ML4Cyber — Lab 5 API Reference

Colab Setup (pip + dataset unpack)

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

- Use `%pip install numpy pandas scikit-learn matplotlib seaborn imbalanced-learn` to install required libraries.
- Use `!tar -xJf credit-card.tar.xz` to unpack the dataset archive in Colab.

Pandas (`pandas`)

Docs: `https://pandas.pydata.org/docs/`

- Use `pandas.read_csv(...)` (`https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html`) to load `creditcard.csv` into a DataFrame.
- Use `df.head(n)` (`https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html`) to preview the first rows.
- Use `df.describe(...)` (`https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html`) to get summary statistics (useful for percentiles).
- Use `df["col"].value_counts()` (`https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html`) to count fraud vs. non-fraud labels.
- Use `df.drop([...], axis=1)` (`https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html`) to remove `Class` from features.
- Use boolean filtering like `df[df["Class"]==0]` to split fraud vs. non-fraud subsets.

NumPy (`numpy`)

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

- Use `np.arange(start, stop, step)` (`https://numpy.org/doc/stable/reference/generated/numpy.arange.html`) to create histogram bin edges.
- Use `np.arange(0.1, 1, 0.1)` (`https://numpy.org/doc/stable/reference/generated/numpy.arange.html`) to create percentile points for `describe(percentiles=...)`.
- Use `arr.reshape(-1, 1)` (`https://numpy.org/doc/stable/reference/generated/numpy.reshape.html`) to make a 2D column vector for scikit-learn scalers.

Matplotlib (`matplotlib.pyplot`)

Docs: `https://matplotlib.org/stable/`

- Use `plt.hist(...)` (`https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html`) to plot class-conditional time distributions.

- Use plt.hist(..., density=True) to plot normalized distributions (fractions/density rather than raw counts).
- Use alpha=... in plt.hist to increase transparency when overlaying fraud vs. non-fraud.
- Use plt.xlabel(...) / plt.ylabel(...) (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlabel.html) to label axes.
- Use plt.title(...) (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.title.html) to set plot titles.
- Use plt.xlim((xmin, xmax)) (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xlim.html) to constrain plot ranges.
- Use plt.legend(...) (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.legend.html) to show series labels.
- Use plt.show() (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.show.html) to render the plot.

Seaborn (seaborn)

Docs: https://seaborn.pydata.org/api.html

- Use sns.histplot(...) (https://seaborn.pydata.org/generated/seaborn.histplot.html) to plot smooth distributions.
- Use kde=True in histplot to overlay a KDE curve.
- Use stat="density" in histplot to plot a density (normalized) distribution.

PCA Visualization (sklearn.decomposition)

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

- Use IncrementalPCA(n_components=3) (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html) to reduce V1-V28 to 3D.
- Use ipca.fit_transform(X) to compute the 3D embedding for plotting.
- If your lab provides helpers (e.g., lab_6_util.visualize_samples), use them to plot the 3D samples by class.

Feature Scaling (sklearn.preprocessing)

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

- Use StandardScaler() (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) to standardize Time after dividing by 24.
- Use RobustScaler() (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html) to scale Amount robustly under outliers.
- Use scaler.fit_transform(X) to learn scaling parameters on training data and apply the transform.

- Caution: scalers expect 2D input; use `df[["Time"]].values` or `arr.reshape(-1,1)` and assign back via `df["Time"] = ...`.

Train/Test Split (`sklearn.model_selection`)

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

- Use `train_test_split(X, y, test_size=..., random_state=...)` (`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`) to create training and test sets.
- Use a larger `test_size` when the positive class is very rare, so the test set contains enough fraud cases.

Imbalance Handling (`imblearn.over_sampling`)

Docs: `https://imbalanced-learn.readthedocs.io/en/stable/api.html#module-imblearn.over_sampling`

- Use `SMOTE(random_state=...)` (`https://imbalanced-learn.readthedocs.io/en/stable/references/generated/imblearn.over_sampling.SMOTE.html`) to over-sample the minority class by synthetic interpolation.
- Use `BorderlineSMOTE(random_state=...)` (`https://imbalanced-learn.readthedocs.io/en/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html`) to over-sample mainly near the decision boundary.

Pipelines (`imblearn.pipeline`)

Docs: `https://imbalanced-learn.readthedocs.io/en/stable/references/generated/imblearn.pipeline.Pipeline.html`

- Use `Pipeline(steps=[("smote", smote), ("dt", clf)])` to combine re-sampling + model training in one fit call.
- Use `pipeline.fit(X_train, y_train)` to apply SMOTE only on training data inside the pipeline.
- Use `pipeline.predict(X_test)` to run the fitted pipeline on the untouched test set.

Decision Trees (`sklearn.tree`)

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

- Use `DecisionTreeClassifier(...)` (`https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`) to create a decision tree classifier.
- Use `dt.fit(X_train, y_train)` to train on original (non-resampled) data.
- Use `dt.predict(X_test)` to produce predictions for evaluation.

Evaluation (`sklearn.metrics`)

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

- Use `classification_report(y_true, y_pred)` (`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`) to print precision/recall/F1 per class.
- Use `confusion_matrix(y_true, y_pred)` (`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`) to compute TP/FP/FN/TN counts.
- Use precision/recall/F1 (not just accuracy) when the dataset is highly imbalanced.

Common TODO Patterns in This Lab

Docs: `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html`

- Time-by-class histogram: use `plt.hist(time_nonfraud, bins, density=True, alpha=..., label="Non-Fraud")` and a second `plt.hist` for fraud with a different color/label.
- Time conversion: use `df["Time"]/(60*60)` to convert seconds to hours; use `time/24` to convert hour-of-day ratio before `StandardScaler`.
- Scaling columns: assign back to DataFrame columns after `fit_transform` (remember scalers return 2D arrays).
- SMOTE + DT training: prefer a `Pipeline` so over-sampling happens only on the training split.