Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Machine Learning versus Deep Learning for Public Financial Report Classification and Popularity Prediction: A Feasibility Study on Ghana's Controller and Accountant-General's Department

Emmanuel Osei-Dwomoh
Controller and Accountant General's Department, Kumasi, Ghana
Gabriel Osei Forkuo
gabriel.forkuo@unitbv.ro

Transilvania University of Brasov: Universitatea Transilvania din Brasov

Research Article

Keywords: automated classification tools, class imbalance, download popularity prediction, hyperparameter search, small-sample machine learning, eXtreme Gradient Boosting

Posted Date: August 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10815711/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

Government agencies increasingly publish financial reports online, but small public-sector portals rarely have the labeled data needed to test automated classification tools. This study evaluates the feasibility of these tools using a corpus of 131 official reports from Ghana's Controller and Accountant-General's Department (CAGD). We compare classical machine learning models (Logistic Regression, Random Forest, and eXtreme Gradient Boosting [XGBoost] with Synthetic Minority Over-sampling Technique [SMOTE]) and character-level deep learning models (bidirectional Long Short-Term Memory [LSTM] and Text Convolutional Neural Network [TextCNN]) across three tasks: category prediction, binary popularity classification, and log-download forecasting. Models were evaluated using a stratified 70/15/15 train, validation, and test split, randomized hyperparameter search, and a final fine-tuning phase. The fine-tuned XGBoost classifier achieved 1.00 accuracy on the five-class category task. Logistic Regression reached 0.75 accuracy (Area Under the Receiver Operating Characteristic curve [ROC-AUC] of 0.80) on the binary popularity task, and Random Forest reached an R-squared of 0.85 on log downloads. An ablation study removing twenty-five keyword features dropped category accuracy from 1.00 to 0.80, a stable drop across five randomized data splits. This indicates keyword features inflated initial results, though remaining performance confirms the models learn non-tautological structure. This work serves as an honest feasibility check, showing that thorough feature ablation is essential before deploying these models on live government portals.

1. Introduction

1.1 Background

Public financial management agencies around the world are under growing pressure to publish their reports, budgets, and audit findings online, both as a matter of transparency and as a practical way to reach citizens, auditors, and development partners who cannot visit a government office in person (OECD, 2025). Ghana’s Controller and Accountant-General’s Department (CAGD) is one such agency. It publishes a steady stream of documents, including chart-of-accounts editions, annual and quarterly financial statements, payroll circulars, and International Public Sector Accounting Standards (IPSAS) materials, through its website and through archived snapshots of earlier versions of that site. This kind of digital public financial management effort is part of a broader shift happening across Ghana and other developing economies, where digitizing financial records and reporting is seen as central to improving accountability and reducing opportunities for mismanagement (Osei-Dwomoh & Forkuo, 2025). The same shift is visible in other kinds of African institutional publishing, where artificial intelligence tools are increasingly used to organize, tag, and manage growing volumes of documents once handled entirely by hand (Salani & Tapfuma, 2025), and where the quality and traceability of published records is itself becoming a subject of formal study as institutions adopt more information and communication technology (Wartini-Twardowska & Twardowska, 2025).

A practical problem follows from this growth in online publishing. As the number of documents on a government portal increases, so does the burden of organizing them, tagging them correctly, and understanding which kinds of reports actually reach and engage the public. Manually reviewing every new upload to assign it the right category, or to judge whether it is likely to be widely downloaded, does not scale well once a portal has hundreds or thousands of documents spanning several years and several report types. Automated classification and popularity prediction, built directly from a report's own metadata, title, and description, offer one way to support that workload, provided the underlying models can be trusted and their limitations are clearly understood (Jiménez-Gómez et al., 2025).

At the same time, there is a real risk in applying machine learning or deep learning tools to a dataset like this one without first checking whether they are appropriate for its size and structure. Public-sector document archives are often small by machine learning standards, sometimes only a few hundred records, and the categories within them are rarely balanced. A report type that is published once a year will naturally have far fewer historical examples than one published every month. Classical guidance on learning from imbalanced and small samples warns that models trained under these conditions can look far more accurate than they really are, either because a handful of rare classes get folded into common ones, or because engineered features happen to leak the very label the model is supposed to be predicting (Little & Rubin, 2019; Van Buuren, 2018). This study takes that warning seriously, and one of its central goals is to report results honestly, including the parts that look impressive on paper but should not be trusted at face value.

1.2 Aim, Objectives, and Research Questions

This study aimed to compare classical machine learning and character-level deep learning models for classifying CAGD report categories, predicting whether a report will be a high-download report, and forecasting the log of its download count, and to judge how much confidence should be placed in the resulting models given the very small size of the underlying corpus.

To achieve this aim, the study pursued the following objectives:

- To assemble a clean, structured dataset of CAGD report metadata suitable for supervised classification and regression.
- To build two parallel representations of that metadata, engineered tabular features for classical machine learning and character-level text windows for deep learning, so that each model family works from an input structure suited to it.
- To train, validate, and fine-tune representative models from each family, Logistic Regression, Random Forest, and XGBoost for machine learning, and a bidirectional LSTM and a TextCNN for deep learning, under one identical evaluation protocol.
- To compare the resulting models on held-out test performance across the three tasks, and to identify where the results should be treated with caution rather than accepted as a genuine measure of model skill.

- To draw out practical implications for anyone considering automated tagging or popularity prediction tools on a small public-sector document portal like CAGD's.

Four research questions (RQ) guided the empirical work:

RQ1: How do classical machine learning models compare with character-level deep learning models for classifying CAGD report category, predicting whether a report is high-download, and forecasting the log of its download count?

RQ2: Does fine-tuning each family's champion model on the combined training and validation data improve its held-out test performance, or does it sometimes make performance worse, particularly for the small deep learning models trained on this corpus?

RQ3: How much of the strong category classification performance is likely explained by keyword features that closely mirror the category labels themselves, rather than by genuinely learned document structure?

RQ4: What do these results imply for the practical use, and the limits, of automated classification and download-popularity prediction on a small, imbalanced public-sector document corpus like CAGD's?

2. Literature Review

2.1 Theoretical Framework

This study draws on five bodies of work: research on digital public financial management and government transparency, which explains why automated document tagging matters in this setting; a growing applied literature on artificial intelligence in auditing, financial reporting, and corporate governance, which shows the same document- and record-level classification problem appearing across public and private financial oversight; statistical learning theory, which explains how the model families compared here actually learn; the literature on small-sample and imbalanced classification, which frames how the results should be interpreted; and precedent for character- and word-level text classification, which motivates the choice of the two deep learning architectures used here.

Digital public financial management research treats online publication of government reports as one part of a broader accountability and transparency agenda (OECD, 2025). Studies of Ghana's own public financial management systems point to persistent gaps between the ambition of digital reform and its practical implementation, including uneven data quality and inconsistent reporting across agencies (Osei-Dwomoh & Forkuo, 2025). Work on data science applications in public-sector accounting argues that machine learning tools can support this agenda, for example by automating document classification or by surfacing which kinds of reports draw the most public engagement, but only if those tools are evaluated honestly rather than assumed to work simply because they are technically sophisticated (Jiménez-Gómez et al., 2025).

A closely related and rapidly growing literature applies artificial intelligence directly to auditing, financial reporting, and corporate oversight, and it is worth drawing out because it faces essentially the same problem this study does: classifying and monitoring administrative and financial records at a scale that has outgrown manual review. Recent work argues that AI-based tools can materially improve audit quality in public companies by flagging anomalies and supporting evidence review (Apaflo et al., 2026), and a parallel line of work proposes AI-driven compliance monitoring designed specifically for continuous audit readiness under IFRS-based financial reporting rules (Dankwah & Enyejo, 2026). Other recent studies have examined how AI tools might have helped address well-known audit failures and strengthen regulatory compliance in the United States (Yeboah et al., 2026), and how AI-supported business modeling can feed directly into corporate governance and decision-making (Katere et al., 2025). Even a report's financial outcomes can become a classification target in their own right, as in a recent study that used an artificial neural network to classify companies by their return-on-equity performance based on leadership style (Nunoo & Jaganathan, 2025). Taken together, this literature suggests that the CAGD case sits inside a broader, active research agenda on applying classification and prediction tools to public and corporate financial records, rather than being an isolated or unusual use case.

There is also a small but relevant precedent for applying machine learning directly to Ghanaian public-sector administrative data, which matters here because it establishes that models like the ones used in this study have already been tried, with mixed but generally encouraging results, on other kinds of government records from the same country. One recent study used a machine learning approach to predict citizen satisfaction with Ghana's land administration system from routine service records (Tabase et al., 2026), a task that, like this study's category and download-popularity tasks, works from structured administrative metadata rather than free-form data collected specifically for research. Another recent study applied spatial and machine learning analysis to district-level health insurance inequities in Ghana (Ghanem, 2026), again using existing administrative records rather than a purpose-built dataset. Several further Ghanaian studies apply small-sample tabular classification and clustering to institutional data outside the health sector specifically, including predicting the honesty of public transport users returning lost items (Ocansey et al., 2026), clustering patterns in student stress using engineered behavioral features (Dake, 2026), clustering how students accept mobile learning management systems (Dake & Gyimah, 2025), predicting programme choice among tertiary students (Yakubu & Ofori, 2026), and using multinomial logistic regression within an educational assessment framework (Bosson-Amedenu et al., 2026). None of these studies works with CAGD data specifically, but together they show that the general approach used here, applying classical machine learning to modestly sized, real-world Ghanaian administrative or institutional datasets, has working precedent rather than being untested.

Statistical learning theory supplies the technical core of the framework. Every classifier used here, from Logistic Regression to XGBoost to the character-level neural networks, learns a function that maps input features to a prediction by minimizing error on a training sample. Logistic Regression assumes a linear, regularized relationship between predictors and the log-odds of each class (Hoerl & Kennard, 1970).

Random Forest and XGBoost relax that linearity assumption and can represent interactions and threshold effects between predictors without those interactions being specified by hand (Breiman, 2001; Chen & Guestrin, 2016). Comparable tree-ensemble methods have recently been benchmarked against each other on Ghanaian and West African classification problems outside the financial domain, for example comparing boosting, random forest, and other ensemble classifiers for land cover mapping (Mantey et al., 2026a, 2026b) and comparing support vector machines against random forests for tracking land use change from mining activity (Larbi et al., 2025), which is useful supporting evidence that these specific algorithms transfer reasonably well to African, resource-constrained modeling settings. The bidirectional LSTM and the TextCNN work directly on the raw character sequence of a report's title and description, learning to detect useful patterns in the text itself rather than relying on features an analyst engineered in advance (Hochreiter & Schmidhuber, 1997). This mirrors comparable applications of character and word level neural text classifiers in financial and administrative document settings (Agbesi et al., 2024; Wicaksono & Khodra, 2024), and it echoes recent findings that a well-tuned gradient-boosted tree model can match or outperform far larger neural architectures on structured or short-text classification tasks, particularly when the amount of labeled data is limited (Bohacek & Bravansky, 2024). Choosing sensible hyperparameters and reporting the right evaluation metrics both matter for this comparison to be fair; a recent narrative review sets out the main performance metrics used for classification and regression tasks in machine learning and the trade-offs between them (Ayintareba et al., 2026), and recent applied work on telecommunications customer churn shows concretely how randomized hyperparameter tuning and ensemble voting can be used together to squeeze more reliable performance out of a modestly sized tabular dataset (Mettle et al., 2026a, 2026b), a methodological pattern this study follows closely for its own machine learning models. More broadly, machine learning applied to customer-facing administrative or service data, of which download-popularity prediction is one example, has its own applied literature on which features and modeling choices tend to work in practice (Abugbilla, 2025).

Text classification is the task the bidirectional LSTM and TextCNN models are built for, and there is now a substantial applied literature on deep learning approaches to it that this study draws on directly. Recent comparative work in a Ghanaian setting evaluates several deep learning architectures for text-based sentiment classification and reports which design choices help most on shorter, noisier text (Afriyie & Weyori, 2026), which is directly relevant given that CAGD report titles and descriptions are themselves short. TextCNN-style convolutional architectures, the same family used in this study, have recently been applied successfully to classifying customer-facing text for other purposes, such as linking brand-related language to purchase intention (Shen et al., 2025), and convolutional neural networks tuned with metaheuristic optimization have been used for sentiment classification of social media text more broadly (Li et al., 2025). Character- and word-level text classifiers have also been applied to specifically administrative and integrity-related text classification problems that resemble this study's category task in spirit, including detecting plagiarism in written text (Sajid et al., 2025) and detecting fine-grained antisocial behavior in social media posts (Asante & Hajek, 2025). Given how much this study's category results depend on a small set of keyword features, it is also worth noting that building and annotating

text classification resources for lower-resource or code-mixed languages, comparable in spirit to how CAGD report titles mix formal and informal English, remains an active and nontrivial task in its own right (Ndao et al., 2025), and a recent scoping review of natural language processing technologies applied to African public health data found that many models built elsewhere transfer only partially to African-context text, reinforcing the caution this study takes toward its own text-based results (Hu et al., 2025).

