# Classifying Laboratory Test Results Using Machine Learning

Sizhe Chen Kenny Chiu William Lu Nilgoon Zarei

The University of British Columbia

Data Science for Social Good Fellowship 2018

in partnership with

The BC Centre for Disease Control

## Table of Contents

# 1 Introduction

This report was the result of the collaboration between the University of British Columbia's (UBC) Data Science for Social Good (DSSG) 2018 program and the BC Centre for Disease Control (BCCDC). In this section, we introduce the two parties involved and summarize the project context and goals. In Section 2, we provide a brief overview of related work in literature. We share the main results of our analysis in Sections 4 and 5 before concluding this report with some discussion and comments in Section 6.

## 1.1 Data Science for Social Good

The DSSG program is a 14-week fellowship that is offered by the UBC Data Science Institute (DSI) and sponsored by Microsoft. The program is designed to provide UBC undergraduate and graduate students of diverse backgrounds the experience of working on a data science project that pertains to the concept of "social good". While the concept in itself may be broad and abstract, these projects are generally brought forth by nonprofit organizations or organizations in the public sector and make use of data to benefit society and improve people's well-being in some aspect. Through the program, the students have the opportunity to learn and apply data science, and the organizations have the opportunity to make progress on these projects with minimal resources.

## 1.2 Background

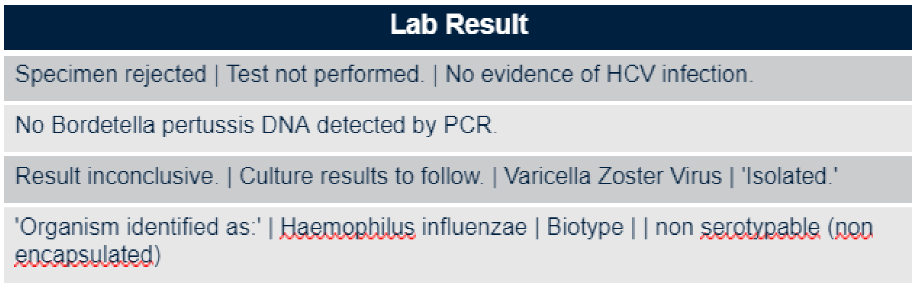
Operated by one of BC's health authorities, the BCCDC provides several health-care services including:

- managing various health clinics and laboratories throughout BC
- providing programs that contribute to public health
- analyzing and monitoring disease trends at the population level

Through their services, the BCCDC has collected over twenty years of patient and laboratory test result data. As part of their work in analyzing disease trends, it is now in their interest to review the historical laboratory tests and extract various pieces of information such as the purpose of the test, the microorganism that is being tested for, and the outcome of the test. However, this process of labeling each test result is currently being performed manually, and is expensive to carry out for the following reasons:

1. The test result description text can be difficult to interpret

The laboratory test results are stored as semi-structured text in their data warehouse. The semi-structure form is the consequence of the laboratory technician having the option of either



| Lab Result |
|---|
| Specimen rejected \| Test not performed. \| No evidence of HCV infection. |
| No Bordetella pertussis DNA detected by PCR. |
| Result inconclusive. \| Culture results to follow. \| Varicella Zoster Virus \| 'Isolated.' |
| 'Organism identified as:' \| Haemophilus influenzae \| Biotype \| \| non serotypable (non encapsulated) |

Figure 1: Examples of laboratory test results.

inputting a code that auto-generates the text, inputting completely free-form text, or inputting a combination of the two where free-form text is appended to auto-generated text. As a result, there is no consistent structure in the test results that can allow for simple rule-based processing.

In addition, the actual content of the text further complicates the issue more. Unlike standard natural language documents, the test results are often recorded in point-form and are mostly composed of laboratory-specific terminology including microorganism names, laboratory procedure names, and abbreviations. It is also not uncommon that the entirety of the text is a single number, or that the free-form text contains typographical errors. The free-form text may also include information such as names and phone numbers that may require special attention during processing. Figure 1 shows examples of what a test result description may contain.

2. There is a high-volume of test results that still need to be interpreted and labeled

BCCDC's data warehouse contains over 330 million test results. As of the time of this report, they have labeled only a subset of the data that is relevant to some of their data marts. Manually going through the large number of remaining unlabeled test results along with the new test results that are added each day would require additional resources that are simply not available.

### 1.3 Project Goals and Scope

The process of manually labeling the test results in BCCDC's data warehouse is expensive to carry out. Thus, the goal of this project is to achieve the following objectives:

- G1. To identify the appropriate machine learning or natural language processing (NLP) solutions for automating the labeling process
- G2. To provide an example code solution implementing G1 that can be integrated into the extract, transform, and load process of the data warehouse
- G3. To recommend changes in how the data is stored to improve the efficiency of the labeling process

This report focuses on the work that we associate with G1. Specifically, we evaluate algorithms for labeling the outputs of interest shown in Table 1. Our code solution for G2 is detailed in a separate technical documentation produced along with this report. Section 6.3 provides a summary of our recommendations for G3.

| **Output** | **Description** | **Classes** |
|---|---|---|
| `test_performed` | whether the test was performed | *Yes*<br>*No* |
| `test_outcome` | whether the microorganism being tested for was found | *Positive*<br>*Negative*<br>*Indeterminate*<br>*Missing* |
| `level_1` | the genus name of the microorganism being tested | 30 classes and counting |
| `level_2` | the species name of the microorganism being tested | 200 classes and counting |

Table 1: The outputs to label for each test result.

## 2 Literature Review

We provide a brief overview of similar work in literature in this section. In 2014, Kang and Kayaalp explored the problem of extracting laboratory test information from biomedical text and compared the performance of an original symbolic information extraction (SIE) system to various machine learning-based NLP systems [2]. Their results showed that well-tailored symbolic approaches may outperform machine learning-based approaches. The main difference between our problem and theirs is that the document corpus they extract from are decision summaries from the U.S. Food and Drug Administration. These summaries are written in natural language unlike our test result descriptions which are found mostly in point-form. We expect that this difference will be significant enough that a symbolic approach to our problem will require more complex logic before it can achieve similar results to theirs.

