Predicting Personality Type

Kathy Do
kdo@uvic.ca

Steven Liu
qingfengliu@uvic.ca

Jian Wu
wujian@uvic.ca

Nikita Shymberg
shymberg@uvic.ca

***Abstract*—We used data mining techniques to create a system able to predict one's MBTI personality based on a sample of text the person has written. We obtained user data consisting of the MBTI personality type label for a user and a sample of text the user has written from a personality forum. We performed text analysis on the users written texts and applied different data mining techniques including support vector machines, neural networks, and logistic regression to predict the MBTI personality type of the author of a written piece of text. We then reported and discussed our findings.**

This report was written as part of the requirements for SENG 474 / CSC 578D: Data Mining, offered at the University of Victoria, taught by Dr. Alona Fyshe.

***Index Terms*—data mining, personality, Myers-Briggs, text analysis, natural language processing**

I. Introduction

The human personality is a vast and complex field of study that we do not yet fully understand. There have been various approaches to classifying one's personality that have existed in the field of psychology over the years. One of these is the popular Myers Briggs Personality Type Indicator (MBTI) classification system that standardizes ones personality type across 4 axes: Introversion (I) vs. Extroversion (E), Intuition (N) vs. Sensing (S), Thinking (T) vs. Feeling (F), and Judging (J) vs. Perceiving (P). The system, developed by Isabel Briggs Myers and Katharine Briggs, aims to make the psychological types described by the psychologist Carl Jung easily accessible to people for them to understand themselves and their relationships with others [1]. Although the MBTI has been criticized of its validity and reliability in recent years, the system is still used in many areas making it a relevant problem to explore [2]. One of the aims of this project is to see whether we can detect any correlation in regards to writing style and personality type, being able to do this successfully will further validate the use of the MBTI test in categorizing and analyzing ones personality.

In our project, we used data mining techniques to create a tool to predict a person's MBTI personality based on some sample text they have written. Our models will be using user data from a popular personality forum called Personality Cafe where users post several text comments on various different discussion topics. We will analyze a person's past posts and run it through our algorithms to predict their MBTI personality type code. One application for a personality predictor tool like this is that it could be used by marketers and businesses to understand their customer base better. Companies could target specific advertisements and make appropriate recommendations based on personality.

II. Related Works

Personality analysis is commonly based on the Big Five model [3] or the Myers-Briggs Type Indicator (MBTI) theory [4]–[6]. To be specific, we only focus on the related works based on the MBTI theory in this overview.

Halawa et al. [4] predicted students' personality types based on data from the student's engagement with a learning management system, Moodle, and a social network, Facebook. They investigated 10 different classifiers, such as Naive Bayes, Bayes Network, K*, Random forest, J48, OneR, and etc. They found that the OneR classifier had the highest accuracy, followed by the Random forest and J48 classifiers.

Kim et al. [5] used the TiMBL classifier, a memory-based learning algorithm based on the k-nn algorithm to predict an author's personality. They first extracted the reliable syntactic features from the input text using the Memory-Based Shallow Parser (MBSP). Then they calculated the expected and observed frequency for each item in each category. Then they performed ten-fold cross-validation experiments using TiMBL. They concluded that the first two personality dimensions (Introverted-Extroverted and Intuitive-Sensing) can be predicted with fairly high accuracy.

Ma et al. [6] used the recurrent neural network (RNN) with a long short-term memory (LSTM) model to predict a writer's personality type. They claimed that they can predict the writer's personality only using five sentences from the writer's novels. However, the reported accuracy was only 37%, which was relatively low compared with Halawa's work.

III. Data Description

The dataset that was used for this project is publicly available and can be found on Kaggle [7]. The *Myers-Briggs Personality Type Dataset* on Kaggle contains 8676 rows of data, where each row represents a user. The data was collected from a website called Personality Cafe which hosts a forum where people, with classified MBTI personality codes, discuss various topics and share opinions with other users [8]. There exists two attributes in the dataset: *type* and *posts*. The *type* attribute is the four letter MBTI personality code for the user and is of data type *string*. The *posts* attribute is also of type `string` and is a concatenation of the last fifty posts the user has published, each separated by ||| (3 pipe characters). For our predictions, user text posts could be classified into 16 different personalities based on the MBTI personality code, which are all possible combinations of the binary classes across the four different personality dimensions. However, the distribution of users across the 16 personality types were found to not be

