Myers Briggs personality predicting

Matthew Rogers [*]
School of Computing
Clemson University
`mwr2@clemson.edu`

Yue Zhang
School of Computing
Clemson University
`yzhng@clemson.edu`

Abstract

The Myers Briggs personality assessment puts a personality type to an individual. There are four components to an individual's personality. For each component, an individual takes on one out of two attributes. Since each of the four components has two attributes, then there are sixteen different personality types that an individual can be labeled as. For the first component there is introversion vs. extroversion, for the second there is intuition vs. sensing, for the third thinking vs feeling, and for the last component there is judgment vs perception. The objective of this project is to investigate the effectiveness of the perceptron to label each component of an individual's personality type.

1 Introduction

The purpose of this project was to investigate the efficiency of different classification models to predict the personality type of an individual based on twitter posts. The motivation of exploring this subject is this research can help companies to provide personalized services and recommendations based on one's personality preferences and needs and this research can also help social media sites enhance a user's presence and reputation by choosing the most suitable posts and interactions. The dataset used in the research included 6940 anonymous individuals, each labeled with a Myers Briggs personality type. Each individual had a series of twitter posts associated with them. To help in the classification of the individual's personality linear, polynomial, sigmoid, and gaussian radial basis SVM, and perceptron were used. The Myers-Briggs personality type is a way to label individual with a certain personality. This personality type is made up of 4 personality trait pairs. The first is introversion vs extroversion. This pair is the measure of an individual's preference for the inner or outer world. The second pair is intuition vs sensing are the information gathering functions. They describe how new information is gathered and interpreted. People who prefer sensing are more likely to trust information in the present, tangible, and concrete. People who prefer intuition tend to trust information that can be associated with other information. The third pair is thinking vs feeling. These are the decision-making functions. Those who prefer thinking tend to decide things from a more detached standpoint, measuring the decision what seems logical, reasonable, causal, consistent, and matching a given set of rules. Those who prefer feeling tend to come to decisions based on associating or empathizing with the situation. The fourth pair is judging vs perceiving. Judging refers to whether an individual displays their thinking or feeling function whereas perceiving refers to whether an individual displays their intuition or sensing function.

2 Background

Due to the nature of the Myers-Briggs type , we can break down the classification task with 16 classes in to 4 binary classification tasks. This is because a MBTI type is composed of 4 binary classes,

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

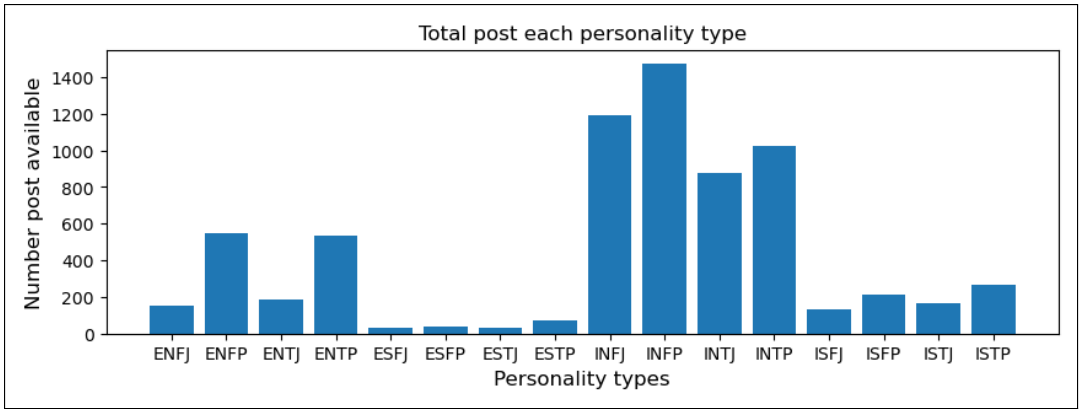

Figure 1: Distribution of MBTI types in the data set.

where each binary class represents a dimension of personality of the MBTI personality model as theorized by the inventors. Therefore, instead of training a multi-class classifier, we instead train 4 different binary classifiers, such that each specializes in one of the dimensions of personality.

