Mushroom Analysis: A Guide to Predicting Mushroom Lethality

Andrew Leonard
Santa Clara University
aleonard@scu.edu

Dominic Komarek
Santa Clara University
dkomarek@scu.edu

Alton Chau
Santa Clara University
achau@scu.edu

John Guthmiller
Santa Clara University
jguthmiller@scu.edu

Abstract—**The premise of our project aims to classify mushrooms based on a set of external features that allow for their classification as edible or poisonous. Edible natural resources can be essential to outdoorsman, survival enthusiasts, and even biological study. One can classify mushrooms through physical features and continuous biological testing, but a predictive model eliminates mushrooms from potential categories and predicts their category placement as edible or poisonous. This early category elimination and placement allows for a much faster, more efficient classification method than any method without the use of machine learning. The group hopes to create a system to easily classify mushrooms by the features in order to assist those that need to survive in the wild.**

1.) Introduction:

Mushroom's vary heavily in their external features, creating plethora of fungi across different regions. Within this plethora of mushroom types exists poisonous mushrooms that can kill an unsuspecting, hungry individual exploring regions containing these genera of fungi. Creating a predictive model for testing two genera of mushroom can eliminate the need for biological testing if supplied with sufficient training data. While the data collected and distributed limited the scope of our analysis, the intent of this project is to demonstrate potential for expansion on vegetation classification.

Mushrooms are important organisms that exist as a member of several ecosystems as well as a common part of the human diet, which can lead to potentially fatal situations. Ray LaBarge originally conducted the analysis within the same scope [4]. The group hypothesized that equal or higher accuracies could be achieved using different methods than those employed by the original analysis. In combination with replicating equal or more accurate results, the group also aimed to create a baseline for further analyses by both biologists and other data scientists.

2.) Potential Applications:

The potential applications for this analysis exist mainly within field testing. The group achieved potential for field testing by creating a graphic user interface for Android mobile devices that allows one to employ Weka's machine learning algorithms. Beyond the applications found within field testing, the applications of our analysis seem to possess enormous potential with other edible vegetation known to include poisonous variants. Within the scope of our analysis, our potential applications would allow one to use the application (Mushroom Analyzer) in a specific region containing two different genera of mushroom [5]. The scope of this analysis likely shrank down to its current size due to instabilities that might arise from introducing more genera of mushroom. The potential of our analysis with an expanded scope in mind is much more practical. Because the group chose to rank features from most to least statistically significant, our group could potentially further develop our analysis methods to work with regions in which each genus of mushroom is specified, therefore only analyzing mushrooms that could potentially be eaten.

One could expand the scope of the analysis further to test foods that can be identified as poisonous through physical traits. While the traits may not be as numerous among other foods, feature-weighting in combination with Weka's implementation of logistic regression (assigning the highest weights to the most indicative features) could potentially play a significant role within this expanded scope for potential applications. One might be able to analyze berries, nuts, or other naturally occurring foods with poisonous genera within their family. To expand further one might argue that any food with physical traits indicative of toxicity could be analyzed using our analysis method as a starting point. The group's chosen analysis toolkit can work with big data, meaning that one could potentially develop a near-perfect classification system depending on the abundance of data on external vegetation features.

3.) Related Work and Data:

3.1) Weka's Data

The group's original, provided data came in the form of a csv file, requiring some manipulation for data-to-toolkit compatibility reasons. A major advantage of the related data that the group found was that previous analysts encountered the same issues, leading a contributor to Weka's toolkits to create repositories containing the exact same data the group received in the form of an arff file. Because arff is a compatible file format with Weka, the file found within this repository significantly expedited the analysis and visualization process.

3.2) Previous Analysis: Ray LaBarge

Ray Labarge's 2008 study employed the K-nearest neighbors classification algorithm, as well as a developed logical ruleset. The group initially intended to implement a logical ruleset and K-nearest neighbors hybrid algorithm to produce results based on explicit decision boundaries developed by each group member. The issue of reliability quickly arose, as a homemade implementation of any logical ruleset requires massive amounts of data to train, test, and verify as reliable. Unfortunately, the scope of the group's analysis required the use of an open-source toolkit that did not contain the implementation the group was looking for. The need for a new algorithm led our group to use LaBarge's study as an opportunity to identify and work-around potential issues encountered using the methods in his analysis.

