Myers-Briggs Personality Type Prediction with Text Classification

Raehyun Kim, Tianhao Wu & Shiping Xu [*]
University of California, Berkeley
Berkeley, CA 94720, USA
`{rhkim79,thwu,sxu2}@berkeley.edu`

Abstract

Personality prediction has wide applications in various fields such as recruitment, recommendation systems, etc. However, traditional methods of assessments for personality prediction are often inefficient and time-consuming. The goal of our project is to using machine learning to build a classifier to automatically predict people's Myers-Briggs personality type based on texts from their social media posts. Our final implementation with Time Frequency-Inverse Document Frequency (TF-IDF) features, and fine-tuned classical classifier achieves a best test accuracy of 67%.

1 Introduction

Personality is regarded as an influential construct and is widely researched by many psychologists because it is predictive of many consequential outcomes. For example, more and more companies nowadays utilize personality types to help them streamline recruitments and find candidates to build effective teams (De Jong et al., 2019). Unfortunately, traditional methods of personality assessments are often time-consuming and inefficient. To address this problem, our project aims to use machine learning and natural language processing techniques to build a classifier that automatically predicts people's personality types based on texts from their posts on social media. We hypothesize that a person's writing style from social media posts is largely related to their personality traits. We believe that by developing such a text-based classifier, more people could gain understanding of their personality types in a faster and more reliable way.

2 Background & Related Work

Out of all personality models developed by psychologists, the Myers-Briggs Type Indicator (MBTI) model is the most widely used in a variety of disciplines. The model uses four dimensions to divide people into 16 distinct personality types. The four dimensions are: Extraversion (E) vs Introversion (I), Sensing (S) vs Intuition (N), Thinking (T) vs Feeling (F), and Judging (J) vs Perceiving (P). The final result is a four letter representation from each of the four dimensions. In this study, we choose to use the MBTI personality model because it has more applications in a variety of fields.

Many recent works have developed successful methodologies to predict MBTI types. Komisin & Guinn used classical machine learning techniques, including Naive Bayes and SVM, on word choice features using the bag-of-words model. Their model was able to predict MBTI types pretty accurately with precision and recall around 70% (Komisin & Guinn, 2012). Alternative work done by Li et. al has successes in predicting partial (1-letter) MBTI typing with high accuracy using novel feature extraction methods(Li et al., 2018). Our work builds on the foundational studies. We explore a variety

[*]The authors are arranged in lexicographical order.

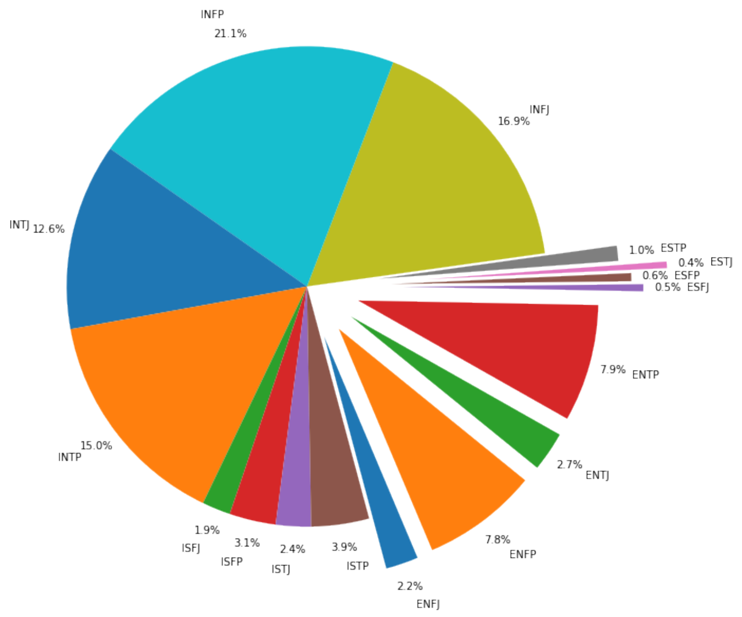

Figure 1: Percentage of occurrence for each MBTI personality type in the dataset.

of classical machine learning techniques, as well as boosting and neural networks on key features extracted using different feature engineering methods to address personality problems.