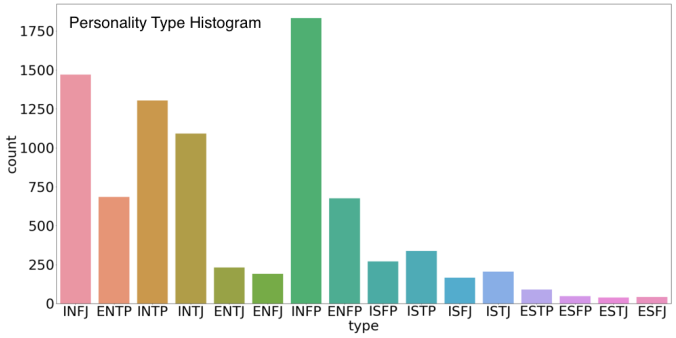

Fig. 1. Distribution of the 16 MBTI class types in the dataset

distributed very evenly. For example, as seen in Figure 1, there was very little data falling into categories ESFJ and ESTJ.

When looking at each of the four personality dimensions separately, there is a better ratio of data within classes. In our dataset, the classification of users in our dataset by each personality dimension were as follows:

- Introversion: 76.95%, Extroversion: 23.04%
- Intuition: 86.20%, Sensing: 13.80%
- Thinking: 45.89%, Feeling: 54.11%
- Judging: 39.58%, Perceiving: 60.41%

Although the ratio within classes is still not evenly distributed, there is more data for each classification problem compared to the 16 way classification approach.

IV. Project Approach and Implementation

The models that we explored and implemented for this project were logistic regression, support vector machines, and neural networks. We chose these supervised learning models as the data set was already labeled and because we wanted to explore these methods we learned in class more in depth. In order to evaluate each model, the data was split into two stratified sets: 80% as the training set, 20% as the testing set. We used 8 fold cross validation on the training set to test the models and decide if any parameters need to be altered. Stratification was especially important in this project since the classes were heavily unbalanced in our data set. The final accuracies of the trained models were based on the test data and are reported in our final results.

Because of the uneven data across the 16 personality types, we decided to incorporate four binary classifications, one for each personality dimension, as an intermediate step to predict the final MBTI personality type. The approach for each algorithm we implemented can be broken down into two high level steps. The first step was to implement a model to perform a binary classification for each dimension in the MBTI Personality Type. The second step was to take the output probabilities of the 4 binary classifications to input into a 16 way ensemble classifier to predict the final MBTI personality type.

A. Data Cleaning and Processing

The MBTI Personality Type Dataset was a .csv file downloaded from Kaggle which contains the MBTI personality type code and text the user has written. The Pandas library was then used to import the .csv file into a Pandas dataframe for easy processing and manipulation. Before the data could be used by our algorithms, the text posts had to be cleaned to remove irrelevant features (words) and to transform the data into a useful format to pass to our algorithm implementations. During the initial inspection of the data, we realized that there would be several types of features that needed to be ignored when vectorizing the user text posts. These irrelevant feature types included URL links that the user posted because, although these could be considered as further data that could be used to predict personality, it is not considered text written by the user, and therefore was omitted for this project. In addition to URLs, we omitted punctuation, standalone numbers, and stop words which are common words that are not useful for defining context, (eg. like, and, or, but, etc). Once the data was cleaned, the next step was to create new target vectors for the MBTI personality types. Our algorithm designs involved classifying each dimension separately (ie, I vs. E, N vs. S, T vs. F, J vs.P), and then combining the results to construct the MBTI code. Therefore, four new target vectors were created for each personality dimension.

One concern we had with our data was that because the text posts were sourced from a casual, online forum, there were instances of misspelled words that may affect the accuracy of our predictions. The implication of misspelled words is that if misspelled variants of a word have been detected in our dataset, these would be treated as separate features in our feature vector, meaning the count for a specific word may not be counted accurately. We decided to handle this issue by ignoring these cases by removing very infrequent words from our vocabulary. This was done by setting the `min_df` parameter to 2 when using CountVectorizer in python to convert each text post into a vector. This parameter setting meant that any words that appeared in only one document would be ignored.

B. Logistic Regression Design

Logistic Regression is a very suitable algorithm for this project since it can compute the probability of personality types directly and could handle less irrelevant features such as words common to all personality types very well by using a weight parameter close to 0 [9]. Based on the four personality dimensions, the first step was to separate each MBTI personality type into four classification groups and use logistic regression to do a binary classification on each group. In Logistic Regression, there are two possible outcomes Y1 and Y2, with P(Y1) = P(1-Y2). By comparing the prediction probability to the threshold, we can find the likelihood that Y1 or Y2 happens. Then, the algorithm chooses the most probable outcome to be the classification for that data point. To implement logistic regression, we used the built-in Logistic

