# Exploring text classification techniques to determine different personality types

Agata Logvin
*Student №*: 101164720

Alexei Tipenko
*Student №*: 100995947

Prepared for Data Mining I (STAT 4601/STAT 5703) course
taught by Dr. Mills at Carleton University in March 2020

## 1 Motivation and Research Problem

The topic of our research project is text categorization and document clustering. These are data mining concepts of supervised learning (classification/categorization) and unsupervised learning (clustering). We chose to investigate this problem from a supervised learning perspective, focusing on text categorization.

The goal of text categorization is to assign natural language texts to labels from a predefined set [1]. In the context of our project, the text documents are posts of users left on a forum and the categories — different personality types.

According to Myers-Briggs Type Indicator (MBTI), there exist sixteen distinct categories of personality types based across four axes: introversion (I) vs. extraversion (E), intuition (N) vs. sensing (S), thinking (T) vs. feeling (F), and judging (J) vs. perceiving (P) [2]. Combining an element from each of the axes gives a personality type, for example INTP and ESFJ. The text data was collected through *PersonalityCafe* [3] forum, which provides access to user comments along with their MBTI personality types.

Thus, the aim of our research project is to attempt to categorize user text-based comments to different MBTI personality types using data mining techniques.

## 2 Literature Review

Text categorization was introduced in the early 1960s, but grew its popularity in the 1990s due to the availability of more powerful hardware and increased use of electronic data storage [1]. At first, texts were categorized by the knowledge engineering (KE) approach, which required manually defining a set of rules to classify a document [1]. However, machine learning (ML) became favoured to this approach; ML classifiers performed at equal accuracy to those defined by experts, but did not require the same labour power as KE since it was an automated process [1].

To this day, text categorization has continued to evolve and is used in a wide variety of applications, such as flagging an email as spam, identifying the language of a text, classifying a book by its genre, and more [4]. In general, text categorization can be considered a binary, multiclass, or multilabel problem [1]. For binary categorization, an observation is classified into two labels; in the case of email, the interest lies in splitting spam and non-spam. When determining the language a text is written in, a multiclass problem is being solved, as the observation is assigned to one of *k* categories ($k > 2$). Multilabel categorization occurs when a document may be associated with one or more categories simultaneously. For instance, a book may be classified into the genres of romance and comedy, where romance and comedy are two separate labels. Multiclass and multilabel categorization problems are generally split into *k* binary classifications [1].

Under the machine learning approach, text categorization can be divided into three main categories: unsupervised learning, supervised learning, and semi-supervised learning [5]. To best summarize the methods and provide examples for each, **Figure 1** is presented below.

