Machine Learning Report

Leïna Ben Bella
Cholé Daunas

1 Introduction

This report presents the methodology, preprocessing steps, models, evaluations, and results obtained during the analysis of connection log data. The goal of the project is to classify network events into different categories, including binary classification of connection types (Normal vs Suspect) and multiclass prediction of failure types.

2 Data Preprocessing

The preprocessing phase aimed to transform the raw log data into a structured and clean dataset suitable for analysis and modeling. Initially, the Time column, when present, was converted to a datetime format, allowing the extraction of the hour of each event into a new column, Hour. This temporal feature enables the computation of metrics such as the number of connections per IP per hour. In cases where the Time column was missing, a default value of -1 was assigned to the Hour column to avoid missing values.

Next, several features were extracted directly from the text content of each log entry. IP addresses were identified using a regular expression and stored in a dedicated column, with missing values replaced by "0.0.0.0". Usernames were similarly extracted by detecting the word following `user`, with unknown entries replaced by "unknown". Port numbers were captured from the text, converted to numeric values, and missing entries replaced by 0. Additionally, a boolean feature was created to indicate whether an event involved pre-authentication, based on the presence of the `[preauth]` tag.

Event classification was then performed to reduce the complexity of the data. Using the EventId column, events were labeled as either normal "N" or suspect "S" (cf documentation with information about each event). This approach reduced the number of categories from 26 to just 2, simplifying subsequent analysis and machine learning tasks.

Finally, unnecessary columns were removed from the dataset. While the original EventId was discarded after classification, the Content column was re-

tained for reference despite being decomposed into more meaningful features. The final dataset thus included `ip`, `user`, `port`, `preauth`, `Hour`, `conn_per_hour`, `connection_type`, and `Content`.

Overall, this preprocessing pipeline effectively converted raw, unstructured log data into a clean, structured, and enriched dataset that supports both descriptive analysis and predictive modeling.

Figure 1: Failed SSH

3 Baseline Models

Baseline models were trained to evaluate the predictive power of the engineered features. For binary connection-type prediction (Normal vs. Suspect), a Logistic Regression classifier was used within a preprocessing pipeline: numerical features (`port`, `Hour`, `conn_per_hour`) were passed through, boolean features (`preauth`) were kept, and categorical features (`ip`, `user`) were one-hot encoded. The initial model achieved near-perfect accuracy, revealing that `ip` strongly correlated with the target. Removing both `ip` and `user` produced a more realistic performance of 96.5% accuracy, with high precision and recall for both classes.

For predicting failure types, a new variable `FailureType` was extracted from the log content. A Linear SVM was trained on scaled numerical features, boolean features, and one-hot encoded categorical features. Including `user` initially led to overly optimistic results, so it was removed. The final model achieved 89% accuracy, performing well on frequent failure types while reflecting lower recall on rarer classes.

These baseline models demonstrated the predictive value of the engineered features and highlighted potential pitfalls such as feature leakage, providing a reference point for further optimization with hyperparameter tuning and ensemble methods.

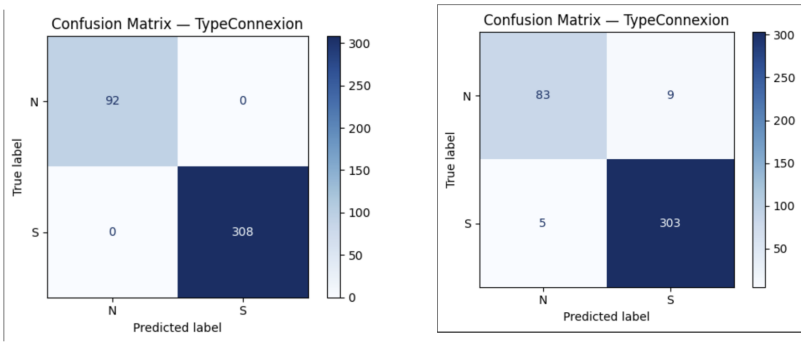

Figure 2: Confusion matrix with user and ip and Confusion matrix without user and ip

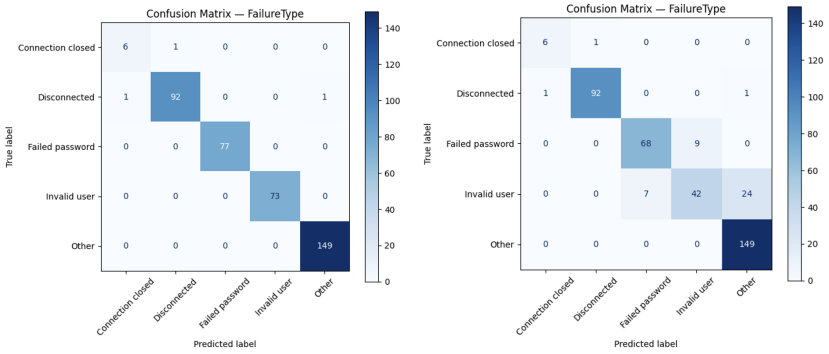

Figure 3: Confusion matrix with user and Confusion matrix without user

4 Grid Search Optimization

GridSearchCV was applied to both the `FailureType` and connection-type models to identify the best hyperparameters using cross-validation.

