COVID-19 Mortality Prediction

Domain Background

COVID-19 strained emergency and inpatient services worldwide, making rapid, objective triage essential for allocating scarce resources such as telemetry beds and ICU capacity. Early mortality risk prediction at the first point of assessment can focus clinician attention on the most vulnerable patients, complementing clinical judgment when time and information are limited. Reputable clinical guidance emphasizes early risk stratification and notes substantially higher mortality among older adults and patients with pre-existing conditions (e.g., diabetes, cardiovascular disease, chronic lung disease, immunosuppression, and obesity). In this educational project, I study a data-driven mortality risk model, using a public, de-identified dataset to explore how tabular machine-learning methods might surface patterns useful for triage research—**without** making clinical claims or handling PHI.

Personal motivation. In early 2021, I spent three months hospitalized with COVID-19, including an ICU stay with intubation. That experience motivates me to explore careful, ethical use of data science to characterize mortality risk, strictly for learning and discussion—not for clinical deployment.

Problem Statement

Task. Supervised *binary classification* to predict **mortality** for patients presenting with COVID-19 symptoms.

Unit of analysis. One patient encounter (one row).

Inputs. Structured tabular features available at or before initial presentation (demographics and pre-existing conditions; presentation/symptom dates; and other pre-admission flags provided by the dataset).

Output. A predicted probability of death plus a class label given a decision threshold. The training label is provided via a configurable pipeline parameter **`LabelColumn`** (default `"label"`), with the project using `"mortality"` at runtime (Assumption) after deriving the label from the dataset's outcome fields (TBD confirmation on exact derivation in the notebook/transform script).

Success criteria (educational). On a held-out test set, improve **PR-AUC** over a logistic-regression baseline by ≥0.05 (Assumption) and improve calibration (Brier score). Optimize recall at a fixed minimum precision (e.g., ≥0.50). Intended for **non-clinical,**

educational use only; predictions are assumed to be made at the initial triage time point, with any post-escalation features excluded.

Datasets and Inputs

Source & access. Kaggle — *COVID-19 Patient Precondition Dataset* by **tanmoyx**:

https://www.kaggle.com/datasets/tanmoyx/covid19-patient-precondition-dataset

I will download locally via Kaggle tools, run EDA and label construction in the notebook, and then upload a CSV to S3 for automation.

Size & schema. Exact rows/columns and any dataset-level caveats not explicitly listed on the Kaggle page are **TBD** (to be confirmed during notebook EDA). Feature groups are expected to include: demographics (e.g., age, sex, pregnancy), comorbidities (e.g., diabetes, hypertension, obesity, renal disease, cardiovascular disease, chronic lung conditions), tobacco use, presentation/symptom dates, and outcome fields.

Target definition. TBD confirmation in notebook. (Common practice with Mexico's surveillance data is to derive mortality from an outcome/date field; this project will implement the label derivation reproducibly in code.)

Missingness & imbalance. The notebook will quantify missingness and class imbalance (expected minority "death" class). Exact percentages are **TBD** until computed. The pipeline's feature-engineering step is parameterized (e.g., `--elderly-age 65`, `--delay-threshold-days 2`) and outputs **`engineered.csv`** with headers; the preprocess step then splits and formats data for JumpStart (label first, no header, train/val/eval outputs).

Train/validation/test. In the notebook, I will create **stratified splits** (e.g., 70/15/15) and preserve a final **held-out test** set. In the pipeline, validation fraction is parameterized (default **`ValFraction=0.2`**) for internal model selection/evaluation; an `eval` artifact is also emitted for offline assessment.

Appropriateness & limitations. A large, real-world surveillance dataset is appropriate for educational triage modeling; however, representativeness (country/time-period), coding conventions, and temporal drift may limit generalizability. All results will be framed as **educational**.

Solution Statement

Local development (notebook). The notebook performs EDA, leakage checks, feature engineering, baseline modeling, and result visualization. Outputs are versioned artifacts used by the pipeline (label mapping, feature lists, split manifests). *(Concrete numbers remain TBD until runs complete.)*

Automation & productization (SageMaker Pipelines + AutoGluon Tabular).

