mbti-prediction

MBTI Personality Prediction using Machine Learning

Introduction

Understanding personality types through textual data has significant implications for psychology and marketing. The Myers-Briggs Type Indicator (MBTI) is a widely used tool for personality assessment, and machine learning techniques have been increasingly applied to predict MBTI types. This project focuses on the MBTI personality type dataset, which contains text data labeled with MBTI types [1]. We will preprocess the text data and apply both unsupervised clustering and supervised classification techniques to enhance the accuracy of MBTI personality type predictions, building on methods used in previous research, including Rajshree's MBTI Personality Predictor.

Problem Definition

The problem we are going to solve with machine learning models is classifying and predicting a user's Myers-Briggs Type Indicator (MBTI) based on their posts on social media, specifically Twitter. We will use word embeddings to convert the text data into word vector representations. Both supervised and unsupervised machine learning models will be applied to classify a user's MBTI based on their posts. Unsupervised techniques such as KMeans, DBSCAN, Fuzzy KMeans, and GMM will be employed for clustering. Supervised machine learning models like RNN and logistic regression will be used for classifying the user's MBTI personality.

Compared to previous literature, we combine unsupervised and supervised machine learning models to obtain more robust classification results and use some unsupervised models that have not been previously used for this task. Previous research has primarily focused on supervised machine learning techniques. For instance, Nisha et al. (2021) used Naïve Bayes, Support Vector Machine, and XGBoost classifiers to predict the personality of Twitter users. Ryan et al. (2023) used logistic regression, linear support vector classification, and random forest to predict user personalities. Mushtaq et al. (2020) employed K-Means and gradient boosting techniques to predict users' personality types.

Methods

Preprocessing Methods:

- **Data Cleaning**:

The following steps were implemented to systematically clean and standardize the text data:

1. **Removing Non-Text Elements**
   - **HTML Tags and URLs**: These elements were removed as they do not add meaningful content for analysis.
   - **Special Characters**: Punctuation and special symbols were stripped from the text to reduce noise and focus on essential words.
2. **Emoji Replacement**
   - Emojis were converted to descriptive text using regex patterns. This allowed us to retain the sentiment and meaning conveyed by emojis in a text-friendly format (e.g., 😊 converted to ":smiling_face:").
3. **Stopword Removal**
   - Common stopwords (e.g., "and," "the," "is") were removed from the text. This step helps focus on the most relevant words by discarding frequent but less meaningful words.
4. **Tokenization**
   - Text was split into individual words (tokens) to enable more granular manipulation of each word, allowing each to be processed independently in later steps.

5. **Lemmatization**
    - Each word was converted to its base form (lemma) to reduce variations. This step standardizes words (e.g., "running" becomes "run") and ensures consistency while preserving the original meaning. Lemmatization was preferred over stemming for more linguistically accurate results.
6. **Saving the Cleaned Data**
    - The final cleaned text data was saved as a new dataset, prepared for further analysis and modeling.

These preprocessing steps improve text data quality by focusing on meaningful content, standardizing variations, and reducing irrelevant noise. The resulting dataset is optimized for natural language processing and machine learning tasks.

- **Feature Engineering**:
    - **TF-IDF**: We calculated word importance by analyzing the frequency of each word within a post and its rarity across all posts. After observing that word counts typically ranged from 0 to 2000, we set a max feature limit of 1500 and used sklearn's TfidfVectorizer to obtain a feature matrix.
    - **Word Embedding**: We used word embeddings with Gensim's Word2Vec, setting the embedding dimension to 100. By averaging the vectors of all words in each post, we transformed each post into a single 100-dimensional vector that captures the overall meaning of its content.
    - **Sentiment Analysis**: We applied TextBlob's sentiment analysis to capture the polarity and subjectivity in posts. Each post was then represented as a 6-dimensional vector that includes both word-level and overall scores.
    - **Sentence Embedding**: We split the posts into a series of complete sentences and use Sentence-BERT to generate sentence embeddings for each sentence. The final 384-dimensional feature vector is obtained by averaging the embeddings of all sentences. Experimental results show that the feature matrix obtained through sentence embedding achieves higher accuracy in training.
- **Dimensionality Reduction**:
  We use PCA to reduce the dimensionality of the feature space to accelerate the training process of certain models, such as reducing a 384-dimensional feature matrix to 100 dimensions.

Machine Learning Algorithms:

- **Unsupervised Models**:
    - **GMM**: Identify the probabilistic distribution of user posts.
- **Supervised Models**:
    - **Random Forest**: As our task is naturally a classification problem and our word vector has a relatively large dimension as compared to our dataset size. Thus fitting a Random Forest Classifier is a plausible and robust choice for our task. For this model training, a random forest classifier is trained on top of all the 3 word vector representations to perform a multi-class classification task on the 16 MBTI labels. Then, among the word vector representation that performs the best, we attempt 4 per-category binary classification on top of each i/e, s/n, t/f, j/p group and obtained 4 separate Random Forest classifiers.
    - **Logistic Regression**: Logistic Regression converts text into numerical features and uses them to predict MBTI personality types. It's fast and effective for classifying text based on learned patterns.
    - **Recurrent Neural Networks**: RNN is widely used in processing time series data. RNN model has circular linkage, so that it can remember and relay information from previous time steps.

Expected Results and Evaluation

Random Forest

As the dataset label has an imbalanced nature, during the model training process, the hyperparameter `class_weight="balanced"` is used to avoid model begin overly biased towards one side.

The initial attempt in training the Random Forest classifer was based on a multi-class classification over the 16 MBTI types. However, the overall accuracies and f1-score are not ideal (around 0.23 to 0.29 for different word vector models). As people commonly believe that each character in MBTI meansures an unique dimension in personality, a following attempt to divide the whole MBTI label as 4 disjoint binary classification tasks is implemented.

Our model accuracies for the 4 binary classification tasks are around 0.65~0.8 and a detailed precision, recall, and f1-measure are reported in below. According to the result, the model has a relatively acceptable accuracy, while the f1-score indicates that we may still suffer from the fact that our dataset is largely imbalanced.

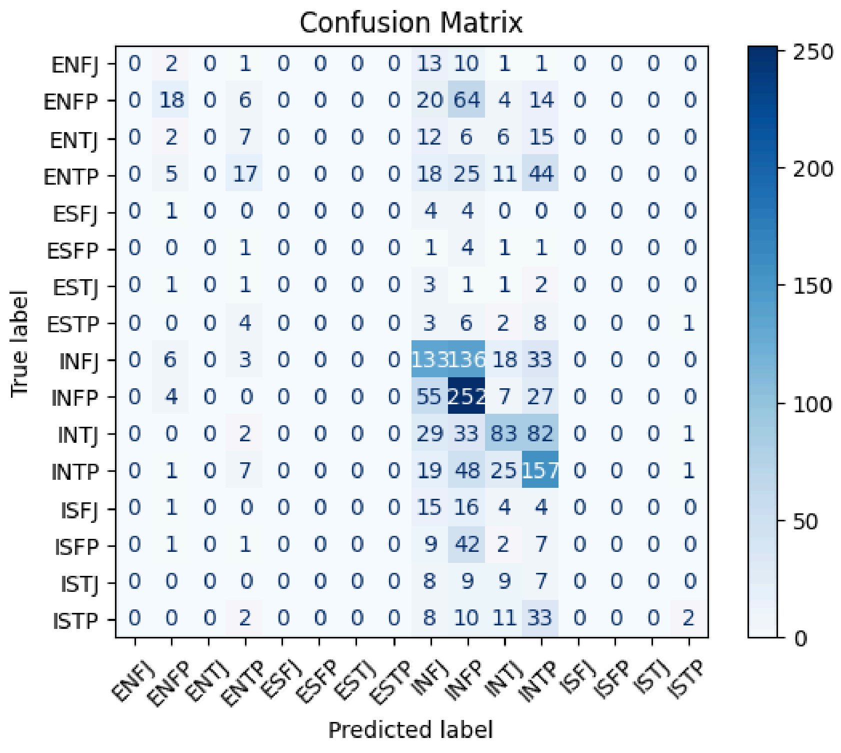
We may attempt to further tuning the word vector representation, and may potentially consider (1) using some pre-trained models or (2) further adjust the vector dimension. Potential attempts can also be taken to (3) adjust the Random Forest's hyperparamter or (4) experimenting on other type of classification models.

