Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project

Final Report

Unlocking Global Capital: AI-Powered Prediction, Access, and Verification of Investor Decision Makers

by

Daniel Bowman

Email: daniel.bowman24@imperial.ac.uk

GitHub username: acse-db1724

Repository: ese-ada-lovelace-2024/irp-db1724

Supervisors:

Antony Sommerfeld

Dr. Yves Plancherel

August 2025

Table of Contents

1. Introduction 9
1.1. Review of Existing Work 9
1.2. Contribution of this Study 10
2. Methodology 10
2.1. Data Exploration 10
a) Missingness 11
b) Duplicates 11
c) Domain Analysis 11
d) Cleaning Steps Taken 11
2.2. Pattern Mining 12
a) Email Tokenisation 12
b) Template Rule Mining 12
c) Template Exploratory Analysis 13
d) Feature Engineering 14
2.3. Prediction Model 15
a) Model Architecture 15
b) Training 16
c) Stratified Error Analysis 16
d) Synthetic Padding Strategy 19
e) Benchmarking 20
f) Segmented Modelling by Name Complexity 21
g) Hyperparameter Tuning and Final Model Selection 23
h) Testing with C++ Inference Engine 24
i) GP Data Brought In 26
3. Results 26

3.1. Final Training 26
3.2. Test Results 27
a) Comparison to Benchmark and Baseline 27
3.3. Cross Domain Generalisation 28
3.4. Feature Importance 28
3.5. Latency and Deployment 28
4. Discussions 29
4.1. Challenges 29
4.2. Strengths 29
4.3. Weakness 30
4.4. Future Work 30
5. Conclusion 31
6. References 32
7. Appendix A. Data Sheet 34
8. Appendix B. ETL Pipeline Diagram 35
9. Appendix C. Feature Ablation Results 36
10. Appendix D. Reproducibility and CatBoost Params 37
11. Appendix E. API Specification 38
12. Appendix F. Production Inference Pipeline 39
13. Appendix G. Feature Importance 40
14. Appendix H. Glossary 41

List of Tables

Table 1: Feature Fail/Overall Rate Comparison 22

Table 2: Unified vs. Segmented Model Performance Comparison 22

Table 3: Hyperparameter Tuning Results 23

Table 4: Failure Signals Identified through Logistic Regression 25

Table 5: Final Test Results 27

List of Figures

Figure 1: Investor Count vs. Template Diversity Scatter Plot 14

Figure 2: Accuracy@1 Across Domains 17

Figure 3: Template Diversity vs. Failure Rate Scatter 18

Figure 4: Accuracy@1 Banded by Firm Investor Count 19

Figure 5: Fail Rate vs Coverage Scatter: 25

Abstract

Automating investor contact maintenance is critical for scalable capital raising in a sector plagued by turnover and data decay, yet teams rely on slow, costly lookups that struggle with real-world naming diversity. This study develops a pipeline coupling template mining, ranking, and domain resolution, to test whether joint discovery and learning-to-rank outperform flat classification and LLM baselines. On a firm-held-out test of unseen firms the system achieves Accuracy@1 = 0.9292 and Recall@3 = 0.9970. A placebo shows synthetic padding improves accuracy (0.9277) while random duplication collapses it (0.5091). CPU inference is <50ms per query; with the main limitation being domain mapping for multi-domain/shared-infrastructure firms.

AI Acknowledgement Statement

Generative AI and AI assistants were used in part throughout code development, namely Google's Copilot (m365.cloud.microsoft/chat/) and GitHub's CoPilot (https://github.com/features/copilot). Despite assistance, all submitted work is my own. The following is a list of areas of code where generative AI was used for assistance:

- CoPilot was used to debug and guide implementation in the REST API service that exposes the underlying model.
- CoPilot was used to help build the CatBoost C++ model as documentation was sparse.
- CoPilot helped guide and debug PyBindings and related code, especially optional params and linking implementations that for some reason failed to build in containerized environments.
- CoPilot suggested and helped implement and debug MSGpack related code.
- CoPilot helped guide the usage of FuzzyMatch C++ library in the inference engine.
- CoPilot helped heavily in debugging ICU NFKD code in C++ as implementation was not as simple as it was in Python.
- CoPilot helped suggest robustness and edge case unit tests for the NameDecomposer in C++.
- CoPilot suggested nlohmann JSON library and helped debug internal library errors in third party CURL code.
- CoPilot helped generate integration tests based on existing C++ tests.
- CoPilot helped generate Python unit tests for NFKD, especially edge cases that feature Unicode characters.
- CoPilot helped debug SQLalchemy and database code in python.
- CoPilot helped consolidate Python errors around stratified split.
- CoPilot helped debug existing firm profiler code.

1. Introduction

Accurate investor contact data is vital yet elusive. In finance, over 20% of decision-makers change roles annually. This volatility, mirrored across financial services turnover rates [1], renders static databases obsolete almost as soon as they are compiled. Existing commercial solutions aggregate large-scale contact lists behind expensive lookup services, creating monopolistic barriers that limit smaller firms' ability to compete.

Email address generation is complex. While a few syntactic patterns (e.g. *first.last@domain*) dominate, the long-tail of rare templates, coupled with shared infrastructure and non-standard naming, resists naïve automation [2]. A robust solution must therefore learn from historical records to rank the most plausible template among hundreds of candidates. Crucially, template ranking itself must operate within sub-50ms latency, leaving headroom for in-house verification and enrichment in future iterations.

1.1. Review of Existing Work

Research on email prediction falls into three main strands, each offering partial solutions but facing critical barriers to real world deployment.

Early work treated email prediction as pattern discovery. Scrupp's analysis shows organizations rely on 6-8 templates covering 90% of addresses [2], suggesting systematic enumeration could suffice. Algorithms such as Apriori [3] and FP-Growth [4] identify frequent structures, while sequential mining methods like TRuleGrowth [5] capture ordering rules. These outputs are interpretable but struggle with noise and runtime explosion at low support thresholds.

Neural text generation has been explored with limited success. Shao et al. [6] found that even 20B parameter models recover only ~3% of email addresses through learned associations. This highlights weaknesses in learning rare, rule-driven structures from distributional data. Character-level networks [7] and sequence-to-sequence architectures [8] perform better on structured tasks, while neural template induction [9] could learn email formats. However, none have been applied specifically to email prediction or show deployment-ready latency.

Supervised template selection reframes the task as classification. Gradient boosting methods such as LightGBM and CatBoost [10] [11] excel at high-dimensional sparse features, while SVMs (Support Vector Machines) demonstrate effectiveness for text classification [12]. These models outperform generative approaches but depend heavily on labelled data and often treat templates as static classes, ignoring their evolution.

Across these strands, two issues persist. Evaluation typically occurs on datasets that lack real world inconsistencies, and prior work tends to separate template discovery from template selection, despite the fact that real-world accuracy depends on their joint optimisation.

1.2. Contribution of this Study

These challenges reinforce barriers in capital markets, limiting smaller firms' access to investors. This study reframes email prediction as a ranking task over candidate templates, supported by lightweight verification, yielding a deployable system with state-of-the-art accuracy and sub-50 ms inference. The approach shifts the paradigm from static, paywalled databases to dynamic, data-driven infrastructure for investor access. This study therefore tests the hypothesis that joint template discovery and learning-to-rank outperform pattern mining, flat classification, and LLM baselines. Ranking is preferable because classes are imbalanced and candidate sets are grouped by firm, which pairwise objectives optimise directly. Domain inference or large-scale enrichment is out of scope for this study.

