**PROBLEM STATEMENT**

**The manual evaluation of research papers for conference submission is labor intensive, time-consuming, and demands significant expertise.** **Participants will have access to a dataset of high-quality, benchmark research papers from conferences or conferences. The objective is to evaluate new submissions, compare them with these benchmark papers, and recommend the most suitable conferences or conference with formal justification.**

**DATASET LINK:** [**KDSH\_2025\_Dataset - Google Drive**](https://drive.google.com/drive/folders/1Z8z4craj36ighb8hzUzeM76OOgpUdsKr)

**Explanantion of both the task:**

**Task 1: Publishability Classification**

**Objective**

To train a model that classifies research papers as **Publishable (1)** or **Non-Publishable (0)** based on text data. The trained model is used later in Task 2 to filter papers for conference recommendations.

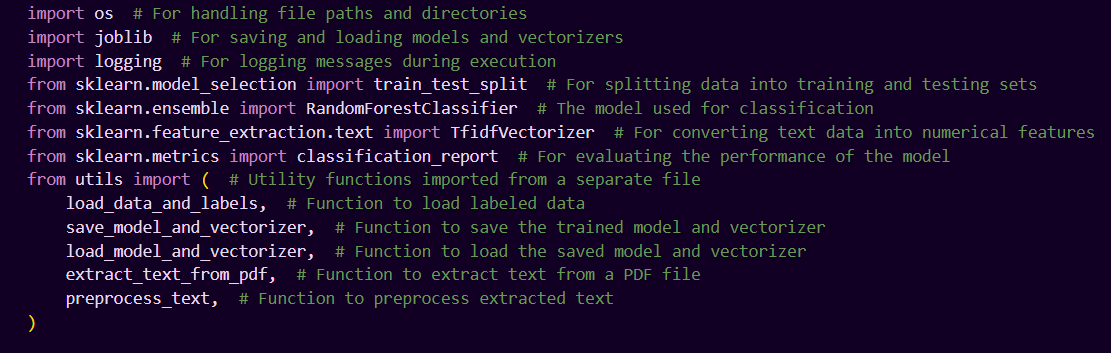
**Implementation**

**Steps**

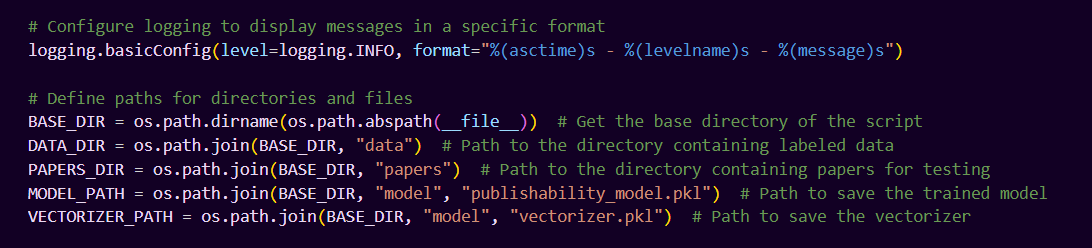
1. **Prepare Data**:
   * Two categories of labeled data are stored in the data/ directory:
     + Publishable/: Contains text files of papers labeled as publishable.
     + Non-Publishable/: Contains text files of papers labeled as non-publishable.
2. **Preprocess Data**:
   * Text data is vectorized into numerical features using TfidfVectorizer.
3. **Train-Test Split**:
   * The dataset is split into training and testing sets (80% training, 20% testing).
4. **Model Training**:
   * A RandomForestClassifier is trained using the TF-IDF features.
5. **Evaluation**:
   * The model is evaluated using the test set.
   * Classification metrics such as precision, recall, and F1-score are reported.
6. **Save Model**:
   * The trained model and vectorizer are saved as publishability\_model.pkl and vectorizer.pkl for use in Task 2.