Naive Approach

```
Accuracy: 0.3816
              precision    recall  f1-score   support

        ENFJ       0.00      0.00      0.00        28
        ENFP       0.43      0.14      0.21       126
        ENTJ       0.00      0.00      0.00        48
        ENTP       0.33      0.14      0.20       120
        ESFJ       0.00      0.00      0.00         9
        ESFP       0.00      0.00      0.00         8
        ESTJ       0.00      0.00      0.00         9
        ESTP       0.00      0.00      0.00        24
        INFJ       0.38      0.40      0.39       329
        INFP       0.38      0.73      0.50       345
        INTJ       0.45      0.36      0.40       230
        INTP       0.36      0.61      0.45       258
        ISFJ       0.00      0.00      0.00        40
        ISFP       0.00      0.00      0.00        62
        ISTJ       0.00      0.00      0.00        33
        ISTP       0.40      0.03      0.06        66

    accuracy                           0.38      1735
   macro avg       0.17      0.15      0.14      1735
weighted avg       0.33      0.38      0.33      1735
```

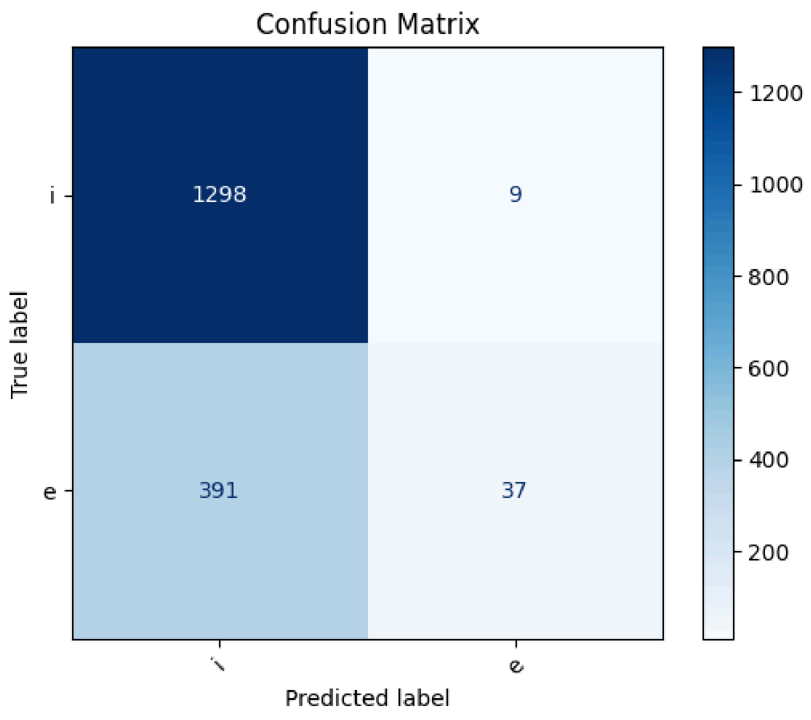


I/E Prediction

```
Accuracy: 0.7695
              precision    recall  f1-score   support

           0       0.77      0.99      0.87      1307
           1       0.80      0.09      0.16       428

    accuracy                           0.77      1735
   macro avg       0.79      0.54      0.51      1735
weighted avg       0.78      0.77      0.69      1735
```

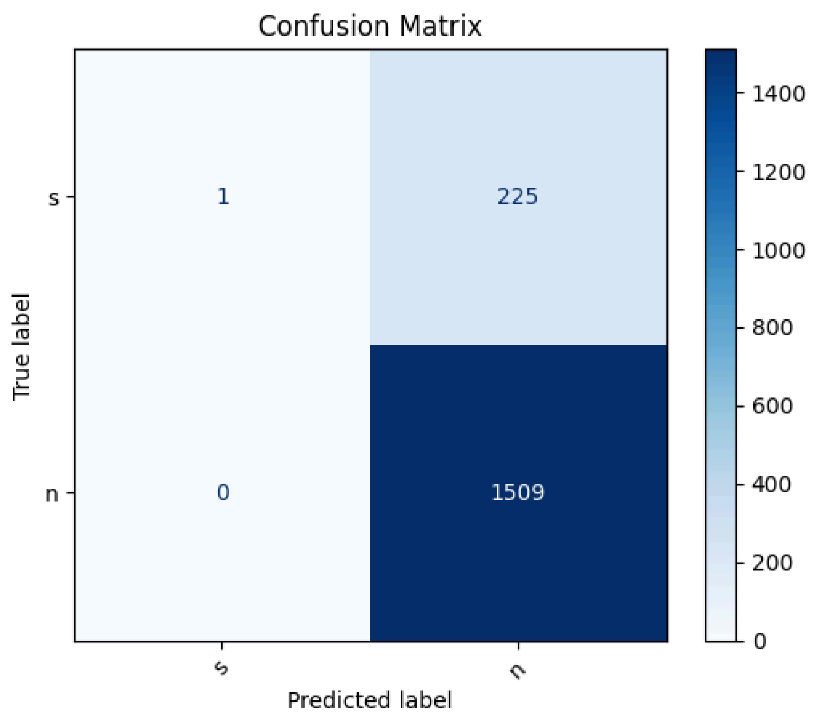


S/N Prediction

```
Accuracy: 0.8703
              precision    recall  f1-score   support

           0       1.00      0.00      0.01       226
           1       0.87      1.00      0.93      1509

    accuracy                           0.87      1735
   macro avg       0.94      0.50      0.47      1735
weighted avg       0.89      0.87      0.81      1735
```

