Capstone 3 - Final Report

Springboard Data Science Career Track
Trey Scofield

Introduction

In this work, we used medical abstracts written by doctors to classify whether a patient's symptoms are due to digestive system diseases, cardiovascular diseases, neoplasms, nervous system diseases, or general pathological conditions. Furthermore, we compared the results of more traditional methods of preprocessing, extracting features, and classifying NLP data with the results of OpenAI's chatbot API.

The Data

The data we are using is a publicly available dataset from *Kaggle* consisting of examples of medical abstracts for patients with various conditions. The database contains both a training and testing set containing approximately 14000 observations each. However, the testing set does not contain labels, therefore, we will not be using it in this work. Luckily, 14000 observations will be suitable for splitting into testing and training sets.

The text data itself varies from patient to patient ranging from only 24 to 596 words. However, we will investigate whether the length of the abstract, among other characteristics, has an effect on the quality of diagnosis. Also, each observation contains a digit which is the diagnosis label itself. The labels are as follows: 0 = neoplasms, 1 = digestive system diseases, 2 = nervous system diseases, 3 = cardiovascular diseases, and 4 = general pathological conditions.

The data needs some initial formatting to be completed to be usable. First, we needed to separate the digits (diagnosis label) from the abstract text itself using regular expressions (RegEx) to find digits followed by a tab. Following this, the observations could be separated into text and label datasets.

EDA

Next, in the data science pipeline is exploratory data analysis, or EDA for short. There are not many useful EDA tasks for NLP projects given that all of the data is text-based. However, one thing we can do is investigate our response variable, the diagnosis label, a bit further. In Figure 1, find the distribution of the five response classes. We can see that the most frequent diagnosis is class 4, or general pathological conditions. However, there is not a large class imbalance in the data, so we shouldn't need to take any special precautions in the modeling stage. The least frequent diagnosis is digestive system diseases. General pathological conditions outweigh digestive system diseases by 4805:1494 or approximately 3.2:1.

Feature Engineering

To understand as much as possible from the diagnosis text, I am going to extract several numerical features from each observation. These numerical features will describe the following:

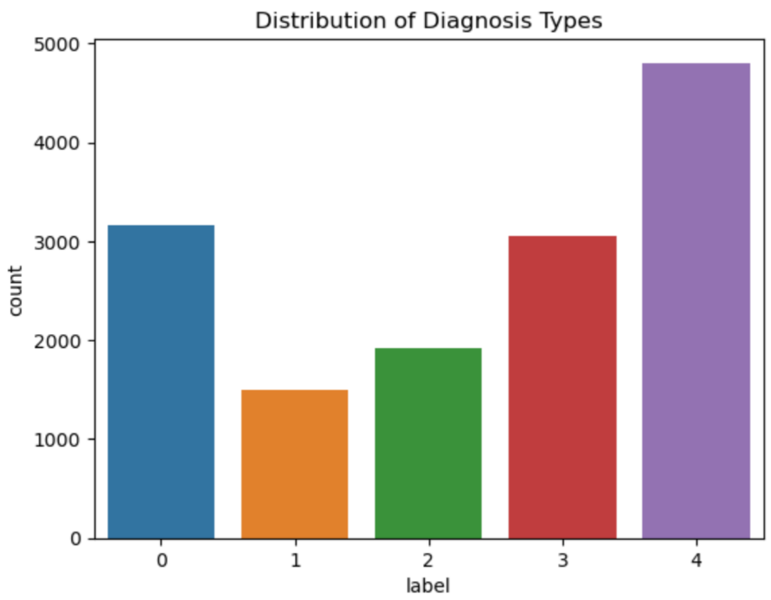


Figure 1: Distribution of diagnosis labels in the *Medical Text* training data

character count, word count, capital word count, quotations count, sentence count unique word count, stopword count, average word length, average sentence length, unique word ratio, stopword ratio, verb count, noun count, adverb count, and adjective count. In total, there are fifteen added features that we can utilize in the modeling stage. Furthermore, we can test our models with and without the added features to determine their value in classification.

