Predictive Modeling Of MBTI Personality Type Based On Social Media Posts

(CS 5785) Applied Machine Learning, Cornell Tech, Fall 2021 Volodymyr Kuleshov

Harper Zhu (rz386) Katherine Wu (kw634) TingKang Zhao(tz275)

ABSTRACT:

While many existing research focuses on training machine learning models to predict MBTI using text-based input, only a few research is dedicated to improving the performance of a given machine learning model with parameter tuning. In this paper, we used an existing dataset to predict users' personality types based on Twitter posts. Using the random forest as our prediction model, we concluded that there are a few parameters that have a significant impact on the performance: 1) the splitting ratio of the training and testing set, 2) the decision tree's maximum depth, 3) the number of estimators 4) the Minimum sample leaf. Our analysis indicates that the best ratio of the training and test sets is 60%: 40%. In our case, the optimal minimum sample leaf is 13; The decision tree's maximum depth is 15, and the number of estimators is 1500. While this result is obviously not applicable to any other studies, as all dataset and machine learning model differs, we hope this paper provides some insights into how hyperparameters can impact the performance of a machine learning model.

KEYWORDS

Machine Learning, MBTI, Social Media, KNN, Random Forest, SVM

1 MOTIVATION

The problem we are trying to solve is to use a Machine Learning algorithm to predict a given person's MBTI personality type based on their social media post. With the growing popularity of MBTI, more and more people started to believe that MBTI personalities can help them find friends on social media hassle-free. As many people reported that their MBTI personality varied from time to time, an interesting application is to use social media posts to determine what is people's true personality types at the moment. It can help individuals understand their own communication preferences and find their most comfortable way of interpersonal interaction.

2 CONTEXT

The Myers-Briggs Type Indicator (MBTI) is one of the most widely used introspective self-report questionnaires that describe personality types. It indicates how people have different psychological preferences in how people perceive the world and make decisions with 4 binary categories and 16 personality types in total. The four categories are 1) Energy: **E**xtrovert / **I**ntrovert 2) Information: **S**ensing / I**N**tuition 3) Decision: **T**hinking / **F**eeling 4) Lifestyle: **J**udging /

Perceiving.MBTI personality is defined as the collection of their four types for the four binary categories, indicated above with the bolded letter for each type. There are 4 types of personality groups among the 16 types: Analysts who are Intuitive and Thinking (I**NT**J, I**NT**P, E**NT**J, E**NT**P), Diplomats who are Intuitive and Feeling (I**NF**J, I**NF**P, E**NF**J,E**NF**P), Sentinels who are Sensing and Judging (I**ST**J, I**SF**J, E**ST**J,E**SF**J) and Explorers who are Sensing and Perceiving (I**ST**P, I**SF**P, E**ST**P, E**SF**P). This type of personality group will be the presentation scheme this paper used to discuss prediction.

While personality type indicator is becoming an increasingly popular tool to understand the inner self and people around us, many researchers at prestigious institutions have started to investigate how to predict MBTI personality type with Machine Learning [1][2][3].

Many of them tried to use machine learning model (e.g. XGBoost, Support Vector Machine) to predict a person's personality based on various text-based data set (e.g. from survey response or social media posts). While many of the studies compare the performance of different machine learning data, few actually analyzed what parameters impact the performance of different models and how parameters could be tuned to improve prediction performance. The analysis we performed in this paper was obviously not revolutionary, but we hope it can add to the understanding of how different parameters impact the performance of different Machine Learning models and potentially provide an idea of how performance improvement can be made.

3 PRELIMINARY EXPERIMENTS AND PREPROCESSING

3.1 Preprosessing Data Visualization

The datasets we used for this paper were based on the Kaggle dataset. We collected, cleaned, and preprocessed the datasets. During preprocessing, we removed URLs, punctuation, emails, and stop words

from the Twitter posts. We also lemmatized all the posts to ensure different forms (e.g. plural form) of the same word are grouped together. The removal of punctuation and stop words helped us avoid interruptions during the word grouping process to ensure less noise and higher accuracy. The use of a lemmatizer allows us to query any version of a base word and get relevant results. The advantage of such preprocessing is to analyze the same words in different forms as one item, which will likely improve the consistency and accuracy of our analysis.

We categorized 16 personalities into 4 groups of personality. However, we realized that the dataset itself has an unbalanced distribution among the four groups. There are significant differences in the proportion of samples for each group. While Diplomats have over 4000 samples, Sentinels only have 400(see figure 1).