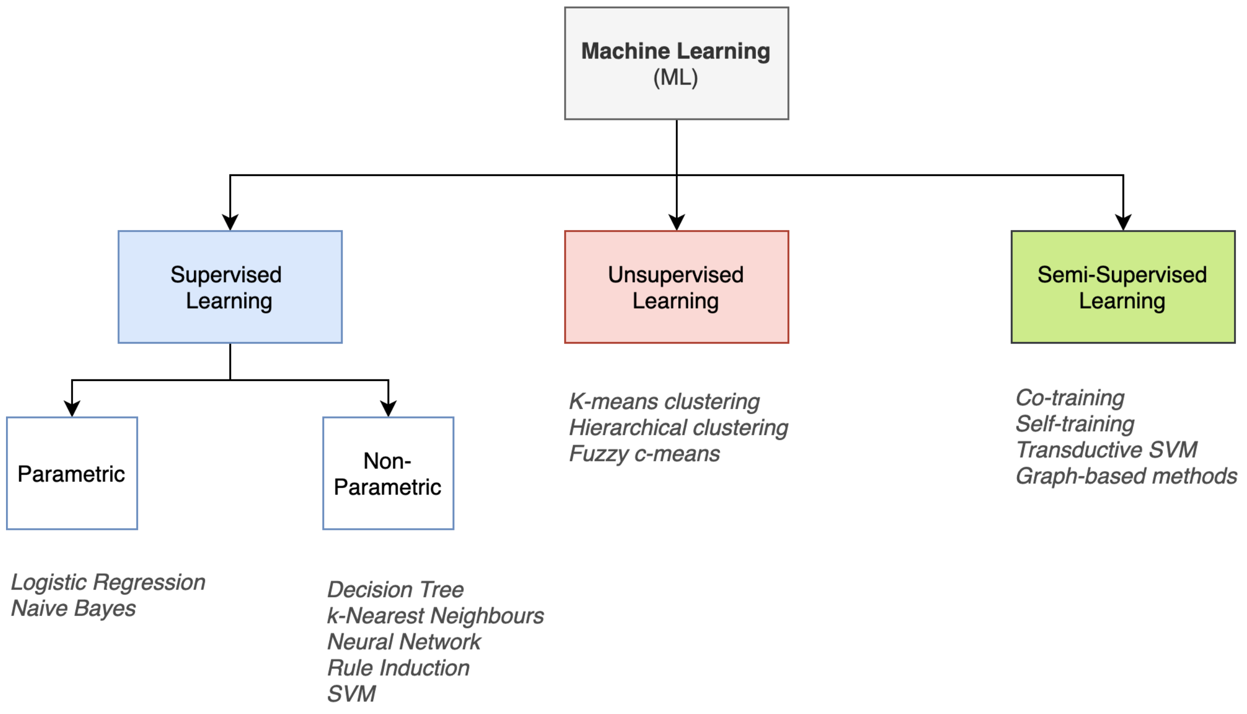

**Figure 1**. Summary of Machine Learning Methods for Text Categorization.

### 2.1 — Supervised Learning

Supervised learning Supervised learning is split into parametric and non-parametric methods for mining the data. As the categories in supervised learning are predefined by human or some other form of intervention, supervised learning is considered an expensive method [5].

Parametric models use a fixed amount of parameters to summarize the data, independent of the size of the data. [6] For example in logistic regression, the probability of response can be written as a function of $p$ parameterized by $\theta$, which is of fixed size: $P(Y = 1 \mid X = x) = p(x; \theta)$. Parametric models are simple to understand, they learn quickly from the data, and require less training data, but can provide a poor fit and are limited to solving simpler problems [7]. Examples of parametric machine learning algorithms include logistic regression and Naive Bayes, which will be defined in detail later; categorizing a movie review written by a critic as positive (1) or negative (0) can be done through binary logistic regression.

Non-parametric models do not have a fixed number of parameters and instead learn functional forms of the data, which is useful for large datasets with limited prior knowledge [6]. These methods are more flexible to parametric models and usually have higher prediction performance; however, they require more training data, risk overfitting it, and perform at a lower speed [7]. Some examples of non-parametric machine learning algorithms include decision trees, $k$-nearest neighbours, and neural networks.

### 2.2 — Unsupervised Learning

In unsupervised learning, the categories are not predefined, meaning the class an observation belongs to is unknown. Semi-supervised learning has a mix of some labeled observations and a larger proportion of unlabeled data; although this method is not the focus of this paper, it is interesting to note that semi-supervised learning may improve classification efficiency as it blends the advantages of supervised and unsupervised techniques [8]. Nevertheless, it is important to note that each machine learning method provides its own benefits and drawbacks.

### 2.3 — Challenges and Performance Measures

Text categorization is useful for knowledge discovery purposes, but presents its own set of challenges. Commonly, traditional text categorization methods rely on word frequencies or similarity between documents in order to perform analysis [9]. However, text document lengths vary according to their application; for example, a dozen words (user comment on a website), a few sentences (text message), and hundreds of pages (a novel). Documents of shorter length contain large amounts of spare data, which cannot be standardized due to misspellings and use of non-standard terms [10]. Thus, pre-processing of the data and feature extraction present challenges in this case. Special classifiers such as semi-supervised short text and real-time classifications may be used to analyze the data [10].

After text categorization implementation, performance measures of the classification rules are set and used for evaluation. A confusion matrix consisting of cells — true positive, true negative, false positive, and false negative — is commonly used and may be extended to include each classification level. According to each type of classification, whether binary, multiclass, multilabel, several performance measures have been developed; accuracy, precision, and recall (sensitivity), examples used for binary classification, are summarized in the table below [11].

## 3 Methodologies for Analysis

As previously stated, this paper focuses on supervised learning techniques for solving text categorization problems. This section will examine four different methods used for analysis, as well as a measurement of accuracy.