The fourth body of work, on small-sample and imbalanced classification, is the one most directly relevant to how this study's results should be read. When a dataset is small and its classes are unevenly represented, standard evaluation procedures such as a single stratified split can behave unpredictably, and a classifier's apparent accuracy can be inflated by features that happen to correlate almost perfectly with the label, rather than by any real predictive skill (Little & Rubin, 2019). The synthetic minority oversampling technique, or SMOTE, is one common remedy for the training-side imbalance problem, generating synthetic examples of minority classes by interpolating between real neighbors in feature space (Chawla et al., 2002), and it is used here for the tabular machine learning models. A recent applied study on health data classification works through this same oversampling approach step by step and reaches a similar conclusion to this study's own experience, that oversampling meaningfully helps a minority class without fully solving every problem imbalance creates (Agyemang et al., 2025). Related methodological work on properly applying logistic regression to small or complex survey samples (Dey et al., 2025) and on handling missing data in modestly sized real-world datasets (Alejo-Sanchez et al., 2025) informed the missing-data and modeling choices described in Section 3. But oversampling and careful missing-data handling address only part of the imbalance problem; neither protects against feature leakage, and a central task of this study is to flag, honestly, where that risk is present in the results reported in Section 4.

2.2 Conceptual Framework

The empirical work in this study follows a simple pipeline (Fig. 1). Report metadata mined from the CAGD portal and its archived snapshots is cleaned and merged into a single table, engineered into two parallel representations, a tabular feature set for the machine learning models and a 128-character text window for the deep learning models, and used to build three supervised targets: report category, whether a report is high-download, and the log of its download count. Both model families then pass through an identical two-stage protocol, a randomized hyperparameter search evaluated on a held-out validation partition, followed by a fine-tuning stage in which each family's best configuration is refit on the combined training and validation data before one final evaluation on a test partition that neither stage ever touched during model selection. The pipeline closes with an evaluation stage that reports standard classification and regression metrics, along with confusion matrices and ROC curves, and that explicitly asks whether the resulting performance reflects genuine document understanding or a simpler, less interesting explanation, such as an engineered feature that happens to encode the label directly. That last question, addressed most directly in Sections <link rid="Sec25">4</link> and 5, is treated as a first-class part of the evaluation rather than an afterthought, because a small, imbalanced dataset like this one makes that kind of leakage more likely, not less.

3. Methodology

This section describes the end-to-end pipeline used to classify CAGD report categories, predict download popularity, and forecast log download counts. The description below reflects the pipeline exactly as it was run for the results reported in Section 4, including several corrections and adaptations that were necessary once the pipeline was actually executed on this dataset. Where an earlier version of this methodology specified a component that was not, in the end, exercised in this run, that is stated plainly rather than left implicit, in the same spirit as the honest reporting recommended by Bergstra and Bengio (2012) for any hyperparameter search protocol.

3.1 Data Sourcing and Cleaning

Report records were assembled from three complementary sources: the live CAGD portal, Internet Archive snapshots of earlier versions of the CAGD website that preserved historical download counters, and document metadata embedded in the site's own JavaScript bundle. Raw tables were merged on normalized title and category, and where the same report appeared more than once, the observation with the higher download count was kept, or, when tied, the more recently published one. The cleaned corpus contains 131 unique reports across eight original categories, Payroll, Quarterly Reports, Annual Reports, IPSAS, Chart of Accounts, Allowance, Manuals, and Conference, published between 2019 and 2024.

3.1.1 Data Quality and Cleaning Rules

A small set of deterministic rules was applied consistently across the whole corpus. Titles were normalized by collapsing whitespace and stripping HTML entities while keeping original casing for display. Category labels from older site versions were mapped onto a single controlled vocabulary. File size and download count were parsed as numbers, with invalid entries set to missing rather than filled with zero. Rows without a usable title or category were excluded from supervised training. Extreme download counts were kept as genuine engagement signals rather than capped or removed, since a report with an unusually high download count, such as a widely referenced chart-of-accounts edition, is a real and informative data point rather than an error.

3.2 Handling Missing Data

Missing data in this corpus comes mainly from incomplete archive records and documents that appear only in the site's JavaScript bundle without a full publish date. Rows without a defined category label were dropped before training, since there is nothing to learn from a row with no answer to predict. For the remaining engineered numeric features, month, quarter, and file size, gaps were intended to be filled using median values computed on the training partition only, following standard practice for avoiding information leakage from validation or test data into training (Van Buuren, 2018). In practice, running the pipeline on this corpus surfaced one further and worth-reporting issue: the file_size column had no observed values at all across the entire dataset, so it carried no usable information regardless of the imputation rule applied to it, and the trained models effectively ignored it. This is flagged here rather than

left to be discovered by a reader working through the results tables. Multiple imputation methods such as MICE were not used, given that a sample of 131 records offers too few degrees of freedom to fit an imputation model with any confidence (Van Buuren, 2018).

3.3 Data Processing and Feature Engineering

Three supervised targets were defined from the cleaned data. The first, category, is the report's category label. The second, high_download, is a binary flag for whether a report's download count is above the corpus median. The third, log_download_count, is a continuous target equal to the natural logarithm of the report's download count, used to reduce the influence of a small number of very heavily downloaded reports on the regression models.

Both download-based targets are built from a single cumulative download count mined at one point in time, not from a time-series of access logs, so a report that has been available for longer has simply had more opportunity to accumulate downloads than one published recently, independent of how genuinely popular either one is. This is a real limitation of the underlying data rather than something the modeling choices described below can fully correct, and it is checked directly rather than left as a theoretical worry: a Spearman correlation between publication year and log_download_count across the 120 reports with a recorded year is weak to moderate and, notably, positive (rho = 0.27, p = 0.004), meaning newer reports in this corpus tend to have somewhat higher, not lower, download counts. That is the opposite of the pattern a pure accumulation-over-time story would predict, and it suggests that whatever is driving download counts in this corpus, growth in the CAGD portal's overall traffic, changes in which categories were published when, or genuine differences in report-level interest, it is not simply older reports mechanically outscoring newer ones. Publication year is also already included as one of the plain numeric features available to every model in Section 3.3.1, so to whatever extent exposure time does matter, the models have direct access to a proxy for it rather than being blind to it. Even so, this is not the same as a proper exposure-adjusted analysis, and Section 5.3 returns to what a more rigorous version of that analysis would look like.

Because several of the original eight categories had very few examples, Conference with a single report, Manuals with two, Allowance with three, and Chart of Accounts with seven, the category target used for modeling groups every category with fewer than eight members into a single "Other" class, leaving five classes in total: Annual Reports, IPSAS, Payroll, Quarterly Reports, and Other. This grouping was necessary for two concrete reasons rather than a stylistic preference. First, a stratified split cannot guarantee that a category with only one or two total examples appears in the training, validation, and test partitions at once, and attempting one raised an error before this grouping was applied. Second, the multiclass XGBoost objective used here requires every class to be present in the training fold, which a singleton class like Conference cannot guarantee under any split. Grouping rare classes into a shared "Other" bucket is a standard and defensible response to this kind of small-sample sparsity, but it does mean that the category task reported in Section 4 is a five-class problem, not the original eight-class one, and that distinction is repeated in the Results and Discussion sections rather than left implicit.

Text fields, title and description, were lower-cased and concatenated for both model families. Twenty-five keyword indicators were also engineered from this combined text, flagging whether words such as budget, payroll, ipsas, quarterly, or annual appear in a report's title or description. As discussed further in Section 4 and Section 5, several of these keyword features are close to synonyms of the category labels themselves, and a dedicated ablation reported in Section 4.6 confirms that this matters a great deal for how the category classification results should be read.

3.3.1 Tabular Representation for Machine Learning

For the machine learning models, each report is one row. The feature matrix combines the twenty-five keyword indicators, engineered numeric features such as title length, description length, and log file size, and a one-hot encoded file extension field. Numeric features were standardized using the training partition's own mean and standard deviation, never the full corpus, to avoid leaking validation or test statistics into training (Pedregosa et al., 2011).

3.3.2 Character-Level Representation for Deep Learning

For the deep learning models, each report's combined and lower-cased title and description text was mapped to a fixed 128-character window, with characters converted to integer codes and padded or truncated as needed. This gives the bidirectional LSTM and the TextCNN direct access to the raw text of a report, rather than to the same hand-engineered keyword features used by the machine learning models, which is the main reason the two model families were compared in the first place.

3.3.3 Class Imbalance Handling

Two separate adjustments address the imbalance present in this corpus. The first, described above, groups rare category classes into an "Other" bucket for both the split and the modeling target. The second is the synthetic minority oversampling technique, or SMOTE, applied to the training partition only for the machine learning classifiers, with the neighbor count capped whenever a class had fewer training examples than the default of five neighbors requires (Chawla et al., 2002). Validation and test partitions were never oversampled. SMOTE was not applied to the deep learning models; instead, a class-weighted cross-entropy loss was used during their training to give minority classes proportionally more influence on the training signal.

3.3.4 Components Specified but Not Exercised in This Run

Two further components were part of the original project design but were not part of the pipeline actually run for the results reported in Section 4, and it is more honest to say so plainly than to describe them as if they had been used. A self-supervised denoising-autoencoder pretraining stage for the tabular and text encoders was implemented in code but was not called during training for this run, so none of the reported results reflect that pretraining. A Vision Transformer for classifying report cover-image thumbnails was also specified in the original design, but the mined corpus contained no usable cover-image URLs at all, so no image model was trained. Both are revisited as future work in Section 5.4.

3.4 Model Selection and Architectures

Two model families were compared, chosen because each represents a different way of extracting signal from the same underlying report metadata: classical tree-based and linear models working from engineered tabular features, and character-level neural networks working directly on raw text.

3.4.1 Machine Learning Models

Three machine learning models were benchmarked: Logistic Regression with L2 regularization, used in its multinomial form for the category task and its binary form for the high-download task (Hoerl & Kennard, 1970); Random Forest (Breiman, 2001); and XGBoost, a gradient-boosted tree ensemble (Chen & Guestrin, 2016). All three were implemented through scikit-learn's standard estimator interface, with XGBoost added through its own scikit-learn-compatible wrapper (Pedregosa et al., 2011).

3.4.2 Deep Learning Models

Two character-level architectures were evaluated: a two-layer bidirectional LSTM with an embedding layer and dropout regularization (Hochreiter & Schmidhuber, 1997), and a TextCNN using parallel convolutional filters of widths three, four, and five over the same character embeddings, a design commonly used for short-text classification tasks (Agbesi et al., 2024). Both models end in a single linear classification head and were trained with the Adam optimizer, gradient clipping at a maximum norm of 1.0 for numerical stability, and a class-weighted cross-entropy loss when more than two classes were present (Pascanu et al., 2013).

3.5 Hyperparameters

Hyperparameters for the machine learning models were tuned through randomized search using scikit-learn's parameter sampler, drawing six candidate configurations per model per task from the search spaces in Table 1, rather than through an exhaustive grid or a full cross-validated search. Randomized search is both computationally cheaper and, according to Bergstra and Bengio (2012), tends to explore a wider effective range of hyperparameter values for the same computational budget. Each configuration was evaluated once against the held-out validation partition; a k-fold cross-validation scheme was considered in the original project design but was not part of the pipeline actually run here, and this is treated as a limitation in Section 5.3 rather than described as if it had been used.

Table 1
Hyperparameter search spaces used for each model family.

Model family	Hyperparameter(s)	Search space	Configurations sampled
Logistic Regression	C (inverse regularization strength)	0.01 to 100, log-uniform	6
Random Forest (classification)	n_estimators / max_depth / class_weight	{200, 400, 600} / {5, 10, None} / {balanced, None}	6
XGBoost (classification)	n_estimators / max_depth / learning_rate	{100, 300} / {3, 5, 7} / {0.05, 0.1}	6
Ridge (regression)	alpha (regularization strength)	0.001 to 100, log-uniform	6
Random Forest (regression)	n_estimators / max_depth	{200, 400} / {5, 10, None}	6
XGBoost (regression)	n_estimators / max_depth / learning_rate	{100, 300} / {3, 5, 7} / {0.05, 0.1}	6
BiLSTM / TextCNN	hidden size / dropout / learning rate / batch size	{64, 128} / {0.2, 0.3} / {0.001, 0.0005} / 16	2 configurations, each repeated over 3 seeds

For the deep learning models, training was capped at 80 epochs with early stopping on validation macro-F1 at a patience of 10 epochs, and each of the two configurations was repeated across three random seeds (42, 123, and 456) so that the results would not depend on a single, possibly lucky, initialization (Bouthillier et al., 2021).

3.6 Training, Validation, and Testing

A single stratified 70 percent train, 15 percent validation, 15 percent test split was constructed once, on the five-class grouped category label, with a fixed random seed of 42, and then reused for all three tasks. This means the high_download and log_download_count tasks were not independently stratified on their own targets, only inherited from the category-based split, which is stated here plainly because it affects how the high_download test set turned out to be composed, as discussed in Section 4. The resulting partition contains 91 training reports, 20 validation reports, and 20 test reports.

