Publication Venue Prediction Challenge

Koutsomarkos Alexandros[1] and Lakkas Ioannis[2]

[1]p3352106 , [2]p3352110
Emails: akoutsomarkos@aueb.gr , ilakkas@aueb.gr

June 1, 2023

Kaggle Competition

1 Description

The objective of this assignment was to explore and apply machine learning and data mining techniques to tackle a real-world classification problem. The problem involved assigning research papers to predefined categories based on their content. Each research paper was associated with a citation network, along with its abstract, list of authors, and year of publication. The task at hand was to develop a model capable of predicting the class label for each research paper, where the class labels represented the publication venues.

The application of machine learning and data mining techniques in this context has significant relevance in various domains, particularly in recommendation systems. The problem shares similarities with text categorization and node classification tasks, both of which have been extensively studied. The pipeline typically followed to address this problem resembles that of any classification problem, which involves learning the parameters of a classifier using a collection of training papers with known class information and subsequently using the learned model to predict the class labels of unlabeled papers.

2 Dataset

The assignment involves evaluating methods on a classification problem focused on predicting the publication venue of research papers. The provided dataset consists of several files, each containing specific information:

- **edgelist.txt**: Represents the citation network created from papers published in various venues, such as machine learning, computer vision, artificial intelligence, data mining, and natural language processing. The graph is directed, where

nodes represent papers, and edges represent citation relationships. The file contains 166,986 vertices (papers) and 591,017 edges (citations).

- **abstracts.txt**: Contains the abstracts of the 166,986 papers. Each row in the file includes the ID of a paper and its corresponding abstract, separated by the string "‖."
- **authors_v2.txt**: Provides information about the authors of the 166,986 papers. Each row contains the ID of a paper followed by its authors, separated by the string "‖." Multiple authors for a paper are separated by commas (,).
- **year.txt**: Contains the publication year of the 166,986 papers. Each row consists of the ID of a paper and its year of publication, separated by a comma (,).
- **y_train.txt**: Includes the class labels for 35,163 samples (research papers). Each row represents a paper's ID and its associated class label, separated by a comma (,).
- **test.txt**: Contains the IDs of 4,023 papers, which belong to one of the five possible classes. These papers will be used for the final evaluation, and the task is to predict the category of each paper.

3 Models' Evaluation

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

The multi-class log loss metric is used, for the evaluation of the performance of a model. Multi-class log loss, also known as cross-entropy loss, is a commonly used evaluation measure in machine learning tasks where the goal is to classify instances into multiple mutually exclusive classes.

The multi-class log loss metric quantifies the dissimilarity between predicted class probabilities and the true class labels. It takes into account the confidence of the model's predictions and penalizes incorrect classifications based on the deviation from the true class probabilities. A lower log loss value indicates better model performance, with zero representing a perfect match between predicted and true class probabilities.

In this evaluation section, the multi-class log loss is computed by comparing the predicted probabilities of each instance across all classes with the true class labels. The log loss calculation involves taking the logarithm of the predicted probabilities and multiplying them by the corresponding true class labels. The sum of these values

across all instances is then divided by the total number of instances, resulting in the average log loss.

By utilizing the multi-class log loss metric, the evaluation section provides a comprehensive measure of the model's classification accuracy across all classes. It enables the identification of potential areas of improvement and facilitates comparisons between different models or iterations of the same model. Additionally, the log loss metric allows for the detection of overconfidence or underconfidence in the model's predictions, as it captures both the magnitude and direction of errors.

4 Exploratory Data Analysis (EDA)

4.1 Imbalanced Dataset

Dealing with an imbalanced dataset is a common challenge in classification tasks, and the citation network graph dataset is no exception. In this dataset, the number of samples in each class (corresponding to different publication venues) ranges between 3,203 and 9,825, indicating an inherent class imbalance. This class imbalance poses a potential issue as it can affect the performance of machine learning models, particularly those that are sensitive to the distribution of classes. Imbalanced datasets can lead to biased model predictions, where the majority class dominates while the minority classes receive less attention. To address this challenge, we employed appropriate techniques such as class rebalancing methods (e.g., oversampling or undersampling), using different evaluation metrics (beyond accuracy) that consider class imbalances.

