**INFRDD ASSIGNMENT**

This document will explain all the steps I have performed in detail.

**Introduction**

The dataset comprises multiple dataframes, each corresponding to an image of a form. Each piece of information extracted from the form is represented as a token and occupies a row in the dataframe. The available features include:

* **start\_index** and **end\_index:** These columns indicate the position of the token as if the entire image were flattened into a single line.
* Four columns represent the x and y coordinates of the top-left and bottom-right corners of the token.
* **transcript:** This column contains the information captured within the token.
* **field:** This column serves as the label, identifying the type of information contained in the token.

**Exploratory Data Analysis and Feature Engineering**

This section draws significant inspiration from the Exploratory Data Analysis conducted in Luis Fernando Torres's project on Wine Quality: EDA, Prediction, and Deployment.

Our initial observation reveals that the data is highly skewed, with the “OTHER” label significantly outnumbering all other labels combined (see Fig. 1). This indicates a clear need for either undersampling the “OTHER” category or oversampling the minority labels. Even after excluding the “OTHER” label, a slight skewness in the remaining labels persists (see Fig. 2); however, this skewness is considerably less pronounced than before.

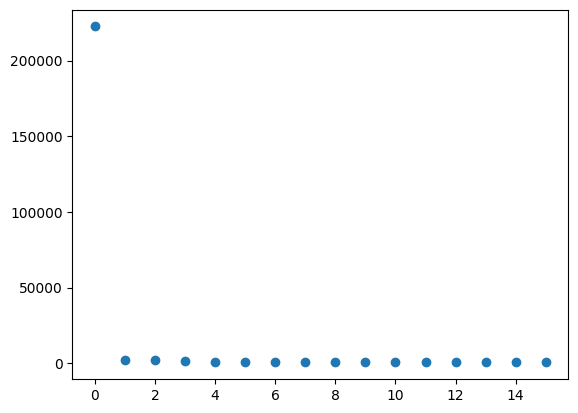


Figure 1: Count of each label

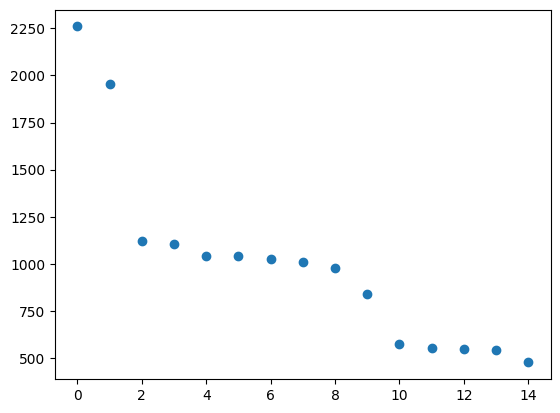


Figure 2: Count of all other labels

Next, we introduce three additional features: **File No.**, which indicates the origin of each row within the respective dataframe; **index\_len**, derived from the difference between the start and end index columns; and **x\_center** and **y\_center**, which represent the coordinates of the token's center point. Additionally, we eliminate the **transcript** feature. This decision stems from the fact that this feature contains diverse information, including numbers, codes (e.g., 72-8989-56), and strings. While this feature could potentially enhance our model through various NLP techniques, I currently lack the time and expertise to pursue that avenue. Given more time, I could learn and implement these techniques effectively. Therefore, for our current analysis, I have opted to remove this feature entirely.

Moreover, we analyze the correlation heatmap of the features to identify which ones are most closely associated with our labels. The heatmap for the complete dataset is presented in Fig. 3, while the heatmap for the dataset excluding the “OTHER” label—hereafter referred to as the filtered dataset—is shown in Fig. 4.