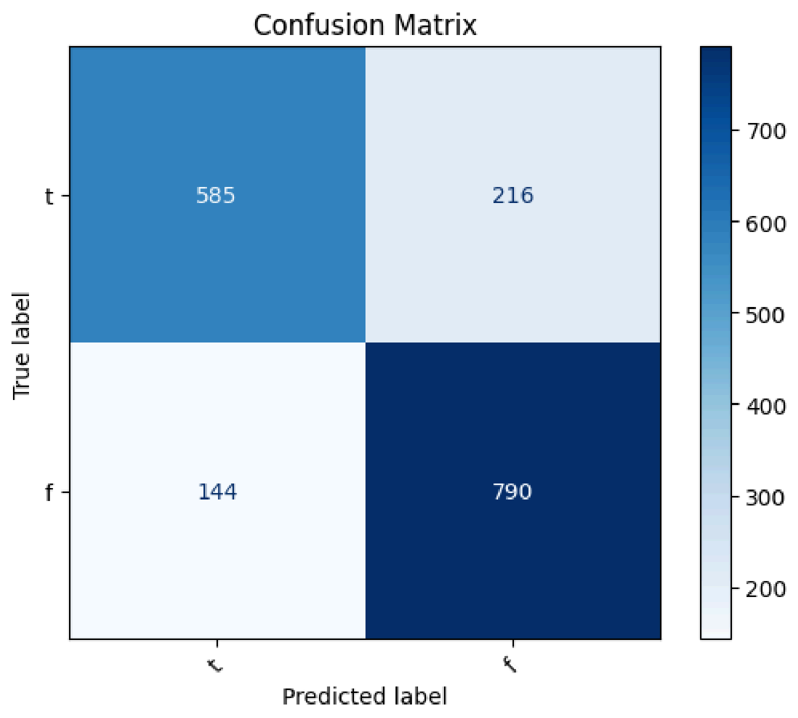


T/F Prediction

```
Accuracy: 0.7925
              precision    recall  f1-score   support

           0       0.80      0.73      0.76       801
           1       0.79      0.85      0.81       934

    accuracy                           0.79      1735
   macro avg       0.79      0.79      0.79      1735
weighted avg       0.79      0.79      0.79      1735
```

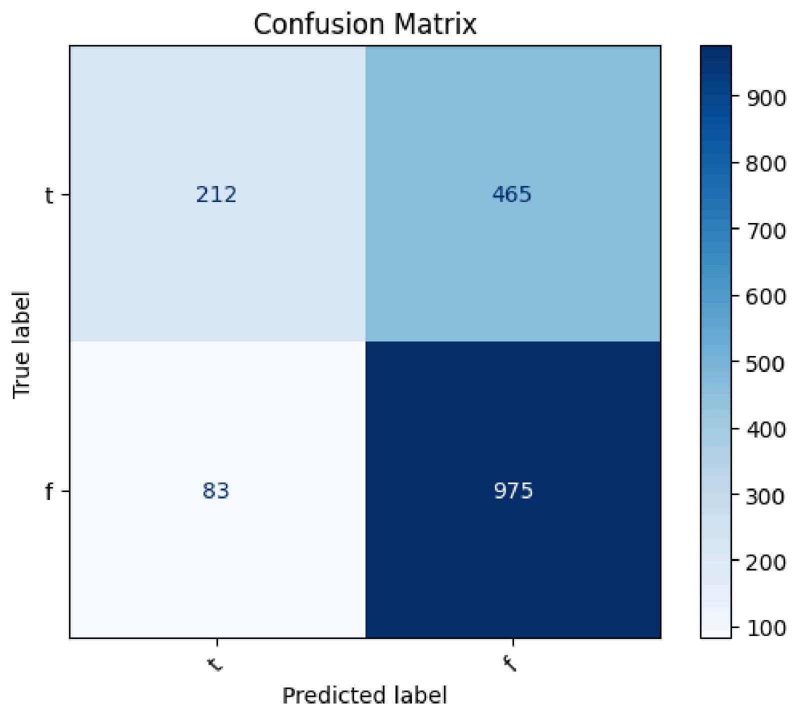


J/P Prediction

```
Accuracy: 0.6841
              precision    recall  f1-score   support

           0       0.72      0.31      0.44       677
           1       0.68      0.92      0.78      1058

    accuracy                           0.68      1735
   macro avg       0.70      0.62      0.61      1735
weighted avg       0.69      0.68      0.65      1735
```



Logistic Regression

Naive Approach

In our initial approach, we trained directly on the dataset with 16 different labels. This was intended to gain a basic understanding of the dataset and identify any immediate issues. However, the model's performance was poor, showing low classification accuracy. This outcome suggests that the dataset is too small for the large number of labels, and the overlapping problem between classes is serious. We need to find a way to augment the dataset.

Naive Approach

```
              precision    recall  f1-score   support

        ENFJ       0.31      0.40      0.35        35
        ENFP       0.52      0.50      0.51       139
        ENTJ       0.43      0.33      0.38        63
        ENTP       0.44      0.50      0.47       135
        ESFJ       0.20      0.12      0.15         8
        ESFP       0.33      0.22      0.27         9
        ESTJ       0.40      0.22      0.29         9
        ESTP       0.45      0.45      0.45        20
        INFJ       0.64      0.59      0.61       308
        INFP       0.62      0.65      0.63       355
        INTJ       0.52      0.53      0.52       202
        INTP       0.64      0.61      0.62       269
        ISFJ       0.42      0.42      0.42        33
        ISFP       0.36      0.27      0.31        55
        ISTJ       0.23      0.31      0.27        35
        ISTP       0.46      0.55      0.50        60

    accuracy                           0.54      1735
   macro avg       0.44      0.42      0.42      1735
weighted avg       0.55      0.54      0.54      1735
```

Dimension-Specific Logistic Regression Models

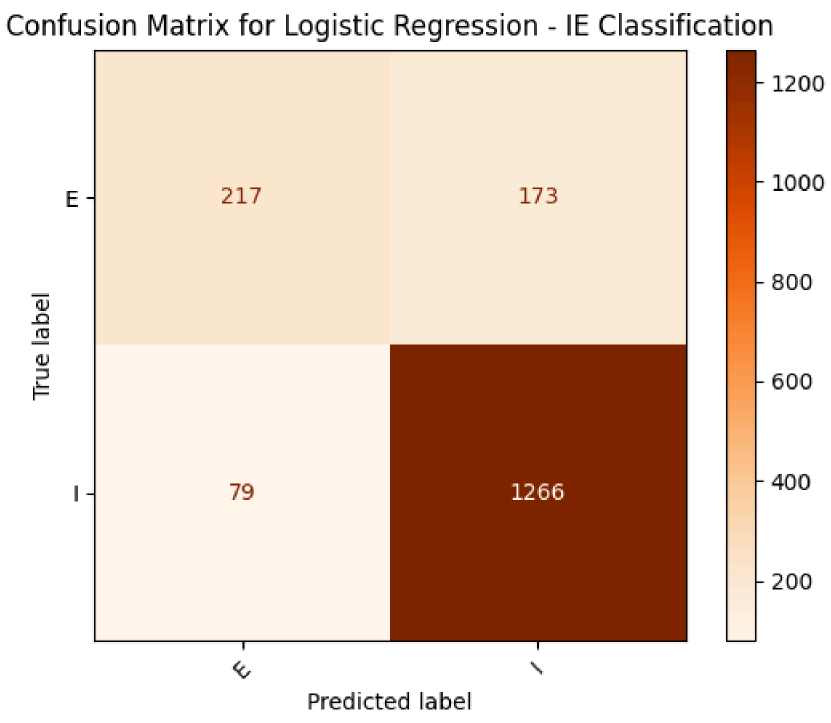
Since MBTI classifies people along four different axis, we decided to apply a logistic model to each of the four dimensions in the dataset to improve performance. The separate logistic regression models showed substantial improvement. This result indicates that treating each axis as an independent binary classification problem can help mitigate the difficulties caused by overlapping patterns in a multi-dimensional setting.