### 3.1 — Logistic Regression (LR) & LR with Principal Component Analysis (PCA)

*3.1.1 — Logistic Regression*

Logistic regression is a supervised method where the response variable is binary, meaning it is a binary classification problem. The probability of success is modelled by the logit equation.

$$ln(\frac{P(Y=1)}{1-P(Y=1)}) = \beta_0 + \beta_1 X_1 ... + \beta_p X_p$$

Parameters are commonly estimated by iterative procedures, such as iteratively reweighted least squares, to derive parameter estimates [12]; thus, this method is parametric as the number of parameters is fixed by the functional form. Once the estimates are obtained, the probability of success is estimated by the equation below.

$$\hat{\pi} = (1 + exp[-(\beta_0 + \beta_1 X_1 ... + \beta_p X_p)])^{-1}$$

It is important to note the fact that logistic regression is a linear classifier, which attempts to linearly separate the categories of the data. When datasets are large and of high dimension, linear classifier techniques work well; in some examined cases, logistic regression proves to be the best method for categorizing text data [5, 13]. Another advantage of logistic regression is that it not only predicts the label of an observation, but also provides an estimate of that observation's probability [14]. This is advantageous for interpretation and further model evaluation purposes.

*3.1.2 — Logistic Regression with Principal Component Analysis*

Principal component analysis is a traditional method for dimension reduction [14]. Principal components (PCs) depend on the covariance matrix of the $X_1, ..., X_p$explanatory variables and identifies mutually orthogonal directions of decreasing variance. The first of the PCs explains the most amount of variance, while latter PCs explain less variance; thus, by dropping latter PCs, the dimension is reduced.

Logistic regression can be combined with PCA to perform text categorization. Although newer techniques have been derived based on PCA, such as kernel principal component analysis (KPCA) and independent component analysis (ICA), they do not perform better than the traditional technique when assessing text categorization based on logistic regression [14].

## 3.2 — Bernoulli Naïve Bayes (NB) and Multinomial Naïve Bayes (MNB)

### *3.2.1 — Naïve Bayes*

The second parametric algorithm examined is Naïve Bayes (NB). This method is based on Bayes' Theorem from probability theory, stated below, with the added assumption that features ($X_j$) are conditionally independent given the target class ($Y$) [15].

$$P(Y \mid X) = \frac{P(X \cap Y)}{P(X)} = \frac{P(Y)P(X \mid Y)}{P(X)}$$

The assumption of conditional independence can be further written as:

$$P(X \mid Y) = P(X_1, \ldots, X_p \mid Y) = \prod_{j=1}^{p} \pi_{ki}^{X_j} P(X_j \mid Y)$$

Therefore, we can write the conditional probability of target on features as:

$$P(Y \mid X) \propto P(Y) \prod_{j=1}^{p} \pi_{ki}^{X_j} P(X_j \mid Y)$$

For categorical features, the required probabilities are commonly derived by frequency counts from the training data; when features are numerical, probability density estimation or discretization are employed [15]. Using marginal probabilities and conditional probability of $P(X \mid Y)$, the prediction of interest — $P(Y \mid X)$ — can be obtained. Note, NB models use methods of maximum likelihood [16]. Also, the target class ($Y$) can be extended to a vector in order to include multiple target classes — $Y_1, \ldots, Y_n$.

### *3.2.2 — Bernoulli Naïve Bayes*

The Bernoulli NB method is widely used for text categorization purposes [16]. The method is implemented when the features are independent binary variables; $X_j = 1$ if $j^{th}$term is present in the vocabulary and $X_j = 0$ otherwise. Since we have Bernoulli data, the NB model will be built on Bernoulli maximum likelihood, defined below.

$$P(X_1, \ldots, X_p \mid Y_k) = \prod_{j=1}^{p} \pi_{ki}^{X_j} (1 - \pi_{ki})^{1-X_j}$$

The $\pi_{ki}$represents the probability of the class $Y_k$ generating the term $X_j$. Bernoulli NB is useful for classifying short texts [16].

### *3.2.3 — Multinomial Naïve Bayes*

In Multinomial NB, the distribution of $P(X_1, \ldots, X_p \mid Y_k)$ is defined to be a multinomial [16]. In the context of text categorization, it captures information on the frequency of the different words in the text documents [16]. The likelihood of observing the data $X_1, \ldots, X_p$ is defined below; once the natural log of the data is taken, it can be expressed as a linear classifier.

$$P(X_1, \ldots, X_p \mid Y_k) = \frac{\left(\sum_j X_j\right)!}{\prod_j X_j!} \prod_{j=1}^{p} \pi_{ki}^{X_j}$$

$$ln\{P(Y_k \mid X_1, \ldots, X_p)\} \propto ln\{P(Y_k) \prod_{j=1}^{p} \pi_{ki}^{X_j}\} = ln\{P(Y_k)\} + \sum_{j=1}^{p} X_j ln(\pi_{ki})$$

One particular concern of the NB algorithm with text categorization is when a certain word is not observed in the training data; since maximum likelihood methods require multiplication of probabilities, there is a risk that this might be zero [17]. In order to avoid this situation, several smoothing methods have been suggested [17].

