RNA Repeat Data SSR of Anaplasma Marginale Exploration

Aditya Jasuja*, Minjian Li+
School of Electrical Engineering and Computer Science, Washington State University

*aditya.jasuja@wsu.edu

+minjian.li@wsu.edu

***Abstract*— Short Sequence DNA repeats (SSRs) are relatively short patterns that recur in the genes of many species. With the development of biological evolution, the relationships among organisms are sophisticated. The phenomenon that two or more different kinds of creatures live together is called symbiosis, and parasitism is one of the harmful relationships in such a phenomenon [1]. Short sequence DNA repeats can be identified in all eukaryotic and many prokaryotic genomes. Parasites are nowadays widely discovered in different species such as shrimp, shellfish and cattle, as well as humans. Anaplasma Marginale is one of the cattle blood parasites that graziers are concerned about. The dataset we used in our project was on SSR data for Anaplasma marginale, a pathogen that commonly infects cattle. Our primary focus was towards extracting a meaningful feature set from the SSR data for use in clustering. In order to obtain more useful information from this parasite, several popular data science techniques, in terms of classification, clustering, and visualization have been implemented in this exploration project.**

Keywords—Data Science, Anaplasma Marginale, Classification, Naïve Bayes, RNA Repeat, SSR, Random Forest, Boruta, K-mers

I. Introduction

Data science, or dataology, is a field in computer science studying and exploring the mystery of data generated from the cyber network via specific techniques and methodology, which is corresponding to big data [2]. The main purpose in this field is twofold — to analyze and explore the data itself, and to provide a new pattern for natural and social science research. In the twenty-first century, massive data is generated and stored on the internet. Such data records not only human activities including works in daily life, social communication, and entertainment but also nature, as well as the universe. Therefore, to seek out hidden patterns and rules in data is exactly equal to explore the laws of the universe, nature, and humanity. As the explosion of data on the cyber network, data science is becoming more and more popular than before, and its development prospects can be improved. As a result, data science should be suitable for the exploration of Anaplasma Marginale RNA repeat.

Due to the outbreak of infection of Anaplasma Marginale in cattle, biologists start to focus on studying the RNA repeat SSR of this parasite in order to develop an effective treatment of such disease. Therefore, this project aims at answering how the classification model (Naive Bayes and random forest) classify RNA

repeat SSR of Anaplasma Marginale, and how well they perform. In addition, this project attempts to discuss if exists certain interesting information and useful hints that imply a specific pattern of such parasite.

Short Sequence Repeats are widely and abundantly disseminated in most nuclear eukaryotic genomes and have been widely used as molecular markers in recent years. In the data set of Short sequence repeats, we noticed that all the features were in the format of characters and getting meaningful insights from them was a challenging task. We started by extracting features from the DNA sequences, initially, we applied bag of characters approach to find how many times each of the characters are repeated in each sequence, after that we searched for the length of each sequence, also counted how many times each of the characters were repeated in the whole dataset and found the first character of each sequence. After that, we divided each sequence in the form of bigram, trigram and four grams to check how many repetitions of bigrams, trigrams and four grams are there in each sequence by using n-gram method and library available in R. After extracting features the next step that we took was of feature selection where we removed various irrelevant features to make the data set ready for modeling. Finally, we divided the dataset into training and test set and applied the Naïve Bayes classifier and found a confusion matrix also we applied a feature wrapping technique using the Boruta algorithm and applied Random Forest on the feature set before and after applying Boruta algorithm and compare the confusion matrix. On comparison of both approaches, we found out that by using the extracted features both the models return similar results.

II. Problem Definition

With the purpose of comparing the selected classification model (Naive Bayes and random forest), a set of evaluation rules are required. Such rules are testing the following aspects:

- Accuracy is the most important and the most direct way to evaluate a classifier. Thus far, none of the classification models can reach a hundred percent accuracy on classifying the dataset generated in real life.
- The overfitting problem is the problem that all kinds of classification models are facing. Generally, the accuracy of a classification model on the training set is always higher than that on its testing set. If the accuracy of a classification model on the training set is much higher (more than 15%) than that on its testing set, it is called overfitting. There are several methods to reduce overfitting; one of the most useful methods to do so is normalization.
- A classifier with high stability shows approximately the same performance on different size of the training set.
- Learning speed is also a significant aspect to evaluate a classification model. The learning speed of a classifier depends on its time complexity function. If the time that a classification model spends on is linear, this classification is efficient.

Moreover, since the feature of the given RNA sequences is not remarkable, a variety of techniques in data science are implemented to extract notable features from the dataset.

III. Methods

Feature extraction, feature selection, and classification are three sections implemented at this project to dig out the hidden pattern on the target dataset and to achieve some useful results.

A. Feature Extraction

Feature extraction is the phase in which informative patterns are extracted from the given data. Suppose that there is a dataset recording the information of fruits, the features of this dataset should be color, smell, taste, species, and weight which are distinctive. However, datasets do not always contain characteristic features. In this project, the values of each example are quite similar because all examples belong to one species. In order to seek out the features that can tell the difference among examples, the following list is feature extraction methods used in this project.

