MBTI Personality Analysis and Prediction

Yutao Zhou
UNI:yz4359
yz4359@columbia.edu

Jiarong Shi
UNI:js6132
js6132@columbia.edu

Qingcheng Yu
UNI:qy2281
qy2281@columbia.edu

Abstract

People behave differently when they are online compared to offline. Therefore, it is hard to draw accurate and stable user profiles. Our team believes personality is a great indicator for predicting user behavior. Our group would use big data analysis methods to predict user personality. More specifically, we are using machine learning and statistical methods to analyze users' tweets to predict their personalities.

In this report, we are first going to give an introduction to our project. Then, we reviewed the related literature presenting how others make use of users' online posts. After that, we introduced the data source we used, including the MBTI datasets and the Twitter streaming data. What's more, we showed our methods from four perspectives: Web application, Data Preprocessing, vectorization, and Machine Learning Model. Following, we presented an overview of our systems. Last, but not least we provided the experiments we have done and concluded our project.

1. Introduction

In today fast growing technology, an exceptional amount of data has been generated every day. This did a great favor for analyzing user behavior. However, it is still very difficult to predict user behavior because of the online disinhibition effect. The online disinhibition effect is used to describe the phenomenon that people behave more expressive and intensely when they are online compared to in-person [12]. As a result, users' behavior would be harder to predict online. Also, users might have different behavior online compared to real life. This is problematic because when doing recommendations to the user, we would love to predict people's reactions when they receive our recommendation.

To address this problem, we are focusing on analyzing users' personalities and this would give us more information to give better recommendations. For example, people with different personalities might prefer advertising in different styles even for the same product. Therefore, it could be economically plausible to use users' personalities as a baseline for user images. Thurs, we propose this MBTI personality analysis and prediction project to present the potential of making use of using social media data.

Our group is using machine learning and statistical methods to analyze big data that users generate in their daily online activity. When a user came to our website we would collect the user's tweets and use our pre-trained model to predict the user's personality. Then, we are going to show the corresponding prediction results including introduction, related jobs, similar people, and word cloud. We have compared 6 models and decided to use Logistic Regression as our model. After training the model with the MBTI dataset, our precision, recall, and f1-score achieved 0.94.

We have developed a user-friendly online application that would catch user tweets and make predictions. Also, we paid great attention to visualizing the prediction result in intriguing ways to make it attractive. For example, we have presented jobs that people with their personalities would usually take. Further, we showed pictures of well know celebrities with the corresponding personality type. Last, we included a word cloud that is formed with social media posts from people with the same personality as the user. The web application is deployed to GCP and we plan to host it there for a few months. Therefore, users who are interested could take a look and try out our fully functional project. The link to our web application is here http://34.23.131.207:5000/

2. Related work

This section includes three parts. Firstly, we discuss the differences between the two main personality indicators, which are the five-factor model(FFM) and Myers Briggs Type Indicator(MBTI), and why we choose MBTI as an indicator in our prediction system. Secondly, personality prediction involves different kinds of datasets, we will have a deep look into them. In addition, several feature extraction methods are listed to compare their pros and cons between them. Lastly, many different classification models vary from traditional machine learning methods to deep learning methods that have been used for personality prediction. We will analyze each of them.

Energy	**E**xtrovert	**I**ntrovert
Information	**S**ensing	I**N**tuition
Decision	**T**hinking	**F**eeling
Lifestyle	**J**udging	**P**erceiving

Table 1. 16 types of MBTI.

2.1. Different predictors of personality

The Five-Factor Model, also called the Big Five, is a personality testing framework that measures openness, conscientiousness, extroversion, agreeableness, and neuroticism. The five-factor model was developed in the 80s and 90s and is largely based on lexical assumptions, which suggest that over time, basic features of human personality have been encoded into language. This hypothesis on the other hand proves the feasibility of predicting personality based on texts.

Despite all its successes, the five-factor model has been heavily criticized by many scholars. One problem involves the lack of a comprehensive theory. The lexical hypothesis, while interesting and reasonable, has been considered by some scholars to be too narrow to be a theory of personality[4].

Different from FFM which is based on lexical hypotheses or traits, MBTI is based on types, which include 16 types and 4 letter abbreviations. The first letter relates to where the individual mostly derives their energy, the second letter relates to an individual's perceptual function, the third letter is a personal judgment function, and the last letter determines how an individual relates to the outside world. They are shown in Table 1.

Many researchers prefer to use MBTI in their work instead of FFM. For instance, in a recent study by Yoon et al.[13], they found that MBTI preferences play a significant role in online purchase decisions, while FFM had limited impact.

2.2. MBTI datasets in online social platforms

The choice of the dataset can significantly affect the accuracy of the model. The most popular MBTI datasets are Twitter users datasets1, the MBTI dataset on the Kaggle website2, the Corpus of Reddit comments and posts labeled with MBTI personality types MBTI19k dataset3, and 8k MBTI dataset from Personality Cafe4. Many related works about personality prediction used these datasets, and they mostly achieved positive results. However, Lukito et al.[8] have trained the Naive Bayes model on the Twitter dataset in Bahasa Indonesian language. Although the algorithm used is efficient but due to the dataset drawback, the result is bad.

