Predicting Chicago Food Inspection Failures

Jingzhi Yang Lingyue Ji Nate Assefa

Abstract

This project explores whether Chicago food inspection failures can be predicted before inspection to help public health officials use limited inspection resources more effectively. The analysis uses two cleaned, schema-matched splits from the City of Chicago data portal (training and test), each with 1,000 inspections from 2010 to 2025. Predictors include facility type, inspection type, city risk level, location variables (ZIP code, latitude, longitude), and calendar features from inspection dates. To further avoid leakage, we add a conservative numeric proxy for violation count that avoids detailed violation descriptions and does not encode the current inspection outcome. For modeling, we benchmark four approaches: class-weighted logistic regression, a decision tree, a random forest, and a gradient-boosted tree, all using a shared preprocessing and modeling pipeline. Categorical variables are one-hot encoded with safeguards for unseen categories at test time, while numeric variables remain unchanged. Model selection uses stratified 5-fold cross-validation, preserving the roughly 19% failure rate in each fold to stabilize estimates under class imbalance. We report both threshold-free metrics (ROC–AUC and PR–AUC) and threshold-based metrics (precision, recall, and F_1 for the fail class), emphasizing PR–AUC since failures are rare. Logistic regression performs best and most consistently, achieving a mean cross-validated ROC–AUC of 0.824 and PR–AUC of 0.582 with a balanced precision–recall trade-off. When retrained on the full training set, it maintains strong test performance (ROC–AUC = 0.811; PR–AUC = 0.548). These results suggest that a regularized linear model captures the most useful predictive structure in this dataset. The findings show that structured pre-inspection metadata can provide a useful early warning signal of inspection failure under a principled no-leakage design, while also highlighting the need for larger samples and time-based validation to assess robustness under changing enforcement and reporting.

1. Data

1.1. Background

Each day, hundreds of restaurants, grocery stores, and cafeterias operate in Chicago. However, only a fraction of the results are inspected at any given time. The city depends on a limited number of public health inspectors to safeguard millions of residents, and their resources are constrained. Some facilities present greater risks than others, but inspectors often determine where to go next based on experience and intuition. This project addresses a practical question: can we predict which facilities are most likely to fail before an inspection occurs?

Although the question can be framed as a binary classification task, the underlying challenge is more complex. Inspection failures are infrequent, facility behavior varies over time, and pre-inspection information is often incomplete or inconsistently formatted. Chicago's open-data portal offers detailed inspection records, yet it also highlights limitations in municipal data collection, such as ZIP codes stored as decimals, inconsistent violation note styles, and occasionally inaccurate geographic coordinates.

Contextualizing the data requires examining the City of Chicago's data collection process. Food establishments are subject to routine inspections, while complaints, license renewals, and follow-ups prompt additional visits. Inspectors complete digital forms during or after inspections, recording results, violations, and facility details. These records are entered into the Food Inspection Reporting System and later published on the city's public data portal. Because this process is ongoing rather than structured as a formal survey, the dataset reflects the realities of fieldwork, including irregular formatting, policy changes, seasonal trends, and occasional missing values.

Our research method builds on this context by constructing a pipeline that cleans and standardizes inspection records, extracts relevant features from dates, risk levels, location, and facility descriptors, and excludes information unavailable prior to inspection. We then benchmark several predictive models and use descriptive comparisons to examine which types of facilities struggle with compliance, how higher-risk sites differ from lower-risk ones, and whether failure patterns shift across time or space.

By combining predictive modeling with analysis of the inspection ecosystem, this project aims not only to build an effective model but also to interpret the structure behind it. In doing so, it offers a perspective on how a city monitors public health, the operational constraints involved, and the patterns that emerge from large-scale inspection data.

1.2. Variables

The dataset contains both original inspection fields and engineered features designed to support modeling. The primary original variables are as follows:

- **inspection_id**: Unique identifier assigned to each inspection event.
- **inspection_date**: Calendar date of the inspection; used to construct temporal features.
- **facility_type**: Category of the establishment (e.g., restaurant, grocery store, school), reflecting differences in operational risk and food-handling practices.
- **risk**: City-defined risk level with three categories: Risk 1 (High), Risk 2 (Medium), and Risk 3 (Low).
- **inspection_type**: Reason for inspection (e.g., canvass, complaint, license, re-inspection).
- **zip, city, latitude, longitude**: Location variables identifying where the facility operates and enabling geographic analysis.
- **results**: Official inspection outcome string ("pass," "pass w/ conditions," "fail," "out of business"); used to define the prediction target.
- **violations**: Free-text list of cited health-code violations, often semi-structured and numbered by inspectors.