Explainability & fairness. I will compute SHAP summaries and subgroup metrics (sex/age) in the notebook; calibration will be applied post-hoc if needed.
Reproducibility & cost. Seeds fixed where supported, pinned environments, versioned S3 paths, **Experiments** (run tracking) as appropriate, and controlled time limits/instance sizes.

Benchmark Model

Two baselines anchor performance and interpretability:

- **Majority-class baseline** ("always survive"), establishing a lower bound for PR-AUC/ROC-AUC and a calibration reference.
- **Regularized Logistic Regression** with class weighting, trained only on triage-time features (e.g., age, sex, key comorbidities, symptom-to-presentation interval).

Both baselines will use the **same stratified splits** and the **same metric suite** as the AutoGluon model to ensure apples-to-apples comparison. Logistic regression is a common, interpretable first-order model for tabular clinical data and therefore a defensible benchmark.

Evaluation Metrics

Because mortality is typically a **minority** class, **PR-AUC** (average precision) is the **primary** metric; it best reflects utility when false negatives are costly. **ROC-AUC** is reported for completeness. I will compute **Recall (Sensitivity)**, **Specificity**, **Precision**, **F1-score**, and the **Confusion Matrix** at (i) the threshold maximizing F1 and (ii) a recall-oriented threshold meeting a fixed minimum precision (e.g., ≥0.50). **Calibration**: **Brier score** plus reliability curves, with isotonic/Platt post-hoc calibration if needed. I will report the **class distribution** and **95% CIs** (bootstrap) for key metrics on the held-out test set.

Pipeline gate. The current pipeline registers a model only if **accuracy ≥** `AccuracyThreshold` **(default 0.80)**. This is acceptable for packaging but, given class

imbalance, final acceptance decisions will consider PR-AUC and recall; a follow-on change will add PR-AUC to `evaluation.json` and/or gate jointly on PR-AUC and accuracy.

Project Design

Step-by-step workflow

1. **EDA (notebook):** schema checks; value ranges; missingness; label derivation; outcome prevalence; leakage audit (drop post-escalation fields like intubation/ICU if present).

2. **Feature processing (notebook → pipeline):** encode binaries/categoricals, safe date parsing, engineered intervals (e.g., symptom→entry), threshold flags (e.g., `elderly_age=65`, `delay_threshold_days=2`).

3. **Splits:** stratified train/val/test in the notebook (70/15/15); the pipeline uses **`ValFraction=0.2`** to produce train/val/eval S3 prefixes.

4. **Baselines:** majority + logistic regression; record PR-AUC/ROC-AUC/recall and calibration.

5. **AutoGluon HPO/ensembling:** enable **`auto_stack=True`**, `presets="medium_quality_faster_train"`, **`time_limit=900s`** (tunable).

6. **Validation & calibration:** reliability curves; apply isotonic/Platt if needed.

7. **Fairness checks:** performance by **sex** and **age** strata (if available).

8. **Error analysis:** inspect high-probability false negatives/positives; SHAP summaries for top features.

9. **Packaging & registry:** pipeline registers models to **Model Package Group "AutogluonTabular"** if the accuracy gate passes; manual approval remains required.

10. **Documentation:** code repo with `/notebooks`, `/scripts` (`transform.py`, `preprocess.py`, `evaluate.py`), `/pipelines`, and `/reports`.

Reproducibility & cost controls. Pinned packages; seeded runs; versioned S3 paths; 30-day step caching; right-sized instances (`ml.m5.2xlarge` processing, `ml.m5.4xlarge` training by default).

Risks & mitigations.

- **Data quality/imbalance:** explicit handling of unknown/“unspecified” values; class weighting; stratified CV.
- **Overfitting:** bagging/stacking with CV; monitor CV–test gaps.
- **Shift/generalizability:** clearly label dataset scope (time/region); no clinical claims.
- **Fairness:** subgroup reporting; threshold adjustments if needed.

Timeline & deliverables.

- **Week 1:** EDA, label derivation, leakage audit, feature pipeline stubs.
- **Week 2:** Baselines; first AutoGluon runs; interim report.
- **Week 3:** Calibration, fairness/error analysis; finalize metrics.
- **Week 4:** Pipeline polish, model registration, write-up, reproducibility checklist.
  Deliverables: Git repo (code + notebooks), pipeline definition, metrics report (PDF), calibration/SHAP plots, registered model package.