2.3. Feature extraction methods used in MBTI prediction

Multiple NLP methods are used when dealing with text data. Firstly we should extract features from the text and convert the human language text into computer-understandable vectors or matrices.

Latent Dirichlet Allocation (LDA) is one of the feature dimension reduction methods, where each document can be described by a distribution of topics and each topic can be described by a distribution of words. When the sample classification information of LDA relies on variance instead of mean, the dimensionality reduction effect is not good and the data may be overfitted. Gjurkoviic et al.[3] used an LDA model to deduce topic distribution from user comments, but found that the LDA was not very good at predicting MBTI.

The bag of words is a simple feature extraction method, which detects whether a document contains certain words in a dictionary. The bag of words model method is simple, fast, and sometimes for short text processing has a good effect. However, it ignores the syntax and semantic information in the text, resulting in the loss of important information.

TF-IDF, TF stands for Term Frequency, and IDF refers to Inverse Document Frequency. The TF-IDF model can estimate the importance of a word to a document in a document set. The more times a word appears in a document, the more important the word is to the document, and the more frequently it appears in the document set, the less important the word is. Also in Gjurkoviic et al.[3], the best model in the research uses TF-IDF weighted n-grams over logistic regression. However, same as the bag of words model, TF-IDF ignores the relationship between each word in the document and fails to take into account important information such as word order, semantics and syntax. Mikolov[1] et al. proposed CBOW(Continuous Bag-of Words) model and the Skip-gram model respectively. The CBOW model is based on the content of the context to infer the possible probability of the central word, while the Skip-gram model is based on the known central word to infer the content of the context. The way of word embedding considers the relationship between words in the sentence and expands the word vector to higher dimensions, which has better generalization ability.

2.4. Classification methods used in MBTI prediction

Over the years many different models have been attempted to predict personalities. Logistic regression is one of the most commonly used machine learning algorithms, which generally works with binary classes. Plank et al.[9] used the LR model on the Twitter dataset to classify MBTI personality types. Raje et al.[10] also used the LR classification model to predict personality types and found that it was better than several neural network models.

Naive Bayes models assume that feature conditions are independent of each other and classify them based on Bayes' theorem, resulting in a low error rate and high computational efficiency. Moreover, the algorithm itself is simple and easy to understand and has been widely used in the field of text classification. However, Rennie et al.[11] stated that multinomial NB could not model text data well because the text is not generated according to a multinomial model.

KNN, NN means the nearest neighbor, and k is referred to the number of neighbors to be considered. KNN is a classification algorithm that has high accuracy and stability, but it will lead to error in the face of sample imbalance. Li et al.[6] used this classifier on MBTI prediction and they aimed to compare and find the best distance computation approach for KNN in MBTI personality prediction.

The goal of a support vector machine (SVM) is to find a suitable linear separator between different categories. In SVM, the linear indivisible problem is transformed into a higher dimensional space using the nuclear technique. Common kernels are radial basis function, polynomial kernel, and Gaussian function. Bharadwaj et al.[2] compared SVM with NB and Neural Net Model and found that SVM outperformed the others in MBTI prediction.

A random forest is a classifier containing multiple decision trees, and its output category is determined by the mode of the categories output by individual trees. In Lima et al.[7] the study, random forest outperformed SVM and NB across multiple feature sets in MBTI prediction.

Pre-trained language models are a new but useful method for model training. For example, Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. Kel et al.[5] used BERT on the personality cafe dataset on Kaggle and obtained a better result.

3. Data

Datasets are essential for model training because they must be carefully curated to ensure that they are representative of the problem being solved and contain enough data to accurately train the model. Generally, to fully understand and see the whole picture of one dataset, we always use a 3Vs model. The 3Vs of data are Volume, Variety, and Velocity. Volume refers to the amount of data that is available, Variety refers to the different types of data that are available, and Velocity refers to the speed at which data is generated and processed. Our project has two parts, firstly we tend to train a machine learning model, then we crawl users' tweets to do MBTI personality type prediction. So our project involves two kinds of data, which is the MBTI dataset in Kaggle and Twitter streaming data. In the following, we will briefly introduce the reason we choose the dataset and discuss the 3V dimensions of these two data separately.

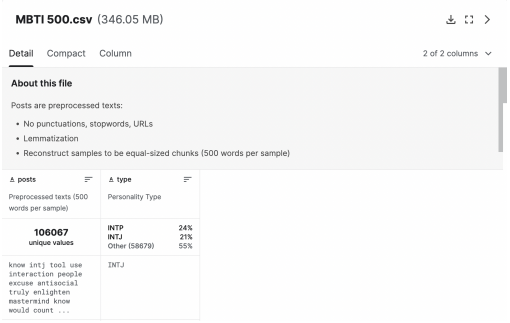


Figure 1. A quick look at the MBTI datasets.

3.1. MBTI datasets

To train a machine learning model which takes texts from social platforms as input and MBTI personality type as output, we need a kind of data that is extracted from social media platforms and has corresponding personality type labels. And this MBTI Personality Types 500 Dataset from Kaggle is quite a fit.

