PREDICTING PERSONALITY TYPE FROM WRITING STYLE

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Tanay Gottigundala

December 2020

COMMITTEE MEMBERSHIP

TITLE:	Predicting Personality Type from Writing Style
AUTHOR:	Tanay Gottigundala
DATE SUBMITTED:	December 2020
COMMITTEE CHAIR:	Franz Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Bruno da Silva, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Carrie Langner, Ph.D. Professor of Psychology

ABSTRACT

Predicting Personality Type from Writing Style

Tanay Gottigundala

The study of personality types gained traction in the early 20th century, when Carl Jung's theory of psychological types attempted to categorize individual differences into the first modern personality typology. Iterating on Jung's theories, the Myers-Briggs Type Indicator (MBTI) tried to categorize each individual into one of sixteen types, with the theory that an individual's personality type manifests in virtually all aspects of their life. This study explores the relationship between an individual's MBTI type and various aspects of their writing style. Using a MBTI-labeled dataset of user posts on a personality forum, three ensemble classifiers were created to predict a user's personality type from their posts with the goal of outperforming existing research as well as outperforming the test-retest reliability of online questionnaire-based personality assessments. With the increasing amount of textual data available today, the creation of an accurate text-based personality classifier would allow for user experience designers and psychologists to better tailor their services for their users.

ACKNOWLEDGMENTS

Thanks to:

- I would like to express my gratitude to Dr. Franz Kurfess, my advisor, for his assistance and support at every stage of this process.
- I would like to extend my thanks to Dr. Carrie Langer and Dr. Bruno da Silva for their insightful comments and suggestions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Attempting to describe and explain individual differences in behavior and thoughts has been a highly researched topic in personality psychology in the past century. In 1921, Carl Jung published *Psychologische Typen*, in which he theorized that people could be categorized using a personality typology. In his original theory of psychological types, he proposed three separate dimensions: extraversion vs introversion, sensation vs intuition, and thinking vs feeling [33]. Jung's early theories were iterated upon by Katharine Cook Briggs and Isabel Briggs Myers with the creation of the Myers-Briggs Type Indicator (MBTI). The MBTI has served as an instrument that people have used in attempts to better understand their own beliefs and motivations. While the reliability and validity of the MBTI has often drawn criticism, it remains the most widely used personality measure.

Our personality manifests in all aspects of our life, and it affects the way we perceive and interact with the world. This study specifically focuses on the predictive strength of an individual's MBTI type based on their posts and comments from an online personality forum. Existing research in this area has shown that there is a correlation between between an individual's writing and their personality type, but they have not yet managed to outperform the test-retest error of the MBTI, which stands at 38.82% [15]. This represents the rate at which an individual received identical personality classifications when they took the test twice with a 5-7 month break between the attempts.

With the increasingly abundant amount of textual data available today [42], classifying an individual's personality type can become a more accurate tool than the standard questionnaires used in personality classification today. More accessible and accurate personality classifiers serve as a useful tool for user experience designers by allowing for better understanding of their users and customized experiences based on personality type. It is also important for consumers to understand what assumptions can be made based on their social media presence, with targeted ads becoming more intrusive and the advertising-based business model gaining popularity. The MBTI also serves as a tool that can help facilitate introspection for individuals, helping them better understand their thoughts, behaviors, and interpersonal relationships. This study explores a machine learning approach to classifying an individual's MBTI type based on their writing style.

Chapter 2

BACKGROUND AND RELATED WORK

2.1 Myers-Briggs Personality Type Indicator

The Myers-Briggs Type Indicator (MBTI) is a widely used personality classification tool that attempts to identify behavioral and cognitive patterns in people. The theory that people's seemingly unpredictable variation in behavior and thought patterns can actually be interpreted as consistent and orderly became increasingly popular in the early and mid 1900s. This idea was pioneered by Carl Jung, a renowned psychologist and student of Sigmund Freud. In his book, Psychological Types [33], Jung designed a psychological typology that he believed could reduce people's personalities into a set of classifications, which he believed was key to understanding other people and their motivations. Using this as the backbone of their research, Isabel Briggs Myers and Katharine Briggs iterated on Jung's original theory, formulating a set of hypotheses, which were then used to create the four dichotomies that are currently viewed as the main categories for the test [18]. This allowed for a new wave of personality research since it became easier to study personality in a quantitative manner.

2.1.1 Testing Categories

The MBTI is split up into four main categories: extraversion vs introversion, sensation vs intuition, thinking vs feeling, and judging vs perceiving. These four dichotomies are described in further detail in Figure 2.1. The combination of these four binary classifications gives us sixteen total possible personality types. Each personality type has distinct characteristics and the combinations of multiple of these categories have

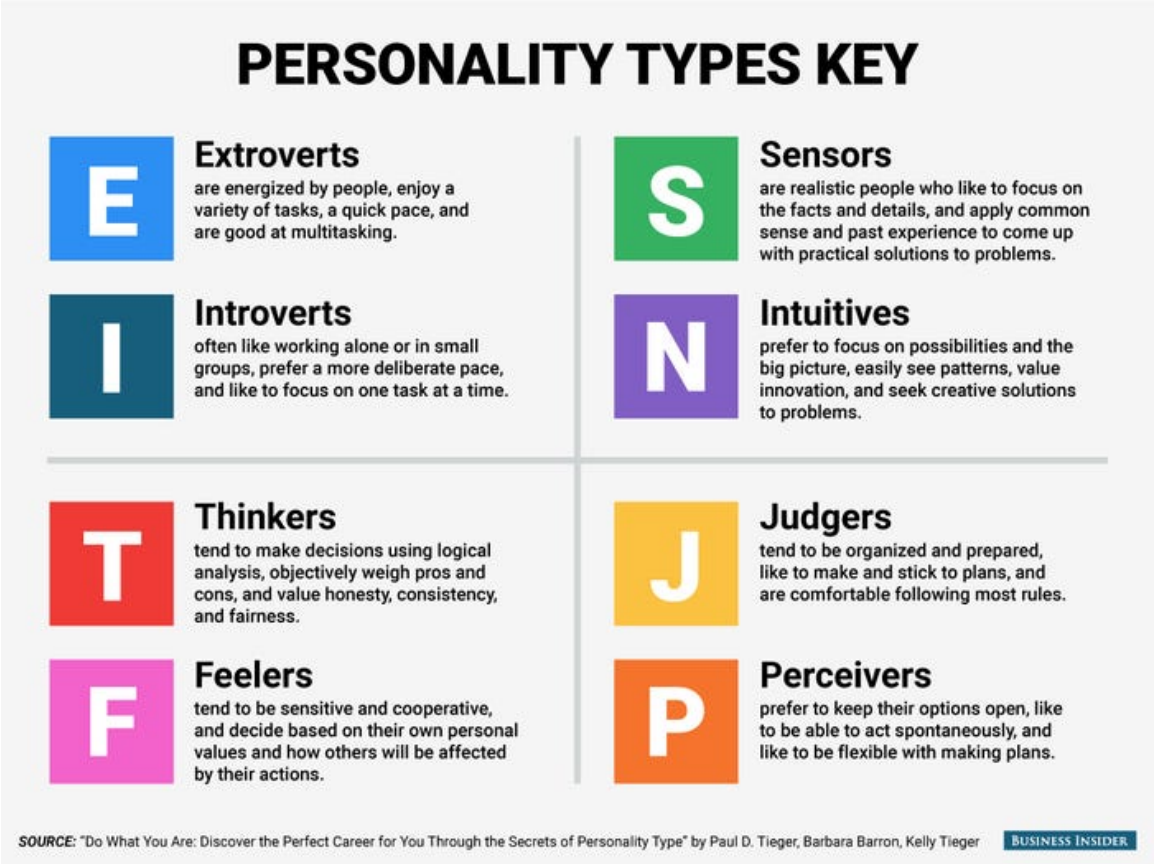

Figure 2.1: The four categories of MBTI and their characteristics as described by a Business Insider article [16]

unique effects. It's also important to note that each of these categories are on a continuum, so individuals within the same classification will still have noticeable variation.

2.1.2 Shortcomings of Myers-Briggs

While it is estimated that about five million people take the MBTI test each year, the validity of this test has long been disputed. Critics of the MBTI have questioned nearly every aspect of the MBTI [18], including its excessive reliance on Jungian theories, categorical classification, and self-reported metrics.

