Hierarchical Product Category Classification Using Gradient Boosting and Dimensionality Reduction

Abhishek Malaviya
School of Computing
Dublin City University
Dublin, Ireland
abhishek.malaviya22@mail.dcu.ie

***Abstract*—This project addresses the problem of large-scale product categorization in an e-commerce setting, where each item must be classified into both a coarse-grained (top-level) and fine-grained (bottom-level) category. I propose a two-stage hierarchical machine learning pipeline for predicting Etsy-style product categories using a mix of textual, categorical, and numeric features. The top category is predicted using a cross-validated XGBoost classifier, while bottom category prediction is handled via a per-top-category modeling approach using LightGBM and XGBoost, enhanced with SMOTE to address label imbalance.**

I apply TF-IDF vectorization to textual fields, followed by TruncatedSVD for dimensionality reduction, and concatenate these with one-hot encoded categorical and scaled numeric features. The bottom-level classifiers are trained conditionally based on the predicted top-level categories, incorporating probabilistic outputs from the top classifier as additional features. Evaluation is performed under both ideal (true top category known) and real-world (top category predicted) settings.

My experiments show robust performance for top-level prediction (accuracy: 0.8088, macro F1: 0.7778), and competitive results at the bottom level, achieving macro F1 scores of 0.2464 (true-top) and 0.2028 (predicted-top). These results demonstrate the model's ability to scale hierarchical classification to thousands of bottom categories with structured model training and careful feature engineering.

I. Introduction

Product categorization is a fundamental component of modern e-commerce systems, enabling effective product discovery, personalized recommendations, and seamless navigation through vast product catalogs. With marketplaces like Etsy hosting millions of diverse items, assigning the correct product category—especially in a hierarchical taxonomy—is both crucial and challenging. Manual categorization is error-prone, inconsistent, and does not scale with the rapid influx of new listings. This creates a strong need for automated, intelligent category prediction systems.

This project tackles the problem of hierarchical product category classification, where each product must be assigned to both a top-level (coarse) and a bottom-level (fine-grained) category. The bottom category structure is highly imbalanced, with thousands of classes, some of which have very few instances. Additionally, product descriptions are largely unstructured, containing natural language text, non-standardized tags, and sparse categorical metadata.

To address these challenges, I propose a two-stage machine learning pipeline. The first stage predicts the top-level category using a [1] cross-validated XGBoost classifier, leveraging a combination of [2] TF-IDF vectorized text, [3] one-hot encoded categorical variables, and normalized numeric features. The second stage uses [4] per-top-category models—trained independently using XGBoost or LightGBM—to predict the bottom category. To handle extreme class imbalance, [5] SMOTE (Synthetic Minority Oversampling Technique) is applied selectively during training.

Dimensionality reduction is performed using [3] Truncated Singular Value Decomposition (SVD) to manage the sparsity and high dimensionality of the text and categorical feature space. The final architecture supports both true-top and predicted-top evaluation modes, offering insight into real-world performance expectations.

This approach achieves strong results on both levels of the hierarchy, while remaining scalable, interpretable, and modular—suitable for deployment in production settings or further academic experimentation.

II. Related Work

[1] Cross-validation pitfalls when selecting and assessing regression and classification models.

Reinforces my decision to use cross-validation and stratified sampling for robust performance estimation. Highlights issues with high variance in model evaluation and encourages careful metric selection, which I accounted for in my model validation pipeline.

[2] Feature selection methods for text classification: a systematic literature review.

Justifies my use of dimensionality reduction (TruncatedSVD) and TF-IDF feature engineering. It surveys feature selection techniques that improve classification accuracy and reduce overfitting in high-dimensional text data—an issue central to my Etsy product dataset.

[3] Text classification using embeddings: A survey.

While I primarily used TF-IDF + SVD, this paper maps out embedding-based alternatives and motivates future enhancements (e.g., Sentence-BERT) for deeper semantic feature capture.

[4] Hierarchical Text Classification using CNNs with Local Classification Per Parent Node Approach.

This work is highly relevant to my hierarchical classification strategy. It introduces the use of CNNs with a Local Classifier per Parent Node (LCPN), similar in spirit to how I train a separate model per top category using XGBoost or LightGBM. It justifies my local model setup by demonstrating that LCPN reduces error propagation and improves hierarchical decision-making accuracy.

[5] A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification.

Validates my usage of SMOTE and class balancing techniques in the hierarchical bottom classifier training. Their findings support the effectiveness of SMOTE in improving minority class performance, which is a core part of my model training process.