- Bag-of-Words
- N-gram and K-mers
- Longest Common Subsequence
- Alphabetical Counter
- Length-based

Bag-of-Words is a popular approach in feature extraction. It is typically used for text classification. Bag of words simply breaks apart the words in the review text into individual word count statistics. It is a simplified representation used in natural language processing and information retrieval. The main idea of this approach is to define each word as a feature, afterward the frequency of each word in each example is calculated, For example, in this project, suppose that the SSR sequence is "addaqsss". The alphabet "a", "d", "q", and "s" are features with frequency 2, 2, 1, 3 respectively.

Another popular approach, especially in the area of computational genomic analysis, is N-gram. It is a type of probabilistic language model for predicting the next item in such a sequence in the form of an (n-1) order Markov model. Words are modeled such that each n-gram is composed of n words. N-grams are a natural extension of a bag of words and are simply any sequence of n tokens (words). We have tried three different variations in this approach:

1. Bi-gram approach: A bigram is an n-gram approach with a value of n = 2. It is a sequence of two consecutive characters found in a word or sequence. The possible output of bi-gram approach for the 'adsssaggqqqe' string would be {qq= 2, ss= 2, ds= 1, sa= 1, ad= 1, gg= 1, gq= 1, qe= 1, ag = 1}.

2. Tri-gram approach: Tri-gram is another variation in the ngram approach with having the value of n = 3. It search for three consecutive characters in a word or sequence and gives its frequency as the output. Using the same string as an example we will get {gqq= 1, ggq= 1, dss= 1 , ssa= 1, qqq= 1, ads= 1,sss= 1, sag= 1, agg= 1, qqe= 1} as the output.

3. Four-gram approach: Four-gram was performed by changing the value of n to 4. Taking the same example string 'adsssaggqqqe' we would have {qqqe= 1, aggq= 1, ggqq= 1, gqqq= 1, ssag= 1, adss= 1, sssa= 1 , dsss= 1, sagg= 1} as our output.

The core concept of N-gram illustrated in Fig.1 is to separate long strings into N-length substrings and take the substrings as features. For instance in Fig.1, "...GQAST..." is a fragment of genetic code in sample M. If N is three, every three continuous elements in genetic code would be signed as a feature, that

is “GQA”, “GAS” and “AST”. By following the above concepts, redo a similar process when N equal to other numbers (Note that N should greater than zero, and less than the total length of each given string).

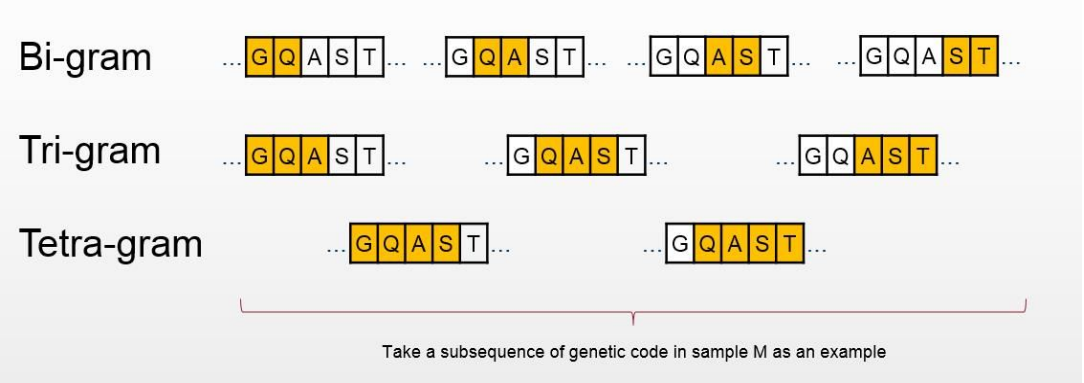


Fig.1 Example of N-gram

Another feature extraction approach that we followed was the K-mers approach. K-mers helps to find out all the possible substrings of length K that are contained in a string. By changing the value of K, we can find out the possible substrings in a text of length K. The possible output of Kmers approach for the string ‘adsssaggqqqe’ having the value of K = 2, would be {qq= 2, ss= 2, ds= 1, sa= 1, ad= 1, gg= 1, gq= 1, qe= 1, ag= 1}.

The next approach used in this project is to define the longest common subsequence as features. This method attempts to select the longest sequence that can be derived from another sequence by removing several elements without switching the order of remaining elements [4]. In addition, the dataset would be reordered alphabetically, and then the feature would be based on the first alphabet. Also, features can be extracted according to the length of each RNA sequence.

