From Clues to Capabilities: ML-Powered MITRE Technique Prediction

Gil Eskayo
7 August 2025

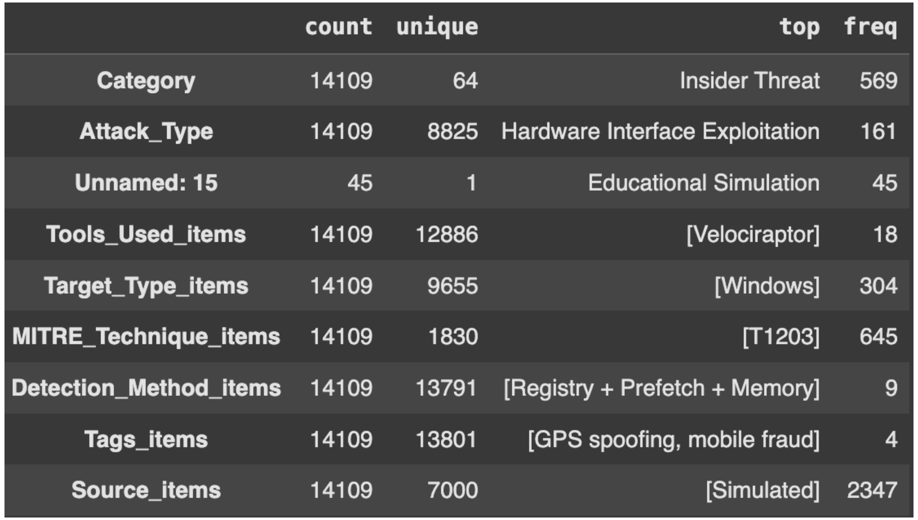
Project Overview

- Problem I am trying to solve
  - I am using this dataset to look at giving insight to common fixable vulnerabilities in a system by predicting types of MITRE Att&cks given categorical features about previous attacks.
- Why did you choose this dataset or topic?
  - I chose this topic because I am interested in Cyber security

Dataset Summary

https://www.kaggle.com/datasets/tannubarot/cybersecurity-attack-and-defence-dataset

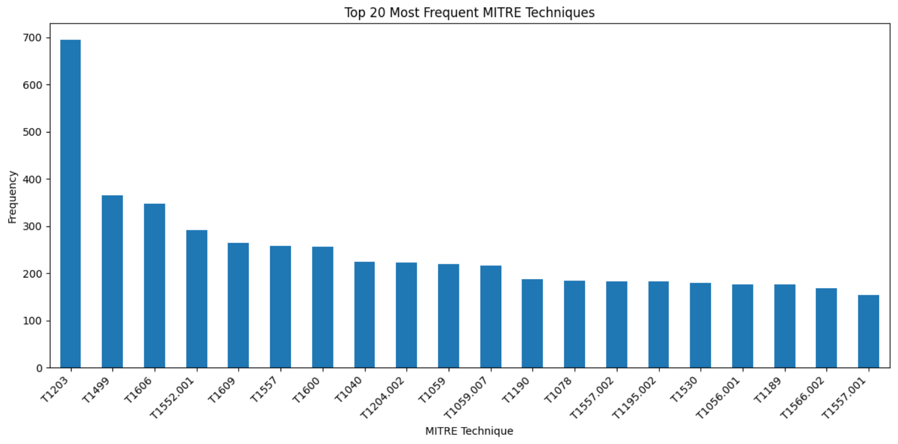
- Dropped 24 missing target rows (one-hot encoded classification)
- Standardized inconsistent columns—fixed multiple labels per row by converting them into dictionaries
- Reduced dimensionality by creating "Other" bins to manage skew and large feature sets

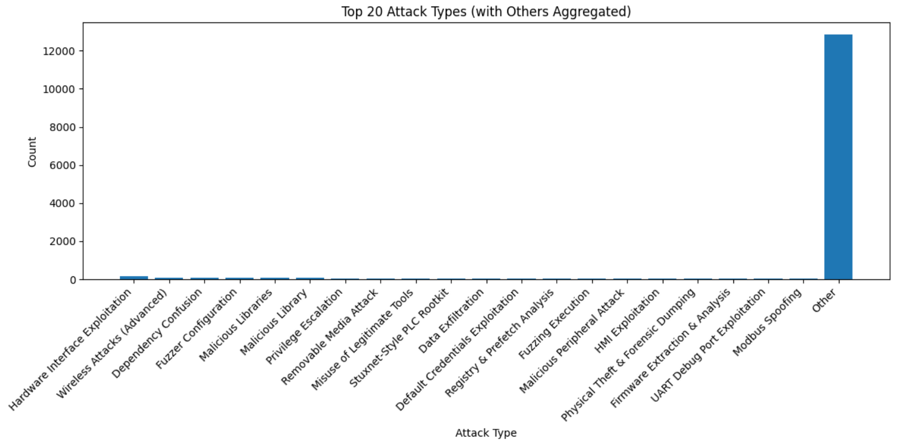


	count	unique	top	freq
Category	14109	64	Insider Threat	569
Attack_Type	14109	8825	Hardware Interface Exploitation	161
Unnamed: 15	45	1	Educational Simulation	45
Tools_Used_items	14109	12886	[Velociraptor]	18
Target_Type_items	14109	9655	[Windows]	304
MITRE_Technique_items	14109	1830	[T1203]	645
Detection_Method_items	14109	13791	[Registry + Prefetch + Memory]	9
Tags_items	14109	13801	[GPS spoofing, mobile fraud]	4
Source_items	14109	7000	[Simulated]	2347

