Predicting MBTI using Multi-layer Perceptrons

Introduction

The Myers-Briggs Type Indicator is one of a number of pseudoscientific measures in the field of personality theory, the foundation of which is in Carl Jung's work in cognitive psychology and psychological types. The 16 MBTI types are well-defined but unfortunately rely mainly on self-reporting, which is inherently biased.

Objectives

With this project, I sought to improve upon current methods of MBTI classification by using machine learning to build a model that takes in text (in this case, posts from an online forum for personality theory) and returns a prediction of the most likely personality type of the user or writer of that text. The production of an accurate natural language based classifier would be a significant advancement in the field of personality psychology, indicating a demonstrable connection between natural language and personality type.

My goal is to build a model that can be applied to social media data from a user and return that user's MBTI type. Within a business scenario, a system like this can be used for team organization or placement, so a manager can assemble teams with complementary skills and communication styles in order to encourage better productivity and a more positive working experience for those involved. Conversely, the system could also be applied to customers, scraping data from online reviews and comments and using that to segment the customer base by type or by specific dimensions (I/E, N/S, F/T, J/P) to assist with targeted marketing, research, and product development, and improve the overall customer experience.

Possible applications

A text-based classifier can be applied directly to data scraped from social media profiles in order to predict the type of those users. It can be applied in a number of situations, including for talent acquisition and recruitment, as well as for sociological purposes, ie. a study or analysis of type distributions in a certain population.

Applying a similar classifier to other data sources is also possible, but would likely require different training data that is more closely related to the genre of the input text. The model for this project was trained specifically on forum posts, which naturally would have a different syntax and reading complexity level than, say, a thesis paper. This particular model would likely only be effective with a certain range of genres, like posts from other forums, social media, blog posts, or possibly even transcriptions of casual conversations. It would not be as effective if used for other purposes, but I believe the basic concepts would be similar and can also be applied to other data sources.

Dataset

The data for this project were obtained from a Kaggle dataset with nearly 9000 samples from personalitycafe.com, each consisting of a person's MBTI type and a set of text written by that person. Larger datasets which would be usable for this purpose do exist, for example this 600MB dataset

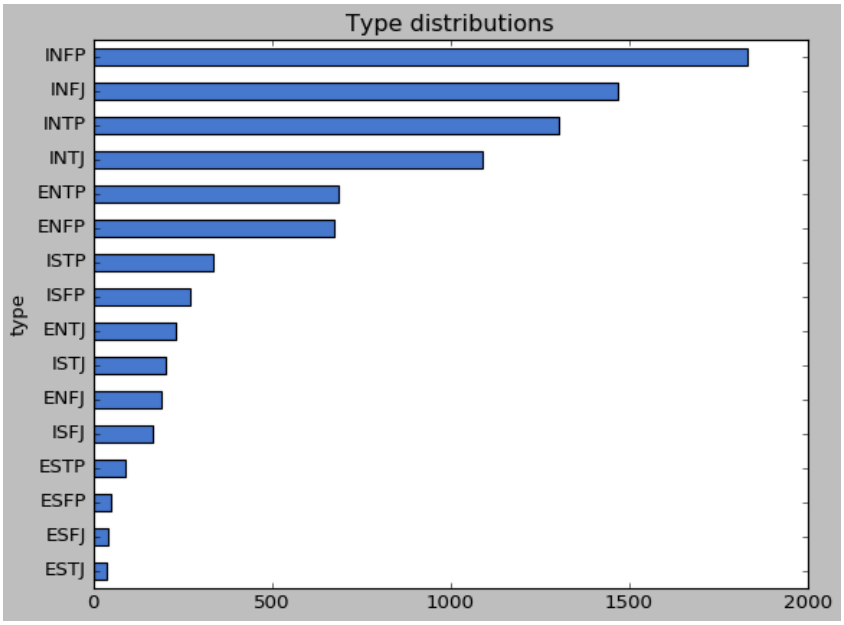
containing over one million posts scraped from Reddit, but due to technological limitations I worked with the smaller dataset (26MB) for this project. This project could be done again using the larger dataset or by scraping data from other social media sites, for example, Twitter posts, or even Instagram or Tiktok comments, but this would require the collection of each user's actual MBTI type by other means.

Data wrangling and cleaning

Problems specific to the data used for this project were non-uniform distributions (Fig. 1), which were extremely disproportionate to the generally accepted population distributions, and the presence of non-meaningful text, such as stopwords and hyperlinks.

Class imbalance

Fig. 1: Original class distribution



For the first problem, one option would have been to artificially reproportion the dataset to accurately reflect the general population, but I found that this had a huge negative impact on model performance. This technique might be possible given a larger dataset, since with this data it required either dropping a huge number of samples from overrepresented types and/or resampling a huge number of samples from underrepresented types. I chose to use coefficient-based class normalization, which resized classes based on how out of scale they were in relation to other classes in the dataset itself.

Cleaning text

For the second problem, I cleaned the data by first dropping non-meaningful text, then lemmatizing the text to correct for inflection when the same word appears in different tenses. I considered also dropping explicit mentions of personality types from the data, in case the model tried to learn to predict type by recognizing direct mentions, but there was no obviously discernible pattern as to whether these mentions

were more often someone mentioning their own type or someone mentioning another type, so it seemed that keeping this data could still lend some value to the model.

Other changes

Other modifications made to the dataset before modeling included building in additional variables to the vectorized text data, including variables for emotions (anger, fear, joy, love, sadness, surprise), polarity, and subjectivity (see "Engineering Emotion Features" on page 7).

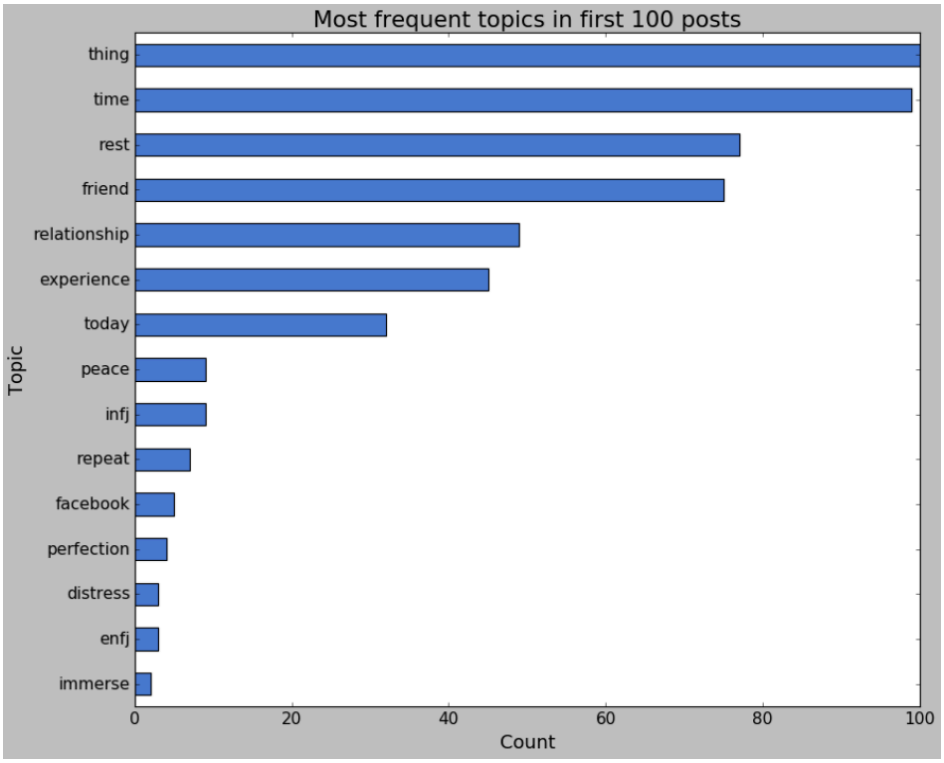
The emotion vectors were created using a emotions analysis model (logistic regression) built on a separate dataset assigning emotion labels to text samples, which was then applied to the type dataset. The polarity and subjectivity vectors were created using TextBlob's "sentiment" method.