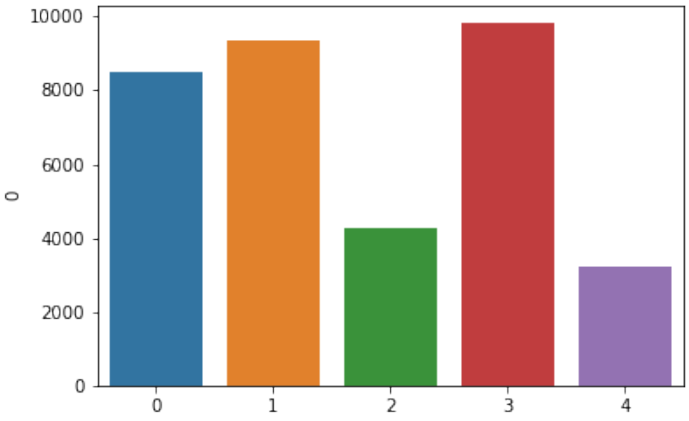

Figure 1: Class distribution.

4.1.1 SMOTE

To address the class imbalance issue in the dataset, we applied the SMOTE (Synthetic Minority Over-sampling Technique) algorithm. SMOTE is a popular technique used to generate synthetic samples of the minority class by interpolating the feature space between existing minority class samples.

The SMOTE algorithm works by randomly selecting a minority class sample and finding its k nearest neighbors. It then creates synthetic samples along the line segments connecting the minority class sample with its neighbors. By introducing these synthetic samples, the algorithm effectively increases the representation of the minority class in the dataset.

In our case, we used the SMOTE algorithm to oversample the minority classes and balance the distribution of samples across all classes. This approach helps prevent bias towards the majority class during model training and improves the performance of classifiers in handling imbalanced datasets.

By applying SMOTE, we were able to mitigate the class imbalance in our dataset, providing a more balanced representation of the different publication venues. This allowed our models to learn from a more representative training set and make more accurate predictions across all classes.

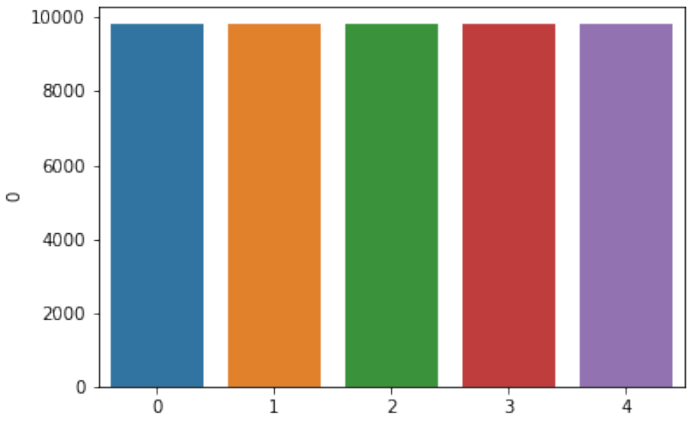

Figure 2: Class distribution after SMOTE .

4.2 Graph

The accompanying graph visually represents a sample (200 nodes) of the citation network of research papers from various publication venues. The nodes in the graph

correspond to individual papers, while the edges represent citation relationships between them. The graph provides a visual depiction of how papers from different or same venues cite each other.

In the graph, the connections between nodes indicate that a paper cites another paper. The directed edges indicate the direction of the citation, with the citing paper represented by the source node and the cited paper represented by the target node. By observing the connections between nodes, we can gain insights into the relationships and influence among papers from different venues.

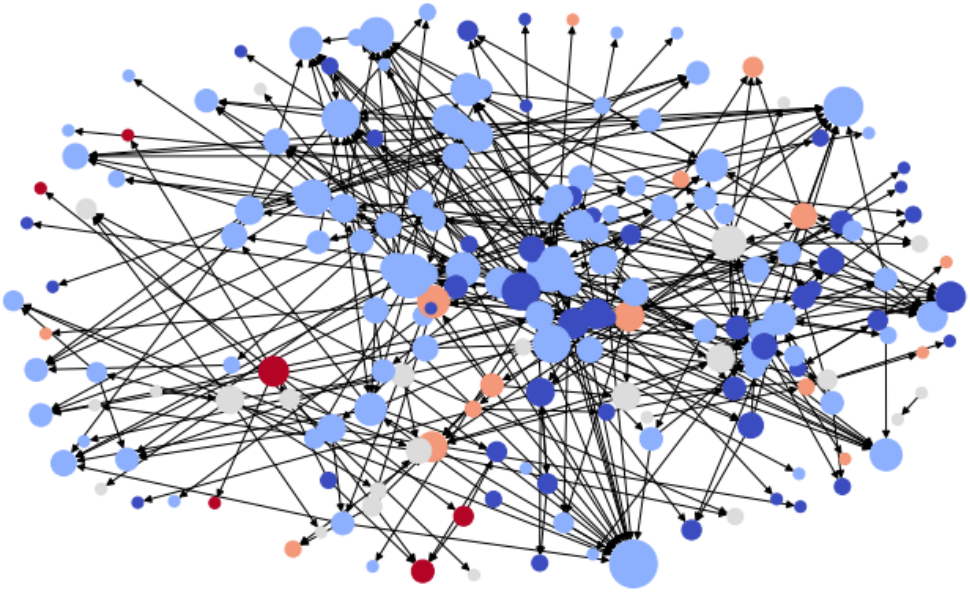

Figure 3: Graph plot which illustrates the relationships (citations) between the papers.