2. Methodology

2.1. Data Exploration

This study utilizes AIP's proprietary database containing over 300,000 individual contacts across ~150,000 investment groups, segmented into Limited Partner (LP) and General Partner (GP) investors. Each record includes 20+ categorical fields with *investor*, *firm*, and *email* forming the core predictive triad. Both datasets exhibit high investor diversity (~85–90% unique names), moderate firm consolidation (~15% unique), and substantial email sparsity.

a) Missingness

Email fields showed 40–50% missingness across both LP and GP datasets, while investor and firm fields had near-complete coverage (Appendix A). Analysis of missingness patterns revealed no cascading dependencies between fields, indicating that missingness was largely unstructured. As such, no imputation was attempted as recovering email addresses from auxiliary fields would have risked introducing bias.

b) Duplicates

Duplicate detection identified 6,642 LP and 11,868 GP records with matching investor names and contact details (email or LinkedIn). These were retained deliberately, as repeated records provide valuable training signal for modelling recurring name-to-template mappings across firms, outweighing the risks of overrepresentation.

c) Domain Analysis

Domain analysis revealed two major sources of complexity.

(i) Domain Reuse Across Fims

Personal domains (e.g., *gmail.com*) allow arbitrary local parts, while shared corporate domains (e.g., *lpl.com*) often serve multiple subsidiaries. To capture this, a binary *is_shared_infra* feature was engineered.

(ii) Firms Using Multiple Domains

While most firms rely on one or two domains, outliers (e.g., Equitable Advisors with 20+) highlight decentralised IT administration. This motivated the *firm_is_multi_domain* feature, which later proved important in understanding template ambiguity.

d) Cleaning Steps Taken

A targeted cleaning pipeline standardized key fields (Appendix B), removed invalid records, and ensured structural validity of emails while preserving legitimate template variation. After cleaning, the dataset contained ~92,000 investors (down from ~170,000) and 15,125 unique firms (from 24,133).

2.2. Pattern Mining

The next stage systematically identified syntactic templates in investor email addresses. Email local-parts were tokenised into structured sequences capturing positional and semantic information. This avoided superficial matches that obscure genuine structural differences arising from cultural naming, separators, or abbreviations. The resulting template set both quantifies format diversity for feature engineering and defines the candidate pool from which the ranking model selects.

a) Email Tokenisation

Initial tokenization was performed on the LP dataset per project specifications, with GP records incorporated in later iterations without additional analysis. This expansion introduced only edge-case templates without altering the core rule set. The tokenizer encoded constant tokens and name variants, capturing cultural conventions such as multiple last names. Nickname and abbreviation lookups normalised common variants, while NFKD decomposition – a standard Unicode normalization form that decomposes multibyte Unicode and reformulates into compatible ASCII byte or bytes – handled accented characters (*ä* to *ae*). Unrecognised variants were marked as UNK and excluded. Of 81,000 sequences, 6,683 were flagged as UNK, primarily reflecting non-standard abbreviations.

Template mining revealed 404 unique templates with the expected Pareto distribution. Most templates covered 0-5% of cases while a small number of high-frequency templates dominated coverage.

b) Template Rule Mining

To extract statistically significant structures, sequential rule mining was applied to tokenised sequences. Traditional FP-Tree methods [4] identify frequent subsequences but require costly post processing. TRuleGrowth [13], implemented via the SPMF library [14], directly mines sequential rules with support and confidence by incrementally extending frequent token sequences, making it well suited to generating deployable template statistics. High confidence patterns include forms such as *first.last* (97% confidence), as well as lower frequency variants incorporating initials, middle names, or multiple surnames.

These mined rules provided explicit features – support, confidence and in mined rules – for integration with the prediction model.

Rule mining thresholds were set to be as permissive as possible. This was done to maximise statistics for engineered features. Despite this, the Pareto coverage of templates meant that the miner returned only 12 rules. A full threshold sweep was not performed due to this.

c) Template Exploratory Analysis

Analysis confirmed limited template diversity across firms. The majority of firms (median = 1) employ a single syntactic template for email local-parts, with 75% using two or fewer templates. A small number of outliers use up to 21 distinct formats (mean = 1.32, std = 0.88), creating a long-tail distribution typical of organizational email practices [2].

(i) Name Structure Complexity

Feature flags were engineered for middle names, multiple name components, and structural variations to assess relationships with template diversity. While no strong correlations emerged, weak signals were found for names requiring NFKD normalization and multiple surnames, supporting their inclusion in the feature set.

(ii) Firm Size and Template Diversity

Firm size was analysed under the hypothesis that larger organizations adopt more complex templates to distinguish investors with similar names. While the majority of firms use 1-2 templates regardless of size, some larger organizations exhibit increased template diversity (Figure 1). However, the relationship shows high variance, indicating that firm size is only one factor influencing template complexity.

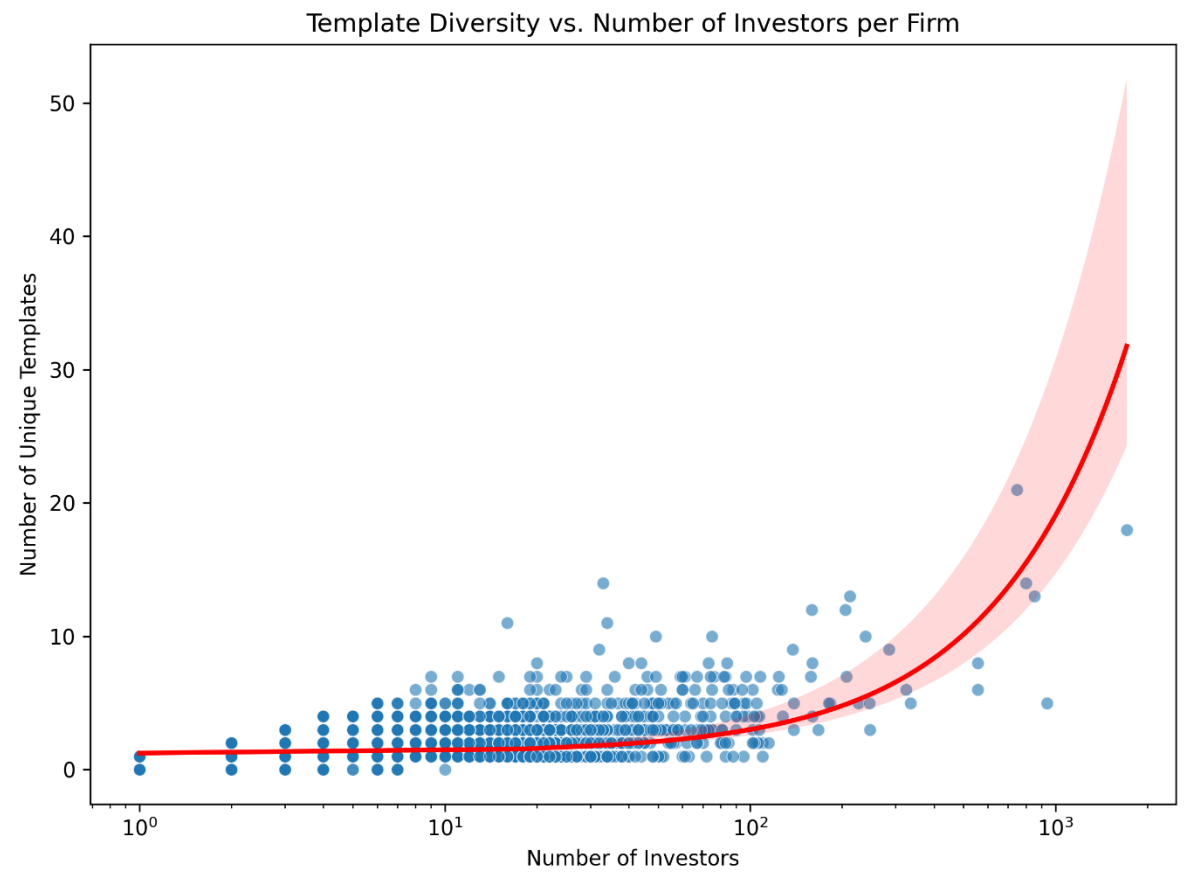


Figure 1: Investor Count vs. Template Diversity Scatter Plot

d) Feature Engineering

Pattern mining produced 404 structural templates, stored in SQL with unique identifiers for inference. Each investor–firm pair was represented as a feature matrix with each row representing a candidate template, enriched with three feature families. Structural metrics such as template support and name complexity flags, mined-rule statistics (support, confidence) from TRuleGrowth, and context features including infrastructure indicators, name–template clashes, and firm-level usage.