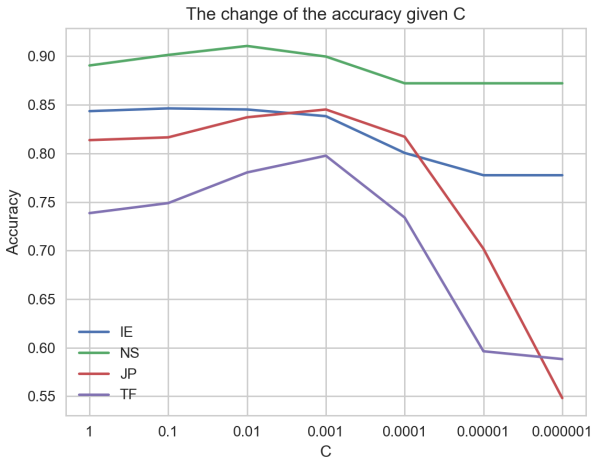

Fig. 2. The Change of Accuracy Given C

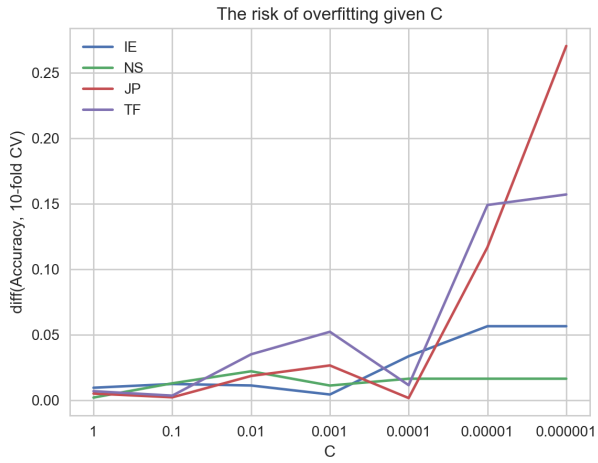

Fig. 3. The Risk of Overfitting Given C

Regression algorithm on sklearn to train and predict the data. In the beginning, no parameters were set.

The default regulation strength was set to 1 and the default tolerance for the stopping criteria is 0.0001 which gave a fairly high accuracy (Fig. 2). By increasing the regulation strength from 1 to 10, and 10 to 100, the accuracy of the model increased except for the Introversion vs. Extroversion dimension. However, when we set the regulation strength to 1000, the accuracy decreased. So in our initial tests, the optimal settings for C were:

- I/E C = 0.1
- N/S C = 0.01
- J/P C = 0.001
- T/F C = 0.001

Since lowering the stopping criteria meant a smaller step size for each iteration, it may help increase the accuracy. However, there is a trade-off as making the stopping criteria smaller makes the minimum error converge more slowly. We obtained the highest overall accuracy by setting C as above without considering the risk of overfitting.

We decided to compute the difference between accuracies and the cross-validation score to measure how much the model was overfitted (Fig.3.). We found that some of our original settings of C caused overfitting, so we decided to change all C equal to 0.1. Afterwards, we tried tuning other hyperparameters, but none of them helped improve the accuracies. The second step was to construct the 16-way classifier. Our first approach for the 16-way classifier was to label the personality types from 0 to 15 and train the model on the whole data. The second approach is to combine the prediction of the data in the first step and trained them correspond with the labels from 0 to 15. We wanted see how likely the combination of these results can produce the right 16 labels. In the end, we found that the first approach had a better accuracy than the second one. However, the 16-way classifier produced a much lower accuracy than the results we got from the first step.

C. Support Vector Machine Design

As with the other models, classification with the SVM model was done in two steps. The first step was to perform binary classification on each of the personality dimensions. To achieve this, four different SVMs were trained on the training sets and the hyperparameters were evaluated on the validation sets using 8-fold cross validation. Our initial anticipation of the possibility of the data not being linearly separable appeared correct as the linear kernel function produced suboptimal results with the best kernel function being rbf for all four models. The most time consuming hyperparameter to optimize was the penalty parameter C. Our initial expectations were that the dataset would be very noisy for which a very low C value would work best for, however this turned out to not be the case as the optimal C values were quite high:

- I/E: c = 17
- N/S: c = 90
- T/F: c = 91
- J/P: c = 14

Figure 5 shows the change in accuracy with respect to the change in the C hyperparameter.