[1] Cui, B., & Qi, C. (2017). Survey analysis of machine learning methods for natural language processing for MBTI Personality Type Prediction.
[2] Sönmezöz, K., Uğur, Ö., & Diri, B. (2020, October). MBTI personality prediction With machine learning. In *2020 28th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.
[3] Amirhosseini, M. H., & Kazemian, H. (2020). Machine learning approach to personality type prediction based on the myers–briggs type indicator®. *Multimodal Technologies and Interaction*, *4*(1), 9.

One way of balancing the sample distribution is to randomly select 400 samples from each of the categories. While this would potentially reduce bias in prediction, it would significantly reduce the size of our training and testing dataset, increasing the margin of error and rendering the study meaningless.

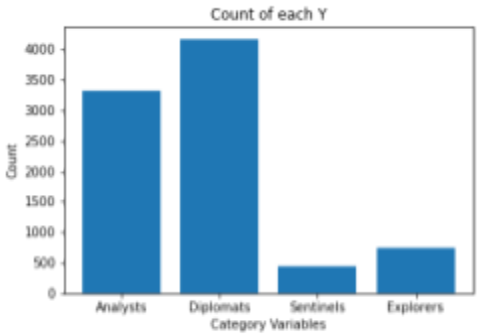


Figure 1: Imbalance data

3.2 Hypothesis

Our hypothesis is that the increase of decision tree's maximum depth will make the training error go down (or at least not go up). This is because the decision tree's maximum depth measures how many splits a tree can make before coming to a prediction. Splitting can occur until the tree is as pure as possible. Thus, for the training dataset, it will likely reduce the training error with the potential issue of overfitting. Similarly, on the testing dataset, the prediction error should go down.

For the splitting ratio of the training and testing set, our hypothesis is that 80:20 as the most commonly used splitting ratio would bring out the highest accuracy. This is because more training set will decreases the generalization error as our model becomes more general by virtue of being trained on more examples.

In terms of the number of estimators, our hypothesis is that increasing the number of estimator will increase the predictive accuracy at the cost of slower learning. This is because the number of estimators are an equation for picking the most likely accurate, data model based upon observations in realty. The number of estimators will thus mean more time-consuming evaluation and higher chance of prediction accuracy.

In terms of the minimum number of samples, our hypothesis is that a smaller minimum number of samples can lead to a model with higher complexity and a greater potential for overfitting the training data. By the same logic, a larger minimum number of samples can lead to a model with lower complexity and a greater potential for underfitting the training data.

4 METHOD

4.1 KNN

K-nearest neighbors (k-NN) is a type of supervised machine learning algorithm that is used for classification and regression tasks. For classification, the k-NN algorithm works by taking a new data point and finding the k-nearest data points in the training set. In order to determine the class label of the new data point, the class labels of these k-nearest data points are then combined using a majority vote. The value of k determines the number of nearest neighbors that are used to make the prediction, which can be chosen based on the characteristics of the data and the desired performance of the model.

The KNN model is a non-parametric, supervised learning classifier. While it can be used for either regression or classification

problems, it is most commonly used as a classification algorithm based on the assumption that similar points can be found near one another. We would choose different n and to see which one has the best results.

The reason why we chose KNN is that it is widely used in various applications of NLP, such as text categorization, question answering, and information retrieval. The advantage of KNN is that it is simple to implement and easy to understand.In addition, It can provide good performance on a variety of classification tasks and can offer the flexibility of use with different distance metrics and different values of k. Furthermore, it does not make any assumptions about the underlying data, so it can handle complex non-linear relationships. As a non-parametric model, KNN has the advantage of not losing any information at training time.

The disadvantage of KNN is that as a non-parametric model, the function uses the entire training dataset to make predictions, and the complexity of the model increases with the dataset size. KNN is also computationally less tractable and may easily overfit the training set because it requires storing the entire training set and finding the k-nearest neighbors for every new data point.

Despite this disadvantage, k-NN is a popular algorithm because it can provide good performance on various tasks and is relatively simple to use.

4.2 SVM

Support vector machines (SVM) is a supervised learning method used for classification, regression, and outliers detection.

For a classification problem, an SVM works by 1) mapping the input data into a high-dimensional space using a kernel function and 2) finding the best hyperplane that separates the data points into different classes. The distance between the hyperplane and the closest data points is called the margin. The goal of SVM is to choose a hyperplane with the latest margin so that it can maximally separate the class. Once the hyperplane has been determined, new data points can be classified by determining on which side of the hyperplane they fall.