B. Feature Selection

When the number of features of a dataset is large (high-dimensional data), feature selection should be a required process to reduce the dimension of the data via removing the redundant and correlated features, or other methods such as Boruta. In feature extraction sections, certain methods generate redundant and correlated features. For example, in the K-mers with K equal to three, sequence “GQAST” and “GAAST” can generate features “GQA”, “QAS”, “AST” and “GAA”, “AAS”, “AST” respectively. In this situation, “AST” is a redundant feature (as well as a correlated feature since it correlated to another “AST”) carrying reduplicate information of the data. Thus, removing highly correlated and redundant features is significant since such features will increase overfitting when a classifier is trained on a dataset contains full of redundant features. Moreover, several features are contained in all examples. Removing such features will decrease the number of features without changing the result of classification. Once the number of features decreases, the cost of a training classification model can also be reduced. As a result, several zero- and near-zero variance features will be removed in this section.

The last method used for feature selection is Boruta. In Boruta, by comparing the importance among features using the random forest, certain irrelevant features will be eliminated from the data. It is an all relevant feature selection wrapper algorithm, capable of working with any classification method that output variable importance measure. It performs a top-down search for relevant features by comparing original attributes importance with importance achievable at random, estimated using their permuted copies, and progressively eliminating irrelevant features to stabilize that test.

C. Classification

Before training the classifier, we define labels using hierarchical clustering. In this process, in order to determine the class labels, we compute Levenshtein distance and generate a hierarchical clustering model of the sequences. After discussion meetings, Naïve Bayes and random forest are selected classification models for this project.

In the training process, certain classifiers like perceptron needs to be repeatedly trained in several iterations to improve its accuracy. However, Naïve Bayes is a speedy probabilistic based learning model that only needs to be trained once. The mechanics of Naïve Bayes is to calculate the probabilities of each category using Bayes rules. For example, in this project, suppose that a sequence is "GAQQSS". If a Naïve Bayes classifier tries to classify this example to a label "y_k" based on the feature "GAQQ", "AQQS", and "QQSS", the probabilities amount different labels will be computed as follows, where, in (1), y_k is the k-th label, α is evidence but it does not require to be calculated, n is the

$$probability(y_k) = \alpha \cdot P(y_k) \prod_{i=1}^{n} P(x_i \mid y_k),\ k \le n \tag{1}$$

the number of labels which is five in this case, and $P(x_i \mid y_k)$ is representing the probability of example containing feature x_i belongs to label y_k. According to the above formula, the probability of "GAQQSSSD" classified as label "y_k" shows in (2).

$$probability(y_k) = \alpha \cdot P(y_k)P("GAQQ" \mid y_k)P("AQQS" \mid y_k)P("QQSS" \mid y_k),\ k = 1, 2, 3, ..., n \tag{2}$$

Moreover, while the size of the dataset changes on a large scale, and the size of the training set is insufficient but contains enough features, the performance of Naïve Bayes is approximately remain stable. This is the reason why Naïve Bayes is selected in this project.

For random forest, it reduces the risks of overfitting and learns much faster than other classifiers. The core concept of the random forest is voted decision trees. Random forest, first of all, randomly creates a particular number of decision trees, collects results from all decision trees, and eventually chooses the merging result with high frequency as guessing value. It is worth noting that the random forest is able to balance error for imbalanced datasets. In the training process, the random forest can detect the influence among features, since inside the model, there are weight vectors for each class, and they are independent. Furthermore, the accuracy of random forest stays on a high level although there are missing values for certain features in the dataset. As a result, the random forest is considered as a classification model for this project.

IV. Analysis

To start with the implementation, we would like to discuss the dataset that we used for our project. Table 1 shows a few tuples of the dataset. The dataset contains amino acid sequences and each sequence is represented by a unique name. The dataset was of a pathogen called Anaplasma marginale. This pathogen commonly infects cattle. In some cases, base pairs are replaced by the amino acids. Short-sequence repeats (SSRs) mostly occurs in both prokaryotic and eukaryotic DNA. These sequences or SSRs were extracted from different sources and were not in a tidy manner.

Extracting features from the repeats of the SSRs was a challenging task as we needed to find meaningful features from the data set which was all characters. In order to do so, we used different feature extraction approaches as discussed above.

At the initial step, we studied different approaches that can be helpful us to extract features from the dataset. We started by applying the Bag-of approach on the data set and counted the number of times each character repeated in the SSR sequences. We made twenty bags where we found that the most repeated characters were “d”, ”s”, ”a”, ”g”, ”q”, ”e”, ”v”, ”t”, ”l”. And there were eleven characters that were rarely repeated in the sequences. Figure 2 shows the output of the bag-of approach.

Moving further, we tried the Ngram approach. Ngram helps to find out n consecutive characters in a word or string. We used three different variations by changing the value of n to 2, 3, and 4 respectively. In the bi-gram approach we took the value of n as 2 and were able to find 154 features. The most commonly used bigram was “ss”. By changing the value of n to 3 we got trigrams of the sequence. We find 370 features using trigram and the most common trigram was “ssq”. Finally, we applied four-gram using n = 4 and searched for 4 consecutive characters matching in the strings and find 581 features and the most common feature in four grams was repeated only one time. N-grams can help us get meaningful information in biological contexts such as they can represent the sequences of amino acids or nucleotides[1]. For instance, the sequence “AAACG” its unigram are A,A,A,C,G. The 2-grams are AA, AA, AC, CG. Similarly, 3-grams are AAA, AAG, and ACG. Figures 3, 4 and 5 show the features extracted from the bi-gram, tri-gram, and 4-gram.