3 Data

3.1 Dataset

Our work uses a dataset provided by Kaggle (Lao et al. 2017). The dataset was crawled from Personality Cafe forum in 2017, consisting of 8,675 rows and 2 columns, type and posts. The "posts" column contains the last 50 posts made by each user and the "type" column contains their MBTI type. There are no NULL values in this dataset.

Figure 1 shows the percentage of occurrences for each MBTI personality type in our dataset. It is clear that our raw dataset is highly disproportional and is not commensurate with the roughly uniform distribution in the general population. Therefore, some cleaning would be necessary in order to improve the accuracy of representation of each MBTI type.

We split the raw dataset into training and testing datasets in the ratio of 80:20. Since the dataset is unbalanced, we use Stratified K-Fold Cross-Validation to split the dataset in a more stratified manner. After the split, the training set contains 6,940 users and the testing set contains 1,735 users.

In this study, we follow the steps in training pipeline in Appendix A and compare their performances.

3.2 Preprocessing

For better modeling results, we apply several pre-processing steps on the posts data. We first convert all posts into lowercase, and remove all URLs in posts because they do not provide any useful information about a person's personality. We then convert all emoticons into English phrases using an Emot library (Shah & Rohilla, 2021) on Python, since they might carry some valuable information. Next, we remove all punctuations, special characters, most and least frequent words and stop words from the texts so that they only contain meaningful words. Stopwords include common filler words such as 'a', 'you', 'for', etc. Since they are not useful in predicting personality types, we remove them using Python's NLTK library. Lastly, we use `nltk.stem.WordNetLemmatizer` to lemmatize the posts. This means inflected forms of the same root word are grouped together so they could be analyzed as a single feature.

3.3 Feature Engineering

To extract key features from the social media posts, we convert the clean posts to a matrix of term frequency-inverse document frequency (TF-IDF) features using `TfidfVectorizer` from the `scikit-learn` library (Pedregosa et al., 2011). TF-IDF measures how important a word is to a document. Term frequency evaluates how frequently a term occurs in a document, while inverse document frequency measures how much information the word provides. We pick the top 2,000 words ordered by term frequency across the whole clean posts and assign the TF-IDF value for each user's clean post. As a result, we obtain a design matrix $X_{tf} \in \mathbb{R}^{\#\text{users} \times 2000}$. Note that we need to generate the vectorizer V from the training dataset and apply V to the training and test dataset. Besides the TF-IDF value, we include three additional features which count the number of words, question marks and URLs in the raw posts.

Considering that the TF-IDF design matrix is sparse while containing a large number of features, we give an option for the linear dimensionality reduction using singular value decomposition. In addition, we include the whitening option for the TF-IDF features and the three additional features. Since we can determine the effect of our feature engineering schemes only in an empirically way, we build 10 baseline training datasets with different options. The detailed description on the datasets is provided at Appendix B.

We also explore several embedding techniques to extract features. For word embedding, we embed the posts and obtain 50-dimensional GloVe vector representations using pre-trained word vectors on Wikipedia 2014 + Gigaword 5 corpus (Pennington et al., 2014). For sentence embedding, we use pre-trained Universal Sentence Encoder (USE) based on Deep Averaging Network (DAN) encoder model to encode each user's posts into a 512 dimensional vector (Abadi et al., 2015).

4 Models

4.1 Classifiers

In this project, we build, train and test the following models:

1. Linear Discriminant Analysis(LDA)
2. Quadratic Discriminant Analysis(QDA)
3. Logistic regression Classifier
4. Support Vector Classification(SVC)
5. Decision Tree Classifier
6. Random Forest Classifier
7. Multi-Layer Perceptron (MLP)
8. Gradient Boosting Classifier from `XGBoost` (Chen & Guestrin, 2016)

For each classifier, we use the `scikit-learn` library except the Gradient Boosting Classifier.

