Make Data Count (MDC): Finding Data References — Viva Cheat Sheet

Use this as your script. Keep answers short and confident. Wherever you see “(show example)”, scroll to that snippet in your notebook.

---

1) Competition overview

Q: What competition did you select and what’s the goal?
A: *Make Data Count – Finding Data References* on Kaggle. Given scientific articles (XML) and possible dataset identifiers, the goal is to predict whether the mention is **Primary**, **Secondary**, or **Missing**.

Q: Why is this useful?
A: It helps track dataset reuse and impact by automatically finding dataset mentions in papers.

Q: What exactly is a “mention”?
A: Any DOI or accession-like string in the article text that likely points to a dataset (e.g., `10.xxxx/...`, `GSE1234`).

---

2) Data understanding

Q: What’s the dataset? Where does it come from?
A: The competition provides: - A **train_labels.csv** file with columns like `article_id`, `dataset_id`, and `type` (label). - XML files with full text for each `article_id` (train/test folders).

Q: What’s the difference between primary and secondary?
A: - **Primary**: The paper **generates** or **releases** the dataset. - **Secondary**: The paper **uses** or **cites** a dataset created elsewhere. - **Missing**: A candidate string that isn’t a clear dataset reference or gives no usable context.

Q: Primary vs Secondary quick cues?
A: - Primary cues: *“we collected”, “we generated”, “this dataset”*
- Secondary cues: *“as reported”, “based on”, “according to”*

(These cues show up in your n-gram EDA and help your intuition.)

---

3) Your pipeline (end-to-end)

Q: How does your system work, step by step?
A: 1) **Load labels** (`train_labels.csv`).
2) **Parse XML** for each article and **locate mentions** (regex for DOIs and accessions).
3) **Extract context** (surrounding sentence) and **clean** (drop "Missing", filter too-short lines).
4) **Vectorize** text with **TF-IDF** (uni- & bi-grams).
5) **Train** a **Logistic Regression** classifier (multiclass).
6) **Predict** on test XMLs and **build submission.csv**.

Q: What did you actually implement?
A: A robust **baseline**: TF-IDF + Logistic Regression with `class_weight='balanced'` to handle class imbalance. It's fast, reproducible, and easy to explain.

Q: Why TF-IDF first, not BERT?
A: Time and simplicity. TF-IDF + Logistic Regression trains quickly and gives a solid baseline. Transformer fine-tuning can be added later if time permits.

---

4) Preprocessing

Q: How do you find dataset mentions inside XML?
A: Regex patterns: - DOI pattern: `10.\d{4,9}/...` - Accession patterns: `GSE\d+`, `E-*-\d+`, `PRJ*\d+`, `CHEMBL\d+`, `PDB <id>`

Q: How do you create training rows?
A: For each labeled paper (`article_id`) and its `dataset_id`, scan the XML; when the cleaned DOI appears in text, save the sentence as a **context** with the known label. Then **drop** contexts with < 5 tokens.

Q: Why drop short contexts?
A: Very short strings (e.g., a bare DOI) don't carry semantics and hurt the classifier.

---

5) Features & model

Q: What features do you use?
A: TF-IDF with unigrams and bigrams; English stop-words removed; up to 3000 features.

Q: What model do you use and why?
A: Logistic Regression (multinomial). Reasons: strong baseline for high-dimensional sparse text; interpretable; fast to train; works well with TF-IDF.

Q: How did you handle class imbalance?
A: `class_weight='balanced'` during training so minority classes (Primary) get more weight.

Q: Train/validation split?
A: 80/20 split with fixed random seed for repeatability.

Q: Planned deep learning?
A: If time allowed: fine-tune **BERT/SciBERT** on the same contexts; expected improvement on subtle Primary vs Secondary cues.

---

6) Evaluation & results

Q: Which metrics and why?
A: Macro Precision/Recall/F1—treats all classes equally, important under imbalance.

Q: Baseline performance (macro numbers)?
A: Precision ≈ **0.81**, Recall ≈ **0.78**, F1 ≈ **0.79** (illustrative from your report). Use the confusion matrix story: *Missing* easiest; *Primary* hardest.

Q: Any typical errors?
A: Secondary sometimes mis-as-Missing when context is too short/ambiguous; Primary under-detected due to rarity and subtle wording.

---

7) Inference & submission

Q: How do you predict on test?
A: Iterate test XMLs → extract all DOI/accession mentions → vectorize mention text → predict class → deduplicate rows → save **submission.csv**.

Q: What's inside submission.csv?
A: Columns: `row_id`, `article_id`, `dataset_id`, `type`.

---

8) Reproducibility

Q: How can someone reproduce your results?
A: Run the notebook in Kaggle with the competition dataset mounted. The script loads paths under `/kaggle/input/...`, builds features, trains, and writes `submission.csv`.

---

9) Limitations & improvements (say these proactively)

- **Imbalance** (Primary rare) → handled with class weights; could also try focal loss or resampling.
- **Short contexts** → expand window around matches; sentence splitter or XML section-aware extraction.

- **Rule coverage** → better regex + a learned **NER** for dataset spans.
- **Model capacity** → upgrade to **BERT/SciBERT** fine-tuning; try class-balanced sampling; ensembles.

---

10) "Show-and-tell" snippets (where to point in your notebook)

When asked "show me", scroll to these parts: - **Loading labels** and pointing to `train_labels.csv` path.
- **Regex patterns** for DOI and accessions.
- **Context extraction** and **<5 tokens** filter.
- **TF-IDF (1–2 grams, 3000 features)** settings.
- **LogisticRegression** with `class_weight='balanced'`.
- **Train/test split (80/20)**.
- **Loop over test/XML → submission.csv**.

Use these lines to narrate exactly what happens.

---

11) 60-second elevator pitch

"We built a baseline NLP pipeline for the Make Data Count citation task. It parses XMLs, regex-extracts dataset mentions, forms sentence-level contexts, and learns a TF-IDF + Logistic Regression classifier with class-balancing. We evaluate with macro F1 due to imbalance and see that 'Missing' is easiest and 'Primary' is hardest. The pipeline is simple, fast, and fully reproducible on Kaggle. With more time, we'd fine-tune a BERT-family model (e.g., SciBERT) and widen context windows, which should improve Primary vs Secondary disambiguation."

---

12) Likely Viva questions (with short answers)

- **What's the task?** Multi-class text classification: Primary/Secondary/Missing dataset mentions.
- **Primary vs Secondary?** Primary = dataset creators; Secondary = dataset users/citers.
- **Data sources?** Kaggle competition: labels CSV + article XMLs.
- **Preprocessing?** XML parse → regex mentions → context sentence → clean.
- **Features?** TF-IDF unigrams/bigrams, 3k max features, English stop-words.
- **Model?** Logistic Regression (multinomial), class-balanced.
- **Split?** 80/20 with fixed seed.
- **Metrics?** Macro P/R/F1.
- **Imbalance handling?** Class weights (and optionally resampling).
- **Submission format?** row_id, article_id, dataset_id, type.
- **Weak spots?** Short contexts and rare Primary.
- **How to improve?** Larger context window, NER, SciBERT/BERT fine-tune, ensembles.

---

13) Defence tips

- Keep terms simple; define jargon (TF-IDF, primary/secondary) in one line.
- When stuck, *show the code cell* that does it and read it aloud.
- Tie every answer to your pipeline steps; don't drift into unrelated theory.
- End answers with a forward-looking improvement (e.g., "with SciBERT we can…").

Good luck—you've got this. 👊