3.6.1 Training Procedure

Machine learning classifiers were fit on the SMOTE-augmented training partition for each hyperparameter configuration, evaluated on the untouched validation partition, and the configuration with the highest validation macro-F1, or the lowest validation RMSE for the regression task, was designated that family's search-stage champion. In the fine-tuning stage, the winning configuration was refit on the combined training and validation data before a single evaluation on the test partition. The

deep learning models followed the same two stages: an initial supervised training run with early stopping monitored on the validation partition, followed by a fine-tuning run that resumed from the best validation checkpoint, reduced the learning rate to one tenth of its original value, and continued training on the combined training and validation data for additional epochs before the final test evaluation.

One methodological detail is worth stating plainly rather than leaving implicit, in the same spirit as similar caveats raised in comparable forecasting studies. Because the validation partition is included in the fine-tuning training data, and, for the binary task, the same validation partition is also used to re-tune the decision threshold after fine-tuning, the fine-tuning stage does not have a fully independent signal guiding it. The test partition itself was never touched by either stage of model selection and remains an honest basis for comparison, but the size of the fine-tuning effect reported in Section 4.3 should be read with that mechanism in mind.

3.6.2 Evaluation Metrics

For the category and high_download classification tasks, accuracy, macro-F1, weighted-F1, and ROC-AUC are reported. ROC-AUC for the binary task uses the predicted probability of the positive class; for the five-class category task, it is computed as a weighted one-versus-rest average across classes. For the log_download_count regression task, root mean squared error, mean absolute error, and R-squared are reported.

3.7 Decision Threshold Selection

For the binary high_download task, the default 0.5 probability threshold does not necessarily give the best balance between sensitivity and specificity once a model's predicted probabilities are shaped by the class-imbalance corrections used during training. Instead, the decision threshold was selected on the validation partition by maximizing Youden's J statistic, the sum of sensitivity and specificity minus one, evaluated across a grid of candidate thresholds from 0.05 to 0.95. This threshold was re-selected once at the search stage and again after fine-tuning, since fine-tuning can shift a model's predicted probabilities enough to change which threshold is optimal. For the five-class category task, the model's predicted class was simply the class with the highest predicted probability.

3.8 Best Model Selection and Fine-Tuning

Model selection followed two stages for each task. In the first stage, every candidate model within a family, machine learning or deep learning, went through the randomized hyperparameter search described in Section 3.5, and the configuration with the best validation score within its family became that family's champion. In the second stage, each family's champion was fine-tuned on the combined training and validation data as described in Section 3.6.1, and evaluated once on the test partition.

The single overall champion for each task was then chosen by comparing the fine-tuned validation score of the machine learning champion against that of the deep learning champion. Where the two families' validation scores were tied, as happened for the category task, the machine learning champion was selected, since it was evaluated first in the underlying implementation. This is a plain implementation

detail rather than a principled tie-breaking rule, and it is reported here rather than left to be inferred from the results.

3.9 Keyword-Leakage Ablation

Because several of the twenty-five engineered keyword indicators are close to synonyms of the category labels themselves, for example a flag for the word payroll appearing in a report whose category is Payroll, a targeted ablation was run to check how much of the category task's headline accuracy depends on that overlap rather than on other, less tautological signal in the report metadata. The ablation reused the identical stratified split, the identical randomized hyperparameter search procedure described in Section 3.5, and the identical two-stage search-then-fine-tune protocol described in Section 3.6, changing only one thing: all twenty-five keyword columns were removed from the machine learning feature matrix before the category task was re-run from scratch, leaving eight remaining numeric features (year, file size, title length, description length, log file size, publish month, publish quarter, and a featured-report flag) plus the one-hot encoded file extension. The with-keyword run was also repeated under this same ablation script as an internal consistency check, to confirm it reproduced the original result exactly before trusting the without-keyword comparison. To check whether the resulting gap depended on the one train, validation, and test partition used throughout this study, the same with-keyword versus without-keyword comparison was then repeated under four additional stratified splits, drawn with different random seeds, while keeping every other part of the pipeline unchanged. Results for both the single-split ablation and the five-split robustness check are reported in Section 4.6.

4. Results

All results reported in this section come from a single, complete run of the pipeline described in Section 3, covering all three tasks, both model families, the search stage, and the fine-tuning stage. No test-set number in this section was used to choose a model, a hyperparameter configuration, or an early-stopping point at any earlier stage.

4.1 Search-Stage Model Comparison

Table 2 reports the best validation score achieved by each candidate machine learning model during the randomized hyperparameter search, and Fig. 2 shows the same comparison as a bar chart for all three tasks. For the category task, XGBoost reached a validation macro-F1 of 1.00, ahead of Random Forest at 0.91 and Logistic Regression at 0.76. For the high_download task, the ordering reversed: Logistic Regression reached the highest validation macro-F1 at 0.76, ahead of Random Forest at 0.73 and XGBoost at 0.69. For the regression task, Random Forest reached the lowest, and therefore best, validation RMSE at 0.59, ahead of XGBoost at 0.64 and Ridge at 0.88, so Random Forest was carried forward as the machine learning champion for that task on the same basis used for every other model in this study: the best validation score within its family.

Table 2
Best validation score reached by each machine learning model during hyperparameter search, by task.

Task	Logistic Regression / Ridge	Random Forest	XGBoost
Category (macro-F1)	0.76	0.91	1.00
High download (macro-F1)	0.76	0.73	0.69
Regression (validation RMSE)	0.88	0.59	0.64

Among the deep learning models, the TextCNN clearly outperformed the BiLSTM on the category task, reaching validation macro-F1 of 1.00 in four of its six seed and configuration runs, compared with a range of 0.48 to 0.63 for the BiLSTM. On the high_download task, both deep learning models were noticeably weaker than on the category task, with the TextCNN's validation macro-F1 ranging from 0.60 to 0.73 across its six runs and the BiLSTM ranging from 0.58 to 0.64.

On the held-out test partition at the search stage, before any fine-tuning, the machine learning champions reached test accuracy of 1.00 for category (XGBoost) and 0.75 for high_download (Logistic Regression), with a test ROC-AUC of 1.00 and 0.81 respectively. Deep learning results at the search stage varied more by seed. The best individual TextCNN run on category reached a test accuracy of 1.00 (matching XGBoost), while the best individual TextCNN run on high_download reached a test accuracy of 0.85, noticeably higher than the machine learning champion on that task, though this particular run was not the one carried forward into fine-tuning, since model selection is based on validation score rather than test score, as it should be for a fair comparison.

4.2 Lightweight Text Baseline Comparison

The comparison so far is between engineered tabular features and character-level neural networks, but it leaves out a simpler possibility: a standard bag-of-words text classifier, without any hand-engineered keyword features or character-level modeling at all. To check where such a baseline lands, a TF-IDF representation (word unigrams and bigrams, up to 2,000 features, fit on the training fold only) was paired with a Logistic Regression classifier and a linear support vector machine, using the identical stratified split, identical training-then-fine-tuning protocol, and identical test partition as every other model in this study. Table 3 reports the result for both classification tasks.

Table 3
Lightweight TF-IDF word n-gram baseline, fine-tuned test performance, using the same split and protocol as the main pipeline.

Task	Model	Test accuracy	Test macro-F1	Test weighted-F1	Test ROC-AUC
Category	TF-IDF + Logistic Regression	1.00	1.00	1.00	N/A
Category	TF-IDF + Linear SVM	1.00	1.00	1.00	N/A
High download	TF-IDF + Logistic Regression	0.80	0.79	N/A	0.76
High download	TF-IDF + Linear SVM	0.80	0.79	N/A	0.79

Two things stand out. First, on the category task, a plain bag-of-words classifier reaches the same perfect test accuracy as the engineered-feature XGBoost champion, with no keyword engineering at all. This is an important piece of evidence for the leakage question addressed directly in Section 4.6: the category names themselves, or close variants of them, are often simply present as words in a report's title or description, so any model with access to that text, whether through a hand-built keyword flag or a generic TF-IDF vector, picks up the same shortcut. The keyword features did not introduce the leakage; they made explicit a signal that was already sitting in the raw text. Second, on the high_download task, where no such shortcut exists, the TF-IDF baseline reaches 0.80 test accuracy, slightly above the tabular Logistic Regression champion's 0.75, while its ROC-AUC of 0.76 to 0.79 is slightly below the tabular champion's 0.80. In other words, a simple, off-the-shelf text baseline is fully competitive with the more elaborately engineered tabular pipeline on the one task in this study that does not have an obvious leakage shortcut, which is a useful, humbling check on how much the engineered feature set in Section 3.3 actually buys over the raw text it was built from.

4.3 Effect of Fine-Tuning

Table 4 compares each task's champion test performance at the search stage with its performance after fine-tuning on the combined training and validation data. The effect of fine-tuning was not uniform, and it is reported here exactly as it came out rather than smoothed over.

Table 4
Search-stage versus fine-tuned test performance for each task's family champions.

Task	Model	Search test metric	Fine-tuned test metric	Change
Category	XGBoost (ML champion)	Accuracy 1.00, macro-F1 1.00	Accuracy 1.00, macro-F1 1.00	No change
Category	TextCNN (DL champion)	Accuracy 1.00, macro-F1 1.00	Accuracy 0.85, macro-F1 0.73	Fine-tuning reduced test performance
High download	Logistic Regression (ML champion)	Accuracy 0.75, ROC-AUC 0.81	Accuracy 0.75, ROC-AUC 0.80	Accuracy unchanged, ROC-AUC slightly lower
High download	TextCNN (DL champion)	Accuracy 0.70, macro-F1 0.64	Accuracy 0.65, macro-F1 0.56	Fine-tuning reduced test performance
Regression	Random Forest (ML champion)	RMSE 0.54, R-squared 0.86	RMSE 0.56, R-squared 0.85	Fine-tuning slightly reduced test performance

For the machine learning champions, fine-tuning left the category result completely unchanged and produced only a small reduction in ROC-AUC for the high_download task and a small increase in RMSE for the regression task. For the deep learning champions, fine-tuning reduced test performance on both classification tasks, most noticeably on the category task, where the TextCNN's test macro-F1 fell from a perfect 1.00 at the search stage to 0.73 after fine-tuning. This pattern is consistent with the methodological caveat raised in Section 3.6.1: because fine-tuning both trains on and re-tunes its stopping and threshold decisions against the same validation partition, and because this corpus is very small, the fine-tuning stage on this dataset behaved more like a source of added variance for the deep learning models than a dependable route to better generalization. This should be read as a property of fine-tuning a small neural model on 111 combined training and validation examples, not as a general claim that fine-tuning does not work.

4.4 Classification Diagnostics

Figure 3 shows the confusion matrix for the fine-tuned XGBoost category champion on the 20-report test set. Every one of the 20 test reports was classified into its correct grouped category: 4 of 4 Annual Reports, 3 of 3 IPSAS, 6 of 6 Payroll, 5 of 5 Quarterly Reports, and 2 of 2 Other. Figure 4 shows the corresponding one-versus-rest ROC curves, all of which reach or nearly reach the top-left corner of the plot, consistent with the perfect accuracy score.

This result is treated with real caution rather than reported as a straightforward success, for the reasons discussed in Section 5.1. Several of the twenty-five keyword indicators used as model features, for example a flag for whether the word payroll appears in a report's title or description, are close to synonyms of the category labels themselves. A report titled "2024 Payroll Circular" containing the word payroll is not a hard case for a model to classify correctly once it has access to a feature that essentially

repeats the answer, and Section 4.6 shows directly, through a keyword-removal ablation, that this is indeed most of what is driving the perfect test score rather than any deeper understanding of report content.

Figure 5 shows the confusion matrix for the fine-tuned Logistic Regression high_download champion, and Fig. 6 shows the ROC curve comparing the fine-tuned Logistic Regression and TextCNN models on this task. The test set for this task, inherited from the category-based split rather than independently stratified, contained 14 reports below the download median and 6 above it. The Logistic Regression model reached a ROC-AUC of 0.80, compared with 0.65 for the fine-tuned TextCNN, and this task's confusion matrix shows a more typical, imperfect pattern of correct and incorrect calls than the category task does, which is itself indirect evidence that the category task's perfect score is unusual rather than representative of what these models can normally achieve on this corpus.

Because the test partition for this task is imbalanced, 14 reports below the download median against 6 above it, ROC-AUC alone can look more encouraging than a model's practical usefulness on the minority class actually warrants. Table 5 reports two additional diagnostics chosen specifically to check that: the area under the precision-recall curve, which is more sensitive to minority-class performance than ROC-AUC under imbalance, and the F1 score computed separately for each class rather than averaged together.

Table 5
Supplementary imbalance-aware diagnostics for the fine-tuned Logistic Regression high_download champion (test set, n = 20).

Metric	Value
Precision-recall AUC (average precision)	0.74
F1, class 0 (not high download, n = 14)	0.81
F1, class 1 (high download, n = 6)	0.62

The gap between the two per-class F1 scores is the more informative number here. The model is noticeably better at correctly identifying reports that are not high-download than at correctly flagging the ones that are, which is the expected direction of difficulty on an imbalanced minority-detection problem and is masked by looking at accuracy or ROC-AUC alone. A precision-recall AUC of 0.74, compared with a naive baseline of 0.30 (the minority class's share of the test set), still indicates real, usable skill at ranking reports by download likelihood, but the practical takeaway for CAGD is that this model is currently a better tool for confirming which reports are ordinary than for confidently flagging the next high-download report, and that asymmetry should inform how any deployed version of it is used.