EDA

Topic modeling

I started with topic modeling on the first 100 samples, which gave me a quick look at what topics seemed to be most common in the dataset (Fig. 2). Some were less interesting ("thing" or "today" seemed fairly vague) but the high frequency of topics like "time", "relationship", and "experience" gave an interesting look at what kinds of discussions the forum as a whole tended to lean toward.

Fig. 2: Topic mentions in first 100 samples



Overview of topic modeling by type

Breaking down the topic modeling by type also led to some interesting discoveries and appeared to give credence to the binary dimensions of the MBTI, especially (or most obviously) the intuition/sensing and feeling/thinking dimensions.

It was immediately evident that the eight intuitive types favored more abstract topics and topics relating to the internal facets of the human experience, with words like “theory”, “humanity”, “relationship”, “peace”, and “interest” appearing high in their frequency charts.

This verified the basic expectation that the data would reflect Jungian definitions of the four personality dimensions used in MBTI. Intuitives, according to Jung, look inward for guidance, consider theory as or more valuable than experience, and tend toward discussions of the abstract, such as feelings, theories, the future, and unrealized possibilities.

Sensors, on the other hand, are said by Jung to prefer using external data to understand themselves and the world, place far less value on theory than on fact and experience, are strongly rooted in the present and have difficulty assessing future possibilities, and tend toward discussions of the concrete, such as physical experiences, memory, and the past.

Topics like “power”, “order”, “food”, “money”, “drink”, “body”, and “action” appeared high in the frequency charts for the eight sensing types.

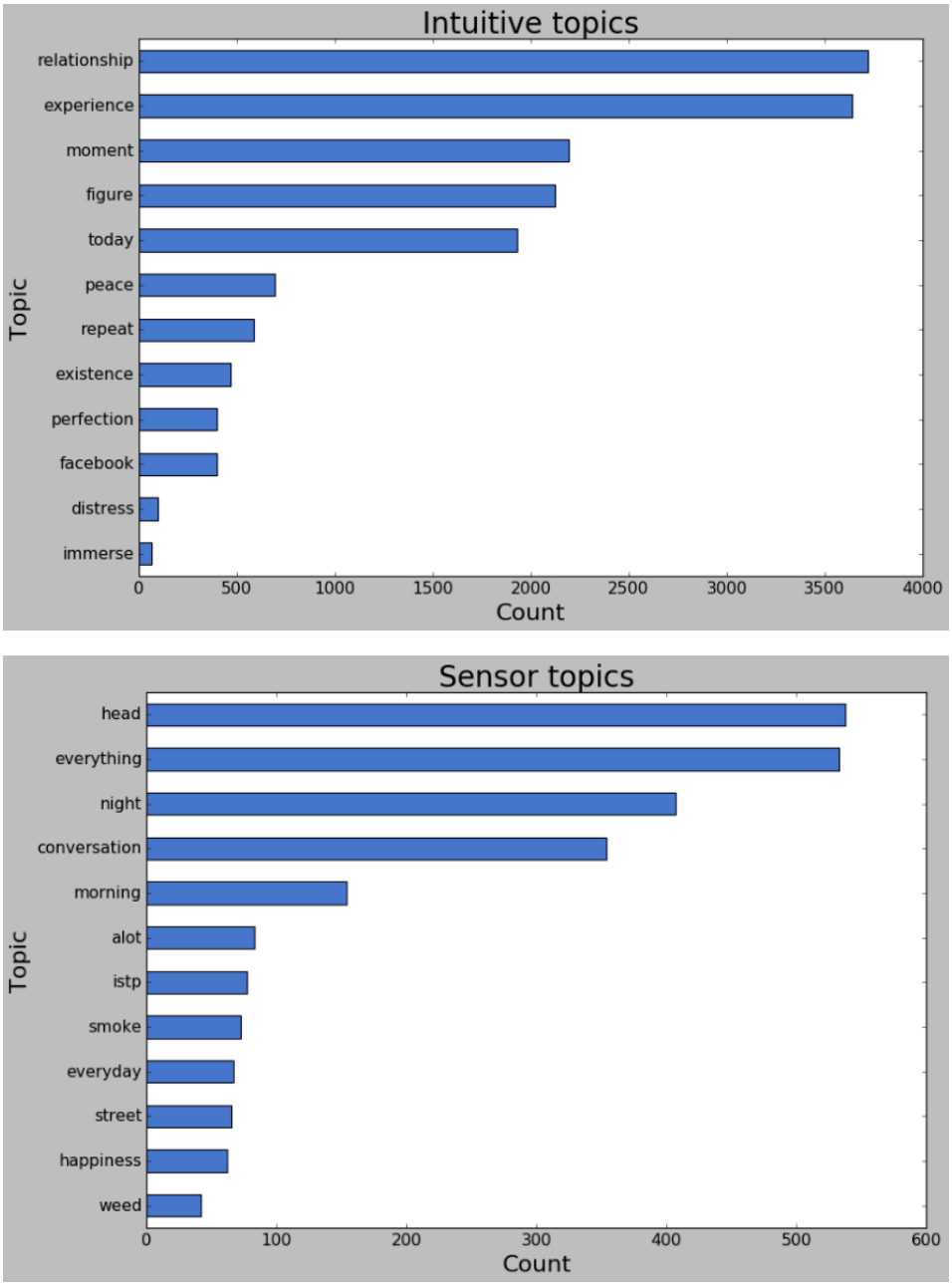
Clarifying topic modeling with binary comparisons

There was some overlap in topic frequencies between types, which might be attributed to the overarching bias in the data, which was sourced from a forum dealing specifically in personality theory. For example, this likely meant a majority of the users on the site, intuitive or sensing, would lean toward an interest in the abstract to begin with and naturally gravitate to those topics.

Intuitives vs. sensors

Directly dividing the data into intuitives and sensors gave a slightly clearer look at the differences between the topics each group favored (Fig. 3). Intuitives frequently mentioned relationships, experiences, peace, existence, and perfection, while sensors frequently discussed conversations, times of day, and physical/sensory topics like smoking and/or smoking weed.

