Diagnosing medical transcription

Medical Transcription

Before we start the main question that arises is “what is a medical transcription?”.

Transcription means to listen to a recorded voice and to write down what it is saying.

Medical transcription means to do this for a hospital, doctor's office or other medical place.
When doctors and nurses help someone in the hospital, they do not have time to write down what they saw or heard. So later they record what they saw and heard, so someone else can write it down. The notes are then added to the hospital’s history. The person who does this as a profession is called a **medical transcriptionist**.

e.g., this is a medical transcription:

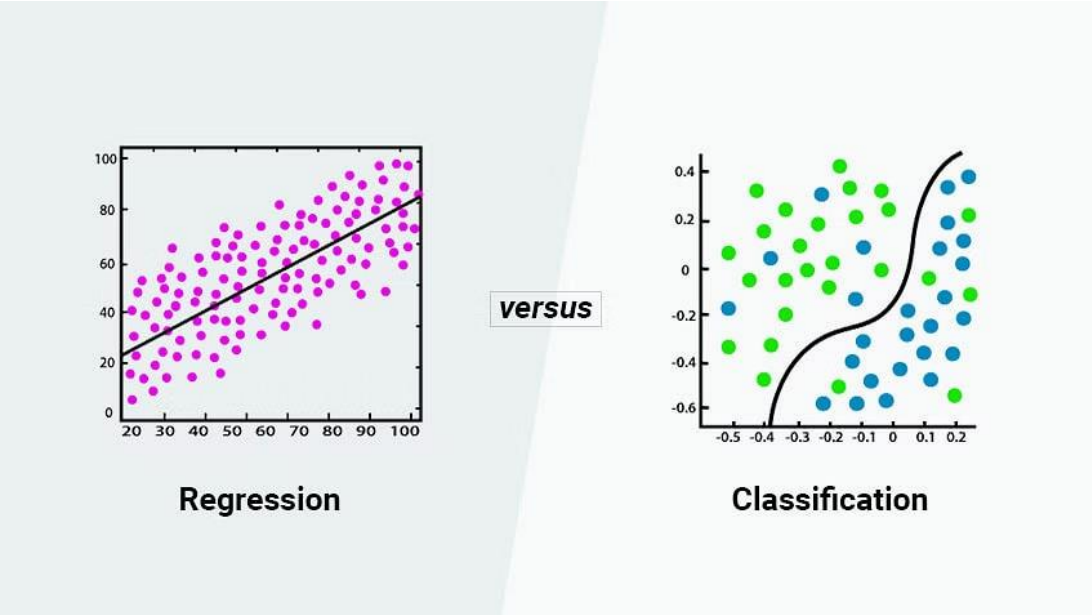
> *“This 5-year-old male presents to Children's Hospital Emergency Department by the mother with "have asthma." Mother states he has been wheezing and coughing. They saw their primary medical doctor. He was evaluated at the clinic, given the breathing treatment and discharged home, was not having asthma, prescribed prednisone and an antibiotic. They told to go to the ER if he got worse. He has had some vomiting and some abdominal pain. His peak flows on the morning are normal at 150, but in the morning, they were down to 100 and subsequently decreased to 75 over the course of the day.”*

What is our purpose?

We want to take transcriptions as inputs and give the category of the illness as our answer.

Our medical specialties are Allergy / Immunology, Autopsy, Bariatrics, Cardiovascular / Pulmonary, Chiropractic, Cosmetic / Plastic Surgery, Dentistry, …

How can we predict?



When we want to use computers to predict, we have two types of prediction. One is to predict a number like 230.4 which can be the predicted price for a bicycle. This type of prediction is called regression. The other type is called classification. As the name recommends, it's used to assign different classes to a desired input. e.g., assigning 0 or 1 to indicate the win or loss of a certain game.

Our case is a classification problem cause we want to assign different categories of an illness to a desired input which is a transcription.

To predict a regression problem, in common sense, we should fit a line (not necessarily straight) to our data. Then when we want to know a new input's answer, we check where that point lands on our line and call that point our prediction. But for predicting a classification we determine a line called boundary. We use this line to divide our data. Where our input lands concerning our boundary lines, decides our answer.

Please notice that for the sake of simplicity, we are describing everything in 2 dimensions. But in our case, we are dealing with inputs much larger.