The volume dimension of this dataset is 346.05MB large and has about 106k preprocessed records of posts and personality types. For the variety dimension of the dataset, there are two columns of the data, and the data are extracted from two sources. As we can see in Figure 1, The post data are preprocessed texts which have no punctuations, stopwords, or URLs, have finished lemmatization, and have been reconstructed into equal-sized chunks (500 words per sample). The second column is a personality type, which has 16 unique values. Most parts of the data are collected from Reddit using Google big query by Dylan Storey, the remaining parts are collected from the PersonalityCafe forum, where each record has the last 50 posts written by the corresponding user. The velocity dimension of the dataset is an existing dataset that has stopped updating. To remedy this, in the next part we use Twitter streaming data which will change over time, and in the future, we could collect data from the web application we built and extend the existing dataset.

We can also take a quick look at the data distribution. From figure 2 the number of samples for 16 types of personality in the original dataset, we can see that this is an imbalanced dataset. The most numerous personality type is INTP, which makes up about a quarter of the data and it is unreal in the natural world. To fix this problem, in the later part of the article, we described how we use downsampling or upsampling methods for data class balance and achieve better accuracy. Fig X shows the distribution across types of indicators, and also can find the imbalance problem, while in other words the imbalanced data also shows

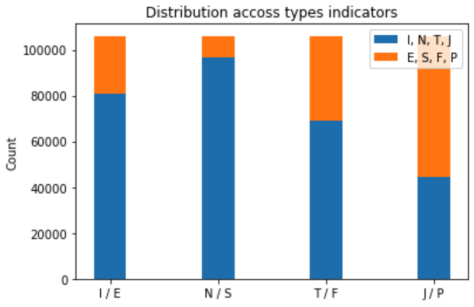
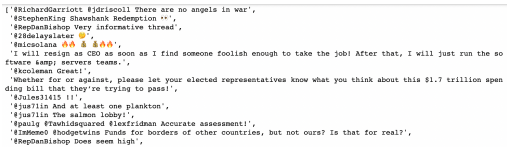


Figure 2. Distribution across types indicators.



```
['@RichardGarriott @jdriscoll There are no angels in war',
 '@StephenKing Shawshank Redemption ^^',
 '@RepDanBishop Very informative thread ',
 '@20delayalater 😉',
 '@micsolana 🔥🔥 👌 &🔥🔥',
 'I will resign as CEO as soon as I find someone foolish enough to take the job! After that, I will just run the so
ftware &amp; servers teams.',
 '@xcoleman Great!',
 'Whether for or against, please let your elected representatives know what you think about this $1.7 trillion spen
ding bill that they're trying to pass!',
 '@JulesJ1415 !!',
 '@jus71in And at least one plankton',
 '@jus71in the salmon lobby!',
 '@paulg @Tawhidsquared @lexfridman Accurate assessment!',
 '@ImMemeO @hodgetwins Funds for borders of other countries, but not ours? Is that for real?',
 '@RepDanBishop Does seem high',
```

Figure 3. Raw streaming data from Tweeter.

what categories of people are more inclined to use these social networking sites.

3.2. Twitter streaming data

Tweet streaming data is data that is collected from Twitter in real-time. This data can be used to analyze trends, sentiment, and other insights about what people are saying on the platform. To predict the user's current personality type, we use the most recent tweets of a user. If a user does not have enough tweets to meet the minimum text length requirement of our model, we just copy his or her tweets to get 500 words. Actually, the number or length of tweets has an effect on prediction accuracy.

We collect tweets using functions in a python package called tweepy. Firstly we apply for a consumer key, a consumer secret, an access token, and an access token secret to get an authorized tweepy client, then we use the get_user() function to convert the user name which the user entered into our system into the user id. After that, we use the get_users_tweets() function and user id to get the user's tweet. Except for the id field, there are other parameters that we can modify in this function. For example, tweets to try and retrieve, up to a maximum of 100 per distinct request. If one user's most recent tweets are all too short to meet the 500 words length requirement, we can change the start_time field to request tweets in older times. A picture of the response to the request is shown in Figure 3, as we can see there are some tag fields, emojis, and URLs in the original tweets, so the data preprocessing is of great importance.

4. Methods

4.1. Web Application

We built a web application to demonstrate the result and interact with people, and we used a number of methods to implement this. A lightweight web application framework called Flask is the main architecture for a website. We use python to write the backend code and develop route API to handle requests from the front. In the front end, we use HTML, CSS, and JavaScript to build webpages.

4.2. Data Preprocessing

The methods we used in this project are divided into three sections depending on the purpose. First of all, data preprocessing and feature extraction are necessary for Twitter streaming data. In order to fit the format of the MBTI dataset, we need to filter punctuation, stopwords, and emoji in the original tweets. Then implement lemmatization to convert tweets to the normal format (no state of affairs).

Moreover, the samples in the dataset distribute unbalanced, and some of the features only contain a few records. Therefore, we use oversampling and undersampling to reconstruct the dataset to make it balanced. There are two oversampling techniques are implemented in our project, random oversampling and Synthetic Minority Oversampling Technique (SMOTE). Random oversampling refers to duplicating the samples from the minority class. Therefore, it may increase the likelihood of occurring overfitting, since it makes exact copies of the minority class examples. The defect of random oversampling is it does not give variation to the ML model and only has limited performance improvement. SMOTE method is an improvement in duplicating examples from the minority class. It synthesizes new examples according to the feature space. SMOTE creates larger and less specific decision boundaries that increase the generalization capabilities of classifiers, therefore increasing their performance. Because SMOTE creates new samples instead of simpling duplicates of similar samples. it will lead to a better prediction model than random oversampling. Therefore, we finally choose SMOTE as our oversampling method.