	Name	Sequence
1	A	ddsssasgqqqessvssqseastssqlg
2	B	adsssaggqqqessvssqsdqastssqlg
3	C	adsssaggqqqessvssqsgqastssqlg

Table 1

Name	Sequence	d	s	a	g	q	e	v	t
A	ddsssasgqqqessvssqseastssqlg	2	12	2	2	5	2	1	1
B	adsssaggqqqessvssqsdqastssqlg	2	11	3	3	6	1	1	1
C	adsssaggqqqessvssqsgqastssqlg	1	11	3	4	6	1	1	1
D	adsssasgqqqessvssqseastssqlgg	1	12	3	3	5	2	1	1
E	adsssasgqqqessvssqseastssqlg	1	12	3	2	5	2	1	1

Figure 2. Bag of words

Comparing the results of the bag-of approach with the Ngram and Kmers approach find that the bag-of approach only provides the frequency counts of the characters it was not following any order. However, N-gram helps in parsing the sequences and describe how well the language model predicts a new text composed of unseen sentences.

Name	Sequence	ss	qq	sq	as	sg	gq
A	ddsssasgqqqessvssqseastssqlg	4	1	2	2	1	1
B	adsssaggqqqessvssqsdqastssqlg	4	1	2	1	0	1
C	adsssaggqqqessvssqsgqastssqlg	4	1	2	1	1	2
D	adsssasgqqqessvssqseastssqlgg	4	1	2	2	1	1
E	adsssasgqqqessvssqseastssqlg	4	1	2	2	1	1
F	tdsssasgqqqessvssqsgqastssqlg	4	1	2	2	2	2

Figure 3. Bi-gram approach

Name	Sequence	ssq	sgq	sss	ssa	tss
A	ddsssasgqqqessvssqseastssqlg	2	1	1	1	1
B	adsssaggqqqessvssqsdqastssqlg	2	0	1	1	1
C	adsssaggqqqessvssqsgqastssqlg	2	1	1	1	1
D	adsssasgqqqessvssqseastssqlgg	2	1	1	1	1
E	adsssasgqqqessvssqseastssqlg	2	1	1	1	1
F	tdsssasgqqqessvssqsgqastssqlg	2	2	1	1	1

Figure 4. Tri-gram approach

Name	Sequence	sssa	tssq	stss	asts	dsss
A	ddsssasgqqqessvssqseastssqlg	1	1	1	1	1
B	adsssaggqqqessvssqsdqastssqlg	1	1	1	1	1
C	adsssaggqqqessvssqsgqastssqlg	1	1	1	1	1
D	adsssasgqqqessvssqseastssqlgg	1	1	1	1	1
E	adsssasgqqqessvssqseastssqlg	1	1	1	1	1
F	tdsssasgqqqessvssqsgqastssqlg	1	1	1	1	1

Figure 5. 4-gram approach

Our main focus was to extract as many features as possible. In order to do that, we took another approach known as K-mers. It also helps to find K consecutive characters in a string and return its frequency. In K-mers we also used 3 different variations by picking k as 2, 3, and 4. The following figures show the result features of all the approaches.

Name	Sequence	ss	qq	sq	as	sg	gq
A	ddsssasgqqqessvssqseastssqlg	4	1	2	2	1	1
B	adsssaggqqqessvssqsdqastssqlg	4	1	2	1	0	1
C	adsssaggqqqessvssqsgqastssqlg	4	1	2	1	1	2
D	adsssasgqqqessvssqseastssqlgg	4	1	2	2	1	1
E	adsssasgqqqessvssqseastssqlg	4	1	2	2	1	1
F	tdsssasgqqqessvssqsgqastssqlg	4	1	2	2	2	2

Figure 6. 2-mers approach

Name	Sequence	ssq	sgq	sss	ssa	tss
A	ddsssasgqqqessvssqseastssqlg	2	1	1	1	1
B	adsssaggqqqessvssqsdqastssqlg	2	0	1	1	1
C	adsssaggqqqessvssqsgqastssqlg	2	1	1	1	1
D	adsssasgqqqessvssqseastssqlgg	2	1	1	1	1
E	adsssasgqqqessvssqseastssqlg	2	1	1	1	1
F	tdsssasgqqqessvssqsgqastssqlg	2	2	1	1	1

Figure 7. 3-mers approach

Name	Sequence	sssa	tssq	stss	asts	dsss
A	ddsssasgqqqessvssqseastssqlg	1	1	1	1	1
B	adsssaggqqqessvssqsdqastssqlg	1	1	1	1	1
C	adsssaggqqqessvssqsgqastssqlg	1	1	1	1	1
D	adsssasgqqqessvssqseastssqlgg	1	1	1	1	1
E	adsssasgqqqessvssqseastssqlg	1	1	1	1	1
F	tdsssasgqqqessvssqsgqastssqlg	1	1	1	1	1

Figure 8. 4-mers approach

One thing we find out in this approach is that by using Kmers approach we would get the same results as that of the N-gram approach. So, to avoid data duplication in our feature set we ignored the output of one of the approaches.