Jang et al. used hidden Markov models to text mine doctor notes in 2006 [1]. The document corpus that they worked with were more challenging as the notes were written in a mixture of English and Korean. Their models achieved around 60%-70% accuracy and aimed to be robust to unknown phrases not seen in the training corpus. One notable tool that they also used is *MetaMap*, which can be used to annotate input text with medical tags and semantics. MetaMap is also commonly used in other health domains such as for text mining of cancer-related information [3].

## 3 Dataset

The dataset that we work with was provided in the form of multiple tables in a Microsoft SQL Server 2016 database. The dataset is a subset of the data available in BCCDC's data warehouse and consists of approximately 950,000 test results. The test results were selected in a way such that the majority contain no personally identifiable information. Not all test results in the dataset are labeled. Figure 8 in Appendix 7.1 shows the proportion of labeled data for each output. We note that the proportions vary significantly ranging from 100% in the case of `test_performed` to around 11% for the microorganism name outputs.

Figure 9 in Appendix 7.2 shows the class breakdown of labeled data for each output. We observe that `test_performed` has severe class imbalance, with 94% of the labeled rows being *Yes*. `test_outcome` also has severe class imbalance between 69% of the labeled rows being *Missing* and 1% *Indeterminate*. We also note that there are some `level_1` classes with very few rows in the dataset. The smallest class has only 50 rows.

We will refer to the data fields shown in Table 2 frequently throughout our analysis in Section 4.

| Data Field | Description |
|---|---|
| `result_full_description` (RFD) | the full test result text. A RFD may consist of multiple observations |
| `test_type` | describes the type of test, e.g. `Culture` for culture tests |
| `test_code` | the code for the name of the test, e.g. `CULT` for some culture tests |
| `obs_seq_nbr` | the sequence number in which the observation appears in the RFD |

Table 2: The relevant data fields we use in our analysis.

## 4 Results and Analysis

We highlight our key analysis and results in this section. Our analysis was primarily carried out using Python 3.6 with the MetaMap components done in Java 10.

### 4.1 Preliminary Analysis

Before we explore possible solutions to our main problem, we first run simple queries to get a better sense of what the data consists of. In particular, we attempt to identify common patterns

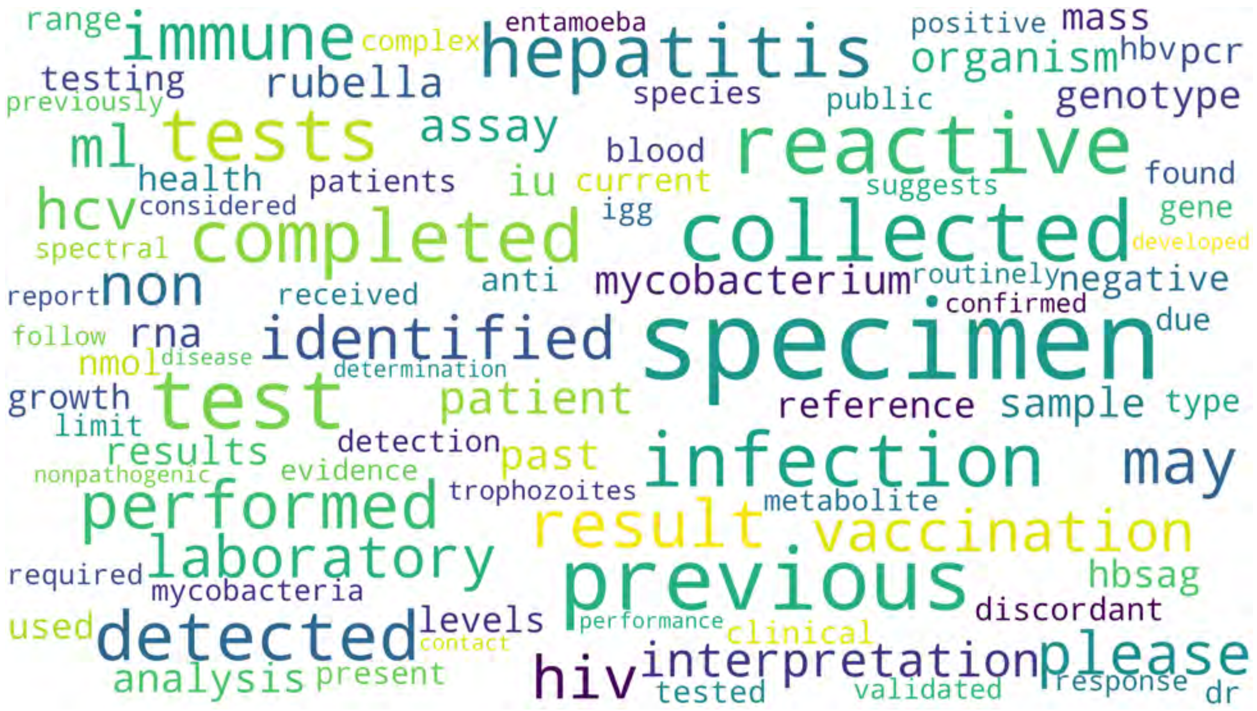


Figure 2: A word cloud of the most frequently occurring words in the RFD's (stop-words removed).

in the RFD fields of the test results. A word cloud of the most frequently occurring words in the RFD's is shown in Figure 2. Stop-words (common English words that do not provide much insight such as "the" or "and") were removed prior to the generation of the word cloud. We observe that the remaining most frequent words are generally biological, medical, or laboratory terminology such as "specimen", "assay", and "reactive". We also note the presence of abbreviations such as "hiv" and "hbv", and lab techniques such as "pcr".

## 4.2 Baseline Machine Learning Classifiers

We determined that a machine learning approach would not be appropriate for `level_2` due to the number of classes and some classes having only one example in our dataset. We revisit `level_2` in Section 4.5.