Imbalanced data:

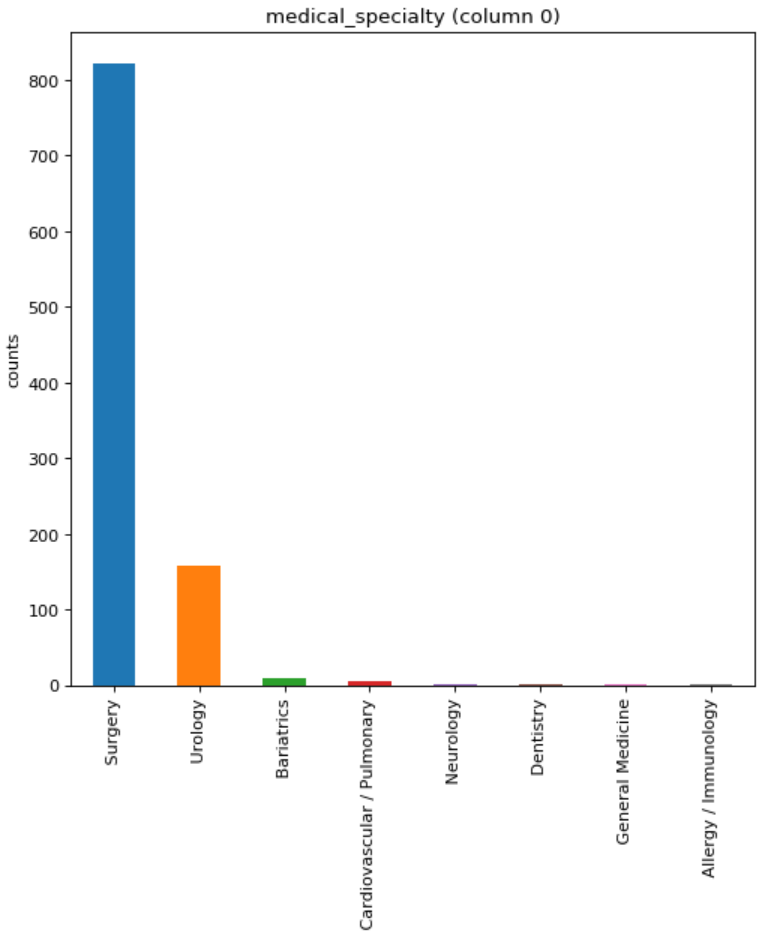
A frequently occurring issue is that our data is not evenly divided between our classes. This can result in disastrous outcomes. If one class appears much more frequently during training, the computer may assume that most of the answers come from that class, causing it to assign all inputs to that specific class during prediction, leaving no chance for other categories.

After plotting our data, we see that there is an imbalance. So, how should we deal with it?

There are two main solutions:

- If the least frequent class has 'n' members, we select 'n' members from each class and train on them.
- If the most frequent class has 'm' members, we duplicate the data in each column as many times as necessary to reach 'm'. Then, we start our training.
- In our case, we chose three classes out of all the available classes. To avoid imbalance, it's important to choose classes with close frequencies. However, we should also consider using distinct categories. Closer relationships make it harder for a computer to make accurate predictions. For these reasons, we chose

  - Obstetrics / Gynecology
  - ENT – Otolaryngology
  - Urology

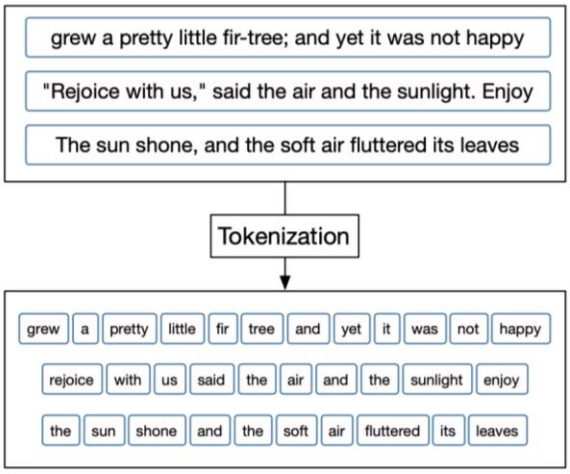


Preprocessing:

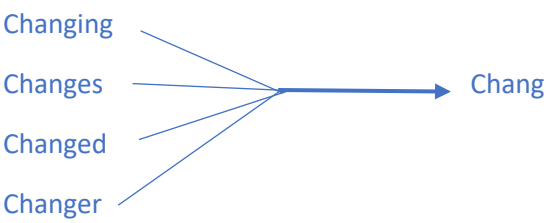
When working with text, one of the most important steps is preprocessing, which means we make our data ready to be used for prediction.

