2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

A Computational Approach to Uncertainty in DNA Sequences

Melissa N. Melaugh[1], and Prof. Sonya Coleman[1] and Dr. Dermot Kerr[1]

***Abstract*— DNA sequencing is the process of reading individual base pairs from a section of DNA. Genes are the name given to parts of the DNA which encode proteins; for example ion channels are proteins that maintain concentrations of ions within cells. The sequencing of these genes can offer insights into factors such as evolution and disease. During the sequencing process, unknown values 'N' can be substituted in the sequence where the sequencing machine is unable to identify a nucleotide as Adenine (A), Cytosine (C), Thymine (T), or Guanine (G). These gene sequences vary in length; this includes individual genes across the same species. This has led to the use of a process known as k-mer encoding so that a machine learning algorithm can assess these genes without the need for pre-alignment. K-mer encoding works by taking small sections of the sequence and tallying the number of times that such a sequence appears, such as, how many times the k-mer 'ACCT' appears in the overall sequence. The unknown 'N' value presents a problem in k-mer encoding, as this value increases the size of the k-mer feature vector exponentially as the k-mer length increases. In this paper we research the accuracy and computational impact of including, removing, or ignoring this 'N' value for the k-mer lengths 3, 6, and 9 across four Machine Learning algorithms: Random Forest, Multinomial Naive Bayes, Neural Networks, and Linear Support Vector Machine.**

I. INTRODUCTION

Every gene is a section of a bigger strand of DNA; a gene, like all DNA, is composed of the four nucleotides Adenine (A), Cytosine (C), Thymine (T), and Guanine (G). These genes can be read through DNA sequencing, a process in which the particular gene is amplified and sequenced. Two popular sequencing techniques are Nanopore sequencing and Single Molecule Real Time (SMRT) sequencing. Each method will produce an output of the amplified gene saved to a file, often referred to as FASTA or FASTQ files. FASTA files originated from the FASTA program, and in the case of DNA sequencing, they return sequences under manually input headers. FASTQ files have superseded these, adding a quality aspect, which encodes the accuracy of the read at that nucleotide. The nucleotide sequences are often used to identify species, such as in 16S and 18S sequencing for bacteria and fungal communities.

Typically, nucleotides are copied into mRNA before being processed and read into non-overlapping sets of 3, in order to create the associated protein. In this way a gene sequence can be considered as a sentence describing a protein, and these sets of three nucleotides can be thought of as words, hereby referred to as k-mers.

When a gene is sequenced, there can be noise in the data. This can be due to a number of factors including single site mutations and machine error. Where no clear reading can be obtained from the nucleotide position (A, C, T, or G), the reading is noted as an unknown 'N' value.

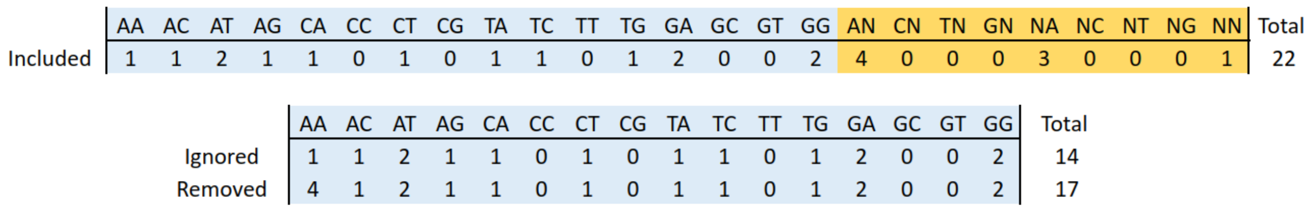
Mohamed et. al used an 'N included' approach for classification of gene groups, in which all nucleotides that were not A, C, T, or G were included as a 'Z' character [1]. Juneja et. al tested a dataset with K2, K3, K4, K5, and K6 using only a MNB classifier [2]. They split their DNA sequences into the desired k-mer length and used these values to construct their final feature vector [2]. This aligns with an 'N included' approach but also reduces the overall feature vector as k-mers that are not present in the training sequences are not included. Solis-Reyes takes an 'N removed' approach in their open source tool called KAMRIS, in which 15 classifiers were tested for k-mers classification; they found that the Linear SVM had the best balance for accuracy/timing tests for the K6 values, using the 'N removed' method [3]. Other methods tested were: Cubic SVM, Quadratic SVM, Linear SVM, NN, Logistic Regression, KNN using a k of 10, Nearest Centroid Median, Nearest Centroid Mean, Decision Tree, Random Forest, SGD, Guassian Naive Bayes, LDA, QDA, and Adaboost [3]. Uddin et. al presented a matrix style k-mer encoding for phylogenetic tree reconstruction, in which k-mer dictionaries were explicitly created. In their work, they did not mention these unknown 'N' values nor any pre-processing of the DNA data, prior to input into their pipeline [4]. If they did not remove the 'N' values, it would align with an 'N ignored' approach.

