SSH Shell Attacks - Appendix

ANDREA BOTTICELLA*, Politecnico di Torino, Italy
ELIA INNOCENTI*, Politecnico di Torino, Italy
RENATO MIGNONE*, Politecnico di Torino, Italy
SIMONE ROMANO*, Politecnico di Torino, Italy

Contents

A DATA EXPLORATION AND PRE-PROCESSING

This appendix contains additional plots and visualizations related to the data exploration and pre-processing phase of the analysis. The figures are grouped into subsections based on their thematic relevance.

A.1 Temporal Analysis of Attacks

A.1.1 Attack Frequency by Hour

The plot shows the distribution of SSH attacks across different hours of the day. It reveals specific hours when attack activity peaks, which could indicate targeted times for malicious activities. Understanding these patterns can help in implementing time-based security measures.

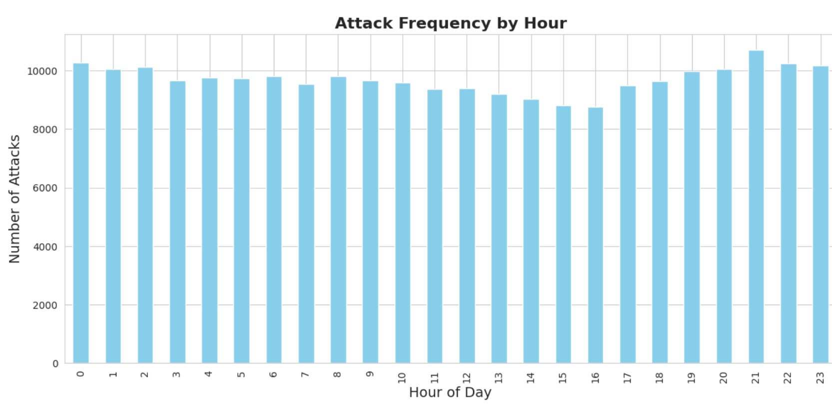


Fig. 1. Distribution of SSH attacks by hour of the day - The plot highlights peak hours during which attacks are most frequent

*The authors collaborated closely in developing this project.

Authors' Contact Information: Andrea Botticella, andrea.botticella@studenti.polito.it, Politecnico di Torino, Turin, Italy; Elia Innocenti, elia.innocenti@studenti.polito.it, Politecnico di Torino, Turin, Italy; Renato Mignone, renato.mignone@studenti.polito.it, Politecnico di Torino, Turin, Italy; Simone Romano, simone.romano2@studenti.polito.it, Politecnico di Torino, Turin, Italy.

Publication date: January 2025.

The attacks are relatively well-distributed throughout the day, but notable peaks occur between 19:00 and 6:00, with lower activity observed from 9:00 to 16:00. This pattern might be explained by the likelihood that attackers schedule their activities during non-working hours when individuals and organizations are less likely to monitor or respond to security incidents.

A.1.2 Attack Frequency by Month

This plot illustrates the distribution of SSH attacks across different months. It highlights seasonal trends, showing months with higher attack frequencies. Such insights can be useful for anticipating periods of increased security threats and allocating resources accordingly.

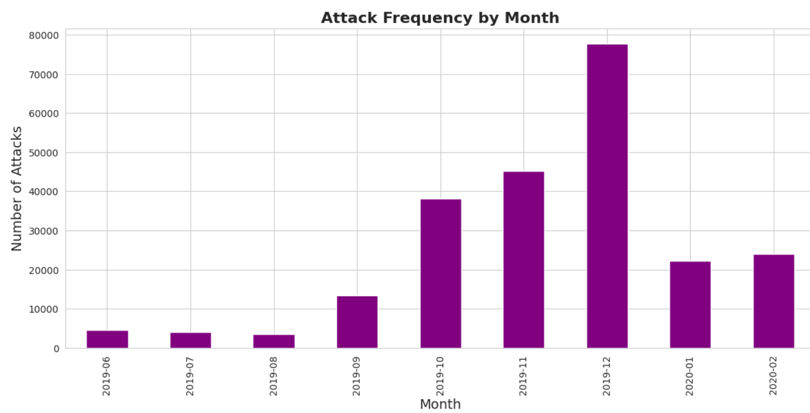


Fig. 2. Distribution of SSH attacks by month - The plot reveals seasonal trends in attack frequency

The number of attacks shows a noticeable increase from October to December, followed by a return to lower and more constant levels in the subsequent months. This trend raises the question of whether a seasonal pattern exists, possibly linked to factors such as end-of-year activities, holidays, or specific campaigns by attackers during this period.

