Corporate Financial Distress Early - Warning System

Project Group 22:

- Bryan Zeng
- Harshit Raheja
- Sankalp Hegde
- Piriyajeishree Murali Naidu
- Nandana Pradeep
- Palak Tanwar

1. Introduction

Corporate financial distress, such as bankruptcy, prolonged liquidity shortages, or involuntary delisting, poses significant risks to investors, lenders, and regulators. In many cases, warning signs exist in financial statements and macroeconomic conditions well before a firm officially fails. However, these signals are often difficult to interpret, dispersed across multiple reports, or identified only after severe financial deterioration becomes public.

Traditional approaches to distress detection rely on static financial ratios (e.g., Altman Z-score), periodic analyst reviews, or external credit ratings. While useful, these methods are largely **reactive**, infrequently updated, and limited in their ability to capture evolving financial trends or macroeconomic shocks. As a result, stakeholders often receive actionable signals too late to mitigate losses or adjust risk exposure.

This project proposes a **Corporate Financial Distress Early-Warning System** that applies machine learning techniques to publicly available financial and macroeconomic data in order to estimate the probability of corporate distress **6–12 months in advance**. The system leverages structured financial statements from SEC EDGAR filings, macroeconomic indicators from the Federal Reserve Economic Data (FRED) repository, and optional market and textual signals to generate forward-looking, interpretable risk scores.

Rather than replacing professional credit analysis, the system is designed as a **decision-support and educational analytics tool**, demonstrating how modern data pipelines and ML models can be applied to financial risk monitoring at scale. Emphasis is placed on **explainability**, allowing users to understand which financial or macroeconomic factors contribute most to elevated risk levels.

From an academic perspective, the project also serves as a case study in building end-to-end machine learning systems for structured, time-dependent data. Beyond model development, the work emphasizes data ingestion, feature engineering, versioned datasets, reproducible training pipelines, and monitoring considerations, key components of real-world ML systems. This makes the project well-suited for demonstrating applied machine learning, data engineering, and MLOps principles in a controlled, reproducible setting.

Key characteristics of the proposed system include:

- **Predictive orientation:** Identifies early warning signals before public distress events

- **Interpretability:** Highlights key drivers behind risk predictions rather than relying on black-box outputs

- **Data-driven modeling:** Integrates financial ratios, temporal trends, and macroeconomic context

- **Academic feasibility:** Built entirely on public, well-documented data sources suitable for reproducible student research

This project aligns closely with course objectives in machine learning systems, data engineering, and MLOps, while addressing a real-world financial analytics problem.

2. Dataset Information

2.1 Dataset Introduction

The project relies exclusively on **publicly accessible financial and economic datasets**, ensuring transparency, reproducibility, and suitability for academic research.

Primary Dataset: SEC EDGAR Financial Filings

- Filing Types:
    - **10-K (Annual Reports)**
    - **10-Q (Quarterly Reports)**
- Data Format:
    - Structured **XBRL financial statements** accessed via the SEC EDGAR API
- Contents:
    - Income statements
    - Balance sheets
    - Cash flow statements
    - Selected textual disclosures and footnotes

SEC EDGAR filings provide standardized, longitudinal financial records for publicly traded U.S. companies, enabling temporal analysis of firm-level financial health over multiple years.

The structured nature of XBRL-tagged filings enables consistent extraction of financial variables across firms and time, reducing manual preprocessing complexity. The longitudinal coverage supports temporal modeling techniques, allowing the system to capture deteriorating financial trends rather than relying solely on point-in-time indicators.

Supplementary Datasets:

- **Macroeconomic Indicators (FRED):**
    - Federal Funds Rate
    - Inflation (CPI)
    - Credit spreads
    - Unemployment and growth indicators
      These variables capture broader economic conditions that influence corporate solvency.
- **Distress Event Labels (Public Sources):**
    - Bankruptcy filings (e.g., UCLA LoPucki Bankruptcy Database, Kaggle datasets)
    - Exchange delisting records from public NASDAQ/NYSE announcements

  For academic feasibility, distress events are defined using observable and verifiable public outcomes rather than proprietary credit ratings. These include bankruptcy filings, exchange delistings due to financial non-compliance, and prolonged trading suspensions. This definition ensures labels are objective, reproducible, and obtainable without reliance on restricted data sources.

- **Optional Market Data (Enhancement):**
    - Stock prices, returns, and volatility
    - Trading volume indicators
      Sourced via Yahoo Finance (`yfinance`) or Alpha Vantage APIs.