Classify between 'I' and 'E'

```
Number of 'I': 6675
Number of 'E': 1998
              precision    recall  f1-score   support

           E       0.73      0.56      0.63       390
           I       0.88      0.94      0.91      1345

    accuracy                           0.85      1735
   macro avg       0.81      0.75      0.77      1735
weighted avg       0.85      0.85      0.85      1735
```



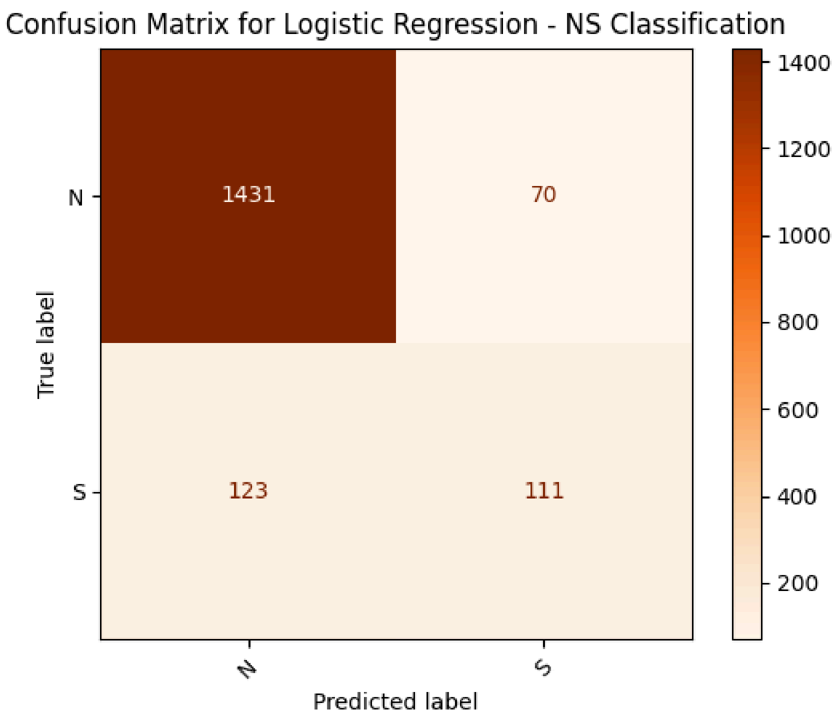
Classify between 'N' and 'S'

```
Number of 'N': 7476
Number of 'S': 1197
              precision    recall  f1-score   support

           N       0.92      0.95      0.94      1501
           S       0.61      0.47      0.53       234

    accuracy                           0.89      1735
```

```
   macro avg       0.77      0.71      0.74      1735
weighted avg       0.88      0.89      0.88      1735
```

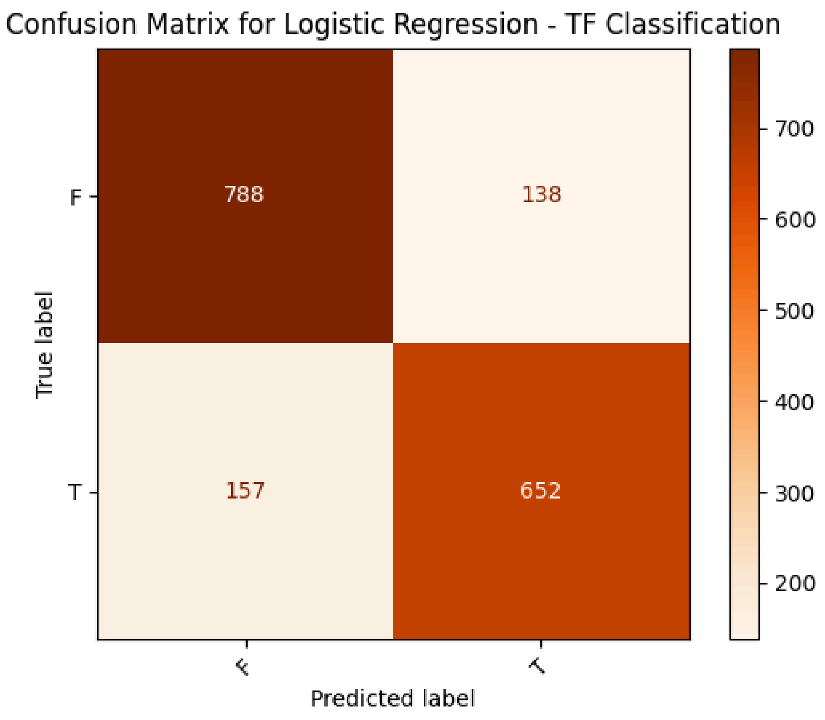


Classify between ‘T’ and ‘F’

```
Number of 'T': 3980
Number of 'F': 4693
              precision    recall  f1-score   support

           F       0.83      0.85      0.84       926
           T       0.83      0.81      0.82       809

    accuracy                           0.83      1735
   macro avg       0.83      0.83      0.83      1735
weighted avg       0.83      0.83      0.83      1735
```

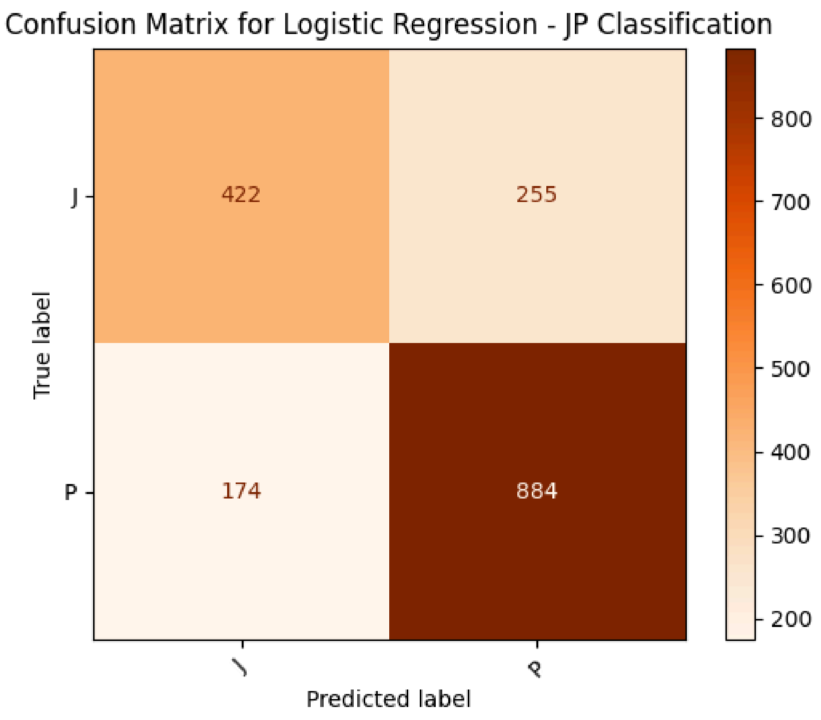


Classify between ‘J’ and ‘P’

```
Number of 'J': 3433
Number of 'P': 5240
              precision    recall  f1-score   support

           J       0.71      0.62      0.66       677
           P       0.78      0.84      0.80      1058

    accuracy                           0.75      1735
   macro avg       0.74      0.73      0.73      1735
weighted avg       0.75      0.75      0.75      1735
```



k-fold Logistic Regression Models

To further validate the model performance, we used K-Fold cross-validation to provide a more reliable evaluation by training and testing the model across multiple data splits, thus reducing the risk of overfitting to a single data partition.