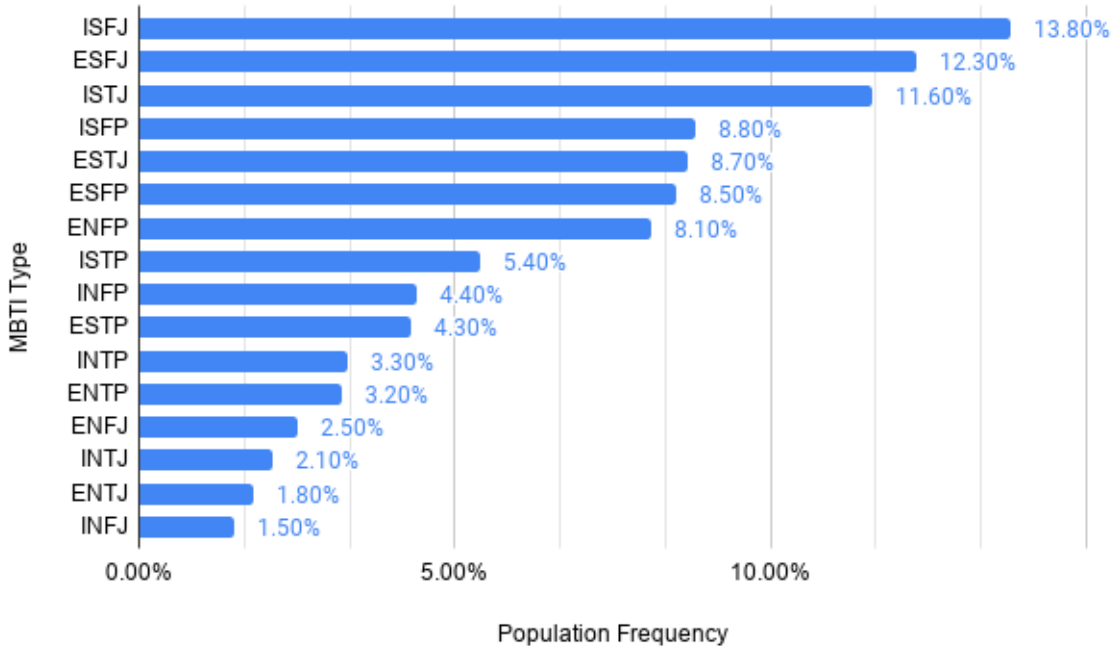

Figure 2.2: The frequency of each personality type in the population of the United States as stated by Isabel Briggs Myers [43]

A lot of the criticism of the MBTI is valid, with most of Jung's theories lacking empirical evidence to support his claims. In addition, Jung's categories have drawn criticism from the basis that the covariance between the pairs of variables within each category is limited. Critics have also taken issue with the fact that the MBTI attempts to classify people into one of two types within each category when in practice, most people fall somewhere in the continuum between the two classifications. This is problematic because an individual who only has a slight preference for a certain metric will be classified in the same way as someone with a strong preference while in reality, that individual may have a combination of characteristics from both sides of the spectrum. Finally, the self-reported responses that the MBTI test asks individuals to take can be problematic. While questionnaires are inherently lacking in validity, it is also important to consider that each of the categories are not bimodally distributed

as one might expect. This leads to an unbalanced distribution of personality types as shown in Figure 2.2.

Another potential shortcoming of the MBTI is the lack of evidence for the reliability of the dimensions. The test-retest error of the most popular MBTI questionnaire is 38.82%. In other words, if an individual took a MBTI questionnaire and received a classification, the chance of receiving the same classification is 38.82%. This number is quite low and indicates that the MBTI suffers from poor test-retest reliability [15]. In general, the introversion vs extraversion scale has the highest reliability and the sensation vs intuition scale has the lowest reliability.

In spite of its shortcomings, the MBTI has been the most popular and well-researched measure of personality typology. One of the main motivations of this study was to eliminate the lack of validity that the self-reported test brings by creating a system to classify users without asking them any direct questions. Despite the flaws described in this section, empirical evidence has shown that certain aspects of the MBTI serve as good predictors of certain behaviors and preferences.

2.2 Applicability of MBTI to Software Design

The notion that an individual's MBTI type is correlated with their behavior and thoughts is backed by empirical evidence [21], but research specifically involving the correlation between MBTI and human computer interaction has been relatively sparse. Existing research in this topic has generally focused on the user experience differences between various personality types. The dimensions that these studies have measured include software usability, user interface design, and task completions.

The link between personality and software usability is hard to measure because both are hard to accurately measure in a quantitative manner. In one study, a researcher attempted to examine the correlation between the MBTI and the Software Usability Measurement Inventory (SUMI), which is a widely used user experience tool that attempts to measure and maximize an application's usability [37]. The study examined various MBTI types against the subscales of the SUMI, which include efficiency, affect, helpfulness, control, and learnability. While most of the subscales did not show a correlation with MBTI, the helpfulness metric showed a statistically significant correlation. Helpfulness aims to measure the extent to which a piece of software is self-explanatory as well as things like documentation and user guides. This indicates that MBTI can impact software usability.

Dr. James Pennebaker's research has shown that there is a strong correlation between various aspects of an individual's writing style and certain personality traits. He found that people counting the different kinds of words that a person says could be indicative of how they view the world [58]. This research also suggested that variations in an individual's feelings of insecurity, threat, and emotional state. Pennebaker created a classification method that classified each type of word into one of several categories. This was an important finding because it demonstrates that aspects of an individual's psychology can be seen in their writing.

One study attempted to find patterns in personality traits on Stack Overflow using IBM Personality Insights. They found that users with high post scores and high reputation on Stack Overflow tended to exhibit a greater degree of Openness and Neuroticism [2]. They also found an inverse correlation between the popularity of an answer and the agreeableness score of that user. These are valuable insights because they indicate that certain online platforms tend to draw in users of some personality types better than others.

Other researchers have attempted to optimize reward contingencies in video games based on the personality of each individual user [44]. They found that optimizing reward contingencies based on player personality helped keep players motivated in the game. While some players tend to prefer rewards based on performance, others tend to prefer rewards for just completing the tasks. While both of these reward types performed better than no rewards, the difference between the two reward contingencies could serve to increase a player's intrinsic motivation. This study showed that personality-based customization could have a significant effect on the user experience.

An individual's personality type also affects the way they complete tasks in software applications. There is a statistically significant correlation between personality type and task-oriented software use [39]. In a study examining how personality interacts with software use, researchers found examples of tasks where MBTI factors had a strong influence on what and how tasks get accomplished. They noted that "Judgers (J) tended to delete or move their email to folders while Perceivers (P) tended to keep email in their inbox even when they didn't need it anymore" and that Feelers (F) were more likely to read reviews than Thinkers (T). These differences are important to keep in mind when designing software because using them appropriately can help increase conversion rates throughout an application.

In summary, existing research suggests that an individual's MBTI type affects the way they interact with software. This highlights the importance of incorporating users' personality types when designing software. Designing software that is customized by a user's personality type could be a paradigm shift in the field of user experience, and the implementation of an automated personality type predictor could be instrumental in facilitating it.

2.3 Automation of Personality Type Prediction

The practice of predicting personality type without a self-reported questionnaire would be a significant advancement in personality psychology. Existing research has shown that predicting personality type from writing style through machine learning systems is possible. Three main pieces of existing research will be used in this study as a point of comparison. Two models used recurrent neural networks (RNN) with long short-term memory (LSTM) and another used gradient boosting. The following subsections will provide a brief overview of the methods used in these studies.

2.3.1 Recurrent Neural Networks and Long Short-Term Memory

Recurrent Neural Networks (RNNs) use past outputs as inputs while having hidden states. While this type of network is often slower, it has several advantages in the field of Natural Language Processing (NLP). The first advantage is that it can process inputs of any length without increasing the size of the model. Another advantage which is that it keeps track of historical information when classifying an input. This is especially useful in this case because it helps the network understand the context in which something was said. They achieve this by executing in loops, which allows the information to persist [53]. Finally, the weights are shared across time, which is a key differentiating feature from a normal feedforward network.