After checking for character repetitions, we also tried to find out some other features that may be useful for our final dataset. We gathered some more features by using three different measures. The first one was to count the total length of each sequence, which gave us an integer value between the ranges of 20 to 31. This means that the SSRs that we have in our dataset are neither smaller than 20 characters and nor longer than 31 characters. The other two approaches we carried out in order to find out some meaningful information were counting the first character of each sequence and counting each individual character or amino acid in the whole dataset of 234 sequences.

To achieve the best results, we need to gather more features. We took one more approach known as the Longest Common Subsequence (LCS). In this approach, each SSR is compared with another sequence and tried to find out the length of the longest subsequence and its quality similarity index[4]. Figure 8 shows the features extracted from the longest common subsequence method. We can clearly see in the figure that we compared two strings and find the longest subsequence possible from both the strings.

S1	S2	LCS	LengthLCS	QualitySimilarityIndex
ddsssasgqqqessvssqseastssqlg	ddsssasgqqqessvssqseastssqlg	ddsssasgqqqessvssqseastssqlg	28	1.0000000
ddsssasgqqqessvssqseastssqlg	adsssaggqqqessvssqsdqastssqlg	dsssagqqqessvssqsastssqlg	25	0.8620690
ddsssasgqqqessvssqseastssqlg	adsssaggqqqessvssqsgqastssqlg	dsssagqqqessvssqsastssqlg	25	0.8620690
ddsssasgqqqessvssqseastssqlg	adsssasgqqqessvssqseastssqlgg	dsssasgqqqessvssqseastssqlg	27	0.9310345
ddsssasgqqqessvssqseastssqlg	adsssasgqqqessvssqseastssqlg	dsssasgqqqessvssqseastssqlg	27	0.9642857
ddsssasgqqqessvssqseastssqlg	tdsssasgqqqessvssqsgqastssqlg	dsssasgqqqessvssqsastssqlg	26	0.8965517

Figure 9. Longest Common subsequence

By merging all the outputs from different approaches we took we were able to collect around 1128 features. After collecting features, now the next phase that we worked on was to collect meaningful features and reduce our feature set so that we can build a successful regression or classification model. In order to do so, we followed two different approaches. The first one was by applying caret package available in R using which we removed near-zero value feature sets and highly correlated values. These

two approaches helped us a lot to shrink our feature set. Removing zero-variance reduced the feature set from 1128 features to 184 features by removing all the features that had mostly 0 as an output, in other words, the feature set that had 80% or more 0 values were removed. By reducing the feature set we still had some highly correlated predictors that may deviate the final result. So, by finding highly correlated predictors we find out that we still have 122 highly correlated predictors and we removed those 122 predictors and were left with 62 predictors which means that we have removed a large number of irrelevant data.

Another approach that we took for feature selection was using the Boruta package. It is a wrapper algorithm and it reduces the dataset by identifying meaningful and not so important features. The function in the Boruta package is used to find features and its importance in the feature set. It performs a top-down search for relevant features by comparing original attributes importance with importance achievable at random and highlighting irrelevant features to get stable end results. Using the Boruta algorithm on the feature set we found various shadow attributes, tentative attributes, confirmed and unconfirmed attributes in the feature set.

The next step that we were trying to perform on our feature set was to apply the classification model and predict results for our testing set. But before we proceed with that we analyzed our feature set and came to know that we didn't have any class label from which we can use as a predictor for the feature set. We tried to make clusters of our SSRs using a hierarchical clustering model. We tried to find out distance metrics using different distance formulas, and we were successful in finding it by using the Levenshtein distance formula. Levenshtein is a distance finding formula it finds the distance by finding the similarity between two strings. The distance is the number of deletions, insertions, or substitutions required to transform the source word to target word. By applying the agglomerative hierarchical clustering algorithm on the Levenshtein distances and we get the dendrograms and its split in a cluster can help us to label the data from 1 to 6 as shown in figure 10.