One of the reasons we chose SVM is that it is widely used for Natural Language Processing with good generalization performance, especially when using regularization to prevent overfitting.

However, there are also some disadvantages to SVM. They can be computationally expensive to train, especially for large datasets or when using a non-linear kernel. They are difficult to tune as they have more hyperparameters to adjust and are more sensitive to the choice of parameters. They may not perform well on data with a lot of noise or when the classes are not linearly separable. They could also be hard to interpret as the decision boundary is determined by the support vectors, which are a subset of the training data.

Overall, the advantages of SVMs often outweigh their disadvantages, but they may not be the best choice for every problem. It is important to carefully evaluate the strengths and weaknesses of different machine learning

algorithms and choose the one that is best suited to your specific task and dataset.

4.3 Random Forest

Random forest is a supervised machine-learning algorithm that is used in Classification and Regression problems. It is an ensemble learning method that combines the predictions of multiple decision trees, each of which is trained on a random subset of the data. Each decision tree is trained differently with various algorithms (e.g., the CART and/or ID3 algorithm), and each has its own formula for tree generation. After the individual trees have been trained, the random forest combines the prediction of all trained individual trees using a majority vote or averaging their outputs. Eventually, the random forest algorithm outputs the result with the highest frequency based on the output of those trees.

One of the reasons why we chose Random Forest is because it avoids overfitting. The ensemble approach is the key to reducing overfitting. Since each individual tree is trained on only a small subset of data, it is less likely to overfit the training data. Further, bagging is used to train the trees in a random forest, which ensures that each tree is trained on a different bootstrap sample of the data. Since each tree sees a different subset of the data, they are likely to make different predictions. The combination of prediction also ensures that if there is a prediction that is overfitting, it will be smoothed out by the ensemble. Thus, the ensemble makes random forest predictions more generalizable and less likely to overfit the training data.

Another important reason why we use the random forest is that it is often used for natural language processing tasks and has great performance. This is because random forests can handle large, high-dimensional datasets and learn about the complex non-linear relationships between the input data and the output labels.

However, there are also disadvantages to using a random forest. They are difficult to interpret as they are a BlackBox model.

5 SETUP

5.1 Dataset

Our project is mainly based on the Kaggle dataset, which consists of a total of 2000 tweets. The tweets were selected from random people's social media. These tweets are labeled by sixteen classes, i.e., INFP and ESTJ. We selected this dataset due to its considerably large size, well-controlled quality, and trustworthy labeling.

5.2 Metrix

We monitored accuracy to measure the performance of our model. We define True as correctly classifying data and False as misclassifying data. And the accuracy is calculated as

$$Accuracy = (TP + TN) \div (TP + TN + FP + FN)$$

$$Precision = TP \div (TP + FP)$$

$$Recall = TP \div (TP + FN)$$

$$F1\,Score = 2 \times (precision \times recall) \div (precision + recall)$$

5.3 KNN

KNN is one of our baseline models. We chose KNN because it is easy to train, and the parameter is easy to tune.

Our prediction accuracy for KNN is 30%, and our F1 score is 45%. The weakness of KNN lies in the fact that it is a non-parametric model and thus requires more data as the number of

features increases. The increase in dataset size would also create a potential for overfitting as no one knows the relevance of the data point. A study in Text Categorization using KNN also demonstrates that KNN performs better with low dimensionality[4].

KNN is our milestone, and when we run KNN, we find out the result is around 30%.

5.4 Support Vector Machine(SVM)

SVM is one of our baseline models. We chose SVM because they are known to perform well on high-dimensional data sets with complex boundaries. This makes them a good choice as a baseline model for many machine-learning tasks.

The dataset is the same as the one we applied, KNN and Random Forest. We imported the support vector machine package from sklearn and applied it to the dataset. We split the dataset in half into the training dataset and the testing dataset. The metrics are similar to the ones we used for KNN and random Forest.

Our F1 score for SVM is 0.72, and the prediction accuracy is 0.67. This might be because SVMs are sensitive to the choice of kernel function and hyperparameters, which can make them difficult to tune for optimal performance. In addition, SVMs are not well-suited to problems with large numbers of features or instances, as the computational complexity of the algorithm grows rapidly with the size of the data set.
From the prediction accuracy, we can see that we need a model that can handle a larger dataset and train much faster. In addition, our ideal model is a model that is easy to tune. Random Forest is faster to train with large datasets. In addition, Random Forest is generally easier to use and tune than support vector machines, as they have fewer hyperparameters to adjust and are less sensitive to the choice of parameters. Therefore, we decided to move to Random Forest.