4.2 Evaluation

For each classifier, we first train the model on 10 baseline datasets and determine the dataset with the best potential. Since the classifiers work better on 10 baseline datasets than the design matrices generated by embedding methods, we restrict hyper-paramter tuning on the baseline datasets. Considering that the embedding methods accept the all clean post as one sentence/paragraph, the lack of unity between posts would have had an adverse effect.

After the initial training step, we carry out a further hyper-parameter tuning on the best dataset accordingly. For example, we optimize the XGBoost classifier as follows. Figure 4.2 shows the initial training result of the XGBoost classifier on the 10 baseline datasets. The XGBoost classifier shows the identical result on the dataset 0, 1, 2. For the next step, we conduct further hyperparameter tuning on the dataset 1 which follows the general approach on whitening. Since the training result indicates the overfitting, we form the hyperparameter grid with smaller value of max_depth with larger value of n_estimator. The hyperparameter tuning result is shown at Figure 4.2.

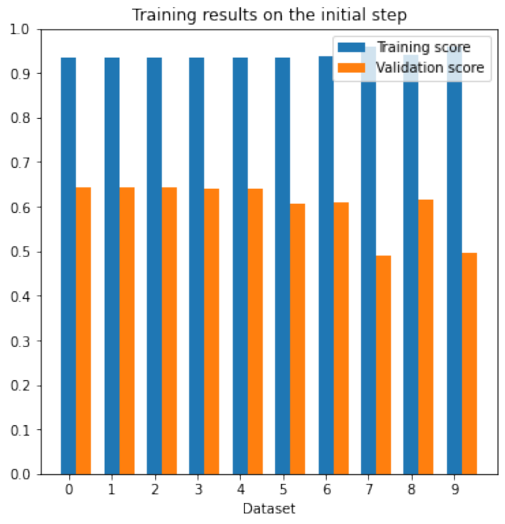

Figure 2: Training results on the initial step

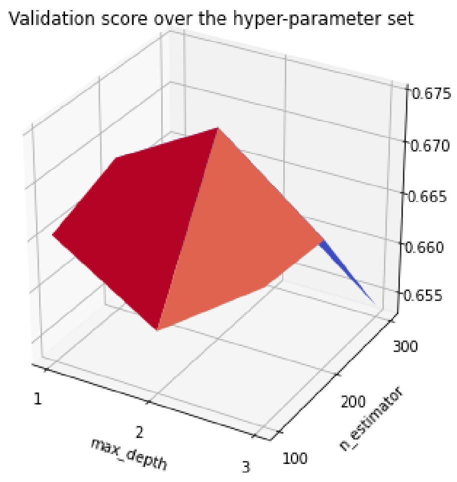

Figure 3: Validation score over the hyper-parameter set

Table 1 presents the best model settings. If '#PC' is recorded, respective principal component are chosen to perform dimensionality reduction on the selected baseline dataset, B. Cross-validation of 5 folds (cv=5) and test accuracy of each classifier with the best hyperparameter setting are also recorded, visualized in Figure 4.
Figure 6 and Figure 7 are the visualizations of scoring metrics (accuracy, precision, recall, f1) of the best model on training and test dataset respectively.

Model	**Baseline Dataset**	**#PC**	**Hyperparameter**	**Cross Validation Accuracy**	**Test Accuracy**
Logistic Regression	3	300	C=0.8; penalty='l1', solver='liblinear'	0.678	0.650
SVM	3	200	C=0.14; kernel='rbf'	0.545	0.577
QDA	3	500	N/A	0.250	0.304
LDA	1	38	N/A	0.680	0.672
Decision Tree	3	500	max_depth=10	0.420	0.430
Random Forest	3	500	max_depth=20; n_estimators=150; min_sample_leafs=2	0.545	0.541
XGBoost	1	N/A	max_depth=2; n_estimators=200	0.684	0.666
MLP	3	500	alpha=0.05; hidden_layer_sizes=(20,)	0.647	0.651

