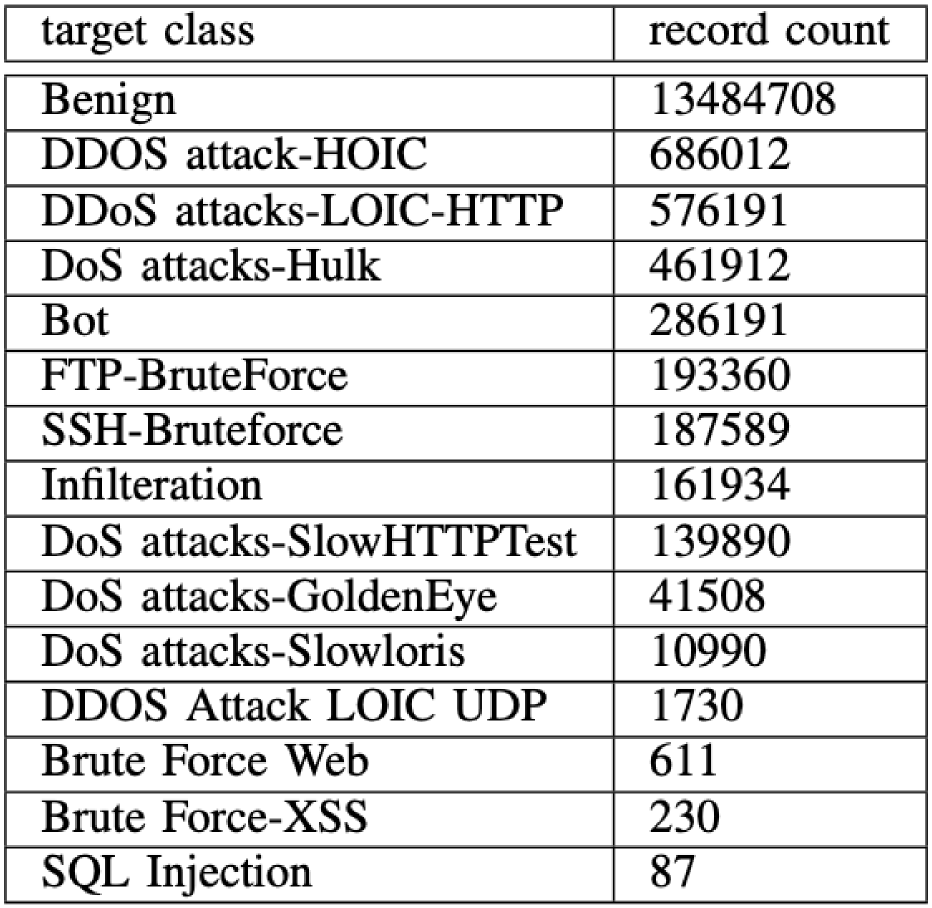
The dominance of majority classes can lead to **bias** in ML classifiers, often resulting in misclassification of minority classes.

Pre-processing

- In the **cleaning phase**, the following actions were taken:
    - Removed observations with feature values of **Infinity** and **NaN** (missing values).
    - Dropped **59 entries** classified as **unknown** (label was "Label").

    - Excluded the **timestamp feature** as it was deemed irrelevant, resulting in **78 usable features**.
- The goal was to extract **5,000 random samples** from each class that remained after cleaning.
    - To address the class imbalance, the classes with very small sample sizes (**Brute Force XSS**, **Brute Force Web**, and **SQL Injection**) were excluded, leaving **12 classes**.
    - Out of these, **11 classes** had **5,000 samples each**, while the **DDOS Attack LOIC UDP** class had **1,730 samples**.
- Ultimately, the dataset consisted of a total of **56,730 samples** after pre-processing.

Problematic Class Identification: The **DoS attacks-SlowHTTPTest** and **FTP-BruteForce** classes were identified as difficult to separate, with **DoS attacks-SlowHTTPTest** being specifically noted as problematic.



target class	record count
Benign	13484708
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS Attack LOIC UDP	1730
Brute Force Web	611
Brute Force-XSS	230
SQL Injection	87