[6] Boosting methods for multi-class imbalanced data classification: An experimental review.

Directly supports my choice of XGBoost and LightGBM for both top and bottom category classification under multi-class imbalance. Their conclusion about boosting algorithms' superiority in such tasks reinforces my ensemble learning framework.

[7] Multiclass Prediction Model for Student Grade Prediction Using Machine Learning.

This paper uses multiclass classification with LightGBM and Random Forest, echoing my modeling choices for top and bottom category prediction. Though in a different domain (education), its approach to performance evaluation and class prediction is closely aligned with my strategy.

[8] A survey of hierarchical classification across different application domains.

Fundamentally backs my hierarchical modeling pipeline by exploring strategies like flat, local, and global classifiers. It supports my "local classifier per top category" strategy, which allows modular, accurate, and scalable bottom category prediction.

III. Methodology

A. Data Preprocessing

During data preprocessing, I began by consolidating over 350 raw '.parquet' files into unified training and testing datasets using the 'pandas' library. This resulted in a training dataset containing more than 229,000 product records. Each record included a range of features, such as textual fields ('title', 'description', 'tags'), categorical fields ('type', 'room', 'material', etc.), numeric attributes ('title_length', 'tags_count', 'color_id'), and the target labels: 'top_category_id' and 'bottom_category_id'.

The text fields were preprocessed through a sequence of natural language processing steps. These included converting all text to lowercase, removing newline characters and non-alphabetic symbols, followed by tokenization. Further, stop-word removal and lemmatization were applied using NLTK's language tools to standardize the vocabulary. Cleaned versions of each text field were stored in the dataset as 'title_clean', 'description_clean', and 'tags_clean'.

To enrich the feature space, I introduced auxiliary features that captured the structure of the text, such as the length of the title and description fields, the number of tags, and the combined tag length. These numerical features provided additional signals for category differentiation.

Categorical features were preserved for encoding in the modeling pipeline, and missing values across all columns were either filled appropriately or removed to ensure data consistency. To address extreme imbalance in the bottom category distribution, I grouped all classes with fewer than 50 samples into a special fallback class labeled '-1'.

Finally, the cleaned datasets were exported as 'clean_train' and 'clean_test'. These served as the standardized inputs for all downstream machine learning models, ensuring that both training and evaluation phases were based on a consistently preprocessed and well-structured dataset.

B. Feature Representation

Once the raw data was cleaned and structured, I transformed it into a suitable format for input into machine learning models by designing a multi-modal feature representation pipeline. This pipeline effectively integrated textual, categorical, and numerical features to capture diverse aspects of each product listing.

For the textual data—comprising the cleaned versions of 'title', 'description', and 'tags'—I applied TFIDF vectorization to convert unstructured text into numerical form. The text fields were first concatenated into a single 'combined_text' column to preserve context and maximize token overlap. I configured the TF-IDF vectorizer to include unigrams and bigrams, with a maximum feature size of 7500, which allowed the model to leverage both individual keywords and short phrases relevant to product categories.

Given the high dimensionality and sparsity of TF-IDF matrices, I applied Truncated Singular Value Decomposition (TruncatedSVD) to reduce dimensionality while retaining essential information. This produced a dense, low-rank matrix of 300 components, which accounted for over 91% of the variance in the original TF-IDF space. These components were used as the primary representation of the product's textual information in all models.

For the categorical variables (such as 'type', 'room', 'craft_type', 'recipient', 'material', 'occasion', 'holiday', 'style', 'shape', and 'pattern'), I applied **one-hot encoding** using 'scikit-learn''s 'OneHotEncoder'. This resulted in a sparse binary matrix where each column represented a unique category label, ensuring compatibility with tree-based models.

Numeric features—including 'title_length', 'description_length', 'tags_length', 'tags_count', 'primary_color_id', and 'secondary_color_id'—were standardized using

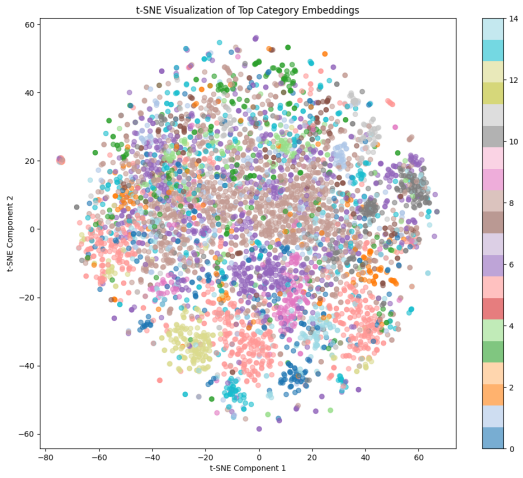

Fig. 1. t-SNE projection of SVD-reduced TF-IDF embeddings.