Fig. 3: Topic modeling, intuitives vs. sensors

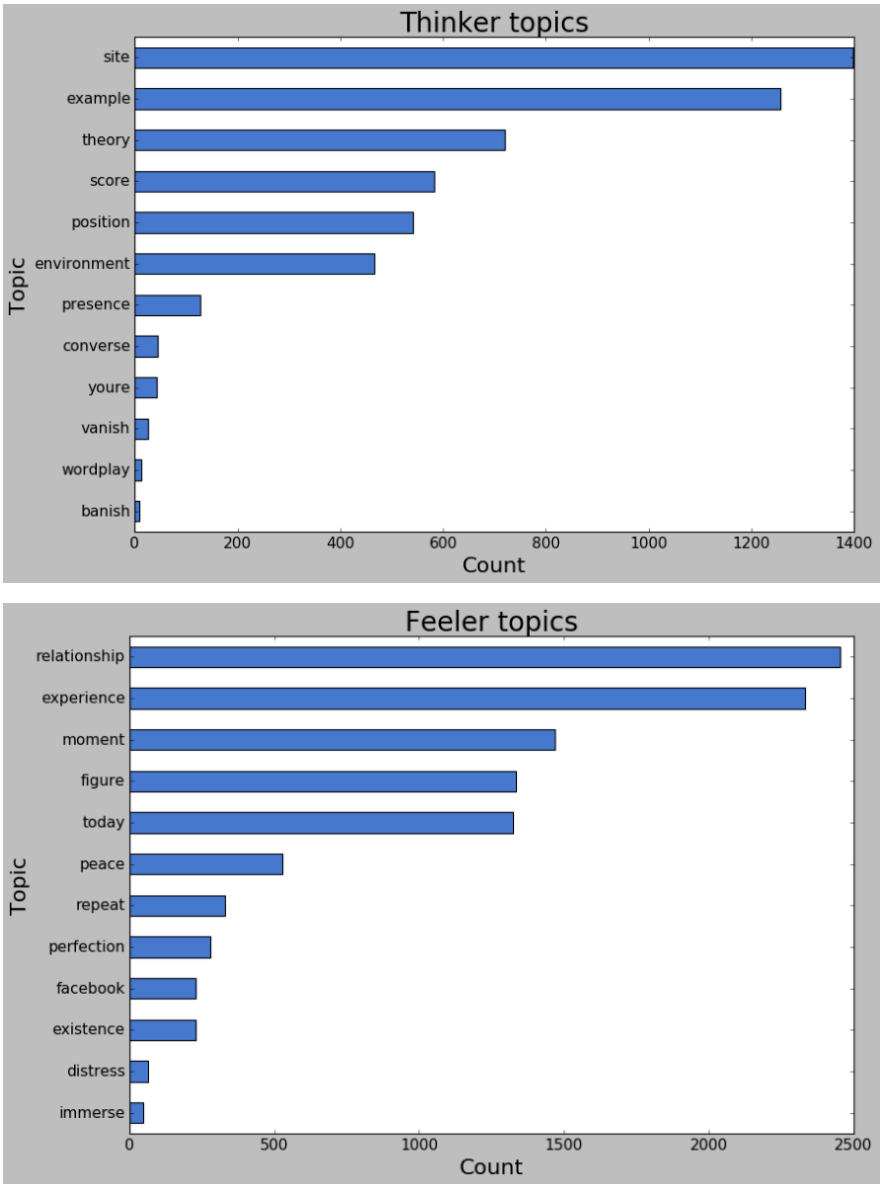


Thinkers vs. feelers

According to Jungian psychology, the divide between thinkers and feelers (T and F) should be most evident in their values. Thinkers are said to prioritize addressing and resolving problems, and only afterward consider whether their decisions should be adjusted according to their feelings or those of others. Feelers are said to prioritize the protection and valuation of their feelings and those of others, and then address and resolve problems after those boundaries have been set. A common misconception is that thinkers cannot feel and feelers cannot think, but it is moreso a question of priority.

In this dataset, thinkers had a high frequency for topics like “question”, “function”, “example”, “theory”, “score”, “action”, “rule”, “power”, “money”, and “intention”. Unlike the feelers’ preferred topics, these topics are directed toward the logic and structure of the world.

Fig. 4: Topic modeling, thinkers vs. feelers



Feelers, on the other hand, had a higher frequency for topics like “care”, “everyone”, “conversation”, “life” “appreciate”, “stress”, “relationship”, “hate”, and “feeling”. These topics seem to center around the people of the world rather than the world itself, including their relationships, their values, and their feelings. Breaking the data in half into thinkers and feelers made this difference even clearer (Fig. 4).

Introversion vs. extraversion & prospecting vs. judging

The differences between introverts vs. extraverts and judgers vs. prospectors were less defined and seemed to reflect the class distribution more so than individual types or subgroups. For example, introverted intuitives heavily outweighed introverted sensors, so topic modeling on the introverts subgroup heavily overlapped with the results obtained from topic modeling on the intuitives subgroup.

As far as theory, introverts are said to value time alone for rejuvenation and inspiration, while extraverts find more value in time spent with others. Conversely, introverts would consider time with others somewhat less valuable and are more likely to find it mentally or emotionally draining, and extraverts would consider time alone more of a waste and find it draining.

Prospectors are said to value possibility over predictability and control, while judgers prefer to exert more control over their decisions, environments, and relationships. Prospectors are more comfortable making decisions in the moment, with regard to the feelings and needs of the present, while judgers have difficulty making such decisions, and require an interval of planning to analyze the circumstances and determine what choice will best fulfill their needs.

Engineering emotion features

Dataset

For this analysis, I used a separate dataset consisting of text samples, with each sample labeled as one of 6 "classes" of emotion: anger, fear, joy, love, sadness, and surprise. Although these labels are relatively limited in terms of the range of possible human emotion, I intended to use it only to create a predictive model that could produce "scores", or vectors, for each emotion class, and then add those emotion vectors as new predictors in the MBTI data.

