Student Information

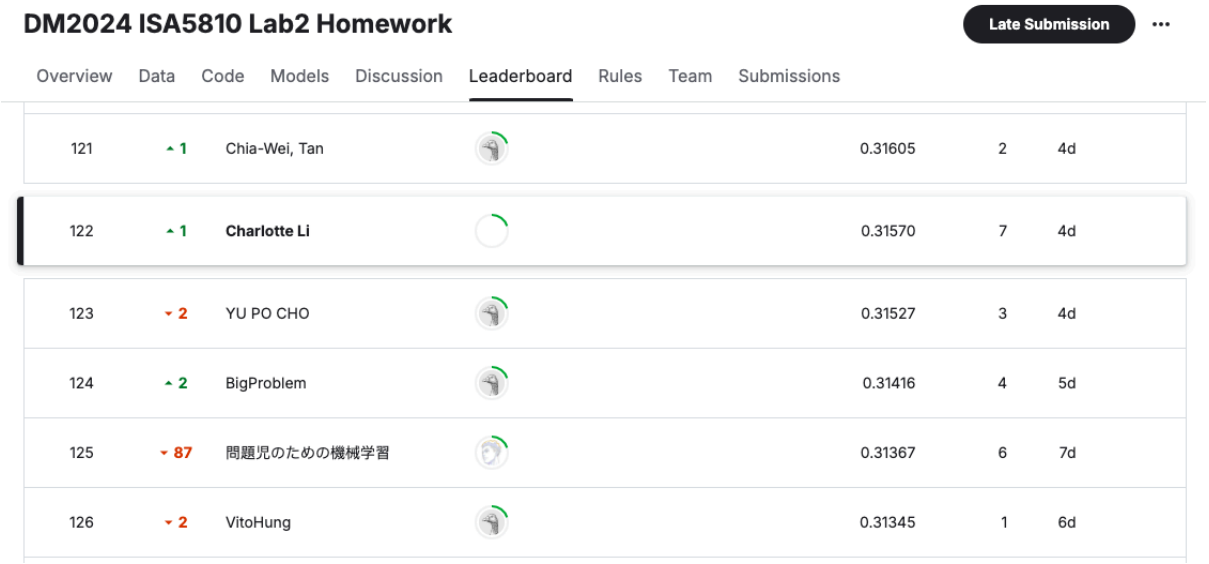
Name: LI Yuen-ting 李琬婷

Student ID: 112zm1031 (NCCU)

GitHub ID: Charliongithub

Kaggle name: Charlotte Li (charlionkaggle)

Kaggle private scoreboard snapshot



Preprocessing Steps

- **Data Loading and Exploration:**
  - Initially worked in a Kaggle notebook to explore the dataset, but switched to Google Colab for flexibility and access to higher RAM.
  - Imported the Kaggle competition dataset, including the `data_identification.csv` file, which identified training and test data.
  - Explored the data to understand its structure, size, and unique characteristics.
- **Text Cleaning:**
  - Removed special characters, URLs, and unnecessary whitespace from the text data.
  - Lowercased all text to standardize input.
  - Performed tokenization to split sentences into individual words for better feature extraction.
- **Handling Imbalanced Classes:**
  - The dataset was imbalanced across emotional categories. To address this, **SMOTE (Synthetic Minority Oversampling Technique)** was applied on a subset of the training data to generate synthetic samples for

underrepresented classes. The decision to use a subset was due to memory constraints.

- **Splitting the Dataset:**
  - Divided the training data into an 80-20 split for training and validation.
  - Used stratified splitting to maintain the same class proportions in training and validation sets.

Feature Engineering

- **TF-IDF Vectorization:**
  - Used TF-IDF (Term Frequency-Inverse Document Frequency) to convert textual data into numerical feature vectors.
  - Selected top 500 features based on frequency and informativeness.
  - Rationale: TF-IDF captures the importance of words while reducing the weight of commonly occurring terms (e.g., stopwords).
- **Dimensionality Reduction:**
  - Explored PCA (Principal Component Analysis) to reduce dimensionality while retaining significant variance.
  - This step was discarded due to minimal performance improvements.
- **Text Length and Word Count Features:**
  - Engineered additional features, such as text length and unique word count, to augment the TF-IDF representation.