Often in similar research, it is unclear how the 'N' value is managed in the k-mer approach. Blaisdell, one of the original authors of k-mer encoding, did not acknowledge their approach to the 'N' value [5], nor did Burge, Campbell, and Karlin with their approach to k-mer distance metrics [6]. Asgari et. al used k-mers on 16S data to identify the location of bacteria on a patient's body, but were also not explicit in their approach to the 'N' values [7]. Ng also did not discuss handling the unknown 'N' in their development of the k-mer tool dna2vec [**?**].

We have identified three strategies employed when dealing with the 'N' values in sequences: 'N included', 'N removed', or 'N ignored', each with its own strengths and weaknesses. The aim of this paper is to determine which methods of k-mer counting provide the best classification accuracy and testing time when using various machine learning methods.

*This work was supported by the Department of Agriculture, Environment and Rural Affairs

[1]Melissa N. Melaugh, Prof. Sonya Coleman, and Dr. Dermot Kerr are with Faculty of Computing, Engineering and the Built Environment, Ulster University, Northern Ireland, UK melaugh-m4@ulster.ac.uk sa.coleman@ulster.ac.uk d.kerr@ulster.ac.uk



	AA	AC	AT	AG	CA	CC	CT	CG	TA	TC	TT	TG	GA	GC	GT	GG	AN	CN	TN	GN	NA	NC	NT	NG	NN	Total
Included	1	1	2	1	1	0	1	0	1	1	0	1	2	0	0	2	4	0	0	0	3	0	0	0	1	22

	AA	AC	AT	AG	CA	CC	CT	CG	TA	TC	TT	TG	GA	GC	GT	GG	Total
Ignored	1	1	2	1	1	0	1	0	1	1	0	1	2	0	0	2	14
Removed	4	1	2	1	1	0	1	0	1	1	0	1	2	0	0	2	17

Fig. 1. Top: The dictionary and corresponding feature vector for the 'N included'. The goldenrod colour has been used to highlight the additional dictionary and features as compared to 'N ignored' and 'N removed'. Bottom: The dictionary and corresponding feature vector for the 'N ignored' and 'N removed'. In all three cases, the total number of k-mers counted has been included.

II. DATASET BACKGROUND AND DESCRIPTION

We use Chauban's DNA Sequence Dataset which consists of seven gene families isolated from three species: Human, Chimpanzee, Dog. These gene families are: G-Protein Coupled Receptors, Tyrosine Kinase, Tyrosine Phosphatase, Synthetase, Synthase, Ion Channels and Transcription Factors [9]. Of the 6,882 sequences in Chauban's DNA Sequence Dataset, 379 contain unknowns. Of the 10,452,663 bases these genes contain, 925 were an 'N' value, giving an overall frequency of less than 0.00009%. Although there is a low 'N' frequency, it is important to determine if classification would be more accurate and/or efficient if the 'N' values were included, removed, or ignored. The difference between these options is best conveyed in an example, using an overlapping k-mer length of 2 (K2), to avoid becoming too large for a manual example.

When counting the k-mers, a dictionary of all possible nucleotide combinations must be created. Its length can be described as

$$L = B^K \tag{1}$$

where L is length, B is the number of bases (4 for 'N ignored'/'N removed', and 5 for 'N included'), and K is the k-mer length. For K1, the dictionary length is 5 (A,C,T,G,N) in the case of 'N included', and 4 (A,C,T,G) in the cases of 'N removed' and 'N ignored'. The feature vector then becomes a count of the occurrences of each of the values in the dictionary. This feature counting can be referred to as a Count Vectoriser, which takes in a dictionary and sequence and determines the corresponding counts from the sequence. The K2 feature length for 'N included' is 25, while the K2 feature length for both 'N ignored' and 'N removed' is 16. This is because there are 25 possible combinations of A, C, T, G and N, while there are 16 possible combinations of A, C, T, and G. The two dictionaries can be seen above the resulting feature vectors in Figure 1.