## 3.3 — Decision Trees & Random Forests

### *3.3.1 — Decision Trees*

Decision trees are a non-parametric method of supervised learning. A very basic example of a decision tree is in **Figure 2**, which demonstrates that this method is similar to how logical decisions are made by humans.

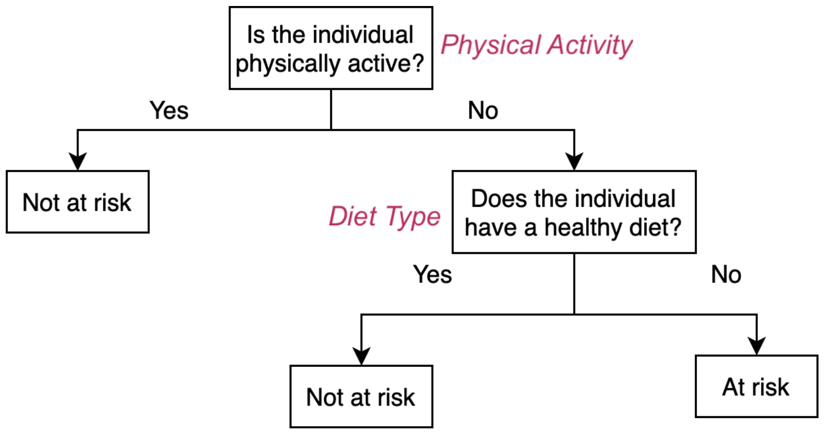
Is the individual physically active?
Physical Activity
Yes
No
Not at risk
Diet Type
Does the individual have a healthy diet?
Yes
No
Not at risk
At risk

**Figure 2**. A simple example of classifying an at risk group for obesity.

From the example above, each node (e.g. the rectangles) on the diagram represents a feature (e.g. $X_j$ in red) to be classified into a certain category; each branch (e.g. yes & no) is a value the node can assume [18]. The decision tree starts at a root node, which best divides the data, and has end nodes called leaves (e.g. at risk & not at risk) [18]. In text categorization, a document may be labelled by the decision tree into one of the leaf categories.

There are several methods that exist in order to identify which feature to select for the root node, such as information gain and gini index, however there is not one best method and different methods should be tested for a particular dataset [18]. However, once a specific method is obtained, it can be used to continue dividing each of the sub-trees.

One challenge of decision trees is overfitting the training data [18]. Overfitting is problematic because if a classifier fits training data perfectly, it may not classify test data well and thus will not be relevant for future work. There exist various approaches to avoid overfitting, one being to refuse a tree to grow to its full size, which is called pre-pruning [18]. Another challenge of decision trees is that they can be expensive and have difficulty when data is noisy [18].

### *3.3.2 — Random Forests*

As a forest is populated with trees, the idea of a random forest is built on the use of decision trees. In 2001, statistician Dr. Breiman proposed a classification and regression method that randomized decision trees and combined several of them to produce predictions [19]. This method has proved to be "extremely successful" and performs well even if data have a higher number of explanatory variables than observations [20].

The random forest algorithm presented by Breiman is based on the "divide and conquer" principle and can be broken down into three basic steps [19]:

1. Take samples of the data
2. Create a randomized decision tree predictor for each small sample
3. Aggregate the predictors together

Of course, random forest algorithms have been further developed and many other authors have published their own works since the first proposed random forest [20].

The benefits of random forests include that they have high accuracy and can be applied to a wide variety of problems [20]. Furthermore, they are powerful enough to deal with small sample sizes [20].

## 3.4 — *k*-Fold Cross-Validation

The algorithms described in *Sections 3.1* to *3.3*, as well as numerous others, can be used to perform text categorization; however, which is most suitable for a specific dataset? A statistical method that can be used to compare and evaluate different algorithms is called *k*-fold cross-validation. As an example, the method is illustrated in **Figure 3**.