We start by experimenting with standard machine learning classifiers using only the RFD field for the `test_performed`, `test_outcome` and `level_1` outputs. We describe our data preprocessing procedure in Section 4.2.1 and summarize the results in Section 4.2.2.

### 4.2.1 Data Preprocessing

Due to the presence of free-form text in the RFD's, there are many issues with the content that would be problematic if we were to train machine learning classifiers using the RFD's as-is.

For example, many RFD's consist of a single number or contain meaningless symbols. We start by cleaning and transforming the data into a usable form.

We start by removing all the empty RFD's and RFD's consisting of only a single number from our dataset. For the remaining RFD's, we remove all irrelevant symbol characters (e.g. '`_`') from the text. We then replace all standalone numbers (e.g. "16") by a special token to prevent our classifiers from overfitting to specific number values. Lower-casing the existing test result labels is also necessary as there are inconsistencies such as '`Yes`' and '`yes`'. Another potential issue in the data is the presence of specific dates in different formats. We choose to ignore the dates due to the lack of time and usable tools that could identify all the differing forms of a date. In future work, it may be reasonable to replace all dates by a special date token similar to what we did with numbers.

After cleaning the data, we transform the RFD's into numeric vectors using the *bag-of-words* representation and drop the low frequency tokens (less than 10 occurrences in our analysis). We experimented with other representations such as TF-IDF and positioning-based forms, but found no significant differences in the results between using those and bag-of-words. For `test_performed`, we find that using unigrams to trigrams resulted in the best performance while there is little to no difference compared to just using unigrams for the other outputs.

### 4.2.2 Classifier Results

The standard machine learning classifiers we experiment with include *naïve Bayes* (NB), L2-regularized *logistic regression* (LR), 100-tree *random forest* (RF), and linear-kernel *support vector machine* (SVM). To evaluate the classifiers, we perform 5-fold cross validation and average the test accuracy across the five folds. The classifiers achieve high test accuracies in the case of `test_performed` as seen in Figure 3. The trigram "test not performed" is seen to be the strongest predictor across all our models. Close inspection of the resulting confusion matrices however show symptoms of bias towards the majority class. We address this issue in Section 4.3.

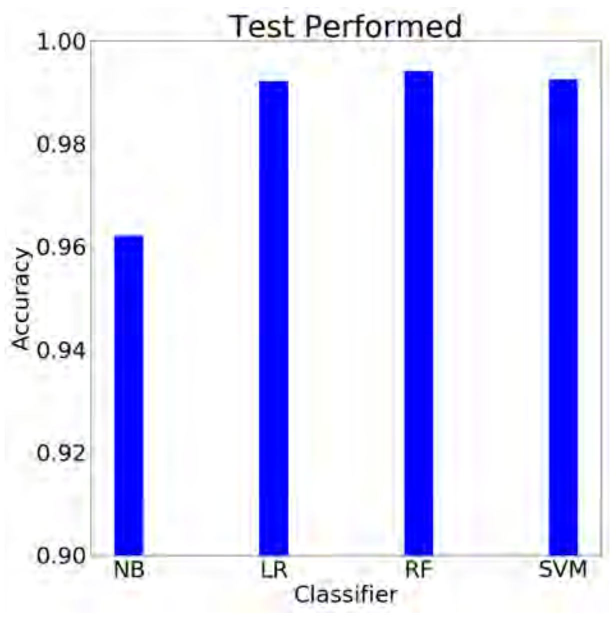


Figure 3: Baseline classifier performances for `test_performed`.

In the case of `test_outcome`, the standard machine learning classifiers achieve similar results. The previously mentioned class imbalance issue does not appear to significantly impact results for `test outcome`. Figure 4 shows the classifiers achieve around 95% test accuracy for NB and around 99% for the others.

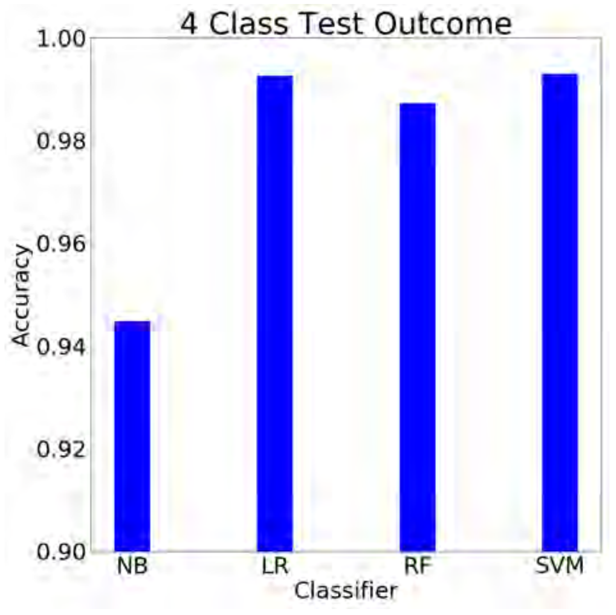


Figure 4: Baseline classifier performances for `test_outcome`.

The `test_outcome` classifiers retain their high accuracy scores even when feature selection is done to reduce the size of the feature space. We opted for `scikit`'s chi-square feature selection algorithm for ease of use and convenience. From Figure 5, we see that only around 150 features are required for the performance of the LR, RF, and SVM classifiers to asymptote. We see a similar trend in the case of `test_performed`.