Classify between ‘I’ and ‘E’ Mean Accuracy: 0.8408

Classify between ‘N’ and ‘S’ Mean Accuracy: 0.8973

Classify between ‘T’ and ‘F’ Mean Accuracy: 0.8414

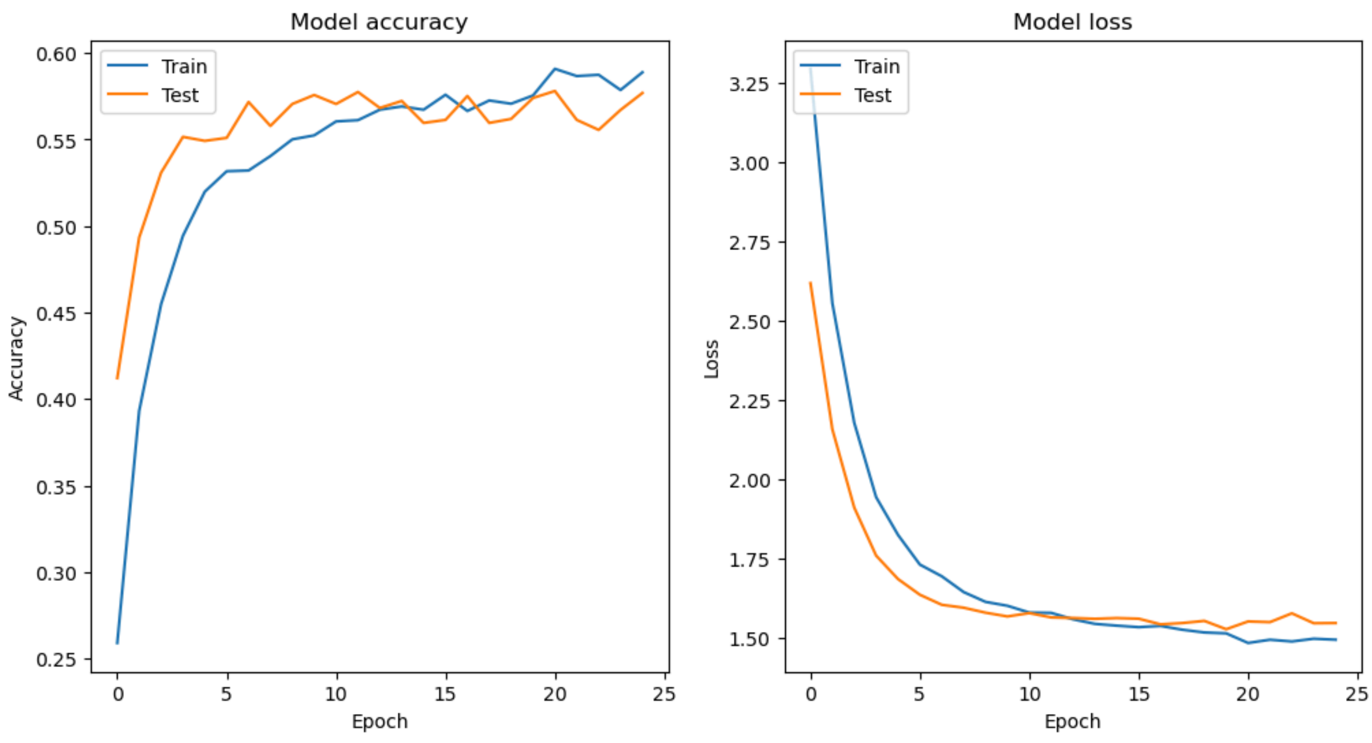
Classify between ‘J’ and ‘P’ Mean Accuracy: 0.7578

Recurrent Neural Networks

We used SimpleRNN as the main parts of the RNN model. But at first, we find that there is overfiting. The training accuracy is low while the test loss is high. So we introduced severl methods to solve this problem. Firstly, we use L2 regularization to control the complexity of the model. We also add BatchNormalization to accelerate the training and stabalize the network. We also introduce Dropout which is 0.4, so 40% neurons will be dropped out in each epoch. And the DropOut rate is added is SimpleRNN and Dense layer.

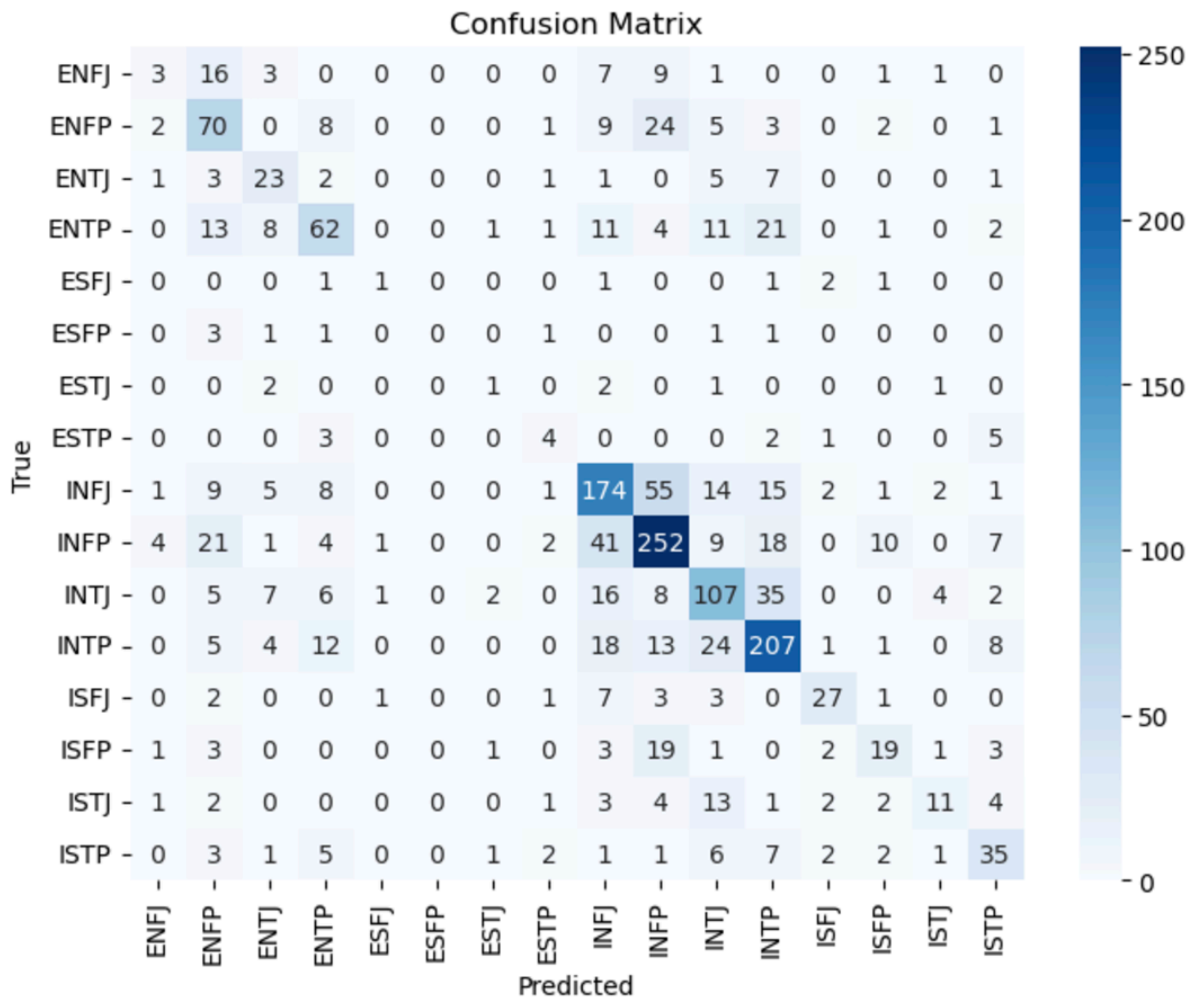
RNN Accuracy and Loss

The following figure shows the RNN model accuracy and loss. We can see that the training and test accuracy increases over epochs.And the training and test accuracy is around 0.6. We can also see taht the model loss decraeses sharpely at the begining of the training and test. The training and tst loass decreases from 3.25 to 1.5. This means that the model can successfully learn and update.



