Task: Given a scientific paper about a particular genetic variation, is it possible to predict which type of effect the variation will have on the protein function?

Data: Training data included one file containing variant IDs and text from scientific papers that mentioned the variants. Another training data file included the variant IDs, the gene that is mutated, the type of variation (for example, a point mutation that turns one amino acid into another, a truncating mutation that cuts off the end of the protein, etc.), and the class of the mutation: a number 1-9 that is grouped by the mutation's effect on the protein function, such as causing the protein to be hyper-active (gain-of-function) or inactive (loss-of-function). However, these labels associated with each numbered class were not provided. Altogether there were about 3000 variants provided, which represented mutations in 269 genes.

Results:
The best submission on the public leaderboard had a log loss of 0.44, which placed 136th. However, due to some confusion and cheating during the contest, the private leaderboard ended up being based on a small number of samples. Other competitors noticed that the private leaderboard test set was poorly matched to the training set, and most competitors' log loss scores significantly increased. On the private leaderboard, my model scored 175th out of 357 with a score of 2.85.

Approach:
I first read in the training data from the two sources and merged them into one dataframe. I created a split in the data so that 80% of the data would be used for training and 20% would be used for cross validation.
I found that the 9 classes were not equally represented. To get a sense of a baseline score, I made a model that simply predicted a class drawn at random with probability proportional to how frequently that class was present in the training data. My log loss with this approach on my cross validation set was 1.8.

Class	Count	Class Label (as provided on discussion boards)
1	662	Likely loss-of-function
2	498	Likely gain-of-function
3	96	Neutral
4	751	Loss-of-function
5	267	Likely neutral
6	297	Inconclusive
7	1054	Gain-of-function
8	21	Likely switch-of-function
9	43	Switch-of-function

Next I attempted to use the scientific paper text in a simple way by making a bag of words (filtering out the default stopwords in NLTK) and using a Naive Bayes classifier. However, my log loss was about 12, so it was clear that this was not a viable approach.
I hypothesized that because genes work in networks, genes involved in the same network might be mutated to cause similar effects. Related genes would likely be mentioned in the

papers, so I built a bag of words using just gene names. I built a list of 269 gene names that were present in the my training set, then searched for their presence or absence in each paper. I also kept track of the identity of the gene that was mutated in each training sample. Feeding just this information into an sklearn logistic regressor I scored 1.1, and feeding it into an sklearn random forest classifier with the default settings I scored 1.3. Based on this, I made the logistic regressor my default model to use for most future comparisons.

After tokenizing the scientific papers (using NLTK's built in word_tokenize function), I found that there were over 300,000 unique words in my paper corpus. The list, sorted by frequency of use, revealed that English stopwords were highly present as well as words that could be considered biological stopwords-- "mutation," "analysis," "data," etc. There was also a long tail of rarely-used words that were unlikely to provide any meaningful predictive power given their rarity. I made a new, abridged list of 3862 words that appeared in more than 10% but fewer than 90% of my training papers. This allowed us to build a more reasonably-sized bag of words that was more likely to be meaningful. Using a logistic regressor, this provided my first sub-1.0 score, and using a random forest it was 1.17. Appending my previous genes-mentioned features decreased my log loss slightly. I later experimented with different cutoffs for my abridged list of words, such as ones that were present in 20-80% or 1-99% of papers. Generally, I found that including more words gave us slightly smaller log losses, but increasing the number of words greatly increased the computation time for my model, so I stayed with my 10-90% cutoffs.

Next I read some of the papers to get a sense of what the nine classes might correspond to. While reading, I paid attention to the way that I was analyzing the text to get an idea of where informative passages might be. I realized that almost all of the critical results and conclusions from a paper are provided in the abstract. By using just the abstracts, I hypothesized that I might reduce the papers to a more manageable size. Although some papers contained formating artifacts that designated the transitions between abstracts and the main text, this depended on the publisher. Since most publishers require abstracts to be around or under 250 words, I used the first 250 words of a paper as the abstract. I tokenized and built a bag of words for the abstracts using my previous abridged list of words appearing in 10-90% of all papers. Using these features with the logistic regressor gave a log loss of 1.2, so I decided that using the rest of the text would be worthwhile.