A.1.3 Attack Frequency by Year

The plot compares the frequency of SSH attacks between the years 2019 and 2020.

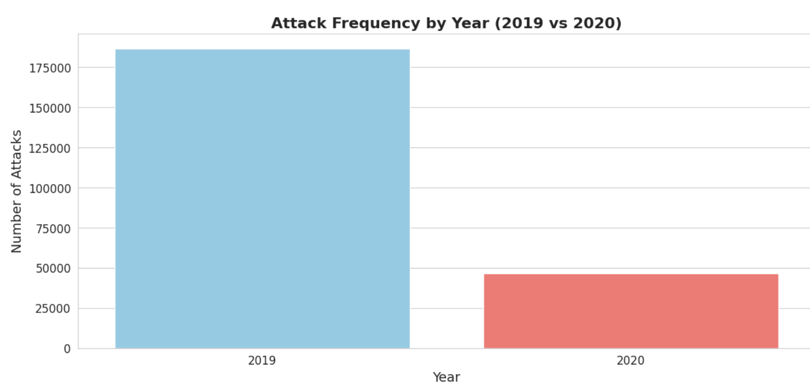


Fig. 3. Distribution of SSH attacks by year - The plot shows the overall trend of attacks over multiple years

A.1.4 Temporal Series of SSH Attacks

This time series plot illustrates the distribution of SSH attacks over time, providing a clear view of how attack frequency fluctuates throughout the dataset. When combined with other plots, such as those highlighting daily or seasonal patterns, it enables a more comprehensive temporal analysis, helping to uncover deeper insights into attack trends and dynamics.

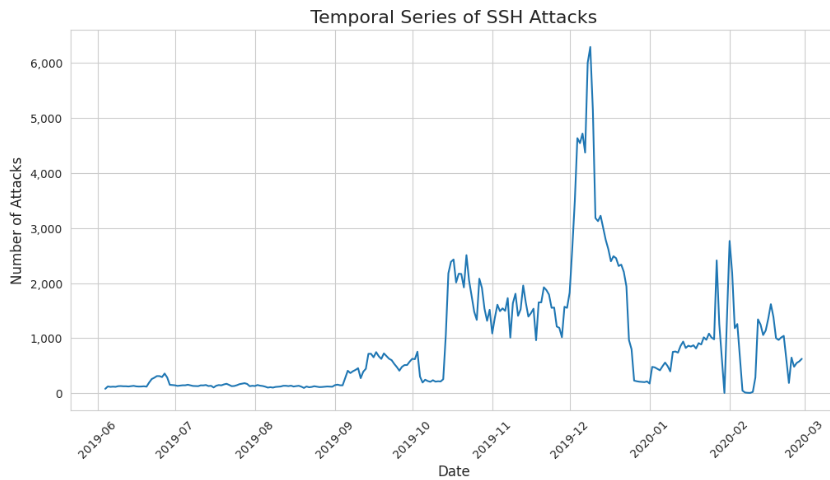


Fig. 4. Time series plot of SSH attacks over the entire dataset - The plot provides a view of attack patterns over time

A.1.5 Intents Over Timestamps

The plot visualizes the distribution of different attack intents over time. It categorizes the intents: Defense Evasion, Harmless, Impact, Discovery, Persistence, Execution, and Others. By analyzing this plot, we can identify which intents are more prevalent at specific times, helping to prioritize security measures based on the nature of the threats.