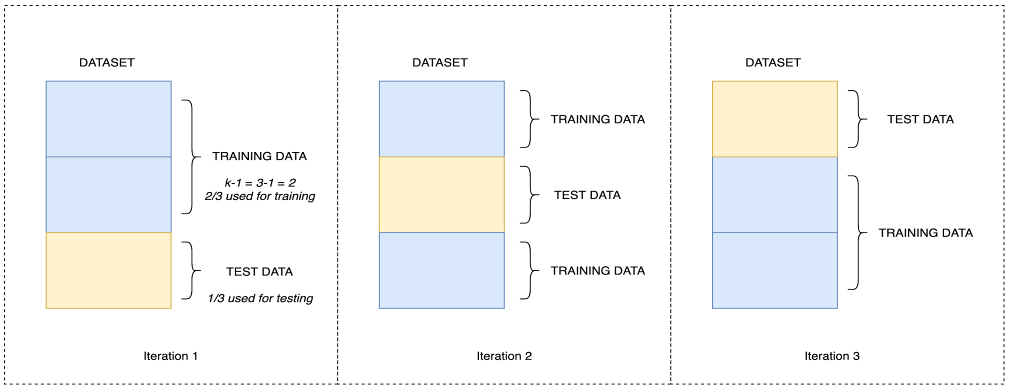

**Figure 3**. An example of 3-fold cross-validation (k = 3).

The figure above shows that the first step of *k*-fold cross-validation is that it divides the data into *k* sections of approximately equal size [21]. In this approach, (*k*-1) sections are used to train the model, while the remaining section is used to test and validate it [21]. In order to go through all the possible training-test combinations, this procedure is performed *k* times (i.e. *k* iterations).

Thus, each algorithm has *k* samples of performance metrics. In order to obtain an aggregate measure for an algorithm, different methodologies have been proposed; for example, averaging can be used [21].

## 4 Applications & Results

As previously mentioned in *Section 1*, this project attempts to categorize user posts to different MBTI personality types, with text data obtained from *PersonalityCafe* forum.

### 4.1 — Data Collection

A dataset for text categorization of MBTI personality types was found on *Kaggle* [22]. A corresponding dataframe was created by reading the contents of the `mbti_1.csv` file in Python, which was the programming language used for this investigation. To provide insight on the data, a few rows of the file are shown in **Figure 4**. The original file had 8675 rows of data with two columns (i.e. 8675 different users), the first being one of the sixteen possible personality types and the comments stored in the second column.

| MBTI Type | User Comments on the Forum (Last 50) |
|---|---|
| INTJ | 'Dear INTP, I enjoyed our conversation the other day. Esoteric gabbing about the nature of the universe and the idea that every rule and social code being arbitrary constructs created...\|\|\|Dear ENT... |
| ENTJ | 'Hello! I am working on a presentation by type. Part of each presentation is feedback from a range of people of the type being reviewed. I would greatly appreciate it if you could take a few...\|\|\|M... |
| INFJ | 'http://www.youtube.com/watch?v=qsXHcwe3krw\|\|\|http://41.media.tumblr.com/tumblr_lfouy03PMA1qa1rooo1_500.jpg\|\|\|enfp and intj moments https://www.youtube.com/watch?v=iz7lE1g4XM4 sportscenter not top t... |
| ... | ... |

**Figure 4**. Two rows of raw data from the file. The last 50 posts posted by a user appear in the second column, each individual comment separated by three pipe characters (|||).

*4.1.1 — Data Visualization*

Before cleaning the data and building machine learning algorithms, data visualization was conducted in order to see whether the comments were balanced for the sixteen MBTI types.

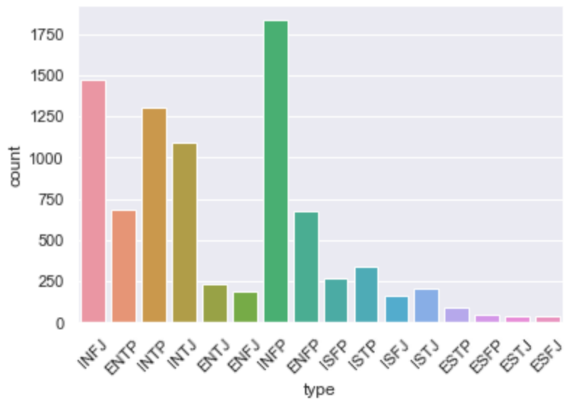

**Figure 5**. Frequency diagram of the number of posts for each personality type.

As seen from **Figure 5** above, there are certain personality types that post more frequently than others. While balancing and sampling are preferred, our analysis ignores the imbalance. These techniques could be attempted for future analysis, but based on our results, lack of balancing may not be significant. **Figure 6** further shows the number of posts for each of the four axes of the personality types.