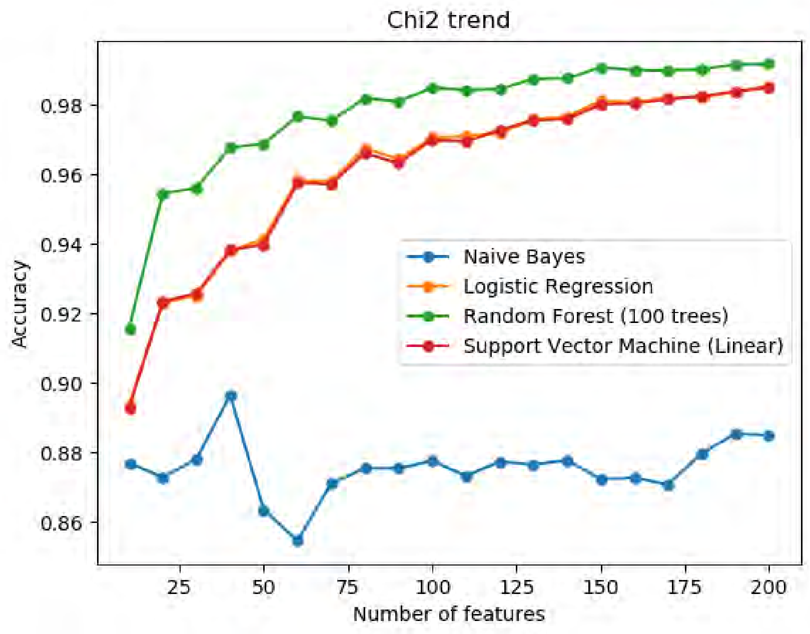


Figure 5: Test accuracy versus number of features retained for `test_outcome`.

From Figure 6, we observe that the `test_outcome` classifiers also retain their accuracy when the training and test set sizes are reduced. This indicates that our text data may contain more structure than we initially expected. Intuitively, this structure may be due to certain snippets of the text that were auto-generated from standardized lab codes.

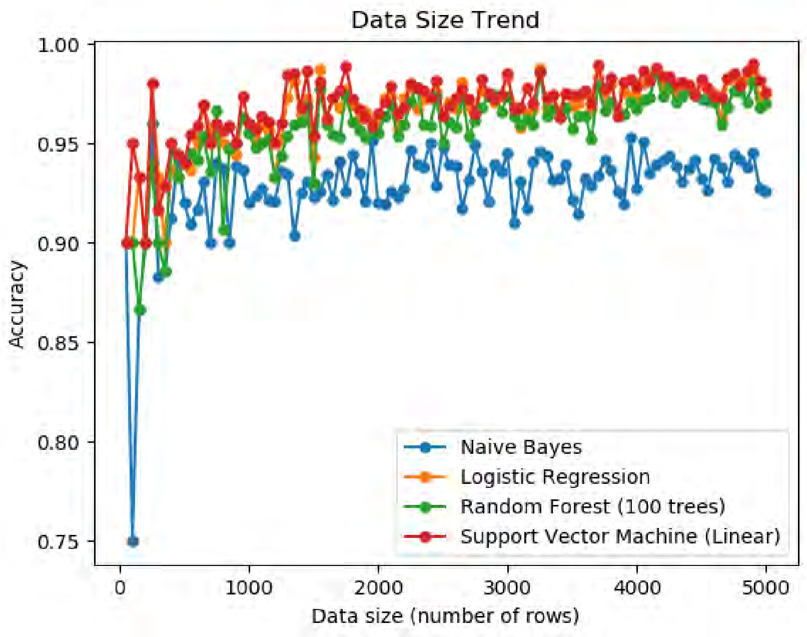


Figure 6: Test accuracy versus size of training set for `test_outcome`.

Error analysis shows that many of the misclassified test results contain contradictory RFD's, where observations at the start are discordant with observations at the end. For example, the RFD

> *Bordetella pertussis DNA detected by PCR. | CORRECTED ON 14/Oct AT 1554: PREVIOUSLY REPORTED AS No Bordetella pertussis DNA detected by PCR. | Culture results to follow.*

is interpreted as "*Bordetella pertussis* DNA was actually detected despite being previously reported as not detected". Our classifiers struggle more with cases like this one where they predict *Negative* for `test_outcome` when the true label is *Positive*. We also note though that our classifiers do correctly classify a significant number of these contradictory test results, and correctly classify the majority of the non-contradictory test results.

The standard machine learning classifiers also achieve high test accuracies for `level_1`. The recall and precision scores vary greatly between classes, where the classifiers appear to predict better on the classes having more data and more structured RFD's. The important features as reported by RF consist mostly of organism-specific terms such as "salmonella", "streptococcus", and "hiv". This provides some support that our classifiers are learning reasonable patterns for predicting `level_1`. However, we also reconsider the use of machine learning for labeling `level_1`. We save this discussion until after we revisit `level_1` in Sections 4.5 and 4.5.1.

### 4.3 Class Imbalance

The `test_performed` classifiers achieved high test accuracies (i.e. well above 95%). Upon closer investigation however, the resulting confusion matrices show signs that our classifiers are heavily biased toward the majority class *Yes*. Table 3 shows the confusion matrix for one of the folds. Specifically, note that the recall score for rows with true class *No* is 79%, which is noticeably lower than the other scores.

| | **Predicted *Yes*** | **Predicted *No*** | **Recall** |
|---|---|---|---|
| **True *Yes*** | 67,696 | 411 | 99% |
| **True *No*** | 947 | 3,475 | 79% |
| **Precision** | 99% | 89% | |

Table 3: The `test_performed` confusion matrix for one of the folds.

We consider three possible solutions for the class imbalance problem observed in the baseline result.

1. *Downsampling*, where rows with true class *Yes* are randomly removed from the training set until the training set is balanced. The disadvantage of this approach is that due too how imbalanced the classes are, much of the data is left unused. This can also lead to an overfitting problem where the classifiers only learn patterns from the few remaining data that is left to train on.

2. *Upsampling*, where rows with true class *No* are randomly duplicated in the training set until the training set is balanced. The disadvantage of this approach is the impact on training time. By duplicating rows until the training set is balanced, we are almost doubling the size of the training set.

3. *Class weighting*, where we introduce heavy penalties (weights) when the classifier misclassifies the minority class during training. One point to consider with this approach is the choice of weights. Because the weights may be arbitrarily chosen, it is arguable that this approach introduces a bias that may or may not be non-trivial.