4.3. Vectorization

Because we do text classification, the input of the text document should convert into a numerical representation that can be adapted to machine algorithms to make predictions. The method we use is TfidfVectorizer. Tokens generated by this method are followed by the matrix of TF-IDF features. TfidfVectorizer has regulations that can adjacent the weight of tokens that occur frequently and tokens that occur rarely. Therefore, it can extract primary features better than a normal vectorizer.

4.4. Machine Learning Model

Moreover, the samples in the dataset distribute unbalanced, and some of the features only contain a few records. Therefore, we use oversampling and undersampling to reconstruct the dataset to make it balanced. There are two oversampling techniques are implemented in our project, random oversampling and Synthetic Minority Oversampling Technique (SMOTE). Random oversampling refers to duplicating the samples from the minority class. Therefore, it may increase the likelihood of occurring overfitting, since it makes exact copies of the minority class examples. The defect of random oversampling is it does not give variation to the ML model and only has limited performance improvement. SMOTE method is an improvement in duplicating examples from the minority class. It synthesizes new examples according to the feature space. SMOTE creates larger and less specific decision boundaries that increase the generalization capabilities of classifiers, therefore increasing their performance. Because SMOTE creates new samples instead of simpling duplicates of similar samples. it will lead to a better prediction model than random oversampling. Therefore, we finally choose SMOTE as our oversampling method.

In order to improve the accuracy of prediction, we try different classification models such as Logistic Regression, LinearSVC, Naive Bayes, Random Forest, and KNN. The reason we chose these classifiers is that our project is a multi-class classification. There is a total of 16 classes corresponding to each type of classification.

We also try to use the poly Kernel SVC method to train the model but it does not work. The reason we consider this ploy Kernel has excessive complexity when dealing with a mass of data. Moreover, we try to use PySpark to train our model but the results get from PySpark using the same method are different from the result using python only. In Figure 4, we show the accuracy of the logistic regression model in PySpark is approximately 0.91. However, according to Table 2, the best result for logistic regression is 0.94. The reason may be the algorithm for PySpark has a litter difference. Finally, we decided to choose the higher accuracy method and not use PySpark.

5. System Overview

Our software mainly consists of three parts in terms of functionality: Website, Tweet fetching, and Machine learning models. From a web development standpoint, our project has a front end that is written in CSS and HTML and a back end that is written in Python with Flask.

Figure 5 is a system flow chart for our systems. The dataset comes from Kaggle and is a trained NLP model on the Google Cloud Platform (GCP) virtual machine, and it is part of Web application development. The Web application

Figure 4. Logistic regression model results come from PySpark

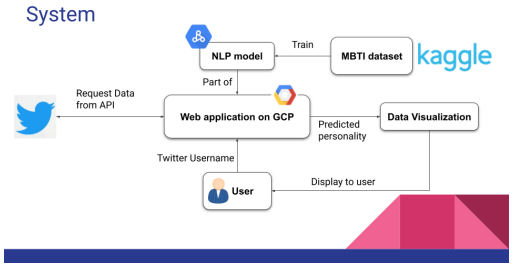


Figure 5. A flow chart for our system.

Figure 6. Landing page of the web application.

requests the streaming data from API provided by Twitter, and finally, it will display the visualization figures and predict results to the User. The User should enter their Twitter User Id to get these results.

For our website, We formulate the structure of our website with Python. More specifically, we used a Python package called Flask as a backend that connects and render each HTML template. The entire backend functionality is happening in the 'app.py'. When the user came to our website they will first see the landing page that is rendered from 'index.html'. The landing page is shown in figure 6.

The user would enter their user name in the search bar located at the center to predict their personality. At the moment that the user pushed the 'enter' button, the backend would use that username to fetch the nearest 100 tweets from the Twitter API by using the tweepy python packet. After getting all of the tweets we are going to lower all letters, strip out unnecessary space, and split out each word by

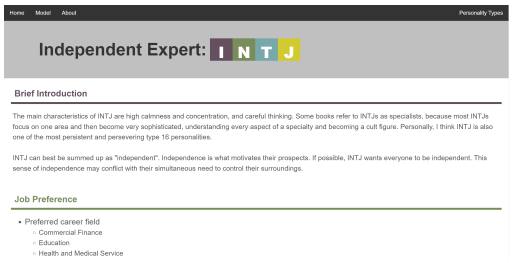


Figure 7. INTJ result.

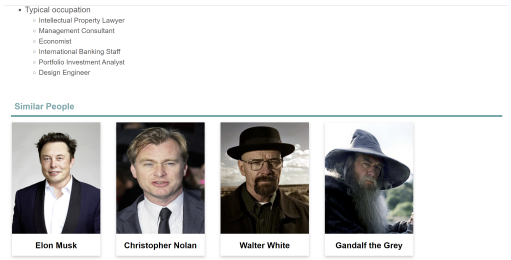


Figure 8. INTJ result continues.