This step contains 4 small steps which are:

- Lowercase:
  - When dealing with words, the value of “happy” and “HAppY” is the same for us. But in computer happy != HAppY. So, we should make them look the same.
- Tokenize:
  - We understand meaning of a sentence by using the meanings of the words appeared in it. So, when we want to train a computer to understand a sentence, we should break it into words.



- Stop words:
  - Some words appear so frequently that they add nothing new to the meaning and value of our document. So, it's important to eliminate them from our list of words. This is because if a word appears frequently, the count vector for the document won't be normalized, and other words will behave as if they were never there.
- Stemming:
  - When describing the same idea, we can use different versions of the same word, such as "she creates" and "she is creative." Despite having the same value and meaning, when we tokenize these phrases, we get ["she", "creates"] and ["she", "creative"], which are not the same vectors. To solve this problem, we stem our words, meaning that we use the root of each word.



Countvectorizer:

We can change a list of words into a vector of numbers by using the frequency of each word in the list. In this method, each document is represented as a vector of the size of our vocabulary, where the frequency of each word in the document is used as the corresponding value in the vector.

	Word1 (hello)	**Word2 (give)**	**Word3 (health)**	**Word4 (secure)**	**Word5 (round)**	**…**
Doc1	0	2	0	1	4	
Doc2	5	0	0	0	7	
Doc3	1	0	7	5	4	
…						

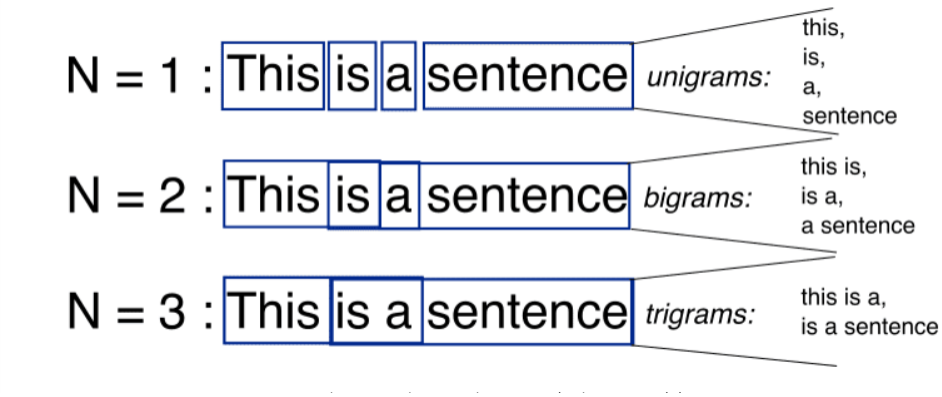
However, if the vocabulary is very large, this approach may consume a lot of storage and decrease computation speed. To avoid this issue, there are typically two approaches that can be used.

- First, we can sort our matrix in descending order and choose the top n most frequent words, as less frequent words usually do not contribute much to the data.

- Another approach is to use Principal Component Analysis (PCA) on our matrix, which involves performing Singular Value Decomposition (SVD) and transforming it into a smaller matrix while preserving as much information as possible.

N-grams:

We don't have to limit ourselves to single words for Countvectorization. We can use 2 words, up to 2 words, three words and …



But using bigrams or trigrams instead of unigrams has both advantages and disadvantages. Advantages include that using more than one word as the tile of meaning will give us more accurate meanings. For example, consider the phrases "river's bank" and "city's bank". If we were using unigrams, we would have the same word "bank", but with bigrams, we can distinguish the difference. However, if we use bigrams or trigrams, it is less likely to see these words in other documents, making our vectors become orthogonal and sparse, which will result in less effective models.

In this issue, we'll examine various models utilizing different n-grams, including (1,1), where each column signifies a singular word, to (3,3), which incorporates trigrams. The use of both unigrams and bigrams is indicated by (1,2).