5.5 Random Forest

After preprocessing our dataset, we split it into two parts(training dataset and test dataset). We first set the ratio to be 80%: and 20%, respectively, run the random forest, and then got our results.

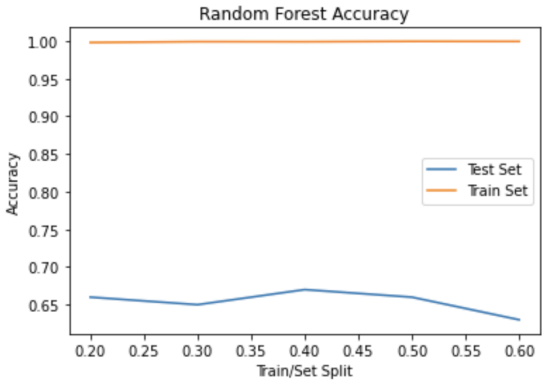
	Training set	Test set
Accuracy	99.87%	66.22%

5.5.1 1st Optimizing: the splitting ratio of training and testing set

As we see in the table above, there existed an overfitting issue since our training set gave us high precision while the test did not. In order to solve this issue, we chose to increase the size of our training dataset by setting the ratio of the test set from 20% to 60%. And the results are shown below.

The ratio of the training set and test set	Training set accuracy	Test set accuracy
80%: 20%	99.87%	66.22%
70%: 30%	99.92%	65.81%
60%: 40%	99.9%	67.00%
50%: 50%	99.95%	66.20%
40%: 60%	99.94%	63.32%

[4] Jiang, S., Pang, G., Wu, M., & Kuang, L. (2012). An improved K-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, *39*(1), 1503-1509.



As the table shows above, increasing the test ratio helped us increase the accuracy of the test set when it increased from 20% to 40%. Then the accuracy decreased as it continued to increase.

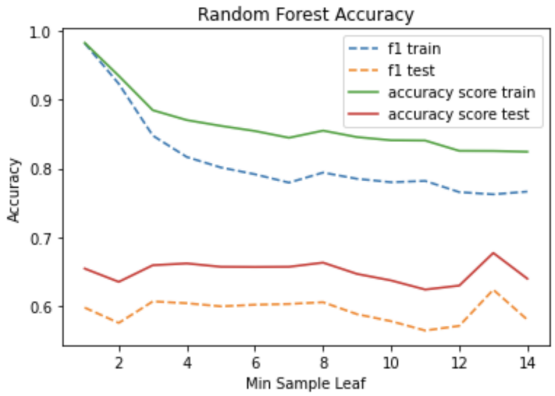
Thus, we concluded that the best ratio of the training and test sets is 60%: 40%.

5.5.2 2nd Optimizing: Minimum sample leaf

After finding out the best ratio of the training set and test set, we still assumed there was an improvement in accuracy. And then, we used a for loop to go through from 1 to 5 as the minimum sample leaf. Then we saw the results of the train set and test set as below.

Accuracy for different sample leaves (60%: 40% ratio)

The Minimum sample leaf	Set	Accuracy	F1 Scores
2	Training set	99.31%	99.83%
	Test set	65.12%	55.75%
6	Training set	88.62%	80.21%
	Test set	65.19%	60.08%
13	Training set	88.68%	80.18%
	Test set	69.68%	63.48%
14	Training set	83.22%	79.18%
	Test set	65.41%	59.15%

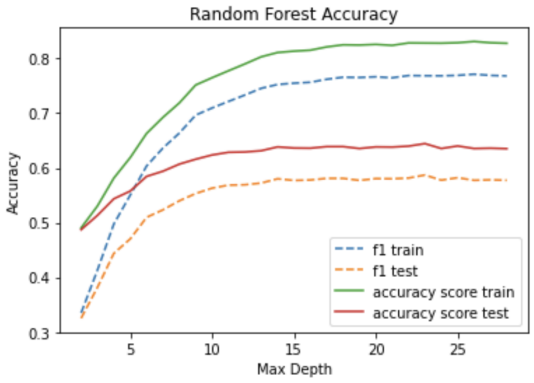


From these four experiments, we found that the optimal number of the minimal sample leaf is 13. When the minimal sample leaf is equal to 13, the prediction accuracy on the test set reach its peak at 69%. The f1 score of the test set also reached its peak at 63%. After the minimum sample leaf hits 13, the prediction accuracy and the f1 score in both the training set and the testing set decrease as the minimum sample leaf increases. Thus, the optimal minimum sample leaf, in this case, is 13.