Ablation showed that context features drove performance. Removing them cut Accuracy@1 by ~50%, while structural and mined-rule features had minor impact (Appendix C).

Template selection was cast as pairwise ranking, scoring correct templates higher than alternatives within each firm. This formulation directly optimised ordering and delivered substantial accuracy gains over flat classification.

2.3. Prediction Model

a) Model Architecture

The prediction pipeline (Appendix F) comprises two gradient-boosted decision tree models. One model is dedicated to *standard name* inputs, while the second is specialised for *complex names* inputs involving additional structural features (e.g., multiple given names, middle names, or special character handling). The dual pathway design was adopted because ~95% of given investor names followed a simple first-last name format. Training a single unified model diluted predictive signal by over weighting rare, complex cases. By isolating standard names, the model can focus on standard candidate templates more precisely, while a specialised model handles structurally complex but low frequency cases.

(i) Feature Construction Stage

Feature construction precedes model inference, extracting structural, statistical, and mined-rule features for each candidate template paired with the investor-firm query. A deterministic rule-based assignment directs inputs to either the standard or complex name pathway – with each pathway constructing its own feature matrix from its respective candidate template set.

(ii) Inference Model

Email template prediction is framed as a ranking problem, where the objective is to order candidate templates for each investor-firm query. This aligns with learning-to-rank methodologies for information retrieval, where pairwise objectives optimize directly for ranking quality rather than classification accuracy.

CatBoost with the YetiRank objective was selected as the final inference model. YetiRank is a pairwise learning-to-rank objective that directly optimises the relative ordering of candidates by comparing pairs, rather than independent classes. Compared to LightGBM baselines, CatBoost offered two advantages. Stronger handling of sparse categorical features [10], which arise in firm and template metadata, and gradient-aware optimisation - using residual gradients to weight and prioritise correction of misordered pairs – that improves discrimination between similar templates. Empirical evaluation confirmed this with YetiRank producing higher accuracy and greater stability against a LightGBM baseline.

This combination of mined structural features with a ranking formulation represents a key methodological contribution of the study, enabling the system to model both dominant templates and rare, structurally complex edge cases within a unified framework.

(iii) Email Resolver

The resolver transforms ranked templates into full email predictions by combining the top-3 local part candidates with the resolved firm domain. This top-k strategy balances accuracy and robustness as in practice, outreach systems can attempt multiple candidates.

Domain resolution itself is performed through a hierarchical mapping strategy designed to preserve both precision and coverage. Exact matches against a canonical firm-to-domain table maximises precision, while a fuzzy matching fallback ensures coverage for variant firm names without discarding otherwise valid records.

b) Training

For computational feasibility, engineered features were stored in SQL and trained on high-memory cloud environments, allowing full-batch model fitting. To stabilise early training, the candidate set was temporarily pruned from 404 to 294 templates containing only essential tokens.

Evaluation was conducted using ranking-oriented metrics - Accuracy@1 (top rank correctness), Recall@3 (top three coverage), and Mean Reciprocal Rank (MRR), which penalizes deeper placements of correct templates.

Baseline results reached ~38% Accuracy@1 and ~67% Recall@3 on both training and validation sets. Although modest, these results established a consistent ranking baseline and highlighted the need for strategies to improve recall in sparse or high diversity domains, motivating the introduction of synthetic augmentation and segmentation approaches.

c) Stratified Error Analysis

(i) Domain-Level Performance Breakdown

Stratification by email domain revealed a striking bimodal distribution with approximately 1,250 domains achieving perfect (100%) Accuracy@1, while

~2,750 domains achieved none, with very few domains performing at intermediate levels (Figure 2). This pattern suggests that the model tends to generalises near-perfectly when domain history is sufficient and template usage is consistent, but fails completely when training signals are sparse or highly diverse. Low-performing domains were characterised by very small sample sizes (median = 5 investors), high intra-domain template diversity (mean diversity ratio = 0.40), or use of shared infrastructure domains (e.g. gmail.com). Together these factors eroded the discriminative value of firm level statistics.

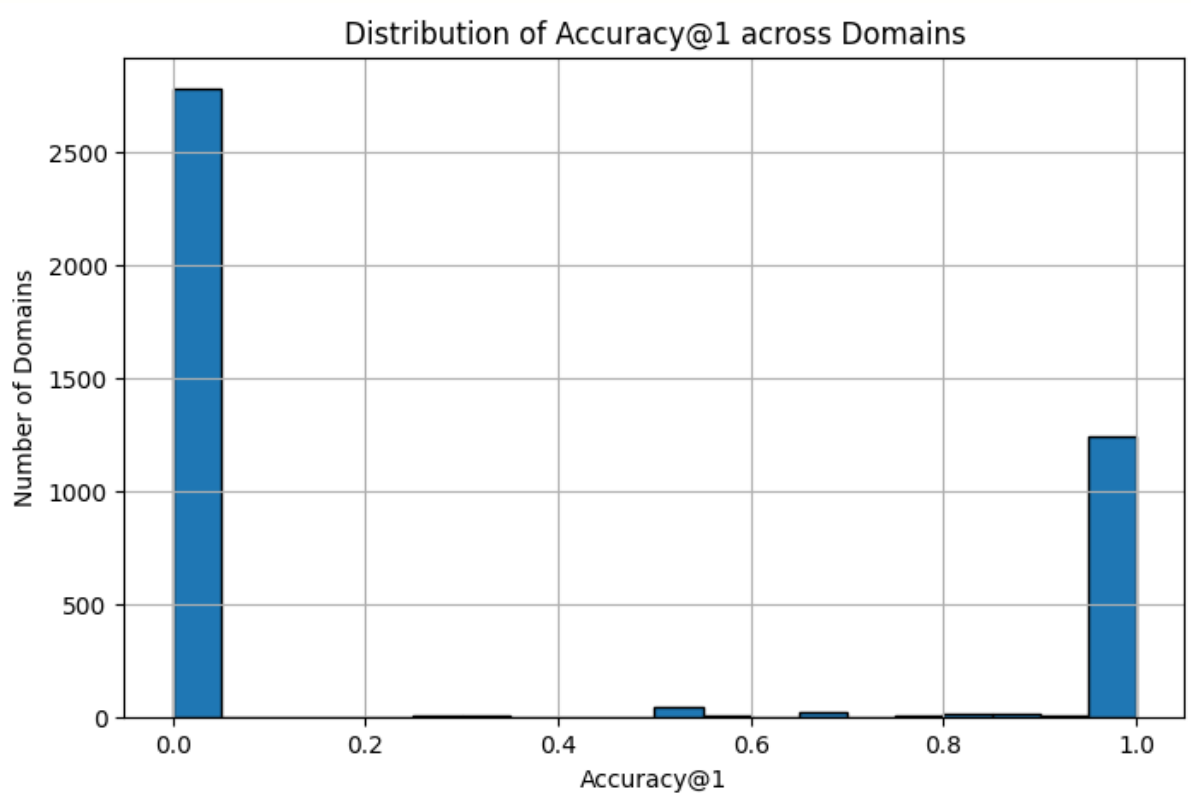


Figure 2: Accuracy@1 Across Domains

(ii) Template Diversity and Failure Correlation

Template diversity was associated with higher failure rates (Figure 3). Firms using multiple distinct formats often experienced reduced Accuracy@1, since more candidates became plausible and template support fell. Notably, several high-diversity domains still achieved good performance, indicating that diversity alone is not fatal.