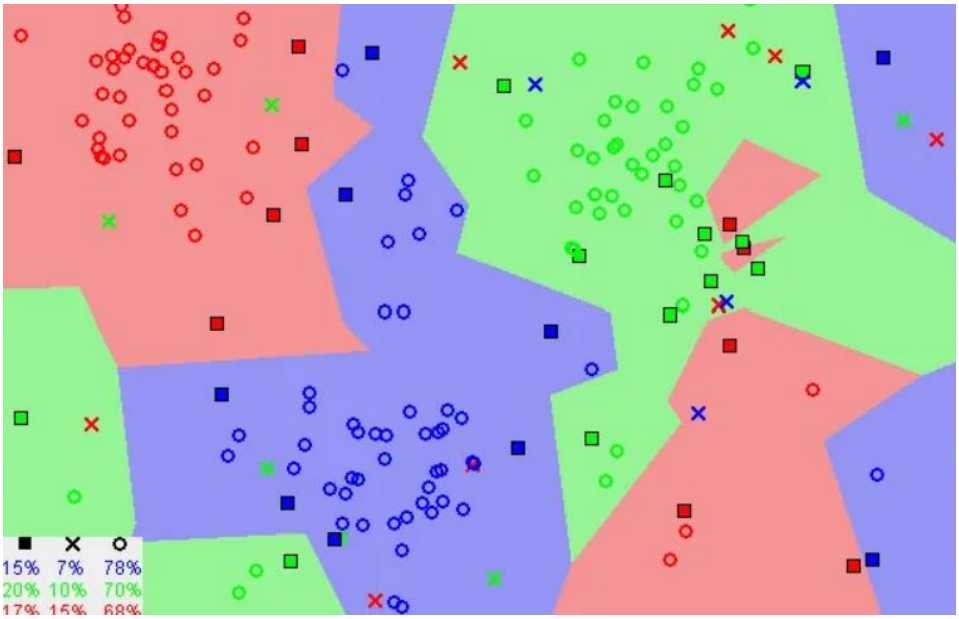
Prediction:

First, we need to divide our data into two parts: training data and test sample. The training data is used to train our model, and then we evaluate the accuracy of our model using the test sample. The training data is like the homework given to students to learn the subject, and the test sample is like the exam taken to assess their knowledge. After dividing the data, we use three well-known classification models to train our data.

K-nearest-neighbor:

The algorithm works by first calculating the distance between the new data point and all the data points in the training set. Then, it selects the "k" number of nearest data points, where "k" is a user-defined parameter. Finally, the algorithm determines the class label of the new data point based on the majority class of its "k" nearest neighbors.

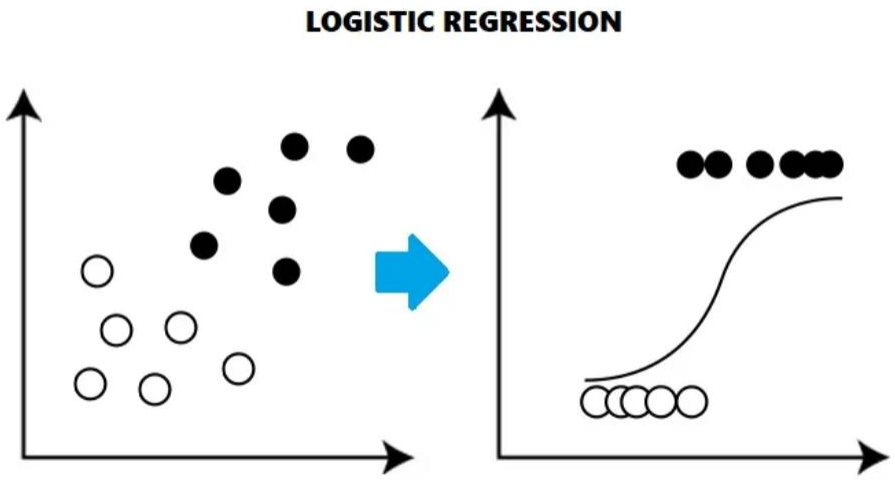
For example, if "k" is set to 5, and the 5 nearest neighbors are all labeled as "A", then the new data point will be labeled as "A" as well. On the other hand, if the nearest neighbors are divided among two or more classes, then the algorithm may use additional techniques, such as voting or weighted voting, to determine the final label for the new data point.