The second step after building the four binary classifiers was to feed the entire data set through them to get the probability predictions - how likely was the SVM to classify an entry as a particular binary class. We then put these probabilities in a matrix where each row represented an entry and each column represented the probability prediction of the matching personality dimension. This was our input data for the final 16-way classification. Again we split this data into a training (80%) and testing set (20%) and trained a new 16-way SVM classifier using 8-fold cross validation. Generally, SVMs are used for binary classification, however scikitlearn's implementation allows for multiway classification by incorporating multiple SVM sub-models to do several binary classifications and produce a final multi-way classification. Similarly to the binary SVMs, the rbf kernel yielded the highest accuracies,

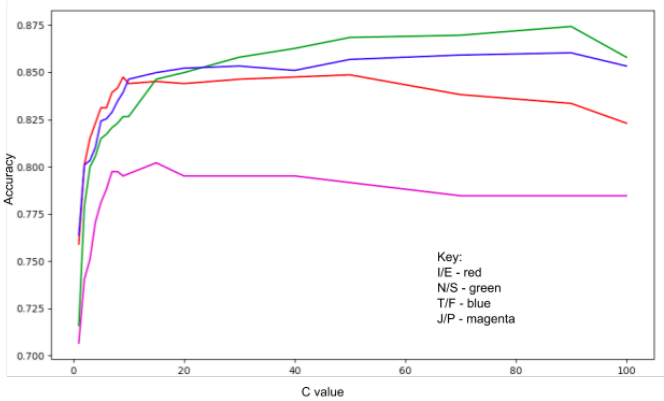

Fig. 4. The change in accuracy with respect to the change in the C hyperparameter

however the penalty hyperparameter C was much lower at 0.08. This meant that the final classifier wasnt overfitting to the noisy probabilities produced by the binary classifiers and instead created a wider separation margin. Finally the test data was run through the 16-way classifier to produce the final reported accuracy.

D. Convolutional Neural Network Design

The reason that we chose to use convolutional neural network (CNN) was because a CNN has the shared-weights feature and thus can significantly reduce the number of parameters to train compared to a fully connected neural network (NN). A pre-trained embedding layer was used to transform the count vectors (or index vector in our case) to the word vectors (word embeddings) before the first convolutional layer in the CNN model. In other words, the actual input for the CNN model was the word vector, not the count vector. The input vector had a fixed length of 500 since we simply assume that the posts for one person has an average length of 500 words. We used the Tokenizer class in keras (similar to scikit-learns CountVectorizer) to build the vocabulary and vectorized the input text. The input vector will thus be an index (in the vocabulary) vector. If the input text has more than 500 words, we truncated the less important words (i.e., words with lower tf-idf score); if the input text had less than 500 words, we added paddings (zeros) to make the length of input vector to be 500. After the input index vector, we added a pre-trained embedding layer to transform each word index into a word embedding (a word vector). We used GloVe (chose 100 dimension, the length of a word vector) to create the embedding matrix for the embedding layer. Therefore, the actual input for the CNN (after the embedding layer) is 100-dimension word vectors for the input text, not the word index vector. First, we built a CNN model to directly predict the 16 personality types. The structure of this CNN model is shown in Fig. 5. Following the embedding layer, there were two sets of convolutional and max-pooling layers. In each of the convolutional layer, there are 50 filters with a window size of 4. The max-pooling window is also set to be 4. The

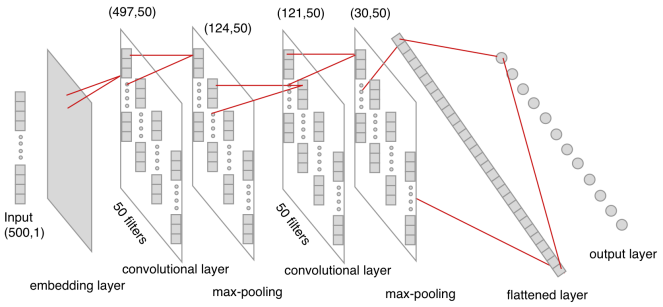

Fig. 5. Structure for a single CNN 16-way classifier

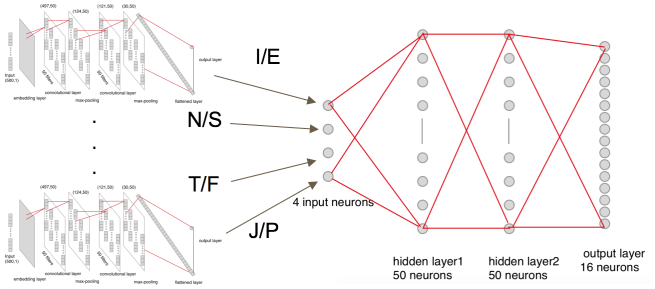

Fig. 6. Structure for an ensemble 16-way classifier by stacking 4 binary classifiers with a fully connected NN model