RNN Confusion Matrix

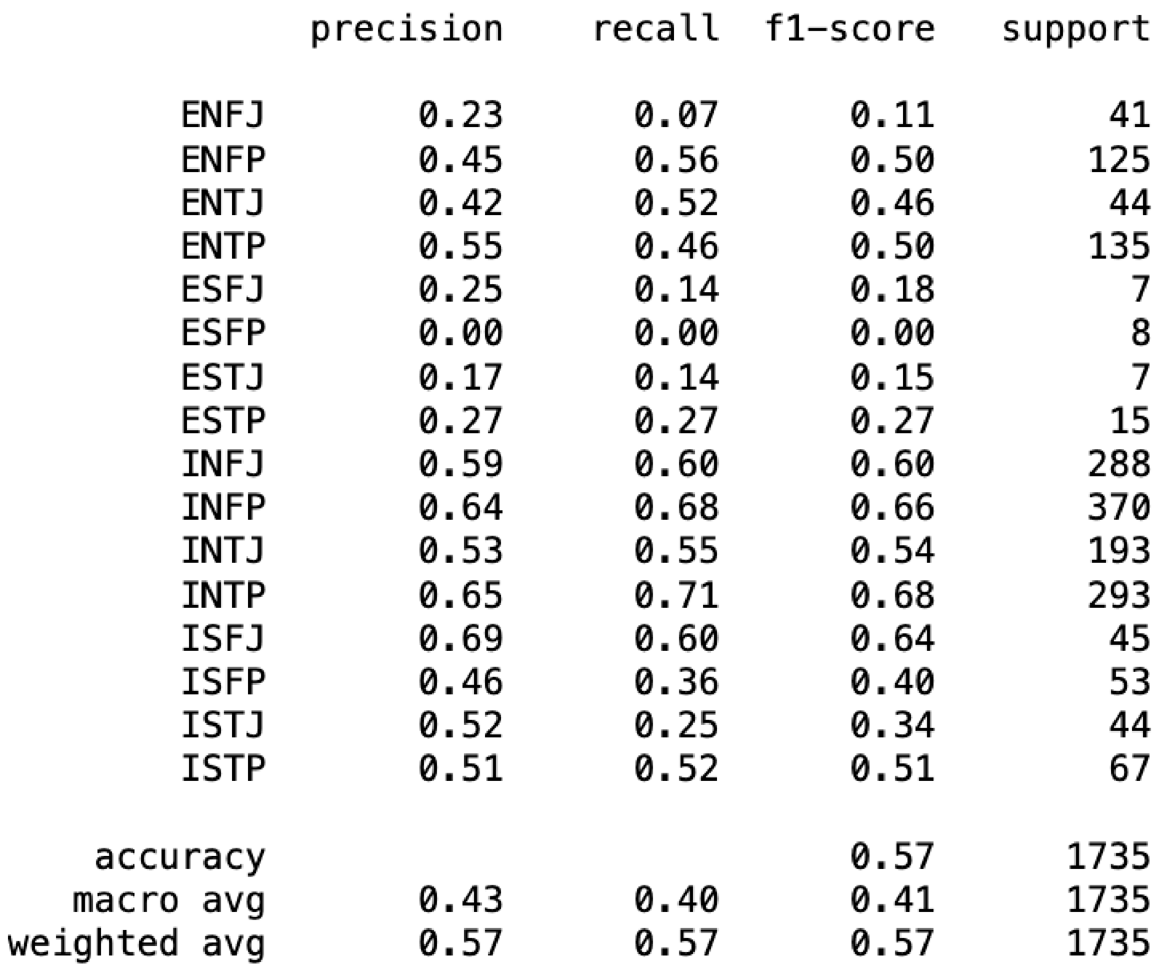
From the confusion matrix, we can see that the diagona area is dark, especially for INFP, INTP, INFJ and INTJ. But some MBTI type don’t have enough samples, like ESFP, ESTJ. Overall, the RNN model can predicts well on ENFP, ENTJ and ENTP.



True \ Predicted	ENFJ	ENFP	ENTJ	ENTP	ESFJ	ESFP	ESTJ	ESTP	INFJ	INFP	INTJ	INTP	ISFJ	ISFP	ISTJ	ISTP
ENFJ	3	16	3	0	0	0	0	0	7	9	1	0	0	1	1	0
ENFP	2	70	0	8	0	0	0	1	9	24	5	3	0	2	0	1
ENTJ	1	3	23	2	0	0	0	1	1	0	5	7	0	0	0	1
ENTP	0	13	8	62	0	0	1	1	11	4	11	21	0	1	0	2
ESFJ	0	0	0	1	1	0	0	0	1	0	0	1	2	1	0	0
ESFP	0	3	1	1	0	0	0	1	0	0	1	1	0	0	0	0
ESTJ	0	0	2	0	0	0	1	0	2	0	1	0	0	0	1	0
ESTP	0	0	0	3	0	0	0	4	0	0	0	2	1	0	0	5
INFJ	1	9	5	8	0	0	0	1	174	55	14	15	2	1	2	1
INFP	4	21	1	4	1	0	0	2	41	252	9	18	0	10	0	7
INTJ	0	5	7	6	1	0	2	0	16	8	107	35	0	0	4	2
INTP	0	5	4	12	0	0	0	0	18	13	24	207	1	1	0	8
ISFJ	0	2	0	0	1	0	0	1	7	3	3	0	27	1	0	0
ISFP	1	3	0	0	0	0	1	0	3	19	1	0	2	19	1	3
ISTJ	1	2	0	0	0	0	0	1	3	4	13	1	2	2	11	4
ISTP	0	3	1	5	0	0	1	2	1	1	6	7	2	2	1	35

RNN Accuracy Report

From this figure, we can see the precision, recall, F1 score and suport. The precision is higher for ENTP, INFP, INTJ, INTP, ISFJ, ISTJ and ISTP. We still needs to tuning the hyperparameters to improve the RNN model performance.



```
              precision    recall  f1-score   support

        ENFJ       0.23      0.07      0.11        41
        ENFP       0.45      0.56      0.50       125
        ENTJ       0.42      0.52      0.46        44
        ENTP       0.55      0.46      0.50       135
        ESFJ       0.25      0.14      0.18         7
        ESFP       0.00      0.00      0.00         8
        ESTJ       0.17      0.14      0.15         7
        ESTP       0.27      0.27      0.27        15
        INFJ       0.59      0.60      0.60       288
        INFP       0.64      0.68      0.66       370
        INTJ       0.53      0.55      0.54       193
        INTP       0.65      0.71      0.68       293
        ISFJ       0.69      0.60      0.64        45
        ISFP       0.46      0.36      0.40        53
        ISTJ       0.52      0.25      0.34        44
        ISTP       0.51      0.52      0.51        67

    accuracy                           0.57      1735
   macro avg       0.43      0.40      0.41      1735
weighted avg       0.57      0.57      0.57      1735
```

Gaussian Mixture Model (GMM)

The implementation utilizes an unsupervised learning approach with Gaussian Mixture Model clustering to classify MBTI personality types. Two different dimensionality settings were experimented with to evaluate the impact on classification performance.