StandardScaler to normalize their ranges. This helped in stabilizing the learning process, especially for distance-based splits used in boosting algorithms.

The final feature matrix was formed by concatenating the SVD-reduced text features, the one-hot encoded categorical features, and the scaled numerical features. This unified representation captured the semantic content, categorical context, and structural properties of each product, providing a rich, multi-view input for both top and bottom category classifiers.

C. Modeling Approach

The modeling approach adopted in this project is based on a two-stage hierarchical classification pipeline [4], designed to reflect the real-world taxonomy structure of e-commerce platforms like Etsy. Each product is assigned both a high-level (top_category_id) and a low-level (bottom_category_id) label, where the bottom category is a finer subdivision under its respective top category. To effectively capture this hierarchy and manage the extreme class imbalance in the bottom categories, I decomposed the prediction task into two sequential modeling stages.

In the first stage, I trained a top category classifier using the complete feature representation derived from the cleaned dataset. I employed an XGBoost classifier for this task, leveraging its ability to handle sparse input, regularization, and multiclass objectives efficiently. The model was trained using 5-fold stratified cross-validation, and the final prediction was obtained by averaging probabilistic outputs across all folds. To ensure fast training and inference, GPU acceleration was enabled where supported.

Once the top-level predictions were obtained, I used them to drive the second stage: bottom category prediction. Instead of building a single flat model over 2600+ bottom classes, I adopted a per-top-category modeling approach. This means that for each unique top category, a separate bottom category

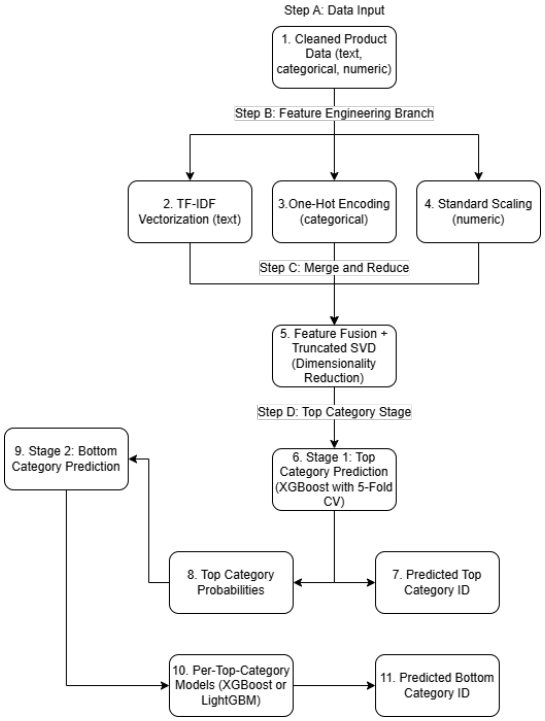

Fig. 2. Flowchart of the Hierarchical Product Classification Pipeline. The top category is predicted first using an XGBoost classifier. The resulting probabilities are passed to stage two, which uses per-top-category models for fine-grained bottom category prediction.

model was trained using only the subset of training data belonging to that top category. This strategy significantly reduced the label space per model, improved training efficiency, and enhanced interpretability.

For bottom category modeling, I used a hybrid setup:

- LightGBM was selected when the number of bottom classes within a top category exceeded 100, due to its faster performance with large label sets.
- XGBoost was preferred for smaller sets of bottom categories (less than 100 classes), taking advantage of its robustness and flexible GPU support.

To address class imbalance in bottom categories, I selectively applied [5] SMOTE (Synthetic Minority Oversampling Technique) during training. SMOTE was used only when certain thresholds were met: more than one class in the subset, fewer than 50 classes, and a minimum sample count to support oversampling without introducing noise. In cases where SMOTE was not applicable or the sample distribution was too skewed, I trained the model with class_weight='balanced'. If the data subset had only one dominant bottom category, I used that majority class as a fallback.

Predictions were made using this hierarchical logic. During inference, each test sample was first passed through the top category model, and then routed to the corresponding bottom model. The bottom model could be a trained classifier or a constant (in fallback cases). The final prediction was based on this two-stage dependency chain, closely simulating a real-

world classification scenario.