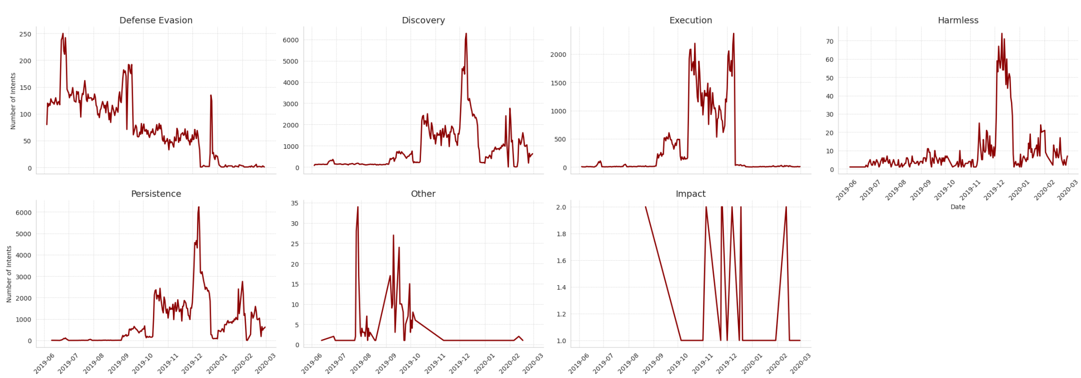


Fig. 5. Attack intents over timestamps - The plot provides insights into the temporal patterns of different attack intents

B SUPERVISED LEARNING - CLASSIFICATION

B.1 Logistic Regression

In this section, we detail the steps and results obtained from using the Logistic Regression model during the supervised learning phase of the project. Logistic Regression served as a baseline model to provide an initial understanding of the classification problem. Despite its simplicity, it offered valuable insights into the multi-label classification task.

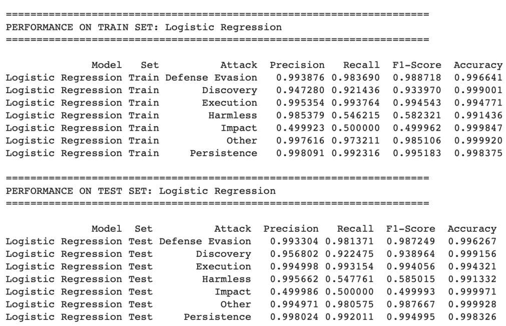
B.1.1 Model Training

The Logistic Regression model was trained using its default configuration. Specifically, the `lbfgs` solver was utilized with a regularization parameter $C = 1$. The training process aimed to identify potential overfitting or underfitting issues and establish baseline performance metrics.

The dataset was preprocessed using the TF-IDF representation of the session texts, which assigned weights to words based on their frequency and relevance within the dataset. Multi-label binary encoding was applied to the `Set_Fingerprint` column to ensure compatibility with the model.

B.1.2 Evaluation Metrics

The Logistic Regression model was evaluated using standard classification metrics, including weighted F1-scores, precision, and recall. The evaluation metrics highlighted the strengths and weaknesses of the model in handling imbalanced classes.



PERFORMANCE ON TRAIN SET: Logistic Regression

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
Logistic Regression	Train	Defense Evasion	0.993876	0.983690	0.988718	0.996641
Logistic Regression	Train	Discovery	0.947280	0.921436	0.933970	0.999001
Logistic Regression	Train	Execution	0.995354	0.993764	0.994543	0.994771
Logistic Regression	Train	Harmless	0.985379	0.546215	0.582321	0.991436
Logistic Regression	Train	Impact	0.499923	0.500000	0.499962	0.999847
Logistic Regression	Train	Other	0.997616	0.973211	0.985106	0.999920
Logistic Regression	Train	Persistence	0.998091	0.992316	0.995183	0.998375

PERFORMANCE ON TEST SET: Logistic Regression

Model	Set	Attack	Precision	Recall	F1-Score	Accuracy
Logistic Regression	Test	Defense Evasion	0.993304	0.981371	0.987249	0.996267
Logistic Regression	Test	Discovery	0.956802	0.922475	0.938964	0.999156
Logistic Regression	Test	Execution	0.994998	0.993154	0.994056	0.994321
Logistic Regression	Test	Harmless	0.995662	0.547761	0.585015	0.991332
Logistic Regression	Test	Impact	0.499986	0.500000	0.499993	0.999971
Logistic Regression	Test	Other	0.994971	0.980575	0.987667	0.999928
Logistic Regression	Test	Persistence	0.998024	0.992011	0.994995	0.998326