Long Short-Term Memory (LSTM) networks are a type of RNN that are capable of learning long-term dependencies [28]. Remembering information for long durations of time is their distinguishing behavior. LSTMs work similarly to normal RNNs, but they have some key differences. While RNNs have a repeating input layer, LSTMs have four that interact with each other. This allows the model to remember infor-

mation for long periods of time and consequently understand context better. These features are ideal for NLP problems such as this one since the context of words in a sentence and sentences in a paragraph are important.

Hernandez and Knight used text samples from a personality forum as their primary dataset[27]. This is the same dataset that this research will be using and will be described in further detail in the next section. They constructed a binary classifier for each aspect of the personality type and got a 28 percent accuracy rate. They discussed the fact that half of the people that retest their MBTI type get a different type the second time, which is a fairly high error rate. They hoped that their system would work well as a verification system for the test. The main drawback to this piece of research was the somewhat low accuracy rate, but the fact that textual data is becoming available in abundance makes the prospect of automating personality type prediction with higher accuracy an increasingly plausible idea given a larger dataset.

Ma and Liu used text samples from books of authors with known personality types as their dataset [40]. One significant flaw in their research was the size of the dataset, which was quite small for a deep learning model. This naturally caused them to overfit their model to the training data. They could've addressed this in part by adding dropout and regularization. In addition, they also used a relatively simplistic loss function that did not adapt based on how close the prediction was. In other words, they penalized the same amount if an INFJ was classified an ISFJ as they did if it was classified and ESTP. This led to a strange distribution of predictions where we might expect the classifier to miss closer than it did.

2.3.2 Gradient Boosting

Currently, the most effective model for predicting personality type seems to be Extreme Gradient Boosting. One study found that when performing binary classification, they were able to achieve up to 86% accuracy on individual MBTI categories [1]. This is a significant finding because psychology research indicates that specific MBTI categories can influence the way users interact with software. Their study uses the same dataset used in this paper. This study highlighted the fact that some categories of the MBTI are more likely to noticeably manifest in an individual's writing style. Another key takeaway from this study was the effectiveness of Term Frequency–Inverse Document Frequency (TF-IDF), which essentially tries to assign words an importance level based on their frequency. One of the most effective parts of this paper was its' robust preprocessing, and a lot of the preprocessing done in my study, which will be explained in the implementation section, derives its influence from their research.

Their main advancement to the state of the art was the use of a library called Extreme Gradient Boosting (XGBoost). Gradient boosting is a popular machine learning technique that is often used in classification problems and it works by building an ensemble of prediction models [45]. Ensembling is the practice of combining multiple machine learning models to build one strong predictive model. Gradient boosting and ensembling are also very effective at limiting overfitting compared to other approaches. These ensembles generally tend to consist of weak prediction models such as decision trees, which are non-parametric algorithms with a tree-like structure. In a decision tree, a node represents an attribute, with each branch representing the outcome of that attribute. A leaf in a decision tree describes the output, which would be a personality type identifier (e.g. ENFP) in this case, given a certain input, which

would be a user's post data. Figure 2.3 below is a simplified example of a decision tree.

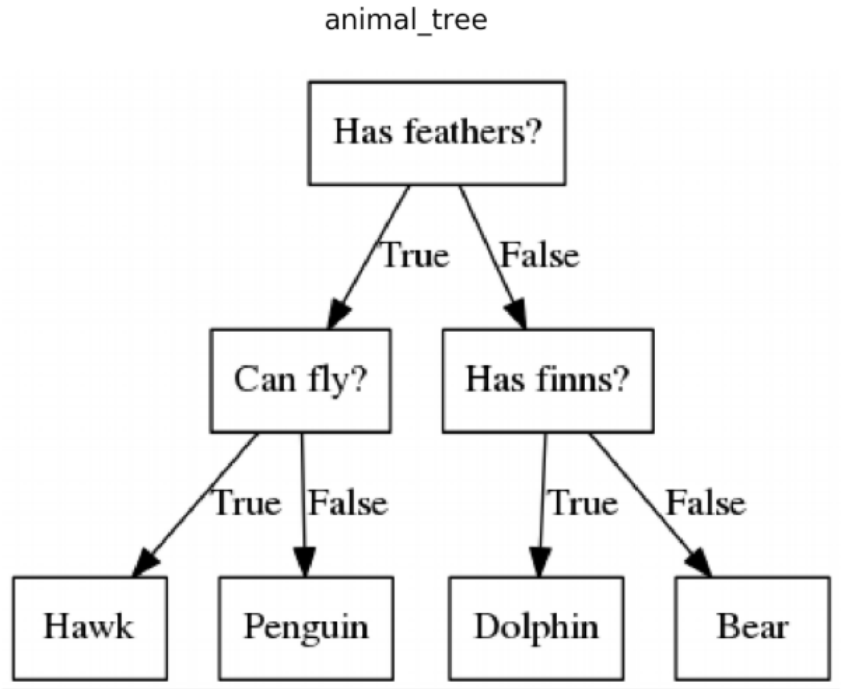

Figure 2.3: A simple example of a decision tree that classifies animals [12]

When these decision trees are iteratively constructed, it allows for more accurate predictions by attempting to correct each model based on the residual errors of the previous one. These steps are repeated with each iteration attempting to fit the residuals of the previous predictor. The final prediction made from boosting takes into account the aggregate result from the predictions made by the models. Amirhosseini and Kazemian showed the effectiveness of ensembles and boosting, and consequently influenced the direction of this research.

Chapter 3

IMPLEMENTATION

3.1 Data

The dataset used in this research was scraped from an online personality forum on personalitycafe.com and it is freely available on Kaggle, an online data science community [32]. This dataset contains posts on the forum from 8675 users. Each user has 50 text samples separated by a '|||' sequence, which totals to 433,750 total user comments. These users each have a labeled MBTI type with all four dimensions: introversion (I) - extraversion (E), intuition (N) - sensing (S), thinking (T) - feeling (F), and judging (J) - perceiving (P). The distribution of personality types in this dataset is quite different from the overall population distribution, as shown in Figure 3.1. Intuitively, it is understandable that introverts would have a larger presence on a semi-anonymous online forum. Figure 3.2 describes the distribution of the prevalence of each individual MBTI attribute in the dataset. This dataset appears to have a significantly distorted proportion of people with each attribute. higher proportion of people with the Introversion and Intuition attributes in their types than their Extraversion and Sensing counterparts. In the general population, the distribution of the the individual MBTI attributes is more balanced, as seen in Figure 3.3. The users from this dataset tended to be heavily introverted while the general population tended to be fairly split in that category. In addition, people with the Intuition attribute were heavily overrepresented when compared to the the population numbers. The Thinking-Feeling and Judging-Perceiving dimensions were relatively consistent with the population statistics. It's important to note that this dataset does not prop-

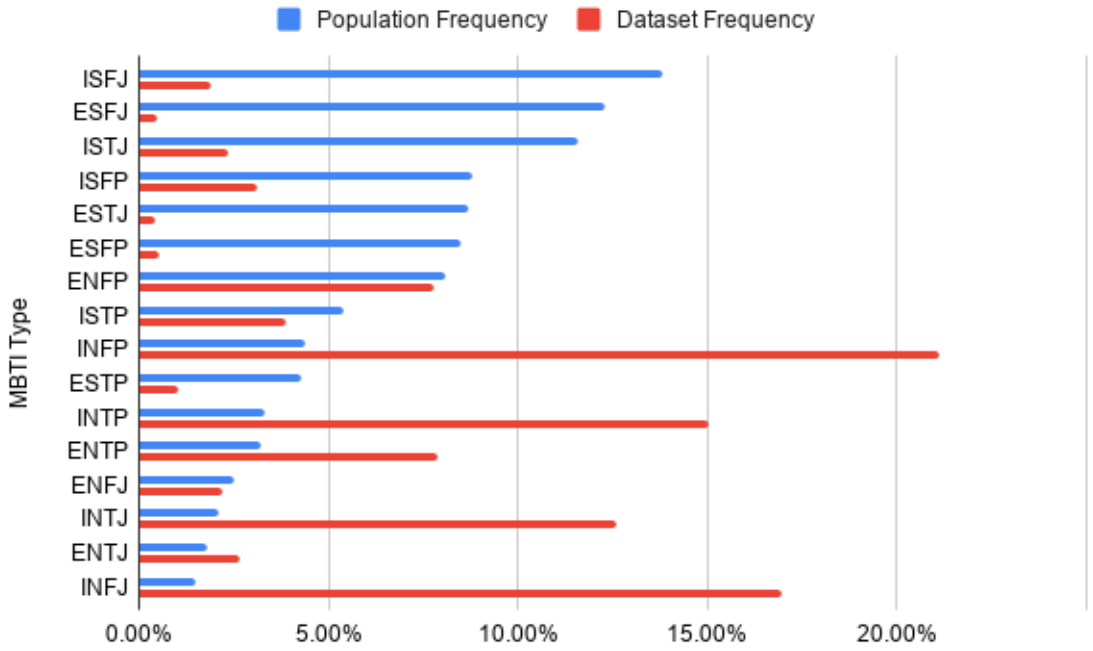

Figure 3.1: The frequency of each personality type in the population vs the dataset

erly represent the population because it was not randomly sampled and likely has self-selection bias. This means that the characteristics under which the individuals in this group were selected suffers from sampling bias [25].