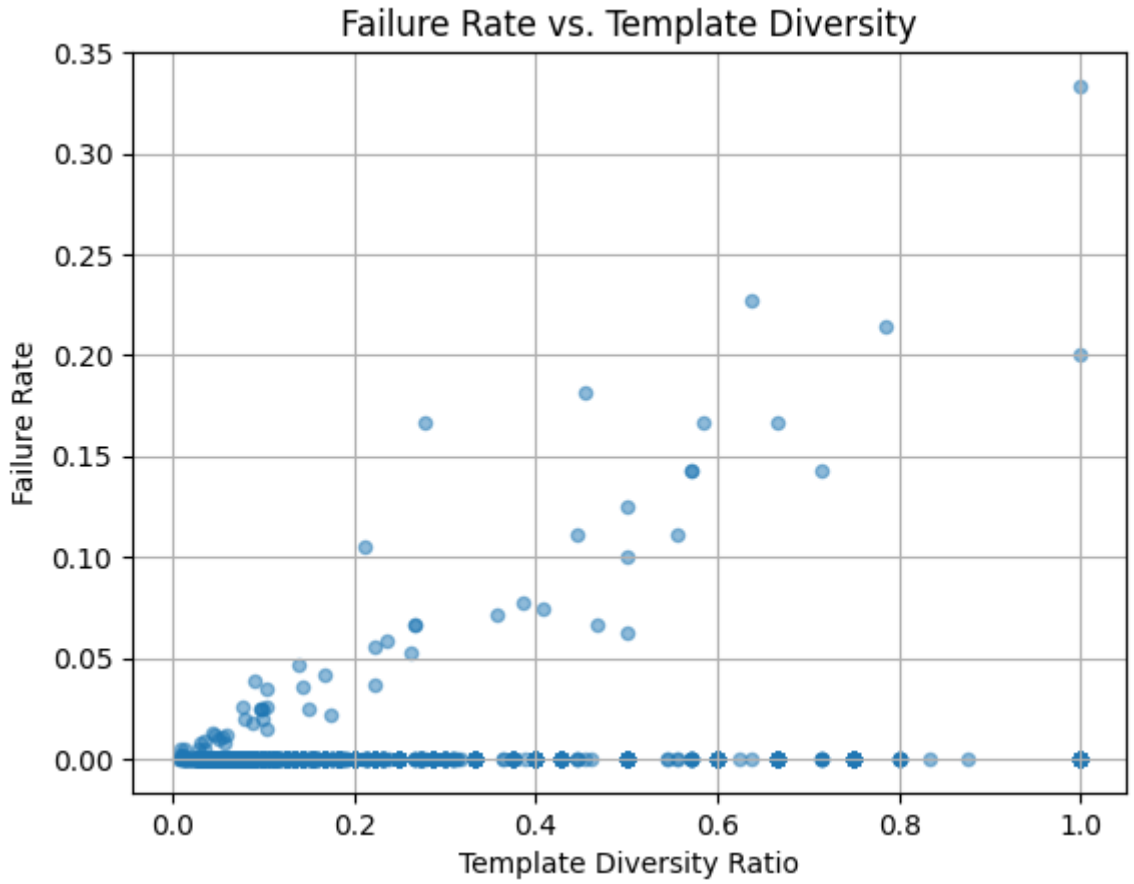


Figure 3: Template Diversity vs. Failure Rate Scatter

(iii) Investor Count Band Analysis

Performance was stratified by firm size using investor count bands to assess sample size impact. Results revealed a strong positive correlation between investor count and ranking accuracy, with peak performance (~67% Accuracy@1) achieved for firms with 201-300 investors (Figure 4).

Small firms (<20 investors) consistently underperformed (~30% accuracy), while the surprising drop at 301-400 investors suggests potential data quality issues or outlier effects in the largest firms. This pattern indicates the model benefits disproportionately from abundant firm-level examples, highlighting potential overfitting to well-represented firms and the value of data augmentation strategies for small-investor firms.

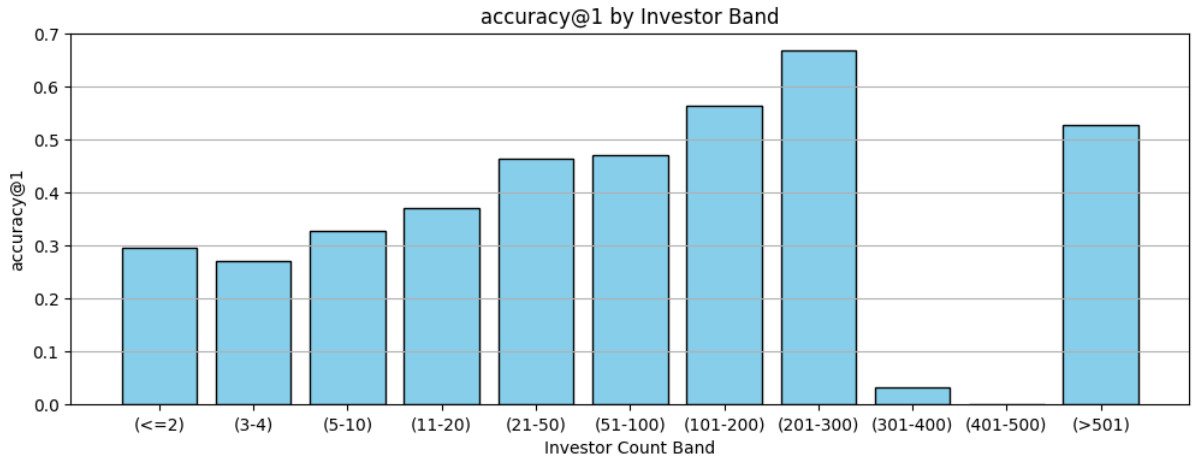


Figure 4: Accuracy@1 Banded by Firm Investor Count

d) Synthetic Padding Strategy

The error analysis highlighted that systematic failures occurred in two regimes. Small firms with <20 investors and high-diversity firms where multiple plausible templates diluted per-template support. To mitigate this imbalance, a synthetic padding strategy was developed to augment training data for underrepresented firms while preserving global feature distributions.

(i) Sampling Process and Drift Minimisation

For each eligible firm, a statistical was profile constructed. This consisted of template selection probabilities (per firm), and conditional probabilities ($(P(flag \mid firm, template)$) for structural flags per template. Synthetic investors were generated from these profiles, padding low-investor firms with 10–20 additional samples. Ideally, a low investor firm threshold would be set at 200 investors and all firms falling below this would be padded to that point, aligning with the evidence in Figure 4. However, due to memory restraints thresholds were largely selected for computational feasibility rather than achieving ideal distributions. Random placeholder names were assigned only for record tracking. Distributional integrity was validated by recomputing firm profiles after augmentation and comparing them against the original profiles. Unit tests confirmed that for basic cases the drift between real and augmented sets remained within tolerance, ensuring synthetic padding preserved the underlying statistical structure. Future work should investigate these drifts rigorously to ensure sampling is faithful to global statistics.

Training incorporated both real and synthetic records, while validation and test splits were drawn exclusively from real data to ensure fairness. A 60/20/20 split was adopted, where 20% of investors were reserved for validation and 20% for held-out testing (with the remaining 60% plus generated synthetic investors comprised training data). This proportion balances the need for robust evaluation metrics while maximising training data availability. To avoid bias, validation and test sets were constructed using a stratified domain sampling, ensuring that template diversity and firm representation matched the overall dataset distribution. This design prevents data leakage from synthetic augmentation and provides a fair estimate of generalisation performance.

(ii) Impact on Performance

Augmentation produced dramatic improvements. Accuracy@1 rose from 38% to 85% and Recall@3 from 67% to 97% on validation. Importantly, these gains were achieved on the full 404-template candidate set (versus 294 in the baseline). This demonstrates that synthetic padding not only expanded coverage but also enabled the model to resolve correct templates under more challenging conditions.