TABLE I: target classes

B. Experimental setup

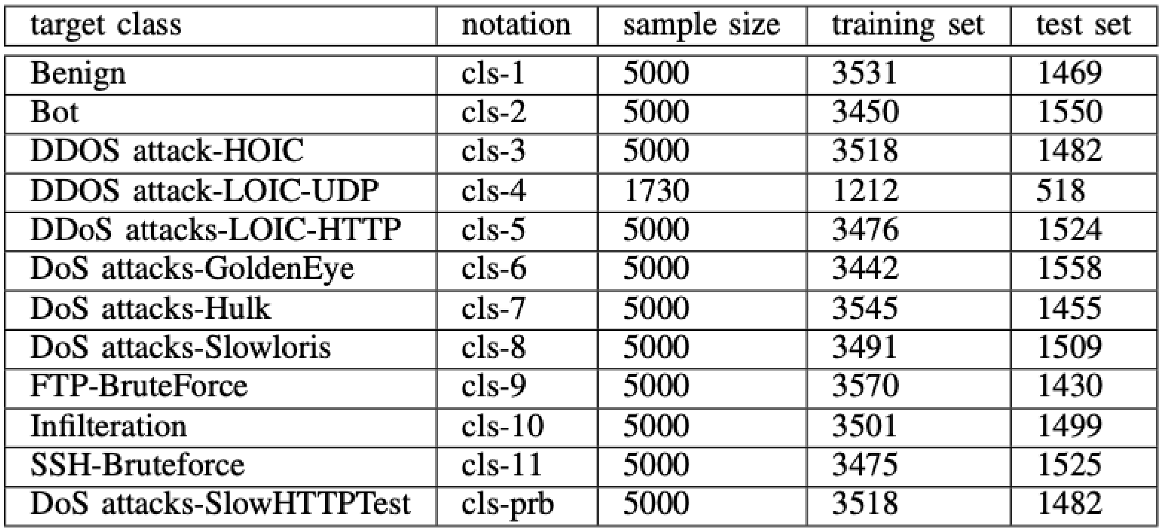
The experiments were conducted using Python 3, employing the `scikit-learn` package for training and testing machine learning models, and `matplotlib` for graph plotting. A 70:30 split was used for the training and testing datasets.

Initially, there were 36,211 training observations and 15,519 testing observations. In the second phase, 5,000 random samples of the problematic class, DoS attacks-SlowHTTPTest, were added, maintaining the 70:30 split. This resulted in 39,711 training samples and 17,019 testing samples.

Class sizes are outlined in Table II, with the first 11 rows showing data from the initial experiment and the last row reflecting the additional samples for the problematic class.

TABLE II

- The experiments were conducted using Python 3.
- **Libraries**: `scikit-learn` for model training and testing, `matplotlib` for graph plotting.
- **Data Split**: A 70:30 ratio was used for training and testing datasets.
- **Initial Dataset**:
  - *Training set: 36,211 observations.*
  - *Testing set: 15,519 observations.*
- **Second Phase**:
  - Added *5,000 random samples from the problematic class: DoS attacks-SlowHTTPTest.*
  - Updated training set: *39,711 observations.*
  - Updated testing set: *17,019 observations.*
- **Class Sizes**: Shown in Table II, with the *first 11 rows for the initial experiment* and the *last row for the added class samples.*



target class	notation	sample size	training set	test set
Benign	cls-1	5000	3531	1469
Bot	cls-2	5000	3450	1550
DDOS attack-HOIC	cls-3	5000	3518	1482
DDOS attack-LOIC-UDP	cls-4	1730	1212	518
DDoS attacks-LOIC-HTTP	cls-5	5000	3476	1524
DoS attacks-GoldenEye	cls-6	5000	3442	1558
DoS attacks-Hulk	cls-7	5000	3545	1455
DoS attacks-Slowloris	cls-8	5000	3491	1509
FTP-BruteForce	cls-9	5000	3570	1430
Infilteration	cls-10	5000	3501	1499
SSH-Bruteforce	cls-11	5000	3475	1525
DoS attacks-SlowHTTPTest	cls-prb	5000	3518	1482

TABLE II: Sample dataset description

TABLE III

Model: Logistic Regression using the SAGA solver (a variation of Stochastic Average Gradient).

Regularization Parameter: Inverse regularization set to 0.5.

Training: Model trained on the full feature set from the first part of the experiment.

Output:

- Generated coefficients for each feature.
- Calculated the mean accuracy of the model.

Feature Ranking:

- **L1 Regularization**: Applied Logistic Regression with L1 regularization (LR+L1) to rank features by their coefficients, generating an L1 rank.
- **L2 Regularization**: Similarly, used Logistic Regression with L2 regularization (LR+L2) and ranked features by coefficients for an L2-rank.

Sorting: Both feature sets were sorted in descending order based on coefficient values.

Table III: Lists common features with their ranks in L1 and L2 regularization, highlighting differences and similarities.

Imagine we are teaching a robot to make decisions, like telling if something is good or bad, using some clues or hints.