Table 1: Best Models Parameters

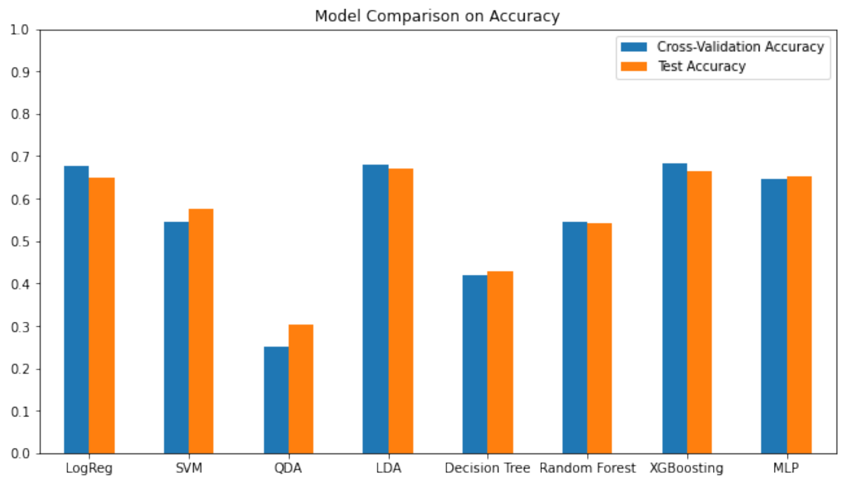

Figure 4: Model Comparison on Training and Test Accuracy

5 Analysis

From Figure 4, we can tell that models such as Logistic regression classifier, LDA, Gradient Boosting classifier, and MLP classifier are performing well, achieving over 0.6 both training and test accuracy. Worse-performing tree-based models, such as Decision Tree classifier and Random Forest classifier, show a lower training and test accuracy. This phenomenon may be caused by the deep max depth of trees for the best models, which would overfit the training data. The worst performing model, QDA, is too flexible that fits the noises in the training data, thus overfitting and resulting in unsatisfactory cross-validation and test accuracy.

Considering that our dataset consists of a large number of features with a small number of samples, it highly possibly overfits the training dataset. Therefore, simple models such as Logistic regression, LDA outperforms other classifiers as reported in the previous work (Gjurkovic & Snajder, 2018). Also, Gradient Boosting classifier with short trees could efficiently deal the overfitting problem because it works as a form of the feature reduction. Interestingly, the MLP does not outperform the Logistics regression in the test accuracy which means that the size of the current training dataset is not sufficient enough for the MLP to achieve a higher performance on classification.

Though the best model for each classifier are found on various baseline training datasets, a commonality is that they all use un-whitened TF-IDF features. It could be partially explained by the fact that the TF-IDF features shares the same unit while the additional features do not. Furthermore, through empirical search, we found that Logistic regression and MLP perform better using additional features whitened, whereas LDA and Gradient Boosting perform better using additional features un-whitened. It shows that the effect of the modification on the design matrix(e.g. whitening, dimensionality reduction) can be determined only by the empirical way.

As a general guidance to dimensionality reduction, when the model overfits the full features data, which may be observed through high variance during cross-validating, we should perform dimensionality reduction to reduce the effect of overfitting on the noise. Well-performing models such as the Logistic regression classifier, LDA, and MLP classifier require dimensionality reduction through principal components analysis. On the other hand, if the model underfits with the original data, dimensionality reduction shall not be performed as it would decrease the complexity of the already simple model. For future work, we suggest to reduce the features that make the model overfits through backward elimination or forward selection.

6 Code

We have made the source code of our implementation available at
`https://github.com/xushiping713/CS289-Myers-Briggs-Prediction`

7 Conclusion

Our classifiers could predict the Myers-Briggs Personality Type well with the best test accuracy of 0.67. It shows that the personality type prediction is feasible using Linear Discriminant Analysis(LDA) classifier with 2000 term frequency-inverse document frequency(TF-IDF) unwhitened features combining 3 additional whitened features. Besides LDA, the Gradient Boosting, Logistic Regression classifiers show relatively good performance, as they keep the models in reasonable complexity and not overfitting on the training dataset. We hypothesize tree-based classifier with large max depth and flexible Quadratic Discriminant Analysis(QDA) do not work well because they overfits too much on training set, and unable to predict from unseen test dataset.