4.) Algorithms:

4.1) Naive Bayes:

Initially, the group chose to pursue the naive bayes algorithm. The decision to use naive bayes was based off initial thought that naive bayes is an excellent choice as a classification algorithm in general cases. The group thought that regardless of the results, the naive bayes algorithm would provide a great baseline for the rest of the project.

Other factors also influenced the group’s decision to use this algorithm. Naive bayes is known to perform well with smaller datasets. The data set obtained by the group contained 8124 instances of mushrooms. In the grand scheme, though this isn’t a small amount, it also isn’t an extremely large data set either. The group thought that the algorithm would fare well even though the data set was not at optimal size. Another advantage that naive bayes exemplifies is the ability to perform with low runtimes.

4.2) Logistic Regression:

The follow-up algorithm used in this project was logistic regression. Logistic regression is normally used in cases where binary results are produced. Quite conveniently, the classification the group faced in this project was to determine if the examined mushroom was poisonous or edible, a binary classification. This algorithm became a very ideal fit for the group’s project. However, as described above, the project data set is just above 8000 instances. Logistic regression tends to overfit data if the data set is too small. But through experiments done in this project, the group could determine that 8124 items in the data set was more than enough for logistic regression. Since this was the case, logistic regression does much better than naive bayes if the data is correlated, which happened to be the case. The group concluded that the logistic regression algorithm would be not only the most ideal classification method for this analysis at the current phase of analysis, but also in the long run if more mushrooms were to be added later.

5.) Analysis Procedure:

5.1) Exploratory Data Analysis

The exploratory data analysis was crucial to the success of the project. The group aimed to test individual mushrooms by using the smallest number of features without sacrificing accuracy. Each mushroom in our data set is represented using a character array of 22 mushroom features with a classification (e or p). For example, figure 1 shows an edible mushroom due to the ‘e’ at the beginning of the array. For example, the x at the beginning of the character array means that the cap shape of the mushroom is convex. More information about the features is available at the UCI machine learning repository [3].

e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

Figure 1: An example representation of a mushroom

The group had to figure out which features to keep and which features to remove based on how influential and important it was to the final classification. The approach to this problem consisted of running 6 different ranker algorithms on our test data using the Weka application. One of the ranker algorithms the group used is called information gain, which evaluates the worth of an attribute by measuring the information gain with respect to the class. Fortunately, these algorithms are native in Weka and incredibly easy to call from the user interface. These six algorithms returned a unique sequence of features, and each feature was then given a specific rank. The rank of an attribute represents how important that feature was in deciding a mushrooms edibility. For example, figure 2 shows our most important feature, which was odor, compared to our least important feature, which was Veil-type. In this diagram, blue is the edible class and red is the poisonous class.

You can see that if a mushroom has a specific odor, it is most likely going to be either poisonous or edible. But, if you look at Veil-type, just because the veil-type is partial, does not tell you anything because of the near 50-50 split. The group then map reduced the six sequences into a master sequence and then found the mode of each attribute. Figure 3 shows the top nine most class-indicative features.

The group then ran the classification algorithms on different variations of these features to find the most accurate combination. The resulting combination with the lowest number of features and highest accuracy was attributes 5 (Odor), 7 (Gill Spacing), 12 (Stalk-Surface-Above-Ring), 20 (Spore Print Color) and 21 (Population). Decreasing the size of attributes was essential to the project’s success. Originally, when testing a single mushroom, one would have to enter 22 features. This feature elimination was excessive and hurt the resulting accuracy. Now, with the reduced features, one would have to enter in a total of five features. This also helped increase accuracy. For example, when testing the Naive Bayes classifier with 10-fold cross validation and all 22 features, Weka produced 95.8% accuracy. On the other hand, when testing the Naive Bayes classifier with 10-fold cross validation and the reduced feature set (five features), Weka produced 98.9% accuracy.