Once these new datasets were produced, each dataset (each one was a csv file) was transformed into a table in MATLAB. The rows were sorted on the attribute column so that one type would appear first before the other one. For example, if the dataset with the first attribute was being used, then each individual labeled as an introvert would appear first and then the individuals labeled as extroverts would appear second. Then, a certain number of rows for each type were taken and were combined to form a training set. The other rows remaining were combined to form the test set. After each set was randomly permutated and standardized (the zscore function was used), training set was put through a linear perceptron algorithm to find the weights. Using these weights, the test set was put through another algorithm to test the accuracy of this perceptron.

3 Methods

3.1 Preprocessing

When we examined other studies of MBTI using machine learning, we were surprised to find that researchers rarely made a point of cleaning their data set to accord with the actual proportions of MBTI types in the general population (e.g. ISTJ = 0.11, ISFJ = 0.09, etc.)[3]. Since our raw data set is severely disproportional (see fig. 1) compared to the roughly uniform distribution for the general population, it was clear to us that some cleaning of the proportional representation of each MBTI type would be necessary. Therefore, we artificially made our test set reflect the proportions found for each type in the general population, so as to prevent any misinterpretation of results due to skewed representation of classes in the test set.

3.1.1 Meaningless words removal

As the dataset is sourced from an online forum where communication is solely through written text, certain modifications were necessary. Notably, instances of data points containing hyperlinks were eliminated, aiming to foster the model's generalization to the English language. Additionally, to enhance the meaningfulness of each word in the dataset, so-called "stop words," such as common fillers like "a," "the," "or," etc., were excluded using the NLTK library in Python. Lastly, considering the dataset's origin from a website dedicated to explicit discussions about personality models, particularly MBTI, references to specific types (e.g., 'INTJ,' 'INFP,' etc.) were expunged. This measure was taken to prevent the model from potentially "cheating" by learning to recognize MBTI mentions explicitly by name.

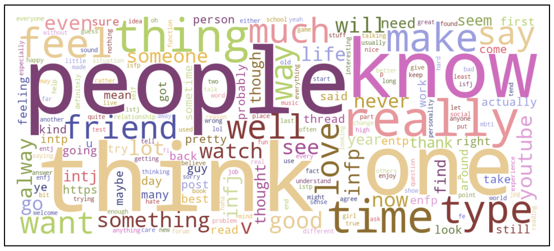

Figure 2: Before Preprocessing.

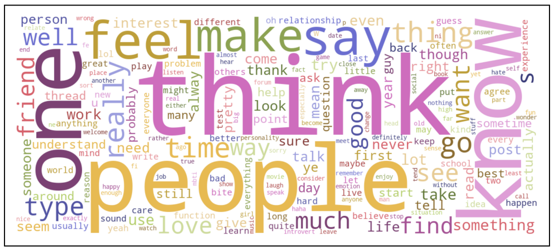

Figure 3: After Preprocessing.

3.1.2 Transfer to root words

To transfer to root words on the text, we employed the nltk.stem.WordNetLemmatizer. This process involved transforming inflected forms of the same root word into their dictionary form; for instance, "walking," "walked," and "walk" were all lemmatized to "walk." This approach enables us to capitalize on the shared meaning inherent in various inflected forms of a given word.

3.1.3 Input format

First, we picked up 500 most commom words from the proprocessed dataset,
('think', 39658),('get', 28446),('people', 27788),('know', 24659),('one', 21477),('feel', 21454),.....

Then we build a vector of 500 elements to represent each post which each element is the frequency of these 500 words in the post.

$$[f_0, f_1, ..., f_{499}]^T \tag{1}$$

3.2 Dimesion removal

To prevent overfitting and enhance efficiency, we have decided to attempt using PCA for data dimensionality reduction. To determine the appropriate number of dimensions, we created a figure in which The x-axis represents the number of selected dimensions, and the y-axis represents the percentage of information represented after selecting the current number of dimensions. from the

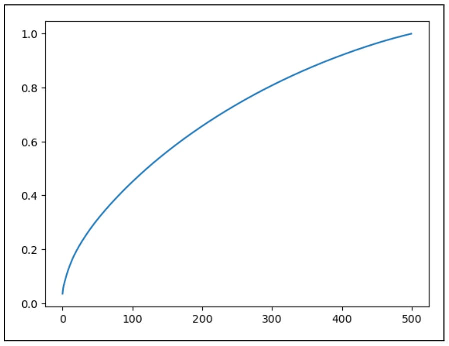

Figure 4: dimension vs percentage of info..

