MLE - 1 Parspec Assignment Report

1. Introduction

1.1 Project Overview

This project aims to classify documents based on their content. The data consists of two columns: datasheet_link and target_col. The datasheet_link column contains URLs to PDF documents, and the target_col contains the target class labels.

1.2 Objectives

- Extract text data from PDF documents.
- Clean and preprocess the extracted text.
- Vectorize the text using different methods.
- Train classification models.
- Evaluate the models using precision, recall, and F1 score.
- Develop an inference pipeline to predict the class of a new document based on its URL.

2. Data Extraction and Preprocessing

2.1 Data Extraction

- **Library Used**: PyMuPDF
- **Process**: Extracted the text from the PDFs linked in the datasheet_link column.
    - Initially the training dataset is of size 1895 rows, after dropping the duplicate rows it becomes - 1199 rows. Then after processing the pdf url, I only left with 487 rows including :
        - lighting 299
        - fuses 75
        - cable 68
        - others 45
- **Time:** This step took 2-3 hours as there were a lot of exceptions while dealing with the links provided.

2.2 Data Cleaning

- **Techniques Used**: Removed tabs, newlines, and other irrelevant characters. Then remove stopwords, apply stemming to the tokens and at last merge them in string form.
- **Tools**: Custom preprocessing function.

- **Time:** Initially, I analyzed the data then applied the technique mentioned above. It hardly took 30 minutes to clean both the train and test files.

3. Text Vectorization

3.1 CountVectorizer

- **Description**: Converts a collection of text documents to a matrix of token counts.

3.2 TfidfVectorizer

- **Description**: Converts a collection of text documents to a matrix of TF-IDF features.

4. Model Training

4.1 **Logistic Regression**

Description: A simple linear model for binary and multiclass classification.

4.2 **Random Forest**

Description: An ensemble method that combines multiple decision trees to improve classification accuracy and control overfitting.

4.3 **Support Vector Machine (SVM)**

Description: A powerful model that finds the optimal hyperplane for separating classes in high-dimensional space, with support for non-linear classification via kernels.

4.4 **Naive Bayes**

Description: A probabilistic classifier based on Bayes' theorem with strong (naive) independence assumptions between features, often used for text classification tasks.

5. Evaluation

5.1 Techniques Used

- Accuracy Score, Precision, Recall, F1 Score

5.2 Evaluation Table

Vector	TfidfVectorizer				CountVectorizer			
Models	Logistic Regression	Random Forest	SVM	Naive Bayes	**Logistic Regression**	Random Forest	SVM	Naive Bayes
Accuracy	0.85	0.85	0.85	0.82	**0.91**	0.85	0.82	0.87
Precision	0.76	0.76	0.76	0.74	**0.92**	0.76	0.74	0.86
Recall	0.85	0.85	0.85	0.82	**0.91**	0.85	0.82	0.87
F1-Score	0.80	0.79	0.80	0.76	**0.89**	0.79	0.77	0.83

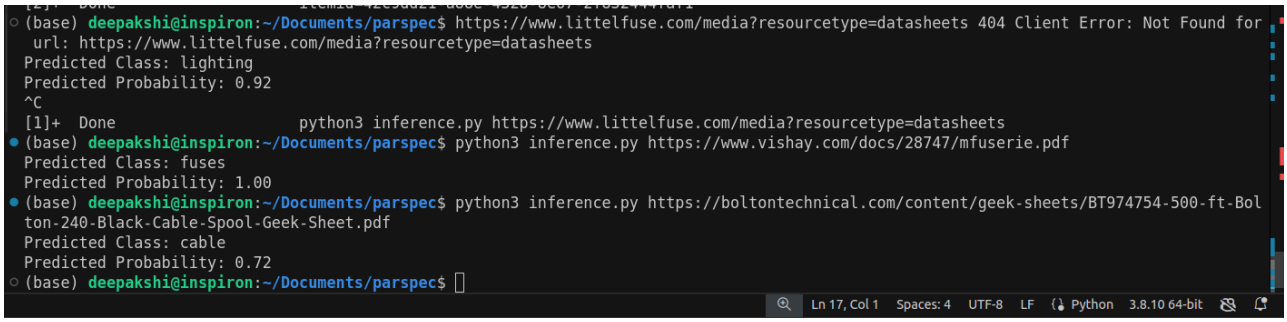
6. Inference Pipeline

6.1 Pipeline Description

- **Input**: URL of a new document.
- **Process**: Extracts and preprocesses text, vectorizes it, and uses the trained model to predict the class.
- **Output**: Predicted class and its probability.

6.2 Implementation Details

- **Steps**:
    1. Extract text from the provided URL.
    2. Preprocess the text.
    3. Vectorize the text using the best-performing vectorizer.
    4. Predict the class using the trained model.
    5. Return the prediction and its probability.



7. Conclusion

- **Best model :** Logistic Regression with Count Vectorizer.
- **Future scope :** We can apply word2vec technique for vectorization of the text.
- **Problems :** Majorly the prediction fails for the category type **“others”** as there is no pattern recognised by the model for that category.