We find that the approaches above reduce the number of false positives (rows with true class *Yes* that are predicted as *No*), but do so at the expense of introducing many more false negatives (rows with true class *No* that are predicted as *Yes*). Table 4 shows the confusion matrix for one of the folds when the classes are weighted. In particular, the recall score for rows with true class *No* improves, but the precision score for rows with predicted class *No* decreases.

|  | **Predicted *Yes*** | **Predicted *No*** | **Recall** |
|---|---|---|---|
| **True *Yes*** | 66,355 | 1,800 | 97% |
| **True *No*** | 429 | 3,945 | 90% |
| **Precision** | 99% | 69% |  |

Table 4: The `test_performed` confusion matrix for one fold when class weighting is used.

We consulted a domain expert and determined that false positives are more acceptable than false negatives. Hence we retain the original `test_performed` classifiers without the additional solutions for addressing class imbalance in our pipeline code.

## 4.4 MetaMap

MetaMap is a tool that can identify and annotate input text based on concepts in the Unified Medical Language System (UMLS) medical vocabulary. In particular, MetaMap can identify words as concepts within specific *Semantic Types* such as *Finding* or *Bacterium*. Figure 7 shows an example of MetaMap's annotation. Further details regarding MetaMap's output can be found in our separate **MetaMap Technical Documentation** produced along with this report.

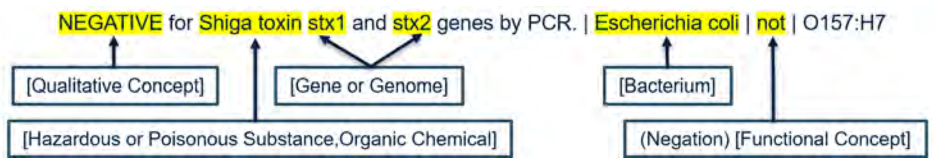

NEGATIVE for Shiga toxin stx1 and stx2 genes by PCR. | Escherichia coli | not | O157:H7
[Qualitative Concept]
[Gene or Genome]
[Bacterium]
[Hazardous or Poisonous Substance,Organic Chemical]
(Negation) [Functional Concept]

Figure 7: Example of concepts (yellow) and corresponding *Semantic Types* identified by MetaMap.

We experiment with MetaMap in an attempt to improve the generalizability of our classifiers. As part of the preprocessing procedure, we send all the RFD's to MetaMap and store the annotations for each RFD. We then apply the unigram bag-of-words transformation on only the annotations, which results in a feature set of approximately 100 unique semantic types. The classifiers trained on this feature set achieve approximately 95% test accuracy, which is slightly worse than what our baseline classifiers achieved. However, the classifiers also do not appear to generalize as well as expected to simple example sentences that we create. Further analysis shows that perhaps some of the MetaMap annotations are too general and that using only the annotations results in significant loss of meaning. For example, the words "positive" and "negative" are both annotated as *Finding*. The semantics of these words would be important particularly in the case of `test_outcome`, and so the generalization from MetaMap does more harm than good in situations like this.

## 4.5 Symbolic Approach to Microorganism Name

Though using the MetaMap annotations as features did not improve the performance of our classifiers, we recognize that there may be value in MetaMap for other approaches. In particular, we consider a non-machine learning approach for labeling the `level_1` and `level_2` outputs.

In what we call the symbolic approach, we again use MetaMap to annotate the RFD's. For each RFD, we then collect all the terms in which MetaMap identified as *Bacterium* or *Virus* into a *candidates* list. Our output label would then be one of the names in the candidates. The advantage of using MetaMap here is that not only can it recognize microorganism names as an entity, it can also recognize abbreviations of the names (e.g. *E. coli*) and map them to their full *preferred* name (e.g. *Escherichia coli*).

The crucial step in this approach is how the final label is chosen from the candidates. In our algorithm, we use a pre-constructed dictionary based on the existing `level_1` and `level_2` labels. If a name in the candidates exists in the dictionary, we prioritize that name as the output. Otherwise we arbitrarily pick a name from the candidates. While this approach is somewhat naïve, we obtain 90% accuracy on `level_1` and 50% accuracy on `level_2`.

### 4.5.1 Machine Learning Versus Symbolic

We have tried both a machine learning approach and a symbolic approach for `level_1` and `level_2`. We briefly discuss the advantages and disadvantages of both approaches in this section.

The machine learning approach has shown better performance on the microorganism name outputs overall. The approach also requires minimal manual configuration and is simple to implement. However, the key downside is that the trained classifier cannot label previously unseen microorganism names. How the classifier is labeling the microorganism name can also be difficult to interpret and adjust if desirable to do so.

Our symbolic approach does not do as well compared to the machine learning approach. The algorithm also fails when a microorganism is detected but its name does not appear in the test RFD. On the other hand, the approach does allow the labeling of new microorganism names unlike the machine learning approach. The approach is also transparent in terms of how it is labeling the outputs, and can easily be modified to fit more complex algorithms to improve results.

In summary, the machine learning approach performs better compared to our current symbolic algorithm, but the symbolic approach shows more promise as it can grow and be adjusted as needed. Specifically, the construction of the dictionary and how the algorithm chooses the label from the candidates are the key steps that have much potential for improvement.

## 4.6 Unlabeled Data Evaluation

To further evaluate our algorithms, we predict on new data and manually review the results with the BCCDC. The three sample datasets we use for this purpose include:

D1. An external set of 29 test results that the BCCDC had microorganism name previously labeled as *Not in Hierarchy* that were re-labeled at a later date. It is expected that this set is most similar to the test results we have in our training dataset.

D2. A random sample of 100 unlabeled `test_code` = `CULT` test results in our dataset. It is expected that this set is somewhat similar to the labeled test results in our dataset which we are training on.

D3. A completely random sample of 100 unlabeled test results in our dataset. It is expected that there may be test results in this set that are very different from the ones we are training one.

Based on our evaluation, the results can be summarized as follows.