Purpose of the Dataset:
The combined dataset supports supervised learning models that analyze financial ratios, temporal trends, and macroeconomic signals to identify firms exhibiting trajectories consistent with elevated distress risk.

To maintain scope discipline, textual disclosures and market data are treated as optional enhancements rather than core dependencies. The baseline system relies exclusively on structured financial and macroeconomic variables, ensuring that a complete and functional pipeline can be built even without advanced NLP components.

2.2 Data Card

Attribute	Description
Dataset Size	~10,000–20,000 U.S. public companies with quarterly observations over multiple years
Format	XBRL / JSON (SEC EDGAR), CSV (FRED, labels, market data)
Data Types	Numeric (financial metrics), categorical (sector, industry), limited text (filing disclosures)
Frequency	Quarterly (10-Q) and Annual (10-K)
Features	40–80 engineered features, including financial ratios, growth rates, rolling trends, and volatility measures
Target Variable	Binary label: DistressNext12Months
Missing Values	Present due to reporting gaps; handled via systematic imputation and validation

2.3 Data Sources

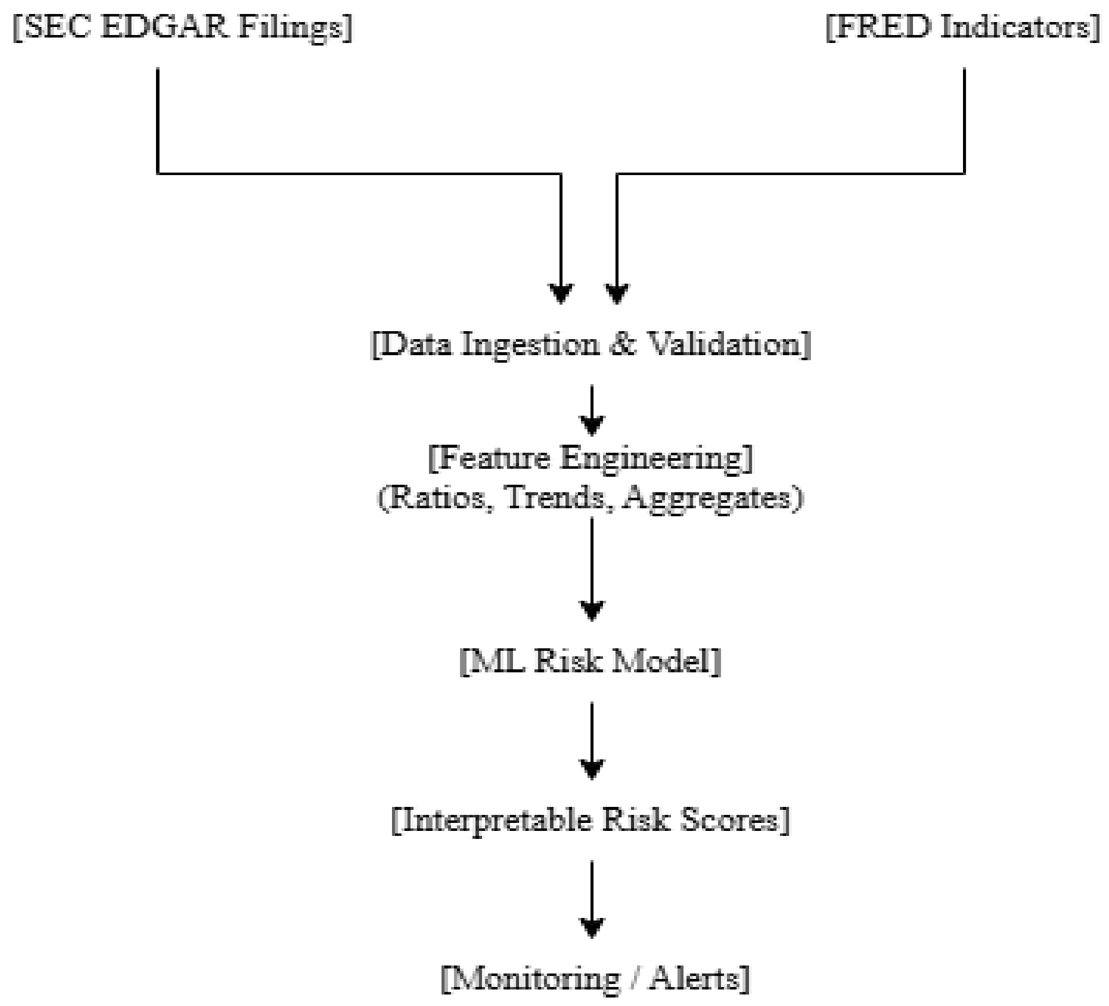
- **SEC EDGAR & XBRL API:** https://www.sec.gov/edgar
- **Federal Reserve Economic Data (FRED):** https://fred.stlouisfed.org
- **Bankruptcy & Delisting Data:**
  - UCLA LoPucki Bankruptcy Database
  - Public Kaggle financial distress datasets
- **Optional Market Data:**
  - Yahoo Finance (yfinance)
  - Alpha Vantage

All datasets are publicly available and appropriate for non-commercial academic use.