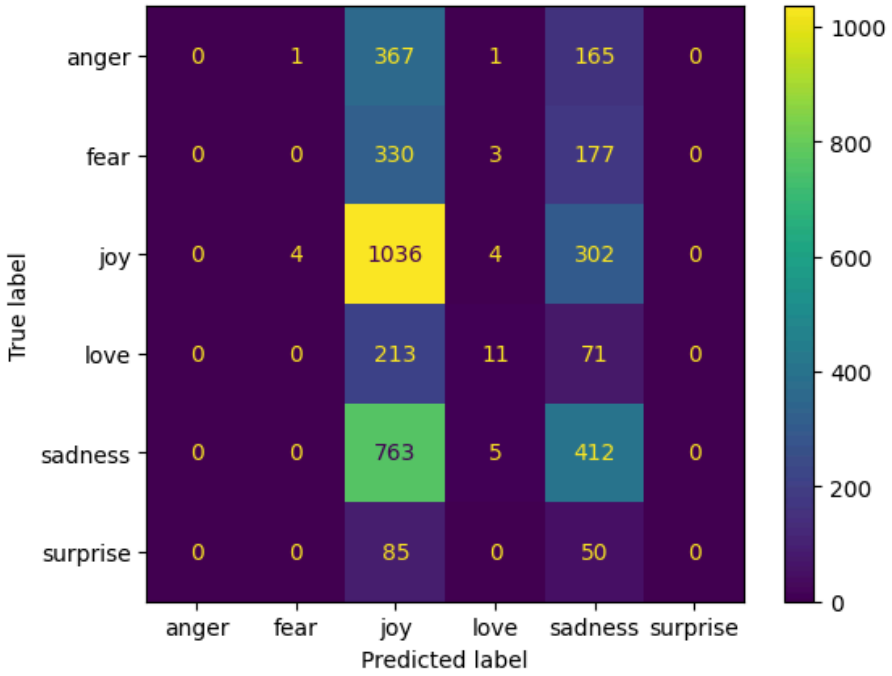
The structure of this dataset closely resembled the MBTI dataset. While the MBTI data were pulled from personalitycafe.com, the text samples for the emotions dataset were scraped from Twitter and emotion labels were assigned using a graph-based algorithm to extract emotion-relevant features and pattern-based emotion features derived from various open-source conventional and deep learning models used for text-based emotion recognition.

As a result of the work done to build this dataset, I was able to engineer a set of emotion features for the MBTI data that proved very useful in analysis and modeling.

Modeling and feature engineering

To engineer the emotion features used in modeling, I ran a separate emotions analysis. As the confusion matrix in Fig. 5 shows, the final model used for feature engineering was strongest on predicting joy and sadness, and gave comparatively few predictions for the other emotions, partly due to a class imbalance in the data (joy and sadness represented about 60% of the data that were set aside for testing).

Fig. 5: Emotion analysis model, confusion matrix



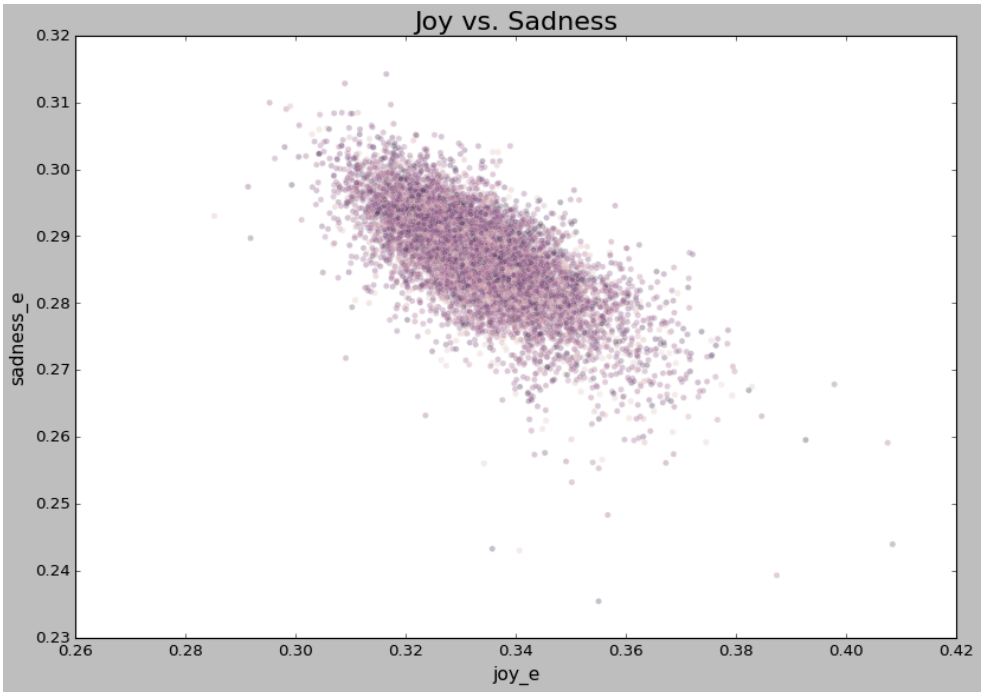
The overall accuracy of the model was 36.5%. In comparison, a random baseline would only give about 16.7% accuracy. I put this model to its intended use and applied it to the MBTI dataset so that each sample was given an additional 6 features scoring the text on each of the 6 emotions. These predictors were added directly to the TF-IDF matrix and later passed into feature importance analysis and working on the final model.

Confirming usefulness

The value of this emotions analysis (and the original emotions dataset) was also confirmed by the correlation between different emotions. Fig. 6 shows the correlation between probability predictions on text samples labeled “joy” and “sadness” and demonstrates a clear negative correlation. In other words, the emotional analysis model recognized that as the “joy” score of a specific text sample increased, the corresponding score for sadness decreased, and vice versa.

This supported the ability of the model to predict at least these two emotions, and proved that it had, to a certain extent, learned to recognize the relative semantic difference between happiness and sadness in natural language.

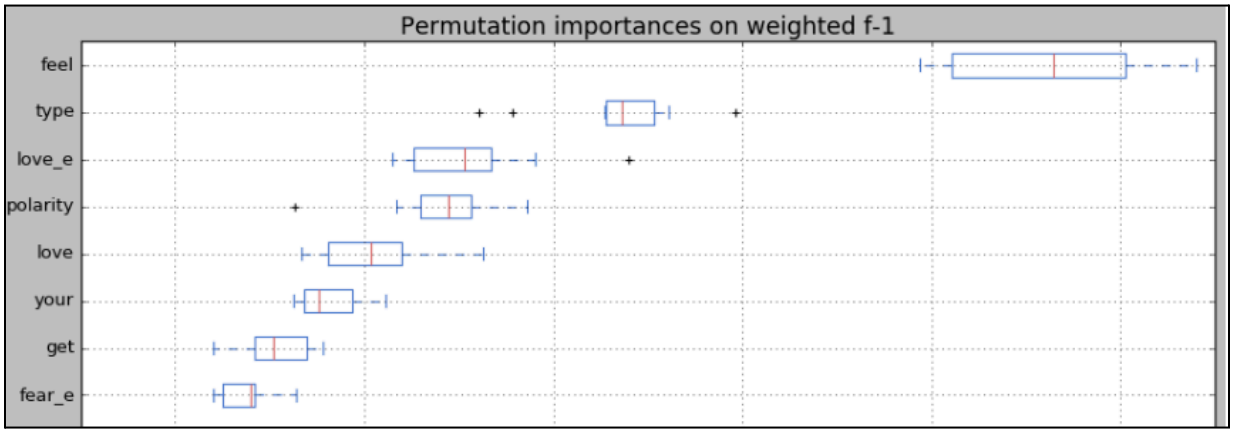
Fig. 6: Joy and sadness, negatively correlated



Random Forest feature importance

As my next step in EDA, I ran a permutation importance analysis on a subset of the training data (validating on another subset of the training data) to analyze which features seemed to be most useful in predicting MBTI (Fig. 8). Love (one of the newly created emotion vectors), polarity, and fear (another of the emotion vectors) all scored high in permutation importance, confirming the utility of these features and the emotions analysis conducted earlier.