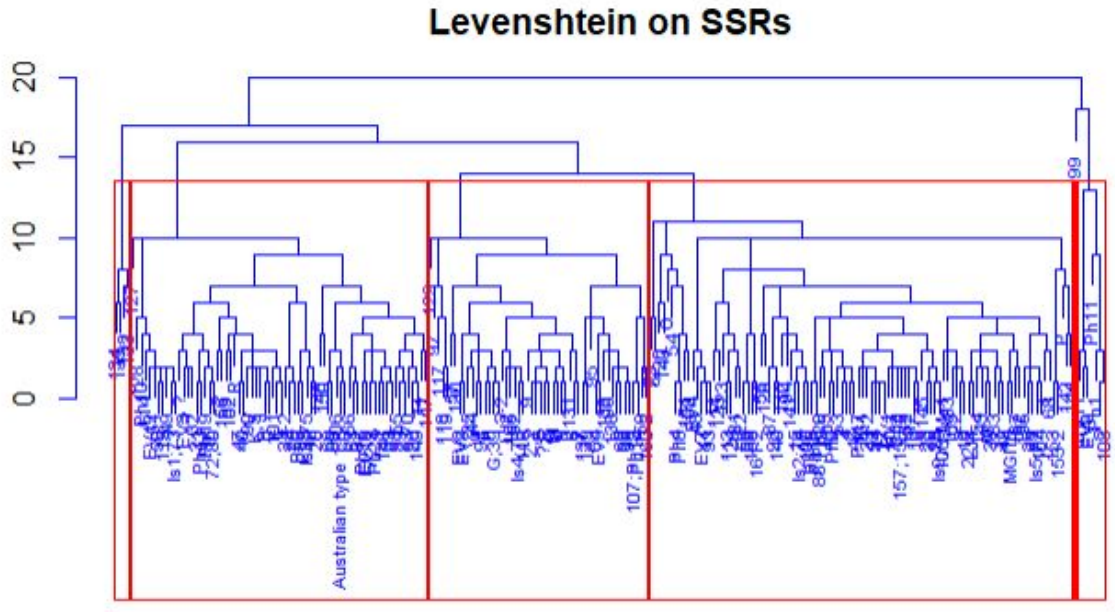


Figure 10. Hierarchical Clustering

After creating the class, we randomly divided the feature set into training and testing set to use for classification models by using 75:25 rule.

We used two different kinds of classification models to predict results and compare those results. The first approach we took was the Naïve Bayes classification. Naïve Bayes is a method used to construct classifier models that can assign class labels to problem instances. The next classification approach that we followed was using Random Forest. It is a learning method for classification, regression and other tasks that operates by constructing a variety of decision trees at training time and giving output as the class that is the mode of the classes of the individual trees.

V. Results and Discussion

Based on our implementation and measures we took to perform operations on our dataset, the results generated are as follows:

We extracted features using different approaches. On comparison of the results of the bag-of approach with the N-gram and K-mers approach, we found out that the bag-of approach only provides the frequency counts of the characters it was not following any order. As we can clearly see in Figure 2 that the bag of approach returns only the count of each character in the sequences. For example, in the first. sequence character s is repeated 12 times and character d is repeated 2 times and so on. Whereas, n-gram and Kmers try to find the pattern of characters following order. In other words, it works by matching characters in sequence based on the window of K or n [2].

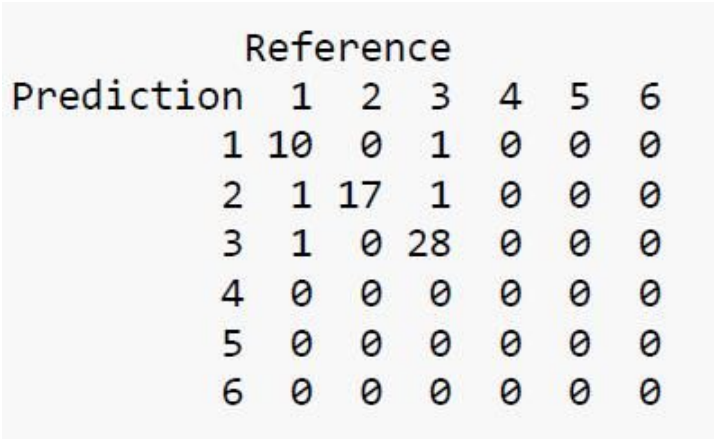
From the features that we extracted using n-gram and K-mers approaches, we found out that by using Kmers approach we would get the same results as that of the n-gram approach. So, rather than having the same features from different approaches we included only the output of the n-gram approach so that we can avoid data duplication.

Another feature extraction method we used was finding the longest common subsequence. Finding LCS is a time-consuming process and the complexity of this approach is quite high. In order to avoid the time complexity of our program, we decided not to include LCS in our final feature set.

From the feature extraction process, we were able to extract more than 1000 features, which contained irrelevant features, which may impact the final results. In order to get meaningful and useful features, we removed near-zero variance predictors as well as highly correlated predictors. The reason behind using these functions was to extract useful features and get rid of untidy data.

Our dataset does not contain any class labels from which the data can be distinguishable or classified. In order to have some class labels for the dataset, we used Levenshtein distance metrics which helped us to make clusters of the data sequences. Based on those clusters we grouped the data as shown in figure 10. The dendrograms split into clusters can help us to label our data.