Engineered features are developed to enhance the modeling strategy:

- **viol_count**: Number of violations extracted from the free-text field.
- **risk_ord**: Ordinal encoding of the risk level (High = 3, Medium = 2, Low = 1).
- **ins_year, ins_month, ins_wday**: Year, month, and weekday derived from the inspection date.
- **target_fail**: Binary target variable equal to 1 if the inspection result is "fail," 0 otherwise.

Table 1. Summary of the cleaned dataset.

	Training	Test
Inspections	1,000	1,000
Variables after cleaning	18	18
Failure rate	18.6%	$\sim$19%
Risk levels	3	3
Facility types	41	41
Inspection types	19	19

These variables capture facility characteristics, regulatory risk context, and the circumstances of each inspection. They form the basis for predicting inspection failures while avoiding data leakage by excluding violation text and other post-inspection information from the predictive model.

1.3. Data Cleaning

Each data split contains 1,000 inspections. After cleaning, 18 variables remain, with approximately 19% of training records classified as failures. Facility type encompasses roughly 41 categories, predominantly restaurants. The dataset includes approximately 19 inspection types, three risk levels, and geographic coordinates centered in Chicago (41.88° N, -87.67° W). Inspections span from 2010 to 2025 and are evenly distributed. Failure rates increase with risk level, although Risk 2 slightly exceeds Risk 3 in this sample.

A reproducible pipeline processes both training and testing data to reduce split-specific bias. The code verifies required fields, removes duplicate `inspection_id` values, trims whitespace, and maps entries to standardized categories. City names are normalized, and facilities located in Chicago receive a binary indicator. ZIP codes are converted from floating-point numbers to five-digit strings, with invalid or missing values set to `NA`. Inspection dates are converted to `datetime` objects, and year, month, and weekday features are extracted. Results are consolidated into a compact outcome set, and risk labels are ordinally encoded.

To summarize issues while preventing information leakage, violation notes are converted to a numeric count using a conservative parsing approach. Non-informative placeholders are assigned a value of 1, while true `NA` entries are assigned 0. Latitude and longitude values outside Chicago's bounding box (41.60–42.10° N, -87.95 to -87.50° W) are set to missing to limit the influence of clear geographic errors.

1.4. Challenges

The dataset has common municipal record issues, including inconsistent spelling, literal "nan" strings, and varied ZIP code formats. The violations field is semi-structured;

therefore, a simple counting rule is preferred over more complex parsing that could introduce brittleness or information leakage. Some facility types in the test set may be absent from training. We address this by using `OneHotEncoder(handle_unknown="ignore")`, which handles unseen categories at test time safely without requiring manual relabeling. They help promote reproducibility and robustness while preserving the integrity of the pre-inspection signal.

2. Methods

2.1. Modeling Framework

Our modeling framework predicts facility inspection failure using only pre-inspection information and is formulated as a supervised binary classification problem. The analysis follows schema validation, construction of a no-leakage feature set, model benchmarking, and final test-set evaluation.

We include only predictors available before an inspection begins, such as calendar variables, risk levels, location, facility descriptors, and a numeric violation count, and exclude free-text violation descriptions and any post-inspection fields. The design prevents information leakage and aligns with real-world constraints faced by inspectors.

2.2. Model Choices

We benchmark a small but diverse set of models appropriate for mixed tabular data, with increasing complexity across the set. Our baseline is logistic regression with ℓ_2 regularization, chosen for its interpretability, stable probability estimates, and strong performance with one-hot-encoded categorical predictors. The model estimates the probability of failure as

$$\Pr(y{=}1 \mid x) = \sigma\left(\beta_0 + \sum_j \beta_j x_j\right),$$

where σ is the logistic function. Because failures are relatively rare, we apply class weighting so the model places greater emphasis on correctly identifying failed inspections.