**main.py Code Explanation**:

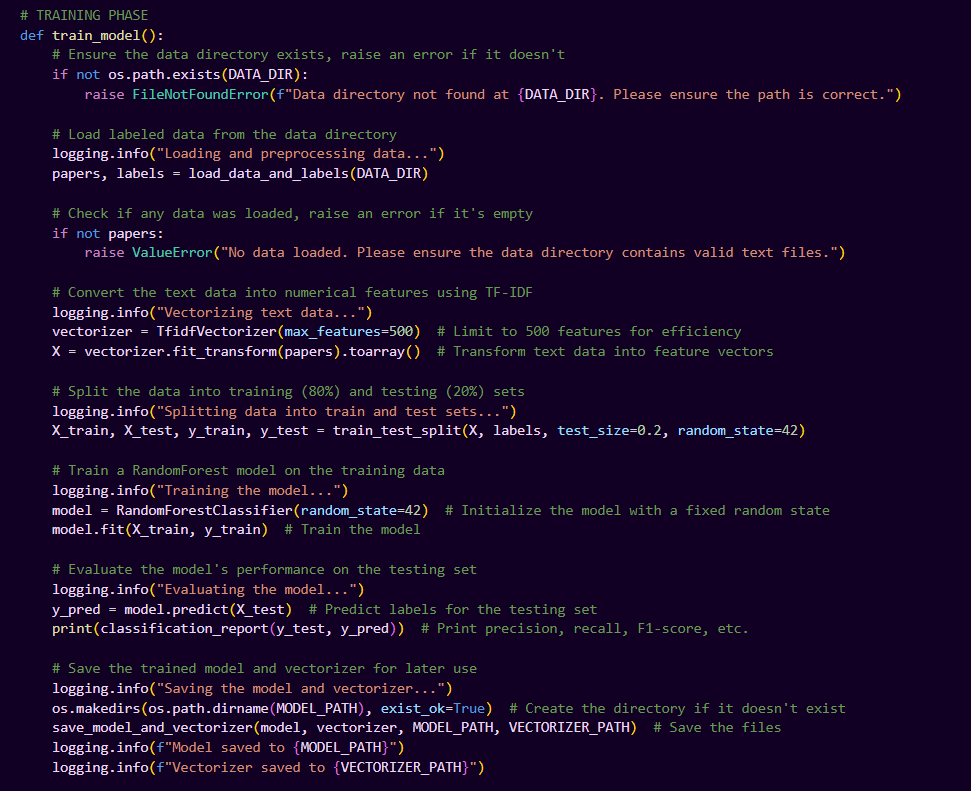
* Import necessary Libraries



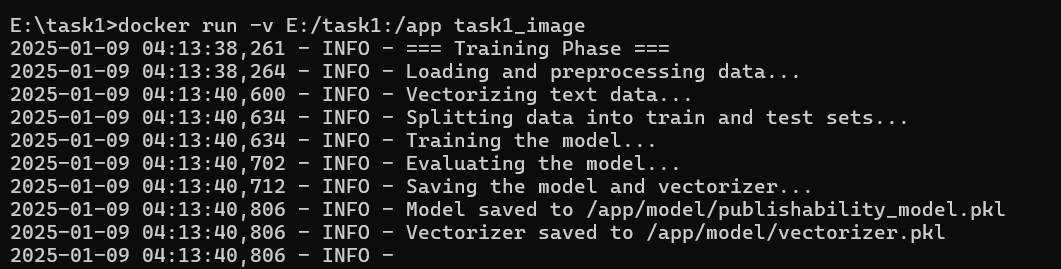
* Adding directories path to the code



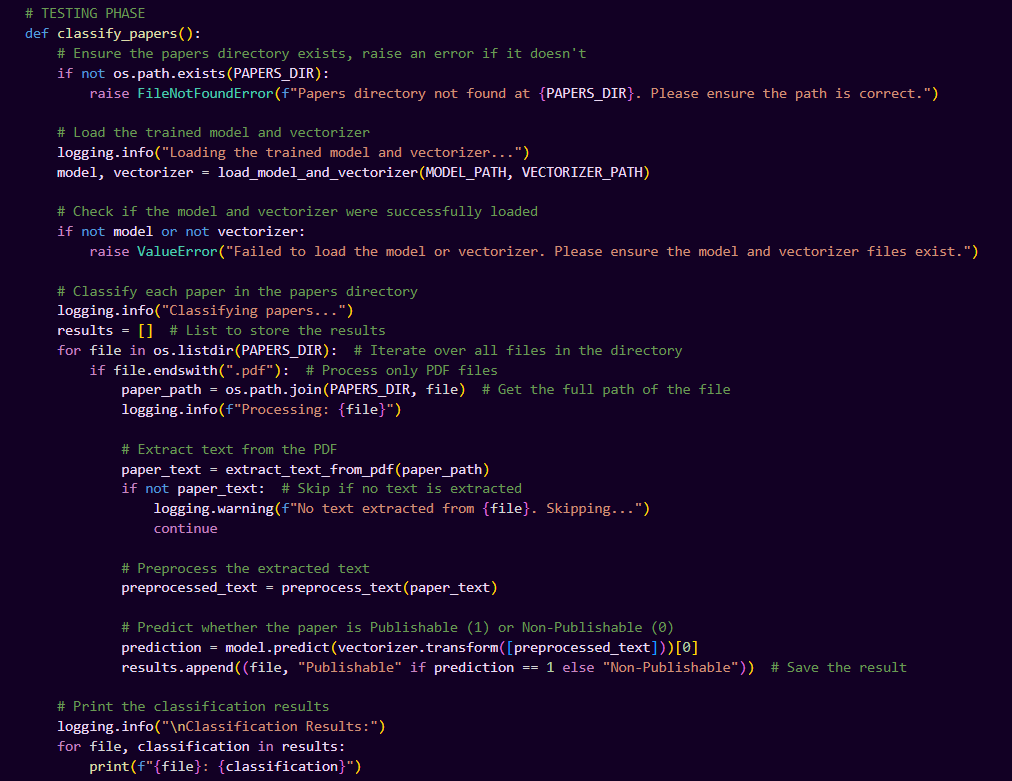
* Training phase for Task1

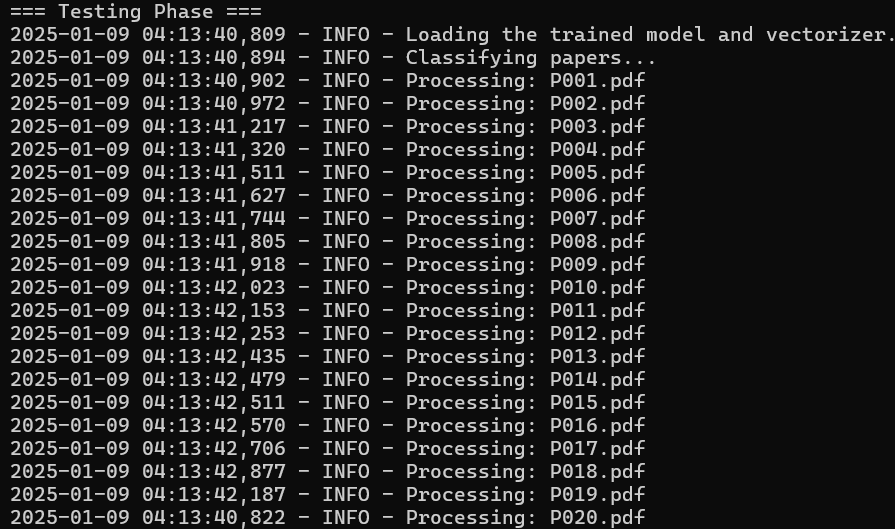


Actual Output of Training Phase:

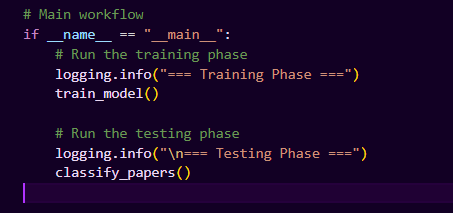


* Testing Phase of Task1

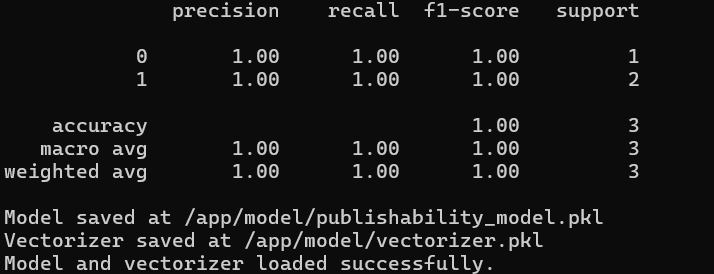


Actual Output of Testing Phase

* Main Function



Accuracy and F1 Score of determining Publishability & Classification Results:





**Task 2: Conference Recommendation**

**Objective**

To recommend the most suitable conference for **Publishable** papers identified by Task 1 and generate a rationale for the recommendation.