Upon first look at the dataset, there don't seem to be obvious differences between the length of the posts or variance of the length, as shown in Figure 3.4.

3.2 Libraries

The main libraries used in this study were NumPy, pandas, Natural Language Toolkit (NLTK), and Scikit-learn. This implementation was written in Python 3.7.6.

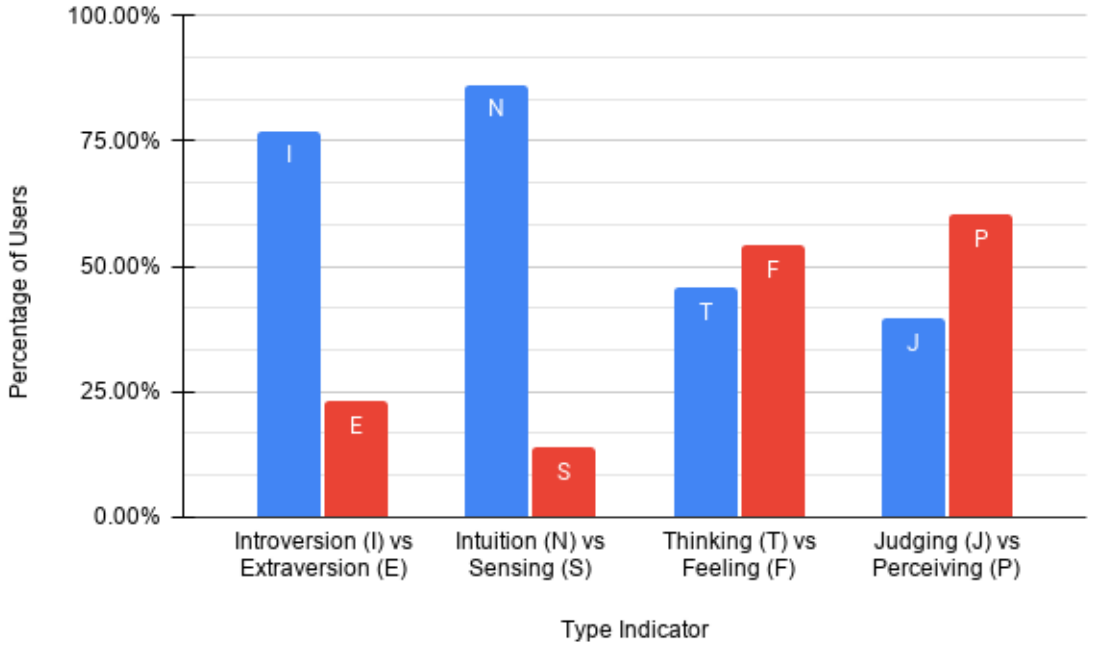

Figure 3.2: The frequency of each dimension of MBTI in the dataset

3.2.1 NumPy

NumPy is an effective tool for managing and manipulating large array-like data [23]. NumPy is an open source project that was first released in 2005, iterating on the work done by earlier numerical computing libraries, with the goal of simplifying the task of working with big data in Python. It is widely used in machine learning applications because of its ability to work with large matrices and higher order functions. The use of NumPy in this study is limited to the evaluation step of the model.

3.2.2 pandas

pandas is a popular, open source data analysis and manipulation tool and its DataFrame object is often used to work with machine learning models [59]. pandas was written by Wes McKinney in 2008 in an attempt to build a quantitative analysis tool for

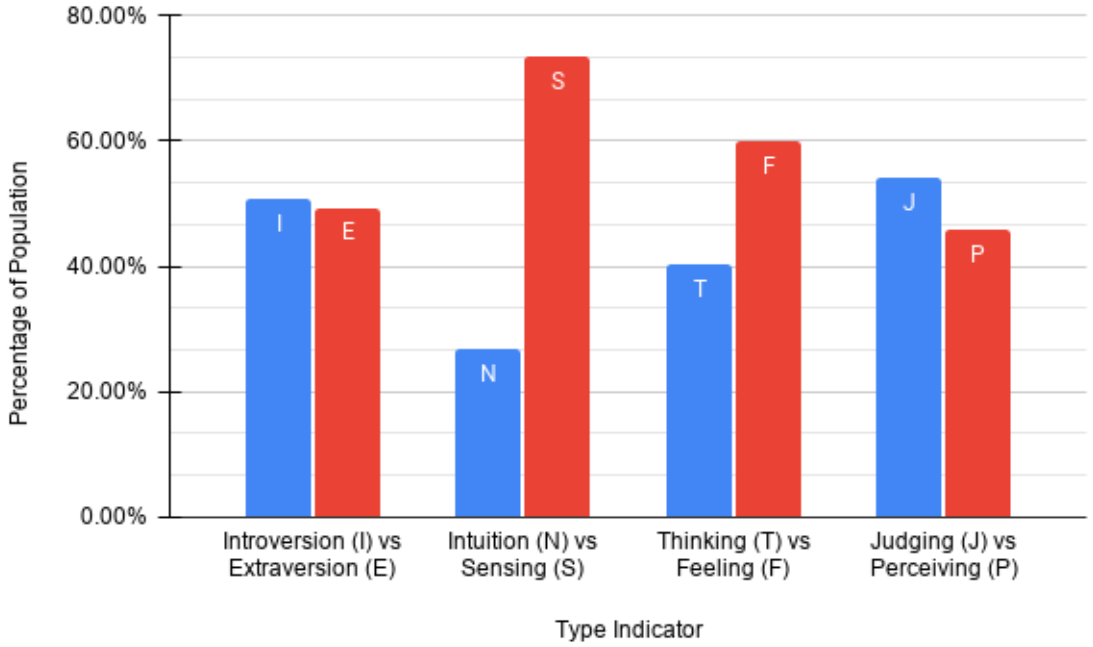

Figure 3.3: The frequency of each dimension of MBTI in the US population [35]

financial data. It currently serves as a building block for machine learning because of its flexibility and ease of data manipulation. This library allows for easy importing of various forms of data, including comma-separated values (CSV), excel spreadsheets (XLS, XLSX), and JSON. It works best with table-like data, and is popular due to its emphasis on performance with big data, which is critical in machine learning. It contains functions that simplify data aligning, reshaping, slicing, and mutating, among other applications. The DataFrame object, a two-dimensional labeled data structure, was the primary data structure used in this research.