We include three tree-based models to assess whether nonlinear structure or feature interactions improve predictive performance. A decision tree serves as a simple, interpretable rule-based reference. A random forest reduces variance through ensembling and captures moderate nonlinearities. A gradient-boosted decision tree provides a more flexible learner that often performs strongly on structured data. Together, these models test whether the predictive signal in pre-inspection metadata is largely linear at this sample size or whether interaction-rich models yield meaningful gains.

For preprocessing, predictors are grouped by type. Continuous variables include latitude, longitude, `viol_count`, `ins_year`, and `ins_month`. The ordinal variable `risk_ord` captures inspection risk. Categorical variables include `facility_type`, `inspection_type`, ZIP code, and `city_is_chicago`. Categorical fields are one-hot encoded with safeguards for unseen categories at test time. The engineered violation count replaces raw violation text to preserve the no-leakage constraint.

2.3. Training and Tuning

All models use the same preprocessing and modeling pipeline to allow for fair comparison. Categorical predictors are encoded with `OneHotEncoder`, which can handle unknown levels, and numeric variables are left unchanged. This approach ensures that both linear and tree-based models work with the same input structure.

We train and select models using stratified 5-fold cross-validation on the training set and keep the failure rate at about 19% in each fold to obtain stable estimates for the imbalanced outcome. To address the low number of failed inspections, we use class-weighted loss functions in both logistic regression and tree-based models. For logistic regression, we tune the regularization strength `C` with a small grid search instead of relying on default values. Tree-based models are tested with conservative baseline settings, which allows for fair comparison at this sample size and leaves room for more tuning in future research.

We report both threshold-free and threshold-based metrics. ROC–AUC and PR–AUC measure ranking quality, with PR–AUC highlighted because failures are rare. We also report precision, recall, and F_1 for the fail class to show practical trade-offs. After cross-validation identifies the best model, we retrain it on the full training set and evaluate it once on the held-out test set to estimate performance on new data. Since failures are rare in our sample, PR–AUC is especially useful for showing whether the model can identify higher-risk facilities better than the base rate.

2.4. Validation and Reproducibility

The provided test set is used only for the final evaluation and is never accessed during model development. This separation ensures that test metrics reflect generalization rather than tuning feedback. The workflow uses fixed random seeds and applies a versioned cleaning function identically to both splits. These measures allow for the replication of the main results from the original data and code.

3. Results

3.1. Main Results

The results show that pre-inspection metadata can help predict which facilities are likely to fail food safety inspections.

Table 2. Mean 5-fold cross-validation performance on the training set.

Model	ROC–AUC	PR–AUC	Precision	Recall	F_1
Logistic regression	0.824	0.582	0.402	0.651	0.497
Gradient boosting	0.802	0.514	0.657	0.269	0.380
Random forest	0.792	0.487	0.724	0.076	0.134
Decision tree	0.626	0.273	0.382	0.404	0.391

The patterns we observe align with the expectations of the domain. Most risk scores fall into higher-risk categories (Figure 1), and violation counts are right-skewed, with a small number of inspections reporting many violations (Figure 2). The facilities are clustered within the Chicago area (Figures 3 and 4), and inspections are fairly evenly distributed over years, months and days of the week (Figures 5, 6, and 7). Higher-risk establishments fail more often (Figure 8), and failure rates vary between common types of facilities, with mobile food dispensers and long-term care facilities among the highest in the top ten (Figure 9). In general, these descriptive patterns have no leakage, with `viol_count` treated as a conservative proxy for previous compliance.

3.2. Hyperparameter Validation

For model performance, logistic regression gave the most reliable results. In stratified 5-fold cross-validation (Table 2), it reached ROC–AUC $= 0.824$ and PR–AUC $= 0.582$, with a good balance between precision and recall for the fail class (Precision $= 0.402$, Recall $= 0.651$, $F_1 = 0.497$). Gradient boosting also performed well in ranking (ROC–AUC $= 0.802$), but it favored precision over recall. The random forest had very high precision but very low recall. The single decision tree was the weakest baseline.

