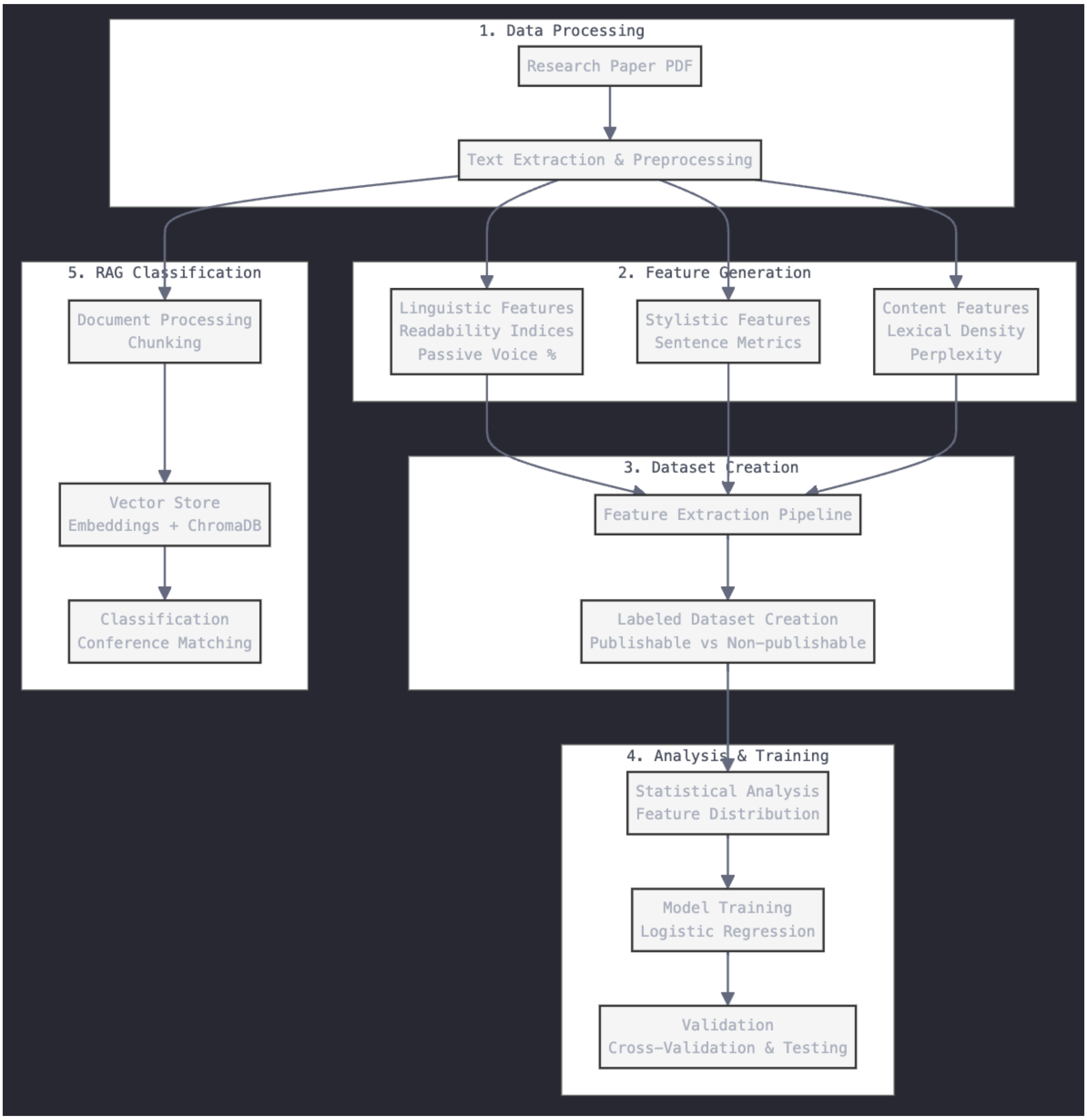
Research Paper Publishability Prediction System