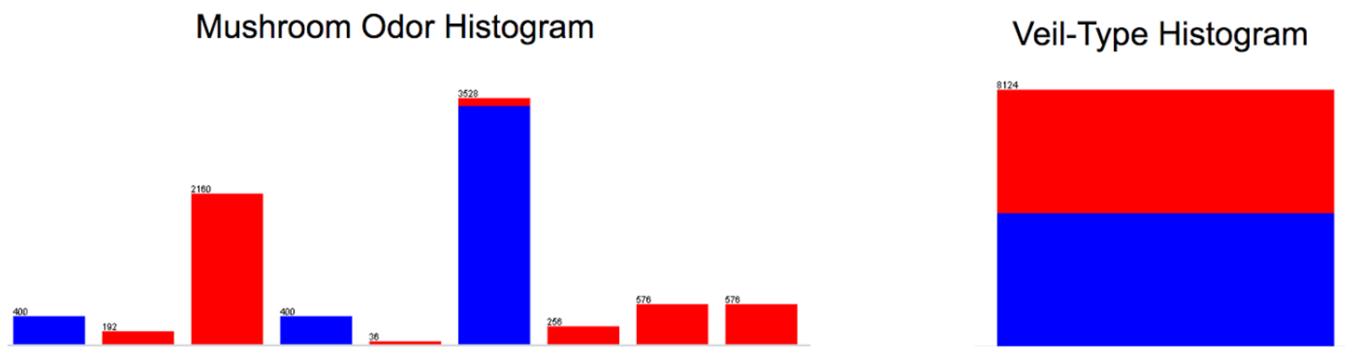


Figure 2: Distribution of mushrooms shown based on category [1]

The group concluded that this is because of how Naive Bayes calculates how features are weighed. With less but more statistically significant features, the group achieved higher accuracy.

5.2) Choosing A Base Algorithm

Identifying the algorithm that maximized efficiency required a cyclical trial-and-error style of process consisting of the steps: 1.) Process, clean, and manipulate our data until encountering an issue with the current base algorithm. 2.) Identify the issue as a problem with the features chosen, or as an issue with the base algorithm. 3.) Replace the features or replace our base algorithm. After several trial-and-error cycles, the group encountered success with a logistic regression algorithm provided by Weka and data cleaned to exclude features that provided no shape to classification boundaries.

6.) Methods:

6.1) Weka: What Is It?

Weka is an open-source toolkit developed and released about one year ago (April 2016) for portable, efficient machine learning. Weka's website describes the suite as "Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes" [1].

6.2) Choosing Weka: Practical Advantages

Weka provides a simple user interface and implements Java tools that expedite the data wrangling and manipulation, and is compatible with many more data formats. The mushroom data's original formatting came in the form a csv file containing 8124 lines of single characters, each line representing an observation and each character representing 1 of 22 external mushroom features. The group initially planned to reformat the data so that it could be represented by a binary array indicating if a single trait was present using 1 or 0, leading to an entirely new challenge of finding a toolkit capable of working with such a dataset. Weka provides access to the University of California, Irvine's data repository, including the mushroom data set the group is analyzing. The advantage of Weka's data, however, is its arff file format, Weka's compatible file format [1]. Using a differently formatted file containing the same data expedited the process of wrangling data for compatibility and identifying software capable of working with the original csv format. Ranking features, identifying the subset of insightful features, dimension reduction, and analysis using algorithms became a matter of simply selecting the tools available as a part of Weka's implementations. As one of the most accessible programming languages, Java ensures that Weka is portable and extensible. The option to analyze and work with Weka's open-source implementations ensured that the group (in which all members have Java experience) had an exact understanding of what was happening to the data upon entering each of Weka's functions.

6.3) Working With Weka: Use and Method

Upon discovering Weka, each member of the group took the time to look at different parts of the interface so as to quickly learn the interface. After becoming familiar with the user interface and importing the correctly formatted arff dataset, Weka was able to return the top ten most class-indicative features. Upon noticing that the using the top five feature returned highest accuracies, the data was passed through the three algorithms. The group used three algorithms to test for the highest accuracy: naive bayes, K-