dimensionality for each layer's output is labeled in Fig. 5. We did the 8-fold cross validation to study our model's robustness to the overfitting problem as well as to decide the number of iterations (epochs) we will use for the final model instance. Fig. 7(a) shows the train (70% of the original data set) and validation (10% of the original data set) accuracy as a function of the number of iterations. Each point in the figure is an average of the 8 accuracies since we did the 8-fold cross validation. From the accuracy curves, we can clearly see that this CNN model can overfit to the training data with increasing iteration numbers, the training accuracy increases while the validation accuracy, on the contrary, decreases. We chose the diverging point (5 iterations) as the early stopping criterion to prevent overfitting. Then we trained our model using train and validation data (total 80% of the original data) with 5 iterations to generate a instance of the 16-way CNN model. We tested the instance on the testing data to report the 16-way classification accuracy. With this method, we obtained an accuracy of 33%.

In order to improve the prediction accuracy as well as to overcome the overfitting problem, we investigated the ensemble method, more specifically, the stacking method, by combining different models together to increase the accuracy. First, we used a similar CNN model to perform the binary classification on each of the 4 dimensions (I/E, N/S, T/F, and J/P). The structure of the CNN for binary classification is similar to the structure shown in Fig. 6 except that the output layer for binary classification uses sigmoid as the activation function instead of softmax in 16-way classification.

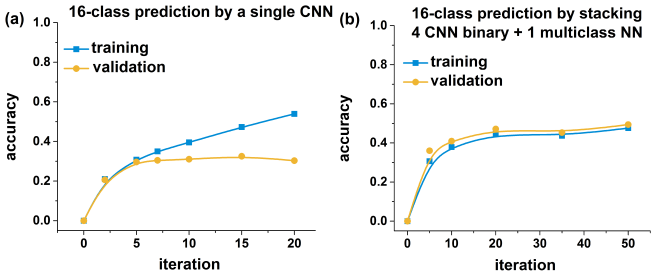

Fig. 7. Training and validation accuracies as a function of the number of iteration for: (a) the single CNN 16-way model; (b) the ensemble model

We used the similar cross-validation method and found 15 to be a reasonable number of iterations to train the binary model. For each of the 4 dimensions, we trained the model separately and outputted the probability from the sigmoid function as the new feature for each of 4 dimensions. After the binary classification, we transformed the original text dataset into a new dataset with only 4 features. We fed the new input data into a fully connected NN model to perform the 16-way classification. The whole structure of the ensemble model is shown in Fig. 6. The fully connected NN model has two hidden layers and each of the hidden layer had 50 neurons. The output layer had 16 neurons representing the 16 personality types. The 8-fold cross validation was used to study the model's robustness to overfitting. Fig. 7(b) shows the training and validation accuracy as a function of the iteration number. We found that this ensemble model was more robust to overfitting since the trend for train and validation accuracy is the same for different number of iterations. We chose 100 to be the final number of iterations to train our ensemble model using 80% of the data (training + validation), reported the final accuracy on the testing data (20% of the original data), and calculated the confusion matrix based on the true label and the predicted label. The reason that the ensemble model was more robust to overfitting is because after the binary classification, we reduced the number of features to 4, which is less than the number of input texts. In other words, the number of input texts we have is quite sufficient for training a model dealing with only 4 features.

V. Results

Here, we report the results of the three algorithms we implemented where we have reported the final accuracy and visualized the predictions for each personality type code using a confusion matrix. The final accuracy was calculated by the number of total correct predictions divided by the the number of samples in our test set.

For the four binary classifiers, we aim to beat the following values for each dimension: I/E: 76.95%, N/S: 86.20%, T:F: 54.11%, and J/P: 60.41% as these accuracies could be achieved by a naive classifier.

For the 16-way classification for the MBTI personality types, we aim to beat the score of 21% which is the proportion of the class with highest representation in the dataset, INFP.

A. Logistic Regression Results

The Logistic Regression on the four dimensions yielded the following accuracies on the testing set:

- I/E accuracy = 84.5%
- N/S accuracy = 91.1%
- T/F accuracy = 75.0%
- J/P accuracy = 83.7%

Final accuracy of 16-way classifier: 61.3%

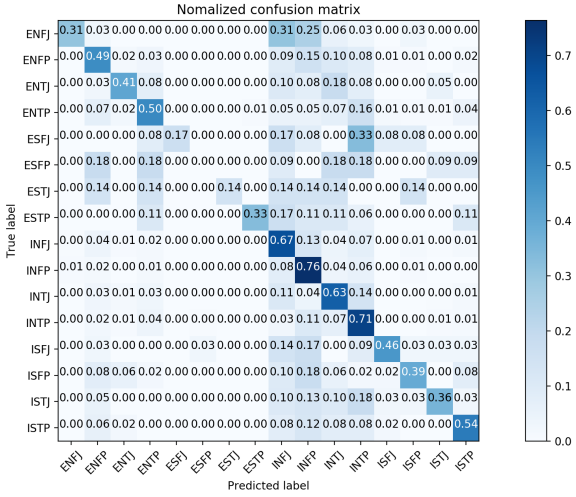

Fig. 8. Logistic Regression Confusion Matrix on the 16 Personalities

B. SVM Results

The SVM algorithm on the four dimensions yielded the following accuracies on the testing set:

- I/E accuracy = 84.5%
- N/S accuracy = 87.4%
- T/F accuracy = 86.0%
- J/P accuracy = 80.2%

Final accuracy of 16-way classifier: 71.9%

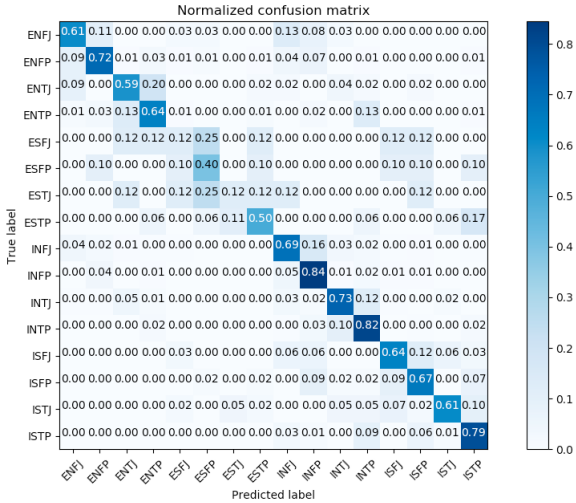

Fig. 9. SVM Confusion Matrix on the 16 Personalities

C. CNN Results

The CNN algorithm on the four dimensions yielded the following accuracies on the testing set:

- I/E accuracy = 75.2%
- N/S accuracy = 85.3%
- T/F accuracy = 74.1%
- J/P accuracy = 67.4%

Final accuracy of 16-way classifier: 60.2%

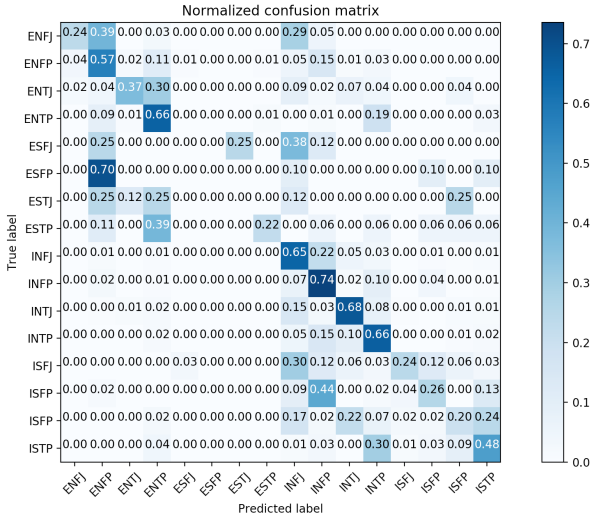

Fig. 10. CNN Confusion Matrix on the 16 Personalities

VI. Discussion and Analysis

A. SVM

Looking first at the binary classifiers, since the data set was quite unbalanced, it is not enough to say that the aim of each binary classifier is to beat 50% accuracy but instead, the goal of each classifier is to beat a random classifiers accuracy on that personality dimension. In this case the random classifiers accuracies would be the following: I/E: 77.0% N/S: 86.2% T/F: 54.1% J/P: 60.4% As such, the SVM binary classifiers still achieved significant improvements over random classifiers in all but the N/S dimension, where the improvement was in the order of 1%. This may be attributed to the very unbalanced distribution of the data we had for that dimension, and the SVM may have performed better had the dataset been more evenly distributed in that axis. Another possibility is that it is just generally difficult to determine whether a person is more inclined to use intuition or sensing from their writing style.

Similarly to the binary classifiers, the accuracy of the 16-way classifier should not be compared against a random accuracy of 6.25% as again the classes are unbalanced. The most common class was INFP accounting for 21% of all the data, so the accuracy of a random classifier would be 21%. Nevertheless 71.9% is far higher than 21% so it is safe to say that the 16-way accuracy of the SVM model was very good. However with 16-way classification, the other value to beat would be the product of the accuracies of the binary classifiers as that would be the average accuracy of a model that simply predicts the binary axes and combines them together. Taking the product of the accuracies of the binary classifiers gives 50.9%, 71.9% is still substantially greater than this naive combination of binary classifications. This validates the use of an additional 16-way SVM.