For future work, we recommend explore more on feature selection, extract more meaningful additional features, eliminate features causing overfitting through backward elimination or forward selection, along with PCA. Considering that we could observe the underfitting even with the best classifier, some effective feature addition would be critical to achieve a better result. We also recommend exploring the effect of separating currently concatenated posts and considering them individually with respective MBTI type.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

De Jong, N., Wisse, B., Heesink, J. A., and Van der Zee, K. I. Personality traits and career role enactment: Career role preferences as a mediator. *Frontiers in psychology*, pp. 1720, 2019.

Gjurkovic, M. and Snajder, J. Reddit: A gold mine for personality prediction. In *PEOPLES@ NAACL-HTL*, pp. 87–97, 2018.

Komisin, M. and Guinn, C. Identifying personality types using document classification methods. *Proceedings of the 25th International Florida Artificial Intelligence Research Society Conference, FLAIRS-25*, pp. 232–237, 01 2012.

Li, C., Hancock, M., Bowles, B., Hancock, O., Perg, L., Brown, P., Burrell, A., Frank, G., Stiers, F., Marshall, S., Mercado, G., Ahmed, A.-W., Beckelheimer, P., Williamson, S., and Wade, R. Feature extraction from social media posts for psychometric typing of participants. In Schmorrow, D. D. and Fidopiastis, C. M. (eds.), *Augmented Cognition: Intelligent Technologies*, pp. 267–286, Cham, 2018. Springer International Publishing. ISBN 978-3-319-91470-1.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Shah, N. and Rohilla, S. emot:3.1 library, 8 2021. URL `https://github.com/NeelShah18/emot`.

A Training Pipeline

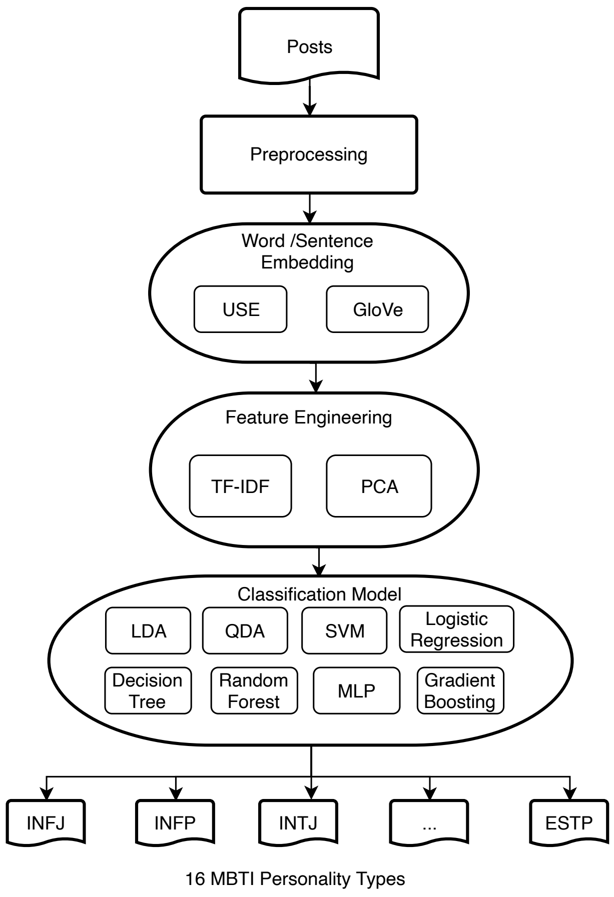

Figure 5: Training Pipeline.