4.3 Year

The years data we used in our study comprised the publication years of the research papers. This information provided temporal context and allowed us to capture potential trends or patterns in the dataset over time.

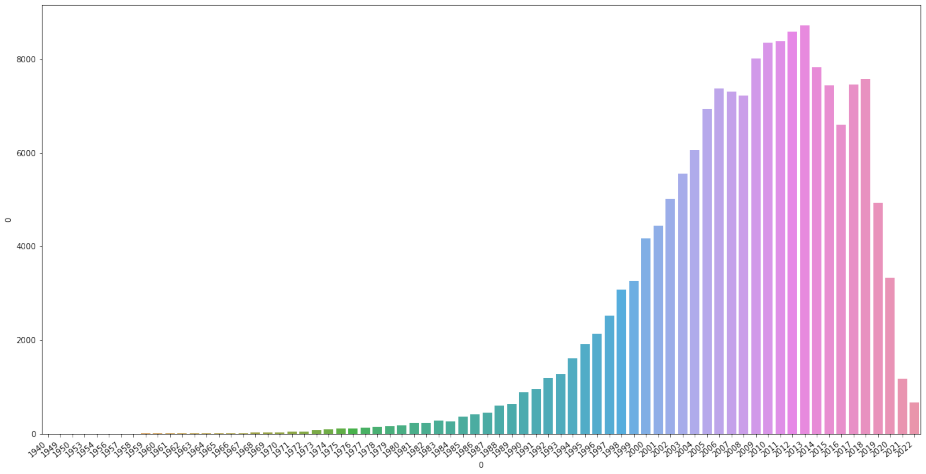

Figure 4: Years bar plot

4.4 SVD

In order to reduce the number of features and mitigate the risk of overfitting in our models, we employed the Singular Value Decomposition (SVD) technique. SVD is a matrix factorization method that decomposes a matrix into three components: U, Σ, and V.

By applying SVD, we aimed to capture the most important information and patterns in the dataset while reducing the dimensionality. The TruncatedSVD variant of SVD was used, which allows us to specify the desired number of components (k) to retain.

During the decomposition process, SVD identifies the singular values and corresponding singular vectors that best represent the original data. The resulting singular vectors represent the transformed features, with each vector capturing a different aspect of the data's variability. We selected a suitable value for k, ensuring that it retained enough information to preserve the essential characteristics of the dataset while reducing the risk of overfitting.

By applying TruncatedSVD, we effectively decreased the number of features in our dataset, making it more manageable and less prone to overfitting.

4.5 Text cleaning

Text cleaning plays a crucial role in improving the performance of the Doc2Vec algorithm, a popular technique for generating document embeddings. We applied the

text cleaning function to paper abstracts before feeding them into the Doc2Vec algorithm. By converting the text to lowercase, removing parentheses and brackets, and eliminating non-word characters, the text cleaning process ensures uniformity in the abstracts' representation. It also removes noise that could potentially mislead the algorithm's learning process. Further steps, such as removing single characters, hyphens, punctuation, and numbers, contribute to creating cleaner and more coherent text. These actions help to focus the algorithm's attention on the meaningful content and semantic relationships within the abstracts. Removing stop words, and applying lemmatization are additional text cleaning steps that enhance the quality of input data for Doc2Vec. These processes improve the algorithm's ability to capture the underlying semantics and context of the abstracts, leading to more accurate document embeddings.In other words the text cleaning that we applied, aids in standardizing the text, reducing noise and irrelevant information, and promoting the extraction of meaningful features. These improvements ultimately enhance the performance of Doc2Vec in generating high-quality document embeddings for downstream tasks, such as our classification problem.