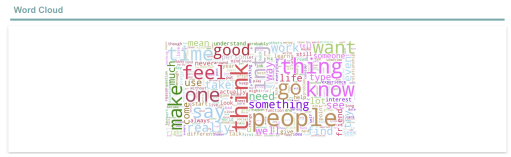


Figure 9. INTJ result continues.

space. Now, we would use a regular expression to filter out non-English letters like emojis and other non-textual content. Also, each word is matched with our stopwords dictionary, which I would explain later, to filter out stopwords. After we cleaned the data, we need to make sure the input to the machine-learning model is exactly 500 words. Therefore, we would use list concatenation to copy the data if we do not have enough words and crop it to make sure there are 500 words. The runtime could be improved by some easy tricks in iteration. But, we decided to use the easy way since our website is more than fast enough. Then the saved machine learning model would be used to make the prediction. After that, we would match the prediction result and use load the corresponding template. This part could be improved with a radix tree and some basic string-matching algorithms. But, we only have 16 possible results each with a short name. Therefore, we abandoned to do so. If the username does not exist, we could not fetch corresponding tweets. On such occasions, the user will be redirected to the landing page to retry. An example result for the INTJ personality type is shown in figure 7-9.

We have also developed an about page that gives a brief introduction to Why people should care about their MBTI.

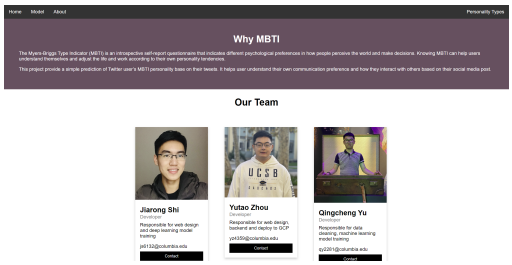


Figure 10. A screen shot of the About page of our web application.

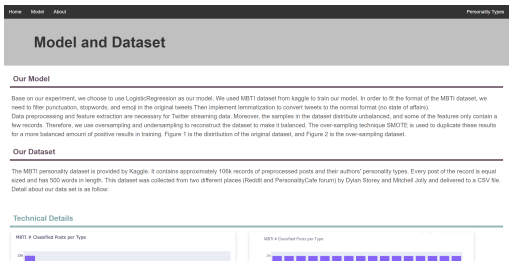


Figure 11. A screen shot of the Model page of our web application.

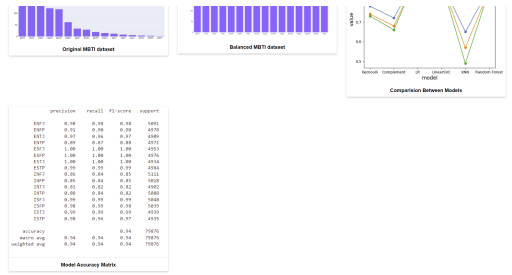


Figure 12. A screen shot of the Model page continue of our web application.

Also, our roles and pictures are presented here. The about page is shown in figure 10. What's more we have also included a model page that introduces our machine learning model and gives information about the dataset we used. A screenshot of our model page is shown in Figure 11-12.

Let me briefly explain the structure of our code to give you an overview of our system. All of the codes for our web application are in the 'Web_Application' folder. For simplicity, from now on I would refer to the 'Web_Application' folder as the 'root directory' in our system. Under the root directory, there are three folders and one python file. The Python file 'app.py' is the main python script that is going to connect different HTML pages together, fetch user data, and use the model to predict the user's personality. The three folders are 'model', 'static', and 'templates'. The 'model' folder contains all of the files that are necessary to predict the personality of the user. The 'static' folder contains all of the images on our websites and the CSS style sheet used to

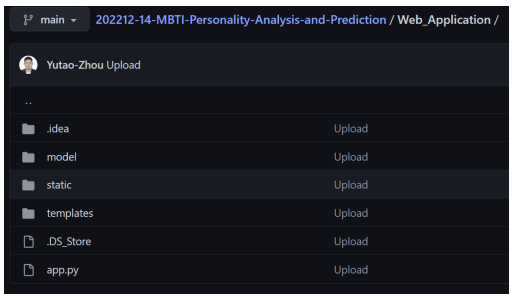


Figure 13. Github directory.

render our web pages. The 'templates' folder contains all of the HTML files which is used to build all of the content of our page. A screen shot of our 'Web_Application' directory is in figure 13.

Now let's dig into our machine learning models. In the folder 'model', there are three files. The first one is 'lm.sav', this is the saved machine learning model that we are going to use to predict the user's personality. The second one is 'stopwords.txt' which is a text file that contains all of the stopwords we used to filter the tweet that we fetched from the Twitter API. The third one is 'vec_tfidf.pkl', this is a pickle file that is used for tokenizing words. This is necessary because we need to convert words into vectors for training.

The machine learning model was trained with a Jupyter notebook named 'Logistic Regression.ipynb'. It was trained on the balanced over-sampling MBTI Personality 500 dataset. After the training, it is saved for our systems to use. A flow chart for our system is shown in figure X.