Team: Binary Bats


1. Data Processing
Research Paper PDF
Text Extraction & Preprocessing
5. RAG Classification
Document Processing Chunking
Vector Store Embeddings + ChromaDB
Classification Conference Matching
2. Feature Generation
Linguistic Features Readability Indices Passive Voice %
Stylistic Features Sentence Metrics
Content Features Lexical Density Perplexity
3. Dataset Creation
Feature Extraction Pipeline
Labeled Dataset Creation Publishable vs Non-publishable
4. Analysis & Training
Statistical Analysis Feature Distribution
Model Training Logistic Regression
Validation Cross-Validation & Testing

1. Premise

The increasing volume of academic publications necessitates efficient mechanisms to determine the publishability of research papers. Leveraging Natural Language Processing (NLP) and machine learning techniques, this system aims to evaluate research papers based on specific linguistic, stylistic, and content-based features, and predict their publishability, along with classifying the papers to the most suited conference.

---

2. Feature Extraction

The feature extraction phase is critical for transforming unstructured research papers into quantifiable metrics. Key metrics derived from the papers include:

2.1 Linguistic Features

1. **Flesch-Kincaid Grade Level:** Measures readability based on sentence length and syllable count.
2. **Gunning Fog Index:** Evaluates text complexity by factoring in sentence length and percentage of complex words.
3. **Coleman-Liau Index:** Focuses on readability using word and sentence lengths.
4. **Passive Voice Percentage:** Indicates the proportion of sentences written in passive voice.

2.2 Stylistic Features

1. **Average Sentence Length:** Represents the average number of words per sentence.
2. **Sentence Length Variation:** Captures the standard deviation of sentence lengths.

2.3 Content-Based Features

1. **Lexical Density:** Measures the proportion of content words to total words.
2. **Perplexity:** A statistical measure of text predictability.

2.4 Text Extraction

Research papers, stored as PDFs, were processed using the `PyPDF2` library to extract textual content. The extracted text underwent preprocessing, including tokenization, removal of stop words, and lemmatization, ensuring consistency across all papers.

---

3. Dataset Preparation

The system utilized a dataset comprising publishable and non-publishable research papers. The following steps were performed:

3.1 Dataset Organization

1. **Publishable Papers:** Papers sourced from conferences such as CVPR, EMNLP, KDD, NeurIPS, and TMLR.
2. **Non-Publishable Papers:** Papers deemed non-publishable based on specific criteria.

