# Intrusion Detection with Machine Learning

Student Lab

## Overview

In this lab, you will work with network log records, train multiple classifiers, inspect their metrics, compare their behavior, and explain what the selected model output would mean for an intrusion detection workflow.

The goal is not to build a production IDS. The goal is to practice connecting a security task to data, model outputs, evaluation metrics, and human review.

Throughout the lab, "validation" refers to the local train/validation split you create from the training data. "Test" is reserved for the private competition test set used for bonus scoring.

## Learning Goals

By the end of the lab, students should be able to:

- describe how network log features can support a simple intrusion detection task;
- train and evaluate several basic classifiers on labeled security data;
- compare at least three classification techniques and explain which worked best;
- compare a train/validation score with five-fold cross-validation results;
- interpret false positives and false negatives as operational tradeoffs;
- explain what a human analyst would still need to review before acting on an alert.

## Dataset

The starter package includes two synthetic tables. Each row represents a single network connection summary.

- `ids_flows_dev.csv`: a small development set for inspection and discussion;
- `ids_flows_train.csv`: a 5,000-row synthetic training set for model development, train/validation evaluation, and cross-validation.

The instructor has a private competition test set for bonus scoring. The competition test set is the real Kaggle "Cybersecurity Threat Detection Dataset" and is not included in the starter package. The synthetic training file uses the same 13 columns as the competition test file so that your trained pipeline can be applied directly.

The fields are:

- `timestamp`, `src_ip`, `dst_ip`, `src_port`, `dst_port`, `protocol`;
- `bytes_sent` and `bytes_received`;
- `user_agent` and `url`;
- `is_internal_traffic` as a boolean;
- `label` as an integer where `0` means benign and `1` means an attack;
- `attack_type` as a string giving the specific attack category (for example `port-scan`, `sql-injection`, `ddos`) or `benign`.

The synthetic training data is intentionally simplified. It is appropriate for demonstrating workflow and tradeoffs, but it should not be presented as realistic network telemetry.

## Dataset Difficulty

Security data can be made easy or hard to classify by design. A dataset becomes easier when one or two fields almost directly reveal the label, the classes are cleanly separated, the class balance is even, and there is little noise. A dataset becomes harder when benign and attack behavior overlap, labels are imperfect, attacks are rare, and the competition test data differs from the training data. The synthetic training set is fairly clean. The private competition test set is real, more imbalanced, and noisier.

In this lab, you should look for signs that the training set is too clean. If a model performs very well on the train/validation split, that does not automatically mean the model will generalize to the competition test set, and it does not mean the model is ready for deployment.

## Feature Leakage Rule

Use `label` as the binary target. Do not use `label` as an input feature. Do not use `attack_type` as an input feature for the binary classifier because it reveals whether the row is benign or an attack. You may use `attack_type` after prediction to analyze which attack categories were easier or harder to detect.

## Your Task

Students complete a Colab notebook that asks them to:

1. load the synthetic training dataset;
2. inspect class balance, attack types, and feature distributions;
3. split the training data into training and validation sets;
4. create and justify derived features in code;
5. train at least three classifier types, choosing from reasonable supervised-learning options;
6. compare which classifier works best and explain why;

7. evaluate accuracy, precision, recall, and the confusion matrix;
8. run five-fold cross-validation to estimate whether the result is stable;
9. designate the best classifier in `best_model.ipynb`;
10. adjust or discuss a decision threshold;
11. use `attack_type` to interpret which categories of attack are easier or harder to detect;
12. write a short interpretation of what the results mean for IDS use.

The notebook contains TODOs in both code cells and markdown cells. Code TODOs affect whether the workflow runs. Markdown TODOs ask you to explain decisions and interpret results. Both kinds of work are graded as part of the total submission.

## Classifier Comparison Requirement

Your submission must include performance results for at least three different classifiers. The first three classifiers should be different classifier types, such as logistic regression, a decision tree, a random forest, k-nearest neighbors, naive Bayes, gradient boosting, or a support vector machine. Do not count the same classifier with small parameter changes as three different classifiers.

For each classifier, report validation performance and five-fold cross-validation performance. Then identify one best classifier, explain why you selected it, and implement that selected pipeline in `best_model.ipynb`.

Your comparison should discuss more than the final score. Consider whether the model is easy to explain, whether it handles nonlinear feature interactions, whether it may overfit, and whether its false positives or false negatives are acceptable for an IDS workflow.

## Evaluation Notes

Complete `evaluation_notes.md` as the main written summary of your model evidence. This file is not a place to paste every notebook output. Instead, use it to copy the key metrics, identify the model settings, and explain what the results mean for intrusion detection.

The model comparison in `evaluation_notes.md` is organized as a list of classifier blocks rather than a table. Fill in one block for each classifier. For each block, include:

- the classifier name and important settings;
- validation F1 from the train/validation split;
- five-fold cross-validation mean F1;
- five-fold cross-validation F1 standard deviation;
- the model's main strength;
- the model's main weakness.

For each prompt in `evaluation_notes.md`, write your response on the indented line that begins `Answer here:`. Replace that placeholder with your own answer. Round numeric metrics to three decimals unless your instructor gives a different format.

`evaluation_notes.md` should also identify your selected classifier, summarize the selected model's accuracy, precision, recall, F1, cross-validation results, and confusion-matrix observation, and explain the security meaning of the observed false positives and false negatives.

## Colab Setup

The notebook runs in Google Colab and reads its data from your Google Drive.