3.2.3 Natural Language Toolkit (NLTK)

Natural Language Processing (NLP) is the practice of helping a computer understand and manipulate natural language data through software [7]. NLP has been extensively

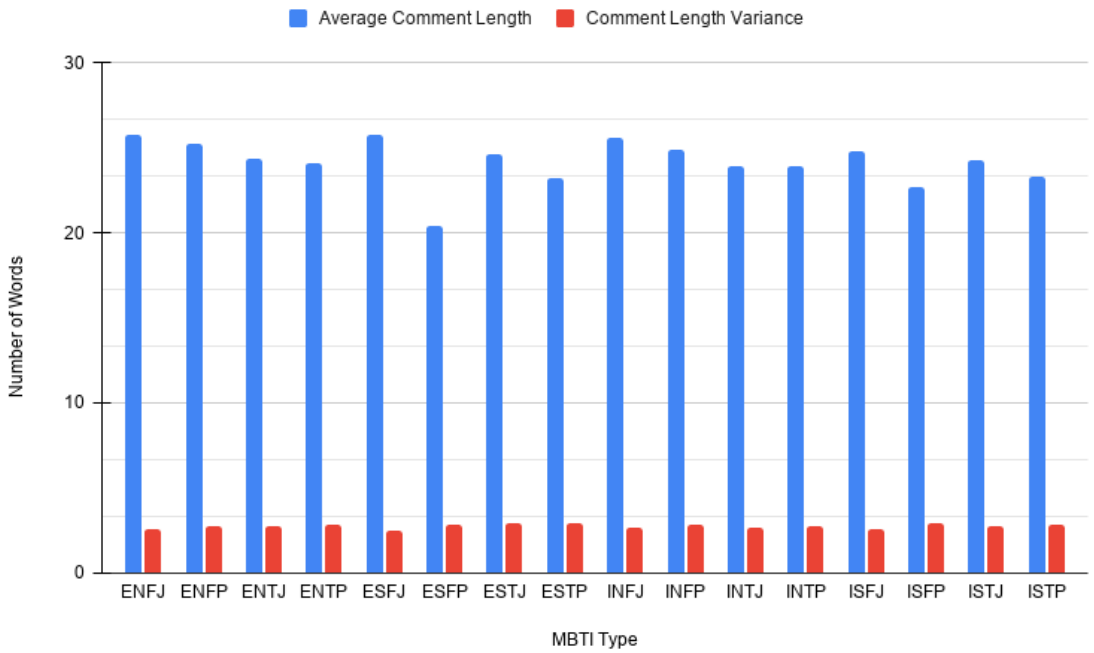

Figure 3.4: Average comment length and comment length variance by MBTI type

studied in recent decades. It has been a cornerstone for services that attempt to analyze and generate text and speech data, such as virtual assistants (e.g. Google Assistant, Siri, Alexa), chatbots, and sentiment analysis. NLP research has even reached the point where models are able to generate text that is indistinguishable from human text [24]. This project focuses on natural language understanding, using a popular NLP library called Natural Language Toolkit (NLTK). NLTK was developed in 2001 with the goal of simplifying the analysis and manipulation of language data [4]. This library was used primarily in the preprocessing phase of this study to prepare the data for training.

3.2.4 Scikit-learn

Scikit-learn is a machine learning library that offers a number of high-performance algorithms that make tasks like classification, regression, and clustering accessible in a high-level language [46]. Scikit-learn has been used by researchers and machine learning practitioners extensively and it has proven itself as a state of the art machine learning library. It was developed by David Cournapeau and Matthieu Brucher, and released in 2010. This library was picked for this study because of its robust classification models and preprocessing functionality. Scikit-learn is the library that this study relies on most, utilizing its Ensemble, Decomposition, Feature Extraction, Model Selection, and Pipeline modules [10]. These modules will be discussed in further detail in the upcoming subsections.

3.3 Preprocessing

Preprocessing for this library was primarily done through several NLTK functions and it was based on the preprocessing done by [1], which was discussed in the previous section. This is an important phase of the study because removing irrelevant data from the dataset in this phase significantly improves the model's performance. The preprocessing phase for this study consisted of three main parts: data cleaning, tokenization, and stopword removal. All of these are crucial components to perform on this dataset in order to achieve optimal performance, and they will be discussed in further detail in the upcoming subsections.

3.3.1 Data Cleaning

Data cleaning is an important component of machine learning in classification problems and is an essential technique in the preprocessing phase for raw data [31]. Data cleaning allows for the reducton of noise, inconsistencies, and errors in a dataset. The dataset used in this study contains 50 comments from each individual user, and because it allowed for virtually any text to be posted by the user, it contains a sizable amount of data that contains insignificant information for our purposes. The following list describes the techniques employed in the data cleaning phase of this experiment, with justification provided:

- Links were removed from this dataset because they often contain information that does not contain any discernible meaning without exploring the contents of the link (e.g. http://www.youtube.com/watch?v=EY21CYqGaMw).

- All text was turned into upper case because the important information that we want to examine in this dataset involves context and we have little use for predicting sentence beginnings. In addition, due to the informal nature of the data source, capitalization is an unreliable source of information.

- Information in square brackets was removed because it usually either contained information that paraphrased text that was already written or nonsensical text that would only serve to obfuscate the data.

- Punctuation and stop words were removed because this is a standard NLP practice since both of those components generally carry little weight for the overall meaning of textual data for this purpose. Stop words in English refer to words that generally do not affect the meaning of a sentence, such as “the”, “and”, and “was”.

- Words with numbers were removed from this dataset due to an abnormally high amount of nonsensical alphanumeric sequences and unicode characters (e.g. 'x93a', '&#x27').

Most of the cleaning of this dataset was performed using regular expressions since it tends to be the best one for performance. The removal of these features had a significant effect on the accuracy of the algorithm and this will be discussed in the next section of this paper.

3.3.2 Tokenization

Tokenization refers to the task of splitting natural language data such as a user's posts into smaller units. These smaller units typically tend to be words and numbers, which is the case in this study. This process is an essential preprocessing step that allows for machine learning algorithms to make sense of textual data. For the purposes of this study, tokenizing by words is the appropriate method, and this is generally a fairly simple task. Because of the lack of need for special features that NLTK's word_tokenize and spaCy provide, the main metric used in picking a tokenier was performance. Tokenization was performed using NLTK's regex_tokenize function because it is typically regarded as the most efficient tokenizer [38].

3.3.3 Stemming

In most natural language datasets, it is common practice to remove suffixes of words in order to generate their root words. For example, if the word is "jumping", the process of stemming would shorten this to its root word "jump". This is a good practice in a lot of NLP applications because the use of variations of a word does not generally change the meaning of the sentence. The first modern implementation of this practice,

which was called Porter Stemming, was developed by Martin Porter in 1980 [48] as a means of improving the process of information retrieval from natural language data. In this study, Porter2 Stemming was used to perform stemming on this dataset. Porter2 was built in a special language called Snowball, which Porter is also credited with creating. According to Porter, the original Porter stemmer inaccurately stems words about 14 percent of the time, which is a significant shortcoming. Porter created the Snowball language as a system that allows for the creation of properly defined stemming algorithms. Porter2 Stemming was chosen because it is generally better than Porter Stemming for most applications, and Lancaster Stemming is typically much more aggressive [54]. its also important to note that the Snowball stemmer has implementations in a large number of natural languages, which allows for this study to be better expanded in the future. While Porter2 generally performs fairly well in most cases, it is not a perfect algorithm. For example, using the Porter2 algorithm on the word "multiply" returns the stem "multipli". This is due to the fact that stemming relies on a set of somewhat rigid rules rather than a dictionary of mappings. Porter2 performs about five percent better than its predecessor and avoids the overstemming problem that Lancaster often runs into, which makes it the clear choice for this study [26].

3.4 Term Frequency-Inverse Document Frequency (TF-IDF)

When processing natural language data, it is important to define a structure that scores each word in the dataset. Because machine learning algorithms work best with numbers, we need another algorithm that can translate the words in our input into a vector. One popular and efficient algorithm for performing this is Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF aims to measure the relevance of a word to a document in a collection of documents [11]. In other words, the importance

of a word in one particular sample goes up when it appears at a higher frequency than in the collection of samples.

TF-IDF is composed of two main components that work together to create a weight for each word: Term Frequency (TF) and Inverse Document Frequency (IDF) [41]. Term Frequency refers to the proportion at which a term appears in a document. The following equation describes how TF is calculated:

$TF(t) = n_t/N$, where n_t is the number of occurrences of term t in the document and N is the total number of terms in the document.

Inverse Document Frequency is a heuristic that attempts to measure the relative importance of the term. This is an important component of this process because TF alone would simply put weight on words that occur most frequently. This would put an unreasonable amount of weight on words like "MBTI" since they occur very frequently on likely every document in this dataset. IDF relieves this shortcoming by decreasing the weight of words that occur in a lot of documents, while increasing the weight of words that occur in fewer documents. It does so by applying this equation on the term:

$IDF(t) = ln(D/n_D)$, where D is the total number of documents and n_D is the number of documents that contain the term t.