4.5 Regression Diagnostics

Figure 7 plots predicted against actual log download count for the fine-tuned Random Forest regressor, alongside the corresponding residual plot. The model reached a test RMSE of 0.56, a test MAE of 0.41,

and a test R-squared of 0.85 on the 20-report test set. The predicted-versus-actual plot shows most points clustered reasonably close to the line of perfect prediction, with the largest residuals occurring for reports at the lower end of the download-count range, where a small absolute error in log-space corresponds to a larger error in the original download-count scale.

.

4.6 Keyword-Leakage Ablation (RQ3)

Table 6 and Fig. 8 report the result of the ablation described in Section 3.9. Repeating the with-keyword run under the ablation script reproduced the original result exactly, test accuracy, macro-F1, weighted-F1, and ROC-AUC all at 1.00 for the fine-tuned XGBoost champion, confirming the ablation setup is a faithful copy of the main pipeline and that only the feature set changed between the two runs.

Removing all twenty-five keyword features changed both the winning model and its performance. The champion switched from XGBoost to a fine-tuned Logistic Regression model, and test accuracy fell from 1.00 to 0.80, macro-F1 fell from 1.00 to 0.74, weighted-F1 fell from 1.00 to 0.79, and ROC-AUC fell from 1.00 to 0.96. Figure 9 shows the test-set confusion matrices for both runs side by side; without the keyword features, four of the twenty test reports were misclassified, compared with zero under the original feature set.

Table 6
Keyword-leakage ablation: fine-tuned test performance on the category task, with and without the twenty-five kw_* keyword features.

Feature set	Champion model	Test accuracy	Test macro-F1	Test weighted-F1	Test ROC-AUC
With keyword features (34 features)	XGBoost	1.00	1.00	1.00	1.00
Without keyword features (9 features)	Logistic Regression	0.80	0.74	0.79	0.96

The single-split result above rests on one 20-report test partition, and Section 5.3 flags that a different split could plausibly move it. To check this directly, the same with-keyword and without-keyword comparison was repeated under four additional stratified splits, using different random seeds (7, 99, 202, and 2024) for the train, validation, and test partition while keeping every other part of the pipeline, model settings, hyperparameter search space, and fine-tuning procedure, identical to the main run. Table 7 and Fig. 10 report the result across all five splits, including the original seed-42 split.

Table 7
Keyword-leakage ablation repeated across five independent stratified splits (fine-tuned test accuracy and macro-F1, category task).

Split seed	With-keyword accuracy	Without-keyword accuracy	Accuracy gap	With-keyword macro-F1	Without-keyword macro-F1	Macro-F1 gap
42	1.00	0.80	0.20	1.00	0.74	0.26
7	1.00	0.80	0.20	1.00	0.68	0.32
99	1.00	0.75	0.25	1.00	0.71	0.29
202	1.00	0.80	0.20	1.00	0.68	0.32
2024	0.90	0.70	0.20	0.75	0.59	0.16
Mean (SD)	0.98 (0.04)	0.77 (0.04)	0.21 (0.02)	0.95 (0.11)	0.68 (0.05)	0.27 (0.07)

The accuracy gap between the with-keyword and without-keyword models is remarkably stable across splits, ranging only from 0.20 to 0.25 with a standard deviation of 0.02. The macro-F1 gap varies a little more, from 0.16 to 0.32, which is expected given how sensitive macro-F1 is to a handful of misclassified reports in the smaller grouped classes on a test set of only 20 reports, but it never closes and never reverses direction across any of the five splits. This is a meaningfully stronger basis for the leakage conclusion than the single-split result alone, since the finding does not depend on one particular, possibly favorable or unfavorable, partition of the 131 reports. It is worth noting that the without-keyword accuracy itself also varies somewhat by split, from 0.70 to 0.80, which is a useful reminder that even the more trustworthy, ablated estimate of category performance still carries real uncertainty at this sample size, even if the gap caused by the keyword features does not.

This gives a direct, quantified answer to RQ3 rather than an informed guess. The twenty-point drop in accuracy and twenty-six-point drop in macro-F1 confirm that keyword features do a great deal of the work behind the original, perfect-looking result, and that the headline 1.00 score substantially overstates what these models can do on report content alone. At the same time, 0.80 accuracy and a macro-F1 of 0.74 on a five-class problem, using only eight non-keyword numeric features plus file extension, is well above what chance alone would produce, roughly 0.20 for random guessing across five classes or somewhat higher if a model simply predicted the majority training class every time. So the honest reading is not that the category task is pure leakage with nothing underneath it. Rather, the keyword features add a large, avoidable inflation on top of a smaller, more genuine signal that the non-keyword metadata, publication timing, title length, description length, and file characteristics, does carry some real information about a report's category. Repeating the comparison across five splits shows this is not a one-off finding tied to a single lucky or unlucky partition of the data. Section 5.1 returns to what this means in practice.

4.7 Overall Champion Models

Table 8 summarizes the final champion model for each task, following the selection rule described in Section 3.8. For category, the fine-tuned XGBoost classifier was the overall champion, reaching test accuracy, macro-F1, weighted-F1, and ROC-AUC all at 1.00. For high_download, the fine-tuned Logistic Regression model was the overall champion, reaching test accuracy of 0.75, macro-F1 of 0.72, weighted-F1 of 0.75, and ROC-AUC of 0.80. For the regression task, the fine-tuned Random Forest model was the overall champion, reaching test RMSE of 0.56, MAE of 0.41, and R-squared of 0.85.

Table 8
Overall champion model for each task, fine-tuned test performance.

Task	Champion	Test accuracy	Test macro-F1	Test weighted-F1	Test ROC-AUC	Test RMSE	Test MAE	Test R-squared
Category	XGBoost	1.00	1.00	1.00	1.00	N/A	N/A	N/A
High download	Logistic Regression	0.75	0.72	0.75	0.80	N/A	N/A	N/A
Regression	Random Forest	N/A	N/A	N/A	N/A	0.56	0.41	0.85

Every test-set number reported so far in this study is a single point estimate computed on a 20-report test partition, and a sample that small can make an estimate look more precise than it really is. To make that uncertainty explicit rather than leaving it implicit, a nonparametric bootstrap (2,000 resamples of the test set, drawn with replacement) was used to compute a 95 percent confidence interval around each headline metric. Table 9 reports the result for every champion, including the keyword-ablated category model from Section 4.6, since that comparison carries the same small-sample uncertainty as everything else in this study.

Table 9
Bootstrap 95% confidence intervals (2,000 resamples of the 20-report test set) for the main headline metrics.

Task / model	Metric	Point estimate	95% CI
Category, XGBoost (with keywords)	Accuracy	1.00	[1.00, 1.00]
Category, XGBoost (with keywords)	Macro-F1	1.00	[1.00, 1.00]
Category, Logistic Regression (without keywords)	Accuracy	0.80	[0.60, 0.95]
Category, Logistic Regression (without keywords)	Macro-F1	0.74	[0.50, 0.94]
High download, Logistic Regression	Accuracy	0.75	[0.55, 0.95]
High download, Logistic Regression	Macro-F1	0.72	[0.47, 0.90]
High download, Logistic Regression	ROC-AUC	0.80	[0.50, 1.00]
High download, Logistic Regression	PR-AUC	0.74	[0.35, 1.00]
Regression, Random Forest	RMSE	0.56	[0.35, 0.76]
Regression, Random Forest	MAE	0.41	[0.25, 0.59]
Regression, Random Forest	R-squared	0.85	[-0.03, 0.95]

The category task's fine-tuned XGBoost result is the one exception with a genuinely tight interval, because it made zero test-set errors and a bootstrap resample of a perfect classifier stays perfect. Every other interval is wide enough to change the practical reading of the result. The high_download ROC-AUC's interval, 0.50 to 1.00, spans the entire range from no better than chance to flawless discrimination, and the regression task's R-squared interval, -0.03 to 0.95, includes the possibility that the model explains essentially none of the variance in the test set, alongside the possibility that it explains nearly all of it. None of this means the point estimates reported throughout Section 4 are wrong; it means they should be read as this study's best single guess from a 20-report test set, not as a precise measurement, and any comparison between models or between this study and a future one should lean on these intervals rather than on the point estimates alone.

5. Discussion

This study set out to compare classical machine learning and character-level deep learning models for classifying CAGD report categories, predicting download popularity, and forecasting log download counts, while paying close attention to how much confidence the very small size of this corpus should allow. Four research questions guided that comparison, and the results in Section 4 give a fairly clear, though appropriately cautious, answer to each.

5.1 Summary of Findings

On RQ1, the two model families performed quite differently depending on the task, and a simple lightweight baseline turned out to matter as much as either of them. For the category task, both the machine learning champion, XGBoost, and the best individual deep learning run, a TextCNN, reached perfect or near-perfect test accuracy, and a plain TF-IDF bag-of-words classifier with no keyword engineering at all reached the same perfect score (Section 4.2), which is itself a strong hint about where that score comes from. For the high_download task, the machine learning champion, Logistic Regression, outperformed the fine-tuned deep learning champion, TextCNN, by a wide margin on ROC-AUC, 0.80 against 0.65, while the TF-IDF baseline was competitive with, and on raw accuracy slightly ahead of, the tabular machine learning champion. For the regression task, only machine learning models were evaluated, since the deep learning architectures used here were designed for classification rather than continuous prediction, and the Random Forest champion reached a respectable point-estimate R-squared of 0.85 on the small test set, though Section 4.7's bootstrap interval for that figure, -0.03 to 0.95, is wide enough that it should be treated as encouraging rather than conclusive.

On RQ2, fine-tuning did not uniformly help. It left the machine learning category result completely unchanged, since XGBoost had already reached a perfect validation and test score at the search stage and had nowhere further to improve, and it produced only small, mixed changes for the machine learning high_download and regression champions. For the deep learning models, however, fine-tuning consistently reduced test performance on both classification tasks, most sharply for the category TextCNN, whose test macro-F1 fell from a perfect 1.00 to 0.73. Given how small the combined training-plus-validation set is for these neural models, only 111 examples, this is a plausible and unsurprising outcome rather than a sign that something in the pipeline is broken: a small neural network fine-tuned for additional epochs on so few examples, using a stopping and threshold-tuning signal drawn from the same data it is training on, is likely to add variance rather than genuine improvement.

On RQ3, the answer is now a measured one rather than an informed guess. The keyword-leakage ablation reported in Section 4.6 removed all twenty-five keyword indicators from the category model and re-ran the identical search-and-fine-tune protocol. Test accuracy fell from a perfect 1.00 to 0.80, and macro-F1 fell from 1.00 to 0.74, on the main split, and repeating this comparison across four further independent splits held the accuracy gap between 0.20 and 0.25 throughout, so this is not an artifact of one favorable partition of the data. That gap is the most direct estimate this study can offer of how much the original result was inflated by keyword features that are close to synonyms of the category labels. It is a large and stable effect, and it confirms the caution raised throughout Section 4 was warranted. The lightweight TF-IDF baseline in Section 4.2 sharpens this finding rather than complicating it: a generic bag-of-words model with no hand-built keyword features reached the same perfect score, which shows the leakage was never really about the specific choice to engineer keyword flags, it is a property of the raw report text itself, where category names or close variants of them are simply often present in the title or description. But it is not the whole story: 0.70 to 0.80 accuracy on a five-class task using only eight non-keyword numeric features and a file-extension flag is still well above chance, which suggests the non-keyword metadata, publication timing and document characteristics in particular, does carry some genuine, non-tautological signal about a report's category, just far less than the original

headline number implied. The high_download task, which has no equivalent keyword-to-label shortcut in its feature set, produced a broadly similar order of accuracy, 75 percent, without needing any ablation to explain it, and that similarity is itself a useful cross-check on the ablation's finding.

On RQ4, the practical implication follows directly from RQ1 through RQ3 together. Classical machine learning models, and Logistic Regression in particular, appear to be a perfectly reasonable starting point for a small public-sector document portal like CAGD's, without any need for the added complexity of a neural text model, at least at this sample size. The high_download task's more modest, believable result, and the ablated 0.80 accuracy on the category task once keyword features are removed, are a more useful benchmark of what to expect from this kind of pipeline in practice than the original, keyword-inflated category score. Any deployment of automated category tagging on a live CAGD portal, based on this pipeline as it currently stands, should be planned around roughly 0.80 accuracy rather than the original 1.00, since the ablation shows directly that much of the higher number came from keyword features that will not always be present, or will not always mean the same thing, for a genuinely new or unusually worded report title.

5.2 Practical Implications for CAGD and Similar Portals

A few practical points follow for CAGD or a similar public financial management agency considering an automated tagging or popularity-prediction tool. First, a classical model like Logistic Regression or Random Forest, trained on simple engineered features, is a sensible starting point given how small this corpus currently is; there is no evidence here that a more complex neural text model would add value at this sample size, and the high_download results suggest the opposite. Second, download-popularity prediction, unlike category classification, does not have an obvious feature-leakage shortcut available in this dataset, so the roughly 75 percent accuracy and 0.80 ROC-AUC reported here are a more trustworthy estimate of what this kind of model can currently offer for identifying which new reports are likely to attract public attention. Third, before any category-tagging model from this pipeline is used to automatically label new reports on a live portal, the keyword-ablation check recommended below should be run, since keyword features that closely track category names will not be available, or will not mean the same thing, for a genuinely new or unusually worded report title. Fourth, classical, interpretable models are not just easier to trust here, they are also easier to explain to a non-technical audience inside CAGD; other recent applications of classical machine learning to financial and market data have paired simple classifiers with interpretability tools such as Shapley values specifically so that the reasons behind a prediction can be communicated to decision-makers rather than treated as a black box (Stawarz, 2025), and the same pairing would suit a public financial management agency that needs to justify, not just produce, its automated classifications.