1. Unzip the starter package on your computer. You will get a folder named `topic-06-ids-lab`.
2. Upload the whole `topic-06-ids-lab` folder to your Google Drive, for example at the top of *My Drive* so it lives at `My Drive/topic-06-ids-lab`. Upload the folder as-is so the `data/` subfolder comes along with it.
3. Open `ids_colab_explorer.ipynb` in Colab and run the first code cell. It calls `drive.mount('/content/dri` and prompts you to authorize Drive access, then sets `DATA_DIR = '/content/drive/MyDrive/topic-06-ids` and loads the CSV files from there.
4. If you uploaded the folder somewhere other than the top of My Drive, edit `DATA_DIR` to match your location.

## Notebook And Submission Workflow

The notebook is your exploration and evidence file. Use it to inspect the data, create derived features, compare at least three classifiers, run validation and cross-validation, choose a threshold if useful, and justify your selected approach.

`best_model.ipynb` is the reproducible scoring file. After you finish the notebook, transfer your final feature engineering and selected classifier into `best_model.ipynb`. The instructor will run that file against the private competition test set. The feature engineering, selected classifier, threshold, and model name in `best_model.ipynb` should match what you describe in the notebook, `evaluation_notes.md`, and `reflection.md`. Your `best_model.ipynb` should define a complete pipeline through:

- `add_features(df)` for any derived features;
- `make_features(df, feature_columns=None)` for dropping leakage columns and encoding features;
- `train_best_model(train_df)` for fitting your selected model;
- `predict_best_model(model, metadata, test_df)` for returning 0/1 predictions.

### AI Agent Use

You may use AI agents or assistants for debugging, explaining code, or brainstorming features and model choices. The starter package includes an `AGENTS.md` file that describes the intended boundaries for agent help. Work within those constraints when using an agent for this assignment.

The point is to learn how the modeling choices, metrics, and security interpretation fit together, not to outsource the reasoning.

An agent may help you understand errors, compare possible approaches, or think through alternatives. You remain responsible for the final feature choices, classifier choices, written explanations, security interpretation, and submitted code. Record meaningful AI help in `ai_assistance_log.md`.

## Questions To Answer

Complete these questions in `reflection.md`. Your answers may refer to specific metrics, confusion matrices, or model-comparison notes from your notebook and `evaluation_notes.md`. As in `evaluation_notes.md`, write each response on the indented line that begins `Answer here:` and replace the placeholder with your own answer.

- Which errors are more costly in this lab: false positives or false negatives?
- What would a human analyst need to see before acting on a model alert?
- What assumptions does the synthetic dataset make about attack behavior?
- How would your conclusions change if the dataset were much larger, noisier, or more imbalanced?
- Did the train/validation result and cross-validation result tell the same story?
- Which classification technique worked best, and how does that connect to supervised learning concepts from Week 1?
- What would make a private competition test set fair for comparing different models?
- What additional evidence would you want before deploying this model in a security workflow?

## Bonus Competition

Submit all three classifier results in your notebook and `evaluation_notes.md`. Also submit the starter file named `best_model.ipynb`. In that file, implement your selected feature pipeline and best classifier. The instructor will run your selected model on the private competition test set, which is the real Kaggle "Cybersecurity Threat Detection Dataset."

Bonus points are awarded by competition ranking:

- top 5 competition scores: 5 bonus points;
- top 2 competition scores: 10 total bonus points;
- top 1 competition score: 15 total bonus points.

## Submission Package

Submit the following files:

- `ids_colab_explorer.ipynb`, completed as your exploration and evidence notebook, with code TODOs completed and notebook `Answer here:` placeholders replaced;

- `evaluation_notes.md`, completed with classifier blocks for at least three classifiers, validation performance, five-fold cross-validation performance, selected-model metrics, and security interpretation;
- `best_model.ipynb`, completed with your final feature pipeline, selected classifier, threshold choice, and prediction logic;
- `incident_review.md`, completed with alert interpretation, false-positive and false-negative review, and escalation notes;
- `reflection.md`, completed with answers to the *Questions To Answer* section, including model-difference, pipeline-transfer, limitation, and human-review reflections;
- `ai_assistance_log.md`, if you used AI assistance.

## How To Submit

Submit your completed package through the course **submission Google Form** (your instructor will share the link on the course page or in class).

1. Enter your name and any identifier the form asks for.
2. Use the file-upload question to attach your files. At minimum attach `best_model.ipynb`, which is the file scored for the bonus competition. Attach the other submission-package files as well unless your instructor collects them separately.
3. In the short-answer fields, enter the name of your selected model (for example, "random forest") and a one-line description of your final pipeline for the competition leaderboard.
4. Submit the form. Uploading requires you to be signed in to your Google account, the same one you used for Colab.

Do not rename `best_model.ipynb`. The form records your name automatically, and the instructor's scoring script expects the standard filename and function names (`add_features`, `make_features`, `train_best_model`, `predict_best_model`).

## Security Takeaway

A model result is not a security decision by itself. You should be able to explain what task the model supports, what evidence it uses, what its errors cost, and where human judgment remains necessary.

## AI Development Acknowledgment

This lab was developed with assistance from AI tools. The instructor reviewed and revised the lab materials, datasets, code, and scoring workflow. If you notice unclear instructions, coding issues, unrealistic data behavior, or other rough edges, please report them so the lab can be improved.

## Student-Facing Rubric

| Category | Points | What strong work demonstrates |
|---|---|---|
| Security task framing | 15 | Clearly explains the IDS task and what the model output supports. |
| Data understanding | 15 | Identifies label balance, feature meaning, and synthetic-data limits. |
| Three-classifier comparison | 15 | Trains at least three classifier types and explains which performs best. |
| Train/validation, cross-validation, and competition-scoring preparation | 25 | Reports validation and cross-validation metrics, selects one best model, and prepares `best_model.ipynb` for private competition scoring. |
| Human review and limitations | 15 | Explains what an analyst still needs to inspect or decide. |
| Communication | 15 | Presents conclusions clearly and makes a defensible security recommendation. |