The ranking offers useful insights. In a dataset with 1,000 training inspections, logistic regression performed best, suggesting that linear boundaries capture most of the predictive structure once categorical variables are one-hot encoded and class imbalance is addressed. This result is also reassuring, since strong performance from a simple, regularized model reduces worries about overfitting or depending on unstable interactions.

To make sure our model did not perform well solely because of default settings, we performed a brief hyperparameter check for the primary model. We used a stratified 5-fold cross-validation grid search and selected models based on ROC–AUC. For logistic regression, we tested regularization strengths $C \in \{0.01, 0.1, 1, 10\}$ while keeping the ℓ_2 penalty and solver fixed. This step confirms that a regularized linear model provides a good fit for our feature set.

The tree-based models were benchmarked using conservative baseline settings with class-balancing where supported. A broader search over parameters such as maximum depth, learning rate, and minimum leaf size would help determine whether their weaker performance reflects limits of the current features or limited tuning.

3.3. Test Performance

After retraining on the full training set and evaluating on the held-out test set, logistic regression achieved strong out-of-sample performance: ROC–AUC $= 0.811$, PR–AUC $= 0.548$, Precision $= 0.429$, Recall $= 0.691$, and $F_1 = 0.530$. The ROC and PR curves show clear improvement over random guessing (Figures 10 and 11). The small drop from cross-validation performance to test performance suggests that the model generalizes well to new inspections under our current design. Moreover, the PR–AUC remains substantially above the baseline failure rate of about 19%, indicating that the model identifies failures more effectively than a naive majority-class prediction.

With a probability threshold of 0.5, the confusion matrix (Figure 12) shows 628 true negatives, 178 false positives, 134 true positives, and 60 false negatives. The model identifies about 69% of failed inspections, but it also flags some facilities that actually pass. If missing an unsafe facility is riskier than doing extra follow-up, this threshold makes sense. On the other hand, if staff resources are tight and false alarms are costly, raising the threshold can improve precision, though recall will drop.

3.4. Confidence and Robustness

We have moderate-to-high confidence that the model captures meaningful patterns for three reasons. First, cross-validation and test results are consistent (Table 2), with ROC–AUC decreasing only slightly from 0.824 to 0.811. Second, the precision–recall curve remains well above the base failure rate (Figure 11), indicating that predicted risk scores meaningfully separate higher- and lower-risk inspections. Third, the model's performance aligns with descriptive patterns visible in risk-level and facility-type comparisons (Figures 8 and 9), supporting the face validity of the learned signal.

3.5. Limitations

There are still some factors that limit the model's reliability. The dataset is small and only represents a simplified version of a much larger municipal record. With only 1,000 training observations, performance may be affected by rare categories, unusual facility types, or differences in how data is reported locally. Since the inspections cover the years 2010 to 2025, changes in enforcement, reporting, or facility behavior over time could also lower performance in recent years. To better test how well the model handles changes

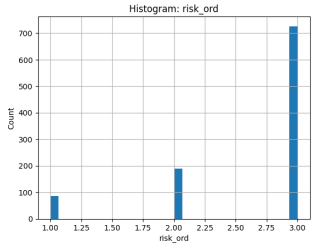

Figure 1. Histogram of the ordinal risk score.

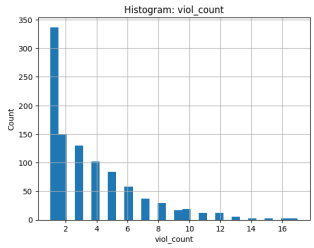

Figure 2. Histogram of violation counts.

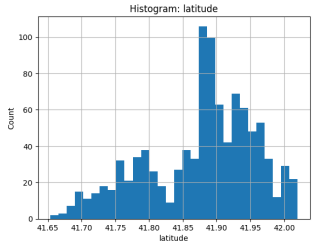

Figure 3. Distribution of facility latitude.

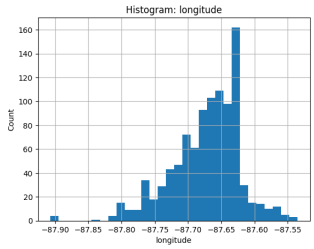

Figure 4. Distribution of facility longitude.