Table 1: All attributes

Kernel	Train Accuracy	Time consumption	Test Accuracy
RBF	0.734047	30.857661	0.317003
Poly	0.987855	35.146751	0.316042
Linear	0.475504	25.300210	0.310279
Sigmoif	0.365377	25.245560	0.306916

Table 2: IE

Kernel	Train Accuracy	Time consumption	Test Accuracy
RBF	0.838617	24.255259	0.770893
Poly	0.991149	26.795339	0.769933
Linear	0.770893	16.092506	0.770413
Sigmoif	0.757102	7.945040	0.770413

figure, we couldn't find a turning point where there is sufficient information at the point. Beyond this point, there is no significant increase in the amount of information In summary, we are unable to perform dimensionality reduction on the data.

3.3 SVM

The final dual formulation of kernel svm can be represented by

$$\max_{\lambda} -\frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j K(x_i, x_j) \tag{2}$$

We tried 4 different kernels in our project, they are

- linear: $K(x, y) = x^T y$
- poly: $K(x, y) = (\gamma(x^T y) + r)^d$
- sigmoid: $K(x, y) = tanh((x^T y) + r)$
- $K(x, y) = e^{-\gamma||x-y||^2}, \gamma > 0$

Also, we test different values for the penalty term C in the original problem

$$\min_{\omega,b,\epsilon} \frac{1}{2}\omega^T\omega + C\sum_i^n \epsilon_i \tag{3}$$

4 Resuts

4.1 All attributes

see Table [1]

4.2 Extraversion vs. Introversion

see Table [2]

4.3 Sensing vs. Intuition

see Table [3]

4.4 Thinking vs. Feeling

see Table [4]

Table 3: NS

Kernel	Train Accuracy	Time consumption	Test Accuracy
RBF	0.870111	21.805165	0.770413
Poly	0.998353	23.301729	0.768012
Linear	0.862907	13.397481	0.770413
Sigmoif	0.862495	6.643756	0.770413

Table 4: TF

Kernel	Train Accuracy	Time consumption	Test Accuracy
RBF	0.931247	23.654586	0.710855
Poly	0.994648	28.503457	0.695485
Linear	0.782627	13.486006	0.709414
Sigmoif	0.695760	16.708707	0.695965

4.5 Judging vs. Perceiving

see Table [5]

4.6 different penalty term on linear kernel

we also tried different C from 0 to 1 to improve the accuracy, see figure 5

4.7 Conclusion

SVMs are not performing very well on this job

5 Based on letters

As mentioned in the proposal, for each row (anonymous individual) there was a personality type made up of four attributes and a certain number of columns with each column containing a whole or part of a twitter post. The first objective was to combine all these columns into one so that there was just one column containing all the posts for each individual. The second objective was to split up the personality type into four separate columns where each column represented an attribute. For example, if an individual was labeled as IFSP, then this column would be split into four attributes where the first attribute would be labeled as I, the second attribute labeled as F, etc. These transformations were done in Excel.
For the first part of our research project, it wanted to be determined if there was an association between the average of different letters and the individual's specific attribute type. To see if there was an association, the frequency of each letter was found for each row (individual). Since, each individual varied on the number of letters used for all of their posts, the average was then calculated for each letter frequency. For example, if an individual used a total of 20000 characters and used the letter 'a' in 1000 instances, then there would be a score of .05 (instances/ total # of characters). After, the average was calculated for each letter for each individual, then this new dataset was split into four datasets where each dataset concentrated on a specific attribute (I vs E, F vs T, etc.). This part of the project was done in Python.

