Intelligent Contract Risk Analysis

Milestone 1: ML-Based Contract Risk Classification

Submitted in fulfillment of the Intro to GenAI Capstone Project

Newton School of Technology (NST), Sonipat

Team Members:
Ashutosh Singh (2401010109)
Ranvendra Pratap Singh (2401010373)
Shreya Suman (2401020068)

March 2026

Declaration of Technical Integrity

The "No GenAI" Rule: We, the undersigned team members, formally affirm that the core logic of this project is our original work. All data parsing modules, Natural Language Processing pipelines, feature engineering implementations (TF-IDF vectorization), Scikit-Learn model training, and the Streamlit inference engine were manually explicitly written and engineered by the team. We have completely successfully completed Milestone 1 without reliance on Generative AI or external LLM API dependencies for our core contractual risk inference mechanism, strictly adhering to the grading rubric.

Contents

1 Problem Statement

Modern corporate entities and legal firms process thousands of complex legal contracts quarterly, ranging from Non-Disclosure Agreements (NDAs) to multi-million dollar vendor agreements. The manual review process of these documents is notoriously labor-intensive, time-consuming, and highly susceptible to human error. Legal professionals must meticulously scan dozens of pages of dense legal jargon to locate and critically evaluate specific clauses related to indemnification, financial liabilities, non-competes, and arbitrary termination conditions.

The primary challenge addressed in this project is the lack of automated, low-latency triage systems for legal teams. The objective is to design, implement, and deploy a supervised machine learning-based Contract Analysis System capable of accepting raw unstructured contract text, intelligently segmenting it into individual semantic clauses, and accurately classifying each clause into distinct functional risk categories (**High, Medium, Low**). By flagging high-severity clauses and providing quantitative confidence scores, this intelligent system drastically reduces the initial review burden, allowing legal domain experts to prioritize critical contractual threats instantly.

2 Data Description

Source & Provenance: The machine learning pipeline was strictly trained and evaluated using the **Contract Understanding Atticus Dataset (CUAD) v1**, a specialized and highly respected open-source corpus curated specifically for natural language processing within the legal domain.

Corpus Structure & Features: The raw dataset consists of a massive, nested JSON architecture containing 510 unique, fully annotated commercial contracts. Rather than simple flat text, each document is structured hierarchically. The dataset contains specific *paragraphs*, and within those paragraphs, domain experts have attached *Question-Answer (QA) pairs*. These QA pairs correspond to 41 distinct types of legal clauses.

Extraction Schema: Our custom data pipeline was engineered to natively parse this JSON. It specifically filtered out any text blocks where the QA `is_impossible` flag was set to true (indicating the clause did not exist in the contract). For valid clauses, the system extracted the `clause_type` (e.g., `__Cap On Liability`, `__Governing Law`) from the QA ID, and the exact legal text string from the `answers` key.

The core predictive variable for the machine learning models was the raw, processed text string of the clause (`cleaned_text`), mapped directly to the categorized risk level label.

3 EDA Process (Exploratory Data Analysis)

During the Exploratory Data Analysis phase, we systematically traversed the entire dataset to understand the fundamental statistical properties of the legal text corpus. Several critical insights fundamentally shaped our machine learning methodology:

- **High Cardinality and Severe Class Imbalance:** The dataset natively contains 41 highly specific unique legal clause types. However, general boilerplate clauses (such as *Signatures*, *Governing Law*, and *Date of Agreement*) appeared at drastically higher frequencies compared to severe penal clauses (like *Liquidated Damages* or *Uncapped Liability*).
- **Variable Context Lengths:** We observed significant variance in text token lengths. While some clauses consisted of a single sentence establishing a date, others (like IP Ownership clauses) spanned multiple dense paragraphs.

Strategic Risk Mapping

Predicting 41 individual, heavily imbalanced classes using classical machine learning algorithms typically results in severe overfitting and poor macro-generalization on unseen contracts.

To resolve this and create a viable real-world utility, we engineered a mapping architecture that consolidated the 41 logical types into three robust, actionable risk tranches:

1. **High Risk:** Defined as clauses inflicting severe financial, operational, or legal threats (e.g., Absolute liabilities, Termination constraints, Non-competes, IP Ownership Assignments).
2. **Medium Risk:** Defined as clauses establishing operational friction or moderate constraints (e.g., Audit rights, general bounds, Insurance requirements).
3. **Low Risk:** Defined as standard administrative/boilerplate language carrying no active threat (e.g., Document names, Dates, Governing jurisdiction).

This engineering decision resulted in the extraction of **12,679 viable training examples**, transforming an unworkable 41-class problem into a much healthier, balanced 3-class distribution:

- **Low Risk:** 5,629 clauses
- **Medium Risk:** 4,340 clauses
- **High Risk:** 2,710 clauses

4 Methodology

The core methodology strictly adheres to Milestone 1 constraints, utilizing classical Natural Language Processing (NLP) combined with supervised classification models, entirely omitting Generative AI or Deep Learning dependencies.

4.1 Preprocessing Pipeline

Raw legal text is inherently dirty and full of formatting artifacts. Our cleaning function executed the following systematic standardizations:

- **Case Normalization:** All text was transformed to lowercase to ensure absolute term uniformity.
- **Regex Scrubbing:** We utilized Regular Expressions (`re`) to strip extraneous whitespace and obliterate special non-alphanumeric symbols. However, we intentionally retained essential punctuation (`.,;:\-'"()/`) to preserve the structural contextual boundaries necessary for logical clause flow.