5.3 Limitations

Several limitations should be kept firmly in mind when interpreting these results. The corpus is very small, only 131 reports, and the test partition for each task contains only 20 reports. Section 4.7 quantifies exactly how much uncertainty that small size introduces: the bootstrap confidence intervals

reported there are wide enough, for the high_download ROC-AUC and the regression R-squared in particular, that the point estimates in Section 4 should be read as this study's best single guess rather than a precise measurement. The original eight-category classification problem was reduced to five classes by grouping every category with fewer than eight members into an "Other" bucket, a necessary adaptation for a corpus this small and imbalanced, but one that means the category results reported here answer a somewhat easier question than the original eight-class problem would have.

The single train, validation, and test split used throughout this study was stratified only on the category label, not independently on the high_download or log_download_count targets, so the composition of the high_download test partition, 14 low-download and 6 high-download reports, was inherited rather than deliberately balanced. A k-fold cross-validation or repeated resampling scheme, considered in the original project design but not part of the pipeline actually run here, would give a more stable picture of how much these results might vary under a different random split.

Both download-based targets carry a deeper limitation than the split, discussed in Section 3.3: they are built from a single cumulative download count rather than a time-series of access events, so a report's exposure time and its genuine popularity are not cleanly separable in this data. Section 3.3 reports a direct check of this, a weak-to-moderate positive correlation between publication year and log download count, which runs counter to a simple older-reports-accumulate-more story but does not rule out subtler forms of the same confound. A more rigorous treatment would define the popularity target over a fixed observation horizon, for example downloads in the first 90 days after publication, or would model download counts directly as a function of exposure time using a Poisson or negative binomial model with an exposure offset, or would evaluate on a rolling-origin, time-based split rather than a random one. None of that was possible with the single cumulative snapshot mined for this study, and it is named here as a real limitation on how the high_download and log_download_count results should be interpreted, not a solved problem.

As discussed at length in Sections 4.6 and 5.1, the near-perfect category classification results are substantially inflated by keyword features that closely mirror the category labels, and the ablation reported in Section 4.6 quantifies that inflation directly, a drop from 1.00 to 0.80 test accuracy once those features are removed on the main split, and a stable 0.20 to 0.25 point gap across five independently drawn splits. That five-split check reduces, though it does not eliminate, the concern that this finding depends on one particular partition of the data; the without-keyword accuracy itself still ranged from 0.70 to 0.80 across those five splits, which is a reminder that even the more trustworthy, ablated estimate of category performance carries real uncertainty at this sample size. Section 4.2's lightweight TF-IDF baseline reaching the same perfect score without any hand-engineered keyword features at all further clarifies that the leakage sits in the raw report text itself, not in a particular choice of feature engineering, which somewhat narrows what future feature-engineering changes alone could fix. The file_size feature that was intended to inform the machine learning models had no observed values anywhere in this corpus and so contributed nothing to any of the trained models, despite being part of the original engineered feature set. The self-supervised denoising-autoencoder pretraining stage

and the Vision Transformer for cover-image classification, both specified in the original project design, were not part of the pipeline actually run for the results reported here, the former because it was not called during training, the latter because the mined corpus contained no usable cover-image URLs.

Finally, fine-tuning behaved inconsistently for the deep learning models, and given how few combined training-plus-validation examples were available, 111 reports, that instability should be read as a property of fine-tuning a small neural model on a very small dataset, rather than as a general finding about fine-tuning as a technique. The deep learning models were also given comparatively little to work with: a 128-character window covering title and description together, chosen because reports in this corpus have short titles and descriptions and that window comfortably covers nearly all of them, and a modest hyperparameter search of two configurations across three seeds. Whether a longer character window, word-level tokenization, or a small pretrained language model would close the gap with the machine learning models on the high_download task is a real, open, and directly testable question that this study leaves for future work rather than answering here.

5.4 Future Research Directions

Several extensions follow naturally from the limitations above. The keyword-leakage ablation was already repeated across five independent splits in Section 4.6 and produced a stable gap, so the most useful next step is no longer to check whether that gap is a one-split artifact, but to grow the underlying corpus, since even the ablated, more trustworthy accuracy estimate still varied from 0.70 to 0.80 across those five splits, a range that will only narrow with more labeled reports. Expanding the corpus by continuing to mine newly published CAGD reports over time would both increase statistical power and allow a genuinely prospective test, evaluating the current models on reports published after this dataset was assembled, which would be a more convincing test of generalization than any split, or repeated split, of the existing 131 reports could offer. Moving from repeated single splits to proper k-fold cross-validation more broadly, or at minimum stratifying the high_download and regression splits independently rather than inheriting them from the category split, would give a more trustworthy picture of how much every result in this study, not just the ablation, might vary under different random partitions.

For the download-based targets specifically, the exposure-time concern raised in Section 3.3 and Section 5.3 points to a concrete next step rather than a vague caveat: redefining high_download and log_download_count around a fixed observation horizon after publication, modeling raw download counts with a Poisson or negative binomial regression that includes an explicit exposure offset for time-since-publication, and evaluating with a rolling-origin, time-based split that trains on earlier reports and tests on later ones, would together give a much more defensible answer to how well these models predict popularity rather than how well they predict age. On the imbalance-handling side, this study used standard SMOTE for the tabular classifiers; more recent oversampling methods that account for the local geometry or cluster structure of the minority class, such as k-means-guided SMOTE (Douzas et al., 2018) or Geometric SMOTE (Douzas & Bacao, 2019), are natural candidates to test on the high_download task's minority class in particular, given how much weaker the model's per-class F1 is there than on the majority class (Section 4.4).

Running the self-supervised denoising-autoencoder pretraining stage that was specified but not exercised in this run, and revisiting the Vision Transformer component once a larger set of cover images becomes available, would complete the originally planned pipeline and might meaningfully help the deep learning models, which currently have far less data to work with than the machine learning models do once SMOTE has augmented the training set. The character-level design used here, a 128-character window with no pretraining, is also a deliberately conservative choice given the corpus size; testing whether word-level tokenization, longer context windows, or a small pretrained language model fine-tuned with strong regularization can close the gap between the deep learning models and the TF-IDF baseline introduced in Section 4.2 is a well-defined next experiment rather than an open-ended one. Finally, given that fine-tuning on this corpus tended to help the already-strong machine learning models very little while adding noise to the smaller neural models, a cleaner nested validation scheme, one where the fine-tuning stopping and threshold decisions are never drawn from the same data used to fine-tune the model, would help separate a genuine fine-tuning benefit from an artifact of this study's very small sample size. As a last point, the dataset, code, and full results behind this study are documented in enough detail to be reproduced or extended by others, in keeping with a broader push across the social science and medical research literature toward publishing the underlying data behind published findings rather than the findings alone (Stacy et al., 2025), and any future version of this work should keep that same standard of openness.

5.5 Conclusions

This study compared classical machine learning and character-level deep learning models for classifying CAGD report categories, predicting download popularity, and forecasting log download counts, on a small, carefully documented corpus of 131 official Ghanaian public financial management reports. Classical machine learning models performed at least as well as, and in the case of the high-download task noticeably better than, the character-level deep learning models tested here, and fine-tuning on the combined training and validation data helped the machine learning models very little while consistently reducing test performance for the deep learning models. The near-perfect category classification result, while numerically striking, is driven substantially by keyword features that closely mirror the category labels themselves, and a lightweight bag-of-words baseline with no engineered features at all reaches the same score, showing the effect sits in the raw report text rather than in a particular feature-engineering choice. A dedicated ablation confirmed the scale of this directly: removing those keyword features dropped test accuracy from 1.00 to 0.80 and macro-F1 from 1.00 to 0.74, consistently across five independent data splits, while still leaving a result well above chance, showing that the report metadata carries some genuine signal about category alongside the larger, keyword-driven inflation. The more modest, more believable download-popularity result, and this ablated 0.80 accuracy figure, are the more trustworthy findings of this study, and are the numbers that should guide expectations for what this kind of pipeline can currently do. Taken together, these results support a specific, practical recommendation: classical machine learning is a sensible starting point for automated tools on a small public-sector document portal like CAGD's, but any category-tagging model built from this pipeline should be planned around its ablated, keyword-free performance rather than its keyword-inflated

headline score, and the whole comparison should be revisited once the corpus has grown, before it is relied on for a live deployment.

Declarations

Author Contribution

Conceptualization, E.O-D. and G.O.F.; methodology, G.O.F.; software, G.O.F.; validation, E.O-D. and G.O.F.; formal analysis, E.O-D. and G.O.F.; investigation, E.O-D. and G.O.F.; resources, E.O-D. and G.O.F.; writing—original draft preparation, E.O-D. and G.O.F; writing—review and editing, E.O-D. and G.O.F. ; supervision, G.O.F. All authors have read and agreed to the publication of the manuscript.

Data Availability

The datasets and code snippets used for data analysis and visualisation are available in the figshare repository at: https://figshare.com/s/21195f9f80d92d9e1903.

References

1. Abugbilla, G. (2025). Machine learning applications in customer relationship management: A comprehensive review. *Journal of Computer Science and Information Technology*, *2*(1), 41–49. https://doi.org/10.61424/jcsit.v2i1.402
2. Afriyie, Y., & Weyori, B. A. (2026). Sentiment analysis based on deep learning approaches for text classification. *Journal of Electrical Systems and Information Technology*, *13*(1), Article 32. https://doi.org/10.1186/s43067-026-00322-4
3. Agbesi, V. K., Chen, W., Yussif, S. B., Ukwuoma, C. C., Gu, Y. H., & Al-Antari, M. A. (2024). MuTCELM: An optimal multi-TextCNN-based ensemble learning for text classification. *Heliyon*, *10*(18), Article e38515. https://doi.org/10.1016/j.heliyon.2024.e38515
4. Agyemang, E. F., Mensah, J. A., Nyarko, E., Arku, D., Mbeah-Baiden, B., Opoku, E., & Nortey, E. N. N. (2025). Addressing class imbalance problem in health data classification: Practical application from an oversampling viewpoint. *Applied Computational Intelligence and Soft Computing*, *2025*(1), Article 1013769. https://doi.org/10.1155/acis/1013769
5. Alejo-Sanchez, L. E., Márquez-Grajales, A., Salas-Martínez, F., Franco-Arcega, A., López-Morales, V., Acevedo-Sandoval, O. A., González-Ramírez, C. A., & Villegas-Vega, R. (2025). Missing data imputation of climate time series: A review. *MethodsX*, *15*, Article 103455. https://doi.org/10.1016/j.mex.2025.103455
6. Apaflo, D. A., Buabin, W. K. A., Anim, B., & Magdalene, Y. M. (2026). *Artificial intelligence for enhancing audit quality in public companies*. Zenodo. https://doi.org/10.5281/zenodo.19949200

7. Asante, A., & Hajek, P. (2025). Beyond trolling: Fine-grained detection of antisocial behavior in social media during the pandemic. *Information*, *16*(3), Article 173. https://doi.org/10.3390/info16030173
8. Ayintareba, A. E., Korda, D. R., & Owusu-Boateng, O. (2026). Performance metrics in machine learning: A comprehensive narrative review. *Journal of Advances in Mathematics and Computer Science*, *41*(5), 183–199. https://doi.org/10.9734/jamcs/2026/v41i52145
9. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(1), 281–305.
10. Bohacek, M., & Bravansky, M. (2024). When XGBoost outperforms GPT-4 on text classification: A case study. In *Proceedings of TrustNLP 2024* (pp. 51–60). Association for Computational Linguistics.
11. Bosson-Amedenu, S., Baah, E. M., Addor, J. A., Ayiah-Mensah, F., Turkson, A. J., & Oduro-Okyireh, T. (2026). Assessing cognitive growth in mathematics using multinomial logistic regression within Bloom's taxonomy in higher education. *Discover Education*, *5*(1), Article 1205. https://doi.org/10.1007/s44217-026-01205-6
12. Bouthillier, X., Laurent, C., & Vincent, P. (2021). Unreproducible research is reproducible. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (Vol. 139, pp. 725–734). PMLR.
13. Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324
14. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953
15. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785
16. Dake, D. K. (2026). Cluster pattern analysis of students stress using machine learning algorithms with feature engineering. *International Journal of Modern Education and Computer Science*, *18*(2), 147–162. https://doi.org/10.5815/ijmecs.2026.02.09
17. Dake, D. K., & Gyimah, E. (2025). Using machine learning algorithms to cluster and analyse students' acceptance of mobile learning management systems. *Informing Science: The International Journal of an Emerging Transdiscipline*, *28*, Article 24. https://doi.org/10.28945/5596
18. Dankwah, H. A. K., & Enyejo, J. O. (2026). Design of an AI-driven adaptive compliance optimization algorithm for continuous audit readiness in IFRS-based financial reporting systems. *International Journal of Scientific Research in Science, Engineering and Technology*, *11*(6), 593–618. https://doi.org/10.32628/ijsrset2512551
19. Dey, D., Haque, M. S., Islam, M. M., Aishi, U. I., Shammy, S. S., Mayen, M. S. A., Noor, S. T. A., & Uddin, M. J. (2025). The proper application of logistic regression model in complex survey data: A

systematic review. *BMC Medical Research Methodology*, *25*(1), Article 15. https://doi.org/10.1186/s12874-024-02454-5