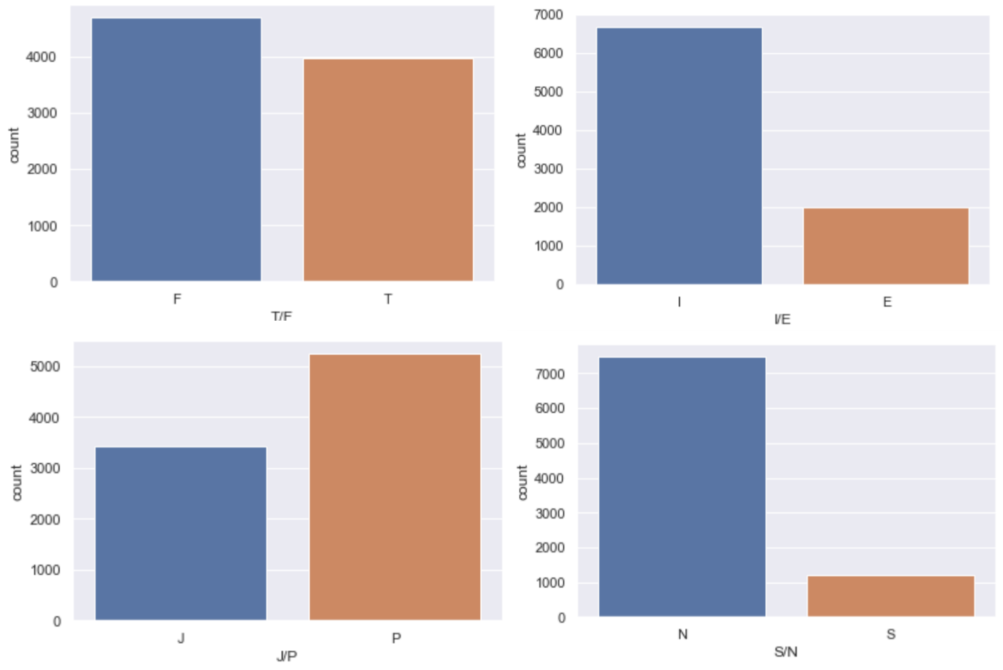

**Figure 6**. Frequency diagram of the number of posts for each axis.

*4.1.2 — Data Preparation (Missing Data and Data Cleaning)*

Since the data was collected from an outside source in raw form, it was crucial to examine its contents and perform data cleaning if required. First, the data was verified to check whether it contained any missing values or broken rows. Luckily, no missing data techniques had to be implemented in this case.

As with most text data, especially that collected from user comment, necessary data cleaning measures must be taken into place. As seen in **Figure 4**, some users posted links, which are not always useful for text categorization. The text data was cleaned using the *Natural Language Toolkit* (NLTK) library in Python, which focuses on:

- Removing punctuation and unnecessary symbols
- Removing all gibberish and stop words
- Tokenizing words
- Converting words to lowercase
- Lemmatizing words (i.e. change word to its root)

The resulting data contained rows of a single string, with the words separated by a space.

Once the data was clean, the next step focused on converting a matrix of text documents (in our case, matrix of words) to a matrix of token counts. The procedure was performed by the *scikit-learn* library from Python using the text feature extraction tool called "CountVectorizer". This conversion allows us to build a vocabulary of words and use it for new documents as well. For Bernoulli NB, the matrix would consist of 1's and 0's (word either present or not); however, for more accurate results it makes sense to have actual word counts.

In order to build a machine learning algorithm, training and test data sets are required. Consider the MBTI types as $Y$ and the user comments as $X$. Thus, the data was split into four matrices (*NumPy* arrays in Python) consisting of two category matrices and two comment matrices: train $X$, test $X$, train $Y$, test $Y$. An 80/20 split was chosen for the data, meaning 80% was used for training and 20% for testing.