Fig. 6. Evaluation Metrics for Logistic Regression Model

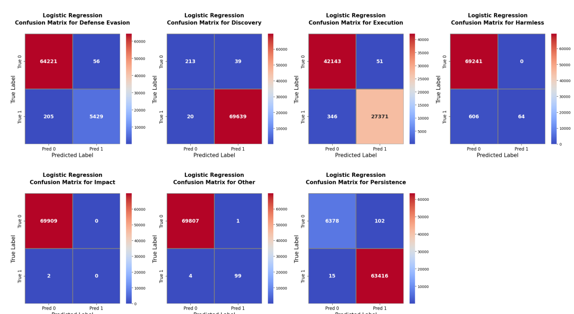


Fig. 7. Confusion Matrix for Logistic Regression Model

The confusion matrix provided a breakdown of true positives, false positives, false negatives, and true negatives for each intent. Figure 11 shows the confusion matrix for the Logistic Regression model.

B.1.3 Hyperparameter Tuning

Grid search was performed to optimize the Logistic Regression model's hyperparameters. The search focused on varying the regularization parameter C over a range of values $[0.1, 1, 10, 100]$ to identify the configuration that maximized weighted F1-scores.

The optimized model exhibited improved performance compared to the baseline, particularly for intents with smaller sample sizes. Figure 10 illustrates the weighted F1-scores for different values of C.

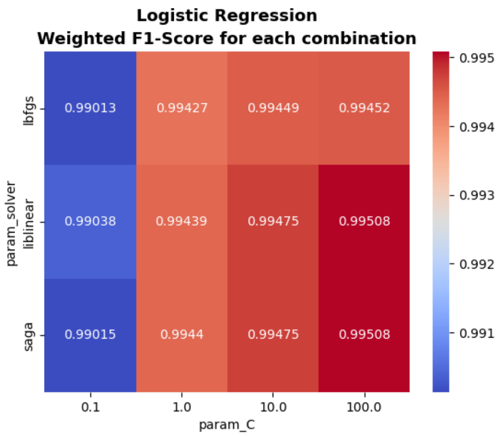


Fig. 8. Weighted F1-Scores for Hyperparameter Tuning

Fig. 9. LR Model Comparison

B.1.4 Comparative Analysis of Baseline and Optimized Models

The optimized Logistic Regression model demonstrated a moderate improvement in precision and recall compared to the baseline. However, its overall performance remained slightly inferior to more complex models like Random Forest and SVM. The comparative analysis underscores the importance of selecting models suited to the dataset's characteristics and problem requirements.