These are the continent of our systems. Now, I would like to present the deployment of our systems. Our system is written with the Flask packet. The deployed environment is greatly simplified. We created a Google Cloud Platform(GCP) compute engine virtual machine. The GitHub was cloned there and the needed packet was installed. The project would be hose on the VM and as long as we don't shut off the VM it is going to have a fixed public IP. One thing that is worth mentioning is the firewall rule has to be set up to allow all traffic coming into our VM. We have set up the firewall to match '0.0.0.0/0'. This would match all traffic because this prefix has a length of 0. A detailed view of the firewall setting on GCP is shown below in figure 14.

Figure 15-17 shown all of the packet We have used for our system.

Our systems could mainly be improved on the machine learning system design aspect. We are currently using a static pre-saved model. This could be improved by saving fetched user tweets. Then we could let the user input the personality they believe they belong to. Thurs, we have created additional data that could be used to retrain and improve the model. Where the text data are the content of the processed tweets and the label is the personality that the user selected. We could then interest this entry into an SQL database. Then we could use Airflow to schedule and retrain the model with the new dataset once a week or once

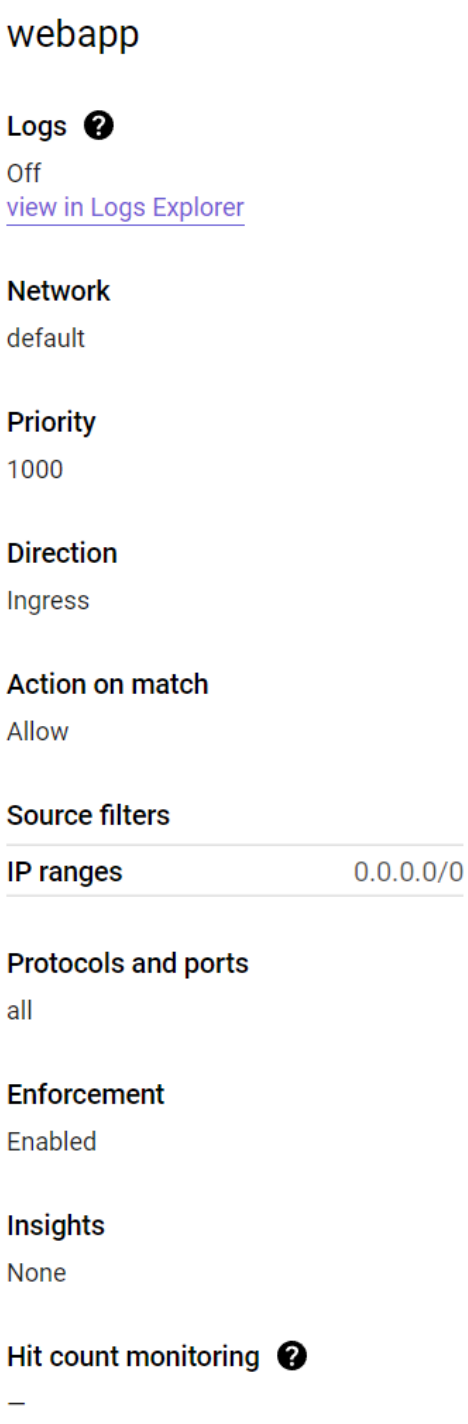


Figure 14. A screenshot from GCP of our firewall rules.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
import pickle
```

Figure 15. Packets we used in machine learning model training.

```
from sklearn.metrics import classification_report
```

Figure 16. Packets we used in machine learning model training.