(iii) Placebo Testing for Synthetic Testing

To ensure observed gains from padding were not due to trivial duplication, a placebo augmentation, where investors were randomly duplicated, were tested alongside a modest synthetic padding campaign. Both datasets were of the same size and used the same test set. Placebo padding degraded performance dramatically (Accuracy@1 50% vs. ~92%), confirming synthetic sampling preserves signal.

These results suggest that statistically faithful augmentation can compensate for long-tail sparsity. Considering also that synthetic records were only used in training, while validation and test sets contained exclusively real data, gains are unlikely to be due to overfitting or distributional artifacts

e) Benchmarking

(i) LLM Baseline Comparison

A zero-shot experiment using OpenAI's GPT-4 assessed LLM-based inference feasibility without feature engineering. Following Shao et al.'s [6] findings that

large language models recover only small fractions of email addresses from name-firm inputs, 500 LP investors were provided to GPT-4 via structured prompts requesting three most likely professional email addresses.

Zero-shot performance was poor (Accuracy@1 = 12%, Recall@3 = 19%) with negligible improvement when historical template hints were added. Latency (~3 seconds per query) and cost considerations rendered this approach unsuitable for production, confirming that specific structured learning is substantially more affordable, performant, efficient, and explainable.

(ii) Model Architecture Comparison

CatBoost with YetiRank objective outperformed the LightGBM baseline on identical feature matrices. CatBoost achieved Accuracy@1 = 90.6%, Recall@3 = 99.7%, MRR = 0.954 compared to LightGBM's Accuracy@1 = 86.2%, Recall@3 = 99.3%, MRR = 0.927. These gains reflect CatBoost's gradient-aware pairwise ranking, enhanced categorical sparsity handling, and superior generalization to rare templates.

f) Segmented Modelling by Name Complexity

Error analysis revealed that investors with complex name structures (multiple names, nicknames, special characters) were disproportionately represented among prediction failures (Table 1). These cases generally mapped to specialised templates with lower support, making unified modelling across all investors more difficult.

Feature	Fail Rate (%)	Overall Rate (%)
has_nfkd_normalized	2.30	1.53
has_german_char	0.00	0.52
has_nickname	9.20	24.55
has_multiple_first_names	14.94	0.60
has_middle_name	24.14	1.67

has_multiple_middle_names	3.45	0.17
has_multiple_last_names	13.79	2.05

Table 1: Feature Fail/Overall Rate Comparison

To mitigate structural complexity, the dataset was split into standard names (~95%, single first/last) and complex names (any additional elements). This reduced template diversity per model and enabled targeted augmentation allowing increased synthetic padding for the smaller complex subset and a refined candidate space for standard names.

(i) Impact on Performance

Both subsets were trained with identical feature engineering under the LambdaRank LightGBM objective. The standard-name model achieved ~86% Accuracy@1 and ~99% Recall@3, while the complex-name model achieved ~85% Accuracy@1 and ~96% Recall@3. Weighted by dataset composition, segmentation produced modest aggregate improvements over the unified baseline (+0.96% Accuracy@1, +0.53% MRR; Table 2). While global gains appear small, in practice the effect is substantial considering the fact that complex cases represent only ~5% of the dataset. Within that subset, however, segmentation measurably reduced failure concentration, indicating that structurally challenging names benefit from specialized modelling.

Metric	**Unified Model (%)**	**Segmented (Weighted) (%)**	**Improvement (%)**
Accuracy@1	85.00	85.96	+0.96
Recall@3	99.40	99.46	+0.06
MRR	92.06	92.59	+0.53

Table 2: Unified vs. Segmented Model Performance Comparison

Segmentation therefore functioned less as a global performance booster and more as a targeted robustness measure, ensuring minority of investors with complex names were not systematically disadvantaged. This complements the

earlier synthetic padding strategy. Whereas padding mitigates sparsity in long tail firms, segmenting reduces structural ambiguity in template-name interactions.

(ii) Cultural Naming Bias Assessment

Test sets consisting of standard and complex investors respectively were evaluated on a unified model to assess cultural naming bias. Accuracy@1 fell slightly (–2.2%) though Recall@3 remained stable, confirming that structural complexity modestly raises error rates, but predictions remain reliable. Segmentation reduces these disparities, protecting minority name groups.

g) Hyperparameter Tuning and Final Model Selection

Systematic hyperparameter optimization was conducted using Optuna (50 trials each) for both LightGBM and CatBoost on standard and complex datasets, targeting Recall@3 maximization with early stopping to prevent overfitting.

CatBoost consistently outperformed LightGBM across both subsets (Table 3). For standard names, CatBoost achieved 91.47% Accuracy@1 and 99.78% Recall@3 versus LightGBM's 90.23% and 99.64% respectively.

Model	Accuracy@1 (%)	Recall@3 (%)	MRR (%)
LightGBM (standard)	90.23	99.64	94.93
CatBoost (standard)	91.47	99.78	95.91
LightGBM (complex)	79.53	95.67	87.04
CatBoost (complex)	83.38	97.63	89.99

Table 3: Hyperparameter Tuning Results

Performance gains were more pronounced for complex names (+3.9% Accuracy@1, +2.0% Recall@3), indicating CatBoost's gradient-aware ranking and categorical handling provide particular advantages for high-noise, low-frequency cases.

h) Testing with C++ Inference Engine

End-to-end evaluation used the C++ inference engine to replicate production conditions. At the time only LightGBM was supported, so tuned LightGBM models were deployed for both name subsets.

Training validation achieved ~92% Accuracy@1 (standard) and ~77% (complex), with ~95% Recall@3 for both. End-to-end testing on combined test sets yielded 74% Accuracy@1 and 94% Recall@3, with approximately 1,200 of 22,000 predictions failing to rank the correct template in the top three.

(i) Name Characteristics in Failing Cases

Comparative analysis revealed that failing predictions were disproportionately associated with structural complexity. Multi-domain firms (+14.08% vs. non-failing cases) and investors with nicknames (+9.03%) or multiple last names (+6.56%) were overrepresented among failures.

A logistic regression model trained on Boolean features confirmed these patterns, with multiple last names (coefficient 2.002), NFKD normalization (1.399), and middle names (1.152) as strongest failure predictors. This supports the segmentation strategy and suggests potential Python to C++ implementation discrepancies for normalized names (Table 4).

Feature	**Coefficient**
has_multiple_last_names	2.002
has_nfkd_normalized	1.399
has_middle_name	1.152
has_multiple_first_names	0.865
firm_is_multi_domain	0.682
has_nickname	0.531
has_multiple_middle_names	-0.057

has_german_char	-0.207
is_shared_infra	-0.376

Table 4: Failure Signals Identified through Logistic Regression

(ii) Firm Level Statistics in Failing Cases

The 1,200 failing cases involved 329 firms, showed some template diversity (median diversity ratio 0.286) combined with small investor counts (median 11 investors per firm). This forces the model to rank among many plausible templates with minimal supporting evidence, aligning with earlier findings that multi-domain firms and complex names drive prediction failures.

(iii) Template Statistics in Failing Cases

Only 91 templates appeared in the failing set. Most failing templates had low coverage (<5%) and support counts (median: 19), confirming that limited training exposure correlates strongly with prediction failure (Figure 5). However, several high-support templates also exhibited some failure rates, indicating that familiarity alone doesn't guarantee correct resolution.