B.2 Random Forest and SVM

B.2.1 Comparative Analysis of Baseline and Optimized Models

text

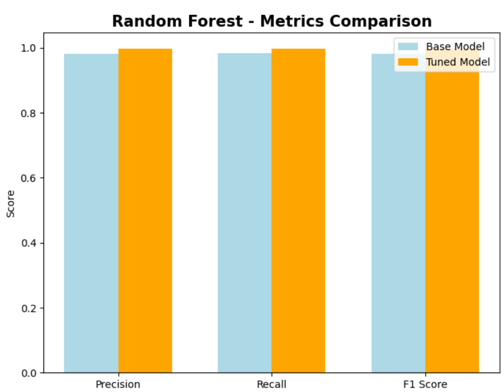


Fig. 10. RF Model Comparison

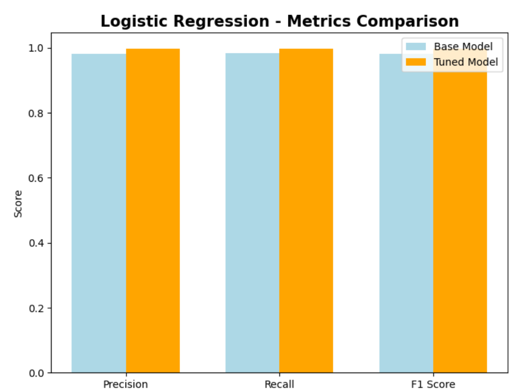


Fig. 11. SVM Model Comparison

C UNSUPERVISED LEARNING - CLUSTERING

```
Cluster 0 - K-Means Community Analysis:

Community 0: ['var', 'wc', 'while', 'wget', 'x13', 'which', 'x17', 'x19', 'xf']

Community 1: ['8m', '20m', '15s', '0kx34uax1rv', '172', '75gvomnx9euwonvnoaje0␣
    qxxziig9elbhp␣glmuakb5bgtfbrkjaw9u9fstdengvs8hx1knfs4mjux0hjok8rvcempecjdy␣
    symb66nylakgwcee6weqhmd1mupghwgq0hwcwsqk13ycgpk5w6hyp5zykfnvlc8hgmd4wwu97k␣
    6pftgtubjk14ujvcd␣9iukqttwyyjiiu5pmuux5bsz0r4wfwdie6i6rblaspkgaysvkprkorw',
    '192', '3s']

Community 2: ['admin', 'bs', '9p7vd0epz3tz', 'bin', 'aaaab3nzac1yc2eaaaabjqaaa␣
    qeardp4cun2lhr4kuhbge7vvacwdli2a8dbnrtorbmz15o73fcbox8nvbut0buanuv9tj2',
    'awk', 'authorizedkeys', 'bash']

Community 3: ['cpuinfo', 'chpasswd', 'chmod', 'cat', 'busybox', 'cd', 'count',
    'cp']

Community 4: ['dota', 'done', 'do', 'dota3', 'dev', 'dd', 'echo', 'crontab']

Community 5: ['enable', 'go', 'free', 'exit', 'grep', 'gz', 'if', 'head']

Community 6: ['ls', 'initall', 'mdrfckr', 'lscpu', 'model', 'mkdir', 'lh',
    'mem']

Community 7: ['mv', 'passwd', 'null', 'name', 'mounts', 'password', 'nohup',
    'mountfs']

Community 8: ['proc', 'rsync', 'root', 'rsa', 'rm', 'rf', 'read', 'print']

Community 9: ['shm', 'ssh', 'system', 'tar', 'systemcache', 'sleep', 'sh',
    'shell']

Community 10: ['tftp', 'tsm', 'unix', 'up', 'top', 'txt', 'tmp', 'uname']
```

Listing 1. Cluster 0 - K-Means Community Analysis

```
Cluster 0 - GMM Community Analysis:

Community 0: ['var', 'wc', 'while', 'wget', 'x13', 'which', 'x17', 'x19', 'xf']

Community 1: ['8m', '20m', '15s', '0kx34uax1rv', '172', '75gvomnx9euwonvnoaje0␣
    qxxziig9elbhpglmuakb5bgtfbrkjaw9u9fstdengvs8hx1knfs4mjux0hjok8rvcempecjdys␣
    ymb66nylakgwcee6weqhmd1mupghwgq0hwcwsqk13ycgpk5w6hyp5zykfnvlc8hgmd4wwu97k6␣
    pftgtubjk14ujvcd9iukqttwyyjiiu5pmuux5bsz0r4wfwdie6i6rblaspkgaysvkprkorw',
    .'192', '3s']
```

```
Community 2: ['admin', 'bs', '9p7vd0epz3tz', 'bin', 'aaaab3nzac1yc2eaaaabjqaa␣
    aqeardp4cun2lhr4kuhbge7vvacwdli2a8dbnrtorbmz15o73fcbox8nvbut0buanuv9tj2',
    'awk', 'authorizedkeys', 'bash']

Community 3: ['cpuinfo', 'chpasswd', 'chmod', 'cat', 'busybox', 'cd', 'count',
    'cp']

Community 4: ['dota', 'done', 'do', 'dota3', 'dev', 'dd', 'echo', 'crontab']

Community 5: ['enable', 'go', 'free', 'exit', 'grep', 'gz', 'if', 'head']

Community 6: ['ls', 'initall', 'mdrfckr', 'lscpu', 'model', 'mkdir', 'lh',
    'mem']

Community 7: ['mv', 'passwd', 'null', 'name', 'mounts', 'password', 'nohup',
    'mountfs']

Community 8: ['proc', 'rsync', 'root', 'rsa', 'rm', 'rf', 'read', 'print']

Community 9: ['shm', 'ssh', 'system', 'tar', 'systemcache', 'sleep', 'sh',     '
    shell']

Community 10: ['tftp', 'tsm', 'unix', 'up', 'top', 'txt', 'tmp', 'uname']
```