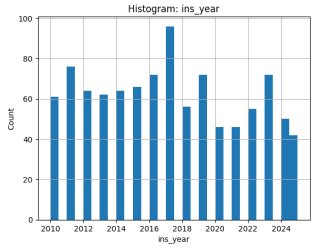

Figure 5. Distribution of inspection years.

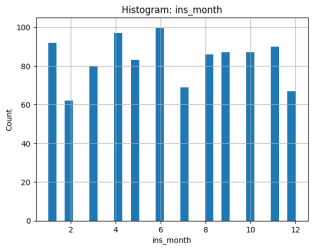

Figure 6. Distribution of inspection months.

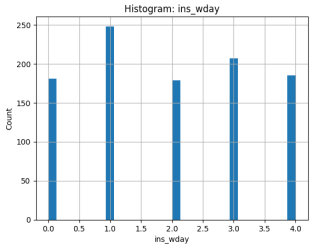

Figure 7. Distribution of inspection weekdays.

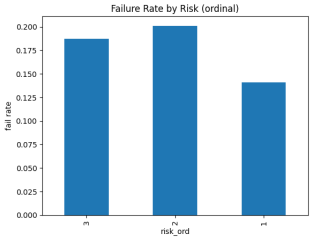

Figure 8. Failure rate by risk level.

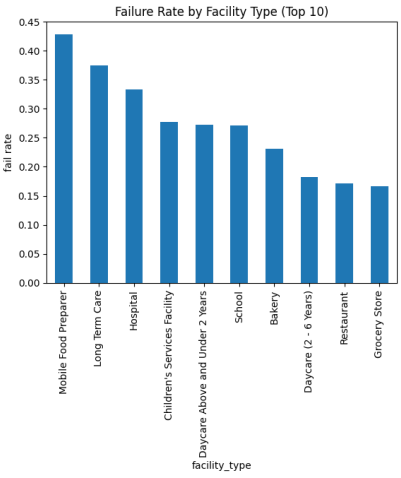

Figure 9. Failure rate by facility type (top 10).

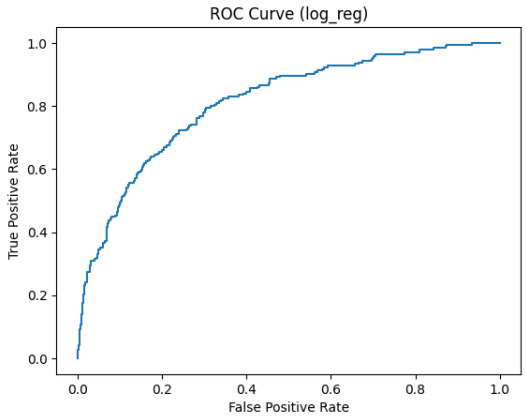

Figure 10. ROC curve for the logistic regression model.

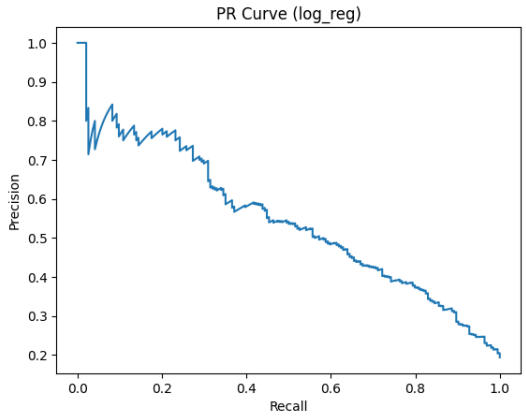

Figure 11. Precision–Recall curve for the logistic regression model.

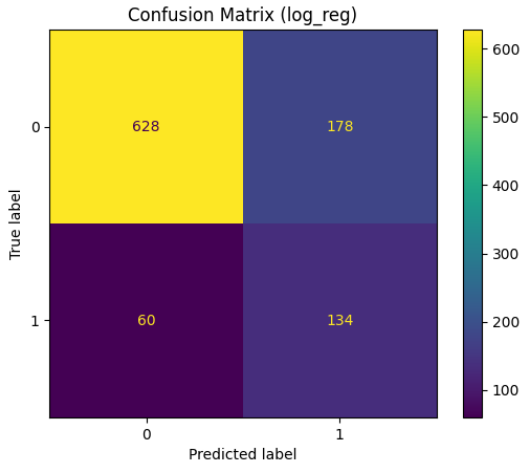

Figure 12. Confusion matrix for the logistic regression model on the test set.

over time, it would help to train it on earlier years and test it on later years. Additionally, it is important to consider how feature selection impacts the results.