20. Douzas, G., & Bacao, F. (2019). Geometric SMOTE: A geometrically enhanced drop-in replacement for SMOTE. *Information Sciences*, *501*, 118–135. https://doi.org/10.1016/j.ins.2019.06.007
21. Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, *465*, 1–20. https://doi.org/10.1016/j.ins.2018.06.056
22. Ghanem, V. G. (2026). Spatial and machine learning analysis of district-level health insurance inequities in Ghana. *Cureus*, *18*(1), Article e101984. https://doi.org/10.7759/cureus.101984
23. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
24. Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67. https://doi.org/10.1080/00401706.1970.10488634
25. Hu, S., Oppong, A., Mogo, E., Collins, C., Occhini, G., Barford, A., & Korhonen, A. (2025). Natural language processing technologies for public health in Africa: Scoping review. *Journal of Medical Internet Research*, *27*, Article e68720. https://doi.org/10.2196/68720
26. Jiménez-Gómez, J. M., et al. (2025). Data science and public sector accounting: Reviewing impacts on reporting, auditing, and accountability practices. *Financial Accountability & Management*.
27. Katere, E., Donkor, N., Ijiyemi, P. O., Ogunsola, O. M., Akomeah, K. B., & Ogundojutimi, O. (2025). AI-driven business modeling and decision support integration for corporate governance. *International Journal of Business Intelligence Research*, *3*(1), 33–55. https://doi.org/10.34218/ijbir_03_01_002
28. Larbi, E. K., Asenso-Gyambibi, D., Peprah, M. S., & Danquah, J. A. (2025). Spatiotemporal assessment of mining impact on land use land cover in Bole District: A performance evaluation of support vector machine (SVM) and random forest (RF) classifiers with future impact prediction. *African Geographical Review*, 1–23. https://doi.org/10.1080/19376812.2025.2524386
29. Li, F., Li, J., & Abza, F. (2025). Sentiment analysis of tweets employing convolutional neural network optimized by enhanced gorilla troops optimization algorithm. *Scientific Reports*, *15*(1), Article 795. https://doi.org/10.1038/s41598-025-85392-6
30. Little, R. J. A., & Rubin, D. B. (2019). *Statistical analysis with missing data* (3rd ed.). John Wiley & Sons. https://doi.org/10.1002/9781119013563
31. Mantey, S., Attipoe, I. S., & Mohammed, T. W. (2026). Evaluating Google AlphaEarth and Landsat 8/9 multisource data using random forest, rotation forest, and canonical correlation forest for enhanced land cover classification. *Nova Geodesia*, *6*(2), Article 650. https://doi.org/10.55779/ng62650
32. Mantey, S., Attipoe, I. S. K., & Lawerteh, J. K. (2026). Boosting algorithm: An ensemble learning tool for land use land cover classification using Google Alpha Earth foundations satellite embeddings. *Advances in Remote Sensing*, *15*(1), 1–22. https://doi.org/10.4236/ars.2026.151001
33. Mettle, H. N., Ayerko, E. H., & Appiahene, P. (2026). Machine learning-based customer churn prediction in telecommunication industry. *Applied Computational Intelligence and Soft Computing*,

2026(1), Article 8496186. https://doi.org/10.1155/acis/8496186

34. Mettle, H. N., Osei, G., Winchester, E. A., & Ayerko, E. H. (2026). Enhancing telecom customer churn prediction through hyperparameter tuning and soft-voting ensemble modeling. *Applied Computational Intelligence and Soft Computing*, *2026*(1), Article 7620084. https://doi.org/10.1155/acis/7620084
35. Ndao, I., Dramé, K., Sambe, G., & Diallo, G. (2025). Annotated tweet data of mixed Wolof-French for detecting obnoxious messages. *Data in Brief*, *60*, Article 111500. https://doi.org/10.1016/j.dib.2025.111500
36. Nunoo, E., & Jaganathan, U. K. (2025). The effect of leadership style on return on equity classification: An artificial neural network approach. *The Economics and Finance Letters*, *12*(4), 654–675. https://doi.org/10.18488/29.v12i4.4612
37. Ocansey, S. A., Agboyi, M., Sackitey, G. L., & Islam, A. K. M. K. (2026). Predicting public transport user honesty: A machine learning approach to lost item returns. *International Journal of Intelligent Systems and Applications*, *18*(2), 85–97. https://doi.org/10.5815/ijisa.2026.02.06
38. OECD. (2025). AI in public financial management. In *Governing with artificial intelligence*. OECD Publishing.
39. Osei-Dwomoh, E., & Forkuo, G. O. (2025). *Digital transformation of public financial management in Ghana and developing economies: A systematic review* (Preprint rs.3.rs-8266703). Research Square.
40. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 1310–1318). PMLR.
41. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
42. Sajid, M., Sanaullah, M., Fuzail, M., Malik, T. S., & Shuhidan, S. M. (2025). Comparative analysis of text-based plagiarism detection techniques. *PLOS ONE*, *20*(4), Article e0319551. https://doi.org/10.1371/journal.pone.0319551
43. Salani, J., & Tapfuma, M. M. (2025). Artificial intelligence transforming the publishing industry: A case of the book sector in Africa. *Frontiers in Research Metrics and Analytics*, *10*, Article 1504415. https://doi.org/10.3389/frma.2025.1504415
44. Shen, Z., Li, H., & Zhang, Y. (2025). The mediating role of brand equity in corporate social responsibility and customer purchase intention based on text-CNN model. *Scientific Reports*, *15*(1), Article 44391. https://doi.org/10.1038/s41598-025-28118-y
45. Stacy, B., Kitzmüller, L., Wang, X., Mahler, D. G., & Serajuddin, U. (2025). Data use in social science and medical articles around the world. *PNAS Nexus*, *4*(6), Article pgaf196. https://doi.org/10.1093/pnasnexus/pgaf196
46. Stawarz, M. (2025). Analysis of global stock market development: Integration of clustering, classification, and Shapley values. *PLOS ONE*, *20*(6), Article e0326809.

https://doi.org/10.1371/journal.pone.0326809

47. Tabase, F., Kumi-Boateng, B., & Yakubu, I. (2026). Prediction of client satisfaction levels with Ghana's land administration system: Using a machine learning approach. *UDS International Journal of Development*, *12*(1), 1255–1258. https://doi.org/10.47740/934.udsijd6i
48. Van Buuren, S. (2018). *Flexible imputation of missing data* (2nd ed.). CRC Press. https://doi.org/10.1201/9780429492259
49. Wartini-Twardowska, J., & Twardowska, N. P. (2025). A twofold perspective on the quality of research publications: The use of ICTs and research activity models. *PLOS ONE*, *20*(1), Article e0308952. https://doi.org/10.1371/journal.pone.0308952
50. Wicaksono, A., & Khodra, M. L. (2024). Evaluating BiLSTM performance with BERT, RoBERTa, and DistilBERT in online bullying news detection. *Jurnal Teknik Informatika*, *18*(2).
51. Yakubu, A. M., & Ofori, E. (2026). Smart choice: Machine learning insight into factors influencing students' programme selection at the tertiary institution. *Scientific Journal of Computer Science*, *2*(1), 107–118. https://doi.org/10.64539/sjcs.v2i1.2026.397
52. Yeboah, M. M., Agyei, E., Anim, B., Nwinyi, P., & Nartey, O. L. (2026). *AI in financial auditing: Addressing U.S. audit failures and strengthening PCAOB/SEC compliance*. Zenodo. https://doi.org/10.5281/zenodo.18188727

Figures

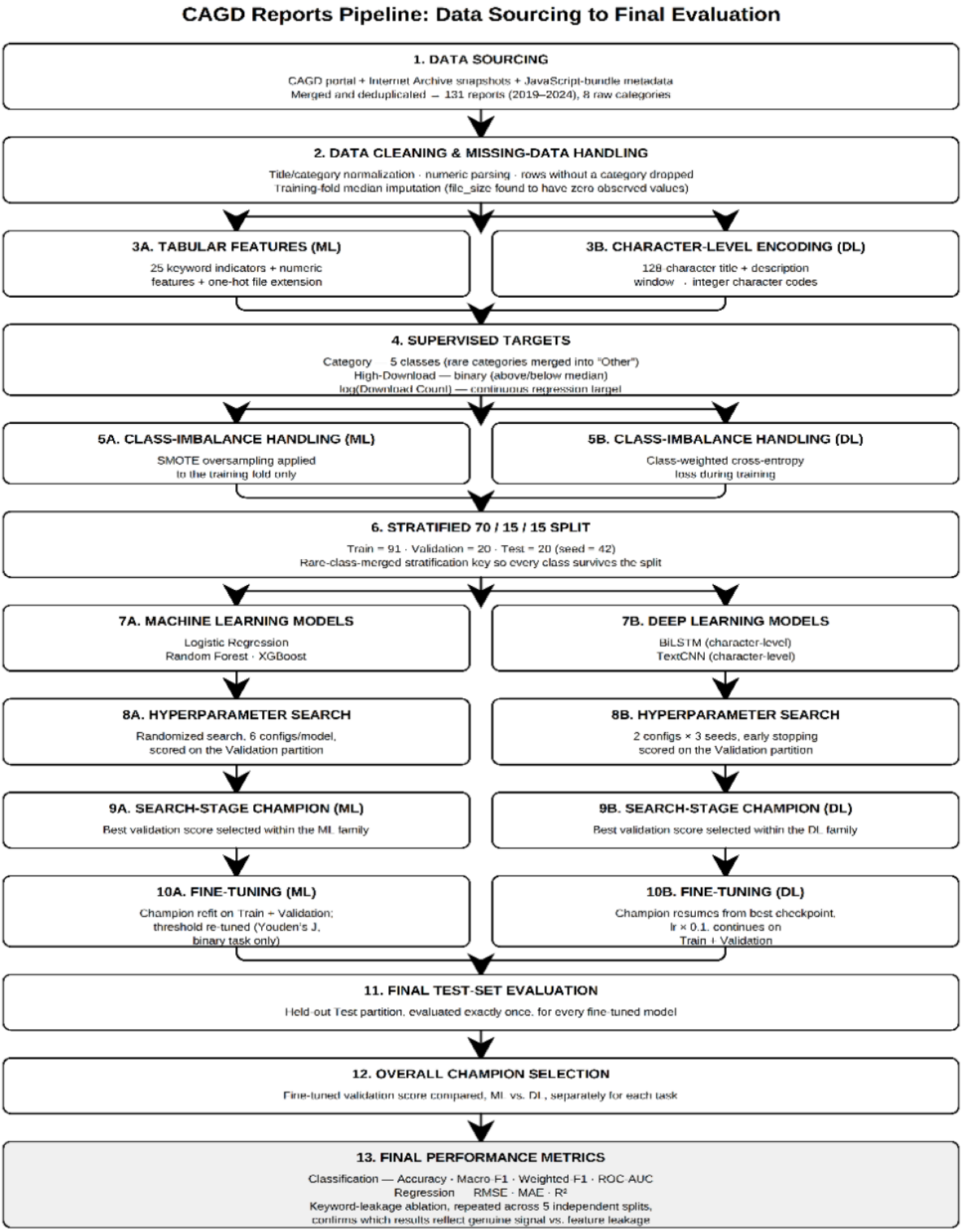


Figure 1

Conceptual pipeline for the CAGD reports study, from data sourcing through feature engineering, class-imbalance handling, the stratified train/validation/test split, model search and fine-tuning for both machine learning and deep learning families, to final test-set evaluation, champion selection, and the keyword-leakage ablation.

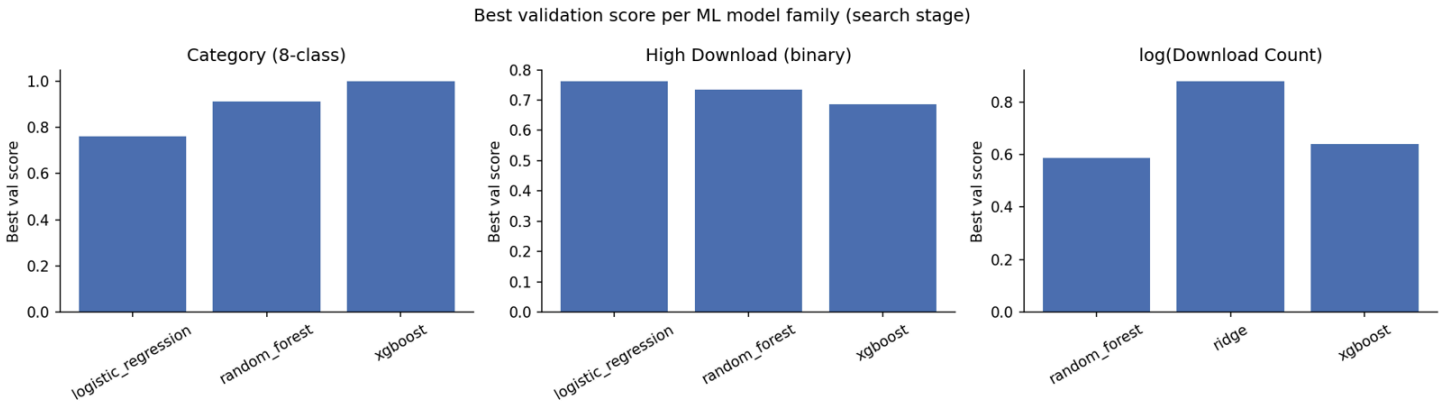


Figure 2

Best validation score reached by each candidate machine learning model, by task (see hpsearch_best_by_model.png).