We explain this further using the following 24 nucleotide DNA sequence: AACTANNATCANATG-GANAGGGAN.The 'N included' k-mer dictionary is composed of all possible combinations of the 4 nucleotides and 'N' at the k-mer length 2, resulting in a feature length of 25. The feature vector is a count of these k-mers within the DNA sequence as shown in Figure 1. This example sequence results in a total of 22 k-mers. Unlike the other two methods, this feature vector includes counts for AN (4), CN(0), TN(0), NA (3), NC(0), NT (0), NG(0), and NN (1), which are not present in the other vectors; this results in a larger feature length.

In the case of 'N removed', the 'N' values are deleted and a new link created between the nucleotides on either side of the removed nucleotide. The DNA sequence with 'N' removed becomes: AACTAATCAATGGAAGGGA (19 nucleotides in length) resulting in 17 k-mers in the feature vector. Notice in Figure 1 that the k-mer pair AA has increased from 1 in 'N included' to 4 with 'N removed'; this is where TANNAT became TAAT, CANAT became CAAT, and GANAG became GAAG after the deletion of the 'N' value. It has also shortened the length of the overall feature vector from 25 to 16 as there are only 16 possible combinations of A, C, T, and G.

'N ignored' essentially breaks up the sequence into slices: AACTA, ATCA, ATGGA, AGGGA, because any k-mer with an 'N' is not counted in the feature vector. The K2 k-mer count results in a total of 14, the smallest count. Where 'N removed' joined the TANNAT to become TAAT resulted in the length 2 k-mers TA, AA, AT, 'N ignored' split it into TA, and AT, resulted in the A nucleotides being counted only once, rather than twice in this set. As the length of the k-mers increases, the problem compounds. If the k-mer length was 6, the feature vector would have no k-mers due to the way this sequence is split, and if it were 5 there would only be 3 features included, due to the nature of 'N ignored'. While it is unlikely that the 'N' value would be this frequent in a study, it can still create anomalies since nucleotides on either side of this 'N' value are disconnected.

This demonstrates how these three strategies ('N included', 'N removed', or 'N ignored') result in different feature lengths and information encodings. This paper explores how each of these strategies performs with various machine learning methods to determine the best overall strategy for handling 'N'.

III. METHODOLOGY

Figure 2 illustrates how the data moves through the pipeline. Dictionaries are created for all possible k-mer combinations at lengths 3 (K3), 6 (K6), and 9 (K9) for 'N included' (A,C,T,G,N), 'N removed' (A,C,T,G), and 'N ignored' (A,C,T,G) to be used with their associated pipeline.

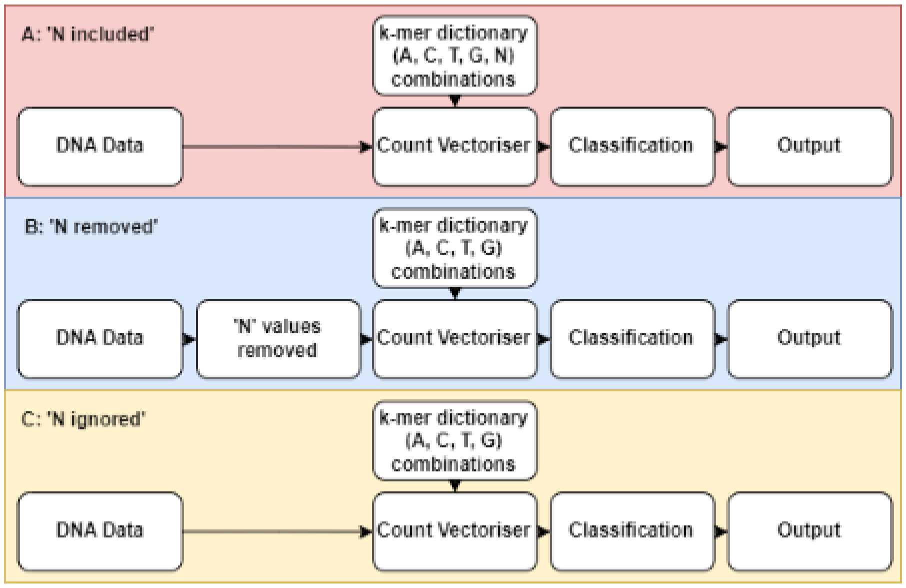


Fig. 2. A. The 'N included' pipeline. B. The 'N removed' pipeline. C. The 'N ignored' pipeline.