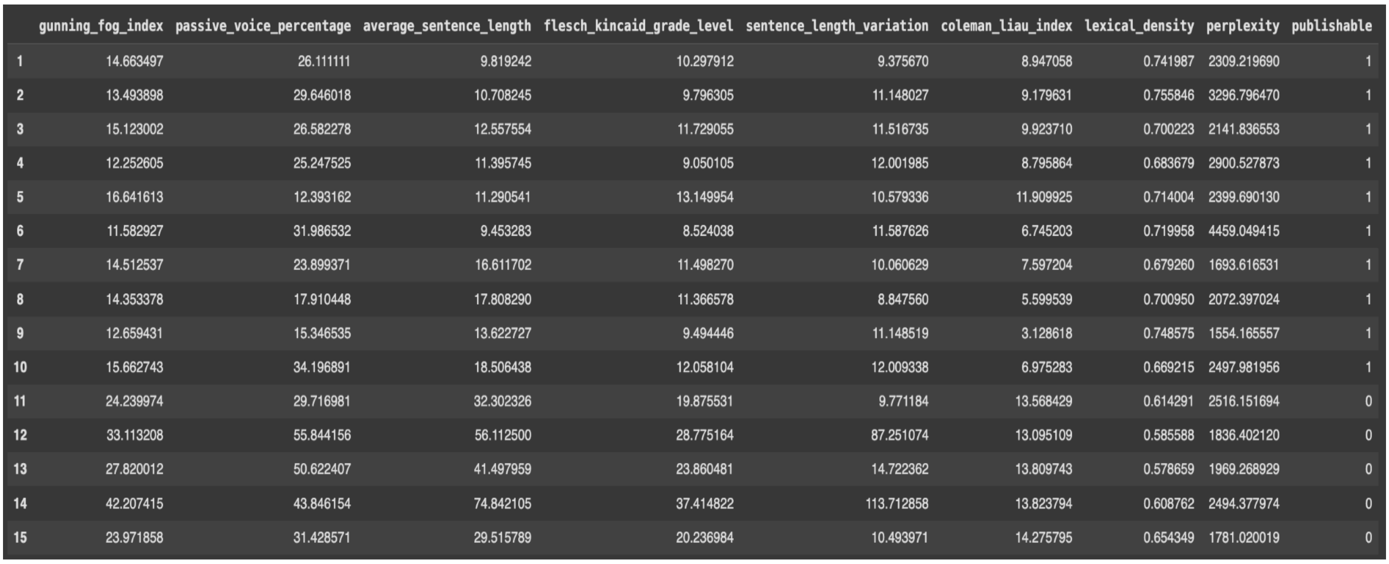
3.2 Feature Extraction Pipeline

A feature extraction function was implemented to compute the metrics for each paper. The metrics, along with the paper's name and category, were stored in a CSV file for further analysis.

3.3 Cleaning and Labeling

The dataset was refined by:

- Removing irrelevant columns.
- Adding a binary label (`publishable`): 1 for publishable papers and 0 for non-publishable ones.



	gunning_fog_index	passive_voice_percentage	average_sentence_length	flesch_kincaid_grade_level	sentence_length_variation	coleman_liau_index	lexical_density	perplexity	publishable
1	14.663497	26.111111	9.819242	10.297912	9.375670	8.947058	0.741987	2309.219690	1
2	13.493898	29.646018	10.708245	9.796305	11.148027	9.179631	0.755846	3296.796470	1
3	15.123002	26.582278	12.557554	11.729055	11.516735	9.923710	0.700223	2141.836553	1
4	12.252605	25.247525	11.395745	9.050105	12.001985	8.795864	0.683679	2900.527873	1
5	16.641613	12.393162	11.290541	13.149954	10.579336	11.909925	0.714004	2399.690130	1
6	11.582927	31.986532	9.453283	8.524038	11.587626	6.745203	0.719958	4459.049415	1
7	14.512537	23.899371	16.611702	11.498270	10.060629	7.597204	0.679260	1693.616531	1
8	14.353378	17.910448	17.808290	11.366578	8.847560	5.599539	0.700950	2072.397024	1
9	12.659431	15.346535	13.622727	9.494446	11.148519	3.128618	0.748575	1554.165557	1
10	15.662743	34.196891	18.506438	12.058104	12.009338	6.975283	0.669215	2497.981956	1
11	24.239974	29.716981	32.302326	19.875531	9.771184	13.568429	0.614291	2516.151694	0
12	33.113208	55.844156	56.112500	28.775164	87.251074	13.095109	0.585588	1836.402120	0
13	27.820012	50.622407	41.497959	23.860481	14.722362	13.809743	0.578659	1969.268929	0
14	42.207415	43.846154	74.842105	37.414822	113.712858	13.823794	0.608762	2494.377974	0
15	23.971858	31.428571	29.515789	20.236984	10.493971	14.275795	0.654349	1781.020019	0

4. Exploratory Data Analysis

Exploratory Data Analysis (EDA) was performed to uncover patterns and relationships among features:

