EE 467 — Lab 7 API Reference

Instruction Set Architecture (ISA) Identification of Program Binaries

Colab Setup (pip + dataset unpack)

Docs: `https://colab.research.google.com/`

- Use `%pip install numpy scipy scikit-learn termcolor`.
- Use `!tar -xJf binaries-dataset.tar.xz` to unpack the dataset.
- Ensure `binaries-dataset/` is in the working directory.

Base64 Decoding (`base64`)

Docs: `https://docs.python.org/3/library/base64.html`

- Use `base64.b64decode(str)` to convert Base64 strings into raw byte strings.
- Use `binary.hex()` to convert bytes into a hex string.
- For byte-level visualization, split hex strings into 2-character chunks.
- Use `set()` to remove duplicate binaries.

Label Encoding (`numpy`)

Docs: `https://numpy.org/doc/`

- Map string labels to integer indices using a dictionary: `{label: i for i, label in enumerate(...)}`.
- Convert mapped labels to `np.array` for scikit-learn compatibility.

Byte-Histogram + Endianness Features

- Create a 256-bin histogram for byte values (0–255).
- Count occurrences of endianness-indicating words: `b"\x00\x01"`, `b"\x01\x00"`, `b"\xff\xfe"`, `b"\xfe\xff"`.
- Use `np.concatenate` to combine histograms (total 260 dimensions).
- Normalize by binary length to ensure scale invariance.

TF-IDF Feature Extraction (`sklearn.feature_extraction.text`)

Docs: `https://scikit-learn.org/stable/modules/feature_extraction.html`

- Use `TfidfVectorizer(analyzer="char", ngram_range=(1,2), encoding="latin1")` for byte-level unigram + bigram features.
- Use `TfidfVectorizer(analyzer="char", ngram_range=(3,3), max_features=10000, encoding="latin1"` for trigram features.

- Use `scipy.sparse.hstack` to concatenate sparse matrices.
- For hex-level features, first convert binaries to hex strings, then apply `TfidfVectorizer`.
- No `latin1` override needed for hex representation.

Train/Test Split (`sklearn.model_selection`)

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

- Use `train_test_split(..., stratify=labels, random_state=...)`.
- Lab default: `train_size=0.2`, `test_size=0.05`.
- Keep `random_state` fixed for reproducibility.

Machine Learning Models

Linear SVM

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`

- Use `LinearSVC(max_iter=200, random_state=...)`.

Logistic Regression

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

- Use `LogisticRegression(max_iter=200, random_state=...)`.

Decision Tree

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

- Use `DecisionTreeClassifier(random_state=...)`.

Random Forest

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

- Use `RandomForestClassifier(random_state=...)`.
- Increase `n_estimators` for stronger performance (higher cost).

Dimensionality Reduction (`sklearn.decomposition`)

Docs: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html`

- Use `KernelPCA(n_components=300, kernel="rbf")`.
- Fit on a 20% stratified subset to reduce memory usage.

- Transform the full dataset using the fitted model.
- KernelPCA does not natively support sparse matrices; fit carefully.

Evaluation (`sklearn.metrics`)

Docs: `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics`

- Use `confusion_matrix(y_true, y_pred)`.
- Use `classification_report(y_true, y_pred)`.
- Report accuracy, precision, recall, and F1-score.
- Since the ISA dataset is balanced, accuracy is meaningful (unlike fraud detection).

Timing Utility

- Use a context manager (e.g., `timeit`) to measure training runtime.
- Helps compare feature extraction and model complexity trade-offs.