Note that the 'N removed' and 'N ignored' associated dictionaries are the same. The Random Forest Classifier, MNB Classifier, NN Classifier, and SVM classifier are used in the 'Classification' step of their associated pipeline.

Figure 2A depicts the 'N included' method; it uses the dictionaries composed of A, C, T, G, and N. The DNA data moves directly into the vectoriser without being altered and is classified in one of the ML methods to produce an output. Figure 2B depicts the 'N removed' method; it uses the dictionaries composed of A, C, T, and G. The DNA data has all 'N' values removed before it moves into the vectoriser. This feature vector is classified in one of the ML methods to produce the output. Figure 2C depicts the 'N ignored' method; it uses the dictionaries composed of A, C, T, and G. The DNA data moves directly into the vectoriser without being altered and is classified in one of the ML methods to produce an output.

A. Feature Selection

The lengths of the k-mer sequences chosen were 3, 6, and 9, hereby referred to as K3, K6, and K9. These values were chosen on the basis that the nucleotides are read in non-overlapping sets of three, to create proteins. It should be noted that unlike protein creation, the Count Vectoriser uses overlapping k-mer sets, in order to give more contextual information to the machine learning algorithm. These matrices are all the same length and k-mer order, which allows the samples to be used in a classifier [10][**?**].

The dictionary for k-mers was created with all possible k-mer combinations using either A, C, T, and G or A, C, T, G, and N at the desired k-mer length and provided to the Count Vectoriser. The vectoriser uses this dictionary to create the counts for the feature vector.

Using Equation 1, the calculated K1 feature length is 5 for 'N included' and 4 for 'N removed'/'N ignored', representing the 4 nucleotides, and the unknown 'N' in the case of 'N included'. At K9, 'N included' has a feature length of 1,953,125, which is over seven times larger than the 'N removed'/'N ignored' feature vector which has only 262,144 features.

B. Classifiers

The Random Forest Classifier builds upon the Decision Tree Classifier. It uses an ensemble method in which a variable number of Decision Trees are trained; a weighted average is used to determine the classifier's final output. Decision Trees are a method of mapping data into a tree like structure, in which branches represent aspects of the data, called decision rules, making them relatively easy to interpret. Decision Trees start at a root node, each node is then split according to some function; this split is exhausted for every new node until the tree can terminate with a resulting classification. The chosen function for the splitting was the Gini impurity, since the data are categorical as opposed to continuous. The Gini impurity is defined as

$$G(p) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2}$$

where p is the probability the data are from class i [11]. For each node, the split is found by calculating the weighted Gini impurity of both child nodes, as it can only perform binary splits. The goal is to create branches until the Gini impurity is 0, which means all remaining features belong to one class (the output). The Decision Tree will iterate until all branches have reached a Gini impurity 0, or until artificially stopped [12]. The Random Forest Classifier used was initialised with 100 Decision Trees without depth limits and used the Gini impurity for learning [10].

The Multinomial Naive Bayes (MNB) classifier is traditionally used for Natural Language Processing. As DNA can

TABLE I
DATA DISTRIBUTION FOR TRAINING AND TESTING GROUPS IN THE HUMAN, CHIMP, AND DOG, AS WELL AS THE TOTALS FOR EACH GROUP.

	Human			Chimp			Dog			TOTALS		
	Total	Train	Test	**Total**	Train	Test	**Total**	Train	Test	**Total**	Train	Test
G Protein Coupled Receptors	**531**	477	54	**234**	210	24	**131**	117	14	**896**	804	92
Tyrosine Kinase	**534**	480	54	**185**	166	19	**75**	67	8	**794**	713	81
Tyrosine Phosphatase	**349**	314	35	**144**	129	15	**64**	57	7	**557**	500	57
Synthetase	**672**	604	68	**228**	205	23	**95**	85	10	**995**	894	101
Synthase	**711**	639	72	**261**	234	27	**135**	121	14	**1107**	994	113
Ion Channel	**240**	216	24	**109**	98	11	**60**	54	6	**409**	368	41
Transcription Factor	**1343**	1208	135	**521**	468	53	**260**	234	26	**2124**	1910	214
TOTALS	**4380**	3938	442	**1682**	1510	172	**820**	735	85	**6882**	6183	699

be thought of as a series of words, it was included for testing. MNB is primarily based on Bayes' Theorem which describes the probability of an event based on prior information as shown in Equation 3.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{3}$$

The Naive Bayes classifier is considered 'naive' as it makes assumptions that each feature is independent and of equal importance. Multinomial refers to there being a discrete number of inputs and outputs [13]. In this case, the inputs will all be of the same length due to the Count Vectoriser, and the outputs will be one of the seven genes. The MNB Classifier was initialised with an alpha value of 1 [10].