2.4 Data Rights and Privacy

- All data is **corporate-level and public**, with **no personally identifiable information (PII)**
- The project complies with **GDPR and CCPA** principles by design
- No confidential, proprietary, or customer-level data is used
- Textual disclosures are limited to publicly released regulatory filings

2.5 High-Level Data and Modeling Pipeline



3. Data Planning and Splits

All preprocessing, feature engineering, and splitting decisions are designed to reflect real-world deployment constraints and prevent information leakage.

Data Preprocessing Steps:

3.1 Data Loading

- Download 10-K and 10-Q filings from SEC EDGAR API using company CIK identifiers, implementing rate limiting to comply with SEC guidelines
- Cache raw JSON responses locally to avoid repeated API calls and improve pipeline efficiency
- Pull quarterly macroeconomic indicators from the FRED API, including Federal Funds Rate, credit spreads, and inflation rates
- Source bankruptcy labels from UCLA-LoPucki Bankruptcy Database and publicly available Kaggle datasets, matching to companies using CIK or ticker symbols

3.2 Cleaning & Imputation

- Apply forward-fill imputation for missing financial ratios with a maximum one-quarter gap, falling back to sector median for longer gaps
- Winsorize extreme values at 1st and 99th percentiles to handle outliers while preserving relative ordering
- Validate fundamental accounting identity (Assets = Liabilities + Equity) to catch corrupted filings
- Remove duplicate filings by retaining only the most recent version for each fiscal period
- Standardize numeric features using z-score normalization computed only on training data to prevent leakage

3.3 Feature Engineering

- Calculate core financial ratios, including Current Ratio, Debt-to-Equity, Interest Coverage, Net Margin, ROA, and Operating Cash Flow to Total Debt
- Compute temporal features, including quarter-over-quarter growth rates for revenue and cash flow, plus rolling 4-quarter slopes to capture trends
- Generate volatility features using the rolling standard deviation of net income and cash flow over 4 quarters
- Create distress indicator features, such as consecutive quarters of negative cash flow and debt acceleration rate
- Optionally extract NLP embeddings from the 10-K Risk Factors section using FinBERT for exploratory text-based risk signals, treated as a non-blocking enhancement.

3.4 Data Splitting

- Use strict time-based splitting to prevent data leakage and simulate real deployment conditions
- Training set covers 2010-2019 (approximately 60% of data), spanning a full economic cycle
- Validation set covers 2020-2021 (approximately 20% of data), including the COVID period, to test robustness
- Test set covers 2022-2023 (approximately 20% of data) for final unbiased evaluation
- Stratify by sector to ensure balanced industry representation across all splits

3.5 Label Assignment

- Define binary label DistressNext12Months as 1 if the company experiences bankruptcy filing, credit downgrade to non-investment-grade, where publicly available records exist, or financial delisting within 12 months
- Use a 3-month grace period to exclude imminent distress events that cannot be predicted early enough for action
- Address class imbalance (expected 2-5% distress rate) using class weights in model training and SMOTE oversampling on training data only, applied after temporal splitting to avoid leakage
- Focus evaluation on imbalance-appropriate metrics, including ROC-AUC, Precision@K, and Recall@K, rather than accuracy