**4.2 — Results**

Building our learning algorithm, we found that using 5 000 features (words) was sufficient; we also tested using 10 000 features. Best results were obtained using unigrams (e.g. "green" or "tree") as opposed to bigrams (e.g. "green tree") and trigrams, which performed much worse. This may be due to the fact that user comments are not focused on a specific topic, but rather are spare.  Upon creating the final training set with the token counts (train $X$), we created the final version of the test set (test $X$) which had to have the exact same token count columns; otherwise, there would be a mismatch between the two.

Next, PCA was performed on the training matrix.

```
PCA explained variance values:
[69.34 23.27 18.54 12.87 11.92  9.73  9.64  8.74  8.45  7.7   7.51  7.16
  6.34  6.13  5.83  5.46  5.16  5.02  4.88  4.75  4.53  4.42  4.29  4.15
  3.98  3.94  3.72  3.65  3.6   3.5 ]
```

**Figure 7**. PCA on training matrix for the top 30 principal components.

From **Figure 7**, it is clear that approximately 70% of the variation in the dataset can be explained by the first component, and subsequently, about 23% by the second. This gives insight on the fact that PCs can be used as input for some classification algorithms, which benefit from the ortogonal nature of the components. Later, our results show the algorithm for which it is advantageous to use PCs is logistic regression.

As stated in *Section 3*, six different machine learning algorithms were used to perform text categorization:

- Logistic Regression (*LR*)
- Logistic Regression with PCA (*LR PCA*)
- Bernoulli Naïve Bayes (*BNB*)
- Multinomial Naïve Bayes (*MNB*)
- Decision Tree (*DT*)
- Random Forest (*RF*)

Once the different models were built, it was time to compare the accuracy, runtime, and other statistics of the algorithms. Thus, *k*-fold cross validation was used from the *scikit-learn* library. Cross validation was performed for all sixteen personality types ($Y_1, \dots, Y_{16}$), as well as each of the four axes (binary $Y$), producing five sets of results for each of the six algorithms. A full set of results is available from running the code, with a portion of it shown in **Figure 8**. In our case, we chose *k* = 5.

```
Running 5-fold cross validation for all 16 personality types ...\
\ Statistics for Multinomial Naive Bayes:
\ Mean       = 0.5436599423631124
\ Std. Dev.  = 0.015109771445438994
\ Minimum    = 0.5230547550432276
\ Maximum    = 0.5655619596541787
\ Avg time   = 0.2899421215057373\
\ Statistics for Bernoulli Naive Bayes:
\...
Running 5-fold cross validation for introversion/extraversion ...\
\...
```

**Figure 8**. Part of output of running 5-fold cross-validation. A total of 30 (6 algorithms ✕ 5 label types) statistics tables produced.

The results show a very low accuracy when using sixteen different MBTI labels; a prediction rate of approximately 50%, which is shown by the mean value in the statistics table. These mean values are summarized in **Figure 9**. Logistic Regression with the top 50 principal components gave us the best results. This is likely due to the fact that Logistic Regression is very sensitive to multicollinearity between variables (word tokens); by using PCA, this problem is resolved, since the components are orthogonal with each other.

| LR | LR PCA | BNB | MNB | DT | RF |
|---|---|---|---|---|---|
| 55.86 % | 58.85 % | 45.13 % | 54.37 % | 42.58 % | 45.91 % |

**Figure 9**. Summary of mean prediction rate for predicting sixteen different personality types.

Examining results of different axes yields much better results. The mean prediction values for the four different axes are summarized in **Figure 10**.

| Axis | LR | LR PCA | BNB | MNB | DT | RF |
|---|---|---|---|---|---|---|
| I/E | 80.89 % | 83.01 % | 76.35 % | 79.84 % | 76.24 % | 77.77 % |
| S/N | 86.96 % | 87.09 % | 80.62 % | 86.25 % | 82.00 % | 86.11 % |
| T/F | 79.48 % | 82.51 % | 78.88 % | 81.96 % | 69.26 % | 79.99 % |
| J/P | 71.96 % | 76.48 % | 66.10 % | 73.11 % | 68.40 % | 72.12 % |

**Figure 10**. Summary of mean prediction rate for predicting one of the axes.

For most cells in **Figure 10**, the values are quite high; specifically for N (intuitive) vs. S (sensor), all prediction rates have above 80% accuracy.