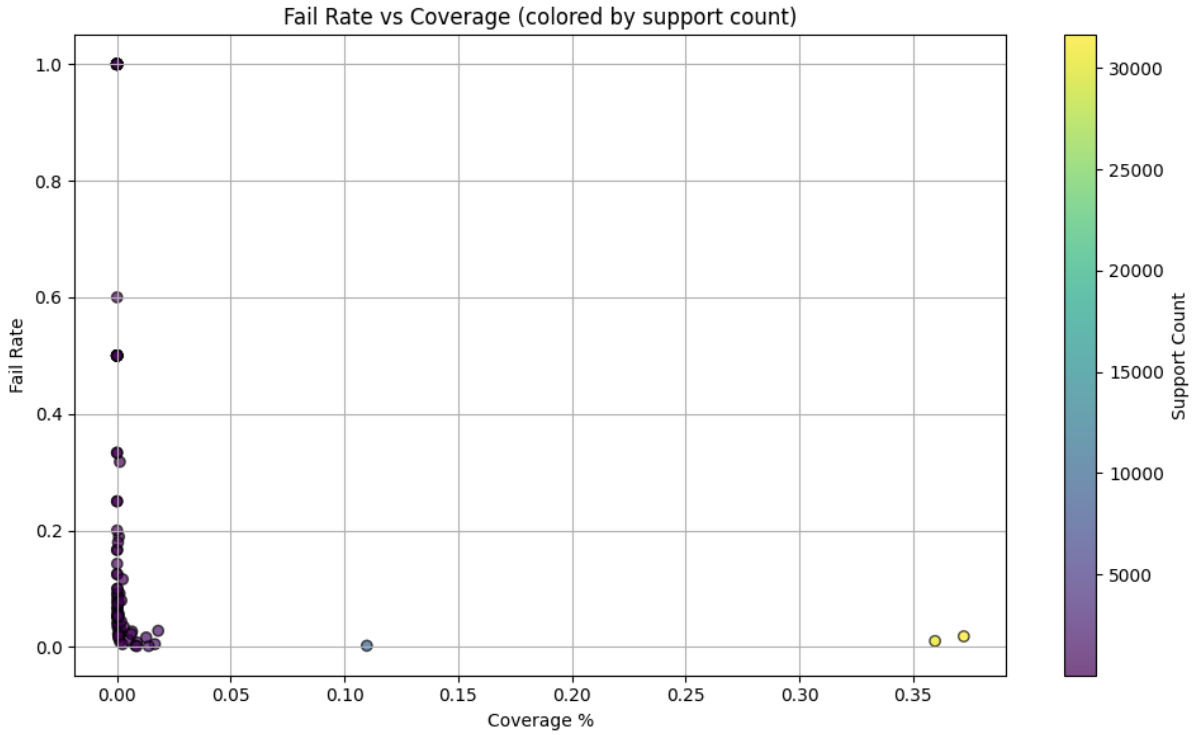


Figure 5: Fail Rate vs Coverage Scatter:

i) GP Data Brought In

Incorporating GP data added ~170,000 investors and ~200 mostly low-support templates, expanding coverage to ~26,000 additional unique firms. Logistic regression confirmed amplified failure signals, with coefficients for multiple first and last names increasing sharply.

Infrastructure effects also reversed. Multi-domain firms shifted from positive to negative predictors, while shared infrastructure became positive, reflecting GP practices where executives often use separate domains.

Template diversity and normalization tokens (NFKD, transliterations) continued to drive errors, motivating default normalization. NFKD and related tokens were pruned from the template set, reducing low-support variants and simplifying ranking.

Despite the distribution shift, overall metrics remained stable, showing that synthetic padding improved robustness rather than overfitting to LP-specific patterns.

3. Results

3.1. Final Training

Validation on the GP dataset showed the baseline model performed well on standard names (Accuracy@1 = 0.9336, Recall@3 = 0.9984, MRR = 0.9652) but much worse on complex ones (Accuracy@1 = 0.7609, Recall@3 = 0.9729, MRR = 0.8664). Pruning redundant normalization tokens produced clear gains, raising complex Accuracy@1 to 0.8419 and Recall@3 to 0.9850, while standard names improved slightly (Accuracy@1 = 0.9383). Analysis of 2,597 failed predictions showed that 40.7% occurred because the model ranked ultra-rare templates (support = 1) in place of highly supported ones (>50,000). Removing these ultra-rare templates actually marginally degraded performance across the two models.

Model Variant	Set	Accuracy@1	Recall@3
Baseline	Standard	0.9336	0.9984

	Complex	0.7609	0.9729
Token Pruned	Standard	0.9383	0.9982
	Complex	0.8419	0.9850
Ultra-rare Pruned	Standard	0.9374	0.9984
	Complex	0.8407	0.9860

Table 5: Final Test Results

3.2. Test Results

Evaluation on the held-out test set confirmed that pruning and segmentation generalised beyond validation. The final CatBoost models, trained separately on standard and complex subsets with fully pruned templates, achieved Accuracy@1 = 0.9310 and Recall@3 = 0.9936 in test. Thus, the correct template was ranked first in over 93% of cases and appeared within the top three in nearly all others (<1% misses). These gains were driven primarily by pruning redundant normalization templates, which simplified the candidate space without sacrificing coverage. By contrast, removing ultra-rare template yielded marginal negative metric changes (Table 5), and error analysis showed that most ultra-rare templates were predicted correctly – with only 3 of 91 failing templates being one of the 183 ultra-rare templates. This indicates that pruning is most beneficial when collapsing redundant variants rather than discarding low-frequency templates outright. Segmentation by identified failure signals, rather than frequency signals alone, proved more effective at reducing residual errors and refining candidate space, particularly for complex names.

a) Comparison to Benchmark and Baseline

The initial LightGBM flat classification baseline on unaugmented LP data achieved only ~48% Accuracy@1 and ~85% Recall@3, showing that flat classification was poorly suited to the imbalance. Reformulating the task as ranking, with template pruning and synthetic augmentation, more than doubled top-rank accuracy while pushing Recall@3 above 99%.

Against the strongest internal baseline, tuned LightGBM LambdaRank, CatBoost with YetiRank delivered consistently higher validation performance. On the complex subset CatBoost improved Accuracy@1 by nearly four points and Recall@3 by two, gains that carried through to the test set and established CatBoost as the final deployed model. These improvements confirm CatBoost's ability to better handle categorical sparsity and long-tail template distributions.

External benchmarks underscored this advantage. A zero-shot GPT-4 baseline achieved only 12% Accuracy@1 and 19% Recall@3 with ~3s query latency. By contrast, CatBoost exceeded 91% Accuracy@1 with sub-millisecond latency, confirming that large language models are ill-suited – as was noted in previous work [6] - to this structured ranking task, whereas lightweight gradient-boosted rankers provide both superior accuracy and deployability.

3.3. Cross Domain Generalisation

To evaluate generalisation, a test was constructed from firms entirely unseen during training. Performance remained stable (Accuracy@1 ~92%, Recall@3 ~99%) indicating the model is not over-fitted to firm specific distributions.

3.4. Feature Importance

Both CatBoost feature importance and SHAP analysis (Appendix G) highlighted firm-level coverage features (template is top in firm, template coverage within firm) as the strongest drivers of prediction. Confirming results from ablation that demonstrated how context features outweigh structural complexity or mined rules in terms of importance.

3.5. Latency and Deployment

Latency was assessed at three levels of the deployment stack. The C++ SDK achieved ~0.05ms per query over 60,000 runs, showing inference itself is near-instantaneous. Exposing the model through the C++ pybind interface increased latency only to ~0.12ms, confirming wrapper overheads were negligible. Deployment as a FastAPI REST service added network and serialization costs, averaging 3.83ms per request—well within the 50ms target for scalable use. Adding third-party verification (Hunter.io) raised latency to 854.82ms across five attempts, but in all cases the top-ranked prediction was confirmed deliverable with

scores above 90. This indicates the core predictor satisfies real-time requirements, while verification is better handled via in-house checks to avoid excessive delays.

4. Discussions

4.1. Challenges

Scale posed the primary challenge with the LP feature matrix alone exceeding 29 million rows with ~400 templates, making local training infeasible since gradient boosted models require full-batch fitting. Migration to cloud services resolved this at significant financial cost.