4. GitHub Repository

Repository Link: https://github.com/sankalphegde/Foresight-ML

Folder Structure:

```
Foresight-ML/
│
├── .github/              # CI/CD workflows & issue templates
│   ├── workflows/          # Automated testing & deployment pipelines
│   └── ISSUE_TEMPLATE/       # Templates for bug reports
│
├── configs/              # Configuration management
│   ├── model/              # Hyperparameters for XGBoost/LSTM models
│   └── hydra/             # Advanced experiment configuration
│
├── data/                # Data storage (Managed via DVC)
```

```
│   ├── raw/                # Immutable raw data (SEC 10-K filings)
│   ├── processed/          # Cleaned financial ratios & market data
│   ├── features/           # Engineered features (Liquidity, Solvency)
│   └── external/           # Macroeconomic indicators (GDP, Interest Rates)
│
├── docs/                   # Project documentation
│   ├── architecture/       # System diagrams & design docs
│   ├── api_reference/      # Auto-generated API docs
│   └── runbooks/           # Deployment & failure recovery guides
│
├── infra/                  # GCP-native deployment & automation
│   ├── docker/             # Dockerfiles for Cloud Run services and jobs
│   ├── cloud_run/          # Cloud Run service & job configuration (YAML)
│   ├── scheduler/          # Cloud Scheduler job definitions (cron triggers)
│   └── build/              # Cloud Build CI/CD pipelines
│
├── monitoring/             # System observability
│   ├── data_drift.py       # Checks for shifts in financial data distribution
│   └── metrics.py          # Custom business KPIs (e.g., Recall/Precision)
│
├── notebooks/              # Experimentation & EDA
│
├── scripts/                # Utility automation
│   ├── setup_env.sh        # Environment initialization
│   └── download_data.sh    # EDGAR API data fetch triggers
│
├── src/                    # Main application source code
│   ├── feature_store/      # Feature definitions (Feast)
│   ├── ingestion/          # Data collection modules
│   ├── models/             # Training & inference logic
│   ├── api/                # REST API endpoints (FastAPI)
│   └── utils/              # Shared helper functions
```

```
│
├── tests/                # Automated testing suite
│   ├── unit/             # Tests for individual functions
│   └── integration/      # Pipeline end-to-end tests
│
├── .pre-commit-config.yaml   # Code quality checks
├── dvc.yaml              # Data pipeline definition
├── Makefile              # Build command shortcuts
├── pyproject.toml        # Python project configuration
├── README.md             # Project overview & setup
└── requirements.txt      # Production dependencies
```

Disclaimer: hypothetical layout for planning; actual contents may vary.

README: Includes project description, dataset sources, setup instructions, example usage, and output interpretation.

5. Project Scope

5.1 Problems

The current landscape of corporate financial health monitoring suffers from significant inefficiencies that lead to "surprise" bankruptcies and delayed interventions. The core problems include:

- **Latency in Detection:** Financial distress is often identified only after official quarterly reports (10-Q/10-K) are released and manually reviewed. By the time a report is published and analyzed, the company may have already been in distress for months.
- **Static & Outdated Thresholds:** Traditional methods rely on rigid threshold-based rules (e.g., "If Debt-to-Equity > 2.0, flag as risk"). These static rules fail to adapt to changing macroeconomic conditions (e.g., high vs. low-interest rate environments) or industry-specific nuances.
- **Data Volume Overload:** Financial analysts cannot manually monitor the thousands of public companies continuously. This leads to a coverage gap where only large-cap companies are scrutinized, while small-to-mid-cap companies often fail without early warning.
- **Lack of Reproducibility:** In many firms, distress prediction is an ad-hoc process performed in disjointed Excel spreadsheets or local notebooks, making it impossible to audit, reproduce, or scale predictions across the market.

5.2 Current Solutions

Existing approaches to solving these problems are largely reactive or manual:

- **Altman Z-Score (1968):** A linear discriminant analysis formula that uses five financial ratios to predict bankruptcy. While a standard benchmark, it is a linear model that fails to capture complex, non-linear relationships in modern financial data.
- **Credit Rating Agencies (Moody's, S&P):** These provide high-quality assessments but are slow to update. Ratings are often "sticky" and may not reflect rapid deterioration in a company's health until a downgrade is officially announced.
- **Manual Analyst Review:** Investment banks employ teams to analyze financial statements. This "human-in-the-loop" approach is accurate but unscalable and subject to cognitive bias and fatigue.
- **Basic Machine Learning Models:** Some firms use random forests or logistic regression, but these are often deployed as static "pickle" files that are rarely retrained, leading to "model drift" as market dynamics shift.