In addition to the *k*-fold cross validation, results for each machine learning algorithm were produced, displaying training and test accuracies and evaluation metrics (precision, recall, f1-score, and support), as well as a confusion matrix; full results can be obtained by running the code.

For the LR PCA model, training and test accuracies are approximately 62% and 58%, respectively. Although these results are quite low, it is important to note that we are attempting to classify sixteen different categories, not just two. However, this model does a fairly good job at classifying users on individual axes as shown in **Figure 11**. The test accuracies are very close to the training accuracies, implying there is very little overfitting happening.

| Axis | Training accuracy | Test accuracy |
|---|---|---|
| All 16 personality types | 61.45 % | 58.16 % |
| Introversion vs. Extraversion | 83.63 % | 82.82 % |
| Sensing vs Intuitive | 87.77 % | 87.32 % |
| Thinking vs Feeling | 82.82 % | 82.88 % |
| Judging vs Perceiving | 77.09 % | 75.27 % |

**Figure 11**. Results summaries for LR PCA model.

More advanced algorithms may need to be used to ensure a better classification, such as ensemble methods (combining various model results together) and neural networks. Furthermore, it may have also been helpful to balance our data or create a representative sample. From **Figure 5**, it is clear that the proportion of comments from introverts (I) is higher than that of extraverts (E); also, there are more users who are intuitive (N) than sensors (S). Nevertheless, balancing or sampling our data may not have yielded significantly better results.

All of our tested algorithms gave best results on the axis of N vs. S, the most unbalanced pair. Despite the unbalanced dataset, our best results come from comparing these two groups directly.

## Citations:

[1] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Journals: ACM Computing Surveys*, 34(1).

[2] The Myers & Briggs Foundation. (2020). MBTI® Basics. Retrieved from https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1

[3] PersonalityCafe. (2016). Myers Briggs Forum. Retrieved from https://www.personalitycafe.com/myers-briggs-forum/

[4] Aggarwal, C. C., & Zhai, C. (2012). A Survey of Text Classification Algorithms. *Mining Text Data*, 163–222.

[5] Thangaraj, M., & Sivakami, M. (2018). Text classification techniques: A literature review. *Interdisciplinary Journal of Information, Knowledge, and Management*, 13, 117-135.

[6] Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach (3rd Edition).* Pearson.

[7] Brownlee, J. (2016). Parametric and non-parametric machine learning algorithms. Retrieved from http://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms

[8] Pavlinek, M., & Podgorelec, V. (2017). Text classification method based on self-training and LDA topic models. *Expert System with Applications*, 80, 83-93.

[9] Rafeeque, P. C., & Sendhilkumar S. (2011). A survey on Short text analysis in Web. *Advanced Computing (ICoAC), 2011 Third International Conference on IEEE*, 365-371.

[10] Song, G., Ye, Y., Du, X., Huang, X., & Bie, S. (2014). Short Text Classification: A Survey. *Journal of Multimedia*, 9(5), 635-643.

[11] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.

[12] Agresti, A. (2013). *Categorical Data Analysis, 3rd Edition*. John Wiley & Sons.

[13] Pranckevičius, T., & Marcinkevičius, V. (2017). Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, 5(2), 221-232.

[14] Musa, A. B. (2014). A comparison of ℓ1-regularization, PCA, KPCA and ICA for dimensionality reduction in logistic regression. *International Journal of Machine Learning and Cybernetics,* 5, 861–873.

[15] Webb, G.I. (2011). Naïve Bayes. *Encyclopedia of Machine Learning*. Springer, 30-45.

[16] McCallum, A., & Nigam, K. (1998). A comparison of event models for Naïve Bayes text classification. *Association for the Advancement of Artificial Intelligence-98*.

[17] He F., & Ding X. (2007). Improving Naive Bayes Text Classifier Using Smoothing Methods. *Advances in Information Retrieval, Lecture Notes in Computer Science*, 4425, 703-707.

[18] Kotsiantis, S.B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 31, 249–268.

[19] Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5-32.

[20] Biau, G., & Scornet, E. (2016). A random forest guided tour. *TEST*, 25, 197–227.

[21] Refaeilzadeh P., Tang L., & Liu H. (2009). Cross-Validation. *Encyclopedia of Database Systems*. Springer, 24-154.

[22] (MBTI) Myers-Briggs Personality Type Dataset. (2018). Kaggle. Retrieved from https://www.kaggle.com/datasnaek/mbti-type#mbti_1.csv