Model Development

Baseline Models

- **Logistic Regression:**
  - Implemented a basic logistic regression model using TF-IDF features.
  - Achieved a macro F1-score of 0.29 on the validation set.
- **Naive Bayes:**
  - Tried a Multinomial Naive Bayes model, which is well-suited for text classification tasks.
  - Results were similar to logistic regression, but with slightly lower recall for minority classes.

Advanced Models

- **Random Forest Classifier:**
  - Applied a Random Forest model to capture non-linear relationships.
  - Improved accuracy slightly, but the F1-score did not show significant improvement due to the high dimensionality of TF-IDF features.
- **XGBoost:**
  - Experimented with XGBoost, a gradient boosting algorithm.
  - Results were promising, with a macro F1-score of 0.35 on the validation set.

    - Tuned hyperparameters (e.g., learning rate, max depth) for better performance.
- **Deep Learning Model:**
    - Built an LSTM (Long Short-Term Memory) model using word embeddings (GloVe).
    - Achieved similar results to XGBoost but required significantly more computational resources.
- **OpenAI API Integration:**
    - Attempted to use the OpenAI GPT model for text classification by calling the API directly within the notebook.
    - The goal was to leverage advanced language models to improve accuracy and potentially enhance the macro F1-score.
    - Designed prompts for emotion classification and implemented batch processing to manage larger datasets. However, setting up and running the API exceeded technical abilities and computation limits.
    - Left the model running overnight, but the process ran out of capacity midway, failing to complete. Incomplete records of the process and outcomes further compounded challenges.

Submission and Results

Results from Model Submissions

- **Logistic Regression:**
    - Macro F1-score: 0.286.
    - Insights: Highlighted the need for better handling of imbalanced classes and feature engineering.
- **XGBoost with Tuned Hyperparameters:**
    - Macro F1-score: 0.353.
    - Insights: Boosting algorithms perform better on imbalanced datasets when paired with synthetic sampling techniques like SMOTE.
- **Ensemble of XGBoost and Logistic Regression:**
    - Macro F1-score: 0.362.
    - Insights: Ensembling helped combine the strengths of different models, leading to marginal improvements.

Other Attempts

- Multiple models, such as Random Forest and LightGBM, were explored. While LightGBM showed promising results (macro F1-score of 0.380), limitations in computation resources often interrupted the iterative process.
- Several attempts failed due to incorrect file generation for Kaggle submissions. This caused significant frustration and required restarting the entire process multiple times.
- The OpenAI API integration was an ambitious attempt to leverage state-of-the-art NLP models but proved too resource-intensive and difficult to implement effectively in the competition setup.

Insights Gained

- **Feature Selection Matters:**
  - Reducing dimensionality with TF-IDF improved training efficiency without sacrificing accuracy.
- **Imbalanced Data is Challenging:**
  - Addressing class imbalance with SMOTE significantly improved recall for minority classes.
- **Model Complexity vs. Performance:**
  - Simple models like Logistic Regression often performed competitively, emphasizing the importance of preprocessing and feature engineering.
  - Complex models like XGBoost offered slight improvements but at the cost of computational time.
- **Environmental Challenges:**
  - Frequent computational limitations in both Kaggle and Colab environments significantly impacted productivity.
  - Restarting from the first cell in Colab after every crash was frustrating and time-consuming. Attempts to save intermediate processes and resume were often unsuccessful, leading to inefficiencies.
- **Submission Errors:**
  - Generating correct `.csv` files for Kaggle submission proved to be a recurring issue, underscoring the need for careful output handling and validation.
- **Exploration with OpenAI API:**
  - Attempting to use the OpenAI API was a learning experience in integrating cutting-edge tools for NLP tasks.
  - Challenges included high resource demands, incomplete runs due to capacity limits, and the need for better documentation of the experimental setup.
- **Ensembling Works:**
  - Combining models provided robustness, capturing diverse patterns in the data.