Extreme class imbalance was inherent - each investor has one correct template among hundreds of candidates, creating massive positive-to-negative ratios. Without careful weighting, models over-predicted positives for recall at precision's expense, particularly problematic for diverse firms with minimal historical data.

The long tail of low-support templates continuously expanded with new data, polluting the dataset and growing the search space without contributing meaningful signal.

4.2. Strengths

CatBoost with YetiRank achieved consistently high performance with >88% Accuracy@1 and ~95% Recall@3 across both name complexity segments. This represents significant improvements over LightGBM baselines and massive gains over LLM one-shot benchmarks.

Dataset segmentation by name complexity proved valuable despite the complex subset comprising only ~5% of data. The +0.96% Accuracy@1 improvement, while modest, was consistent and demonstrated that isolating structurally difficult cases enhances model behaviour.

Synthetic padding for low-investor firms produced dramatic gains, improving their Accuracy@1 by over 50 percentage points without degrading well-represented firm performance, indicating statistical alignment with real-world distributions.

These results establish a clear pathway forward, performance is constrained more by edge case representation and feature engineering quality than model

architecture choice. Given the absence of prior work in investor email template prediction under limited historical data and large candidate spaces, these results provide a strong baseline for both academic and applied contexts.

4.3. Weakness

Domain resolution presents a key limitation. The inference pipeline relies on canonical domain maps rather than ML approaches, degrading performance with firm name misspellings, format variations, or multi-domain contexts. Fuzzy matching partially mitigates this, but no reliable method exists to infer correct domains from user-supplied firm names without introducing false positives or latency.

High template diversity firms remain challenging even with accurate domain resolution, creating ambiguous prediction spaces that reduce Accuracy@1 scores.

Complex name handling shows persistent issues. Despite segmentation improvements, high-support templates following standard patterns still fail when name-structure flags don't correlate with actual email local-part structure. This creates false candidate space splits and increased model uncertainty.

The system over-relies on firm-specific signals. For unseen or sparse firms, missing or unreliable features could degrade ranking quality. This is particularly acute when name-structure flags alone cannot disambiguate templates.

4.4. Future Work

Firm-specific pattern models could capture template preferences from IT behaviour or regional norms, though this requires larger firm-level datasets or third-party integration.

Enriching inputs beyond name and firm (e.g., role, seniority, location) may disambiguate high-diversity or multi-domain cases.

Candidate set refinement through hierarchical clustering could further reduce confusion, while infrastructure-aware models should explicitly address the inverted LP/GP correlations observed.

In-house deliverability verification could mitigate reliance on third-party APIs, with lightweight MX/SMTP checks providing low-cost validation at scale.

While this study segmented only by name complexity, further segmentation by other failure signals - such as multi-domain firms or multiple-surname investors - may yield additional robustness by reducing template diversity within each subset.

Finally, an active learning loop, where predictions are continually verified and incorporated back into training, would allow the model to adapt dynamically as naming conventions and infrastructures evolve.

5. Conclusion

This project reframed investor email prediction as a ranking problem over templates with hierarchal domain resolution. Combining offline mining, a CatBoost ranker, and profile-based padding. It a achieved ~93% Accuracy@1 and ~99% Recall@3 and showed no noticeable degradation on unseen firms, all while maintaining CPI inference <50ms per query across a REST API service. Two problems remain, domain mapping for multi domain/shared-infrastructure firms, and production viable deliverability verification.

Overall, the system provides a proof of concept that is both technically feasible and practically usable, supporting the central hypothesis that joint template discovery and ranking-based machine learning significantly outperforms prior pattern-mining, flat classification, and LLM-based approaches on sparse investor datasets.

6. References

[1] A. B. Association, “Non-officer employee turnover returns to pre-COVID levels,” ABA Banking Journal, 08 09 2022. [Online]. Available: https://bankingjournal.aba.com/2022/09/non-officer-employee-turnover-returns-to-pre-covid-levels/. [Accessed 18 August 2025].

[2] Scrupp, “The Complete Guide to Email Patterns,” [Online]. Available: https://scrupp.com/blog/email-pattern. [Accessed 25 June 2025].

[3] R. I. R. S. A. Agrawal, “Mining association rules between sets of items in large databases,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, DC, 1993.

[4] J. Han, J. Pei and Y. Yin, “Mining frequent patterns without candidate generation,” *SIGMOD Record (ACM Special Interest Group on Management of Data),* vol. 29, no. 2, pp. 1-12, 2000.

[5] A. Abdelwahab and N. Youssef, “Performance evaluation of sequential rule mining algorithms,” *Applied Sciences,* vol. 12, no. 10, p. 5230, 2022.

[6] H. Shao, J. Huang, S. Zheng and K. Chang, “Quantifying association capabilities of large language models and its implications on privacy leakage,” in *Findings of the Association for Computational Linguistics: EACL 2024*, St. Julian’s, 2024.

[7] X. Zhang, J. Zhao and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, 2015.

[8] L. Dong and M. Lapata, “Language to logical form with neural attention,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, 2016.

[9] S. Wiseman, S. M. Shieber and A. M. Rush, “Learning neural templates for text generation,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, 2018.

[10] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush and A. Gulin, “CatBoost: Unbiased boosting with categorical features,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, 2018.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T. Y. Liu, “LightGBM: A highly efficient gradient boosting decision tree,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, 2017.

[12] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *Proceedings of the 10th European Conference on Machine Learning (ECML)*, Chemnitz, 1998.

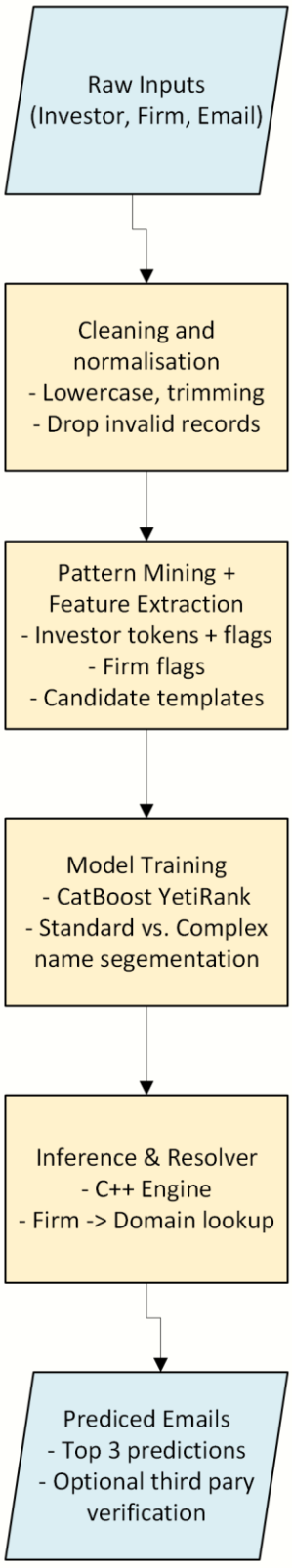
[13] P. Fournier-Viger, L. C.-W. Lin, B. Vo, T. Truong Chi, J. Zhang and H. B. Le, “A survey of itemset mining,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 7, no. 4, p. e1207, 2017.

[14] P. Fournier-Viger, “SPMF — An open-source data mining library,” Philippe Fournier Viger, 5 August 2025. [Online]. Available: https://www.philippe-fournier-viger.com/spmf/. [Accessed 15 August 2025].

7. Appendix A. Data Sheet

Feature	Missingness	Transformations/Usage
Investor Name	0%	Lower cased, multiple spaces collapsed. Used to construct context and structural features. Main predictive backbone.
Firm Name	0%	Lower cased, multiple spaces collapsed. Used to form template per firm statistics. Compute firm profiles for padding. Fuzzy matched and mapped for domain mapping. Derived infrastructure flags.
Email Address	~45%	Lower cased, basic regex checks. Used to derived candidate set through tokenisation. Formed template statistics and labels for predictions.