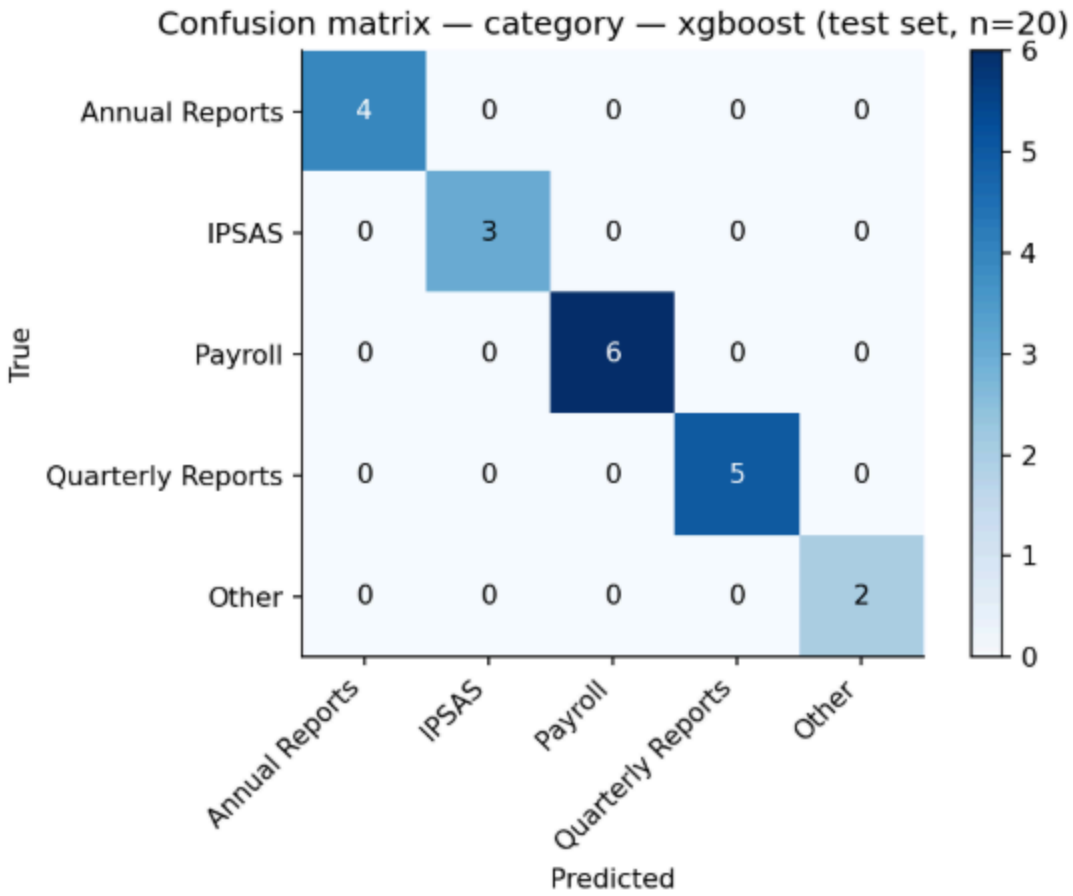


Figure 3

Confusion matrix for the fine-tuned XGBoost category classifier on the test set

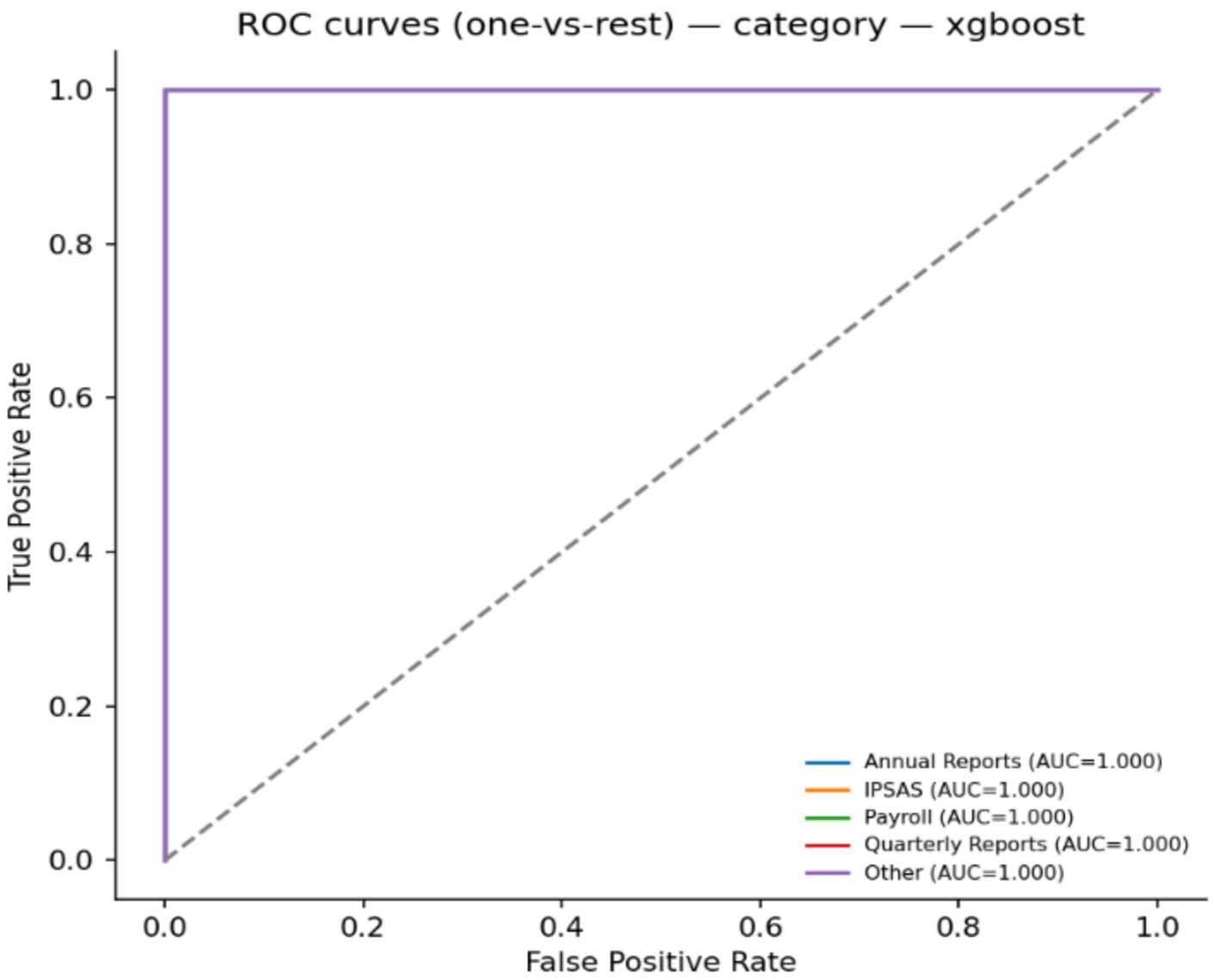


Figure 4

One-versus-rest ROC curves for the fine-tuned XGBoost category classifier on the test set.

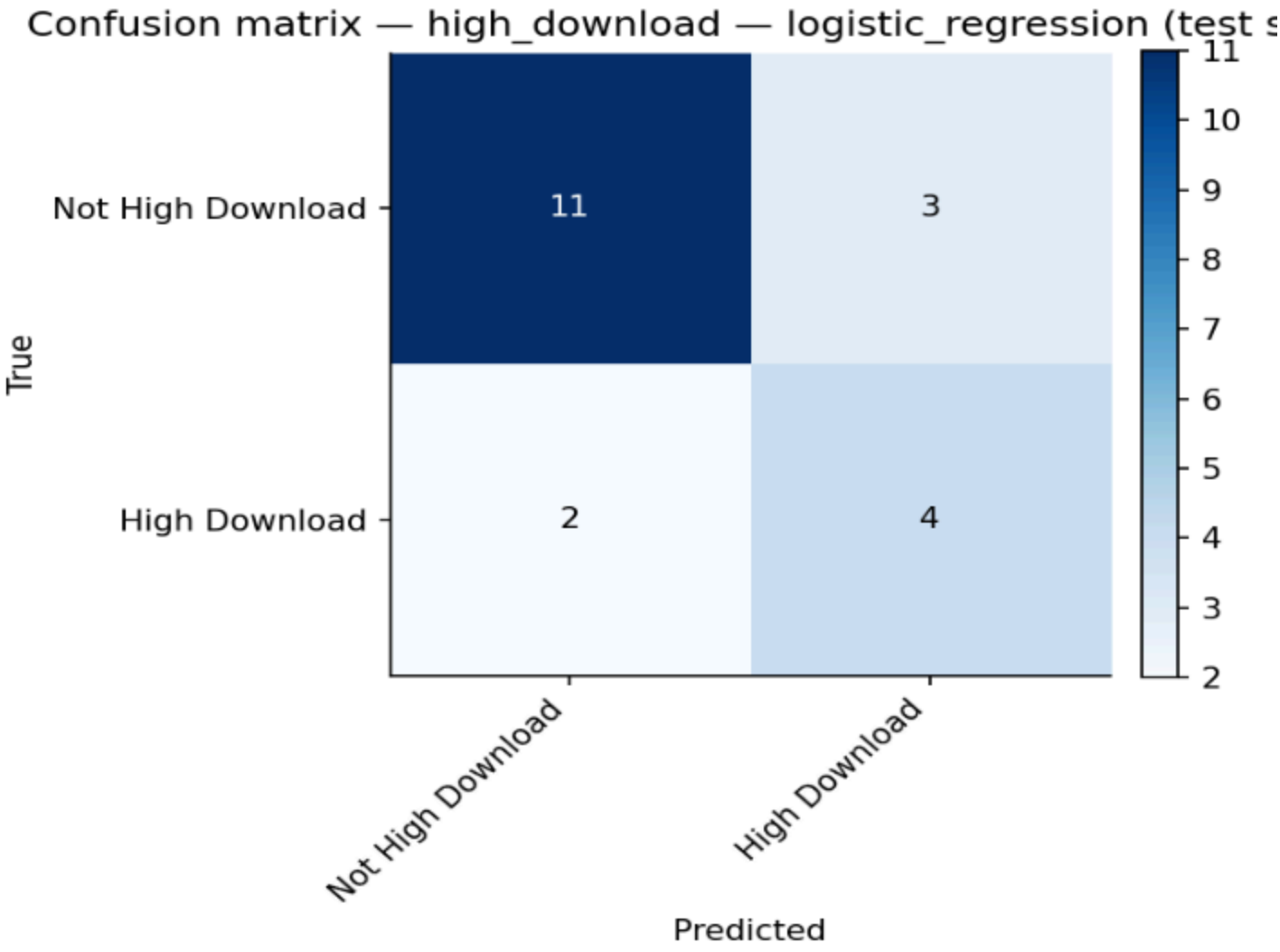


Figure 5

Confusion matrix for the fine-tuned Logistic Regression high_download classifier on the test set.