Exploratory Data Analysis

- MITRE Technique distribution is highly imbalanced → Needed sampling or binning
- Attack Type heavily skewed to “Other” → Created bins to reduce dimensionality
- Surprising Insight: Specific tools (like Burp Suite or AFL++) co-occurred frequently with specific techniques
- Influenced modeling choices: Used label smoothing and feature encoding strategies to handle multi-label & imbalance issues





Preprocessing Pipeline

- Cleaned & tokenized multi-label string columns (e.g., tools, tags) into Python lists using regex
- Dropped raw text columns and null targets to prepare for classification with multi-label encoding
- Applied OneHotEncoding for single-label features and MultiLabelBinarizer for list-type features via ColumnTransformer

```
# OneHotEncode Pipeline
ohe_pipe = Pipeline([
    ('encoder',
OneHotEncoder(handle_unknown='ignore'))
])
# MultiLabelBinarizerTransformer Pipeline
list_column_pipe = Pipeline([
    ('multi_label_binarizer',
MultiLabelBinarizerTransformer())
])
# Preprocess Pipeline
preprocess = ColumnTransformer([
    ('cat',  ohe_pipe, solo_feats),
    ('lists',
MultiColumnMultiLabelBinarizer(list_feats),
list_feats)
])
```

Model Training and Evaluation

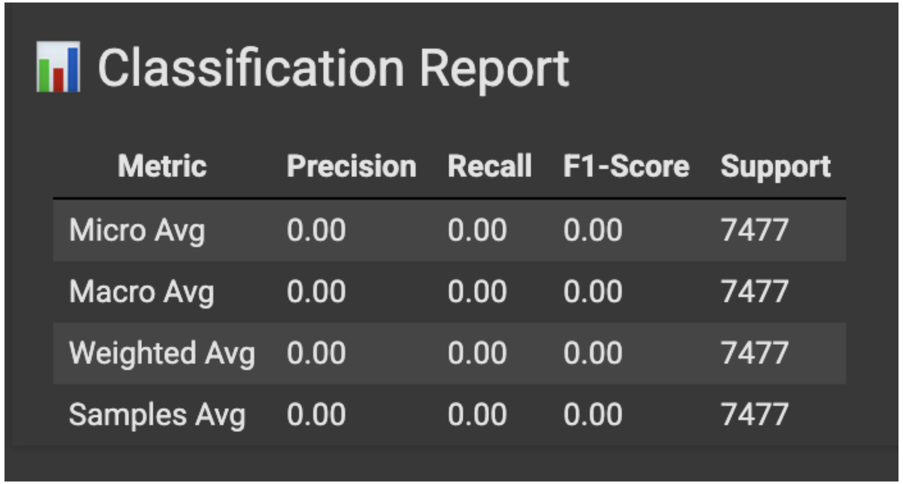
Baseline trio:

- Naive Bayes (fast baseline)
- Linear-SGD (one-vs-rest)
- LightGBM (gradient boosting)

Why not KNN?

- My 10^4+ binary features make nearest-neighbor distances unreliable and slow.

Results and Model Comparison



📊 Classification Report

Metric	Precision	Recall	F1-Score	Support
Micro Avg	0.00	0.00	0.00	7477
Macro Avg	0.00	0.00	0.00	7477
Weighted Avg	0.00	0.00	0.00	7477
Samples Avg	0.00	0.00	0.00	7477

Interpretations and Insights

- Naive Bayes model took over an hour to run and cross-validate
- It failed to predict any labels, indicating poor dataset-model compatibility
- SGD model is still running
- Dataset may not be well-suited for multi-label classification or requires further preprocessing

Reflection and Challenges

- Required extensive data cleaning due to messy, multi-label formatting
- Final processed dataset was very large, causing RAM overload or crashed processes
- Model training was extremely slow or failed, limiting testing and improvement cycles
- Still working on bug fixes and performance optimizations to improve training speed

□Conclusions

- Features like `Tools_Used`, `Attack_Types`, `Target_Types`, `Detection_Method`, `Tags`, and `Source` were **not sufficient** to accurately predict MITRE techniques (vulnerabilities targeted)
- Surprising outcome: A **holistic heuristic approach** didn’t yield meaningful correlation
- Indicates that attack behavior is **more complex or context-dependent** than anticipated

□Next Steps

- **Refine preprocessing and feature engineering** to reduce noise and improve signal
- **Experiment with bagging or ensemble methods** to boost model performance

References

1. Medium article
Murpani, R. (2023, December 15). *A comprehensive guide to multiclass classification in machine learning*. Medium. Retrieved August 7, 2025, from Medium website:
https://medium.com/@murpanironit/a-comprehensive-guide-to-multiclass-classification-in-machine-learning-c4f893e8161d Medium+1

2. Scikit-learn — SGDClassifier API documentation
scikit-learn developers. (n.d.). *SGDClassifier — scikit-learn documentation*. scikit-learn. Retrieved August 7, 2025, from scikit-learn website:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html Scikit-learn

3. Scikit-learn — Naive Bayes documentation
scikit-learn developers. (n.d.). *Naive Bayes — scikit-learn documentation*. scikit-learn. Retrieved August 7, 2025, from scikit-learn website:
https://scikit-learn.org/stable/modules/naive_bayes.html Scikit-learn+1

4. Scikit-learn — Multiclass and multioutput algorithms documentation
scikit-learn developers. (n.d.). *Multiclass and multioutput algorithms — scikit-learn documentation*. scikit-learn. Retrieved August 7, 2025, from scikit-learn website:
https://scikit-learn.org/stable/modules/multiclass.html Scikit-learn