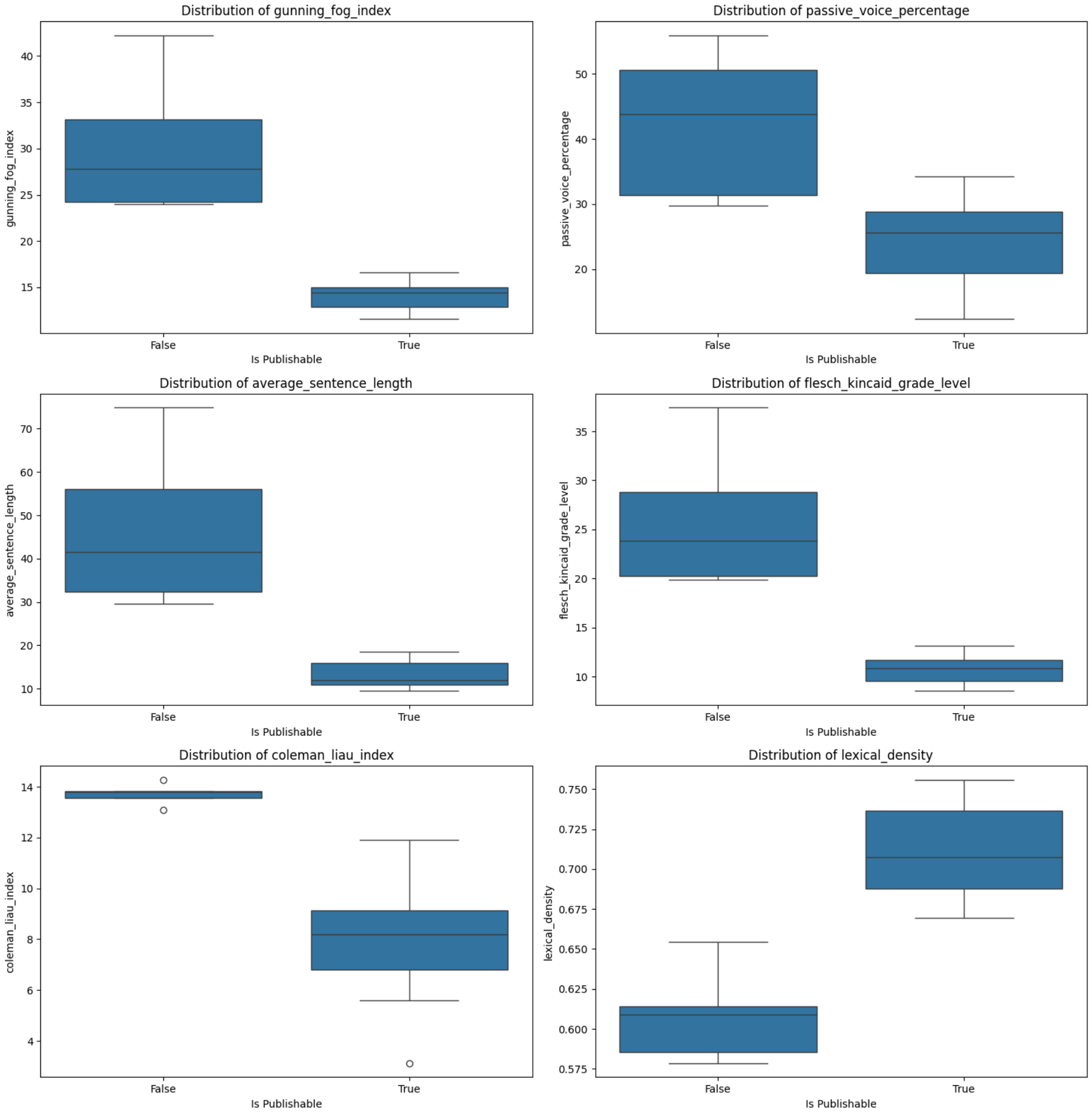
4.1 Statistical Analysis

For each feature, statistical tests were conducted:

1. **T-Test:** Compared the means of publishable and non-publishable papers.
2. **Effect Size (Cohen's d):** Measured the magnitude of differences.