SVM are binary classifiers, having a single hyperplane to separate data. Despite this, the SVM can be used with multiple classes. The most common method is the One Vs Rest (OvR), and is default in the Linear SVM from Scikit Learn [10]. The OvR approach iterates through each class which becomes the positive label, while all other classes become the negative label. Each iteration sets a hyperplane between the positive and negative. The end result is the same number of hyperplanes as classes, allowing for multi-label classification [14]. A linear SVM using a square hinge-loss can be mathematically represented by Equation 4 [10] [14]. Here the training input $x_i \in \mathbb{R}^p$, a vector $y \in 1, -1^n$, and the equation is solved for $w \in \mathbb{R}^p$ and $b\mathbb{R}$ so that the prediction is correct for most samples. The regularisation value, C, controls the degree to which an SVM is hard (large C) or soft (small C) [10] [14].

$$\min \frac{1}{2} w^T w + C \sum_{i=0}^{n} \left(max(0, 1 - y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i - b_i)^2) \right) \tag{4}$$

After testing the RBF, polynomial, sigmoid and linear kernels, it was determined that the linear kernel was the best performing. We also implemented a squared hinge loss function defined in Equation 4, with a C of 0.5; all other values were analogous with the default from the Linear SVM class in Scikit Learn [10].

A Neural Network (NN) is able to use a series of hidden layers in order to take an input sequence and get a corresponding output. Due to memory limits, the K9 feature vector was too large for processing across all of the NN tests. The NN architecture used in this section was crafted using the PyTorch package [15]. Most layers consist of a linear portion, z, represented in Equation 5, followed by an activation function a. A Rectified Linear Unit (ReLU) is one such activation function, and is represented in Equation 6. ReLU is often used for the hidden layers, while a Softmax activation function, Equation 7, is often used for an output layer. Both of these activation functions creates non-linearity within the network, allowing it to learn through backpropagation.

$$z = \sum_i w_i^T * x_i + b \tag{5}$$

where w represents the weights, x the input vector, and b the associated bias for this node [15].

$$f(z_j) = max(0, z_j) \tag{6}$$

where z_j is the output of Equation 5 at a given neuron [15].

$$f(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{7}$$

where z_j is the output of Equation 5 at a given neuron [15]. The NN utilised consists of 3 Linear/ReLU layer combinations of 512, 128, and 32 neurons, followed by a final Linear/Softmax combination composed of the 7 output neurons.

IV. RESULTS

The four classifiers are presented with the three strategies to 'N' across the K3, K6, and K9 tests, where possible. All tests were performed on a Dell Precision 3551 on Linux Ubuntu 22.04.1 LTS using Python 3.9.13 with the Anaconda 3, Scikit Learn v1.2.2 [10], and PyTorch packages [15].

Table I shows the breakdown of the training and testing data. Training and testing groups were created per species per gene. 10% of each gene from each species was held for testing, while the other 90% was used for training the classifiers.

The results for each of the strategies used with the Random Forest Classifier are displayed in Table II. The K3 test is most accurate with 'N ignored' at 97.6%; however the 'N included' and 'N removed' tests are both faster at 0.24 seconds. At K6, the 'N removed' test ties for highest accuracy with 'N included', but is faster at 0.42 seconds.

TABLE II

ACCURACY AND TESTING TIME (IN SECONDS) FOR THE RANDOM FOREST CLASSIFIER.

Random Forest		**K3**	**K6**	**K9**
'N included'	Accuracy	0.8641	0.9356	0.9528
	Test Time	0.24	0.45	1.23
'N removed'	Accuracy	0.8698	0.9356	0.9557
	Test Time	0.24	0.42	1.01
'N ignored'	Accuracy	0.8755	0.9327	0.9456
	Test Time	0.28	0.51	1.28

The K9 'N removed' outperforms the 'N included' and 'N ignored' tests with a 95.6% accuracy, and a 1.01 test time.

