Commit Messages → Developer Role Classification

Context:

A single commit leaves a small footprint — words, counts, and tokens. These footprints, taken together, betray the responsibility the committer held on the project. Your task is to read a commit's footprint and assign the most plausible developer role (e.g., frontend, backend, fullstack, qa/test).

Objective:

Design and deliver a reproducible project that, given a commit record, predicts the developer role. Emphasize careful reasoning and transparent choices rather than only chasing a high number. Submit the following artifacts in a public repository:

- Preprocessing notebook — reproducible, well-documented code showing how raw records become model-ready examples.
- Exploratory analysis notebook — visual and written observations that reveal dataset properties and risk factors.
- Modeling notebook — at least one baseline and one improved experiment, with training/validation protocol, hyperparameter notes, and logged results.
- Evaluation report — concise PDF (≤ 2 pages) summarizing final evaluation, major failure modes, and lessons learned.
- Reflection — short (≤ 300 words) written justification of your most important design decisions.
- Annotated examples — ten commits from different roles annotated in plain language explaining the label assignment.
- A README explaining how to reproduce results end-to-end.

Dataset:

Each row corresponds to a single commit and contains:

- index — id
- role — target label (frontend, backend, fullstack, qa/test, …)
- committype — categorical (feature, bugfix, refactor, …)
- fileextensions — a compact token list (e.g., ['js_ts','py','html'])
- numfileschanged, linesadded, linesdeleted, numcommentsadded — numeric signals
- timeofcommit — human-readable weekday/time token
- commitmessage — raw English text

Constraints & Expectations:

Prioritize reasoning over recipes. The graders value arguments and evidence more than following a particular technical recipe. Explain why you applied each transformation or test.

- Preprocessing is part of modeling. Choices made before model training must be justified.
- Exploration should surface issues that affect modeling (e.g., label imbalance, noisy data).
- Design defensively to prevent data leakage and spurious correlations.
- Report multiple perspectives: per-class performance, confusion patterns, qualitative analysis.
- Explainability matters: provide a human-interpretable explanation of model behavior.
- Reproducibility is mandatory: use fixed seeds, environment specs, and runnable notebooks.

Evaluation:

- Primary metric: macro F1.
- Additional: per-class precision & recall, confusion matrix, calibration/robustness comments.
- Provide evidence (tables/plots) for claims about performance and failure modes.

Rubric:

- Data & EDA (20%) — clarity of cleaning, identification of issues, and how analyses informed design.
- Preprocessing & Features (25%) — justified choices and their experimental impact.
- Modeling & Validation (25%) — baseline, experimentation quality, validation rigor, and honest reporting.
- Interpretability & Error Analysis (15%) — meaningful explanations and corrective thinking.
- Reproducibility & Presentation (15%) — runnable artifacts, clear README, concise reporting.