4.2 Feature Engineering: TF-IDF Vectorization

To make the text comprehensible to mathematical learning algorithms, we utilized **Term Frequency-Inverse Document Frequency (TF-IDF)**.

1. **Vocabulary Constraints:** The vectorizer was parameterized to `max_features=5000`. This critical dimensionality reduction prevents the model from memorizing rare, non-generalizable names or dates unique to specific contracts, forcing it to learn actual legal syntax.
2. **N-Gram Architecture:** We utilized `ngram_range=(1,2)`. This allows the sparse matrix to capture not just standalone legal terms (unigrams like "liability"), but also essential contextual pairings (bigrams like "breach of" or "not exceed").
3. **Stop Words:** Standard English stop-words ('the', 'and', 'a') were removed to eliminate structural noise.

4.3 Algorithms Selected

Multiple classical architectures were evaluated:

1. **Logistic Regression:** Selected for its renowned efficiency and convergence reliability in phenomenally high-dimensional, sparse matrix spaces (like 5000-feature TF-IDF data). Uniquely, it provides straightforward probabilistic outputs (`predict_proba`),

which allowed us to build custom Confidence Thresholding rules inside the Streamlit UI.

2. **Decision Tree Classifier:** Evaluated to determine if non-linear, hierarchical branching rules between isolated legal terms outperformed linear boundaries.

3. **Random Forest Classifier:** An ensemble bootstrap-aggregation approach utilized specifically to combat the high variance and potential overfitting observed in isolated Decision Trees.

5 Evaluation Results

Ensuring the model would perform reliably on previously unseen contracts was paramount. Therefore, all models were evaluated inside isolated data pipelines using **5-Fold Stratified Cross-Validation**. The stratification ensured that the relative class distribution of High/Medium/Low clauses remained strictly identical across all training and holdout folds.

Why F1-Score? While Accuracy is reported, we relied heavily on the **Weighted F1-Score** to determine the champion model. In legal risk analysis, False Positives (flagging safe text as risky) induce wasted review time, while False Negatives (missing a High-Risk liability cap) are legally disastrous. The F1-Harmonic Mean properly balances these precision/recall tradeoffs.

Model Architecture	Cross-Validation Accuracy	CV F1-Score (Weighted)
Logistic Regression	**0.8856 (88.56%)**	**0.8859**
Decision Tree Classifier	0.8117 (81.17%)	0.8084
Random Forest Classifier	0.8138 (81.38%)	0.8058

Final Selection: Logistic Regression decisively outperformed the tree-based ensemble methods, achieving a near 89% F1-Score. It was subsequently serialized as `best_model.pkl` for production inference.

6 Optimisation Strategies

Engineering the mathematical models required significant tuning to ensure high performance and prevent statistical illusions. Key optimizations implemented included:

1. **Pipeline Architecture (Preventing Data Leakage):** The most critical programmatic optimization was encapsulating the `TfidfVectorizer` directly *inside* a `Scikit-Learn Pipeline` object alongside the classifier. If TF-IDF is fit on the

entire dataset prior to cross-validation, the vectorizer "learns" the vocabulary of the validation holdout test set (Data Leakage), artificially inflating accuracy scores. Our pipeline guarantees the vectorizer only extracts features dynamically from the training split of every iterative fold.

2. **Algorithmic Class Weighting:** The Logistic Regression algorithm was explicitly instantiated with the hyperparameter `class_weight='balanced'`. This adjusts the internal cost function to algorithmically penalize misclassifications on the minority 'High Risk' class much more severely than errors on the majority 'Low Risk' class, ensuring the model doesn't simply skew towards predicting safe administrative text.

3. **Real-Time Inference UI & Confidence Thresholds:** The final system leverages Streamlit for a user-friendly frontend. We optimized the UI by integrating `Plotly Express` for interactive data visualizations (Risk distribution pie charts and histograms). To combat False Positives in a production setting, we exposed a custom "Confidence Threshold" slider in the sidebar. This allows users to instruct the backend inference engine to only flag High-Risk clauses if the mathematical `predict_proba` exceeds a certain threshold (e.g., > 0.50).

7 Team Contribution

The workload was systematically divided to ensure comprehensive end-to-end execution of Milestone 1 constraints.

Ashutosh Singh (2401010109)	Developed the custom data preprocessing scripts responsible for parsing the complex hierarchical CUAD JSON architecture. Implemented the core data logic mapping the 41 original clause classes into the stable 3-tier risk system. Architected the base Scikit-learn Pipeline setup preventing data leakage, trained evaluated ML models, and contributed structural design mechanics to the overarching Streamlit UI dashboard logic.
Ranvendra Pratap Singh (2401010373)	Implemented the foundational NLP TF-IDF feature engineering strategy, handling stop-word removal, regex optimization, and n-gram bounding. Co-defined the execution of the pipeline architecture, heavily assisted in monitoring the ML model training/evaluations, and acted as the primary architect for building the interactive Streamlit UI functionalities, including the native Plotly dataset visualizations.
Shreya Suman (2401020068)	Led the execution of the Scikit-learn validation pipeline. Successfully hyperparameterized and trained the various competing ML

classification models (Logistic Regressions, Decision Trees, Random Forests). Primarily performed the critical Stratified K-Fold cross-validation checks to establish the champion model metrics, extracted keyword logic, and heavily designed front-end elements of the final user-facing Streamlit client integration.