5 Embeddings

Embeddings play a crucial role in capturing meaningful representations of data in various machine learning tasks, including graph and text analysis. In the context of our assignment, embeddings refer to the process of transforming high-dimensional data, such as the citation network graph and research paper abstracts, into lower-dimensional numerical vectors while preserving important relationships and semantics. Graph embeddings aim to capture the structural characteristics of the citation network graph, encoding information about paper interconnections and citation patterns. Text embeddings, on the other hand, focus on capturing the semantic meaning of the research papers' abstracts, allowing us to represent textual information in a more compact and numerical format. By utilizing these embeddings, we can apply a wide range of machine learning and data mining techniques to explore the dataset, uncover patterns, perform similarity-based tasks, and build models that effectively predict the class labels of research papers based on their graph structure and textual content. The use of embeddings enhances the overall understanding and analysis of the dataset, enabling us to make informed decisions and derive valuable insights from the available information.

5.1 Graph embeddings

5.1.1 DeepWalk

In order to obtain graph embeddings, we utilized the DeepWalk algorithm on the citation network graph. DeepWalk is a graph embedding method that leverages the concept of random walks to capture structural information within the graph.

To generate the walks, we employed the generate_walks() function. This function performed random walks on the undirected version of the graph, starting from various nodes. The number of walks and the length of each walk were set to 20 and 50, respectively. By performing these walks, we explored different parts of the graph and collected sequences of nodes as our training data.

Next, we applied the Word2Vec algorithm to the generated walks. Word2Vec is a popular method for learning word embeddings from textual data, but in this case, we adapted it to learn node embeddings from the random walks. We created a Word2Vec model named model1 and trained it on the walks, treating each walk as a sentence and each node within the walk as a word.

During training, the Word2Vec algorithm learned to predict the context (neighboring nodes) given a target node. By doing so, it captured the relationships and proximity between nodes in the citation network graph. The vector size parameter (vector_size) determined the dimensionality of the resulting embeddings, and a window size of 5 indicated that the algorithm considered the five neighboring nodes on either side of the target node.

We also set the minimum count to 1, ensuring that all nodes in the graph were included in the vocabulary. By training the Word2Vec model for 7 epochs and utilizing 8 parallel workers, we optimized the learning process and obtained meaningful representations (embeddings) for each node in the graph.

These graph embeddings encode the structural characteristics and proximity relationships within the citation network. By leveraging the graph embeddings, we gain valuable insights and can make predictions and decisions based on the graph's underlying structure and connectivity.

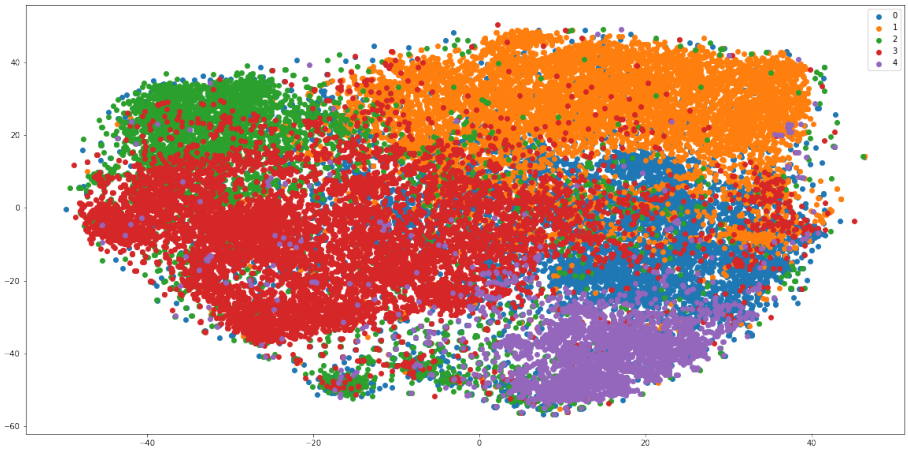

Figure 5: Graph embeddings plot.

5.2 Text embeddings

In order to obtain text embeddings for the research paper abstracts, we employed the Doc2Vec algorithm. This algorithm treats each abstract as a document and learns distributed representations (embeddings) for them. We initialized the algorithm by creating TaggedDocuments, where each document consists of a list of preprocessed words from the abstract, along with a unique identifier corresponding to the paper's node in the citation network. The Doc2Vec algorithm was configured with a vector size of 512, indicating that each abstract would be represented by a 512-dimensional embedding. We set the dm parameter to 0, indicating the use of the DBOW (Distributed Bag of Words) approach rather than the DM (Distributed Memory) approach. Other parameters, such as negative, hs, min_count, workers, and epochs, were also set to specific values to control the training process. By training the Doc2Vec algorithm on the research paper abstracts, we obtained text embeddings that capture the semantic meaning of the abstracts, enabling us to represent textual information in a numerical format suitable for downstream machine learning tasks and analysis.