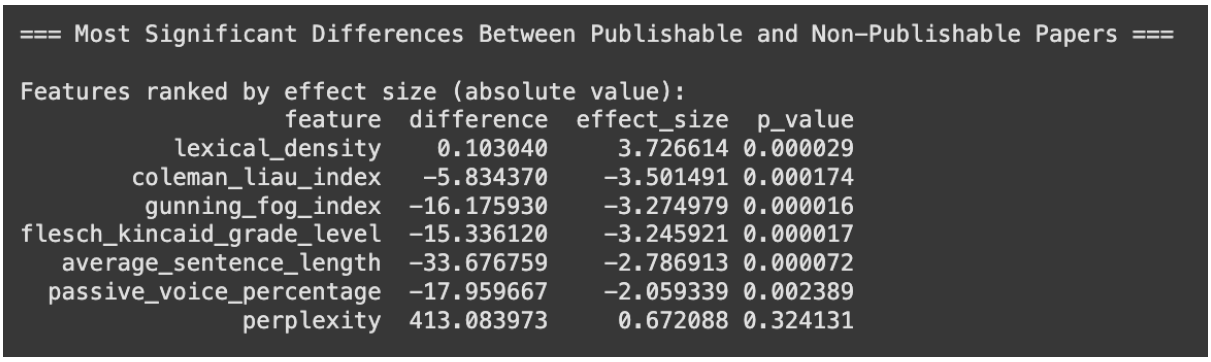
4.2 Feature Distribution

Boxplots were generated to visualize the distribution of each feature across publishable and non-publishable categories.



4.3 Correlation Analysis

A correlation matrix was computed to identify relationships among features and the target variable.



```
=== Most Significant Differences Between Publishable and Non-Publishable Papers ===

Features ranked by effect size (absolute value):
                   feature  difference  effect_size  p_value
           lexical_density    0.103040     3.726614 0.000029
        coleman_liau_index   -5.834370    -3.501491 0.000174
         gunning_fog_index  -16.175930    -3.274979 0.000016
flesch_kincaid_grade_level  -15.336120    -3.245921 0.000017
   average_sentence_length  -33.676759    -2.786913 0.000072
  passive_voice_percentage  -17.959667    -2.059339 0.002389
                perplexity  413.083973     0.672088 0.324131
```

6. Model Training

To ensure uniformity across features, Min-Max Scaling was applied. The scaled features were stored in a separate DataFrame, maintaining column names and indexing for interpretability. The initial model selected for prediction was **Logistic Regression**, owing to the size of the daraset, Linearity and interpretability. Key steps include:

6.1 Model Definition

- **Algorithm:** Logistic Regression.
- **Regularization Parameter (C):** Tuned to control overfitting.
- **Solver:** ‘liblinear’ for efficiency with small datasets.

7. Hyperparameter Tuning

To optimize the Logistic Regression model, hyperparameter tuning was performed using a grid search over the following parameter space:

1. **Penalty:** ‘l1’ and ‘l2’ regularization techniques.
2. **Regularization Parameter (C):** Logarithmic range from to .
3. **Solver:** ‘liblinear’ and ‘saga’.
4. **Max Iterations:** 100, 1000, 2500, and 5000.