1. Our classifiers generally performed well on D1 except on `test_outcome` where *Missing* was the dominating (incorrect) label.
2. Our classifiers did not perform well on D2 and D3. The *Missing* `test_outcome` issue is still present, and *Not Found* is the dominating label for the microorganism name outputs.

One hypothesis that we make is that our classifiers were overfitting to specific groups of test results, such as those involving *Hepatitis C Virus*, because those groups are over-represented in our training dataset. Test results within these groups have very similar RFD's and labels, and so it is very likely that our classifiers try to correctly predict on these test results as much as possible. This might result in poor generalization to other test results that are lesser represented in our training set.

## 4.7 Split by Test Type

Our results from predicting on the new datasets showed that our classifiers seemed to be overfitting to certain subsets of test results that are over-represented in our dataset. To test our hypothesis that our dataset consists of test results from different "true" distributions, we experiment by splitting our dataset into subsets where each subset contains test results of a certain `test_type`.

In particular, we consider the subset of `Antibody` test results and the subset of `NAT-PCR` test results. We train our classifier on the `Antibody` subset and use it to make predictions on the

`NAT-PCR` subset. Our results show that the classifier only achieved 30% accuracy on the `NAT-PCR` test set. This implies that the RFD's of tests of different types may have very different structure (i.e., they come from different distributions). This provides some evidence for our hypothesis, and suggests that we may achieve better results by training classifiers specific to certain subsets rather than training a single classifier on one dataset that may contain data from multiple distributions. We further discuss this idea as future work in Section 6.2.

## 4.8 MetaMap-SNOMED Dictionary

As noted in the discussion of our symbolic approach for labeling microorganism name, one of the key components of our algorithm is the microorganism name dictionary. One problem with constructing the dictionary based on the existing labels is that the naming hierarchy used by the BCCDC is inconsistent. For example, `level_1` is assumed to describe the genus of the microorganism, but also contains classes such as *Hepatitis C Virus* which falls under the lower taxonomic ranking species *Hepacivirus C*. Thus rather than creating a dictionary based on the inconsistent hierarchy, we explore the idea of using MetaMap to create a dictionary of standard microorganism concepts that can then be mapped to the BCCDC's current hierarchy or a future improved version.

To create an initial version of this dictionary, we have MetaMap annotate each of the RFD's and collect all of the *Bacterium* and *Virus* concepts. We then map each MetaMap concept to their corresponding Systematized Nomenclature of Medicine (SNOMED) code using the UMLS REST API. Using the SNOMED codes, we can then retrieve the partial hierarchy of each concept. Through this process, we obtain not only the concept identified but also its "parent" and "children" concepts in its hierarchy. Also, because the SNOMED codes are relatively standardized, the concepts can be mapped to equivalent concepts in other medical vocabularies.

We believe that this dictionary would be suitable for our purposes. Our construction also allows easy extension or modification of the dictionary as necessary. While we do not use this dictionary in our pipeline code, we hope that this idea can be further explored in future work.

## 4.9 Observation-level Classification

The BCCDC currently only labels one class per output for each test result. However, there exist instances where a test result identifies a co-infection. In these cases, multiple microorganisms are found but the given label only names one of them. While not directly in scope of this project, we experiment with the capability of identifying multiple microorganisms and their respective test outcomes by making a simple modification to our algorithm.

For each test result, we separate its RFD into multiple observations using the `obs_seq_nbr` data field. We then apply our classification algorithms on the observations rather than the RFD's.

One important thing to note is that because we do not have existing labels at the observation level, we still train our classifiers on the RFD's and only use the observations when predicting on new data. For this reason, it is also difficult to produce quantitative measures of how well this approach does. Based on our qualitative evaluation, the approach seems to have potential but would require further consideration particularly in how the labels of each observation are aggregated back into the test result level.

## 5 Pipeline Architecture

As one of the main goals of the project, we provide example code implementing the approaches we found to be appropriate in the form of a pipeline. The technical details, design decisions, and explanations of how to use our pipeline are provided in the **Pipeline Technical Documentation** produced along with this report.

## 6 Discussion

We discuss our work and results in this section. We describe the limitations of our work and some points to consider in Section 6.1. We highlight the possible next steps based on where we ended our work in Section 6.2. We provide some recommendations to BCCDC regarding how they store their data to better facilitate efficient analysis in Section 6.3. We then end our report with a brief reflection of the project and its relevance to "social good" in Sections 6.4 and 6.5.

### 6.1 Limitations

As demonstrated, machine learning and NLP solutions have shown promise in the automated labeling of laboratory test results. However, we also caution of using our results as-is. Due to the short timeframe of the program, our intention and focus was to understand what algorithms could be appropriate for automating the labeling process, and not which specific algorithm should be used in a production setting. While the machine learning classifiers we experimented with have achieved good results with our dataset, we cannot guarantee the same with new data particularly when the new data is very different from the data rows that the classifiers were trained on.

One should also be careful when interpreting the fitted parameters of the models. While we are performing feature selection in an attempt to retain only the important features, there may still be instances where important features are dropped, unimportant features are retained, or strong correlations exist between multiple features. These situations are a consequence of the large initial feature space and may lead to unexpected parameter values during training.

### 6.2 Future Work

Our results showed that the classifiers achieve high accuracy on our test sets, but do not generalize well to similar examples believed to have come from a different distribution. The immediate next steps following our work could focus on generalizing our classifiers in order to improve predictions on similar data. For example, removing dates that frequently appear in the result descriptions may require the use of specialized packages or custom regular expressions but would reduce possible overfitting to these tokens. Further incorporating MetaMap's generic annotations in ways that we did not experiment with may also be another direction worth considering.

It is also worth considering an approach where different sets of classifiers are fit to different subsets of the test results. As discussed in Section 4.7, there is evidence that tests of different test types are different enough that a classifier trained on one type of tests may not perform well on another. Additional training data would likely be needed to attempt this however depending on the level of granularity at which the training dataset is split up and the size of each resulting subset.