6 Other features

In our classification task, we expanded the feature set beyond graph and text embeddings to include additional information about the authors of the research papers. These supplementary features provided valuable insights and potential discriminatory

signals, ultimately contributing to a more comprehensive and effective classification framework.

6.1 Authors

To incorporate the information about the authors of the research papers, we utilized author features in our analysis. We generated graph embeddings to capture the relationships between authors in a research network. Actually, we constructed a graph representation using the author and paper information. Each author is added as a node, and edges are created between authors who have collaborated on a paper together. To extract meaningful information from the graph, random walks are performed, simulating the traversal of the graph by taking random paths between authors. This process generates sequences of author nodes. These sequences are then utilized to train a Word2Vec model, which learns embeddings or vector representations for each author. The resulting graph embeddings encode the structural and semantic characteristics of the author collaboration network. In the context of the paper classification task, these graph embeddings could provide valuable insights into the relationships between authors and venues, helping to improve the accuracy of venue prediction models.

7 Modelling

7.1 Neural Network

In this study, we applied a neural network model to tackle the classification problem. The neural network architecture consisted of an input layer, a hidden layer, and an output layer. The input layer's size was determined by the dimension of the input features, which was set to the number of columns in the training dataset. The hidden layer size was set to twice the size of the input layer to allow for more complex representations and feature extraction. The output layer size was determined by the number of unique labels in the training data, in this case 5.

The model was built using the Keras functional API, starting with defining the input layer with the specified shape. A dense layer with a rectified linear unit (ReLU) activation function was then added to the model, comprising 3072 hidden units. A dropout layer with a dropout rate of 0.5 was included to mitigate overfitting and improve generalization.

The output layer was constructed with a softmax activation function to produce probability distributions over the predicted labels. The entire model was instantiated using the Model class from Keras, with the input and output layers specified.

To train the model, the Adam optimizer with a specified learning rate and weight decay rate was utilized. The sparse categorical cross-entropy loss function was chosen as it suited the multi-class classification task. The sparse categorical cross-entropy loss function is a commonly used loss function in multi-class classification tasks. It calculates the negative log-likelihood of the true class label given the predicted probability distribution, effectively penalizing the deviation between the predicted and true class labels. The model's performance was evaluated based on accuracy metrics.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 1536)]	0
dense (Dense)	(None, 3072)	4721664
dropout (Dropout)	(None, 3072)	0
dense_1 (Dense)	(None, 5)	15365

Total params: 4,737,029
Trainable params: 4,737,029
Non-trainable params: 0

Figure 6: Neural network model summary.

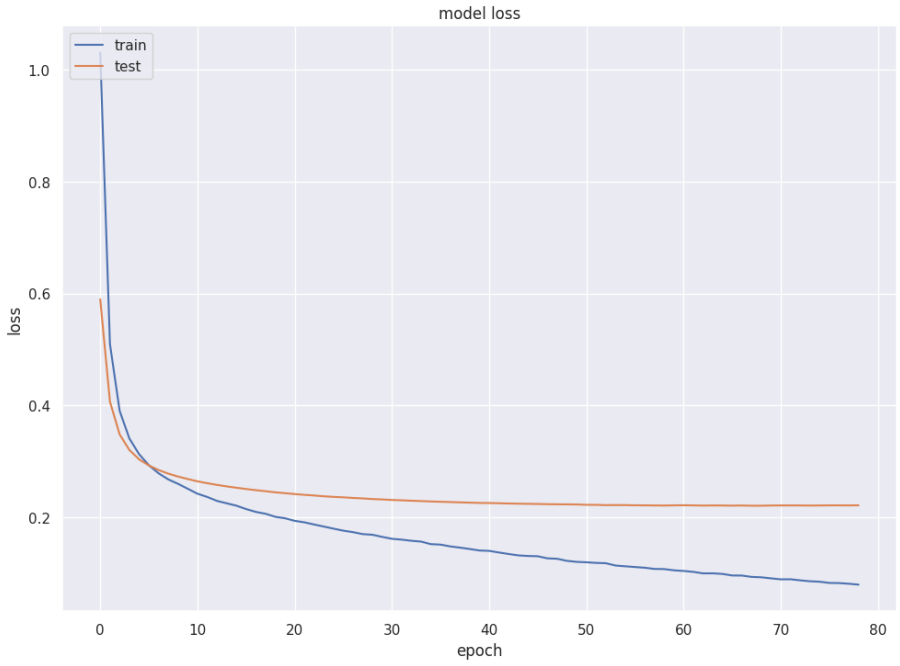

Figure 7: Neural network model history.

```
165/165 [==============================] - 2s 14ms/step
log_loss: 0.221
              precision    recall  f1-score   support

           0       0.88      0.87      0.88      1273
           1       0.94      0.96      0.95      1405
           2       0.93      0.91      0.92       643
           3       0.91      0.92      0.91      1474
           4       0.96      0.95      0.95       480

    accuracy                           0.92      5275
   macro avg       0.92      0.92      0.92      5275
weighted avg       0.92      0.92      0.92      5275
```

Figure 8: Neural network model classification report.