TF-IDF is then calculated using the product of these two equations:

$TFIDF(t) = TF(t) \times IDF(t)$

As a whole, TF-IDF places the most weight on words that occur frequently in one document while occurring infrequently in the collection of documents as a whole. Vectorization is an important process in text classification and TF-IDF provides a relatively reliable measure that machine learning algorithms can work with.

3.5 Truncated Singular Value Decomposition (Truncated SVD)

One key problem with large datasets like the one used in this study is the difficulty of processing data with such high dimensionality. Truncated Singular Value Decomposition (TSVD) is Scikit-learn's implementation of Singular Value Decomposition (SVD) and it is widely used in in NLP applications. SVD is a useful linear algebra tool that performs data reduction on high-dimensional data. SVD helps reduce large amounts of data, like the vectorized textual data used in this study, into its key features that are necessary for analyzing this data [9]. It serves as a preliminary step for a large number of machine learning applications. This technique allows us to tailor the data to a coordinate system to solve a specific problem by allowing us to understand the dominant patterns of the data. This technique allows us to tailor a coordinate system to data in order to understand dominant patterns of the data [6]. SVD works by decomposing a matrix into several component matrices, each of which contain key properties of the original matrix.

The equation for SVD works as follows: $A = U \times \Sigma \times V^T$, where A is the original $m \times n$ matrix that we want to decompose, U is an $m \times m$ matrix, Σ is a $m \times n$ diagonal matrix, and V^T is a transposed $n \times n$ matrix. To find the most relevant features in the dataset, the largest k singular values are selected from Σ. TSVD is a slight variation of SVD that only uses the first k columns of U and V [52].

The definitions of the main parameters of the Scikit-learn implementation of TSVD as well as the values passed are described below:

n_components - **10**. This refers to the dimensionality of the output data.

n_iter - **5**. This refers to the number of iterations used by the SVD solver.

3.6 Classifiers

Classification is a subset of supervised machine learning problems that attempts to predict the class that an element belongs to based on features of that data. Text classification is an increasingly popular use case, with applications like content tagging, search engine optimization, and spam filtering [22]. This study employs multiclass text classification in an attempt to predict users' MBTI based on their text samples. One popular approach to solving text classification problems has been ensemble learning, which attempts to use multiple classification algorithms and aggregate the results in order to achieve better predictive performance. In this study, three Scikit-learn ensemble learning methods were examined: Random Forest Classifier, Extra Trees Classifier, and Gradient Boosting Classifier. The performance of each of these classifiers is evaluated in an attempt to find the best ensemble classifier for this use case.

3.6.1 Multiclass Classification

Of the sixteen personality type indicators that the MBTI presents, there are four individual dimensions. For this study, this means that there are two options for classification: binary classification and multiclass classification. The first gives us the option to create four separate binary classifiers, each of which would tackle one dimension of the MBTI (i.e. One classifier might attempt to classify a user as either an Introvert or an Extrovert). This is the approach that most existing research in this realm have taken. The other approach is multiclass classification, which attempts to fit a user into one of several existing classes. In this case, that would mean that a single classifier would attempt to classify all four dimensions of an individual's personality type. This is the approach that has been chosen in this study because

predicting all four dimensions of an individual's MBTI has been shown to be more beneficial to understanding an individual than any individual component due to the unique interactions that each component of the MBTI have with each other [51]. Choosing a multiclass approach allows for us to define a loss function that focuses on getting all four dimensions correctly since the model will be penalized the same amount regardless of how close it may have been.

3.6.2 Ensemble Learning

Ensemble learning, in the context of classifiers, is the practice of combining decisions from multiple classification models in order to achieve higher accuracy than any of the models alone [49]. Ensemble algorithms generally reduce the amount of noise, bias, and variance that a typical machine learning model might suffer from. The main disadvantages of ensembles is their somewhat poor computational performance as well as the reduced interpretability of the model. This means that figuring out why a certain prediction was made is much harder in ensembles compared to traditional machine learning algorithms.

There are two primary families of ensemble methods: averaging methods and boosting methods. Averaging methods, as the name suggests, involve averaging the predictions of several estimators. This process reduces the variance that a single estimator might encounter. Boosting methods, on the other hand, call for the sequential building of estimators, with each consecutive one attempting to reduce the bias of the previous estimator. Extra Trees Classifier and Random Forest Classifier are both averaging methods while Gradient Boosting Classifier is a boosting method.

3.6.3 Random Forest Classifier

Random forest is an ensemble method that is built on top of the concept of decision trees, which are often considered the building blocks of random forests. In the training phase of a random forest, a large number of decision trees are created from the input data. Each of these decision trees gets a random subsample of the input data and each node in the decision tree represents a feature [19]. These samples are drawn with replacement, which means that each sampling unit can be selected more than once. Random forests are particularly useful because of their tendency to introduce randomness while constructing the classifiers. This is a good technique because it reduces the variance of the base estimators and consequently reduces overfitting. Random forest generally performs better than most other machine learning algorithms on outliers as well as high-dimensional parameter spaces.

In the construction process of each tree in a random forest, it is important to figure out the importance of each feature and place the feature with highest importance at the root of the tree. Traversing down the tree generally also indicates a greater level of uncertainty. In order to calculate the level of uncertainty of each node, the Gini index is used. The Gini index measures how important a feature is to classification by figuring out the amount of randomness in a data point [56]. It's a number ranging from 0 to 0.5, indicating the distribution of paths down the decision tree. The following equation is used to calculate the Gini index of a particular node in a decision tree: $Gini = 1 - \sum_{i=1}^{n}(p_i)^2$, where p_i represents the probability of a sample being classified correctly to a certain class. A node with a low Gini index means that it has a higher level of purity. This indicates that the chance of incorrectly labeling a data point on that node is low, which is the rationale behind placing it higher in the decision tree.

The Scikit-learn implementation of random forest calls for two main parameters: n_estimators and max_features. The parameters used for this implementation as well as the definitions of each parameter are described below.

n_estimators: **200**. This refers to the number of trees in the forest. Using more trees in a random forest typically increases accuracy, but the algorithm suffers from diminishing returns after a certain point.

max_features: **sqrt**. For classification tasks, it is generally recommended to use the square root of the number of features in the dataset (i.e. $\sqrt{num\_features}$).

criterion: **gini**. This is the function that measures the impurity of each node in the decision trees. Gini is computationally faster than entropy, which is the alternative measure.

bootstrap: **True**. The trees are samples with replacement.

3.6.4 Extra Trees Classifier

Extra trees, also known as extremely randomized trees, are based on random forests with two key distinctions. The first is that extra trees, unlike random forest, samples without replacement [3]. This means that extra trees uses the entire original sample, which consequently increases the amount of variance. The second distinction from the random forest algorithm is the method that extra trees uses to split the nodes. While random forest chooses an optimum split at each node by looking for the best discriminative thresholds, extra trees makes this split randomly for each feature. After choosing the split, both algorithms work identically, attempting to choose the best splits between the subsets of features. While this increases the bias of the model, the advantages are two-fold [20]. Because extra trees does not calculate the optimal

split and instead chooses the split randomly, the computational time is significantly reduced. In addition, this further reduces the amount of variance in the model. It is also important to note that extra trees performs with very similar accuracy to random forest.

Scikit-learn's implementation of extra trees calls for the same main parameters as random forest. The size of the model is $O(M \times N \times log(N))$, where M is the number of trees and N is the number of samples. The parameters used for this study are shown below.

n_estimators: **180**.

max_features: **sqrt**.

criterion: **gini**.

bootstrap: **False**. The whole dataset is used to build each tree.

3.6.5 Gradient Boosting Classifier

Gradient boosting, also known as gradient boosted decision trees, is a type of ensemble method that relies on the concept of boosting. Boosting is the idea of improving weak models iteratively until they become stronger predictors [57]. Boosting relies on filtering observations that a weak learner can predict on its own while developing another learner that can handle the observations that the existing learners have difficulty predicting [8]. Gradient boosting was first proposed as a boosting paradigm in 1999, when it proved its effectiveness in classification and regression problems when using regression trees [17].