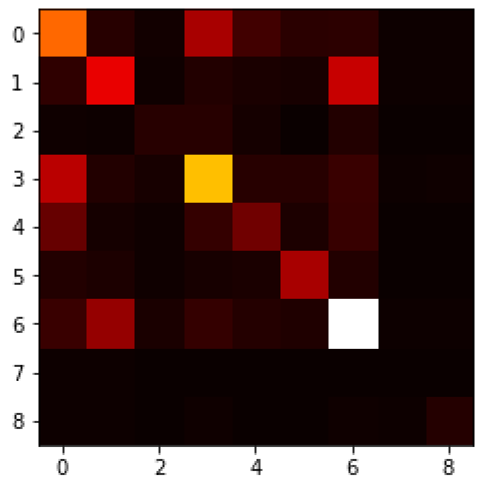
Next, I hypothesized that the type of molecular lesion would likely be informative. For example, a truncating mutation, which cuts off the tail of a protein, often results in a loss of that protein's function. On the other hand, translocations, which can fuse two different proteins together, can be more likely to result in the protein having novel functions. Most mutations were missense mutations, which cause a single amino acid in the protein to change its identity. These variations are typically 6 characters in length or fewer and follow a pattern: "ANB" would designate a change of the Nth amino acid from an A to a B amino acid. Ignoring these mutations by filtering for mutations longer than 6 characters, I found a wide variety of mutations with inconsistent formatting. However, many did include key terms indicating the type of mutation, such as "dup" for duplication or "del" for deletion. When I filtered my training set by these different key terms, I found that there were, as predicted, large differences in the classes associated with each type of mutation. For example, fusion, insertion, and duplication mutations generally were enriched for gain-of-function rather than loss-of-function, and truncations were enriched for loss-of-function classes. I built features corresponding to whether a molecular lesion was a fusion, deletion, duplication, insertion, truncation, and/or splice variant (some contained more than one, such as in insertion deletion) and added these to my previous features of words and genes mentioned in the paper. This resulted in a score of 0.98, a 2% improvement over not including the mutation type.

I realized that multiple mutations sometimes cited the same paper as evidence, even when the different mutations belonged to different classes. By building a bag of words from the entire paper, it would be impossible for us to use the text features to predict different classes from the same paper. I decided to extract relevant sentences so that one paper could provide information specific to the different mutations it mentioned. I extracted sentences including the variation of interest. Sometimes the variation was never mentioned by name, which was not surprising since the variation identifiers were sometimes long and complicated (for example, A750_E758delinsP). In these cases, I extracted sentences that mentioned the gene. By using these sentences, in addition to all previous features, I reduced my log loss to 0.95 on my validation set. Although I was extracting the relevant sentences, the rest of the text features still were useful-- removing them and using only the sentence and gene features scored 1.24. Later I realized that instead of using the variation-mentioning sentences for some mutations and gene-mentioning sentences for other mutations, my accuracy was better if I kept these sentence lists separate and used both.

Next I explored different models a bit. I experimented with optimizing the C parameter of my logistic regressor and also used XGBoost. With its default settings, XGBoost was not better than my logistic regressor. It was more competitive after I changed the max_depth to 6. When I averaged my predictions from the logistic regressor and XGBoost I had a log loss of 0.908.

I experimented with using bigrams as features. To exclude ones unlikely to be meaningful, I removed stopwords and then found bigrams. I found over 775,000 unique bigrams. As I did for the words, I filtered these to only include bigrams that were in fewer than half of the papers and more than 59 papers (59 was empirically determined to remove bigrams that were showing up as being common simply because they appeared in one paper that was associated with dozens of different mutations). Including bigrams in addition to all previous features reduced the log loss to 0.938 from 0.951, but also greatly increased computational cost. I decided to extract bigrams only from the sentences that mentioned the variations to reduce the number and hopefully target relevant language.

I investigated my error matrix and found that I mostly made mistakes discriminating between related classes, such as “likely loss-of-function” versus “loss-of-function” (0 and 3 on this heat map, or 1 and 6 for gain-of-function and likely gain-of-function) Since I don’t know what criteria are used to select between these two labels, and likely different people would make this choice differently, I did not believe I would be able to discriminate between these reliably.



(Note that in this heat map, the classes are 0-indexed, not 1-indexed.)

I also tried stemming (with the snowball stemmer built into NLTK) and lowering all text before analyzing. Lowering all text helped but stemming was worse than just lowering all text. This was true when I used the entire text and also when I used just relevant sentences.