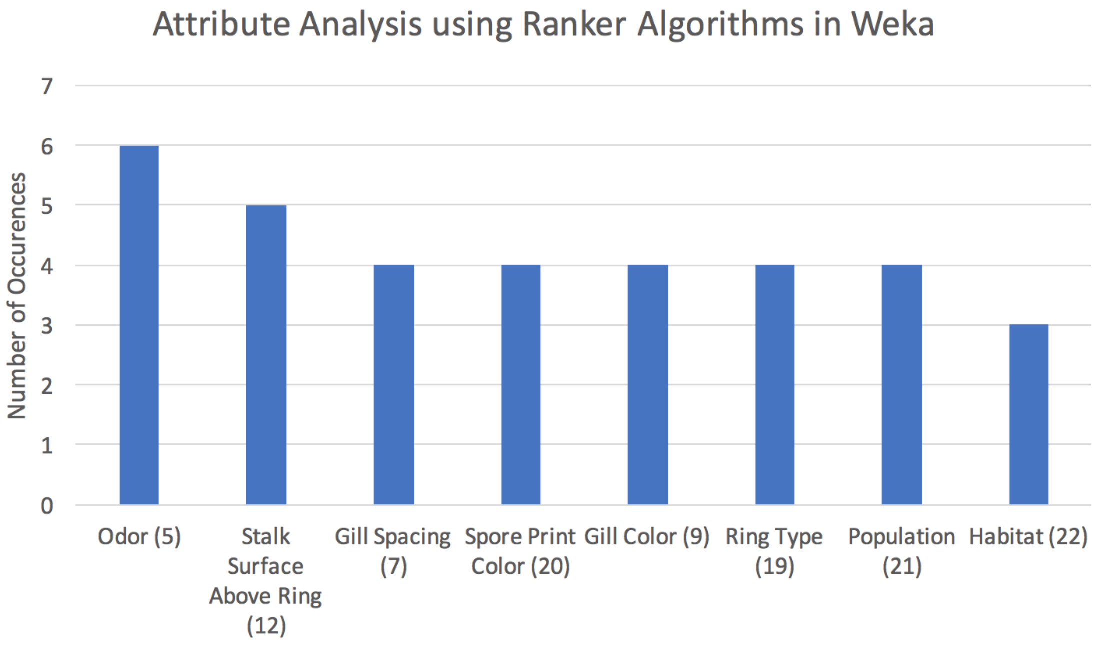


Figure 3: Results of attribute analysis [1]

nearest neighbors, and logistic regression. Data format compatibility in combination with compatible algorithm implementations turned the analysis process into a point-and-click exercise. Much more significant than process simplicity, was the group's expertise to know that the data required further cleaning and manipulation. Weka provided much of the footwork that often occupies the majority of the analysis process, allowing the group to focus on the content of the results and their reliability rather than struggling with the actual interface. The simplicity of Weka's user interface facilitated a change in mindset from completing the analysis to understanding it. This shift in focus led to observing changes in accuracies based on feature presence and algorithm pursuit based on their physical (memory, speed, etc.) performances.

While logistic regression returned 100% accuracy regardless of the feature quantities trialed, naive bayes returned 95.8% accuracy when uncleaned (using the 22 provided features) and 98.9% cleaned accuracy (using the top five indicative features). Upon identifying the most useful features, massive quantities of unrelated data were used to test Weka's implementations for runtime speed and memory efficiency. K-nearest neighbors ran in linear time O(n), while logistic regression and naive bayes ran in linear time O(1). Weka's implementation of the K-nearest neighbors algorithm required intensive memory, mainly because K-nn requires as many comparisons as there are instances of data. Weka's implementation of the naive bayes and logistic regression algorithms returned runtimes that were much faster than K-nearest neighbors. The group made the decision to shift focus from k-nearest neighbors and to the differences between naive bayes and logistic regression for better results with less effort.

7.) Results:

7.1) Optimal Classifier: Logistic Regression:

Logistic regression after the experimental phase returned a 100% classification accuracy rating. Without doubt, an algorithm returning a 100% accuracy will be an optimal algorithm if one can ensure the reliability of their results. The only concern with logistic regression was its tendency to overfit with low amounts of data. Besides the logistic regression algorithm's ability to cater to more indicative features through a ranking algorithm effectively, the advantages of logistic regression exist within the much larger scope of potential application. To abstract one level above logistic regression: the implementation works with individual features to create a predictive model rather than comparing each individual element to every other element or combining features probabilities for a conditional probability with for specific parameters. However, the group successfully avoided overfitting and adopted logistic regression as their base algorithm. In addition, logistic regression allows for a seamless transition for the inclusion of future data. This factor included with the above reasons contributed to logistic regression being our choice.