Text Preprocessing

Now, to set the classification algorithms up for the most success, there are several text preprocessing steps to complete. Moreover, I am going to tokenize the text, lowercase all characters, remove the stopwords, lemmatize the text, and execute a TF-IDF vectorizer. In more detail, the tokenizer is used to split up the text string into words, which will be more useful in analysis. Then, it is standard to lowercase all letters so that words can be matched more easily. Stopwords are words that will not be useful for the NLP analysis. Furthermore, they are often “filler” words or other words that allow us to speak fluently and with correct grammar. They are important to everyday life, but the computer will achieve higher classification if these are removed from the data. Next, we lemmatize the text which groups together different inflections of the same word such as “improver”, “improving”, and “improvements” all stemming down to the same word of “improve”. Lastly, we ran the text through a TF-IDF vectorizer which weights the word counts by a measure of how often they appear in the documents. This step should help identify words that are more important for certain medical diagnoses.

TF-IDF Vectorizer

Given that we had several thousand unique words in the total dataset, we could not train a TF-IDF vectorizer without assigning it a smaller size as a 14000 x 100000 matrix is not plausible for classification. Therefore, we trained multiple TF-IDF vectorizers using various feature sizes to find the optimal size for our scenario. To test the vectorizers we trained baseline Random Forest and XGBoost classification models and found that a feature size of 2500 is optimal for our situation. A full set of results showing the model's accuracy and F1-score for various feature sizes can be found in Figure 2.

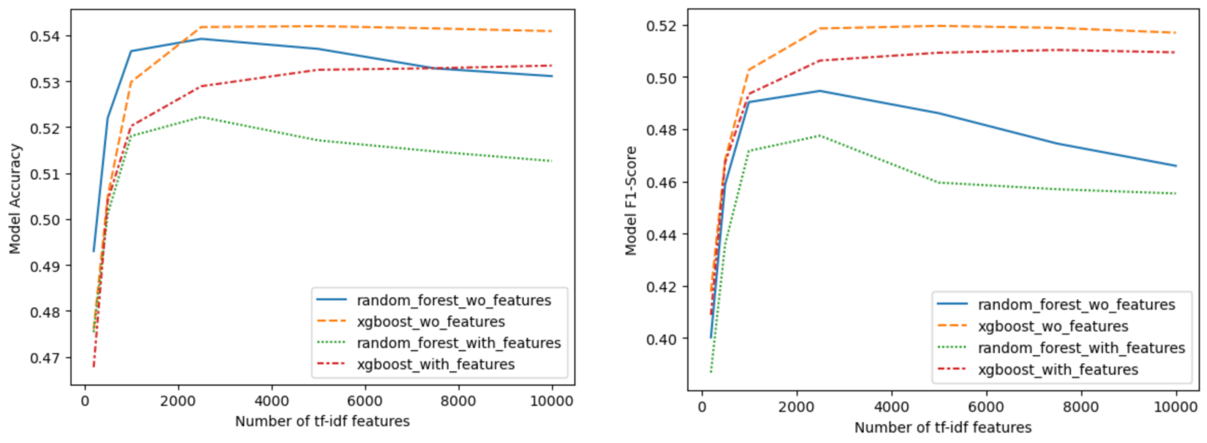
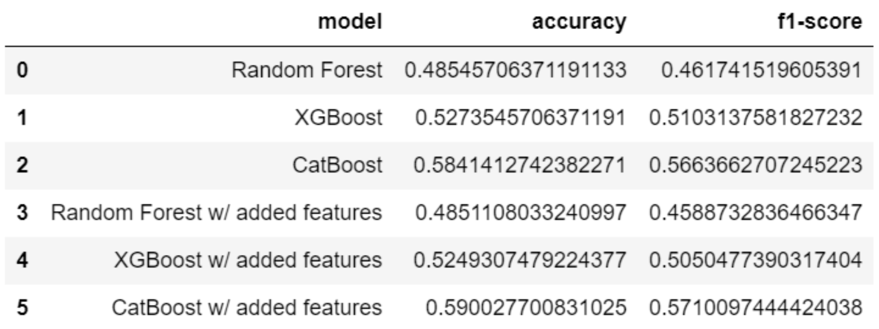


Figure 2: Accuracy and F1-score for Random Forest and XGBoost models with various TF-IDF features

Modeling

In the last step of the data science pipeline, I utilized Random Forest, XGBoost, and CatBoost classifiers to measure how accurately medical abstracts could be assigned diagnosis labels. Furthermore, we want to also test the effectiveness of the added numerical features to the medical classification. Therefore, in the first step of modeling, we trained and tested baseline models for all three algorithms using only the text data and using text data plus the numerical features. The full results can be found in Table 1. The best model for both accuracy and F1-score metrics was a CatBoost classifier using the data with the added features. The full classification results can be found in Figure 3. Also, we calculated the feature importance of the added features for this model. A table of the feature importances can be found in Table 2. We can see that the most important added features to the model are unique word ratio, stopword ratio, and noun count.