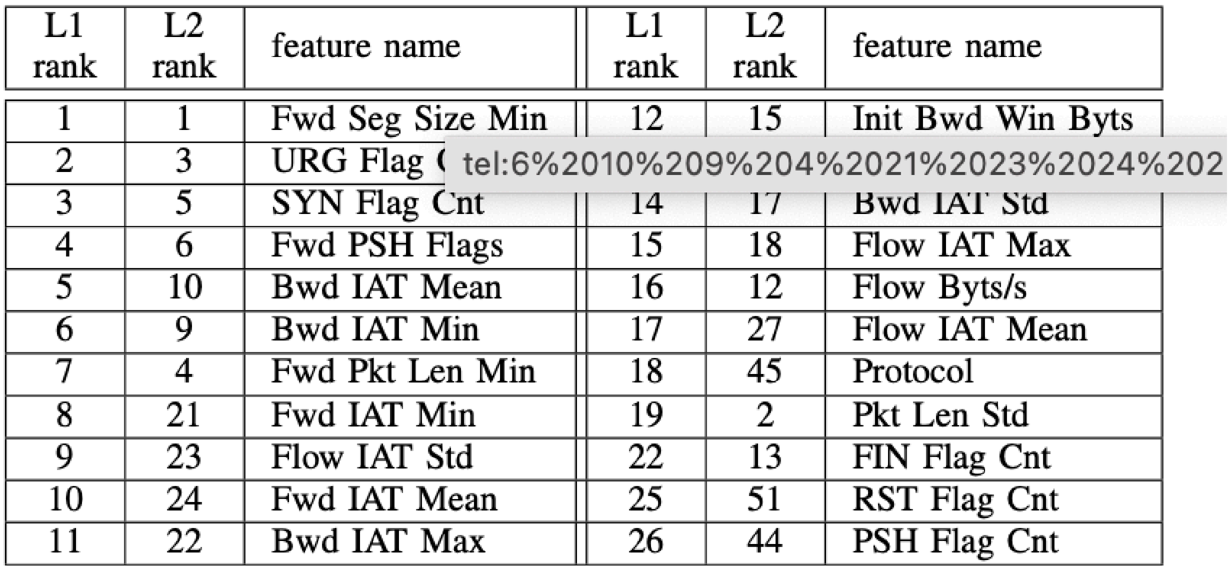
We used a special way called **Logistic Regression** to train our robot. It's like giving it lots of examples and asking it to learn what clues are most helpful.

To teach it well, we also made sure the robot didn't get too focused on little details. We did this by using a setting called **SAGA**, which helps it learn faster and better.

We tried two different ways to help the robot figure out which clues are most important:

- **L1 Way**: This makes the robot choose only the most important clues, kind of like picking just the best friends to play with.
- **L2 Way**: This spreads out the importance of more clues, like trying to include more friends so no one feels left out.

Then, we made lists to show which clues were important for each way and compared them. **Table III** shows these clues and how they were ranked differently.



L1 rank	L2 rank	feature name	L1 rank	L2 rank	feature name
1	1	Fwd Seg Size Min	12	15	Init Bwd Win Byts
2	3	URG Flag [illegible]	[illegible]	[illegible]	[illegible]
3	5	SYN Flag Cnt	14	17	Bwd IAT Std
4	6	Fwd PSH Flags	15	18	Flow IAT Max
5	10	Bwd IAT Mean	16	12	Flow Byts/s
6	9	Bwd IAT Min	17	27	Flow IAT Mean
7	4	Fwd Pkt Len Min	18	45	Protocol
8	21	Fwd IAT Min	19	2	Pkt Len Std
9	23	Flow IAT Std	22	13	FIN Flag Cnt
10	24	Fwd IAT Mean	25	51	RST Flag Cnt
11	22	Bwd IAT Max	26	44	PSH Flag Cnt