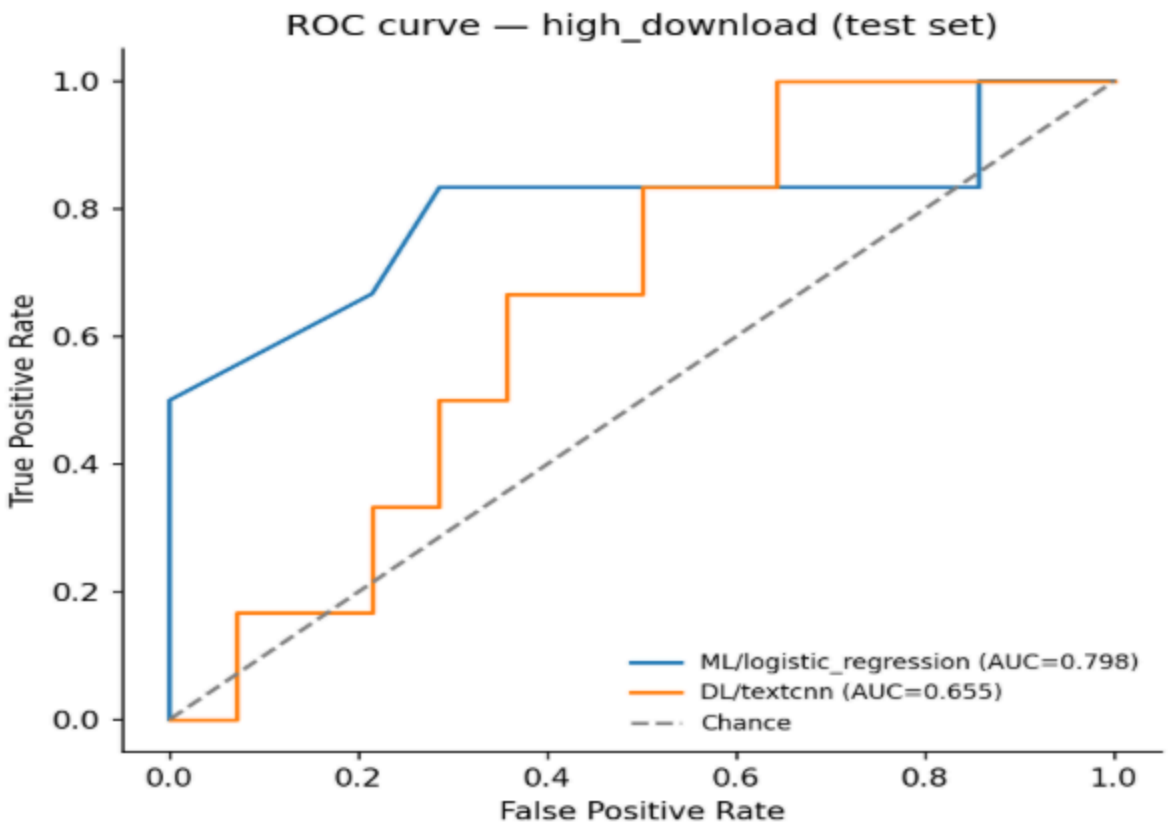


Figure 6

ROC curves for the fine-tuned Logistic Regression and TextCNN high_download classifiers on the test set.

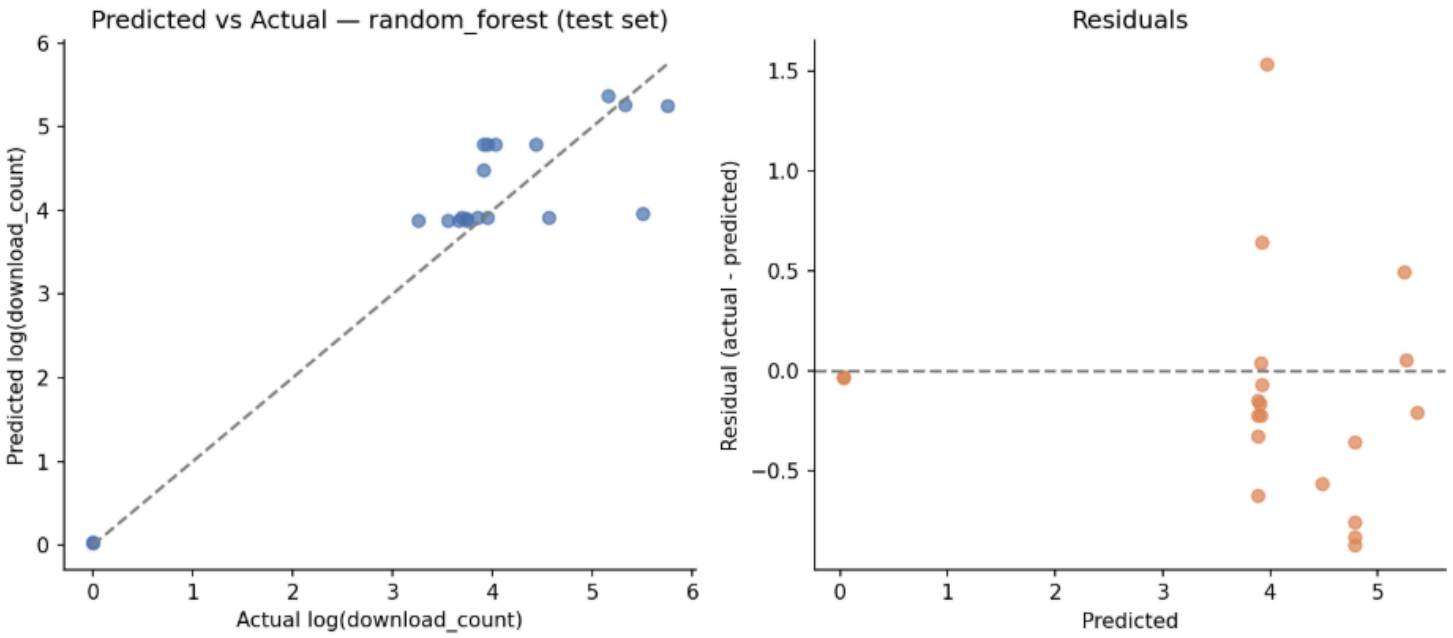


Figure 7

Predicted versus actual log download count and residual plot for the fine-tuned Random Forest regressor

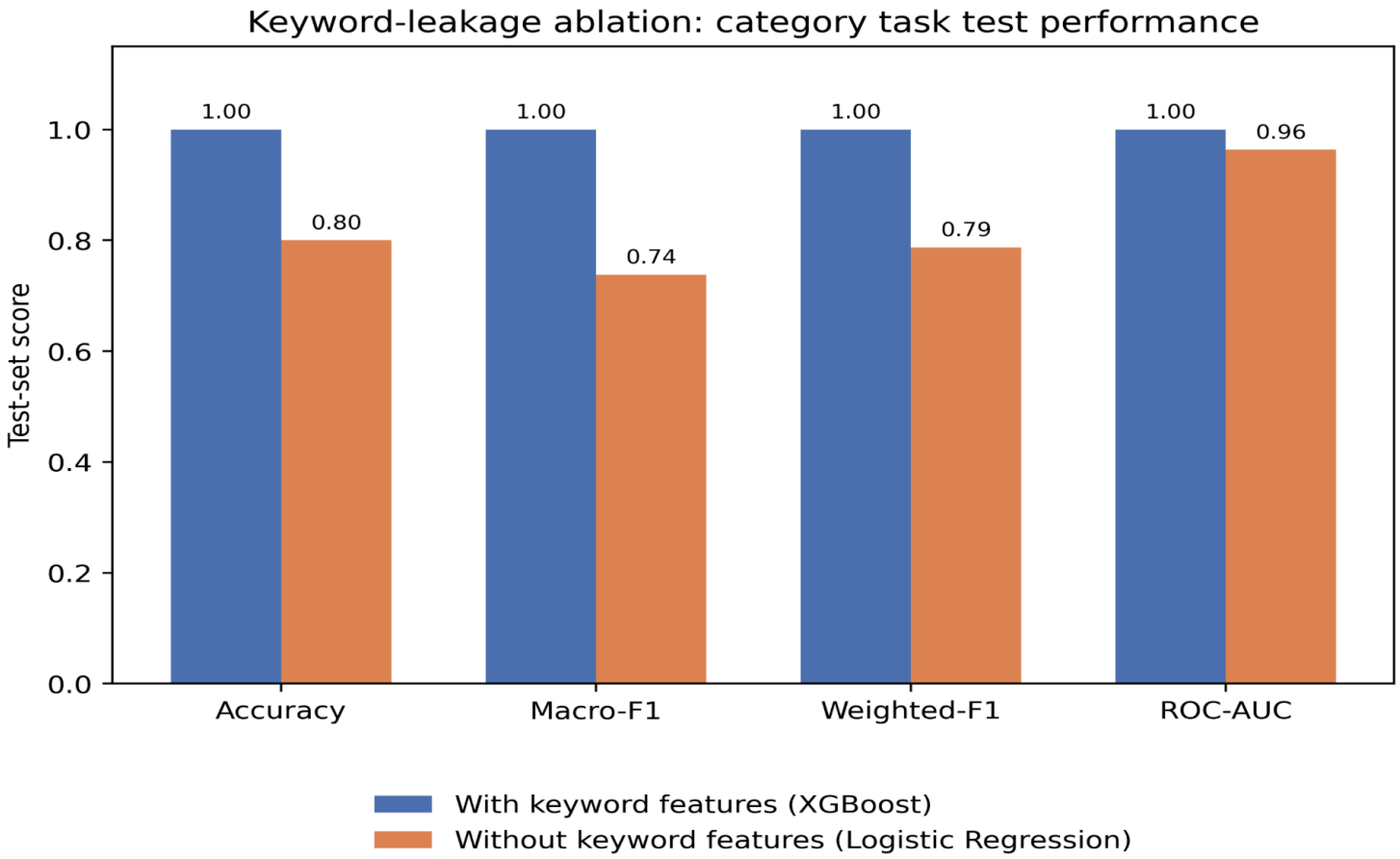


Figure 8

Test-set score by metric, with and without keyword features.

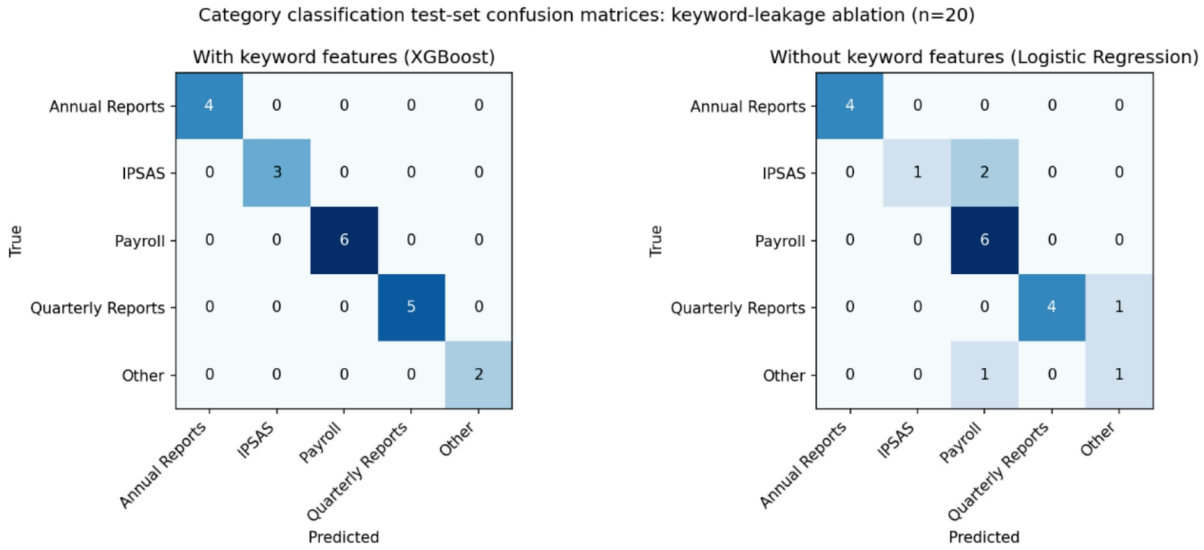


Figure 9

Test-set confusion matrices for the category task, with and without keyword features.

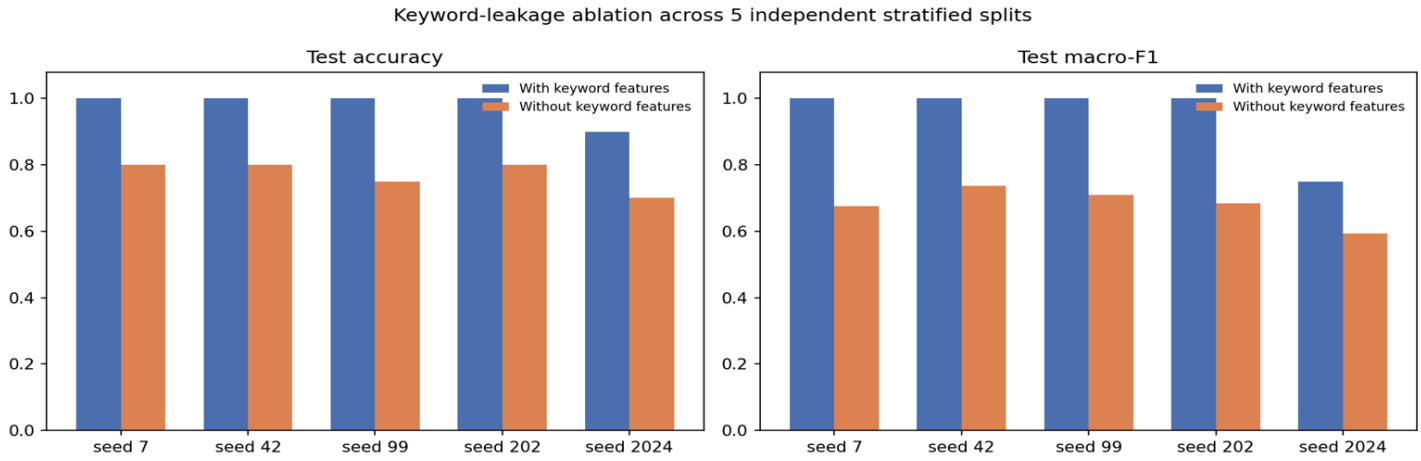


Figure 10

Test accuracy and macro-F1, with and without keyword features, across five independent stratified splits.