8. Model Validation

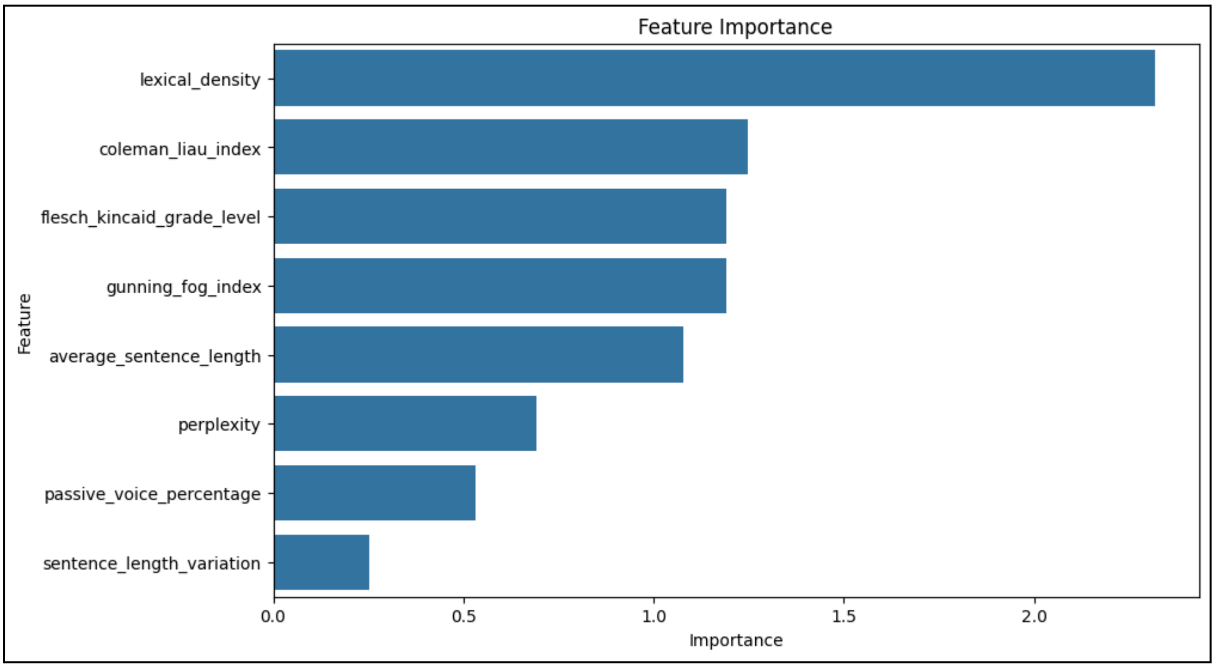
To ensure robust evaluation, two validation strategies were employed:

1. **K-Fold Cross-Validation**
2. **Test-Train Split**

8.1 Key Findings:

- Both validation strategies yielded an **F1 Score** and **Accuracy** of **1.00**.
- The perfect scores are attributed to the small dataset size and the presence of distinct linear separability within the features, which significantly simplified classification tasks.

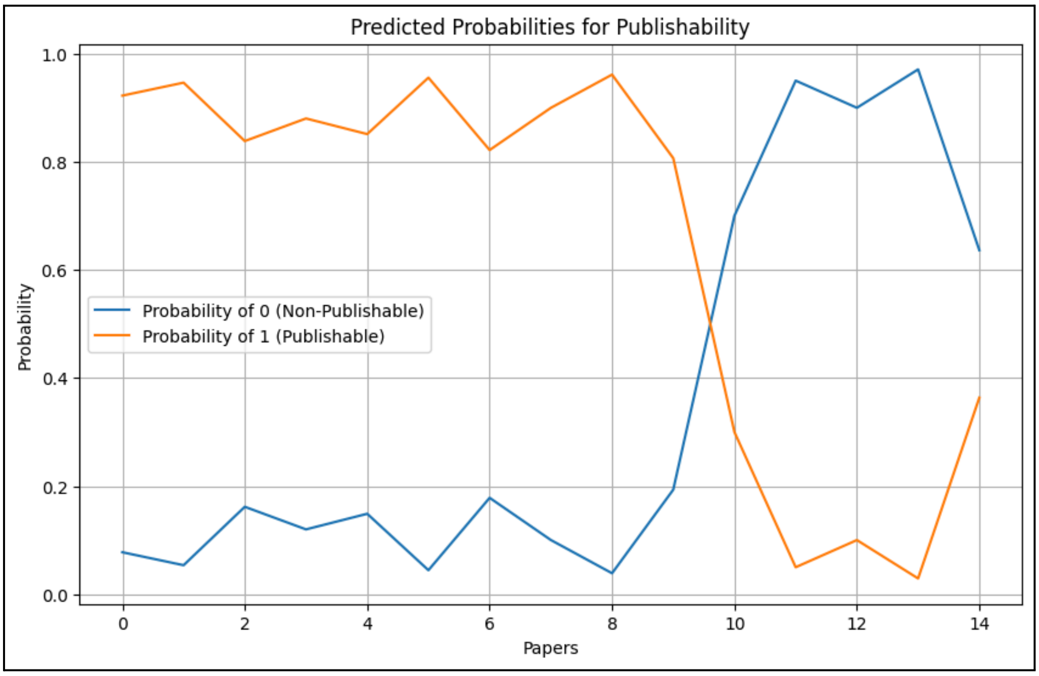
8.2 Feature Importance Analysis:



- The insights from feature importance help understand which linguistic and structural aspects of a paper are most critical for classification.
- A bar plot was generated to visually rank the features, highlighting their respective contributions to the model's predictions.