Gradient boosting relies on three main components: loss function, a weak learner, and an additive component. In most classification cases, a differentiable logarithmic loss function such as deviance are used. In this case and for most use cases of gradient boosting, regression trees are used as the weak learners. Regression trees, unlike decision trees, accept continuous values rather than binary values. Similar to random forest and extra trees, the regression trees are constructed using algorithms to measure the quality of the split in an attempt to find the best split. The significance of using regression trees instead of decision trees is that it gives us the ability to combine the outputs of the other regression trees in order to correct for errors in predictions. The Friedman mean squared error is the criterion used to measure the quality of the split and it refers the the distance of the prediction from the actual values. The last component of gradient boosting is the additive component, which is necessary because existing trees in a gradient boosting model model cannot be changed. The additive component used in most gradient boosting applications is called functional gradient descent, which is similar to gradient descent. Functional gradient descent attempts to create each new tree to reduce the loss values of existing predictors.

The parameters below were used in this implementation of gradient boosting using the scikit-learn api. The two main parameters that were optimized were n_estimators and learning_rate.

n_estimators: **200**.

learning_rate: **0.03**.

criterion: **'friedman_mse'**. The Friedman mean square error function is based on the mean square error function but it also includes an improvement score metric that Friedman describes in his paper [17].

loss: **deviance**.

3.7 Cross Validation

Cross validation is a technique that is commonly used to evaluate machine learning models in an attempt to approximate the usefulness of the model on data that it has not seen before [5]. It is particularly useful when attempting to find a good predictor given a small dataset. In k-fold cross validation, the main parameter, k, determines the number of groups that a dataset should be split into. These groups are referred to as folds and a total of $k-1$ folds, each of size n/k where n is the total number of data points, are used to train the model. This process is repeated k times, with each iteration using a different fold to evaluate the model. The advantages of this process are increased generalizability on unseen data and a lower likelihood of overfitting [29]. It typically yields less biased estimates than the traditional train-test split method. K-fold cross validation is often referred to as the gold-standard for evaluation a model's performance and usually outperforms the alternatives.

Choosing the appropriate k value can be a challenge in k-fold cross validation. The general rule is that that a k value of 5 or 10 is ideal for limiting the amount of bias and variance [30]. Choosing a k that is too high limits the number of samples that can be generated while choosing a k value that is too low can cause increased bias.

3.7.1 Stratified K-Fold

This study uses a slight variation of k-fold cross validation called stratified k-fold. As discussed earlier in this paper, the distribution of each personality type in this dataset is very imbalanced. Stratified k-fold is better choice in this case because of the imbalanced class distribution of this dataset. Stratified k-fold tackles this problem by

keeping the distribution of the datasets the same in each of the folds. The Scikit-learn implementation of Stratified k-fold's parameters are described below.

n_splits: **5**. This is the number of folds used.

shuffle: **True**. This shuffles the data prior to creating the folds.

Chapter 4

DISCUSSION

4.1 Results

Table 4.1: Accuracy of Each Classifier

Model	Accuracy
Random Forest	45.35%
Extra Trees	45.29%
Gradient Boosting	44.88%

The results of this experiment indicate that predicting an individual's MBTI using their writing style from comments on online forums is possible with reasonable accuracy. The three classification methods used in this experiment performed somewhat similarly when their parameters were optimized, as shown in Table 4.1.

These results show that random forest and extra trees performed very similarly to each other. This was expected since extra trees is considered an iteration of random forest. Despite their similarity in accuracy, extra trees is the better choice because it is less computationally expensive due to the way the splits are created in the estimators. This means that a classifier with same number of estimators would make a prediction much faster in extra trees. Extra trees also has less variance than random forest, which means that we can expect greater precision from extra trees.

Extra trees and random forest surprisingly performed better than the gradient boosting implementation in this study. While gradient boosting typically performs better on most datasets after parameter optimization, it likely performed marginally worse in this case due to the amount of noise present in the data, which is expected consid-

ering the source of the dataset. Despite this limitation, gradient boosting appears to be the best approach to this problem

4.2 Comparison To Existing Work

Several pieces of existing research have attempted to solve this problem using various recurrent neural network architectures [27][40], but the approach that has performed best has been gradient boosting [1]. This is likely due to the relatively small size of the dataset since recurrent neural network architectures like long short-term memory would otherwise be ideal for solving this problem. As shown in this study, gradient boosting appears to have a very similar accuracy rate to random forest and extra trees. Despite the fact that it scored marginally lower, it is likely the best approach for this type of problem if the hyperparameters are further tuned and the dataset has less noise. It is also important to note that the gradient boosting study made four separate binary classifiers that attempted to maximize the accuracy of each dimension of the MBTI while this study attempts to optimize the accuracy of getting all four dimensions right. This makes it somewhat hard to compare the two studies directly since they have four individual accuracy rates for each dimension, the product of which is 32.96%.

4.3 Comparison To Test-Retest Rate

One metric that was used to measure the impact of this study was the test-retest rate of questionnaire-based MBTI assessments. In this case, the test-retest rate indicates the likelihood that an individual, who took the assessment twice with a five to seven month gap between the attempts, receives the same personality classification both times. The most popular online assessment, 16Personalities, has a test-retest rate

of 38.82%. This means that the classifier developed in this study marginally beats the test-retest rate, which is a significant finding. The system developed in this study could serve as an effective way to validate the results of a questionnaire-based assessment for some use cases.

4.4 Ethical Considerations

This study has shown that predicting individual's personality types using their post and comment data from online communities is possible. While the benefits of this are significant in some areas, it is important to consider that most social media users generally do not have the freedom to control how their data is used. A number of large technology companies generate the majority of their revenue through targeted advertising. It is estimated that 79% of all websites contain tracking scripts that attempt to capture your web footprint [36]. Targeted advertising also tracks user behavior, including posts and comments on social media. Evidence indicates that targeted ads strongly influence self-perception, consequently affecting personal beliefs and behaviors [50]. The ability to discern an individual's personality gives targeted advertising more power than it should have. Regulation of the data that targeted advertising can use is essential to protecting consumer privacy.

Chapter 5

FUTURE WORK

This work demonstrated that the concept of predicting personality from writing style is possible with reasonable accuracy; however, there are quite a few potential areas for improvement.

5.1 Dataset

While this dataset provided a good start for solving this problem, it is clear that a model developed with this training set is unlikely to generalize well to other forms of textual data. Expanding this dataset to social media posts and comments would greatly increase the power as the diversity of content on those platforms tends to be much greater than that of the dataset used in this study. In addition, this dataset is very unbalanced and while the distribution of personality types in the population is unbalanced as well, the distributions are significantly different from each other as shown in Figure 3.1. As a result, it is is unclear whether this dataset is an appropriate representation of the population. While the prediction power of the classifier trained in this study is weak outside of writing from the personality forum, it is likely that a more complete dataset with increased variance could expand the results of this experiment to a much larger scope.

5.2 Classification

The three main classifiers used in this study proved to be reasonably effective at solving this problem; however, there are limitations to these implementations. As previously discussed, the implementations in this study made the best of a relatively small dataset. While this proved to be a good candidate for data resampling using stratified k-fold cross validation, it is evident that this task does not fully accomplish what a neural network with a large dataset could. Using a convolutional neural network (CNN), especially in hierarchical text datasets, would allow a classifier to consider context, which can often change the meaning and sentiment of a comment [47]. For example, a social media dataset would likely take into account the hierarchy of the comment threads as well as the content of the original post in order to make a more well-informed classification. Another potential area of improvement for the classification task is the method used for word embedding. While TF-IDF performs fairly well in assessing the relevance of each word, Bidirectional Encoder Representations from Transformers (BERT) has shown a high level of effectiveness in NLP tasks and could be a better choice for this study given its prowess in understanding context [13]. It is important to note, however, that both CNNs and BERT are much more computationally expensive than the tools used in this study.

5.3 Regulation of Target Advertising

With the Cambridge Analytica scandal in 2018, it became evident that social media has the power to strongly influence people's beliefs and behaviors [14]. The fact that this power goes virtually unregulated in the United States is a huge invasion of privacy. The possibility of being able to microtarget individuals using data that was meant for different purposes should be considered unacceptable. At it's core, targeted

advertising serves as a manipulation machine that is able to predict our behaviors and influence our thoughts using individuals' data. In a world where algorithms are able to make sense of user data at such a large scale, it is vital that the type of data that these targeted advertising companies are able to use is restricted in order to protect user privacy.