7.2) Data and Corresponding Classification

```
K-Nearest Neighbors (using 22 features) [1]:
Correctly Classified Instances       8124              100      %
Incorrectly Classified Instances        0                0      %
Kappa statistic                         1
Mean absolute error                     0
Root mean squared error                 0
Relative absolute error             0.0029 %
Root relative squared error          0.003 %
Total Number of Instances            8124

Naive Bayes (using 22 features) [1]:
Correctly Classified Instances       7785               95.8272 %
Incorrectly Classified Instances      339                4.1728 %
Kappa statistic                     0.9162
Mean absolute error                 0.0419
Root mean squared error             0.1757
Relative absolute error              8.397 %
Root relative squared error        35.1617 %
Total Number of Instances            8124

Logistic Regression (using 22 features) [1]:
Correctly Classified Instances       8124              100      %
Incorrectly Classified Instances        0                0      %
Kappa statistic                         1
Mean absolute error                     0
Root mean squared error                 0
Relative absolute error                 0      %
Root relative squared error             0      %
Total Number of Instances            8124
```

Figure 4: Results of K-nn, naive bayes, and logistic regression, respectively, using 22 features.

```
K-Nearest Neighbors (using 5 features) [1]:
Correctly Classified Instances       8124              100      %
Incorrectly Classified Instances        0                0      %
Kappa statistic                         1
Mean absolute error                     0
Root mean squared error                 0
Relative absolute error             0.0003 %
Root relative squared error         0.0006 %
Total Number of Instances            8124

Naive Bayes (using 5 features) [1]:
Correctly Classified Instances       8036               98.9168 %
Incorrectly Classified Instances       88                1.0832 %
Kappa statistic                     0.9783
Mean absolute error                 0.0182
Root mean squared error             0.0859
Relative absolute error             3.6541 %
Root relative squared error        17.1915 %
Total Number of Instances            8124
```

```
Logistic Regression (using 5 features) [1]:
Correctly Classified Instances        8124          100     %
Incorrectly Classified Instances         0            0     %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0     %
Root relative squared error              0     %
Total Number of Instances             8124
```

Figure 5: Results of K-nn, naive bayes, and logistic regression, respectively, using 5 features.

7.3) Quantifiable Outcome

The basic results of the analysis simply reveal if a mushroom is edible or poisonous. Noting the outcome of each mushroom led to an insightful collection of summary statistics. The number of false positives and false negatives (mushrooms incorrectly classified as edible or poisonous, respectively) while using logistic regression and naive bayes were indicative of their performance with the provided data. As shown in Figures 6 and 7, the green boxes represent correctly classified mushrooms whereas the red boxes represent the misclassified mushrooms. False positives are shown in the top right box and false negatives are shown in the bottom left box. As described above, logistic regression returned a perfect classification record while naive bayes returned 95.8%.

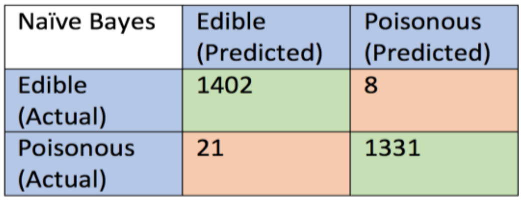
8.) Android Application: Mushroom Analyzer

Not only did the group confirm our hypothesis, but they also created an Android application, called Mushroom Analyzer, capable of carrying out the analysis the group designed in real-life scenarios [5]. The first step was importing Weka into the application. This was not possible, as Android doesn't support some of the libraries that Weka depends on. The group found a modified version of Weka, called Weka for Android, that was available online and open source [2]. This was imported into the application.

The group then sought to create new classifiers in the application. This was deemed as not the optimal method for multiple reasons. Most importantly, it's inefficient, as it requires the app to train exact replicas of the same classifier, which itself requires data set to be included in the app and time that could otherwise be saved. Additionally, it could be prone to error, as it may not be certain the classifier is trained the same way every time. The alternative method that was used involves creating the classifier once on a desktop, saving it to file, and including this file in the application. From there, the classifier could be recreated from file.