5.5.3 3rd Optimizing: decision tree maximum depth

We also assumed there is some improvement in the maximum depth. And then, we used a for loop to go through from 2500 to 3200 as the minimum sample leaf. Then we saw the results of the train set and test set as below.

number of estimator	Set	Accuracy	F1 Score
5	Training set	49.26%	33.18%
	Test set	49.38%	33.29%
10	Training set	77.55%	68.29%
	Test set	61.34%	53.01%
15	Training set	82.62%	73.19%
	Test set	61.75%	54.52%
25	Training set	83.93%	73.23%
	Test set	62.37%	54.39%

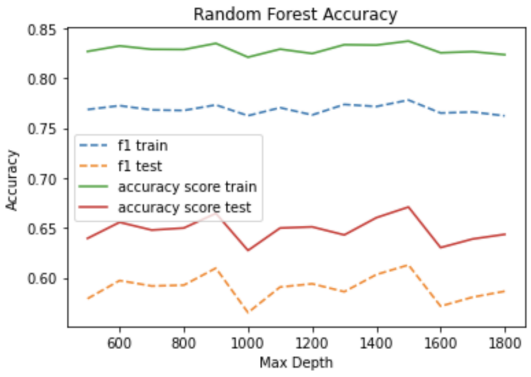


From these experiments, we found that while the f1 score of the train set is around 60%, the f1 score of the test set reached the highest as 77.8% when the maximum depth is 15. Thus, setting the decision tree's maximum depth to a low value can prevent overfitting, and setting it to a high value can improve the model's ability to fit the training data.

5.5.4 4th Optimizing: the number of estimators

To determine if there is an improvement in the number of estimators, we used a for loop to iterate from 2500 to 3200 as the value for the minimum sample leaf. Then we saw the results of the train set and test set as below.

the number of estimator	Set	Accuracy	F1 Score
1000	Training set	82.09%	76.23%
	Test set	62.73%	56.51%
1200	Training set	82.45%	76.30%
	Test set	65.10%	59.40%
1500	Training set	83.70%	77.80%
	Test set	67.08%	61.29%
1600	Training set	82.53%	76.50%
	Test set	63.02%	57.15%



From these experiments, we found that while the accuracy of the training dataset fluctuates around 82%, the accuracy of the test set continues to rise while the number of estimator increases. The trend we observed is that the increase in the number of estimators contributed to an increase in the prediction accuracy for the Testing set and reached its peak of 67% when the number of estimators equals 1500. After the number of estimators hits 1500, increasing the number of estimators seems to negatively contribute to the prediction accuracy. We concluded that there is a trade-off between model performance and computational efficiency. Increasing the number of estimators can make the model slower to train and more memory-intensive. The prime number of estimators is around 1500.

5.5.5 Ablation Analysis

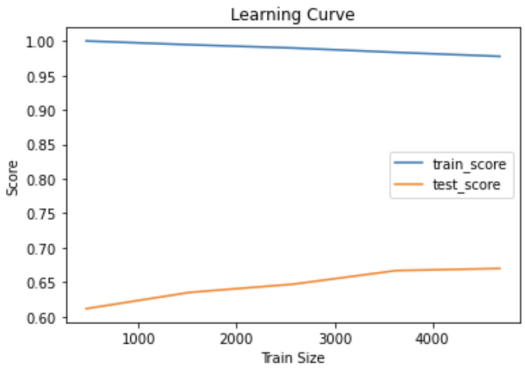
To see which factor is the most important factor for the random forest, we also did an ablation analysis. We first chose to adopt the standard parameters: the maximum depth is 10, the number of estimators is 900, and the minimum sample leaf is 2. As a result, we got an F1 score of 55.17% and the accuracy for the test is 61.59%.