Next, the logical step of the modeling process is to expand on the baseline models and parameter tune to find an improved model. However, due to the large number of features following Tf-IDF vectorization and our lack of computing power, tuning the CatBoost model was not plausible. Therefore, we parameter-tuned the Random Forest model to determine if the baseline models could



	model	accuracy	f1-score
0	Random Forest	0.48545706371191133	0.461741519605391
1	XGBoost	0.5273545706371191	0.5103137581827232
2	CatBoost	0.5841412742382271	0.5663662707245223
3	Random Forest w/ added features	0.4851108033240997	0.4588732836466347
4	XGBoost w/ added features	0.5249307479224377	0.5050477390317404
5	CatBoost w/ added features	0.590027700831025	0.5710097444424038

Table 1: Classification results for baseline models with and without added features

Recall	0.5552
Accuracy	0.5900
Precision	0.5984
F1-score	0.5710

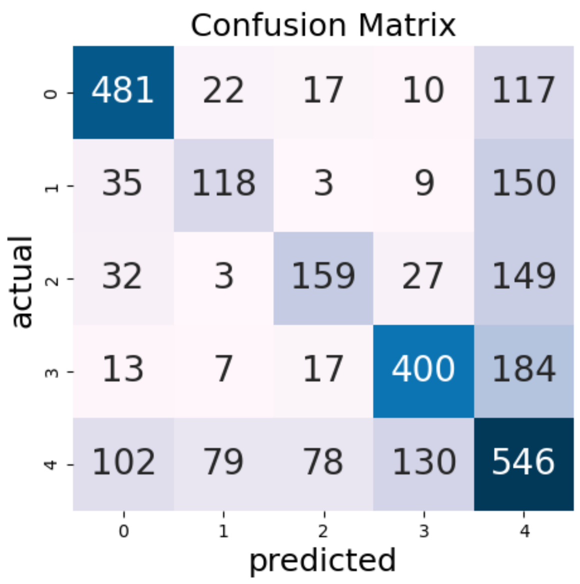


Figure 3: Classification results for CatBoost model with added features

be improved. The results of the parameter tuning can be found in Figure 4. As we can see from these results, the accuracy has increased by approximately 4%, but the results themselves are worse if we look at the other metrics and the confusion matrix. Furthermore, the tuned model had many false classifications of class 4 (general pathological conditions). But because this class has the most observations, it benefited the accuracy to predict a majority of the observations in that class. This exercise concluded that the baseline models are satisfactory for this work. Further parameter tuning of the CatBoost classifier could be included in future work to attempt to enhance the model further.

	feature	importance
9	unique_vs_words	0.333047
10	stopwords_vs_words	0.327831
11	noun_count	0.311376
5	unique_word_count	0.295317
1	word_count	0.291346
8	avg_sentlength	0.261559
2	capital_word_count	0.255978
7	avg_wordlength	0.207939
12	adj_count	0.176331
0	char_count	0.099779
13	verb_count	0.091357
14	adv_count	0.0813
4	sent_count	0.055086
6	stopword_count	0.020947
3	quoted_word_count	0.0014