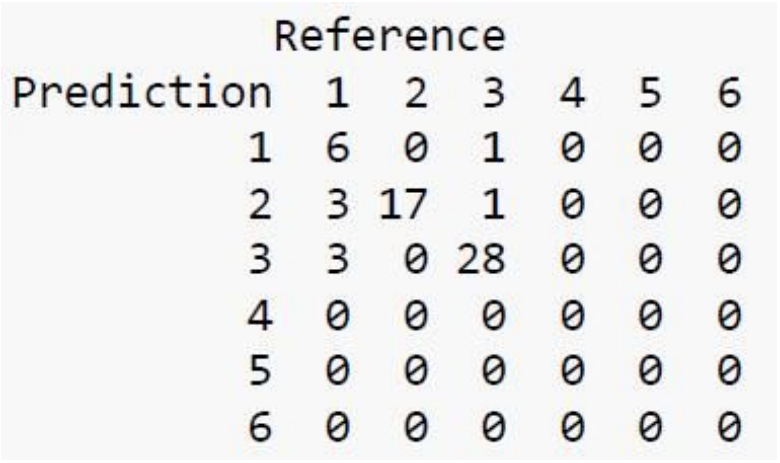
After reducing the feature set, we split the data into training and testing based on a 75:25 ratio and applied the Naïve Bayes classification model on the feature set to classify the models. Figure.11 shows the confusion matrix of the Naïve Bayes Classifier. According to Figure.11 shows below, the values in class 4,



	Reference					
Prediction	1	2	3	4	5	6
1	10	0	1	0	0	0
2	1	17	1	0	0	0
3	1	0	28	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure.11 Confusion matrix of Naive Bayes

5 and 6 are all zero. The possible reason is that as the label defined by hierarchical clustering shown in Figure 10, examples are mainly signed as classes 1, 2 and 3, while a small number of examples are signed as classes 4, 5 and 6. Thus, it is possible that the training data contains all examples regarding classes 4, 5 and 6, while testing data does not contain the example belonging to those classes. At this point, although the confusion matrix illustrates that there are all zero in class 4, 5 and 6, the trained classifier still is a universal model to handle this classification problem. The same situation happens on the confusion matrix of random forest exhibiting in Figure.12 below.



	Reference					
Prediction	1	2	3	4	5	6
1	6	0	1	0	0	0
2	3	17	1	0	0	0
3	3	0	28	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure.12 Confusion matrix of random forest

To compare our model against Naïve Bayes model we applied a feature wrapping algorithm, Boruta, and segregated the features into shadow attributes, tentative attributes, confirmed and unconfirmed attributes in the feature set as shown in figure 13. Whereas, figure 14 represent the density of the segregated features obtained from the Boruta algorithm.