We found the microorganism name outputs challenging to work with based on what was available. The pure machine learning approach achieved better results on `level_1` but is unable to identify new organisms beyond what is in the training set. The symbolic approach did not do as well, but is more appropriate for the task at hand. While our symbolic approach could also be refined with more complex logic, we suggest revisiting this output if a standardized microorganism dictionary is constructed in the future as per our recommendations in Section 6.3.

Our pipeline architecture can also be easily modified to classify the other outputs of interest shown in Table 5 that are not in scope of this project. Tuning the performance for these outputs will likely require further analysis and experiments.

| **Output** | **Description** |
|---|---|
| Proficiency Test | if the test was performed only to test laboratory equipment |
| Level 3<br>Level 4 | lower taxonomic ranks of the microorganism being tested |
| Toxigenicity | whether the microorganism is toxic |

Table 5: The other outputs of interest not in scope of this project.

## 6.3 Recommendations for the BCCDC

As one of the goals of the project, we've identified the following (idealistic) changes that would simplify future analysis:

1. Enforcing structured results

The need for a machine learning or NLP solution is due to the presence of free-form text. By enforcing only coded results (with dynamic inputs for numeric results of a test for example), the coded results themselves would already be labeled with the desired information. This eliminates the need for any classification algorithm and the issue reduces to a simple querying problem.

2. Updating outdated interfaces and infrastructure

Related to the above, the common use of free-form text is partially because the test result field is over-abused and doubling as a space to include extra notes not necessarily related to the result of a test. This signals the need of a separate field specifically for storing additional comments.

3. Standardizing microorganism names

The current level-hierarchy used for microorganism labels is inconsistent. The questionable use of the incorrect `Influzena` label rather than the correct name *influenza* also points towards the need of a formal standardized microorganism dictionary. As we explored in Section 4.8, MetaMap can be used to create an extendible dictionary suitable for this purpose.

## 6.4 Impact on Social Good

We end this report with a comment on this project's impact on "social good". As touched on previously, one of the primary goals of the DSSG program is to explore and expose how data science could be applied to benefit society and people's well-being in some aspect. In this project, we evaluated potential classifiers for labeling laboratory test results. These labeled test results are intended to be aggregated at an episode-level, and the episodes are to be further aggregated at a lab-case- and case-level. By building efficient and accurate classifiers at the test-level, we lay the groundwork for future work in automating the detection of cases of reportable diseases.

Such a system would have direct implications on population health. An accurate automated system would reduce the time needed to identify cases manually, and would also minimize the number of human-made errors. As a result, accurate statistics of reportable disease cases would be readily available in a shorter period of time. Health authorities could then use these statistics to more quickly inform and guide decisions relevant to the improvement of public health.

### 6.5 Conclusion

We hope that our work in this project has provided insight to the BCCDC regarding the possibilities that they hold with their data. We also hope that this report highlights the unique challenges of working with health care data. We thank all those who were involved and provided guidance with the project, and thank the UBC DSI, the BCCDC and Microsoft for making this experience possible.

## 7 Appendix

### 7.1 Proportion of Labeled Data

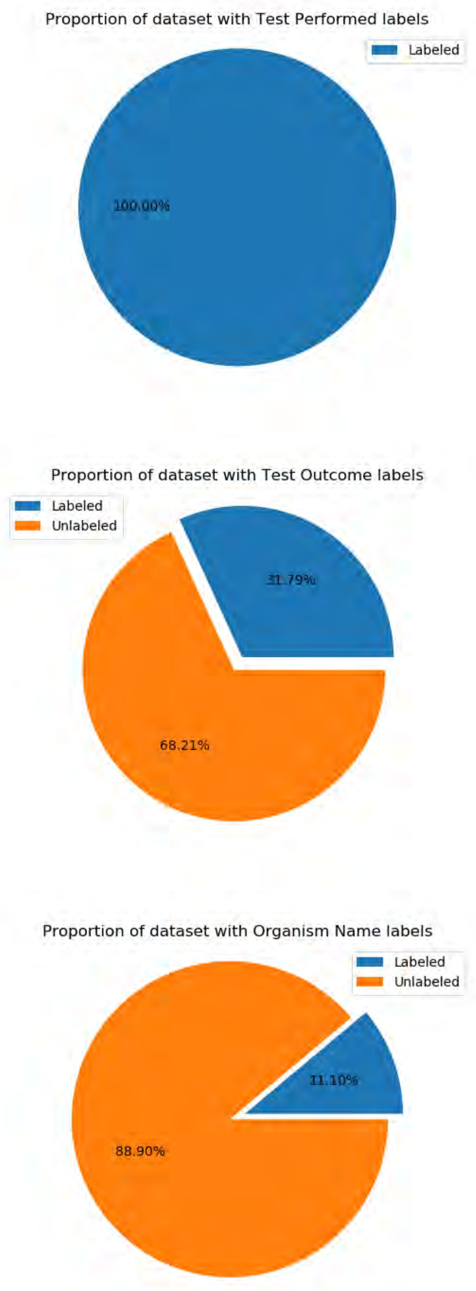


Figure 8: The proportion of labeled data for each output (approximately 950,000 rows total).