Methodology

1. Text data was transformed into feature vectors using sentence embedding
2. A 16-component Gaussian Mixture Model was employed for unsupervised clustering
3. To optimize performance, the average feature vector of each true label was used as the initialization expectation vector for each component
4. Cluster-to-label mapping was determined using minimum Euclidean distance between component expectation vectors and label average feature vectors
5. Performance was evaluated using confusion matrices and classification reports

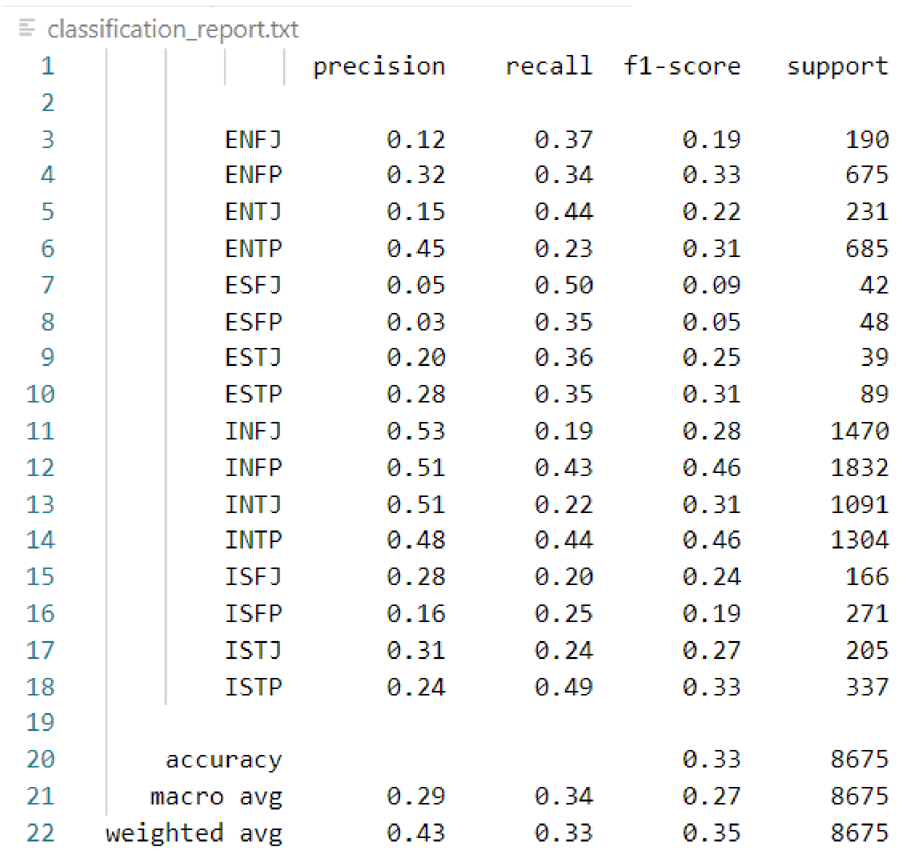
Results Analysis

Classification Performance Matrix

col_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
label																
ENFJ	70	27	16	2	17	9	0	2	14	13	4	6	2	4	1	3
ENFP	89	228	35	28	46	52	4	5	19	91	6	16	6	40	1	9
ENTJ	16	7	102	7	9	19	6	3	7	6	13	13	0	2	6	15
ENTP	36	46	78	159	23	61	7	19	15	33	14	97	4	11	4	78
ESFJ	3	1	1	3	21	4	0	1	1	2	1	0	0	4	0	0
ESFP	5	3	5	3	4	17	0	3	0	3	0	1	0	0	0	4
ESTJ	3	1	4	0	2	6	14	0	0	0	1	3	0	1	2	2
ESTP	1	4	5	6	2	12	1	31	1	2	1	5	1	0	0	17
INFJ	144	99	106	27	97	100	5	5	281	324	61	86	23	44	17	51
INFP	99	176	68	28	76	109	11	6	102	784	33	127	7	138	13	55
INTJ	39	33	143	23	38	77	10	1	40	63	240	202	21	28	38	95
INTP	17	33	67	48	28	78	3	18	25	126	70	574	14	46	19	138
ISFJ	8	15	6	4	22	15	1	0	10	18	4	4	34	13	3	9
ISFP	11	19	9	7	21	36	2	3	5	54	3	14	3	67	3	14
ISTJ	12	9	21	2	13	19	6	2	5	13	10	5	2	9	49	28
ISTP	7	11	25	8	6	24	1	12	3	13	6	36	3	15	1	166

The confusion matrix reveals the distribution of predictions across all 16 MBTI types. The diagonal elements represent correct classifications, while off-diagonal elements show misclassifications between different personality types.

Classification Report



classification_report.txt

```
              precision    recall  f1-score   support

        ENFJ       0.12      0.37      0.19       190
        ENFP       0.32      0.34      0.33       675
        ENTJ       0.15      0.44      0.22       231
        ENTP       0.45      0.23      0.31       685
        ESFJ       0.05      0.50      0.09        42
        ESFP       0.03      0.35      0.05        48
        ESTJ       0.20      0.36      0.25        39
        ESTP       0.28      0.35      0.31        89
        INFJ       0.53      0.19      0.28      1470
        INFP       0.51      0.43      0.46      1832
        INTJ       0.51      0.22      0.31      1091
        INTP       0.48      0.44      0.46      1304
        ISFJ       0.28      0.20      0.24       166
        ISFP       0.16      0.25      0.19       271
        ISTJ       0.31      0.24      0.27       205
        ISTP       0.24      0.49      0.33       337

    accuracy                           0.33      8675
   macro avg       0.29      0.34      0.27      8675
weighted avg       0.43      0.33      0.35      8675
```

384-Dimensional vs 100-Dimensional Embeddings

The higher-dimensional embedding (384D) demonstrated superior performance compared to the 100D embedding:

384-Dimensional Results:

```
Accuracy: 0.33
Weighted Avg:
- Precision: 0.43
- Recall: 0.33
- F1-score: 0.35
```

100-Dimensional Results:

```
Accuracy: 0.27
Weighted Avg:
- Precision: 0.36
- Recall: 0.27
- F1-score: 0.30
```

Key Findings

1. **Initialization Impact**: Using true label average vectors as initialization significantly improved:
    - Model convergence speed
    - Overall classification accuracy
    - Stability of clustering results
2. **Performance by Type**:
    - Strongest Performance:
        - INFP (F1: 0.46)
        - INTP (F1: 0.46)
        - ISTP (F1: 0.33)
    - Challenging Types:
        - ESFP (F1: 0.05)
        - ESFJ (F1: 0.09)
        - ENFJ (F1: 0.19)
3. **Clustering Quality**: Analysis of the confusion matrix reveals:
    - Correctly classified samples (diagonal elements) generally represent the maximum value within their respective rows (Exception: INFJ classification in the 384D model)
    - Demonstrates the effectiveness of sentence embedding combined with GMM for unsupervised clustering