**Implementation**

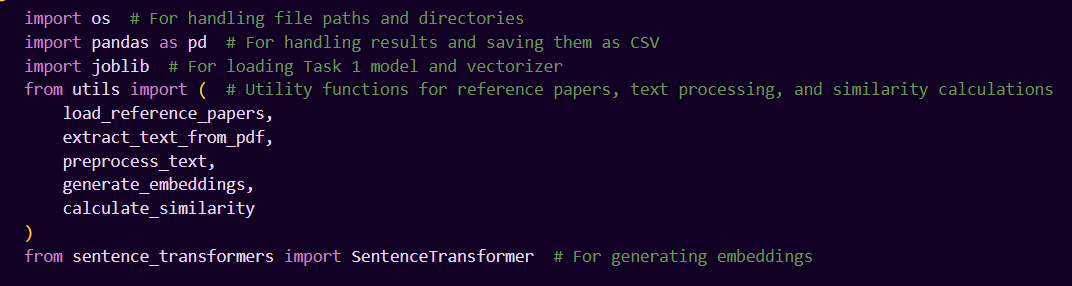
**Steps**

1. **Load Task 1 Model**:
   * Use publishability\_model.pkl and vectorizer.pkl to filter **Publishable** papers.
2. **Conference Embeddings**:
   * Generate embeddings for reference papers for each conference using SentenceTransformer.
3. **Similarity Calculation**:
   * For each paper, calculate cosine similarity with conference embeddings.
   * Recommend the conference with the highest similarity.
4. **Save Results**:
   * Save results in a results.csv file with columns:
     + Paper ID
     + Publishable
     + Conference
     + Rationale

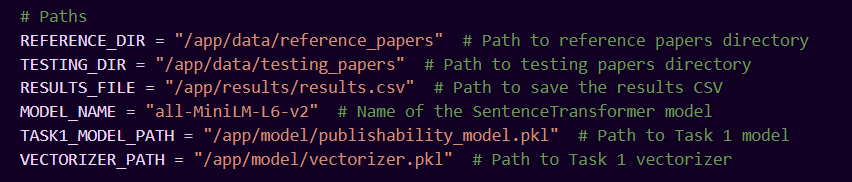
**Complete Function Overview utils of task 1**

1. **extract\_text\_from\_pdf(pdf\_path)**:
   * Extracts text from a PDF file and handles errors gracefully.
   * Uses PyPDF2 to read the file and extract text from its pages.
2. **preprocess\_text(text)**:
   * Preprocesses raw text by:
     + Converting it to lowercase.
     + Removing special characters and keeping only alphanumeric characters and spaces.
   * Returns cleaned text or an empty string if an error occurs.
3. **load\_data\_and\_labels(data\_dir)**:
   * Loads and processes labeled data from directories:
     + Non-Publishable/ is labeled as 0.
     + Publishable/ is labeled as 1.
   * Handles missing directories or empty data with warnings.
4. **save\_model\_and\_vectorizer(model, vectorizer, model\_path, vectorizer\_path)**:
   * Saves the trained model and vectorizer to specified file paths using joblib.
5. **load\_model\_and\_vectorizer(model\_path, vectorizer\_path)**:
   * Loads a saved model and vectorizer from specified file paths using joblib.
   * Handles errors and returns (None, None) if loading fails.