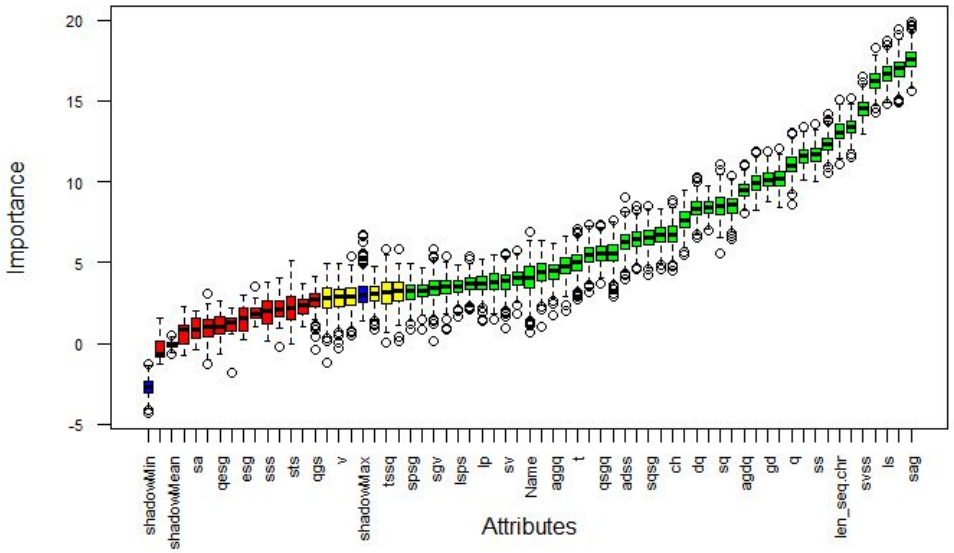


Figure 13. Boruta plot

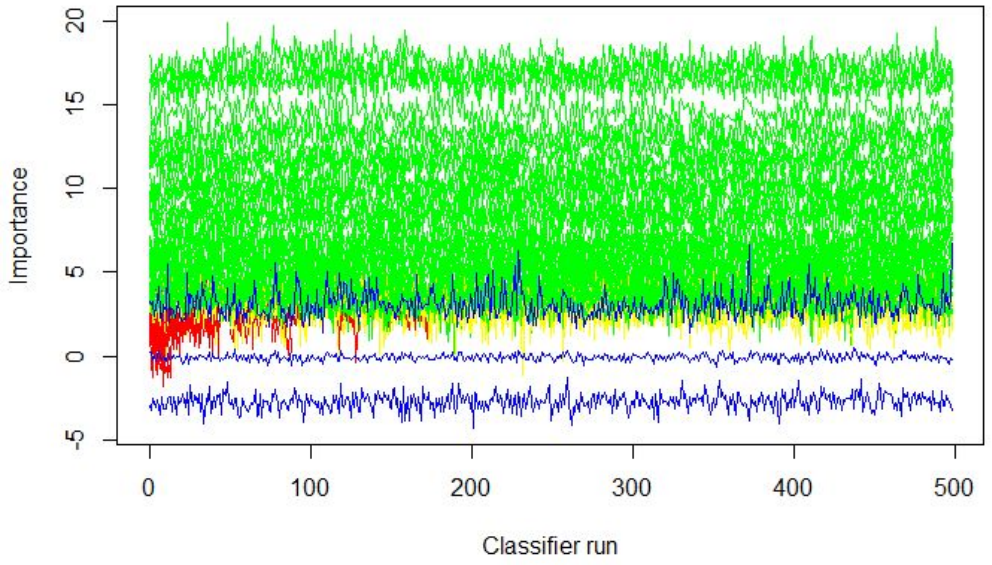


Figure 14. The density of the feature set

Surprising is that the accuracy of Naive Bayes and random forest are approximately 0.9233 and 0.8644 respectively. Basically, the accuracy of Naive Bayes on most kinds of classification problems is around 70%. In this project, it reaches 90%. Perhaps, such high accuracy on the Naive Bayes model is because all features almost meet one of the important requirements of Naive Bayes — conditional independent. In most classification problems, the correlation values among features are not zero, they cannot be filtered in the feature selection process is because their correlation values are low enough, but still not zero. This is also perfectly interpreting the reason why the accuracy of Naive Bayes in most cases is around 70%. Furthermore, in Figure13, because of the imbalanced splitting dataset, several NA's can be seen. Figure 15

	Class: 1	Class: 2	Class: 3	Class: 4	Class: 5	Class: 6
Sensitivity	0.5000	1.0000	0.9333	NA	NA	NA
Specificity	0.9787	0.9048	0.8966	1	1	1
Pos Pred Value	0.8571	0.8095	0.9032	NA	NA	NA
Neg Pred Value	0.8846	1.0000	0.9286	NA	NA	NA
Prevalence	0.2034	0.2881	0.5085	0	0	0
Detection Rate	0.1017	0.2881	0.4746	0	0	0
Detection Prevalence	0.1186	0.3559	0.5254	0	0	0
Balanced Accuracy	0.7394	0.9524	0.9149	NA	NA	NA

Figure.15 Statistics of classes in random forest model

also informs that sensitivity and accuracy of class 1 in random forest are 0.5 and 0.74 respectively, which indicates that random forest is struggling with classifying the class 1, while it performs an excellent result on other two classes. This observation is explaining why the overall performance of random forest is slightly worse than Naive Bayes.

V. Conclusion

In summary, we started with the data set of sequences of SSR's of Anaplasma Marginale and have extracted various features. In order to select meaningful features from the extracted feature set by applying various feature extraction methods like removing zeros from the feature set and removing highly correlated values. We calculated the distance of the sequences which were useful in classifying the data. A feature wrapping technique was used to find the meaningful features. Finally, we checked our features by applying them into two classification models Naïve Bayes and Random forest and found their confusion metrics. In the future, we can use these methods to extract various meaningful features from DNA sequences of patients suffering from different kinds of diseases to extract meaningful patterns from their DNAs to know the causes for such diseases also we can foresee a great application of such feature extraction methods in the field of Bioinformatics.

References

[1] "Symbiosis: The Art of Living Together".(2019). *National Geographic*. [Online]. Available: https://www.nationalgeographic.org/article/symbiosis-art-living-together/3rd-grade/

[2] Dhar, V. (2013). "Data science and prediction". Communications of the ACM. vol.56 pp. 64–73. doi:10.1145/2500499

[3] W. Liang and Z. KaiYong, "Segmenting DNA sequence into 'words,'" p. 12.

[4] A. Apostolico and C. Guerra, "The longest common subsequence problem revisited," *Algorithmica*, vol. 2, no. 1–4, pp. 315–336, Nov. 1987.

[5] M. Ganapathiraju, D. Weisser, R. Rosenfeld, J. Carbonell, R. Reddy, and J. Klein-Seetharaman, "Comparative n-gram analysis of whole-genome protein sequences," in *Proceedings of the second international conference on Human Language Technology Research* -, San Diego, California, 2002, p. 76.

[6] X. Ma and L. Hu, "Extracting Sequence Features to Predict DNA-Binding Proteins Using Support Vector Machine," in *2013 International Conference on Computational and Information Sciences*,

Shiyang, China, 2013, pp. 152–155.

Link to the code

R project code link