5.3 Proposed Solutions

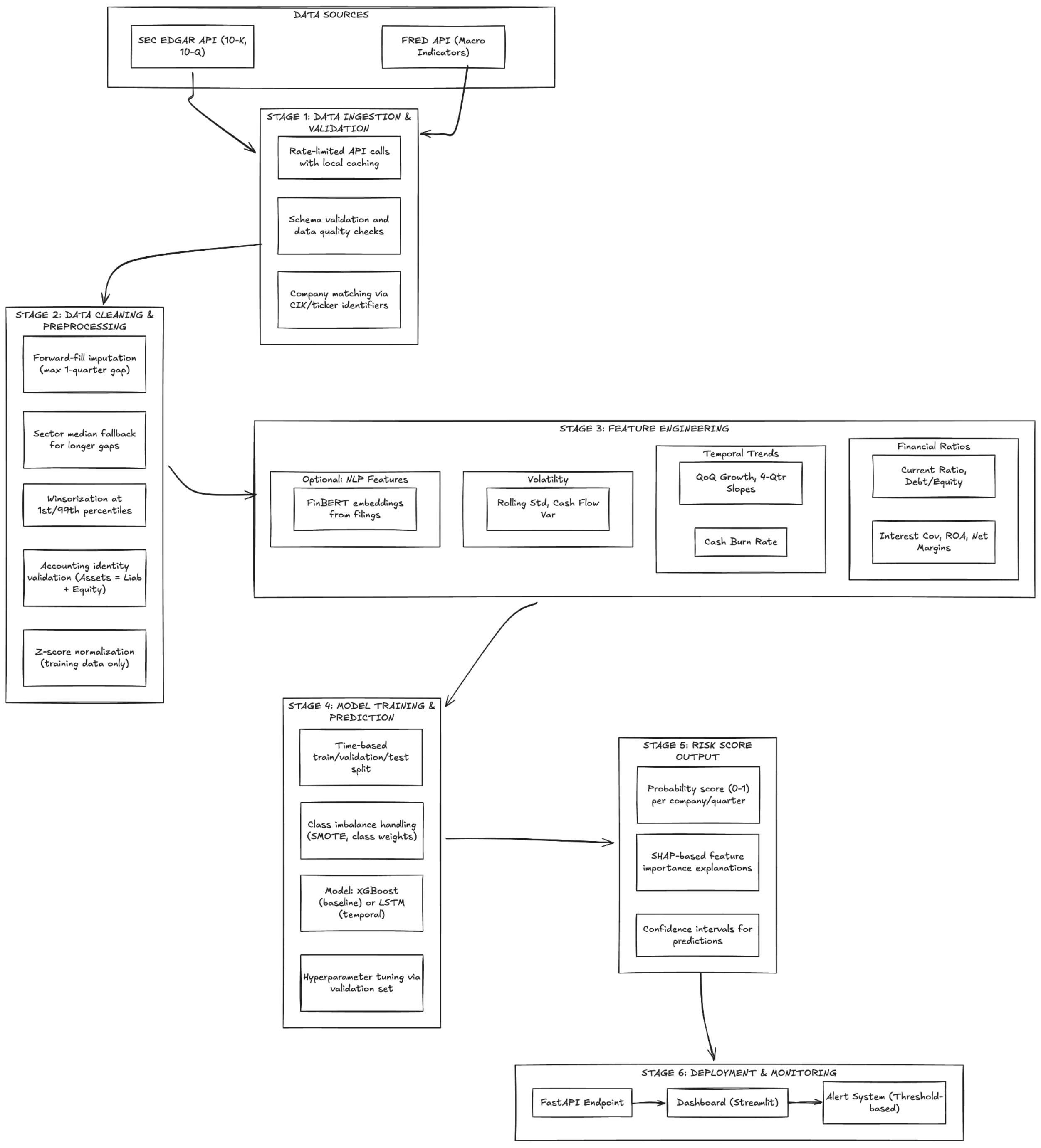
Our project proposes an end-to-end MLOps-driven Early Warning System that automates the lifecycle of distress prediction.

- **Automated Ingestion Pipeline:** Instead of manual downloads, we will build a data pipeline that automatically ingests financial data (SEC filings, market indicators) as soon as it becomes available via APIs, triggering immediate analysis.
- **Advanced Non-Linear Modeling:** We will utilize Gradient Boosting Machines (XGBoost/LightGBM) or Long Short-Term Memory (LSTM) networks to capture non-linear interactions between liquidity, leverage, and operational efficiency ratios over time.
- **Continuous Training (CT) & Monitoring:** Unlike static solutions, our system will include a monitoring layer to detect Data Drift (e.g., shifts in average market leverage) and Concept Drift (e.g., if safe ratios become risky during a recession). Significant drift will trigger automated retraining pipelines to keep the model current.
- **Scalable Inference Service:** The model will be deployed as a containerized REST API (FastAPI + Docker), allowing users to query the distress probability of any company in near real-time, providing an instant "risk score" and explaining the key contributing factors (e.g., "High Risk due to rapid decline in Quick Ratio").

In essence, this project shifts financial distress prediction from a static, reactive manual process to a dynamic, automated MLOps workflow. By integrating near real-time data ingestion with self-correcting machine learning models, we aim to provide a "live" health score for corporate entities that adapts to market volatility instantly, offering creditors and investors an early warning signal that traditional quarterly reviews simply cannot match.