```
from flask import Flask, request, render_template
import tweepy
import re
import pickle
from nltk.stem.snowball import SnowballStemmer
import warnings
```

Figure 17. Packets we used in web application.

a month. More specifically, we could put the entire dataset into an SQL and add an additional entry to the database for every user that entered their user name and used our website. Every week we could let the scheduled Airflow program fetch the SQL database and use the entire larger database for training. As a result, as more and more users use our web application our model will have better accuracy. We have come up with this system in our group discussion. But, we did not implement this because of our busy schedule.

6. Experiments

We will focus on personality prediction using text data or more precisely social media text data. Therefore two main parts of experiments are performed depending on the purpose. First of all, data preprocessing and feature extraction are necessary for Twitter streaming data. In order to fit the format of the MBTI dataset, we filter punctuation, stopwords, and emoji in the original tweets. Then implement lemmatization to convert tweets to the normal format (no state of affairs). Because the text size in the original dataset is 500 words, it is necessary to check the length of the maximum number of text gained from API at once. When text has less than 500 words, tweet data will be recaptured again at another time period to satisfy the constraint. On the other hand, if the text has larger than 500 words, we will intercept the first 500 words from the text. The example is shown below in Figure 18 and Figure 19.

Moreover, the dataset is do oversampling to solve the problem that the amount of data collected is insufficient. The distribution of samples in the dataset is imbalanced because some of the features only contain a few records.

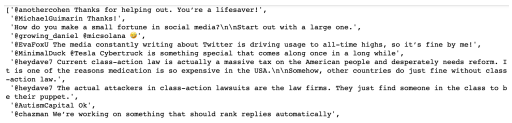


Figure 18. Streaming data from Twitter API before preprocessing.

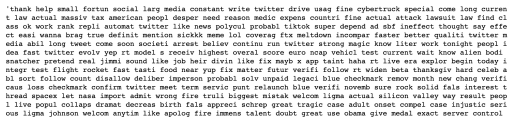


Figure 19. Streaming data from Twitter API after preprocessing.

Aim at unbalanced data sets may lead to severe bias against larger classes, while classes with fewer data points are considered noise and ignored. Therefore, we use oversampling to reconstruct the dataset to make it balanced. We compare the performance of random oversampling and SMOTE. Finally, we choose to use SMOTE because it provides better prediction result. Figure 20 and Figure 21 show the number of samples in each class before and after SMOTE oversampling. In Figure 21, each class has an equal data size.

The second part of our experiments performed is training machine learning models of text classification to make predictions. As we mentioned before, we used five different classifiers to train our model (Naive Bayes, LR, LinearSVC, k-NN, and RF). For the Naive Bayes classifier, the data associated with each feature is assumed to be distributed in a designed distribution. Therefore, we decided to use different sample distributions for training models. Because there are 16 types of personalities, the multi-classes classifier is needed. Finally, we use the Gaussian Naive Bayes classifier and Complement Naive Bayes. For all these six methods, we separate the sample set into two parts: the training set and the test set. The training set contains 80% of the samples, and the test set contains 20% of the samples. Firstly, we used the original dataset which is imbalanced to establish models, the result is not good enough. Table 2 shows the detailed information of each model using the untreated dataset. There are three main parameters to determine the performance and adaption of models which are precision, recall, and f1-score. The precision is the ratio of the number of true positives and the total number of samples, it measures the amount of variance and uncertainties of the data not explained by the fitted values of the model. The recall suggests the proportion of true positives relative to the sum of true positives and false negatives. The f1-score can be interpreted as a weighted harmonic mean of the precision and recall. Table 3 shows the detailed information of each model using a balanced dataset after SMOTE.

Comparing Table 1 and Table 2, it is obviously observed that models training after oversampling has better performance than models training without oversampling. Figure

Model	Precision	Recall	f1-score
Naive Bayes - BernoulliNB	**0.21**	0.17	0.16
Naive Bayes - ComplementNB	**0.26**	**0.29**	0.23
Logistic Regression	**0.83**	**0.65**	0.71
Linear SVC	0.81	0.69	0.74
KNN	0.55	0.47	0.60
Random Forest	0.65	0.77	0.80

Table 2. Accuracy for different models trained on Original dataset

Model	Precision	Recall	f1-score
Naive Bayes - BernoulliNB	**0.78**	0.74	0.73
Naive Bayes - ComplementNB	**0.72**	**0.68**	0.66
Logistic Regression	**0.94**	**0.94**	0.94
Linear SVC	0.90	0.90	0.90
KNN	0.65	0.57	0.49
Random Forest	0.87	0.86	0.86

Table 3. Accuracy for different models trained on over-sampling dataset

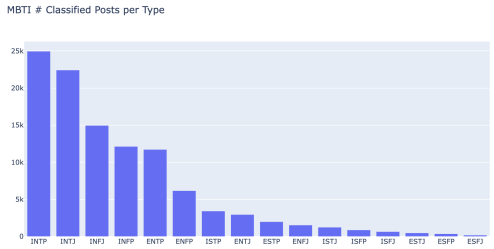


Figure 20. Original MBTI dataset.

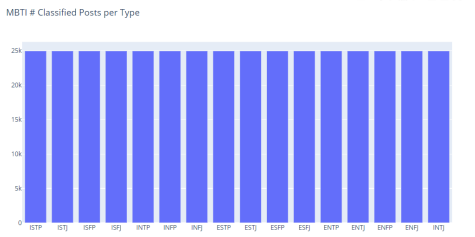


Figure 21. Over-sampling MBTI Dataset using SMOTE method.

22 is a line chart showing the difference between three parameters between six different models. According to Figure 22, the Logistic Regression classifier is the best model we have because it has the highest value of precision, recall, and f1-score. Figure 23 is the classification report for our best model. Figure 23 figures out that the precisions are greater than 0.95 for most classes, and the lowest value is higher than 0.8.