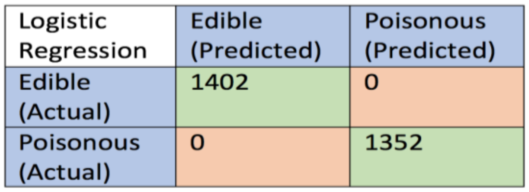
The application interface includes a form which guides the user through a set of questions about the mushroom they wish to test. After completing the form, the user may continue. At this point, the application creates a new instance with the user-given features, recreates the classifier from file, then uses the classifier to classify the instance. The classification is then displayed, along with any similar test mushrooms if there are any.

Two screenshots from the application are below. Figure 8 shows one section of the form that the user answers. Figure



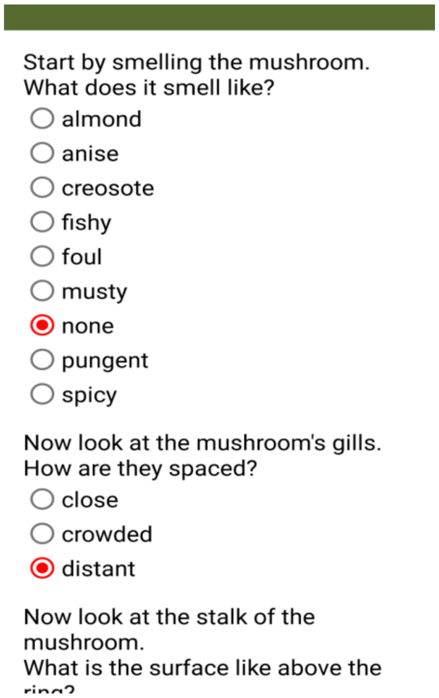
Naïve Bayes	Edible (Predicted)	Poisonous (Predicted)
Edible (Actual)	1402	8
Poisonous (Actual)	21	1331

Figure 6: Naive Bayes Confusion Matrix [1]



Logistic Regression	Edible (Predicted)	Poisonous (Predicted)
Edible (Actual)	1402	0
Poisonous (Actual)	0	1352

Figure 7: Logistic Regression Confusion Matrix [1]



Start by smelling the mushroom.
What does it smell like?

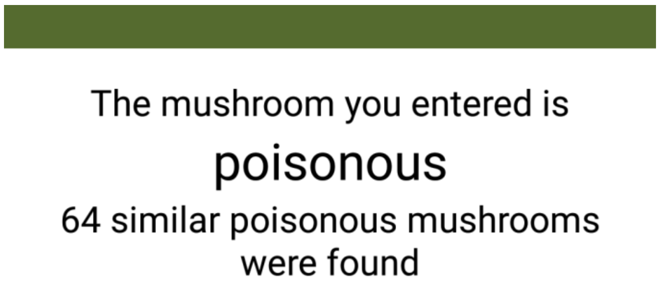
almond
anise
creosote
fishy
foul
musty
[✓ none]
pungent
spicy

Now look at the mushroom's gills.
How are they spaced?

close
crowded
[✓ distant]

Now look at the stalk of the mushroom.
What is the surface like above the

Figure 8: Mushroom Analyzer - GUI Screenshot [5]



The mushroom you entered is
poisonous
64 similar poisonous mushrooms were found

Figure 9: Mushroom Analyzer - GUI Screenshot [5]

9 shows a sample result that might be returned to the user.

9.) Task Distribution:

9.1) Andrew Leonard

Andrew's first focus was implementing Weka in code. Working with Dominic, he read through Weka's open source code along with numerous online guides. This allowed them to create classifiers without the use of the graphical user interface. From there, they could work with them as desired. Some of the tasks performed with these classifiers was creating and testing new observations and saving them to file for later use.

Andrew also created the Android application, Mushroom Analyzer. His prior experience with Android development proved to be invaluable. He also hosted the application on his Play Store account so that it could be downloaded by anyone who wants to use the app.