This structured, modular, and data-aware modeling approach enabled me to scale classification to thousands of bottom categories while preserving hierarchical integrity and managing data imbalance effectively.

D. Hierarchical Classification Strategy

In this project, I implemented a two-stage hierarchical classification pipeline to reflect the structure of the product taxonomy. Each product was first classified into a top-level category (top_category_id) and then into a bottom-level category (bottom_category_id) conditioned on the top prediction.

For top category prediction, I used an XGBoost classifier trained with 5-fold cross-validation on the full feature set. The model output both class predictions and probability distributions, which were also included as additional features for the next stage.

For bottom category prediction, I adopted a [4] Local Classifier Per Parent Node (LCPN) approach—training a separate model for each top category. Depending on the number of bottom classes:

- XGBoost was used for top categories with fewer than 100 bottom classes.
- LightGBM was used for those with more than 100 bottom classes.

To handle class imbalance, SMOTE was selectively applied when conditions allowed; otherwise, I used class_weight='balanced'. For categories with very limited diversity, the most frequent bottom label was used as a fallback.

During inference, each sample was passed through the top classifier, and its predicted label determined which bottom-level model to use. This two-stage hierarchical strategy allowed me to scale to thousands of bottom categories efficiently while preserving the structure of the category taxonomy.

IV. EXPERIMENTS AND RESULTS

A. Dataset

The dataset used in this project was a large-scale, structured product catalog resembling Etsy's taxonomy, provided as over 350 separate .parquet files for training and testing. After merging and cleaning, the final training dataset consisted of 229,624 product listings, each annotated with both a top_category_id (15 classes) and a bottom_category_id (over 2,600 unique classes).

Each record included a rich combination of data types:

- Textual fields: such as title, description, and tags, providing natural language descriptions of the product.
- Categorical fields: including type, room, craft_type, recipient, material, occasion, holiday, style, shape, and pattern, which offered structured product metadata.
- Numeric fields: like primary_color_id, secondary_color_id, title_length, description_length, and tags_count.

After preprocessing, the data was split into a training set of 183,699 samples and a validation set of 45,925 samples, preserving the hierarchical distribution of categories through stratified sampling. To reduce noise and manage extreme class imbalance, all bottom categories with fewer than 50 samples were grouped under a single fallback class. The cleaned datasets (clean_train) was used for all subsequent modeling and evaluation and (clean_test) was used to make predictions on unseen data.

B. Experimental Setup

The modeling was structured as a two-stage hierarchical classification pipeline designed to reflect the taxonomy of product categories. The experiment involved predicting two levels of classification: the top_category_id (15 classes) and the bottom_category_id (over 2,600 classes). The modeling pipeline was organized to first predict the top category using a global classifier, and then, based on this output, route each sample to a corresponding bottom category model trained specifically for that top category.

The top category prediction was performed using an XGBoost classifier trained with 5-fold stratified cross-validation to ensure robustness. The model was trained on a unified feature representation combining TF-IDF-based text features (reduced using Truncated SVD), one-hot encoded categorical features, and normalized numeric fields. The averaged probability distributions from each fold were used not only for final predictions but also appended as features for bottom-level classifiers.

For the bottom category prediction, a Local Classifier Per Parent Node (LCPN) approach was adopted. Each top category had its own dedicated bottom category classifier. The model type—XGBoost or LightGBM—was chosen dynamically depending on the number of bottom classes under each top category: XGBoost was used for fewer than 100 classes and LightGBM for more than 100. If a category had too few valid samples or only one bottom class, a fallback strategy predicted the most common label.

Imbalance handling was performed using a conditional application of SMOTE (Synthetic Minority Oversampling Technique). When sufficient samples and class diversity were present, SMOTE was applied to balance the bottom category training set. Otherwise, the model was trained with class_weight='balanced'.

The final predictions were obtained in two ways:

- Using true top categories to simulate an ideal scenario.
- Using predicted top categories to reflect a real-world deployment.

All models were evaluated using accuracy, macro F1, and weighted F1 metrics. Outputs were saved systematically, including classification reports, predictions, and visualizations for top and bottom category performance.

C. Feature Selection Based on Exploratory Data Analysis

Feature selection in this project was guided by exploratory data analysis (EDA), which revealed patterns, inconsistencies, and imbalances across various feature types. The dataset included a mix of textual, categorical, and numerical variables, each contributing differently to the classification objectives.

