# Checkpoint 1 Guide: Problem Definition, Data Gathering, KPIs

## Problem Definition

* **Clearly articulate the question you’re asking**
  + What decision or action will this analysis inform?
  + Who are the stakeholders and what do they care about?
* **Specify the unit of analysis**
  + Individual, transaction, session, experiment, etc.
* **Define the scope and boundaries**
  + Time horizon, geographic region, population, features included/excluded.
* **Identify anti-goals**
  + Explicitly state what your project will *not* address.

## Data Gathering

* **Source identification**
  + Public datasets (e.g. [Kaggle](https://www.kaggle.com/), [UCI ML Repository](https://archive.ics.uci.edu/)).
  + APIs or web scraping
    - [requests](https://docs.python-requests.org/en/latest/)
    - [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)
    - [scrapy](https://docs.scrapy.org/en/latest/)
  + Databases
    - [sqlite3](https://docs.python.org/3/library/sqlite3.html)
    - [SQLAlchemy](https://docs.sqlalchemy.org/en/20/)
    - [pymongo](https://pymongo.readthedocs.io/en/stable/)
* **Acquisition strategy**
  + One-time download vs. automated pipeline.
  + Handling rate limits or access restrictions.
* **Documentation of provenance**
  + Record URLs, API calls, database queries.
  + Save raw data before transformations.
* **Ethical and legal considerations**
  + Licensing, privacy concerns, sensitive data handling.

## Data Assessment

* **Volume and coverage**
  + Is there enough data to support modeling?
* **Granularity**
  + Does the level of detail match the unit of analysis?
* **Bias and representativeness**
  + Consider missing subpopulations and selection bias.

## Assessing Learnability

* **Signal vs. noise**
  + Do the features plausibly contain information about the target?
* **Data sufficiency**
  + Are there enough examples overall and per class?
  + For time series, do you have enough cycles to capture seasonality or trends?
* **Feature-target alignment**
  + Are features actually available at prediction time (avoid leakage)?
  + Do you have variables that could plausibly explain the target?
* **Back-of-the-envelope model test**
  + Quickly fit some models without investing too much work:
    - Include very basic cleaning and imputation in your pipelines (so that the models can actually run), but no feature selection or engineering.
    - Trivial Baselines
      * [DummyRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyRegressor.html)
      * [DummyClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html)
    - Linear Models
      * [LogisticRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
      * [LinearRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)
    - Tree Based Models
      * [RandomForestClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
      * [RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)
  + Always evaluate with **cross-validation**:
    - [KFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)
    - [cross\_val\_score](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html)
  + If performance is indistinguishable from trivial baselines across folds, the problem may not be learnable with the current data.
* **Domain sanity check**
  + Ask subject experts whether the target is realistically predictable given the inputs.

## KPI Definition (Key Performance Indicators)

* **Primary KPI**
  + What metric directly reflects project success? (e.g., RMSE, accuracy, F1, uplift).
* **Secondary KPIs**
  + Capture trade-offs: precision vs. recall, fairness metrics, latency, cost.
* **Baseline definition**
  + Use the same models as your Back-of-the-Envelope model test but record both primary and secondary KPIs now that you have defined them.

## Deliverables

* **README.md**: Contains the written problem statement and links to notebooks/scripts.
* **data\_inventory.csv or data\_inventory.md**: Tabular list of data sources, access methods, licensing, limitations. Should be generated or updated by scripts when possible.
* **Data acquisition scripts** in src/data/: One-off downloads, API calls, scraping scripts, or database query files. Each should log provenance (URLs, queries, timestamps).
* **Raw data snapshot** in data/raw/: A small immutable sample, or instructions for secure download if too large/sensitive.
* **Baseline modeling notebook** in notebooks/baseline.ipynb:
  + Implements trivial, linear, and tree-based baselines.
  + Evaluates with cross-validation.
  + Reports both primary and secondary KPIs in a table.
* **KPI definition file** (kpis.md): Explicit metrics, formulas, and improvement directions.
* **Environment specification** (environment.yml or requirements.txt): Reproducible package list including pandas, numpy, scikit-learn, and any acquisition libraries used.
* **(Optional) Provenance log** in logs/: Script-generated record of data pulls (timestamps, queries, file hashes, rate-limit notes).

Note: Treat these as suggestions rather than demands. Not all of these deliverables will make sense for all teams. For example, if your data is a single dataframe sourced from Kaggle you will not need to have data\_inventory.csv: you can just link to the data source in the project README.