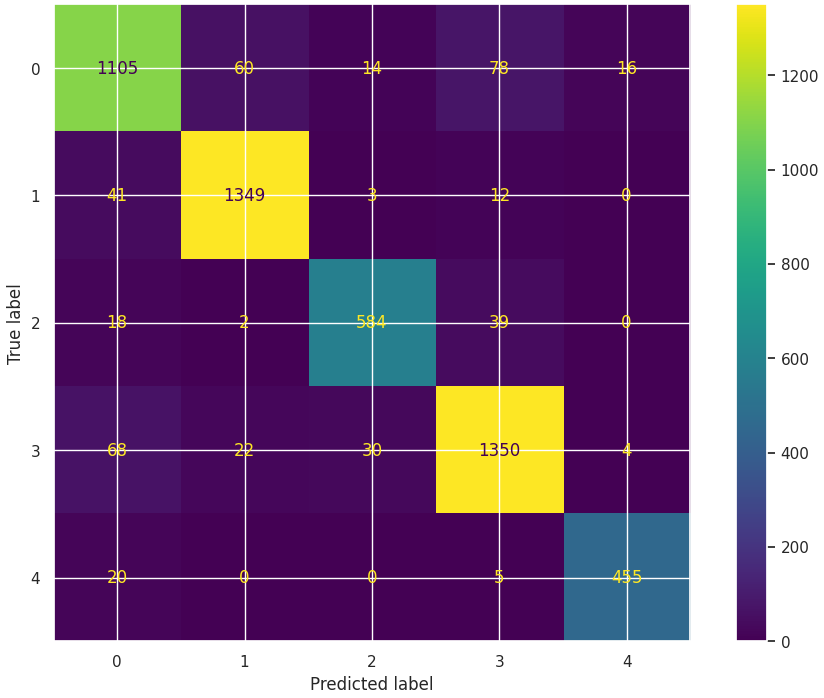

Figure 9: Neural network model confusion matrix.

7.2 LGBM

The LGBMClassifier is an implementation of the LightGBM gradient boosting framework specifically designed for classification tasks. It is a powerful and efficient algorithm that uses a combination of decision trees to create an ensemble model. Additionally, the LGBMClassifier is designed to handle imbalanced class distributions

effectively. It provides options for specifying class weights, allowing the classifier to assign more importance to minority classes and prevent them from being overshadowed by dominant classes. This is particularly beneficial in multi-class classification where class imbalances are common. Moreover, the LGBMClassifier is capable of handling both numerical and categorical features, making it versatile in dealing with diverse types of data. It automatically handles feature transformations and can handle missing values, reducing the need for extensive preprocessing steps. Furthermore, the LGBMClassifier offers high accuracy in multi-class classification tasks. It can learn complex relationships between features and target classes, capturing non-linear patterns and achieving accurate predictions. Overall, the LGBMClassifier's combination of efficiency, ability to handle imbalanced data, versatility, and high accuracy make it a popular choice for multi-class classification problems.

log_loss: 0.249

	precision	recall	f1-score	support
0	0.88	0.86	0.87	1273
1	0.93	0.95	0.94	1405
2	0.92	0.92	0.92	643
3	0.91	0.90	0.91	1474
4	0.93	0.94	0.93	480
accuracy			0.91	5275
macro avg	0.91	0.91	0.91	5275
weighted avg	0.91	0.91	0.91	5275

Figure 10: LGBM model classification report.