Listing 2. Cluster 0 - GMM Community Analysis

D LANGUAGE MODEL EXPLORATION

D.1 Training Configuration

The model was implemented using BERT (bert-base-uncased) with the following key configurations:

- Maximum sequence length: 128 tokens
- Learning rate: 4e-5 with AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e-6$)
- Training epochs: 4
- Gradient accumulation steps: 4
- Mixed precision training enabled
- Linear learning rate scheduler without warmup
- Loss function: Binary Cross-Entropy with Logits

D.2 Model Performance Metrics

D.2.1 3D ROC Analysis The 3D ROC curve (Figure 12) illustrates:

- Superior performance for Execution class (green line)
- Strong performance for Discovery class (orange line)
- Slightly lower but still good performance for Defense Evasion (blue line)

D.2.2 Class-wise F1 Scores The model demonstrates varying performance across classes (Figure 13):

- Excellent performance (F1 $\geq$ 0.98) for Defense Evasion, Discovery, Execution, Other, and Persistence classes
- Perfect scores (F1 = 1.00) for Discovery and Persistence
- Significantly lower performance for Harmless class (F1 = 0.22)
- Impact class shows minimal detection capability (F1 = 0.00)

D.2.3 Detailed Performance Metrics The performance metrics (Figure 14) reveal:

- Most classes achieve balanced precision and recall scores
- The Harmless class shows a significant disparity between precision and recall
- The Impact class shows minimal performance across all metrics

D.2.4 Precision-Recall Analysis The Precision-Recall curves (Figure 15) demonstrate:

- Most classes maintain high precision (>0.95) across different recall thresholds
- The "Other" category shows a sharp decline in precision at approximately 0.2 recall
- Impact and Harmless classes demonstrate poor precision-recall trade-offs
- Defense Evasion, Discovery, Execution, and Persistence maintain near-perfect precision until very high recall values

D.2.5 Prediction Probability Distribution The prediction probability histograms (Figure 16) reveal:

- Most classes exhibit a strong binary separation in prediction probabilities
- Defense Evasion, Discovery, and Execution show high-confidence predictions clustered near 0 and 1
- The Persistence class shows a similar pattern but with a smaller proportion of low-probability predictions
- Harmless and Impact classes show predominantly low-probability predictions, indicating potential class imbalance issues

D.3 Recommendations for Model Improvement

Based on the analysis, several potential improvements could be considered:

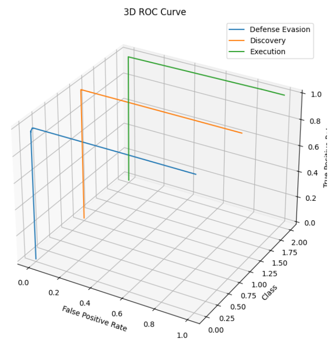


Fig. 12. 3D ROC curves showing the performance trade-off between true positive rate and false positive rate across different classification thresholds for Defense Evasion, Discovery, and Execution classes.

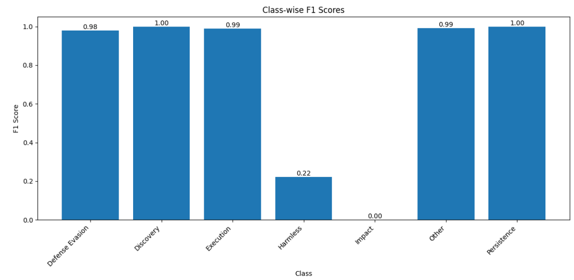


Fig. 13. F1 scores across different classes showing the model's classification performance for each category.