B Baseline Training Datasets

Dataset	Additional Features	Whitening		Dimensionality Reduction	Dimension
		TF-IDF	Additional		
0	True	False	False	False	6940×2003
1	True	False	True	False	6940×2003
2	True	True	True	False	6940×2003
3	False	False	N/A	False	6940×2000
4	False	True	N/A	False	6940×2000
5	True	False	False	True	6940×500
6	True	False	True	True	6940×500
7	True	True	True	True	6940×500
8	False	False	N/A	True	6940×500
9	False	True	N/A	True	6940×500

C Best Model Comparison

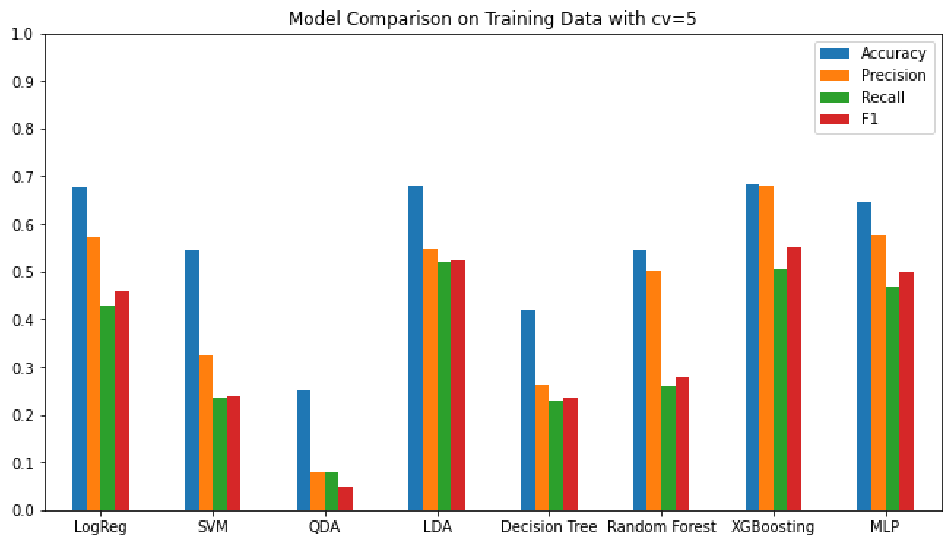

Figure 6: Model Comparison on Training Datasets

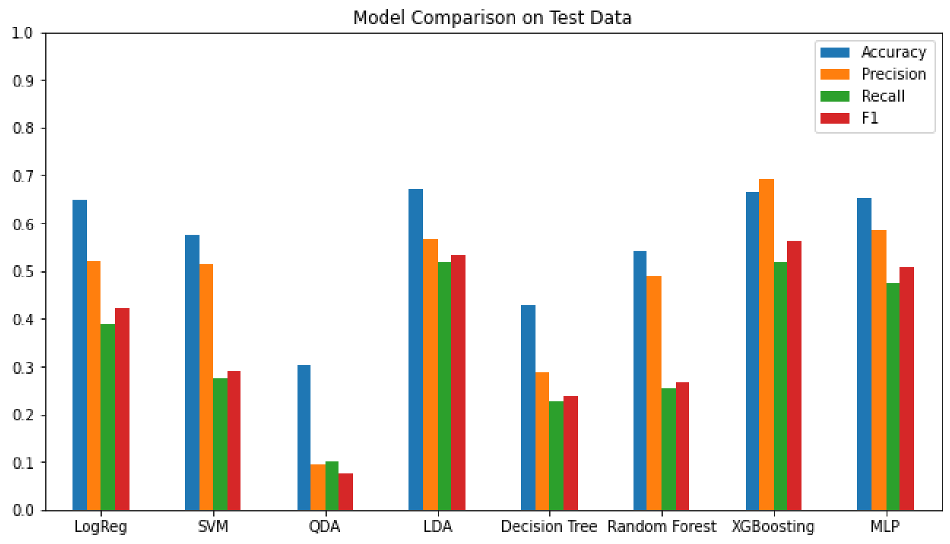

Figure 7: Model Comparison on Test Datasets