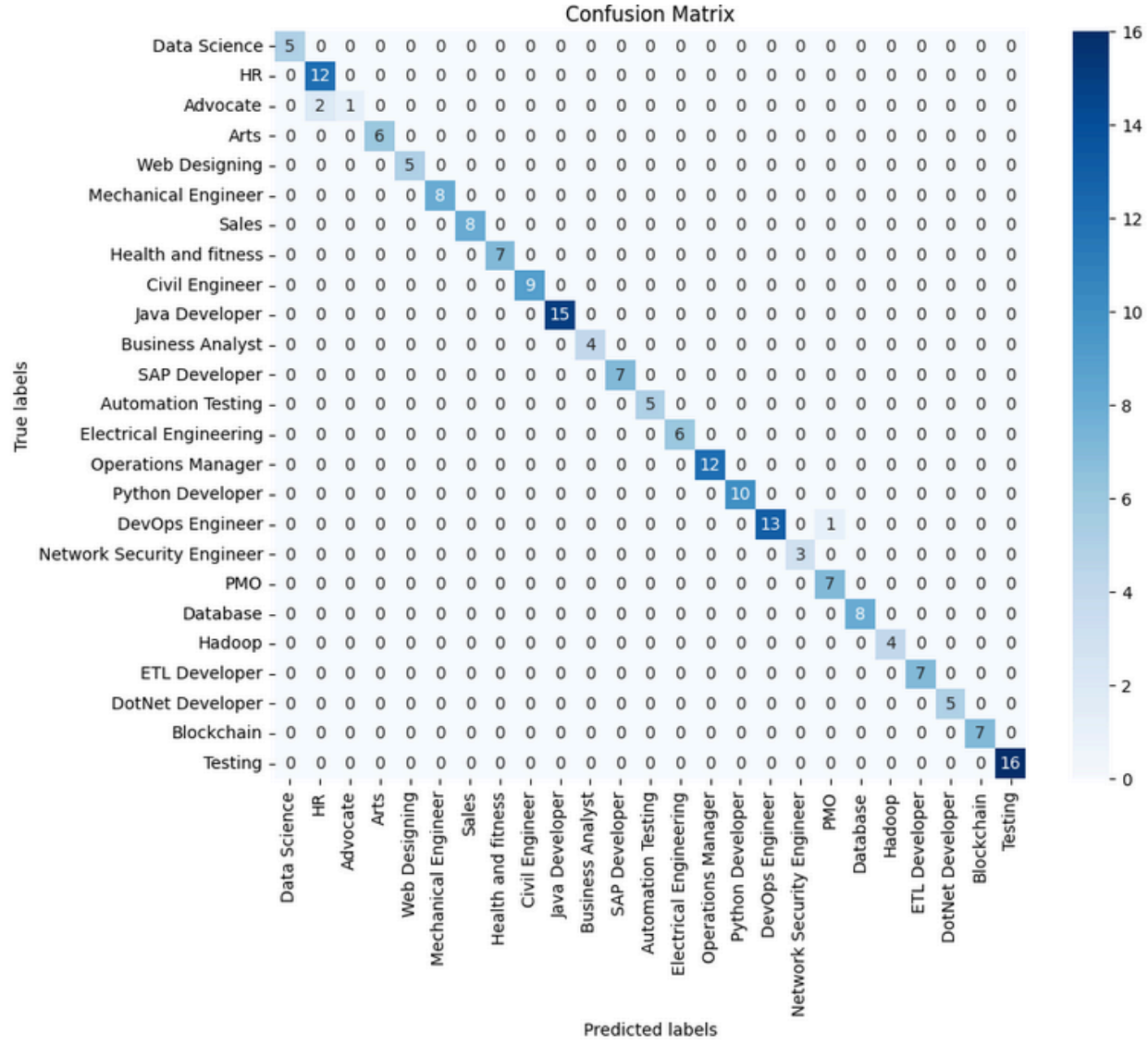
Updated Baseline Results

Dataset taken from: https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset

-> We used TF-IDF vectors to encode textual values to numbers.
-> Label encoded all the categories from 1 to 25.
-> Divided train and test in ratio of 80:20
-> We used Random Forest to classify the textual values from the resume and get probabilistic matching. Used this probability to rank appropriate jobs.
-> Got an accuracy of 98% on the test set.
-> Precision : 0.99
-> Recall : 0.98
-> F1 Score: 0.98



- Website Development -
    - We created a web application that is built using Flask, a Python web framework, which provides routing and rendering capabilities.
    - HTML templates are utilized for rendering the user interface. The `index.html` template is rendered for the home page, and the `jobs.html` template is rendered to display job openings.
    - A form is provided on the home page for users to upload their resumes. The form submits a POST request to the `/predict` route.
    - The `/predict` route processes the uploaded resume, applies a pre-trained machine learning model to predict suitable job fields, and then fetches job openings for each predicted field using the Reed API.
    - The fetched job openings are displayed on the `jobs.html` template. Job details such as title, ID, and URL are presented in an ordered list for each predicted job field.
    - As of now, the website is quite basic. However, we plan to add additional features in order to make the website more useful as well as enhance the appearance.

Proposed Method: Resume-Job Matching and Optimization

1. **Resume and Job Posting Matching**
    a. **TF-IDF with Random Forest:**
        i. In this approach, we utilize TF-IDF (Term Frequency-Inverse Document Frequency) to vectorize the textual content of both resumes and job postings.
        ii. The vectorized representations of resumes and job postings are then fed into a Random Forest classifier.
        iii. Random Forest, a versatile ensemble learning technique, is employed to predict the suitability of each resume for various job postings based on the features extracted by TF-IDF.
    b. **BERT for Semantic Understanding:**
        i. Recognizing the limitations of TF-IDF in capturing nuanced semantic relationships, we employ BERT (Bidirectional Encoder Representations from Transformers) for semantic understanding.
        ii. We fine-tune a pre-trained BERT model on a corpus of resumes to create embeddings that represent the semantic content of each resume.
        iii. Similarly, embeddings are generated for job postings, allowing us to capture the semantic similarities between resumes and job descriptions.
        iv. By utilizing BERT embeddings, we aim to achieve more accurate and contextually relevant matches between resumes and job postings, considering not only keyword matches but also the broader semantic context.
        v. This approach offers superior performance in scenarios where understanding the nuanced meaning and context of text data is crucial, such as matching resumes to job postings based on the underlying skills and experiences required.

2. **Resume Optimization using BERT:**
    a. Leverage the semantic understanding provided by BERT(specifically the skill embeddings) to optimize resumes for specific job postings.
    b. Analyze the BERT embeddings of both the resume and the job posting to identify areas where the resume can be tailored or optimized to better match the requirements of the job.
    c. Develop algorithms to suggest modifications to the resume based on the semantic analysis, such as recommending additional skills or experiences to highlight, or restructuring the content for better relevance.
3. **Website:**
    a. User can upload resumes via a form on the home page.
    b. Uploaded resumes are processed via a pre-trained ML model to predict suitable job fields.
    c. Fetched job openings will be dynamically displayed.
    d. Future updates will focus on improving user experience and visual appeal. Emphasis on enhancing appearance through improved styling, layout design, and interactive elements.
    e. Ensure scalability and maintainability as additional features are added.
4. **Evaluation and Iteration:**
    a. Conduct thorough evaluation and testing, comparing performance of the BERT model with that of the version that used TF-IDF and Random Forest.
    b. Use metrics such as accuracy, precision, recall, and F1 score to assess the effectiveness.
    c. Gather user feedback to identify areas for improvement and iterate on the system accordingly.