Table 5: JP

Kernel	Train Accuracy	Time consumption	Test Accuracy
RBF	0.898312	25.711215	0.645053
Poly	0.994236	31.332261	0.645053
Linear	0.703993	14.384366	0.643132
Sigmoif	0.611980	15.768533	0.640250

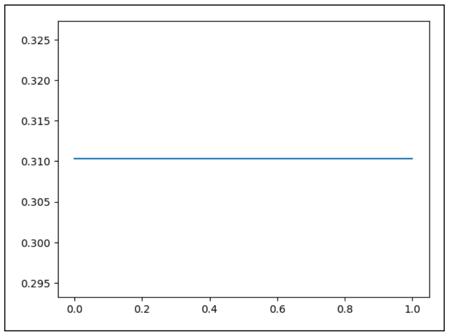

Figure 5: C value vs. Accuracy

After using this process for each dataset and varying the number of entries in the training set and varying the number of iterations the perceptron took to find the weights, it was observed the accuracy varied significantly. Accuracy scores approximately ranged from 35% to 70%. It was questioned whether perhaps the specific sample used could explain these observations. In other words, maybe choosing an appropriate training set could make the perceptron more accurate by being more representative of the rest of the dataset.
A clustering approach in Python was used to investigate this question. Specifically, Kmeans was used from the Sci-kit Learn package. The intent of the overall process about to be outlined was to cluster the whole population (all the entries in a particular dataset) into groups based on a mean value. Then, a representative would be chosen from each group to be used for the training set. Perhaps, this representative training sample could lead to a higher accuracy score using the linear perceptron.
The specific process started with using just one dataset with the second attribute (F vs T). The columns that were not going to be used in the clustering algorithm were dropped. The data frame was then grouped by the binary classification (1 or -1) and then was split into two separate data frames based on this classification.
Kmeans was then used for each dataset with a certain number of means (but the same) to cluster both data frames. After adding a cluster group column in each data frame, each row (in each data frame) was labeled with its corresponding group number. Each data frame, was then sorted in ascending order based on its cluster group number. A representative was then chosen from each group from each data frame. This representative was chosen by being first the first entry for each group. All the representatives were then put into a training set and the rest of the entries were put in a test set. Each set was then randomly permutated and the cluster group column was dropped. The training and test set were then converted to an array so that it was an acceptable form to be used by the linear perceptron model from the Sci-kit Learn package. The training set was used to train this model and then the test set was used to measure its accuracy.

5.1 letters result

See figure6
The accuracy was measured using a different number of training ‘representatives’. This number was only changed a few times (so there were only a few accuracy scores) but the accuracy did not get above 55%. This process was rudimentary since it did not find the ‘best’ representative of the group. In other words, it did not find the group member with the lowest average mean. At this time, it is also uncertain how 70% was even achieved in the initial perceptron process (MATLAB). These questions still need to be investigated. However, it was concluded that looking at frequency averages of letters did not provide a significant way of labeling each attribute. Therefore, the next portion of the project was dedicated to investigating the words used instead of the frequency averages of the letters.

5.2 Observation & conclusion

The accuracy is misleading:

- Consider the trait pair with highest accuracy (N/S)
- There were 5988 N’s and 952 S’s

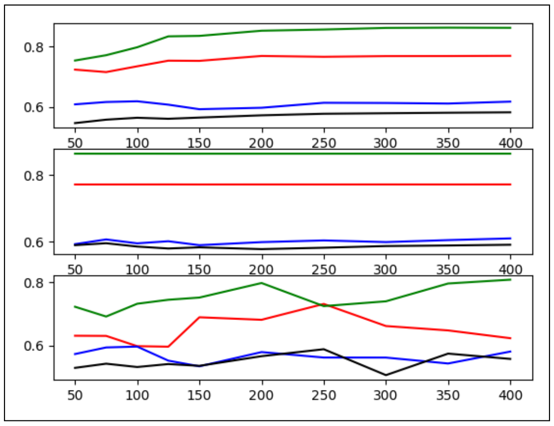

Figure 6: for each attribute

- Algorithm could have guessed N for all samples
- Accuracy $= 5988/6940 = 86.2\%$

Similarly, if algorithm guessed 'dominant' trait for other sets:

- $IE \rightarrow 77.1\%$
- $TF \rightarrow 54.5\%$
- $PJ \rightarrow 60.2\%$

Therefore, it was concluded that the kernel svm model and the other classification models were NOT accurate in predicting personality traits based on letter frequency average