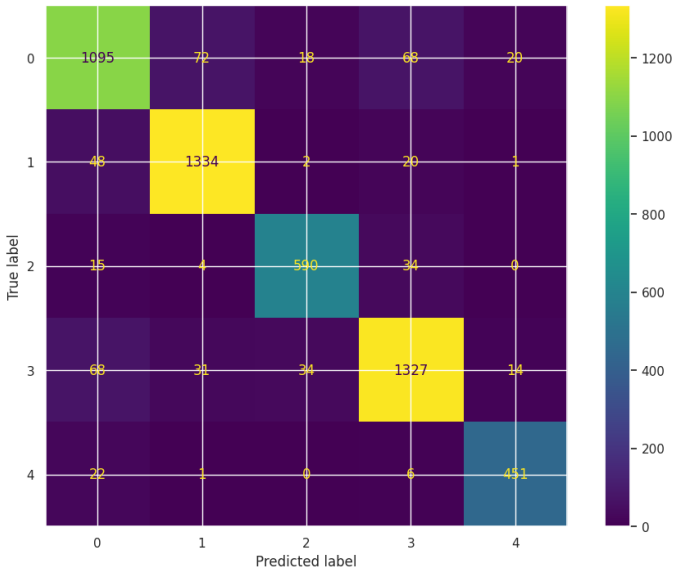

Figure 11: LGBM model confusion matrix.

7.3 XGBM

The XGBoost classifier, represented by the XGBClassifier, is widely used in multi-class classification problems for several reasons. Firstly, XGBoost is known for its excellent performance and high accuracy. It employs a gradient boosting algorithm that effectively combines weak learners to build a strong predictive model. One of the key advantages of XGBoost is its ability to handle imbalanced class distributions.

log_loss: 0.269

	precision	recall	f1-score	support
0	0.88	0.86	0.87	1273
1	0.93	0.95	0.94	1405
2	0.92	0.91	0.91	643
3	0.90	0.91	0.91	1474
4	0.95	0.93	0.94	480
accuracy			0.91	5275
macro avg	0.92	0.91	0.91	5275
weighted avg	0.91	0.91	0.91	5275

Figure 12: XGB model classification report.

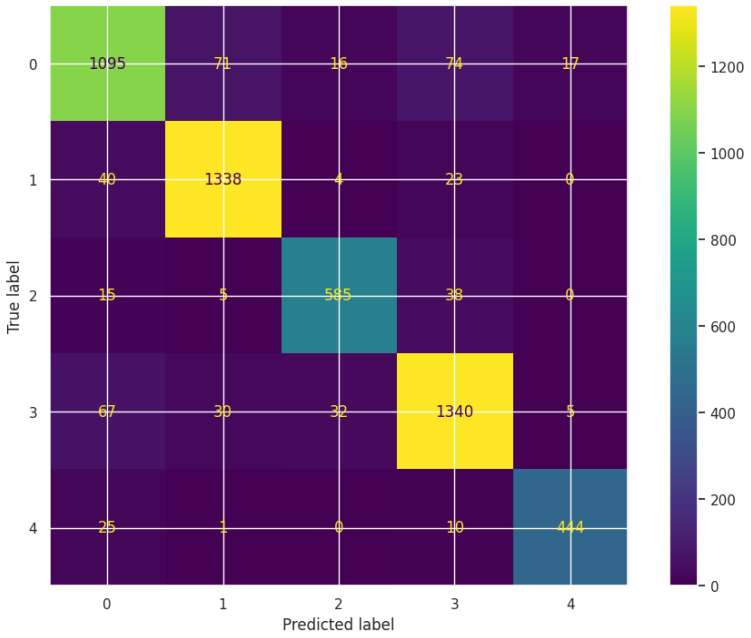

Figure 13: XGB model confusion matrix.

7.4 Ensemble Model

The voting classifier is a powerful ensemble technique commonly employed in classification problems to enhance prediction accuracy. In this study, we applied a voting classifier to combine the predictions of four individual classifiers: LGBMClassifier, XGBClassifier, SVC, and a neural network classifier (the above neural network model). Each classifier contributes to the final prediction through a voting mechanism (In this case, we used the "soft" voting which predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.). We assigned also weights to each classifier, where the LGBM Classifier and the neural network classifier had higher weights, indicating their relatively stronger influence in the voting process. The use of a voting classifier offers several advantages in classification tasks. It leverages the diverse perspectives and modeling capabilities of multiple classifiers, resulting in improved accuracy, enhanced robustness, and better generalization. By combining the predictions from different classifiers, the voting classifier can effectively mitigate biases and errors inherent in any single classifier, leading to more reliable and comprehensive predictions. This ensemble approach has demonstrated its effectiveness in various domains and is widely employed as a go-to technique in classification tasks.