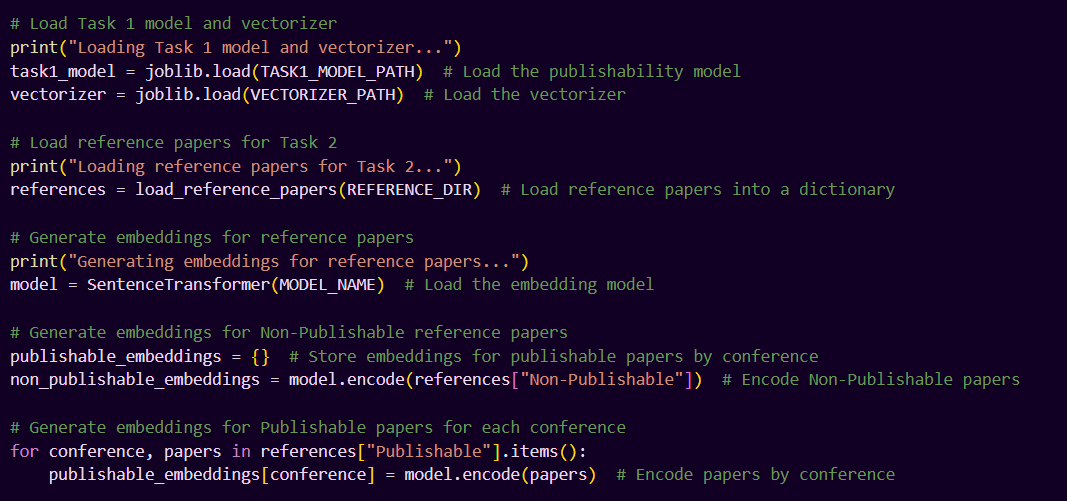
* Import Necessary Libraries

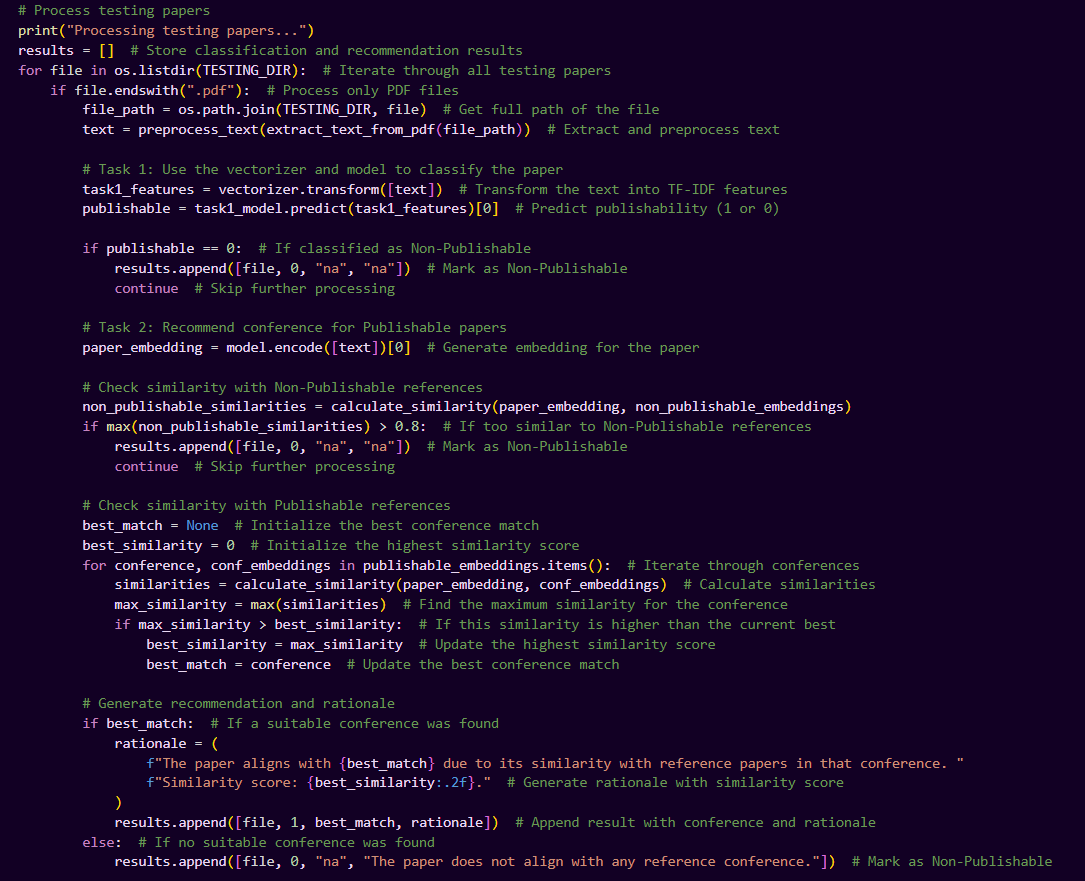


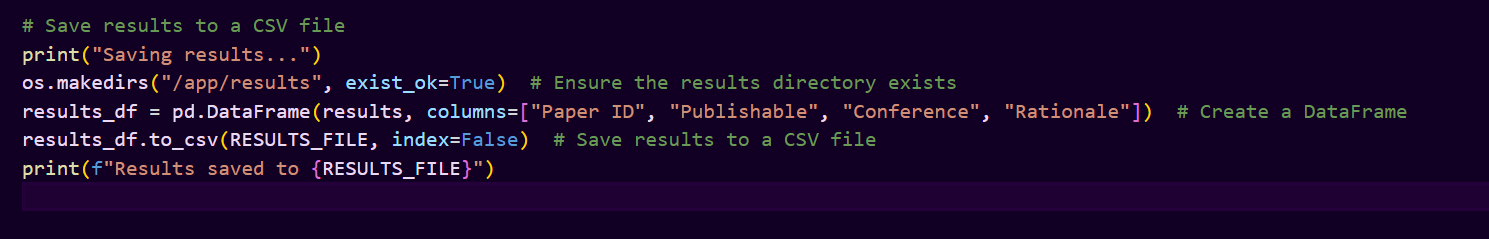
* Adding directories path to the code



* Loading Necessary Data and Embeddings for Task 2



* Testing Phase of Task2
* Saving results in CSV format



* Actual CSV output of the above code