The no-leakage rule means that only information available before the inspection can be used as features. It is a good choice for real-time decision support, but it may limit the predictive performance of the model unless more detailed pre-inspection data is included. Possible improvements could be adding structured records of past inspection results, summaries of earlier violations, or neighborhood-level data, as long as these do not include any information from after the inspection.

4. Conclusion

This project examined whether Chicago food inspection failures can be predicted before an inspection begins using only pre-inspection information. After cleaning and standardizing two schema-matched samples of 1,000 inspections each, we built a no-leakage feature set with facility type, inspection type, risk level, location variables, calendar features, and an engineered violation count. We benchmarked several classifiers using stratified 5-fold cross-validation and evaluated the best-performing model on a held-out test set.

The main finding is that a class-weighted logistic regression model performed best and was the most consistent across evaluation settings. On the test set, it achieved ROC–AUC $= 0.811$ and PR–AUC $= 0.548$, indicating meaningful ranking performance relative to the baseline failure rate of roughly 19%. At a probability threshold of 0.5, the model identified about 70% of failed inspections, suggesting practical value for triage when the goal is to reduce missed failures. These results show that even without free-text violation descriptions or post-inspection fields, structured pre-inspection metadata can provide a useful early-warning signal.

A natural critique is that the dataset is modest and that more flexible ensemble models might outperform linear methods with broader tuning. We addressed this concern in two ways. First, the small gap between cross-validation and test results suggests limited overfitting under the current sampling design. Second, we validated the primary model's hyperparameters by tuning regularization strength across a small grid. However, we did not exhaustively tune the ensemble methods. Expanding the hyperparameter search for gradient boosting and random forests or applying nested cross-validation would provide a stricter comparison and may reveal additional gains.

Another concern is stability across time and context. As inspections span 2010 to 2025, changes in policy, staffing, inspection priorities, or compliance norms could introduce temporal drift. This matters for operational use. A model

that performs well on pooled historical data may be less reliable if the most recent inspection environment differs substantially from earlier years. Future work should incorporate time-based validation and evaluate whether separate thresholds or models are needed for different facility categories or inspection programs.

Despite these limitations, the project demonstrates the feasibility of pre-inspection failure prediction under a principled no-leakage design. With a larger training sample, richer non-leaking pre-inspection features, and explicit time-split testing, this approach could become an interpretable decision-support tool that helps allocate limited inspection resources more efficiently while maintaining public health priorities.

5. Team Collaboration

Jingzhi managed the workflow and brought together the results. She also helped with the technical pipeline. For Milestone 1, she organized files, made the data dictionary, and wrote the report draft. In Milestone 2, she reviewed validation results, checked that everything could be reproduced, and led the writing. In Milestone 3, she compared cross-validation and test-set results and wrote the analysis. For the final project, she sets deadlines, helps review and improve code with Nate, writes the Analysis and Conclusion sections, and puts together the project to ensure the final submission is clear and consistent.

Lingyue led data collection and managed the project's repository. For Milestone 1, she found the raw Chicago inspection data, set up the GitHub repository, and exported cleaned training and testing files. In Milestone 2, she improved these files and ensured the data format remained consistent for modeling. In Milestone 3, she ensured the new features and data changes aligned with the modeling code's requirements. For the final project, she continues managing the GitHub repository and writes the Data and Methods sections based on feedback from the milestones.

Nate was primarily responsible for building the models and ensuring technical details were clear. For Milestone 1, he reviewed the data cleaning and ensured the data format was correct, and created visualizations for the EDA. In Milestone 2, he set up the modeling pipeline and trained basic and ensemble models. In Milestone 3, he ran cross-validation, produced model results, and made diagnostic plots. For the final project, he improves the code, helps write about the model details, and conducts a final check to ensure the writing matches the code and results.

References

City of Chicago. Food Inspections: City of Chicago Data Portal. https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5, 2025. Accessed September 26, 2025.