For the FailureType model (Linear SVM), the best parameters were `C=1` and `max_iter=500`. The optimization did not improve performance, with overall accuracy remaining at 89%. This is likely due to the small, imbalanced dataset and limited lexical variability. Most classes were already well learned, except "Invalid user", which remains difficult to predict without the `user` feature.

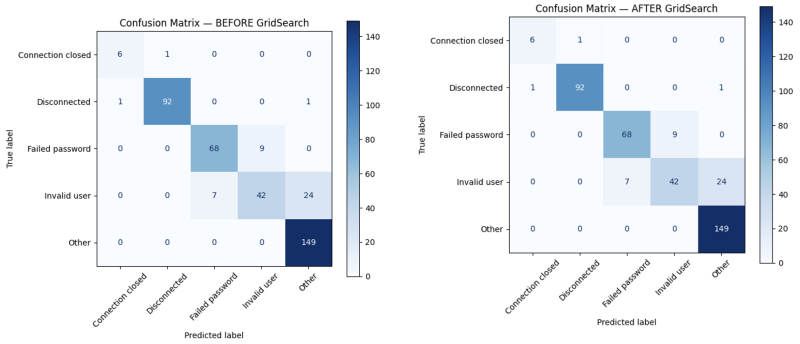

Figure 4: Confusion matrix BEFORE GridSearchand Confusion matrix AFTER GridSearch

For the connection-type model (Logistic Regression), the best parameters were `C=0.1` with `l2` penalty. After optimization, recall for the suspect class reached 1.0, eliminating false negatives, but precision for the normal class dropped slightly, increasing false positives. Overall, total errors rose from 14 to 17. The optimized model became more sensitive to suspect connections but less precise for normal ones, illustrating a trade-off between class sensitivities.

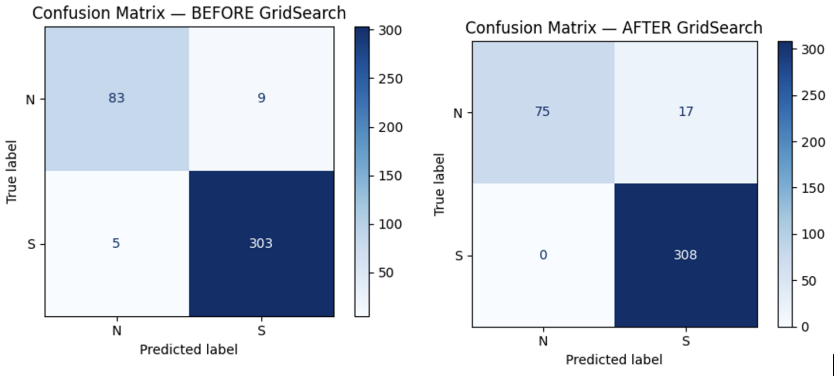

Figure 5: Confusion matrix BEFORE GridSearchand and Confusion matrix AFTER GridSearch

These results show that hyperparameter tuning can adjust class sensitivity but cannot overcome the inherent limitations of the dataset, emphasizing the importance of both data quality and model selection.

5 Ensemble Modeling with Voting and Bagging

To improve `connection_type` prediction, a Voting Classifier combining a Logistic Regression and a Linear SVM was used, with each base model wrapped in a BaggingClassifier. Bagging reduces variance by training multiple estimators on random subsets of the data, while Voting aggregates predictions from the different models.

The ensemble achieved the same results as the baseline Logistic Regression, with 96.5% accuracy and similar precision and recall for both classes. This is expected because the dataset is relatively small and simple, and the features are already highly discriminative, making it difficult to improve performance. Furthermore, Logistic Regression is already a stable classifier, so applying Bagging does not significantly increase accuracy—it mainly helps reduce variance and improve robustness.

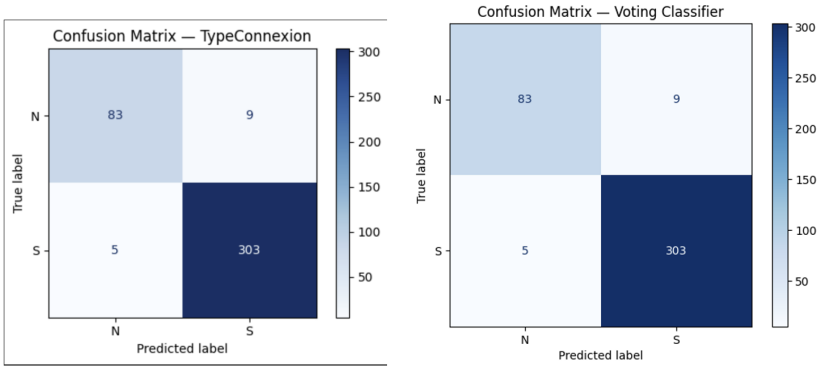

Figure 6: Confusion matrix TypeConnexion and Confusion matrix Voting Classifier

6 Conclusion

The project successfully built, optimized, and evaluated multiple machine learning models for network event classification. Further improvements could be achieved using weighted losses, anomaly detection models, or deep learning architectures.