Table 2: Feature importance of added text features

Recall	0.4142
Accuracy	0.5201
Precision	0.5003
F1-score	0.3981

Max Depth	20
Min. Samples Split	10
Num. Estimators	200

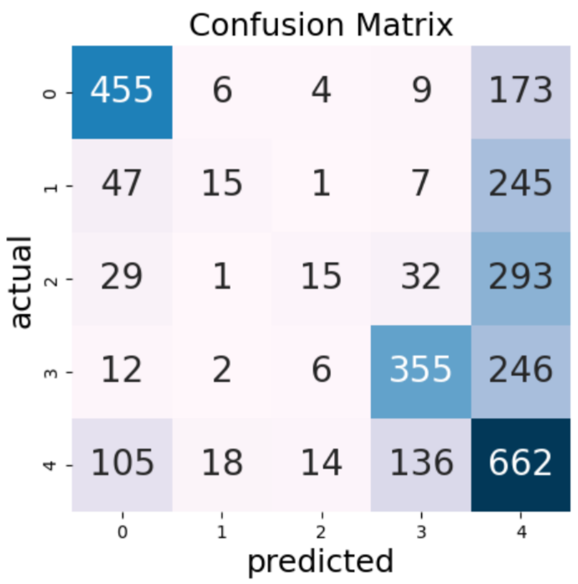


Figure 4: Classification results for parameter-tuned Random Forest Classifier

OpenAI ChatGPT Comparison

In the final step of the project, we wanted to compare our classification results with the ever-so-popular ChatGPT API. Moreover, we fed subsets of our processed text, raw text, and misclassified text (by CatBoost model) into OpenAI's API using both *curie* and *davinci* engines using GPT-3 and GPT-3.5 models, respectively. According to OpenAI's documentation, the curie engine is "very capable, but faster and lower cost than davinci". However, we'll find here that even GPT-3.5 is much more capable than GPT-3. Furthermore, additional engines such as GPT-4 and Google PaLM were not used due to both high-cost and limited public availability at the time. Therefore, it can be assumed that the AI chatbot classifications can be improved upon what is documented here. For all API requests, the following text was used:

> *Is the diagnosis a digestive system disease, cardiovascular disease, neoplasms, nervous system disease, or general pathological condition if the patient has the following conditions:*

plus the text of the medical abstract. This prompt tells the chatbot that we want a specific response. The results for all datasets with both curie and davinci model engines can be found in Table 3.

Dataset	**Engine**	**Accuracy**	**F1-Score**
Raw Data	curie	0.2086	0.1058
(700 random samples)	davinci	0.3186	0.2759
Processed Data	curie	0.2390	0.1336
(3000 random samples)	davinci	0.2453	0.2235
Misclassified Data	curie	0.2420	0.1203
(500 random samples)	davinci	0.334	0.3176

Table 3: OpenAI API performance on classifying medical abstracts for various engines and datasets

Discussion and Conclusion

In this work we have accomplished several things: EDA, NLP feature engineering, text preprocessing, TF-IDF vectorizer parameter testing, machine learning model testing, parameter testing, and comparing the results to OpenAI's ChatGPT performance for both GPT-3 and GPT-3.5 with various sets of data.

We determined that the added features are helpful in machine learning classification. Furthermore, we determined that the most important added features to the best model were unique word ratio, stopword ratio, noun count, unique word ratio, and word count. These added features resulted in approximately a 1% increase in both accuracy and F1-score. Next, we processed the text and trained various TF-IDF vectorizers with various feature sizes to optimize this parameter. We plotted several feature sizes for Random Forest and XGBoost models and found that training with 2500 features was optimal. Then, we trained and tested Random Forest, XGBoost, and CatBoost models with and without the added features to find the best model was using CatBoost with the added features. This combination resulted in a 59% accuracy rate with a 57% F1-score. Furthermore, for a multi-class classification problem with five classes, these results show that the model is able to distinguish differences in the medical text for various conditions. Lastly, we compared our results with OpenAI's GPT-3 and GPT-3.5 engines with various sets of data. Overall, we found that the davinci GPT-3.5 engine performed much better overall than the curie GPT-3 engine. However, davinci is almost 10x the cost and is much slower to compute, so the performance has its tradeoffs. Generally, the AI classifications were not as good as our machine learning methods and the results are not much better than guessing in a 5-class classification (20%). A few of the takeaways from this portion were: (1) the chatbot performs better with raw data which includes stopwords, punctuation, etc. (2) curie is efficient, but does not have the expertise to sort complex data such as this and returns many NaNs; (3) the davinci engine performed best using a sample of the misclassified data from the CatBoost model.

All in all, this project tackled the important task of analyzing and classifying complex medical data in a 5-class classification. Also, it was beneficial to compare the results to the very popular ChatGPT and find where its strengths and weaknesses are.

Future work

With most machine learning applications, getting more data, or in our case, more labeled data would be beneficial to further testing and tweaking our model. Also, finding and including additional feature engineering may be useful in increasing performance. As stated earlier, we did not perform parameter tuning on the CatBoost model because of time and computing power. It would be interesting to see if the baseline model could be improved significantly. Lastly, given the popularity of AI chatbots, there are many new engines being developed by different organizations. Therefore, it would be interesting to test other engine's performance, especially ones like Google PaLM's medical API which is designed specifically for medical diagnoses.