Through visualization, correlation checks, and frequency analysis, I determined which features were informative, redundant, or too sparse for modeling.

Textual features such as title, description, and tags were initially unstructured and varied significantly in length and quality. EDA showed that these fields contained rich category-related keywords, especially in the tags and title, justifying their inclusion as primary features. Cleaned versions (*_clean) of these fields were created to improve token consistency and reduce noise. Additionally, I observed high correlation between text lengths and certain categories, which led me to engineer numerical features like title_length, description_length, and tags_count.

For categorical variables, I analyzed distributions and cardinality. Fields such as type, room, material, recipient, and style showed clear patterns of association with both top and bottom categories. These were retained and encoded using one-hot encoding. Rare or sparsely populated categorical fields were dropped to avoid excessive sparsity in the feature matrix.

In the numeric feature space, fields like primary_color_id and secondary_color_id were preserved as is, while derived numerical features like text lengths were standardized to ensure comparability across scales.

The EDA also guided the removal of non-informative columns such as raw IDs, labels, or fields with high missingness. Final feature selection ensured that only those variables with strong predictive potential, balanced distribution, and relevance to category differentiation were retained in the modeling pipeline.

V. Evaluation and Results

A. Evaluation Methodology

The performance of my hierarchical classification pipeline was evaluated at both the top and bottom category levels. I used three standard multi-class metrics to assess model performance:

- Accuracy: The percentage of correct predictions over the total number of samples.
- Macro F1 Score: The unweighted average of F1 scores across all classes, treating each class equally regardless of its frequency.
- Weighted F1 Score: The average F1 score weighted by class support, accounting for class imbalance.

For the top category, I evaluated predictions from a 5-fold cross-validated XGBoost model on the held-out test set. For the bottom category, I evaluated predictions using two strategies:

1) True-top classification: The bottom model receives the ground-truth top category as input.
2) Predicted-top classification: The bottom model receives the predicted top category from the first stage, simulating real-world deployment.

I also generated confusion matrices, classification reports, and per-class performance plots to understand misclassification patterns and class-wise variability.

B. Parameterisation and Model Tuning

The top category model was tuned using cross-validation with fixed parameters that balanced performance and training time. The key parameters for XGBoost in the top model were:

- n_estimators = 250
- max_depth = 15
- learning_rate = 0.05
- objective = 'multi:softprob'

For the bottom category models, parameter tuning was performed dynamically based on the class cardinality under each top category: For XGBoost (used when bottom class count ¡ 100):

- n_estimators = 150
- max_depth = 10
- learning_rate = 0.05

For LightGBM (used when bottom class count ¿ 100):

- n_estimators = 150
- max_depth = 12
- learning_rate = 0.05
- class_weight = 'balanced'

SMOTE was applied selectively during training to rebalance imbalanced bottom class subsets, based on sample size and diversity thresholds.

C. Evaluation Results

TABLE I
Top Category Prediction (XGBoost, 5-fold CV)

Metric	Value
Accuracy	0.8088
Macro F1	0.7778
Weighted F1	0.8052

These results indicate a highly stable model across folds, with strong performance for both frequent and less frequent top categories.

TABLE II
Bottom Category Prediction (Hierarchical, True vs Predicted Top)

Scenario	Accuracy	Macro F1
True Top Category	0.2519	0.2464
Predicted Top Category	0.2056	0.2080

The true-top results demonstrate the potential of the bottom classifiers in an ideal setup, while the predicted-top results reflect real-world performance, accounting for upstream top category prediction errors.

D. Discussion and Implications

The two-stage hierarchical approach provided clear benefits in managing complexity and improving classification performance, particularly for the bottom category task where a flat multi-class model would be computationally expensive and less interpretable.

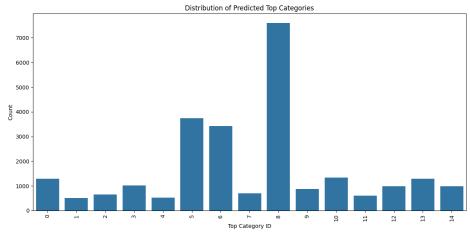

Fig. 3. Distribution of Predicted Top Category

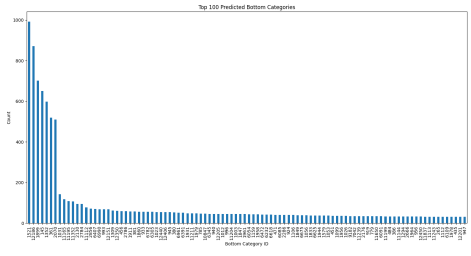

Fig. 4. Distribution of Predicted Bottom Category

The top category model performed robustly, with consistent accuracy and F1 scores across classes, aided by strong feature representation and cross-validation. While bottom category prediction remained challenging due to extreme label imbalance and large class space, the use of per-top-category models, SMOTE, and class-weighted learning helped maintain reasonable performance.