9.2) Dominic Komarek

Dominic was focused on the exploratory data analysis as well as creating the desktop application. Dominic started off by familiarizing himself with Weka and researching its graphical user interface abilities. After talking with Andrew about the app design Dominic was determined to shrink the number of features used in the classification. This would require an EDA. He started off by researching Weka's ranker algorithms as well as understanding the outcomes displayed by the interface. Once he had run the algorithms in Weka and recorded the results of each ranker in excel, he was able to create a master sequence. Dominic then analyzed this sequence in excel and found the top 8 most statistically influential features. He continued his analysis by testing these 8 features against each other in multiple classification tests. The result was a list of five features that maintained a high accuracy. This was an essential find to the success on our mobile platform.

Dominic also worked on coding up the desktop software which consisted of a Java version of the Jupyter Notebook that other groups decided to pursue, and the app concept. To do this Dominic researched the Weka Javadoc as well as many YouTube tutorials on implementing Weka with Java. Once properly implemented, he formatted the output of the Java program to replicate a python-esque style. Next, Dominic put together the run.py program to compile and run the Java code. He also created the documentation to use it. Dominic was the driving force that led to the finalized desktop application as well as a key element to the creation of the app. Dominic's work was integral in the completion of the group's remaining tasks.

9.3) John Guthmiller

John's role in the project focused heavily on transforming the findings into conveyable information both in the form of writing analysis results effectively, as well as compacting significant information into presentable forms. John also played a substantial role in explicitly defining implicit nuances and differences among the three algorithms. These differences came in the form of memory usage, runtimes, and data format compatibility. As a result of observing these traits, much of John's time contributed to research on both similar analyses and different toolkits. This ensured immediate readiness for each phase of the analysis. Along with all other group members, John learned to use and acquire results from the Weka user interface, while also taking individual notes on the advantages that make Weka an open-source software superior to others. John also took the initiative to consider other analysis toolkits to provide ready-to-start, compatible backup plans if Weka presented issues later in the project.

9.4) Alton Chau

Alton's contribution to the project laid mainly dependent on the knowledge of the algorithms present in this project. Working with Weka and its tools, Alton helped determine the optimal algorithm in different situations. After further extensive research on the benefits and drawbacks of each algorithm considered, Alton advised the group on which algorithm to ultimately use. Alton also spent time on Weka to aid in determining which mushroom categories were most important as well as providing the visualization that accompanied the data. Alton was instrumental in coordinating among group members and was heavily involved in the progression of each implementation of the project.

Conclusion:

To summarize the outcome of our analysis, any mushroom within the scope of the provided mushroom genera can be classified efficiently and accurately. Variation among accuracies returned by all three algorithms gave an idea of the direction the group chose to pursue, but also provide insight as to why choosing each algorithm is advantageous in other scenarios. The original analysis intends to accurately classify mushrooms, but the scope of this analysis also included identifying and supporting the optimal algorithm for the data provided. While the group cannot provide exact accuracy reliability measurements, cooperation with biologists would provide the assurance and expertise to ensure the practical use of our Mushroom Analyzer application in field testing. The only limitations on this analysis is the assigned scope and the availability of data. As both an analysis of mushrooms and algorithm performance, one could employ the group's cyclical trial-and-error method. Potential for analyzing larger regions with more genera or different forms of vegetation entirely now exists with this analysis as a launching point.

Acknowledgements:

Thanks to Jeff Schlimmer for donating the data set and University of California at Irvine for hosting it. Thanks to the Machine Learning Group at the University of Waikato for creating Weka. Thanks to user rjmarsan on GitHub for modifying Weka so it could be used in mobile devices. Most importantly, thanks to Professor Manna for her guidance in this project.

References:

[1] "Weka 3 - Data Mining With Open Source Machine Learning Software In Java". Cs.waikato.ac.nz. Web. 11 June 2017.
[2] "Rjmarsan/Weka-For-Android". GitHub. N.p., 2011. Web. 11 June 2017.
[3] "UCI Machine Learning Repository: Mushroom Data Set". Archive.ics.uci.edu. Web. 11 June 2017.
[4] "Distinguishing Poisonous From Edible Wild Mushrooms". homepages.cae.wisc.edu. N.p., 2008. Web. 11 June 2017.
[5] "Mushroom Analyzer". Play.google.com. N.p., 2017. Web. 11 June 2017.