7. Conclusion

In this project, we build an MBTI personality prediction system, which takes users' Twitter usernames as input and

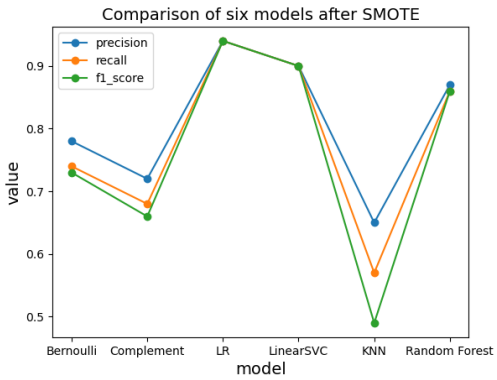


Figure 22. Original MBTI dataset.

	precision	recall	f1-score	support
ENFJ	0.98	0.98	0.98	5091
ENFP	0.91	0.90	0.90	4978
ENTJ	0.97	0.96	0.97	4909
ENTP	0.89	0.87	0.88	4971
ESFJ	1.00	1.00	1.00	4953
ESFP	1.00	1.00	1.00	4976
ESTJ	1.00	1.00	1.00	4934
ESTP	0.99	0.99	0.99	4984
INFJ	0.86	0.84	0.85	5111
INFP	0.85	0.84	0.85	5018
INTJ	0.81	0.82	0.82	4902
INTP	0.80	0.84	0.82	5088
ISFJ	0.99	0.99	0.99	5048
ISFP	0.98	0.99	0.98	5039
ISTJ	0.99	0.99	0.99	4939
ISTP	0.98	0.96	0.97	4935
accuracy			0.94	79876
macro avg	0.94	0.94	0.94	79876
weighted avg	0.94	0.94	0.94	79876

Figure 23. Detailed result for logistic regression with the over-sampling dataset. The overall precision of 0.94

returns personality type results based on tweets. To achieve accurate prediction, we preprocess the Twitter streaming data crawled by Twitter API. Then upsampling and down-sampling are performed on the original MBTI dataset. For classification models, we have tried Naive Bayes, LR, LinearSVC, k-NN, and RF algorithms, compared precision, recall, and f1-score parameters between them and it shows that Logistic Regression has the best performance. At last, we develop a user-friendly web application using python and Flask as the backend, and JavaScript, HTML, and CSS as the front end. We integrate the best prediction model into the web and a user could easily get his or her recent MBTI personality type prediction by just entering the Twitter username. Moreover, they could have more information about their personalities like job preferences and similar people with the same personality type. We have deployed the whole system to GCP and we welcome you to visit our website and play around.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. 2

[2] Srilakshmi Bharadwaj, Srinidhi Sridhar, Rahul Choudhary, and Ramamoorthy Srinath. Persona traits identification based on myers-briggs type indicator (mbti)-a text classification approach. In *2018 international conference on advances in computing, communications and informatics (ICACCI)*, pages 1076–1082. IEEE, 2018. 3

[3] Matej Gjurković and Jan Šnajder. Reddit: A gold mine for personality prediction. In *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*, pages 87–97, 2018. 2

[4] Lewis R Goldberg. The structure of phenotypic personality traits. *American psychologist*, 48(1):26, 1993. 2

[5] Sedrick Scott Keh, I Cheng, et al. Myers-briggs personality classification and personality-specific language generation using pre-trained language models. *arXiv preprint arXiv:1907.06333*, 2019. 3

[6] Charles Li, Monte Hancock, Ben Bowles, Olivia Hancock, Lesley Perg, Payton Brown, Asher Burrell, Gianella Frank, Frankie Stiers, Shana Marshall, et al. Feature extraction from social media posts for psychometric typing of participants. In *International Conference on Augmented Cognition*, pages 267–286. Springer, 2018. 3

[7] Ana Carolina ES Lima and Leandro Nunes de Castro. Tecla: A temperament and psychological type prediction framework from twitter data. *Plos one*, 14(3):e0212844, 2019. 3

[8] Louis Christy Lukito, Alva Erwin, James Purnama, and Wulan Danoekoesoemo. Social media user personality classification using computational linguistic. In *2016 8th international conference on information technology and electrical engineering (ICITEE)*, pages 1–6. IEEE, 2016. 2

[9] Barbara Plank and Dirk Hovy. Personality traits on twitter—or—how to get 1,500 personality tests in a week. In *Proceedings of the 6th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 92–98, 2015. 2

[10] Mehul Smriti Raje and Aakarsh Singh. Personality detection by analysis of twitter profiles. In *International Conference on Soft Computing and Pattern Recognition*, pages 667–675. Springer, 2016. 2

[11] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003. 3

[12] John Suler. The online disinhibition effect. *Cyberpsychology & behavior*, 7(3):321–326, 2004. 1

[13] Myeong-Yeon Yi, O Lee, Jason J Jung, et al. Mbti-based collaborative recommendation system: a case study of webtoon contents. In *ICCASA*, pages 101–110. Springer, 2015. 2

8. Contribution

For all of the presentations PPT, we split equally by the number of slides. For the progress report, we split equally be sections. Jiarong Shi and Yutao Zhou both wrote the CSS and wrote the 'app.py' and other corresponding files that are related to web development. Qingcheng Yu and Jiarong Shi both wrote codes for extracting tweets from Twitter API.

8.1. Yutao Zhou

Wrote 10 MBTI personality pages. Wrote model page. Deployed the web application to GCP. Wrote the Abstract, Introduction, and System Overview part of the final report. Wrote Latex to transform content teammates write on Google docs to Latex for submission. Responsible for editing and uploading Youtube videos.

8.2. Jiarong Shi

Wrote 6 MBTI personality pages. Wrote about page. Wrote Home page. Wrote website structure. Wrote the Data, Related Work, Method(Web Application), and Conclusion part of the final report. Trained Neural Network model.

8.3. Qingcheng Yu

Implemented dataset to a balanced state. Trained model use KNN, Logistic Regression, Random Forest, Naive Bayes, and LinearSVC. Used Pyspark to build a model and compare the results with Python. Wrote Method (Data Preprocessing, Vectorizer, and Machine Learning Model) and Experiment part of the final report.