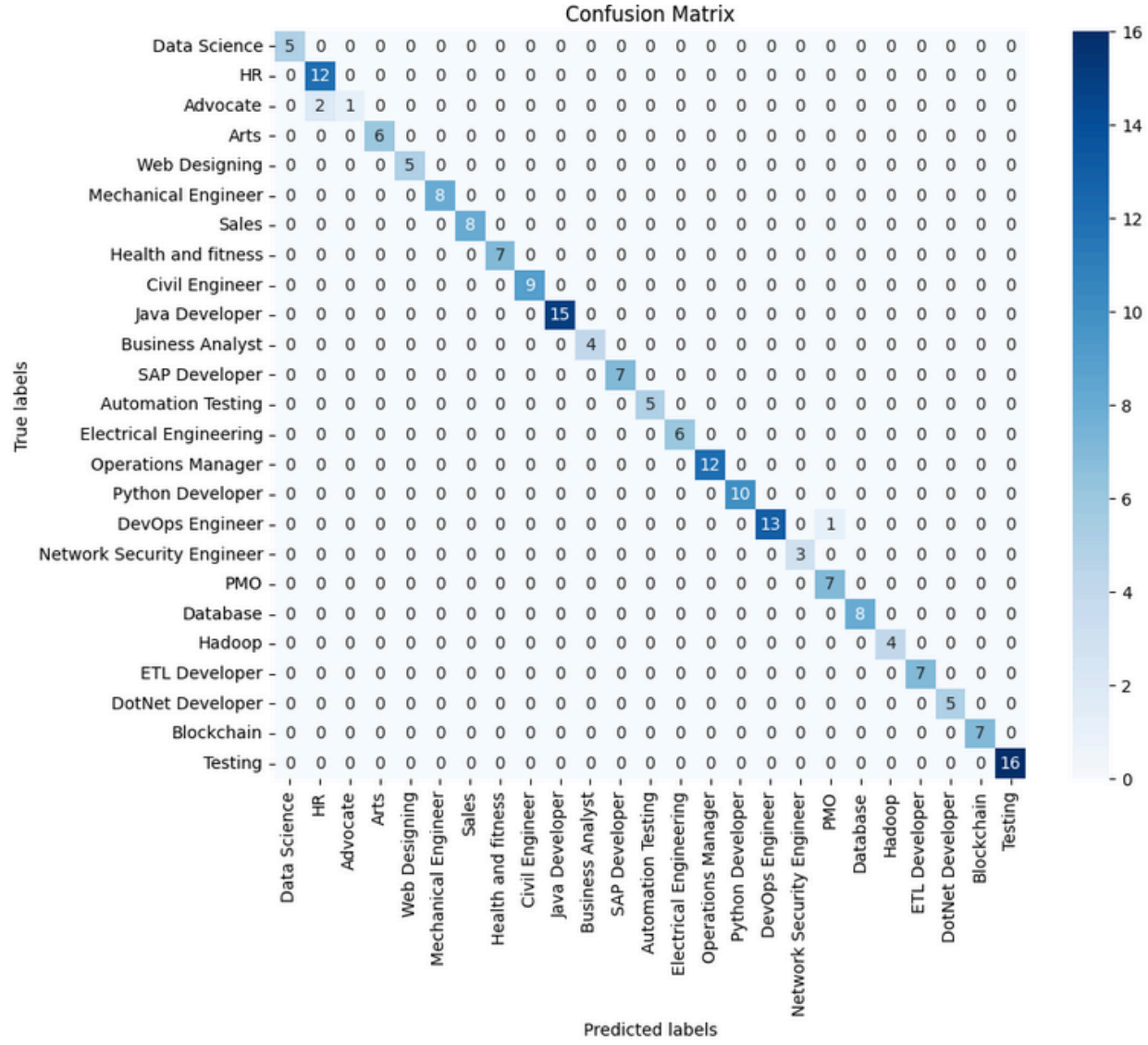
Updated Baseline Results

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: Enhancing Resume-Job Matching and Optimization using BERT

1. Semantics Capture using BERT:

- Fine-tune a pre-trained BERT model on a corpus of resumes to create embeddings that represent the semantic content of each resume. This involves converting the textual data into BERT embeddings for the entire resume.
- Additionally, extract individual skills mentioned in the resumes and create embeddings specifically for these skills using BERT. This will provide a finer-grained representation of the candidate's qualifications and experiences.

2. Improved Matching Algorithm:

- Use the BERT embeddings of the resumes to match them with relevant job postings. This can be done by calculating similarity scores between the BERT embeddings of resumes and those of job descriptions.
- Experiment with various similarity metrics such as cosine similarity or Euclidean distance to determine the best approach for matching resumes to job postings effectively.
- Incorporate the probabilistic matching approach used previously, but now based on the enhanced semantic understanding provided by BERT embeddings.

3. Resume Optimization using BERT:

- Leverage the semantic understanding provided by BERT(specifically the skill embeddings) to optimize resumes for specific job postings.
- 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.

- 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.

4. Website:

- User can upload resumes via a form on the home page.
- Uploaded resumes are processed via a pre-trained ML model to predict suitable job fields.
- Fetched job openings will be dynamically displayed.
- Future updates will focus on improving user experience and visual appeal. Emphasis on enhancing appearance through improved styling, layout design, and interactive elements.
- Ensure scalability and maintainability as additional features are added.

5. Evaluation and Iteration:

- Conduct thorough evaluation and testing of the updated system, comparing its performance against the previous version that used TF-IDF and Random Forest.
- Use metrics such as accuracy, precision, recall, and F1 score to assess the effectiveness of the BERT-based approach in matching resumes to job postings and optimizing resumes.
- Gather user feedback to identify areas for improvement and iterate on the system accordingly.