Looking at the confusion matrix for the 16-way SVM we notice that the highest values tend to be along the diagonal. This is good because it means that the model usually predicts the correct personality type. However, it is interesting to analyze why the model made mistakes. The two classes that the model got least correct were ESTJ and ESFJ. This may be attributed to it being difficult to determine whether a person is more on the thinking or feeling side of the spectrum, however that is unlikely due to the accuracy of the binary T/F classifier being much higher than that of a random classifier. The more likely possibility is that these are the two most underrepresented classes in our data set so it is likely that the model simply didnt have enough data to train on. This accuracy to amount of data correlation is consistent among some other classes such as INFP, however it is not consistent across all of them. For example, ENFJ types were predicted correctly 61% of the time, however they composed under 4% of the data set. This leads us to the conclusion that the accuracy of the model is likely to be affected by both the availability of data as well as some personality types being easier to predict correctly than others.

B. Logistic Regression

Logistic regression also performed better than a random classifier on each personality dimension. However, by comparing the precisions between two personality types in each axis, we noticed that they are very different to each other. For example, in the NS group, we have intuition (47% correct predictions) vs Sensing (96% correct predictions) which differ greatly. After tuning the hyperparameters, we got a much lower accuracy on the 16-way classifier, 61%, than the results we got from binary classification on 4 axis. This corresponds to the discoveries that the precision of classes inside each dimension greatly vary from each other. By looking at the confusion matrix on 16-way classifier (Fig. 10), we found that it performed the worst on minority classes like ESFP (0% correct prediction). This may have been caused by having too few samples those classes. One way we could fix this problem would be to add more samples to the minority classes. Then we can make sure it has more relevant data to train on.

C. CNN

The performance of the CNN binary classifier was close to a random classifier. For example, for the N/S dimension, the accuracy of the CNN binary classifier was 85% while the accuracy for a random classifier was 86%. Therefore, we can say that these CNN binary classifiers are relatively weak learners. Since the dataset is unbalanced on each dimension, the less unbalanced dimension will obtain relatively better performance compared to the random classifier. For example,

the T/F dimension has relatively balanced samples. Therefore, the CNN binary classifier for the T/F dimension can achieve 74% accuracy, much higher than the accuracy of 54.1% from a random classifier for this dimension.

The single CNN 16-way classifier can only yield an accuracy of 33%. However, the model by stacking 4 binary classifiers with a fully connected NN can yield an accuracy of 60%, much higher than the single model. This proves that the stacking method can indeed boost the accuracy, even if the combined models are relatively weak learners separately. Taking the product of the accuracies of the 4 binary classifiers gives 31.6% and so 60% is still greater than this naive combination of binary classifications. This validates the use of an additional 16-way fully connected NN. From the confusion matrix, we can observe that the INFP type has the highest accuracy since the number of INFP samples in the input dataset is the highest. We can also observe that for some types (i.e., ESFJ, ESFP, and ESTJ), the ensemble model will fail to predict. These three types happen to be the three types with the smallest number of samples in the input data, showing that the model has a high chance to fail to learn the features of the three types simply because of the inadequate number of samples. One reasonable method to increase the accuracy for these types would be adding more samples of the three types to the input data.

VII. Conclusion and Future Work

By comparing the accuracy across the three models we investigated, the most accurate model we obtained was the ensemble SVM model by stacking 4 binary classifiers with a multiclass SVM classifier. This model can yield a final accuracy of 71.9%, which is higher than the other two algorithms we investigated. As one of our research questions was to see whether we can detect a relationship between peoples personality types and patterns in their written text, the fact that we achieved such a high 16-way classification accuracy serves as evidence for the validity of the MBTI personality classification model.

If we had more time for this project, there are several next steps that we would have like to explore. By looking at URL links, it would be interesting to follow the link to the type of website, image, or other media to use as features as this may contain relevant information in regards to the poster's personality. Also if we had more time, we would want to explore different types of ensemble classifiers in addition to our stacking method such as boosting algorithms and bagging. We could also try to take the output probabilities output by most accurate model for each personality axis and combine those into an ensemble classifier rather than keeping different types of models separate. Lastly, we would want to expand our dataset so that our training data is more diverse as well as having a broader set of demographics of users. This could be done by scraping posts from other online personality forums that provide the MBTI type classification.

VIII. Ethical Implications

Because our project is concerned with inferring user attributes, there is a greater responsibility to consider the ethical implications and negative social impacts that our tools could potentially have. We will specifically look at the issues regarding exclusion, overgeneralization, and dual use.