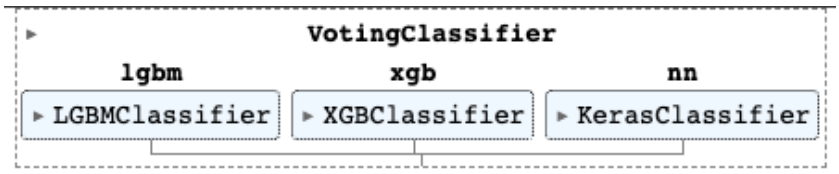

Figure 14: Voting Classifier model summary.

log_loss: 0.218

	precision	recall	f1-score	support
0	0.89	0.87	0.88	1273
1	0.94	0.96	0.95	1405
2	0.92	0.92	0.92	643
3	0.92	0.92	0.92	1474
4	0.95	0.94	0.95	480
accuracy			0.92	5275
macro avg	0.92	0.92	0.92	5275
weighted avg	0.92	0.92	0.92	5275

Figure 15: Voting Classifier classification report.

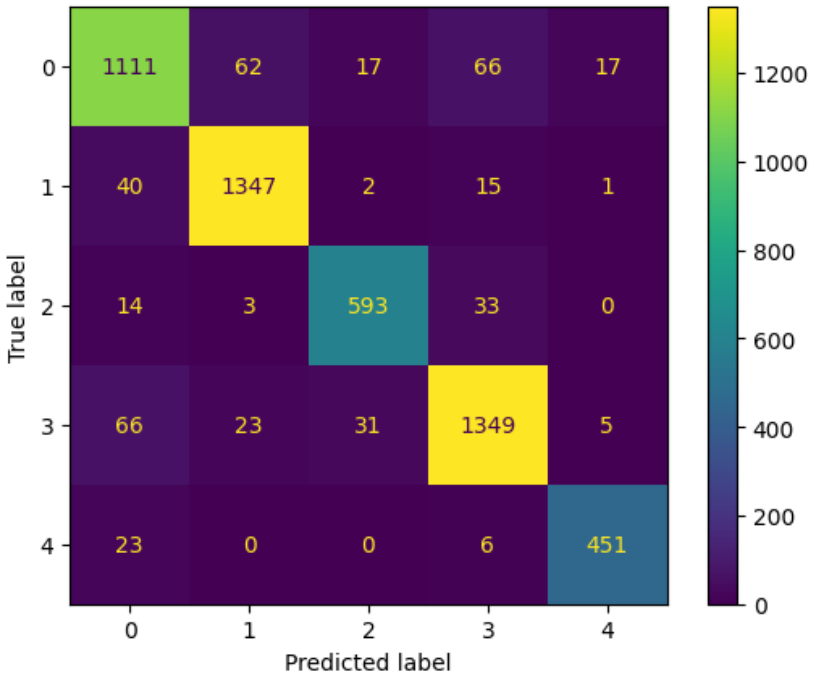

Figure 16: Voting Classifier confusion matrix.

7.5 Overview of classifiers

Classifiers Overview				
Classifier	Train loss	Train accuracy	Dev loss	Dev accuracy
NN	0.0946	0.9714	0.2205	0.9181
LGBM	0.0472	0.9990	0.2493	0.9093
XGB	0.0038	1.0000	0.2689	0.9103
Ensemble	0.0415	0.9988	0.2179	0.9196

8 Discussion

In this study, we employed various techniques and models to address the classification problem of research paper venue prediction. We successfully utilized the tsne algorithm to visualize the high-dimensional feature space of the dataset, allowing us to gain insights into the distribution and relationships of the research papers. Additionally, we applied both text and graph embeddings, including the Doc2Vec algorithm for text and the DeepWalk algorithm and GAE for graph, which effectively captured meaningful representations of the papers' abstracts and citation network structure, respectively. These embeddings proved valuable in improving the performance of our models.

Furthermore, we employed the SMOTE algorithm to mitigate the class imbalance present in the dataset, successfully generating synthetic samples to augment the minority classes. This technique helped to enhance the overall performance but did not mitigate the bias towards the majority classes, so we used the class weights directly to our classifiers in order to manipulate the imbalance.

We employed different classifiers, including a neural network, LGBM, XGBoost, and a voting classifier, to leverage the strengths of each model. The neural network demonstrated its capability to learn complex patterns and capture nonlinear relationships in the data. LGBM and XGBoost, known for their gradient boosting algorithms, exhibited strong performance in handling imbalanced datasets and handling multi-class classification problems. The voting classifier combined the predictions of the above models, providing a robust and balanced approach to classification.

As for the neural networks, we tried networks with greater complexity (more hidden layers) but the results were not the expected, since all of these networks were experiencing the problem of overfitting.

In terms of future work, incorporating more advanced neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could be explored to capture more complex patterns within the research paper data.

Experimenting with different ensemble methods or stacking techniques could potentially yield even better classification results. Moreover, exploring the use of pretrained language models, such as BERT or GPT, could capture more nuanced semantic information from the abstracts.