Chapter 6

CONTRIBUTIONS

The major contributions of this work include:

1. Three ensemble classification methods for predicting personality type from writing style.
2. A multiclass approach in classifying MBTI dimensions.
3. An examination of the applicability of MBTI in human-computer interaction.
4. Guidelines for appropriate use cases for several classification methods to solve this problem.

Chapter 7

CONCLUSION

This study highlighted the effectiveness of ensemble classifiers such as random forest, extremely randomized trees, and gradient boosting, at predicting personality type from writing style given a relatively small training dataset. The accuracy of the system proposed currently performs better than existing research and manages to beat the test-retest rate of the most popular personality assessment. With a more versatile and larger dataset, the gradient boosting approach appears to be the best choice while extremely randomized trees appears to perform marginally better given a small, noisy dataset. With a much bigger dataset, a convolutional neural network would likely have the most predictive power, but the lack of a publicly available dataset of this size makes research on it difficult.

Our digital footprint has grown a significant amount in recent years due to the increasing prevalence of social media and website trackers. With this data comes a great deal of power that has gone relatively unchecked, and the need for regulation has become increasingly urgent given the assumptions big tech companies can make using data as simple as the dataset used in this study. There are also many positive use cases for classifiers such as this one. Existing research claims a better user experience when it is tailored to users' personality types. In addition, this opens up the possibility of replacing questionnaire-based personality assessments in some use cases, which could remove the time consumption and self-serving bias that are inherent to those methods. While the dataset used in this study had limitations on the generalizability of this classifier, the capability of systems proposed in this study can be utilized for more purposes with a more robust and versatile dataset.

BIBLIOGRAPHY

[1] M. H. Amirhosseini and H. Kazemian. Machine learning approach to personality type prediction based on the myers–briggs type indicator. *Multimodal Technologies and Interactions*, Mar 2020.

[2] C. Asato and B. C. da Silva. Understanding the personality traits of stack overflow users: Text analysis with ibm personality insights. Jun 2019.

[3] P. Aznar. What is the difference between extra trees and random forest? *Quantdare*, Jun 2020.

[4] S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python*. O'Reilly Media, 2016.

[5] J. Brownlee. A gentle introduction to k-fold cross-validation. *Machine Learning Mastery*, May 2018.

[6] J. Brownlee. How to calculate the svd from scratch with python. *Machine Learning Mastery*, Oct 2019.

[7] J. Brownlee. What is natural language processing? *Machine Learning Mastery*, Aug 2019.

[8] J. Brownlee. A gentle introduction to the gradient boosting algorithm for machine learning. *Machine Learning Mastery*, Aug 2020.

[9] S. L. Brunton and J. N. Kutz. *Singular Value Decomposition (SVD)*, page 3–46. Cambridge University Press, 2019.

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton,

J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[11] D. Cam-Stei. Tf-idf vs word embedding, a comparison and code tutorial. *Medium*, Feb 2019.

[12] D. H. Chowdary. Decision trees explained with a practical example. *Towards AI - The Best of Tech, Science, and Engineering*, May 2020.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Association for Computational Linguistics*, 2019.

[14] G. Edelman. Why don't we just ban targeted advertising? *Wired*, Mar 2020.

[15] J. Fayard. Your favorite personality test is probably bogus. *Psychology Today*, Sep 2019.

[16] R. Feloni. The best jobs for every personality type. *Business Insider*, Aug 2015.

[17] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[18] A. Furnham. *Myers-Briggs Type Indicator (MBTI)*, pages 1–4. Springer International Publishing, Cham, 2017.

[19] A. Garrido. Random forest: many are better than one. *Quantdare*, Feb 2017.

[20] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[21] A. Gordon. In defense of the myers-briggs. *Psychology Today*, Feb 2020.

[22] S. Gupta. Text classification: Applications and use cases. *Medium*, Feb 2018.

[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[24] W. D. Heaven. Openai's new language generator gpt-3 is shockingly good-and completely mindless. *MIT Technology Review*, Jul 2020.

[25] J. J. Heckman. *Selection Bias and Self-selection*, pages 201–224. Palgrave Macmillan UK, London, 1990.

[26] H. Heidenreich. Stemming? lemmatization? what? *Medium*, Dec 2018.

[27] R. Hernandez and I. S. Knight. Predicting myers-brigs type indicator with text classficiation. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6839354.pdf.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[29] J. Jackovich and R. Richards. *Machine Learning with AWS: Explore the Power of Cloud Services for Your Machine Learning and Artificial Intelligence Projects*. Packt Publishing, 2018.

[30] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[31] P. Jeatrakul, K. Wong, and C. Fung. Data cleaning for classification using misclassification analysis. *JACIII*, 14:297–302, 01 2010.

[32] M. Jolly. (mbti) myers-briggs personality type dataset. https://www.kaggle.com/datasnaek/mbti-type, Sep 2017.

[33] C. G. Jung, R. F. C. Hull, and H. G. Baynes. *Psychological types*. Princeton University Press, 1990.

[34] V. Kostov and S. Fukuda. Development of man-machine interfaces based on user preferences. In *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01) (Cat. No.01CH37204)*, pages 1124–1128, 2001.

[35] G. Lawrence and C. R. Martin. *Building people, building programs: a practitioner's guide for introducing the MBTI to individuals and organizations*. Center for Applications of Psychological Type, 2001.

[36] N. Lindsey. Invasion of privacy: Tracking your online behavior across the web. *CPO Magazine*, Dec 2017.

[37] W. H. Lindsey. *The relationship between personality type and software usability using the Myers-Briggs Type Indicator (MBTI) and the Software Usability Measurement Inventory (SUMI)*. PhD thesis, Nova Southeastern University, 2011.

[38] A. Long. Benchmarking python nlp tokenizers. *Medium*, Sep 2019.

[39] P. Ludford Finnerty and L. Terveen. Does an individual's myers-briggs type indicator preference influence task-oriented technology use? In *INTERACT '03*, 01 2003.

[40] A. Ma and G. Liu. Neural networks in predicting myers brigg personality type from writing style. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2736946.pdf.

[41] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[42] C. G. Miller. The volume of data is increasing. *Quality Magazine RSS*, Nov 2017.

[43] I. B. Myers. *MBTI manual: a guide to the development and use of the Myers-Briggs Type Indicator*. Consulting Psychologists Press, Inc., 1999.

[44] A. Nagle, R. Riener, and P. Wolf. Personality-based reward contingency selection: A player-centered approach to gameplay customization in a serious game for cognitive training. *Entertainment Computing*, 28:70 – 77, 2018.

[45] K. Nishida. Introduction to extreme gradient boosting in exploratory. *Medium*, Mar 2017.

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

[47] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1063–1072, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[48] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[49] J. Ramzai. Simple guide for ensemble learning methods. *Medium*, Mar 2019.

[50] C. S. Rebecca Walker Reczek and R. Smith. Targeted ads don't just make you more likely to buy - they can change how you think about yourself. *Harvard Business Review*, Apr 2016.

[51] R. Reinhold. Understanding the mbti ® and myers briggs personality types. *Myers Briggs Personality Type Dynamics*, Jan 2006.

[52] M. Sawant. Truncated singular value decomposition (svd) using amazon food reviews. *Medium*, Jul 2019.

[53] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020.

[54] T. Srivastava. Nlp: A quick guide to stemming. *Medium*, Aug 2019.

[55] K. Subaramaniam and O. Baker. Human personality types and software interface design: Hci from a different perception. *International Journal on Advanced Science, Engineering and Information Technology*, 1, 01 2011.

[56] N. Tyagi. Understanding the gini index and information gain in decision trees. *Medium*, Mar 2020.

[57] L. Valiant. *Probably approximately correct: nature's algorithms for learning and prospering in a complex world*. Basic Books, A Member of the Perseus Books Group, 2014.

[58] J. Wapner. He counts your words (even those pronouns). *The New York Times*, Oct 2008.

[59] Wes McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.