Fig. 8: Permutation importance analysis



Preprocessing and TF-IDF

When approaching preprocessing and modeling, I chose to use a TF-IDF (term frequency and inverse document frequency) approach which, in contrast to a bag-of-words approach, takes into account the importance of words as well as information on the count of occurrences in each sample. This preserved the syntactic and semantic aspects of the original text samples, but required a little more processing.

First, the text had to be cleaned, which included removing URLs and non-English words and symbols, including the delimiters that had been added to the original text to separate individual posts within each sample ("|||"). Then the text was lemmatized and converted into a TF-IDF matrix using scikit-learn's TfidfVectorizer. The newly engineered emotion, polarity, and subjectivity features were then added to the matrix, as well as the original MBTI labels after one-hot encoding.

As part of preprocessing, I also corrected for the class imbalance using a coefficient-based class normalization method, downsampling on overrepresented classes and upsampling minority classes to achieve a more balanced set. The final dataset that was passed on to model selection and testing consisted of 8664 samples (only about 100 fewer than the original dataset), 108 predictors, and 16 class labels (one for each MBTI type).

Model selection

I experimented with a variety of types of classifiers, including a Random Forest classifier, a K-Nearest Neighbors classifier, a Multinomial Naive Bayes classifier, and a Multi-layer Perceptron network. For assessing model performance I used accuracy as the metric for comparison. As stated above, the baseline for comparison was 16.7% accuracy for a completely random prediction.

Initial Testing

The table in Fig. 9 compares the baseline performance of each classifier on the training data before hyperparameter tuning.

Fig. 9: Model selection, initial performance

Classifier	**Score**
Random Forest	**53.6**
K-Nearest Neighbors	**33.5**
Naive Bayes	**20.6**
MLP (Multi-layer Perceptron)	**52.3**

This initial testing showed that the problem was too complex for a Naive Bayes algorithm and too high-dimensional and/or imbalanced for KNN. Since I had used TF-IDF rather than bag-of-words it is possible that the predictors are not as independent as a Naive Bayes classifier would be assuming. Based on this preliminary modeling, I chose to move forward with tuning the Random Forest and MLP models.

Metrics

After initial tests on a range of different algorithms, model selection was done using balanced accuracy, also known as per-class accuracy, as the scoring metric, due to the characteristics of the data. Regular accuracy compares correct predictions against all predictions and does not take the number or size of classes into account, which risks tuning the model to simply favor the majority classes and neglect to learn the minority classes completely.

Since this dataset consisted of 16 classes which even after class normalization were considerably different in size, I used balanced accuracy instead to measure performance in the grid search and later on in analyzing the final model. This takes into account any imbalance in the dataset and calculates the average of correct predictions across all of the classes, without weighting any classes more heavily than others, regardless of size.

This way, the evaluation metric is not biased toward any of the 16 types in particular and ensures that the model is truly learning how to predict all MBTI types. The balanced accuracy metric is a better representation of how well the classifier is really performing across all classes.

Hyperparameter Tuning

Both the Random Forest and MLP classifiers achieved more than 50% accuracy before hyperparameter tuning. These were the two classifiers I took forward into hyperparameter tuning.

Initially, I ran the grid searches for both classifiers using accuracy as the scoring metric (see “Metrics”) but later, after reflecting on the nature of the dataset, returned to this step and tuned both classifiers again using balanced accuracy as the scoring metric to observe whether changing the scoring metric would affect the performance of either model.

The Random Forest grid search returned nearly the same parameters and the same score for balanced accuracy. The MLP grid search returned slightly different parameters for the model and the performance also increased by a very small margin. Fig. 10 shows the difference in scores after applying the tuned model to the withheld test data, as well as the difference in the hyperparameters that were returned by the grid search.

Based on these results, I moved forward with the MLP classifier, which achieved a 58% balanced accuracy on the withheld test data, as the final model. While the Random Forest classifier’s performance had seemed better during initial testing, the MLP classifier responded much more strongly to hyperparameter tuning and ultimately performed about 4.5% better.

Fig. 10: Model selection, hyperparameters and scoring

Classifier	**Accuracy**	**Hyperparameters**	**Balanced accuracy**	**Hyperparameters**
Random Forest	**53.57**	bootstrap = True max_depth = None max_features = ‘auto’ min_samples_leaf = 1 min_samples_split = 2 n_estimators = 200	**53.57**	bootstrap = True max_depth = None max_features = ‘sqrt’ min_samples_leaf = 1 min_samples_split = 2 n_estimators = 200
MLP (Multi-layer Perceptron)	**57.10**	activation = ‘tanh’ alpha = 0.01 early_stopping = True hidden_layer_sizes = (100, 100, 100) learning_rate = ‘constant’ max_iter = 2500 solver = ‘lbfgs’	**57.97**	activation = ‘relu’ alpha = 0.0001 early_stopping = False hidden_layer_sizes = (100, 50) learning_rate = ‘constant’ max_iter = 10000 solver = ‘adam’