6. Current Approach Flow Chart and Bottleneck Detection

6.1 Current Flow:


DATA SOURCES
SEC EDGAR API (10-K, 10-Q)
FRED API (Macro Indicators)
STAGE 1: DATA INGESTION & VALIDATION
Rate-limited API calls with local caching
Schema validation and data quality checks
Company matching via CIK/ticker identifiers
STAGE 2: DATA CLEANING & PREPROCESSING
Forward-fill imputation (max 1-quarter gap)
Sector median fallback for longer gaps
Winsorization at 1st/99th percentiles
Accounting identity validation (Assets = Liab + Equity)
Z-score normalization (training data only)
STAGE 3: FEATURE ENGINEERING
Optional: NLP Features
FinBERT embeddings from filings
Volatility
Rolling Std, Cash Flow Var
Temporal Trends
QoQ Growth, 4-Qtr Slopes
Cash Burn Rate
Financial Ratios
Current Ratio, Debt/Equity
Interest Cov, ROA, Net Margins
STAGE 4: MODEL TRAINING & PREDICTION
Time-based train/validation/test split
Class imbalance handling (SMOTE, class weights)
Model: XGBoost (baseline) or LSTM (temporal)
Hyperparameter tuning via validation set
STAGE 5: RISK SCORE OUTPUT
Probability score (0-1) per company/quarter
SHAP-based feature importance explanations
Confidence intervals for predictions
STAGE 6: DEPLOYMENT & MONITORING
FastAPI Endpoint
Dashboard (Streamlit)
Alert System (Threshold-based)

- **Data Ingestion and Cleaning**: The system matches different datasets using distinct corporate IDs (CIKs), gathers data while conforming to API restrictions, and saves time by caching it locally. By completing small gaps, removing severe outliers , and checking that the accounting math truly adds up, the unorganized data is corrected.
- **Feature Engineering:** Unprocessed data is transformed into signals such as volatility, growth trends, and debt-to-equity ratios. Additionally, it offers optional use of natural language processing (NLP) for analyzing text-based indications of risk.
- **Model Training:** To prevent bias, divide data by time such as training on the past or testing on the more recent years, use class weight to deal with the frequency of bankruptcies, and tune models like XGBoost or LSTMs for accuracy.
- **Risk Score Output:** A 0–1 chance of distress is generated by the model. Additionally, it explains why using SHAP is beneficial.
- **Deployment:** Users can view near real-time risk scores and receive alerts when a company crosses a risk level due to the final system's packaging into an API (FastAPI) that feeds a live dashboard .

6.2 Potential Bottlenecks:

- **API Rate Limiting & Extraction Speed**: The total number of requests you may submit in a second is strictly restricted by the SEC EDGAR API. If not executed correctly, collecting data for 10,000–20,000 businesses over a number of years can take a long time.
- **Data Inconsistency & Reporting Gaps**: Financial filings frequently have data gaps or lack particular categories of data. As a result, the model may acquire missing features for certain companies, thereby creating a "garbage in, garbage out" problem.
- **Computational Cost of NLP:** Processing thousands of complex "Risk Factors" sections from 10-K documents is extremely resource-consuming and slow if you opt to incorporate the optional text analysis .

6.3 Improvements:

- **Use Local Caching & Batching:** Use of a local cache (such as a SQL database or Parquet files) to store raw JSON replies so as to get past API bottlenecks. This prevents downloading old files again and guarantees the extraction of fresh data
- **Automated Validation Pipelines:** Use of Accounting Identity tests to make sure that Assets = Liabilities + Equity in addition to standard cleaning. To maintain the integrity of the model, a document that fails this should be automatically flagged for manual review or excluded.
- **Precomputing Explanations:** During the batch inference stage, precompute and save SHAP values instead of computing them each time a user clicks on the dashboard. The user interface seems fast and quick as a result.

7. Metrics, Objectives, and Business Goals

7.1 Metrics:

- **ROC-AUC:** Evaluates the model's ability to differentiate between "distressed" and "healthy" businesses.
- **Precision@K & Recall@K:** When identifying a company as high-risk, we want to be certain that we are correct (Precision) and that we aren't missing the ones who actually fail (Recall).
- **Brier Score:** This evaluates whether or not our probability scores, such as 85%, are genuinely reliable and not merely speculative.

7.2 Objectives:

- **6–12 Month Lead Time**: The primary goal is to anticipate problems far enough ahead of time for stakeholders to take the necessary steps.
- **Explainability**: The system should provide an explanation (e.g., "High debt + falling revenue") for why it thinks a company is in jeopardy, not just a figure.