To explore the impact of the number of estimators and minimum samples leaf on the prediction accuracy, we held the maximum depth unchanged and decreased the number of estimators and minimum samples leaf. The resulting F1 score is around 54% and the prediction accuracy is around 60.55%. Next, we kept the number of estimators the same and decreased the maximum depth and minimum sample leaf. As a result, we got an F1 score of around 54% and an accuracy of around 60%. Then we kept the number of estimators the same, while we decreased the maximum depth and minimum sample leaf, then we got the F1 score of around 51% and accuracy of around 60%.

Based on our experiment, the maximum depth and number of estimators have more impact on random forest training and testing. We concluded that this is because the maximum depth of a tree determines the maximum number of nodes that a tree in the forest can have. As a result, the maximum depth of a tree controls the complexity of the model. Thus, setting it to a low value can prevent overfitting, while setting it to a high value can improve the model's ability to fit the training data. Therefore, finding the optimal value for the maximum depth of a tree in a random forest is important for achieving good performance on both the training and test data.

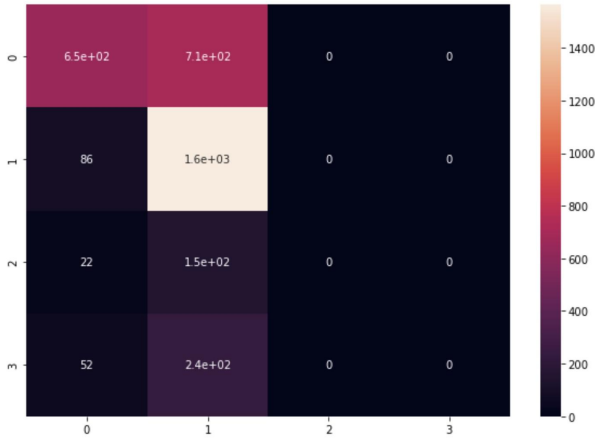
For the number of estimators, it is a hyperparameter that controls the overall complexity and predictive power of the model. Increasing the number of estimators in a random forest typically leads to better performance on the training data, as more trees can model more complex patterns in the data. Finding the optimal number of estimators in a random forest is therefore important for achieving good performance on both the training and test data.

5.5.6 Learning Curve



From this learning curve, we can see that we need a more complex model since even if there is more data, the model has stopped learning. So we need more convolutional neural networks, for example, CNN.

5.5.7 Error Analysis



The error we have first from the imbalanced data. At first, when we got the data, we found out that the four types are not evenly distributed. Though we planned to add the function of balancing data, the data was so unbalanced, one type is the fourth time of another. So if we balanced it, it would reduce by 80%. But we do think that if there is more data in the future, balancing data before can increase the accuracy and F1 score.

6 DISCUSSION AND PRIOR WORK

6.1 Prior Work Revisit

For the previous work, we have done some research about MBTI personality types and some previous analysis between word choice and MBTI personalities. And then we went through some model introduction and model comparison, including KNN, SVM, random forests, and SVM. KNN was the easiest one to implement, that's why we chose to run it the first time. And we found that random forests could avoid overfitting, and since the training set has a big data range, we chose random forests as our final method.

6.2 Takeaways

We implemented and experimented with a variety of machine learning models ranging from KNN and random forest in this project. One important takeaway is that although these models differ fundamentally in terms of algorithm, complexity, and how we train them, their performance on classifying MBTIs is considerably good since there are 16 types at first. Our optimization helped us to find the best number of estimators, minimum sample leaf, and maximum depth for the random forest, and the accuracy has become higher.

7 OUTCOMES AND RESULTS

In this paper, we applied 3 machine learning methods to the same dataset to classify MBTI from Twitter posts. KNN, as our baseline model, has an overfitting issue and thus a lower prediction F1 score.

To tackle the problem of overfitting, we switched to SVM. However, SVM is not suitable for problems with large numbers of features or instances, as the computational complexity of the algorithm grows rapidly with the size of the data set. We then switch to Random Forest as it can handle larger datasets.

So we chose random forest as it is faster to train with a large dataset. In addition, Random Forest is generally easier to use and tune than support vector machines, as they have fewer hyperparameters to adjust and are less sensitive to the choice of parameters. To optimize the performance of the random forest, we chose a different ratio of the training set and testing set, and also set different minimum sample leaves, number estimator numbers, and also the maximum depth. We think it is still possible to increase the performance of random forests. Because when we preprocessed the data, we didn't make sure those data are balanced, and this could be a problem to lower the accuracy and precision. So the improvement could be like finding more data and making data balanced. But we are sure about the effectiveness of using machine learning to identify MBTI.