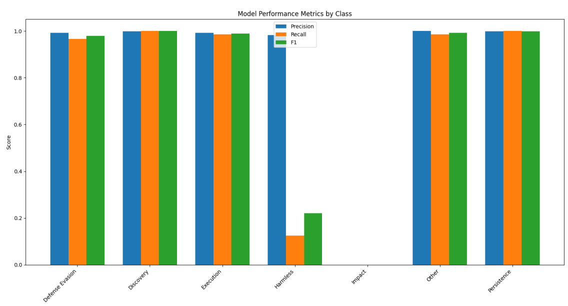


Fig. 14. Detailed breakdown of Precision, Recall, and F1 scores for each class.

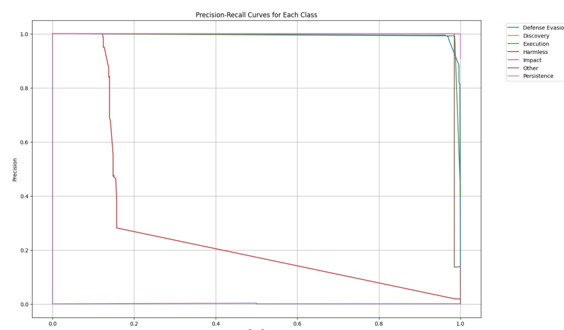


Fig. 15. Precision-Recall curves for each class showing the trade-off between precision and recall at different classification thresholds.

D.3.1 Class Imbalance Mitigation

- Implement class weights or sampling techniques for Harmless and Impact classes
- Consider data augmentation for underrepresented classes

D.3.2 Model Architecture

- Experiment with different BERT variants
- Consider ensemble approaches for improving performance on challenging classes

D.3.3 Training Strategy

- Implement curriculum learning for difficult classes
- Explore different learning rate schedules
- Consider longer training with early stopping

D.3.4 Data Quality

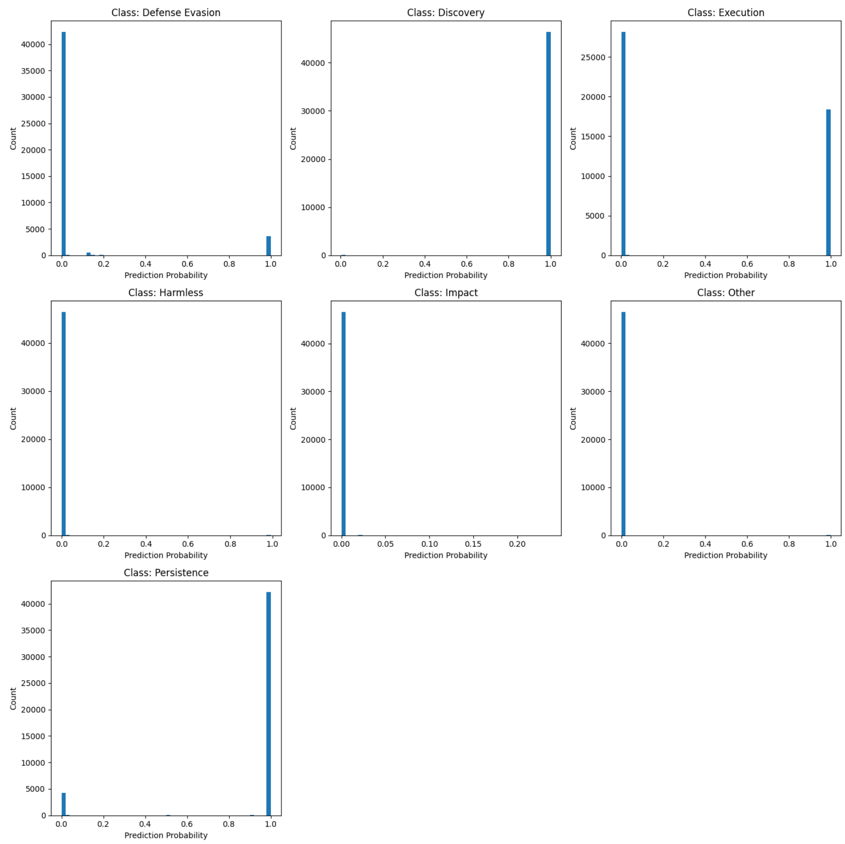


Fig. 16. Distribution of prediction probabilities for each class showing the model's confidence in its predictions.

- Review and potentially relabel samples in the Impact class
- Analyze misclassified examples in the Harmless class