## 7.2 Class Breakdown of Labeled Data

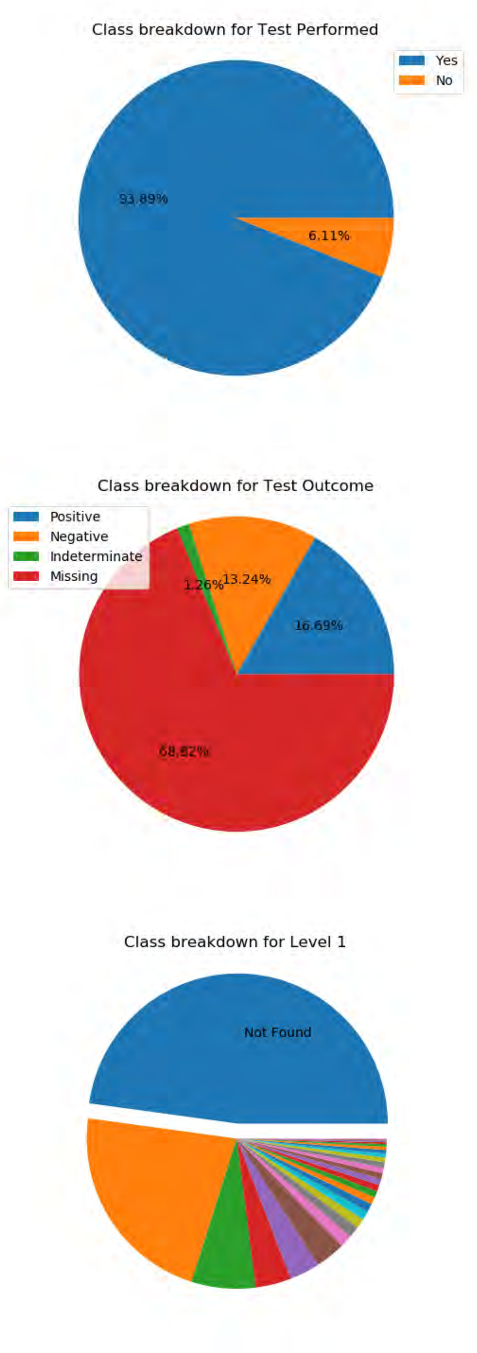


Figure 9: The class breakdown of labeled data for each output.

## 7.3 Test Outcome Binary Case Analysis

*Note: the analysis described in this section uses an early version of the data preprocessing step described in Section 4.2.1. This section is included only to provide further insight into the data but the results may no longer be applicable.*

Before we start our `test_outcome` analysis with all four classes, we run preliminary experiments on the two-class case with classes *Positive* and *Negative*. The baseline classifier results are very similar to the four-class case as can been in Figure 10.

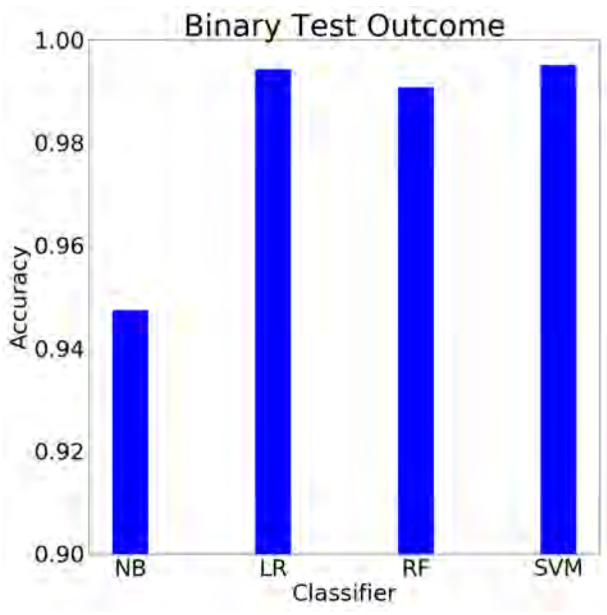


Figure 10: Baseline classifier performances for binary `test_outcome`.

Figure 11 shows the "important" features as per LR weights. Note that this interpretation of feature importance should be taken cautiously as there may be strong correlations between features that may affect the LR weights in unexpected ways. We observe that although the most important features ("no", "non", "nonreactive", "positive") are intuitively relevant for predicting `test_outcome`, other features include some purely-numerical features ("16", "30", "2006", "50") and some organism-specific features ("hbv" related to *Hepatitis B*) which may be surprising. This implies that our classifiers are fitting to non-generalizable words. Future work may focus on using domain knowledge to produce a list of non-generalizable tokens to remove from the feature space.

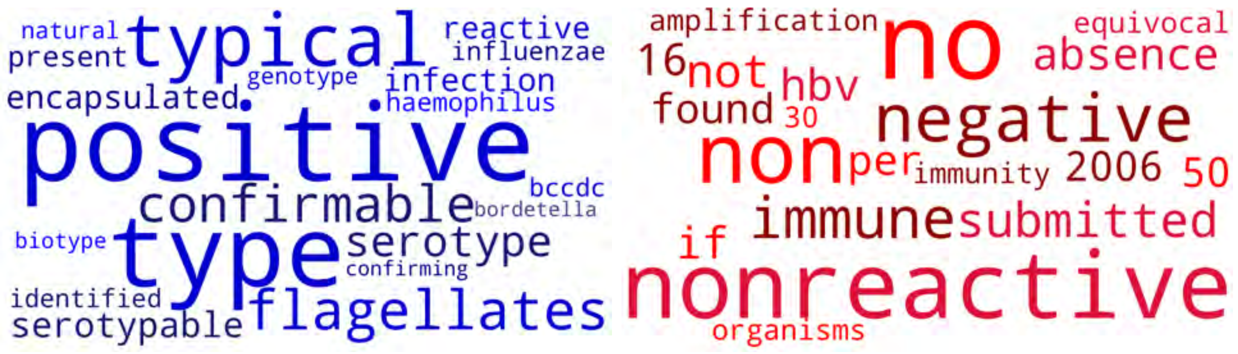


Figure 11: Most important features for *Positive* (blue) and *Negative* (red) `test_outcome` according to LR.

## 8 References

[1] Jang, H., Song, S. K. & Myaeng, S. H. (2006). Text mining for medical documents using a hidden Markov model. *Lecture Notes in Computer Science*, 4182, 553.

[2] Kang, Y. S. & Kayaalp, M. (2013). Extracting laboratory test information from biomedical text. *Journal of Pathology Informatics*, 4, 23.

[3] Spasić, I., Livsey, J., Keane, J. A. & Nenadić, G. (2014). Text mining of cancer-related information: Review of current status and future directions. *International Journal of Medical Informatics*, 83(9), 605.