8.3 Predicted Probabilities for Publishability

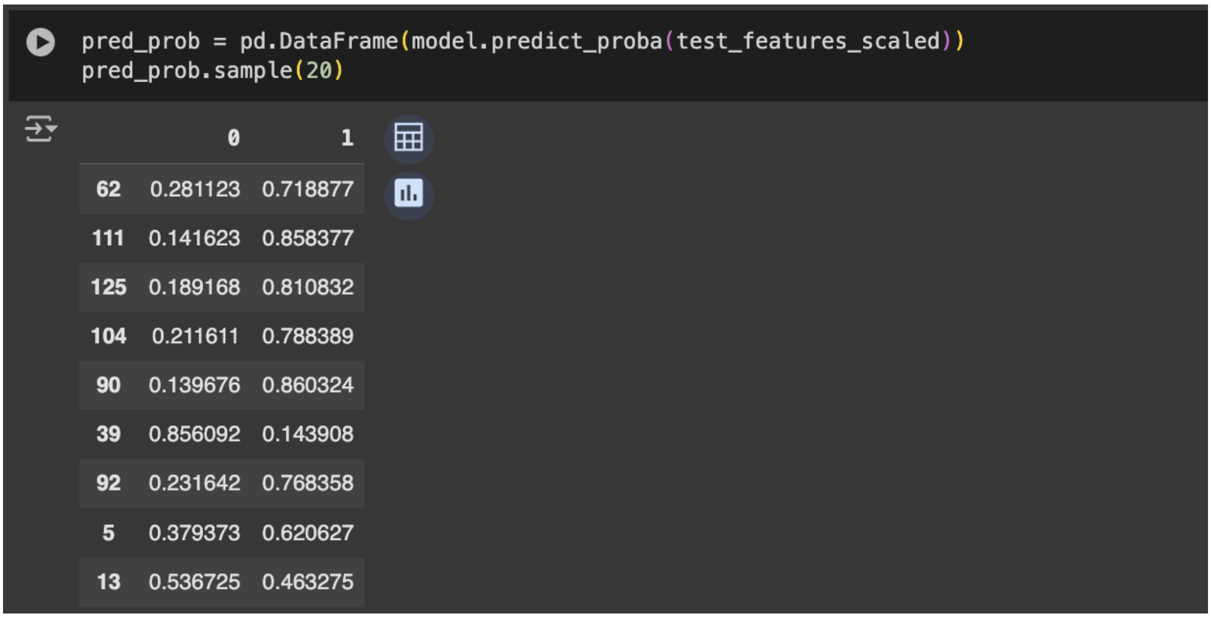
- The model provided class probabilities for each paper, demonstrating its confidence in classifying papers as publishable or non-publishable.
- A line plot visualized the probability distributions for both classes, offering insights into the separation achieved by the model.



---

9. Predictions and Results

- Predictions were made on a separate testing dataset to determine the publishability of research papers.
- Results showed a balance between publishable and non-publishable papers, with **29 non-publishable papers** identified in the dataset.



```
pred_prob = pd.DataFrame(model.predict_proba(test_features_scaled))
pred_prob.sample(20)
```

```
            0         1
 62  0.281123  0.718877
111  0.141623  0.858377
125  0.189168  0.810832
104  0.211611  0.788389
 90  0.139676  0.860324
 39  0.856092  0.143908
 92  0.231642  0.768358
  5  0.379373  0.620627
 13  0.536725  0.463275
```

10. Research Paper Classification using RAG

The system leverages a hybrid approach by combining **SentenceTransformer embeddings**, **BGE (Bidirectional Generative Embeddings) model**, and a **semantic search** mechanism to classify papers into conferences. The system integrates the following components to process, embed, and analyze research papers to predict the most suitable conference for each paper.