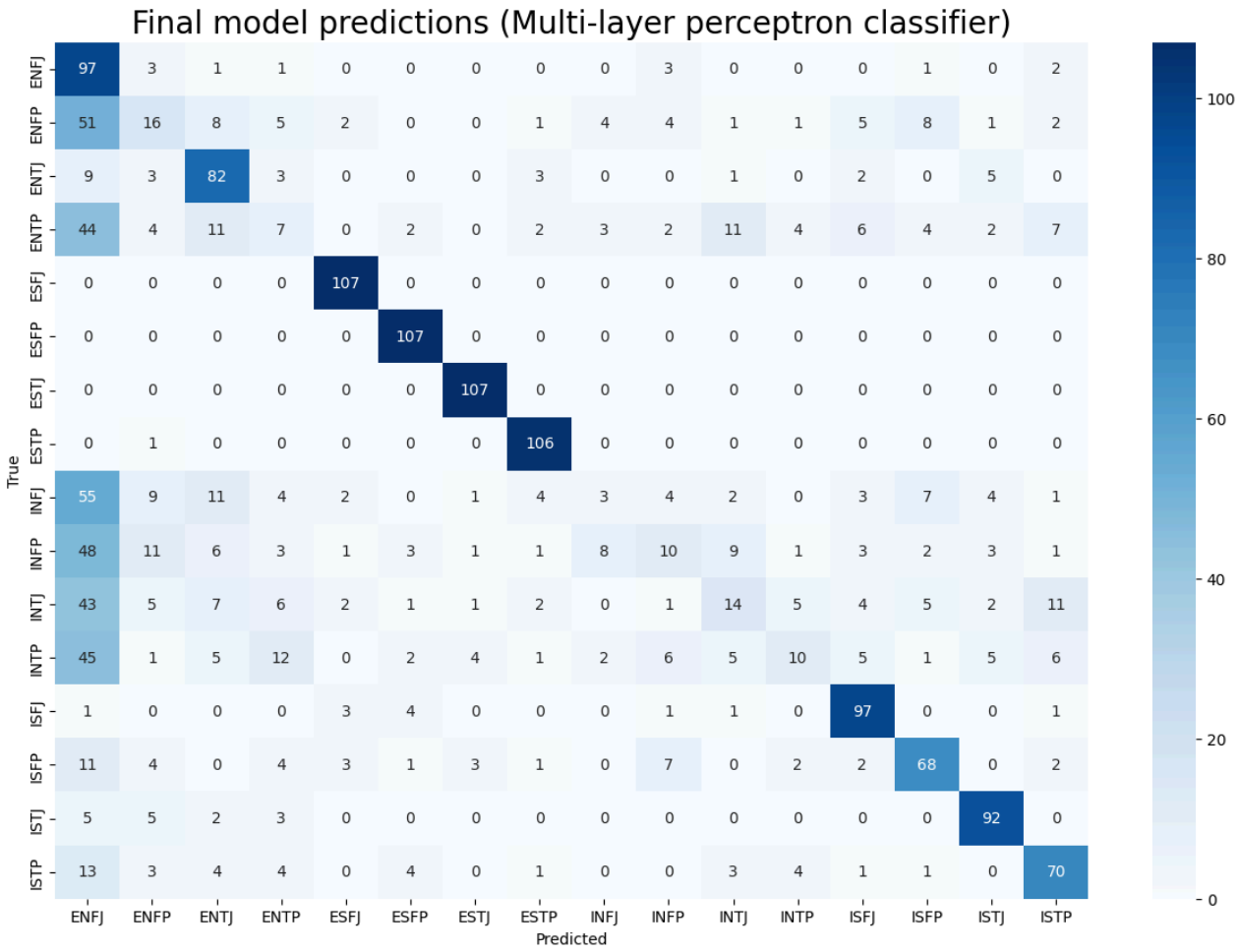
Final model

MLP classifier

Finding the best parameters for the MLP classifier was a much more complex process than with the Random Forest classifier. The model struggled with underfitting at first, but experimenting with hidden layers improved that issue. I ran the grid search multiple times with different hidden layer count and size parameters and found that there was no improvement in performance once increasing the complexity of the network beyond 3 hidden layers.

The final model achieved a balanced accuracy of 57.7% on unseen data. It performed very well on minority classes (in stark contrast to the exploratory analyses that were done before a method was designed for class normalization) and interestingly, gave relatively few predictions for what were originally the majority classes in the data (Fig. 11).

Fig. 11: Confusion matrix, final model



Looking at the classification report on the final model, as well as comparing AUC scores across all classes, also showed that there were definitely certain classes the model had learned to predict better than others (Fig. 12a, Fig. 12b). This was about half and half, but across all classes, the model still did better than expected.

Fig. 12a: Classification report, final model

	ENFJ	ENFP	ENTJ	ENTP	ESFJ	ESFP	ESTJ	ESTP	INFJ	INFP	INTJ	INTP	ISFJ	ISFP	ISTJ	ISTP
precision	0.64	0.26	0.59	0.15	0.88	0.84	0.86	0.83	0.16	0.36	0.24	0.39	0.71	0.61	0.73	0.58
recall	0.81	0.17	0.78	0.09	1.00	1.00	1.00	0.99	0.04	0.20	0.16	0.15	0.91	0.63	0.91	0.69
f1-score	0.72	0.21	0.67	0.11	0.93	0.91	0.92	0.91	0.06	0.26	0.19	0.22	0.80	0.62	0.81	0.63
support	108.00	109.00	108.00	109.00	107.00	107.00	107.00	107.00	110.00	111.00	109.00	110.00	108.00	108.00	107.00	108.00

It seemed that the model had learned to underpredict majority classes and overpredict minority classes, which may be a natural mechanism used to account for the class imbalance when using balanced accuracy

as a scoring metric. The performance on the minority classes was much higher than pre-normalization modeling had achieved, which caused the performance on majority classes to suffer.

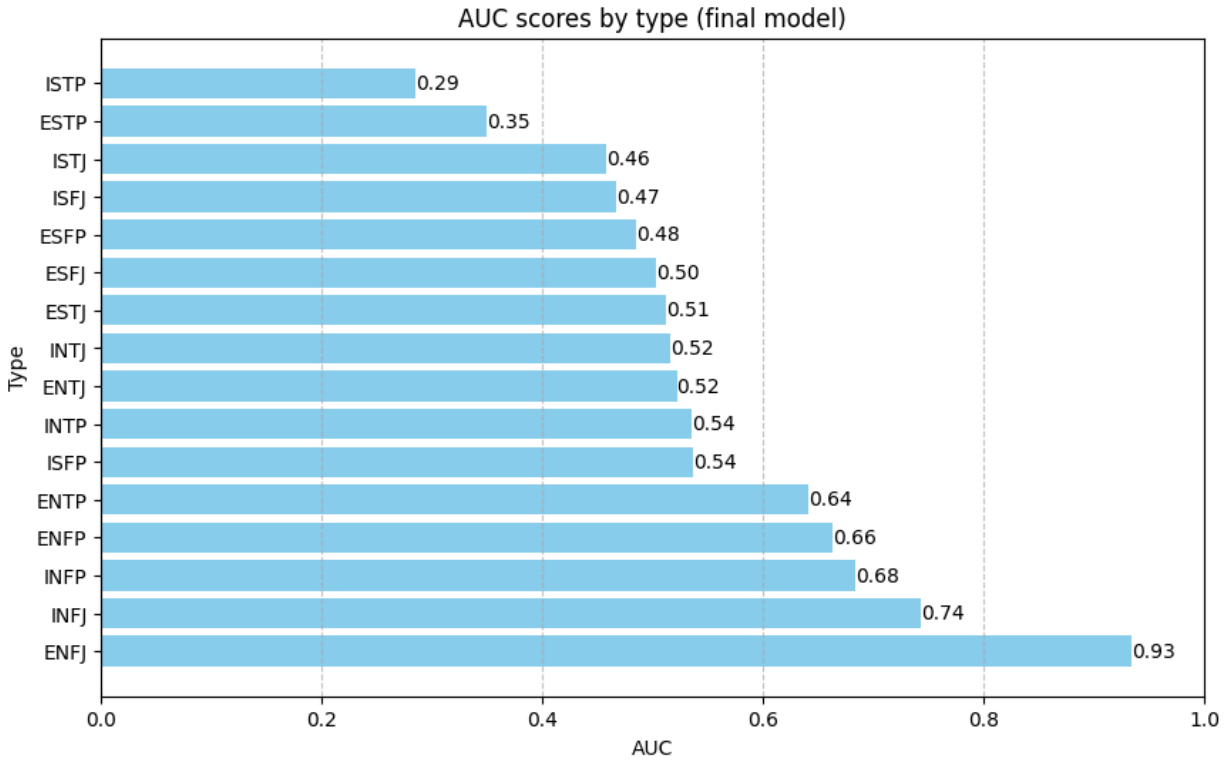


Fig. 12b: AUC scores, final model

Overpredicting on ENFJs

By overpredicting ENFJs, a characteristic of the model that is very visible in the confusion matrix, the model managed a high AUC score despite the fact that ENFJs made up a very small proportion of the original dataset. At first, this was a cause for concern, but taking a closer look at the ENFJ predictions mostly alleviated my questions about whether this could be the result of a flaw in the training process (Fig. 13).