Advantages of the Approach

1. **Stable Performance**: Sentence embedding provided more consistent and interpretable clustering results compared to alternative text vectorization methods
2. **Efficient Convergence**: Mean vector initialization strategy significantly reduced iteration requirements

Limitations and Considerations

1. Individual class precision scores remain relatively low
2. Some personality types (particularly extroverted sensing types) show weaker classification performance
3. Imbalanced class distribution may impact model performance

Future Improvements

1. Explore alternative embedding techniques or fine-tuning existing embeddings
2. Investigate methods to address class imbalance

3. Consider ensemble approaches combining supervised and unsupervised methods
4. Experiment with different numbers of GMM components or alternative clustering algorithms

Model Discussion & Comparison

Supervised Model

The overall performance of the three supervised models is summarized in the table below

Accuracy Type	Random Forest	Logistic Regression	RNN
Multi-Class	~38%	~54%	~60%
Binary (I/E)	~77%	~86%	N/A
Binary (S/N)	~87%	~89%	N/A
Binary (T/F)	~79%	~84%	N/A
Binary (J/P)	~68%	~77%	N/A
Multi-Class F1-Score	0.38	0.54	0.57

By applying the 384-dimension sentence embedding, we can observed some interesting trends

- Logistic Regression out-performs the Random Forest model in all metrics
- RNN slightly out-performs logistic regression model in multi-class prediction and f1-score

Logistic Regression vs. Random Forest

- Potential Overfitting

Given the dataset contains only 1700~ datapoints while the embedding lies in a 384-dimension space, the feature-to-sample ratio is quite high, using a tree-based model like random forest may subject to overfitting problem (After investigation, it really is the case that the model is subjected to serious overfitting, but use some simple grid-search to tune the hyperparamters `n_estimators` and `max_depth` does not provide significant improvements to the overall accuracy)

- Bias & Imbalanced Data

Given the dataset is largely imbalanced, with very-few ES-classes. The tree-based model can likely bias through the majority class and overlook the minority class. However, the weighted loss function used by logistic regression may actually be advantageous in such case

- Feature Space

Given the feature embedding is in a relatively dense and continuous space, the nature of the vector space may favor better the logistic model than a discrete tree-based model

Logistic Regression vs. RNN

- Feature Space

RNN is a powerful model in NLP tasks, especially in modeling (relatively shallow) text sequences. In our case, we applied sentence embedding directly into the model, which essentially make our RNN one-to-one. Our sentence embedding is tuned to capture the information in the sentence quite well. While it is possible that our feature input is not fully utlize the power of RNN so our RNN model out-performs the logistic model, but not to a very high degree.

- Feature-to-Sample Ratio

The RNN model is much more complicated than the logistic model, so give our high feature-to-sample ratio, our dataset may exhibits only “shallow” relationships between text and underlying MBTI types, thus does not make our RNN model very advantageous

Unsupervised Model

Tuning our supervised model to capture clusterings between MBTI types actually takes a lot of efforts. We have applied several clustering algorithms (KMeans, GMM, DBSCAN, …) and our final GMM accuracy is still lower than our supervised model. It might plausible to considering the following reasons

- Nature of the task

Predicting MBTI type from textual input is not a very typical task and the relationship between test and MBTI might be quite complicated. Our unsupervised clustering methods may not be able to directly capture MBTI as the cluster characteristics (like mean or sigma) so it may not work as well as the supervised models

- Lack of Supervision

In our task, we have a grond truth label, the MBTI type, but the training process of the unsupervised models do not actually interact with the label. This could be evident when we “add” some information about the ground truth to the training through initialization, our model performs noticeably better.

Next Steps

- Apply grid-search on the Random Forest model to perform better tuning on the hyperparameters (especially the `n_estimators` and `max_depth`)
- Could potentially attempt the many-to-one RNN model structure for this prediction task
- Instead of using RNN, we can also experiement with the LSTM model to compare and analyze the performance changes
- Additional word or sentence vectorization techniques can also be explored to tune the model performance

References

[1] Nisha, Kulsum Akter, Umme Kulsum, Saifur Rahman, Md Farhad Hossain, Partha Chakraborty, and Tanupriya Choudhury. “A comparative analysis of machine learning approaches in personality prediction using MBTI.” In Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2021, pp. 13-23. Springer Singapore, 2022.

[2] Ryan, Gregorius, Pricillia Katarina, and Derwin Suhartono. “Mbti personality prediction using machine learning and smote for balancing data based on statement sentences.” Information 14, no. 4 (2023): 217.

[3] Mushtaq, Zeeshan, Sagar Ashraf, and Nosheen Sabahat. “Predicting MBTI personality type with K-means clustering and gradient boosting.” In 2020 IEEE 23rd International Multitopic Conference (INMIC), pp. 1-5. IEEE, 2020.

Gantt Chart

GanttChart.xlsx

GANTT CHART

PROJECT TITLE	MBTI personality prediction

					PHASE ONE									
					Sep 23									
TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	M	T	W	R	F	S	U	M	T	W
Project Proposal														
Dataset Selection	All	20/9/2024	24/9/2024	5										
Introduction & Background	Jiatong Ren	25/9/2024	1/10/2024	6										
Problem Definition	Wenwen Li	25/9/2024	1/10/2024	6										
Methods	Jincheng Song	25/9/2024	1/10/2024	6										
Potential Results & Discussion	Jinghao Jiang	25/9/2024	1/10/2024	6										
Video Recording	All	1/10/2024	4/10/2024	4										
GitHub Page	All	1/10/2024	4/10/2024	4										
Midterm Report-Random Forest														
Data Sourcing and Cleaning	Jiatong Ren	4/10/2024	10/10/2024	6										
Model Selection	Jincheng Song	10/10/2024	17/10/2024	7										
Data Pre-Processing	Jinghao Jiang	17/10/2024	24/10/2024	7										
Model Coding	Hanxuan Zhang	24/10/2024	31/10/2024	7										
Results Evaluation and Analysis	Wenwen Li	31/10/2024	7/11/2024	7										
Midterm Report	All	7/11/2024	11/11/2024	4										
Midterm Report-Logistic Regresion														
Data Sourcing and Cleaning	Jiatong Ren	4/10/2024	10/10/2024	6										
Model Selection	Jincheng Song	10/10/2024	17/10/2024	7										

Fall Fall Overview

Contribution Table

Contribution Table

	Name
Hanxuan Zhang	Worked collectively to design and prepare the presentation, ensuring it was clear, concise, and visually engaging. Collaboratively participated in recording the presentation, ensuring that all parts were seamlessly integrated and professionally delivered.
Jiatong Ren	Led the implementation of the Gaussian Mixture Model (GMM), ensuring correctness and efficiency. Participated in discussions about model structure and helped refine the overall coding approach.
Jinghao Jiang	Focused on comparing the performance of various models, highlighting their strengths and weaknesses. Ensured that the analysis was well-documented and supported by clear visualizations.
Jincheng Song	Headed the evaluation process of all model results, conducting in-depth analysis and ensuring statistical accuracy. Collaborated closely with team members to interpret and document the findings.
Wenwen Li	Took the lead in developing and coding the Recurrent Neural Network (RNN), ensuring compatibility with t project framework. Contributed by debugging and optimizing the model for better performance.

Sheet1 Sheet2