8. Appendix B. ETL Pipeline Diagram



9. Appendix C. Feature Ablation Results

Feature Set	Accuracy@1	Recall@3	MRR
Full Features	0.9252	0.9962	0.9603
Context Features Only	0.9223	0.9942	0.9582
Mined Features Only	0.0003	0.3671	0.2105
Structural Features Only	0.3668	0.8788	0.6198
Context Features Removed	0.3985	0.8788	0.6351
Mined Features Removed	0.9261	0.9588	0.9607
Structural Features Removed	0.9242	0.9955	0.9599

10. Appendix D. Reproducibility and CatBoost Params

Final Tuned CatBoost Params

Parameter	Value
loss_function	YetiRank
eval_metric	NDCG:top=3
random_seed	42
learning_rate	0.1328
depth	6
l2_leaf_reg	7.1425
random_strength	3.396
min_data_in_leaf	84
subsample	0.905
colsample_bylevel	0.511
grow_policy	LossGuide

Reproducibility

All experiments were run with a fixed random seed of 442. Training and validation splits were stratified to prevent leakage. Models were trained in full batches on GoogleColab VMs with >300GB CPU RAM.

11. Appendix E. API Specification

HEALTH

GET /

Responfd with JSON with field “message” and a statement confirming health. Error if failure.

PREDICT (No Verification)

POST /predict/catboost

Request body contains “name” and “firm” fields. Also optional fields “domain” and “top_k”.

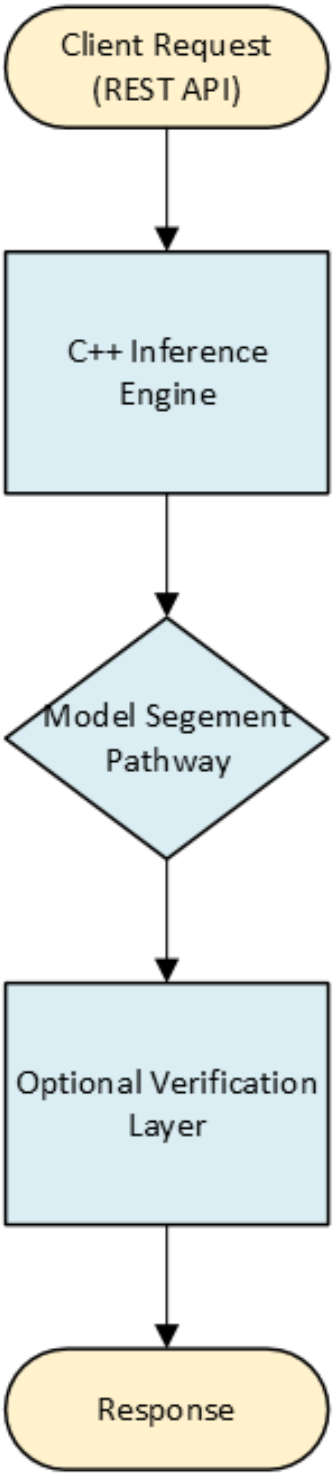
Response will contain an array of predictions with scores and predicted resolved emails.

PREDICT (With Verification)

POST /predict/catboost_verified

Same as previous.

12. Appendix F. Production Inference Pipeline



13. Appendix G. Feature Importance

Most Important Features	Importance	Top SHAP	Mean ABS SHAP
Template firm coverage %	37.656	Template firm coverage %	0.0784
Template is top template	10.895	Firm investor count	0.0587
Template support count	6.975	Firm is single template	0.0438
Template in firm templates	6.688	Template support count	0.0377
Template firm support count	4.740	Investor has nickname	0.0359

14. Appendix H. Glossary

LP (Limited Partner)

An institutional or individual investor who provides capital to private equity or venture funds but does not actively manage them.

GP (General Partner)

The fund manager responsible for deploying capital, managing portfolio companies, and generating returns for LPs.

LightGBM

A gradient boosting framework developed by Microsoft, optimised for efficiency and scalability on large datasets with high-dimensional features.

LambdaRank

A pairwise ranking objective used in gradient boosting that optimises the ordering of candidate items (rather than classification accuracy), particularly effective for information retrieval tasks.

CatBoost

A gradient boosting library from Yandex, designed for categorical features and sparse data, with strong out-of-the-box performance.

YetiRank

A proprietary CatBoost ranking objective that incorporates gradient-aware pairwise optimisation to improve ranking quality.

C++

A high-performance systems programming language widely used for computationally intensive applications, offering fine-grained control over memory and execution.

Python

A general-purpose programming language with extensive libraries for data science, machine learning, and scientific computing.

REST API

A web service architecture (Representational State Transfer) that exposes functionality through stateless HTTP endpoints, commonly used for scalable client-server communication.

Apriori

A classic algorithm for mining frequent itemsets and association rules in large databases, based on iterative candidate generation and support thresholds.

FP-Growth

A frequent pattern mining algorithm that uses a compressed prefix-tree (FP-Tree) to mine frequent itemsets without explicit candidate generation.

FP-Tree (Frequent Pattern Tree)

A compact data structure that represents transactions by common prefixes, enabling efficient mining of frequent patterns.

TRuleGrowth

A sequential rule mining algorithm that incrementally extends frequent sequences to produce rules with support and confidence, implemented in the SPMF library.

Neural text generation

A family of machine learning methods (often RNNs, LSTMs, or Transformers) that generate natural language sequences from learned representations.

Sequence-to-sequence architectures

Neural models (encoder–decoder frameworks, often with attention) that map input sequences to output sequences, widely used in machine translation and structured prediction.

Neural template induction

A neural approach to automatically learning structural templates (e.g., email formats) from examples, rather than relying on predefined rules.

SVM (Support Vector Machine)

A supervised learning algorithm that finds a decision boundary maximising the margin between classes, effective in high-dimensional feature spaces.

AIP

The sponsor of this project and provider of the proprietary dataset.

ASCII (American Standard Code for Information Interchange)

A standard character encoding that represents text using 7-bit codes. Email systems generally require ASCII-compatible addresses.

NFKD (Normalization Form KD)

A Unicode text normalisation method that decomposes characters into canonical equivalents, often used to standardise accented names.

UNK (Unknown Token)

A placeholder used in the tokenisation pipeline to represent unrecognised or out-of-vocabulary elements.

SPMF

An open-source Java data mining library implementing over 200 algorithms for pattern mining, sequential rule mining, and clustering.

Optuna

A Python-based framework for hyperparameter optimisation that uses efficient search strategies to tune machine learning models.

Supervised Learning

A machine learning paradigm where models are trained on labelled datasets, learning a mapping from inputs (features) to known outputs (labels). The model then applies this mapping to predict outputs for unseen inputs.

Classification

A supervised learning task where the goal is to assign each input to one of a set of discrete categories (classes). In this project, flat template prediction can be framed as classification, while ranking reframes it as ordering candidate templates.

Gradient-Boosted (Decision Trees)

An ensemble learning technique where multiple weak learners (decision trees) are trained sequentially, each correcting the errors of the previous ones. Popular implementations include LightGBM and CatBoost.

LLM (Large Language Model)

A neural network trained on massive text corpora to predict and generate human-like language. Examples include GPT-4. In this project, LLMs were tested as baselines for email prediction.

FastAPI

A modern Python web framework for building REST APIs with automatic documentation, type validation, and high performance asynchronous execution.

SNMP/MX Checks

Simple Network Management Protocol (SNMP) and Mail Exchange (MX) record checks are network-level methods for verifying whether an email server is valid and configured to receive mail. Lightweight MX/SMTP checks can serve as in-house deliverability verification.

Google Colab

A cloud-based Jupyter notebook environment provided by Google that allows free or paid GPU/TPU access for Python coding, data analysis, and machine learning experiments.

VM (Virtual Machine)

A software-based emulation of a physical computer that runs its own operating system and applications. VMs are commonly used for isolated testing, cloud computing, and scalable ML training environments.