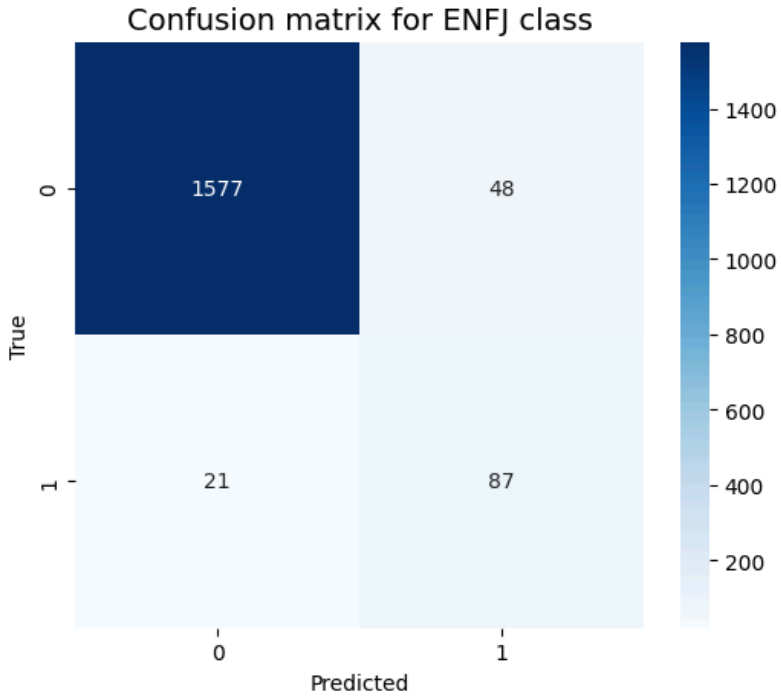


Fig. 13: ENFJ confusion matrix

While the ENFJ class did come up with a relatively high proportion of errors, on the whole the model was doing a good job of recognizing which cases were *not* ENFJs, and the apparent bias visible in the confusion matrix was actually evidence that the model simply did *so* well (58%) predicting all of the other classes that these mistakes were an anomaly.

For comparison's sake, baseline (random guess) accuracy would be around 6.25%, and the untuned Random Forest classifier was only about 53.6%.

Conclusion

Business applications

This model can be applied to a variety of business scenarios, but my original goal was to build a tool that project managers and supervisors could use to assemble more cohesive, productive teams in the workplace in order to foster better output and a better experience for all those involved. Conflict between certain personality types is almost a guarantee, and while it is important to have diverse teams, there are many business situations in which such conflicts in work style, communication, and values can have a direct negative impact on not only the efficiency of the team's work but also the quality of the final product, whatever that may be. Being able to predict MBTI types in the workplace, whether from social media data, workplace forums and message boards, or other sources of text, will help leaders organize and maintain more effective and happy workforces.

A model like this can also be applied to customer reviews, forums, or customer service data to analyze the makeup of the customer base in order to inform marketing decisions and product development. A company looking to offer mental health services, for example, might want to segment their customer base in order to gain a better understanding of what kind of services will benefit the most people. A heavily introverted customer base might benefit more from AI chat services or better email communication, while a more extraverted customer base would likely be more receptive to virtual meetings or group therapy sessions.

Another company might want information on the MBTI clusters in their customer base in order to target certain clusters for marketing. For example, they might run more visual ads on sites or in locations frequented by a large number of sensor-feelers, who gravitate toward sensory stimulation and emotional appeal, and leave the more verbose, informative ads to sites or locations frequented more often by intuitive-thinkers, who require more facts or data to make their decisions.

Alternative approaches

For this project, I chose to take a fairly straightforward approach to predicting MBTI type directly from the labeled data. Considering the number of classes and the relative complexity of the problem, the final model performed strongly, an improvement compared to non-ML methods such as online quizzes asking users to score or classify themselves on a large number of unique indicators or questions. Studies show that with these kinds of tests, about 50% of users will receive a different result when taking the test a second time, even in a relatively short test-retest period. So in light of that, the accuracy of this model (57%) represents a great step forward in the field of personality typing.

Alternative approaches to this problem that might result in different performance, even if I were to use the same data, could include building a set of four individual classifiers to predict each binary MBTI dimension (introversion/extraversion, intuition/sensing, feeling/thinking, judging/prospecting) or to predict Jungian cognitive functions (Ni/Ne, Fi/Fe, Ti/Te, Si/Se) which are immediately related to and can be directly derived from the four-letter MBTI codes. It is possible that these other approaches might lead to higher predictive capabilities, but it is also equally likely that these approaches would still fall victim to the inherent problem in personality theory today, which is the self-reporting aspect of any MBTI data that can be collected from the Internet.

Cognitive function theory

If I were to return to this problem, those are two alternatives I would be very interested in testing, although using cognitive functions as classes does present other issues, such as the need to take into account function placement (each MBTI type has a specific 4-function stack) and the existence of only 16 permutations of these functions (in case the model attempts to predict a function stack that does not correlate to any MBTI types). These are problems that I have as yet been unable to solve, which is why I could not use the cognitive functions approach for this project, but I remain confident that there is a solution, even if that might require a more complex model or a data pipeline with more processes.

Multiple classifiers

Creating separate classifiers for each of the four personality dimensions in MBTI would also present an issue with determining model performance, since this would very likely decrease the rate of "perfect" classifications (all four dimensions predicted correctly for a sample) but might increase the overall strength of the model if approximate or close predictions can also be taken into account. The model built in this project can only be correct or incorrect in its predictions, and unfortunately cannot take into account inherent similarities between types, like the relative proximity of INFP to INFJ as opposed to INFP and a more dissimilar type, such as ESTJ. Some applications, including existing MBTI quizzes on the Internet, might prioritize producing an approximate prediction (for example, three out of four dimensions predicted correctly) over producing a perfect prediction.

Reflections

The draw of social categorization is evolutionary, stemming from the inherent social nature of humanity. Although MBTI and personality theory in general is considered unreliable, there will always be a demand for it, so continuing to refine and improve tools such as this classification model is a valuable endeavor. Given more data, or even data that is less reliant on self-reporting (ie. third-party observational studies assessing behavior to determine type), I think this model and others like it could easily be improved upon, and perhaps even eventually reach a level of accuracy that could elevate personality "theory" into a more recognized "science" of personality, even if in the end, MBTI loses out to other popular theories, such as enneagrams, Jungian cognitive psychology, Keirsey temperaments, and more.

Addendum

Alternative model using multiple classifiers

After completing the original model discussed in the report above, I returned to the problem to test an alternative that occurred to me while reflecting on the business applications of this model. In the scenarios discussed above, when the model in question is used to type a large number of people, ie. an entire workforce, without necessarily providing each person with their results, it is adequate and relatively efficient to produce a single 4-letter type prediction.