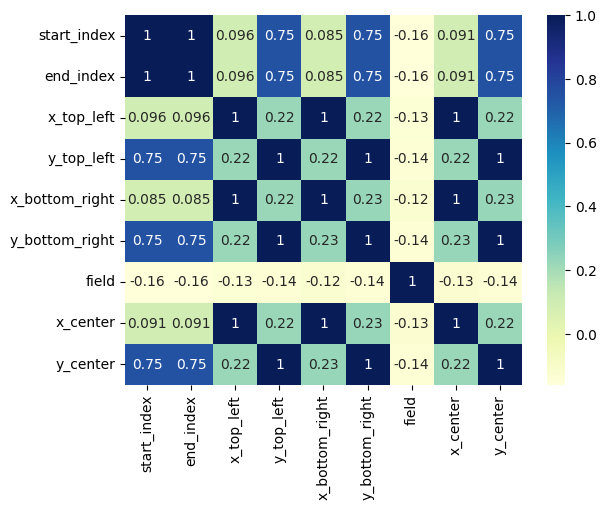


Figure 3: Heatmap of entire dataset



Figure 4: Heatmap of filtered datatset

From the heatmaps, it becomes evident that the **x\_center** and **y\_center** features effectively encapsulate the information provided by the top-left and bottom-right coordinates. Consequently, we will remove these four features from our dataset. Additionally, in the complete dataset, we observe that the **field** label exhibits a near-equal correlation with all other features. However, this trend shifts in the filtered dataset shown in Fig. 4. The disparity indicates that the overwhelming presence of the “OTHER” label—which serves as a placeholder for irrelevant information—is influencing the results. To address this, I propose creating two classification models: the first will determine whether a given data point corresponds to the “OTHER” label. If a data point does not belong to the “OTHER” category, it will then be directed to a specialized model tasked with classifying it into the relevant labels. We will delve deeper into this approach in the following section.

Next, we examine the mean, standard deviation, and other statistical measures of the data and find that our dataset is neither standardized nor normalized. However, given that we are pursuing a classification problem (specifically utilizing Random Forest in this case), such preprocessing is not necessary. We also analyze the datasets through violin plots and box plots.

Out of all the remaining features, we conduct Recursive Feature Elimination with Cross-Validation (RFECV) to identify the most significant features for training our model. The features selected include **start\_index**, **end\_index**, **x\_center**, **y\_center**, and **index\_len**. This analysis was performed in two Jupyter notebooks: “Data Analysis full.ipynb” for the complete dataset and “Data Analysis filtered.ipynb” for the filtered dataset.

**The Model Pipeline**

Based on the preceding analysis, I have opted to implement a two-tier classification system. The first model processes the entire dataset, classifying entries as either “OTHER” or “not other.” Its primary function is to determine whether the provided entry is relevant or not. The second model is trained on the filtered dataset, meaning it focuses on identifying the actual label of entries deemed relevant.

The overall workflow during training begins by consolidating the entire dataset into a single dataframe, followed by feature engineering on the columns. Afterward, a copy of this dataframe is created to accommodate the distinct requirements of both models. For the first model, all relevant labels are converted to “not other,” and we then employ undersampling to address the skewed class distribution. In contrast, the second model drops all entries labeled as “OTHER.” Following this, both models are trained on their respective datasets.

For testing, the data first undergoes feature engineering before being processed by the first model. If the result is classified as “OTHER,” it is reported as such. If not, the entry is forwarded to the second model for further classification.

For this project, I have chosen to use Random Forest Classifiers as the baseline model. While there are certainly other models that may yield better performance, additional time would be required for their implementation and evaluation.

**Note:** Regarding the undersampling technique, I initially considered using Clustering Centroid, which replaces high-count data with cluster centers to maintain the underlying patterns while reducing quantity. However, this approach proved computationally intensive; on my personal device, the code ran for hours before I had to interrupt it manually. Consequently, due to time constraints, I opted for RandomUnderSampling instead.

**Hyperparameter Tuning**

To determine the optimal value for the **n\_estimators** parameter, I employed **Stratified K-Fold Cross Validation**, which preserves the distribution of labels in each fold—an important consideration for addressing our skewed dataset. After testing various ranges, I found that the ideal **n\_estimators** value is **250** for both models, yielding an average accuracy of approximately **96.57%** for the first model and around **94.96%** for the second model. This indicates that our model is well-suited for the task at hand.

Subsequently, I proceeded to train both models separately on their respective dataframes, using **n\_estimators** values close to **250**, as determined through cross-validation. The accuracy achieved by the first model is illustrated in Fig. 6, while Fig. 7 displays the accuracy for the second model. The accuracy levels are quite impressive (ranging from **94% to 96%**), with minimal variation. Therefore, I confidently selected **250** estimators as the final tuned value and proceeded to train the model using the complete dataset.