1. **Document Processing:** The system uses the `DocumentProcessor` class to handle PDF papers. The primary tasks performed here are extracting the textual content from the PDF and chunking it into smaller overlapping sections. This chunking process ensures that the context of each paper is preserved while making the analysis more manageable.

2. **Vector Store Setup:** It is responsible for embedding the paper text and storing these embeddings in a searchable format. It utilizes the `SentenceTransformer` to create embeddings for each chunk of text from research papers and then stores them in a ChromaDB collection. This collection is critical for performing fast semantic searches to retrieve similar research papers.

3. **Ensemble Retrieval:** It employs two distinct models for embedding text: **SentenceTransformer and BGE**. The retrieval process involves searching for semantically similar papers using both models and then combining the results for better accuracy. The hybrid retrieval system ensures that papers from conferences with a similar research domain are identified more effectively, improving the quality of predictions.

4. **Conference Classification:** Once a paper is processed and embedded, the classification is performed by the `ConferenceClassifier`. This class uses the embeddings from the retriever to find the top similar papers, then **ranks the conferences based on the frequency and relevance of matches**. The conference with the highest score is predicted as the most likely venue for the paper. Additionally, a generative AI model from Google (Gemini) is employed to generate a rationale for the classification, providing an explanation for why a particular conference is chosen based on the research content. This rationale is designed to enhance the transparency and interpretability of the model’s decisions.

10.1 Training Phase:

- The system is trained using a set of labeled research papers from various conferences such as CVPR, EMNLP, KDD, NeurIPS, and TMLR. During training, the system processes each paper, extracts text, and divides the content into chunks. These chunks are then embedded and added to the vector store, enabling efficient similarity searches for future classification tasks.
- After training, the model is saved along with the scaler for future use, which allows for the persistence of the trained model and vector store.

10.2 Classification Phase:

- The classification phase takes a new set of research papers and classifies them into conferences. Each paper is processed similarly to the training data: text is extracted, chunked, and searched against the vector store for similar papers. The results are weighted by conference occurrence, and the conference with the highest score is predicted.
- The rationale for the classification is generated by the Gemini model, which provides a concise explanation of why the paper was categorized into a particular conference based on its content. Non-publishable papers are skipped, and their classification results are stored as 'NA.'

10.3 Key Features:

1. **Hybrid Search:** The system uses both SentenceTransformer and BGE for semantic search, combining the strengths of both models to provide more accurate results.
2. **Explainability:** The use of the Gemini model for generating rationales ensures that the classification process is transparent and interpretable. This is crucial for users to trust and understand the system's predictions.
3. **Efficient Classification:** By chunking the papers and focusing on the most relevant sections, the system maximizes efficiency without compromising on accuracy.
4. **Publishability Filtering:** The system selectively processes only publishable papers, making the classification process more targeted and efficient.

10.4 Challenges and Future Improvements:

1. **Scalability:** The system may face performance issues with a large number of papers or conferences. Scaling the vector store and optimizing the retrieval process could address this challenge.
2. **Improving Rationale Generation:** The quality of the rationale depends on the quality of the input context provided to the generative model. Enhancing the context extraction and ensuring that relevant sections of the paper are included can lead to better explanations.

3. **Generalization to Other Domains:** The model is currently specialized for research conference classification. Future work could focus on extending the system to handle papers from multiple domains (e.g., journals, grants) by training on diverse datasets.

---

10. Conclusion:

This project showcases a comprehensive, multi-step pipeline for determining publishability and classifying research papers into conferences. By leveraging machine learning, advanced embeddings, hybrid retrieval strategies, and generative AI, the system provides a robust solution for research paper publishability determination and classification. The model’s ability to explain its predictions further enhances its value, making it a practical tool for researchers and academics in the publication process. The system holds significant potential for automation in academic paper classification, enabling faster and more accurate decision-making in the research community.