Logistic regression:

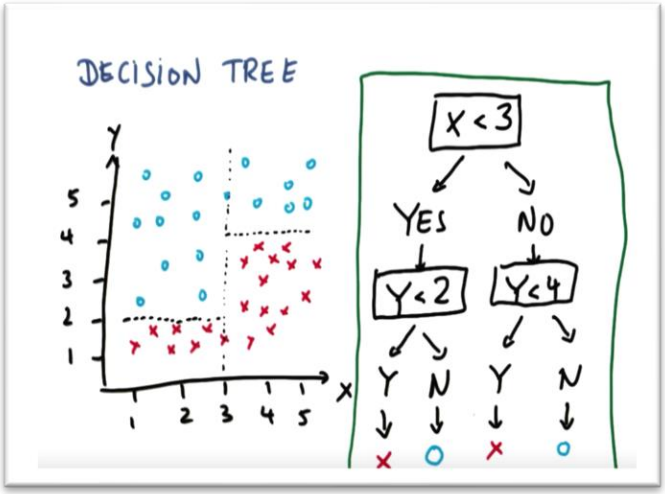
The model for logistic regression is based on the concept of the logistic function, which is a mathematical function that outputs a value between 0 and 1. The logistic function is used to model the relationship between the predictor variables and the response variable, and to make predictions about the probability of the response variable.

In logistic regression, the predictor variables are used to form a linear combination, which is then transformed using the logistic function. This transformed value represents the predicted probability of the response variable. The final prediction is made by thresholding the predicted probability. If the predicted probability is greater than a certain threshold, the model predicts one outcome, and if it is less than the threshold, the model predicts the other outcome.

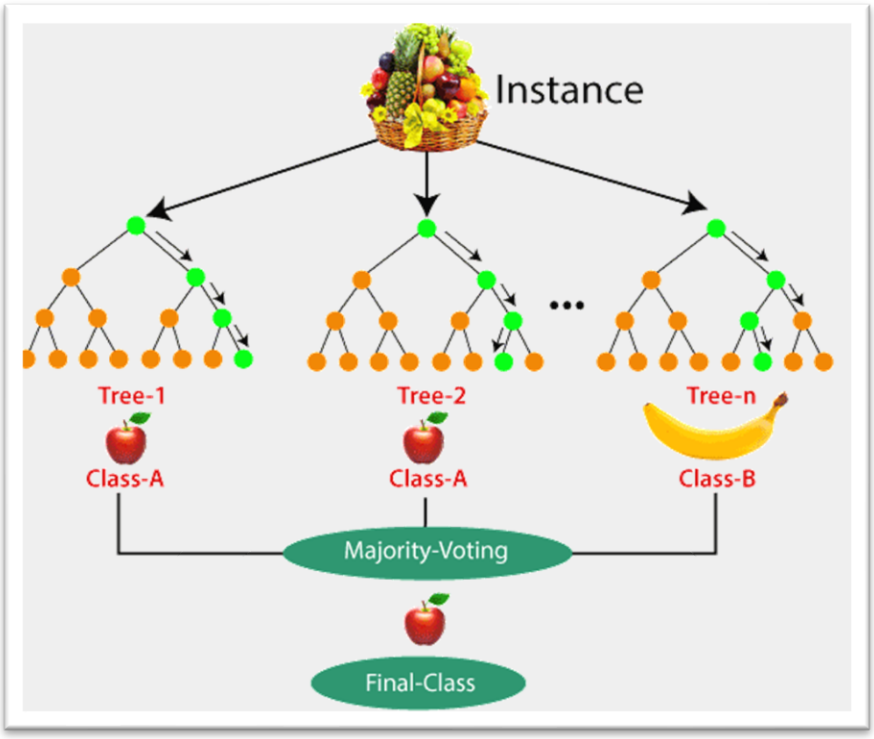


Random forest:

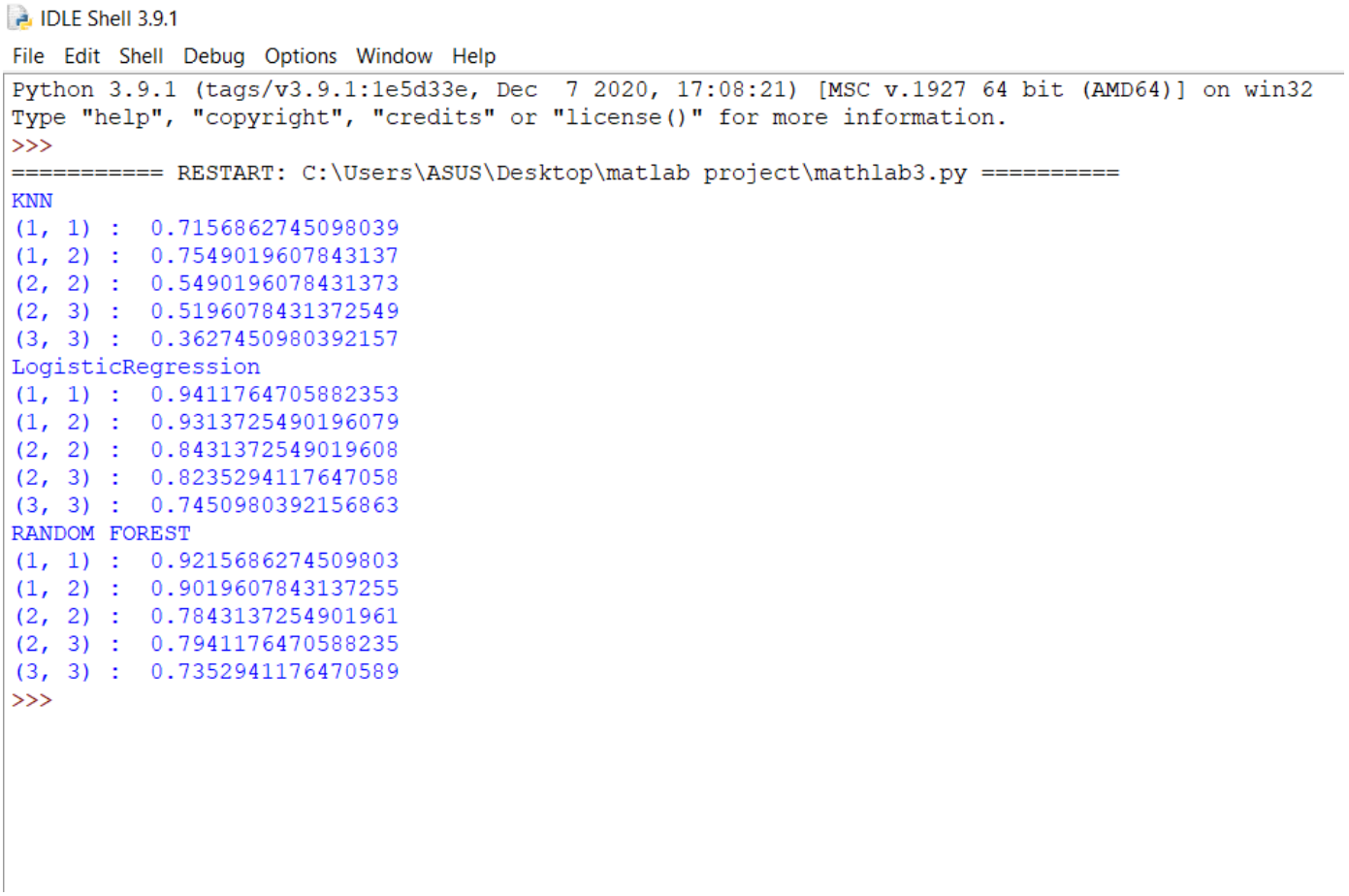
A decision tree is a machine learning algorithm that is used for both regression and classification problems. It works by creating a tree-like model of decisions and their possible consequences. The tree is constructed by recursively splitting the data into subsets based on the most significant features, until the data can no longer be split into smaller subsets. At each split, the algorithm decides which feature to use and what the threshold should be for splitting the data.



Random forest is an extension of the decision tree algorithm, where multiple trees are built and combined to make predictions. Instead of building a single tree, multiple trees are built using random subsets of the data and features. During prediction, the random forest combines the outputs of all the trees, using either a majority vote (for classification) or an average (for regression). This combination of multiple trees helps to reduce overfitting and improve the accuracy of the model.



Results:



As it is evident from the algorithm, KNN was the fastest, but its results were not as good as those of the other models, indicating that similar data were not closely situated. Logistic regression had the highest accuracy rate, but its training time took much longer than the other models. On the other hand, the random forest model had an accuracy rate close to that of logistic regression, but was performed much faster. Thus, if we had a larger dataset and a time constraint, the random forest model would have been a better choice. However, for this particular case with limited data, logistic regression performed the best. Furthermore, as we can see, unigrams almost performed better in every instance, making logistic regression using unigrams the winning model.

As a result, our model performed well and with more balanced data, we hope to use it for all categories and obtain similar great performance.