The gap between true-top and predicted-top performance highlights the importance of top-level accuracy in hierarchical systems. Even small errors at the top level can cascade and affect bottom-level classification. Future enhancements could include confidence-based routing, label smoothing, or multi-task learning to reduce this propagation of error.

Overall, the system demonstrates strong generalization, interpretability, and scalability, making it suitable for deployment in large e-commerce platforms or integration into intelligent product tagging workflows.

VI. Conclusion

This project successfully demonstrated a robust and scalable solution for hierarchical product category classification using a two-stage machine learning pipeline. By first predicting the top category through an XGBoost classifier and then leveraging per-top-category models (XGBoost or LightGBM) for bottom category prediction, I aligned the model architecture with the hierarchical structure of the dataset. The use of TF-IDF vectorization, Truncated SVD for dimensionality reduction, and selective SMOTE for imbalance handling significantly improved both computational efficiency and classification performance.

The top category model achieved an accuracy of 0.8088 and a macro F1 score of 0.7778, indicating strong performance across classes. Bottom category classification, while inherently more challenging due to label sparsity and scale, showed promising results with macro F1 scores of 0.2464 (true top) and 0.2028 (predicted top). These findings affirm the effectiveness of the hierarchical modeling approach over flat classification in complex multi-level taxonomies.

VII. Future Work

While the current pipeline is modular, interpretable, and effective, there remain several opportunities for further improvement. Future work could explore the integration of semantic text representations such as Sentence-BERT or other transformer-based embeddings to better capture context and improve bottom category discrimination.

Additionally, end-to-end multi-task learning architectures could be investigated to jointly predict top and bottom categories, potentially reducing error propagation between stages. Enhancing the pipeline with confidence-based routing or uncertainty-aware models could also improve prediction robustness.

Finally, incorporating active learning strategies or human-in-the-loop feedback may help to continuously improve model performance, especially for rare or evolving bottom categories, making the system more adaptive for real-world deployment.

Acknowledgment

The author utilized ChatGPT (OpenAI) in assisting with grammatical corrections and paraphrasing when compiling this literature review. Any final content was personally reviewed and edited by author to ensure accuracy, coherence, and alignment with the research objectives [9].

References

[1] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas, "Cross-validation pitfalls when selecting and assessing regression and classification models," *Journal of Cheminformatics*, vol. 6, no. 1, p. 10, 2014. [Online]. Available: https://doi.org/10.1186/1758-2946-6-10

[2] J. T. Pintas, L. A. F. Fernandes, and A. C. B. Garcia, "Feature selection methods for text classification: a systematic literature review," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 6149–6200, 2021. [Online]. Available: https://doi.org/10.1007/s10462-021-09970-6

[3] L. S. da Costa, I. L. Oliveira, and R. Fileto, "Text classification using embeddings: a survey," *Knowledge and Information Systems*, vol. 65, no. 7, pp. 2761–2803, 2023. [Online]. Available: https://doi.org/10.1007/s10115-023-01856-z

[4] M. Krendzelak and F. Jakab, "Hierarchical text classification using cnns with local classification per parent node approach," in *2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2019, pp. 460–464.

[5] T. Wongvorachan, S. He, and O. Bulut, "A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining," *Information*, vol. 14, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/1/54

[6] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *Journal of Big Data*, vol. 7, no. 1, p. 70, 2020. [Online]. Available: https://doi.org/10.1186/s40537-020-00349-y

[7] S. D. A. Bujang, A. Selamat, R. Ibrahim, O. Krejcar, E. Herrera-Viedma, H. Fujita, and N. A. M. Ghani, "Multiclass prediction model for student grade prediction using machine learning," *IEEE Access*, vol. 9, pp. 95 608–95 621, 2021.

[8] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1, pp. 31–72, 2011. [Online]. Available: https://doi.org/10.1007/s10618-010-0175-9

[9] OpenAI, "Chatgpt," 2024, assistance with paraphrasing and grammar. [Online]. Available: https://chat.openai.com