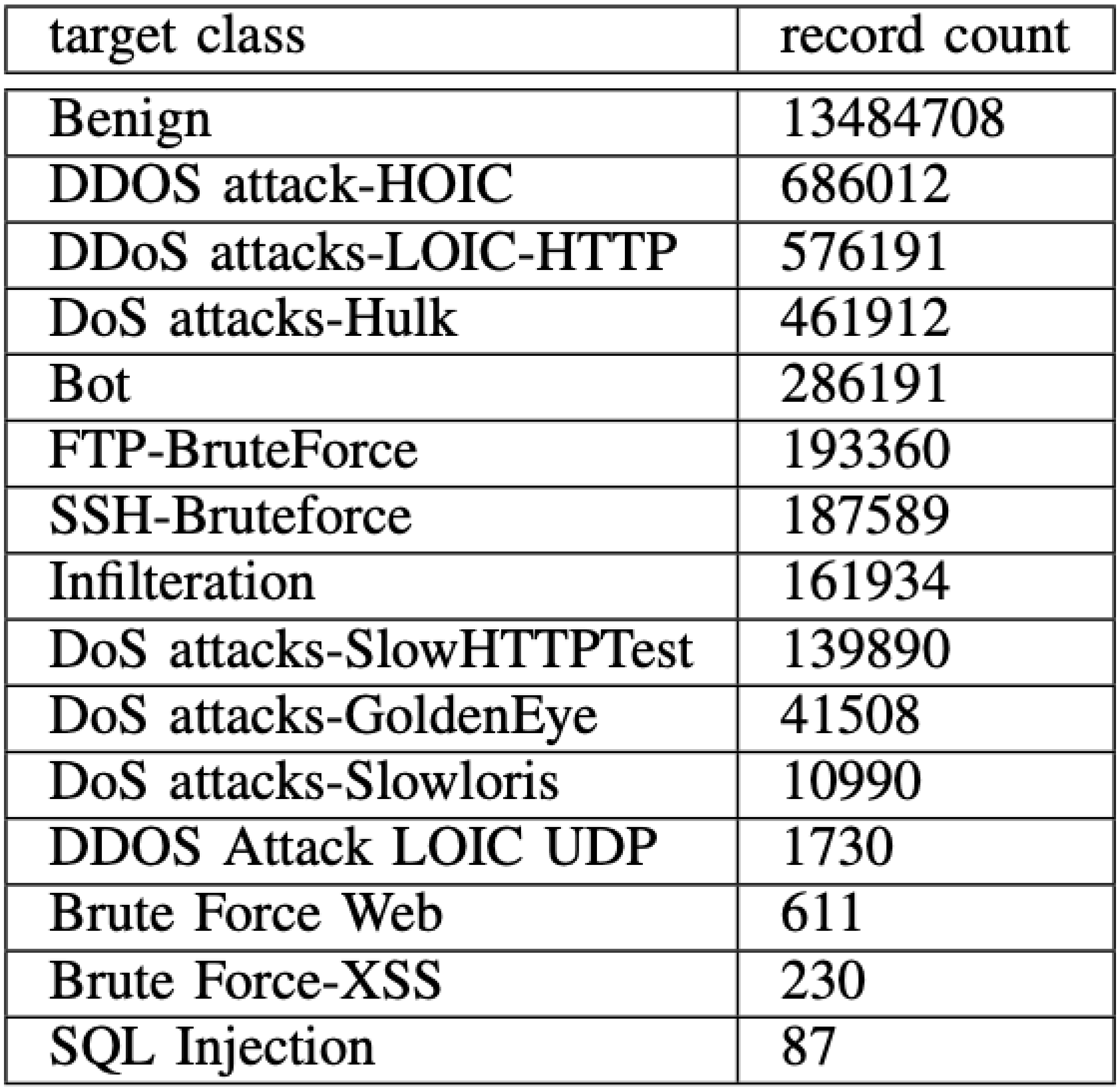
tel:6%2010%209%204%2021%2023%2024%202

TABLE III: common features

Experiment Design:

The experiment was divided into **two parts**:

- **Part One**: *Excluded the problematic class,* resulting in a dataset size of **51,730 samples** from **11 classes**.
- **Part Two**: *Included* the *problematic class* for *further analysis.*



target class	record count
Benign	13484708
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS Attack LOIC UDP	1730
Brute Force Web	611
Brute Force-XSS	230
SQL Injection	87

TABLE I: target classes

The image we see is a table showing the distribution of different types of cyberattacks in a dataset. The table lists each attack type (e.g., DDoS, Brute Force) and the number of records associated with that attack type. The table also includes a "*Benign*" category, which likely *represents normal, non-malicious traffic*.

The data appears to be *heavily skewed towards benign traffic*, with *over 13 million benign* records compared to significantly smaller numbers for each attack type. This suggests that the *dataset* might be *imbalanced*, which could *pose challenges for machine learning models* trained on this data.

Conclusion

In this research, we *developed* a *logistic regression-based feature selection method* to *reduce* the feature set size for training supervised ML models. Using the CIC-IDS2018 dataset, we combined features from LR+L1 and LR+L2 methods, reducing the feature set by 72% (from 78 to 22 features) while maintaining strong model performance.

Our reduced feature set consistently outperformed subsets from individual L1 or L2 rankings and met baseline accuracy targets with fewer features. Even with complex models like Random Forest and Decision Trees, the accuracy loss was less than 1%, demonstrating the method’s effectiveness. Future work will explore applying this approach to other datasets and evaluating its scalability and robustness.