7.3 Business Goals:

- **Loss Prevention:** Helping banks and investors in avoiding millions of dollars from being lost when a business unexpectedly defaults.
- **Systemic Risk Monitoring:** Assisting officials identify pervasive economic "cracks" prior to a banking crisis.
- **Automation:** Transitioning from complicated, manual analyst assessments to a system that can continuously monitor thousands of businesses.

8. Failure Analysis

Even with strong modeling and engineering practices, failures can occur across the lifecycle of a financial distress early-warning system—from ingestion of SEC filings to post-deployment drift under changing market regimes. This section outlines key risks, their impact, and mitigation strategies aligned with MLOps best practices.

8.1. Potential Risks During Project Development

Data Quality and Filing Consistency Issues

a. Risk: SEC filings may contain missing values, inconsistent XBRL tags, restatements, or duplicated reports.
b. Impact: High

c. Mitigation :
  - Automated schema validation and accounting identity checks
  - Deduplication of filings by fiscal period
  - Robust imputation (short-gap forward fill, sector median fallback)
  - Outlier handling via winsorization

Label Noise and Ambiguity in “Distress” Events

a. Risk: Public distress labels (bankruptcy, delisting) may be delayed, incomplete, or inconsistently matched to firms.
b. Impact: High
c. Mitigation:
  - Use objective, verifiable public outcomes only
  - Standardize company identifiers using CIKs
  - Apply a grace window to exclude non-actionable imminent events
  - Track label sources and versions for auditability

Temporal Leakage and Improper Splitting

a. Risk: Random splits or improper feature scaling may leak future information, inflating performance.
b. Impact: High
c. Mitigation:
  - Strict time-based train/validation/test splits
  - Compute normalization and imputation statistics using training data only
  - Validate feature “as-of” dates relative to label windows

Class Imbalance and Prediction Reliability

a. Risk: Distress events are rare, causing models to favor non-distress predictions.
b. Impact: High
c. Mitigation:
  - Class-weighted loss functions and threshold tuning
  - SMOTE applied only to training data
  - Evaluation using Precision@K, Recall@K, and ROC-AUC rather than accuracy

Feature Instability and Engineering Errors

a. Risk: Financial ratios may explode due to small denominators or misaligned fiscal periods.
b. Impact: Medium
c. Mitigation:
  - Denominator guards and bounded ratios
  - Unit tests for feature calculations
  - Distribution monitoring for engineered features

8.2. Potential Risks During Deployment and Post-Deployment

Data Ingestion Failures

a. Risk: API rate limits, format changes, or downtime from SEC EDGAR or FRED.
b. Impact: Medium
c. Mitigation :
    - API throttling and retry logic
    - Local caching of raw data
    - Logged ingestion failures with manual review during evaluation runs

Model Drift Due to Economic Regime Changes

a. Risk: Shifts in macroeconomic conditions degrade model performance over time.
b. Impact: High
c. Mitigation:
    - Periodically monitor feature and prediction distribution drift during scheduled evaluation runs
    - Scheduled performance evaluations
    - Scheduled batch retraining and offline model comparison on recent historical data to assess performance degradation.

Scalability, Latency, and Cost Issues

a. Risk: Batch inference, SHAP explanations, or API usage may strain compute resources or increase costs.
b. Impact: Medium to High
c. Mitigation:
    - Batch scoring executed via scheduled or manual batch scripts using available GCP credits or local execution
    - Precompute SHAP values during batch inference
    - Resource-aware execution using lightweight services or manual batch runs

Security and Misuse Risk

a. Risk: Unauthorized API access or misuse of predictions as financial advice.
b. Impact: High
c. Mitigation:
    - Simple access control using API keys or restricted endpoints and basic rate limiting
    - Secure handling of credentials using environment variables and encrypted storage when available
    - Clear disclaimers positioning the system as decision-support only

8.3. Pipeline Failure Summary

Stage	Failure Mode	Detection	Recovery
Ingestion	API failure/throttling	Job alerts	Retry + cached fallback
Preprocessing	Missing or invalid ratios	Validation checks	Impute or exclude
Feature engineering	Ratio instability	Distribution monitoring	Bounds + recalculation
Training	Overfitting/crash	Validation metrics	Early stopping, resume
Deployment	API latency/crash	Basic service logs	Manual restart using managed services if needed
Inference	Timeout / slow SHAP	Execution-time logging	Serve cached batch results

9. Deployment Infrastructure

The system will be deployed using a cloud-native, containerized architecture on Google Cloud Platform (GCP). The MVP focuses on batch training and inference, with a lightweight API included to expose predictions and alerts.