A. Exclusion

As mentioned previously, the data used for this project was sourced from the online personality forum, PersonalityCafe. It is very reasonable that the type of users active on the website may hold a demographic bias. Although we do not know the exact ethnicities or source countries the users are from, we do know that they are English speaking, that they are familiar with the internet, and that they have the leisure time to post on a personality forum indicating the users most likely come from developed area. From these observations, we can infer that the users of the this website are likely western, educated, industrialized, and relatively rich. This presents the problem of exclusion since this demographic bias is not representative of the global population. Because our model trained on the one source of data, overfitting to the demographic bias in presents an ethical problem as the model may not generalize to demographics outside the ones in our training day. If we were to have the additional demographic information of the users of our training data such as gender or ethnicity, a remedy to the demographic bias could be to downsample any over represented group to even out the distribution in the data.

B. Overgeneralization

Since it is impossible that our machine learning models will be a perfect predictor of every single user due to the complexity of humans and their personalities, it is fair to expect our models to sometimes misclassify users, ie. produce false positives. Therefore, it is important to consider the question of when a false answer be worse than no answer. In the case of our project, this would happen in a situation where the consequences of being misclassified by personality type were great. For example, if the personality prediction was used as a requirement for a job, or evidence for a legal case. To ensure that our models will not be used for these purposes, we must be clear to state that our personality predictions models are indeed imperfect and have a fair chance at producing false answers. Furthermore, since the notion of the MBTI personality types validity should also be considered for anyone wishing to draw conclusions on a person based on it.

C. Dual Use

The intention for this project was to study the relationship between writing style and personality by seeing if we could accurately predict personality type. However, it is possible that a person or organization with a particular motive, could use this model to classify people as a way to discriminate against certain personality types. For example, if a hiring manager or company had a preference for more extroverted people, and used our models to filter out any applicants the model

classified as introverted. On an organizational level, since our project involves text classification, our models could be appropriated for censorship if an organization wanted to silence the opinions of certain personality types. Although there is little we can do to prevent the misappropriation of our tool as developers, we hope that the efforts of the community will bring any misuses of our tool to light so that the perpetrator can then have their actions judged appropriately.

IX. Project Timeline

- January 16 – Project pitches
- February 6 – Submit Proposal
- March 5 – Clean and pre-process data
- March 8 – Train/implement algorithms
- March 9 – Submit Midterm Report
- March 12 – Tweak parameters for algorithms
- March 15 – Formal evaluation and analysis of results
- April 3 – In-Class project presentation
- April 8 – Complete discussion and analysis of results
- April 9 – Finalize and submit report

X. Task Distribution

Throughout the project, each team member was assigned as a task lead for each of the major components of the project. Kathy was responsible for preprocessing the data and formalizing the deliverables, Nikita was responsible for the implementation of the SVM algorithm, Jian was the lead on implementing the neural network and researched related works, and Steven led the logistic regression model. Although tasks were split up by team member, it is important to note that each team member contributed to others tasks through discussion and aiding with supporting tasks.

References

[1] "MBTI Basics", Myersbriggs.org, 2018. [Online]. Available: http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1. [Accessed: 06- Feb- 2018].

[2] D. Pittenger, "Cautionary comments regarding the Myers-Briggs Type Indicator.", Consulting Psychology Journal: Practice and Research, vol. 57, no. 3, pp. 210-221, 2005.

[3] J. Golbeck, C. Robles, M. Edmondson, and K. Turner, "Predicting personality from twitter." In Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, pp. 149-156. IEEE, 2011.

[4] M. S. Halawa, M. E. Shehab, and E. M. R. Hamed, "Predicting student personality based on a data-driven model from student behavior on LMS and social networks." In Digital Information Processing and Communications (ICDIPC), 2015 Fifth International Conference on, pp. 294-299. IEEE, 2015.

[5] K. Luyckx and W. Daelemans, "Personae: a Corpus for Author and Personality Prediction from Text." In LREC. 2008.

[6] A. Ma and G. Liu, "Neural Networks in Predicting Myers Brigg Personality Type From Writing Style."

[7] Personalitycafe.com. (2018). Personality Cafe. [online] Available at: http://personalitycafe.com/ [Accessed 7 Feb. 2018].

[8] Kaggle.com. (2018). (MBTI) Myers-Briggs Personality Type Dataset — Kaggle. [online] Available at: https://www.kaggle.com/datasnaek/mbti-type [Accessed 7 Feb. 2018].

[9] P. Tan, M. Steinbach, A. Karpatne and V. Kumar, Introduction to data mining, 2nd ed. New York, NY: Pearson Education, 2018, p. 248.