However, in the case that the results of the prediction are meant to be received by the actual person in question, then a model which produces predictions that can only be *wrong* or *right* may not be the best option. The benefit of MBTI theory is that each of the 4 letters in any given type corresponds to a dimension of personality, as in Fig. 14 (dimensions are 0-indexed to align with project notebooks).

Fig. 14: MBTI personality dimensions

Dimension 0	I, or introverted	E, or extraverted
Dimension 1	N, or intuitive	S, or sensing
Dimension 2	F, or feeling	T, or thinking
Dimension 3	J, or judging	P, or prospecting

It occurred to me that for the second business case described above, it would be more valuable endeavor to predict as many *dimensions* as possible rather than simply predicting one of 16 types and hoping that the model had captured the right answer. In other words, if a user were applying this model to their own data (text they had written themselves), it would be more "accurate" to the user if 3 out of 4 of the personality dimensions had been predicted correctly.

A model like this would be difficult to measure, since in terms of any of the usual scoring metrics, it would likely perform quite poorly. To test this, I went ahead and built the model (actually a set of models) and analyzed its performance.

Model selection

The first step was testing and building classifiers for each of the 4 dimensions. Each dimension was tested with a logistic regression, an MLP classifier, and a Random Forest classifier.

For the first 3 dimensions, the results of model selection were consistent with the single-model method, and an MLP classifier was tuned and fit for each of those dimensions. Only the fourth and final dimension achieved better performance with a logistic regression. The flexibility of having 4 individual methods meant that it was entirely possible to use a different algorithm for one letter of the 4-letter type, so that is what I did.

Fig. 15 shows the confusion matrices for each of the 4 final models, after hyperparameter tuning.

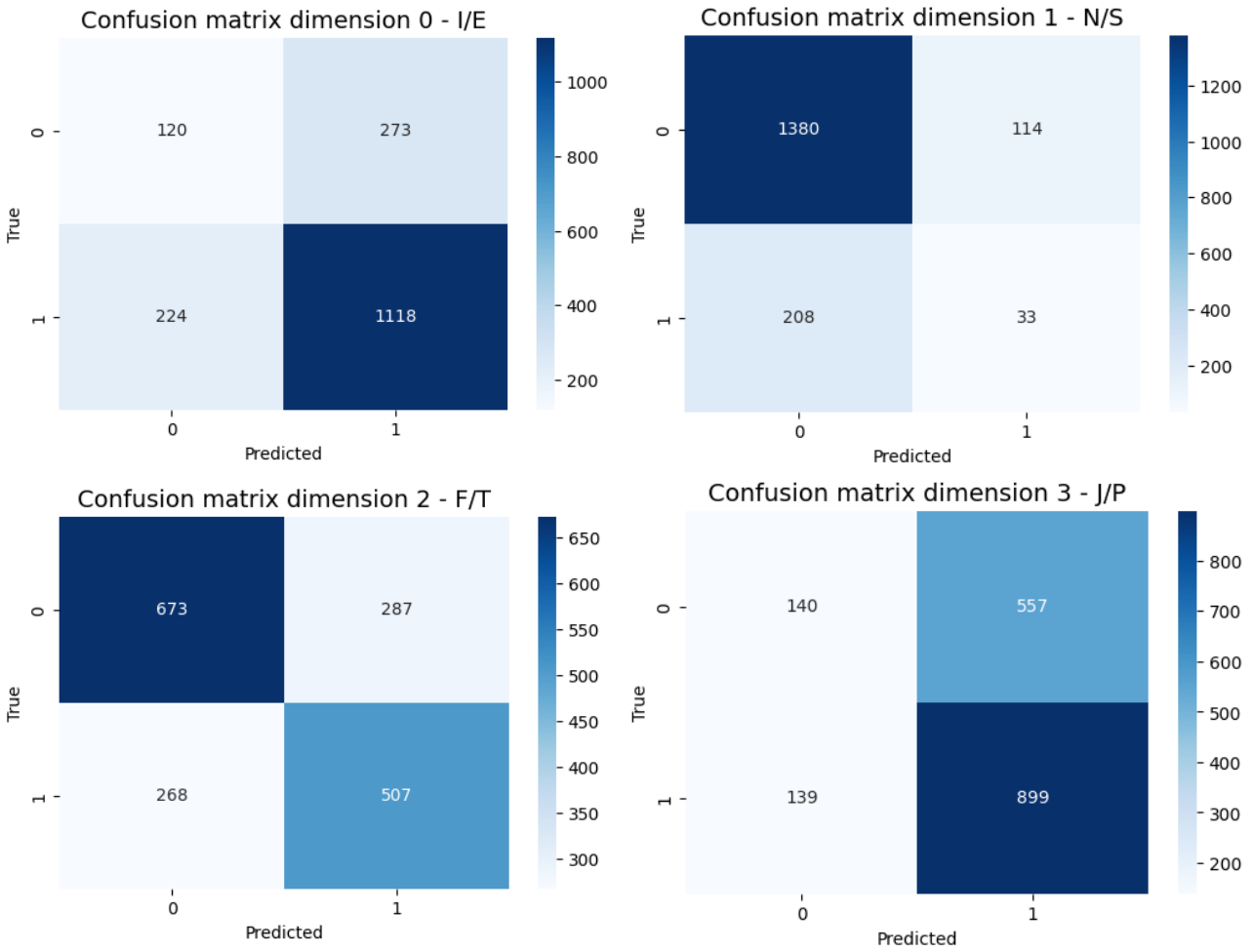


Fig. 15: 4-model structure, confusion matrices

Out of the 4 models, the third dimension (F vs. T) was clearly the strongest (Fig. 16). This might be attributed simply to the fact that natural language is generally used to *express* feeling and *discuss* thinking, and is less obviously used to demonstrate introversion or extraversion, judging or perceiving. This theory would lead me to expect, however, that the intuition vs. sensing model should also be fairly strong, especially considering the topic modeling that was done early on, which showed clear semantic differences between the intuitives and the sensors.

Fig. 16: Performance by dimension

Dimension	Accuracy
Dimension 0, I vs. E	56.92
Dimension 1, N vs. S	53.03
Dimension 2, F vs. T	67.76
Dimension 3, J vs. P	53.35

Another explanation might be that the emotion vectors engineered from the separate emotions dataset were enough to make the difference for separating the feelers from the thinkers. The actual emotions analysis was done fairly quickly and was relatively shallow, but perhaps a closer comparison of that model against the third-dimension model in this experiment could bring a connection to light.

Averaging out the performance of all 4 models in the system gives an overall accuracy of about 58%, which is no better or worse than the original single-prediction model. A more technical analysis, possibly comparing the count of correct dimensions, might give a different result.

As expected, however, directly applying the 4-model process to the test data and combining each prediction into a single response resulted in quite abysmal performance: 26.8%. This number, however, is irrelevant in light of the purpose for which this experiment was done.

Takeaways

This additional experimentation produced some interesting results, especially the quality of the predictions coming from the third model in the system (feeling vs. thinking). I think many more questions could be raised here, and further experimentation could still be done, such as with the second alternative I raised earlier (see “Cognitive function theory”, page 16).