All the analyses mentioned above were conducted in the Jupyter notebook titled “Model Analysis.ipynb.”

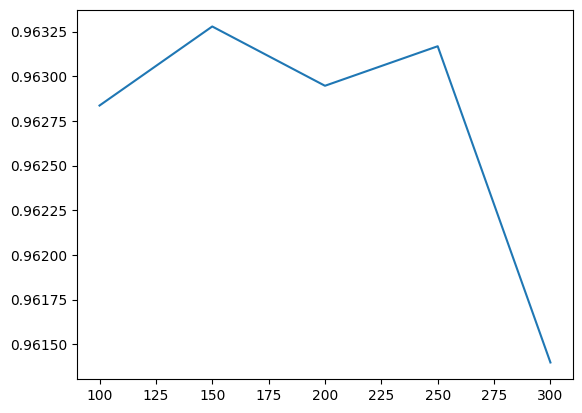


Figure 6: Accuracy vs n\_estimators for 1st model

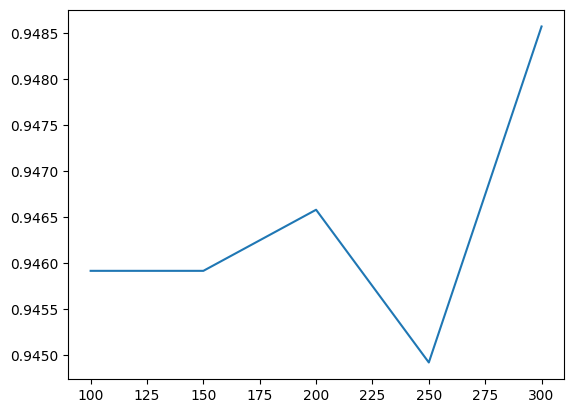


Figure 7:Accuracy vs n\_estimators for 2nd model

**Results**

So, finally I decide to go with ‘n\_estimators’ = 250 and then train the entire training dataset over it. Then, just to test our model a little, I use just one single file from the validation dataset. Surprisingly, the model gives me a 100% accuracy.

The entire code for this section is present in ‘main.ipynb”.

**Error Analysis**

In addition to the performance metrics, I conducted an error analysis to identify potential areas of improvement in the model's predictions. By analyzing the code and the predictions, I observed the following key points:

1. **Misclassification Patterns:** The model frequently misclassified certain entries as “OTHER,” particularly those with unconventional formatting or ambiguous content. This issue was evident in entries that contained a mix of numerical and alphanumeric characters or were poorly structured. For instance, forms with codes or unique identifiers (like 72-8989-56) often resulted in confusion, leading the model to incorrectly classify them.
2. **Class Imbalance Impact:** The significant skewness in the dataset, particularly the overwhelming presence of the “OTHER” label, adversely impacted the model's performance. Although I implemented undersampling techniques to mitigate this issue, the model still exhibited a tendency to favor the “OTHER” class during classification. This was evident when reviewing the false positives, where relevant entries were classified as “OTHER” due to insufficient representation in the training set.
3. **Feature Importance and Selection:** Upon reviewing feature importance, I found that certain features like start\_index, end\_index, x\_center, and y\_center contributed positively to the model's predictions. However, the removal of the transcript feature, which could potentially provide contextual information, might have limited the model’s ability to distinguish between similar tokens effectively. Future iterations may benefit from re-evaluating the decision to eliminate this feature.
4. **Model Limitations:** While Random Forest proved to be a robust baseline model, there were instances where it struggled with highly complex patterns in the data. The reliance on ensemble learning can sometimes lead to difficulties in capturing intricate relationships present in the dataset, especially when the underlying data distribution is highly skewed.

Moving forward, I plan to investigate these specific instances more closely to understand the root causes of misclassification. This will include:

* Implementing additional preprocessing steps to standardize the format of the input data.
* Exploring advanced natural language processing (NLP) techniques to extract meaningful features from the transcript column.
* Testing different machine learning algorithms, such as Gradient Boosting or Support Vector Machines, to assess their performance in handling the misclassification issues observed.

By addressing these areas, I aim to enhance the model's accuracy and reliability for future predictions.