9.1 Containerization & Execution

All components are packaged as Docker containers.

9.2 Workloads

Containers are deployed using Cloud Run (services and jobs), avoiding persistent infrastructure while supporting horizontal scaling.

- Data ingestion
- Feature engineering
- Model training
- Batch inference

- FastAPI service

- Evaluation and monitoring jobs

9.3 Data Storage

- **Google Cloud Storage (GCS):** Serves as the data lake for raw filings, processed datasets, model artifacts, and batch outputs.

- **Cloud SQL (PostgreSQL):** Stores structured data including company metadata, time-indexed features, risk scores, and alert history.

9.4 Model Training & Versioning

Models are trained in batch using scikit-learn / XGBoost. MLflow is used for experiment tracking, metric logging, and model versioning. The MLflow tracking server runs on Cloud Run, with artifacts stored in GCS.

9.5 API Deployment (MVP Scope)

A minimal FastAPI service exposes:

- `POST /predict`
- `GET /alerts`
- `GET /health`

The API serves batch inference results and supports limited on-demand scoring. It is deployed on Cloud Run and designed to be expanded later.

9.6 Scheduling & CI/CD

- **Cloud Scheduler**: Triggers retraining, inference, and evaluation jobs.
- **GitHub Actions:** Builds and deploys Docker images to Artifact Registry and Cloud Run.

9.7 Security

- GCP Secret Manager for credentials
- IAM-based access control

Encrypted storage and network traffic

10. Monitoring Plan

10.1 Operational Monitoring

Using GCP Cloud Monitoring, we track:

- API latency
- Job execution time
- CPU and memory usage
- Error rates and job failures

Alerts trigger on repeated failures or performance degradation.

10.2 ML Performance Monitoring

Using MLflow, we track:

- Training and validation metrics (AUROC, precision, recall)
- Model version performance
- Prediction distribution shifts

Scheduled evaluation jobs log metrics and detect drift, triggering retraining when needed.

10.3 Business Metrics

- Alert precision
- Lead time between alert and distress
- False positive rate

10.4 Logging & Traceability

- Structured logs for all jobs and API calls
- Model version logged with every prediction
- Full experiment lineage maintained via MLflow

11. Success and Acceptance Criteria

- Target ROC-AUC ≥ 0.80 on the test set
- Precision@K ≥ 80% for top-risk companies
- Automated end-to-end execution of the pipeline
- Risk explanations interpretable to domain experts

12. Timeline Planning

The project follows a phased timeline designed to incrementally build, validate, and deploy the Corporate Financial Distress Early-Warning System while allowing flexibility for iteration based on intermediate results. The schedule aligns with academic deadlines and emphasizes early validation to reduce downstream risk.

Phase	Task	Duration
Phase 1	Data Extraction & Initial Preprocessing	2 weeks
Phase 2	Feature Engineering & Exploratory Analysis	2 weeks
Phase 3	Model Training & Hyperparameter Tuning	2 weeks
Phase 4	Evaluation, Explainability & Visualization	1 week
Phase 5	Deployment & Monitoring Setup	1 week
Phase 6	Documentation, Testing & Final Report	1 week

Total Duration: 9 weeks

This timeline is intentionally modular. Each phase produces intermediate deliverables (e.g., cleaned datasets, baseline models, evaluation reports), enabling early validation and reducing the risk of late-stage rework. Adjustments may be made based on data availability, model performance, or integration complexity.

13. Additional Information

Several extensions and enhancements are considered optional and may be explored if time and resources permit. These are **explicitly non-blocking** and do not affect the core functionality of the system.

- **Optional NLP Enhancements:**
  Text-based risk signals may be extracted from the *Risk Factors* section of 10-K filings using transformer-based models (e.g., FinBERT). These features are exploratory and intended to complement, not replace, structured financial signals.

- **Interactive Dashboard:**
  A lightweight dashboard built with Plotly Dash or Streamlit may be used to visualize company risk scores, feature contributions (SHAP values), and historical risk trajectories.

- **Explainability & Transparency:**
  Model outputs will include interpretable explanations highlighting the most influential features driving each prediction, supporting trust and usability for non-technical stakeholders.

- **Future Extensions:**
  The system architecture is designed to be extensible, allowing future work to:

  - Predict multiple types of distress events (e.g., liquidity crisis, sustained cash burn)

  - Incorporate additional macroeconomic or sector-specific indicators

  - Support alternative modeling approaches or evaluation strategies

These enhancements align with the project's educational objectives while keeping the baseline system achievable within the course timeframe.