The second model family we used is multi-class Logistic Regression, which is powerful for text-classification. After converting the text responses in columns A, F, and I into TF-IDF vectors, the feature matrix becomes 1723 dimensions, and sparse. Logistic Regression handles this setting well because it learns linear decision boundaries and optimizes a convex objective, making it stable even when the vocabulary is large.

We trained logistic regression using the **liblinear** optimizer in sklearn. Liblinear minimizes the **L2-regularized cross-entropy loss** using a **Coordinate Descent** method, which updates one weight at a time and is effective for high-dimensional sparse TF-IDF features. Because the logistic loss is convex, liblinear converges deterministically and therefore does not require a learning rate schedule or early stopping.

We used a **50% / 25% / 25% train, validation, test split**. Giving 50% of the data to training improved model performance, because TF-IDF models benefit from having more text examples to learn stable vocabulary weights. The model was trained only on the training set, while the validation set was used exclusively for hyperparameter tuning. The test set was kept completely unseen during tuning and was only evaluated once at the end to avoid data leakage. For TF-IDF, we varied the minimum document frequency (**min_df**), the **maximum number of features(max_features)**, and unigrams and bigrams with **min_df = 5**, and assigned different feature caps to each text column (100 for A, 3000 for F, 1500 for I) so that we kept important phrases without making the feature space unnecessarily large. These choices were selected after trying a small, reasonable range of settings and picking the ones with the best validation accuracy.

To evaluate the model, we used **accuracy, precision, recall, macro-F1, and a confusion matrix**. Accuracy provides an overall performance summary, but we included macro-F1 to give each label equal importance and prevent the classifier from appearing strong simply because one class is slightly more common. Precision and recall allowed us to examine how confidently and consistently the model predicted each label. The confusion matrix further revealed the specific error pattern: ChatGPT responses were the easiest to classify, while Claude and Gemini were more frequently confused with each other. These metrics together provides a clearer picture of the model's strengths and weaknesses beyond a single accuracy value.

We implemented our model in using **sklearn** for the logistic regression classifier, the TF-IDF text vectorizers, the ColumnTransformer, and the StandardScaler. Pandas and NumPy were used for loading data, managing tables, and numerical operations. We also implemented **custom feature engineering functions** for the multiselect survey questions (C and E), manually converting them into one-hot vectors based on a fixed list of target task categories.