The results for each of the strategies used with the MNB classifier are displayed in Table III. Accuracies for K3 and K6 were less than 65% at best. In both cases the 'N removed' tests were at least 0.1 seconds faster than the other two tests, however the 'N included' tests were more accurate. In the K9 test, the highest accuracy was from the 'N removed' and 'N ignored' tests, at 99.1%. The K9 'N removed' test was 0.1 seconds faster than the K9 'N included' and K9 'N ignored' tests.

TABLE III

ACCURACY AND TESTING TIME (IN SECONDS) FOR THE MNB CLASSIFIER.

MNB		**K3**	**K6**	**K9**
'N included'	Accuracy	0.2833	0.6209	0.9585
	Test Time	0.23	0.35	0.72
'N removed'	Accuracy	0.2804	0.6052	0.9914
	Test Time	0.22	0.33	0.61
'N ignored'	Accuracy	0.2804	0.6051	0.9914
	Test Time	0.26	0.40	0.72

The results for the NN classifier are displayed in Table IV. Due to memory limitations, only the K3 and K6 tests were able to be run. The K3 test performed best with 'N included' in both accuracy and testing speed. It performed at 80.5% accuracy, classifying 699 genes in 0.24 seconds. This is over 5% more accurate than both the 'N removed' and 'N ignored' tests, as well as 0.01 seconds faster. In the K6 tests 'N included' was the slowest and least accurate at 89.4% in 0.49 seconds. The 'N removed' test was 0.1 seconds faster, and resulted in an accuracy of 94.6%.

TABLE IV

ACCURACY AND TESTING TIME (IN SECONDS) FOR THE NN CLASSIFIER.

NN		**K3**	**K6**
'N included'	Accuracy	0.8054	0.8941
	Test Time	0.25	0.49
'N removed'	Accuracy	0.7339	0.9456
	Test Time	0.26	0.39
'N ignored'	Accuracy	0.7510	0.9427
	Test Time	0.26	0.47

The results for the Linear SVM classifier are displayed in Table V. At K3, the 'N removed' performs the fastest at 0.22 seconds, while 'N ignored' produces the highest accuracy at 57.9%. The K6 and K9 tests also have 'N removed' performing the fastest at 0.35 seconds and 0.62 seconds respectively. In the K6 and K9 tests, accuracies were approximately the same across the 'N' tests at approximately 93.8% and 97.3% respectively.

TABLE V

ACCURACY AND TESTING TIME (IN SECONDS) FOR THE LINEAR SVM CLASSIFIER.

SVM		**K3**	**K6**	**K9**
'N included'	Accuracy	0.5522	0.9385	0.9728
	Test Time	0.24	0.36	0.73
'N removed'	Accuracy	0.5651	0.9385	0.9728
	Test Time	0.22	0.35	0.62
'N ignored'	Accuracy	0.5793	0.9384	0.9728
	Test Time	0.26	0.39	0.72

V. CONCLUSION

This research suggests that there is a balance between the 'N' approach taken and the k-mer length. A small k-mer length (like K3) benefits from the extra information that the 'N included' feature vector provides, while a larger k-mer length (like K9) benefits from the compact feature vector offered from 'N removed'. Recall that the length of the feature vector becomes exponentially larger as the k-mer length increases, which can be calculated using Equation 1.

At K3, most of the tests had similar accuracies and speeds within the respective classifiers; the NN classifier was the exception with the 'N included' test outperforming the other two tests in both accuracy (80.54% versus 'N removed' 73.39% and 'N ignored' 75.10%) and speed (0.25 seconds versus 0.26 seconds in both 'N removed' and 'N ignored'). However, the K3 test's best accuracy was from the Random Forest's 'N ignored' test at 87.6% taking 0.28 seconds to run.

At K6, most classifiers have an accuracy near or above 90% with the exception of the MNB classifier (accuracies around 60%). Overall, the NN's K6 'N removed' test performed best at 94.5% in 0.39 seconds. In all K6 tests the 'N removed' iteration performed at least 0.01 seconds faster than the 'N included' and 'N ignored' tests, on the same classifier.

The K9 tests all had accuracies above 94% where tests could be conducted, recall that memory constraints limited the NN to the K3 and K6 tests only. Using the Linear SVM, accuracies were the same across all three tests, but like the other tests at K9, the 'N removed' iteration was at least 0.1 seconds faster than the 'N included' and 'N ignored' iterations.

Overall, the highest accuracy and speed combination was from the K9 MNB 'N removed' test, taking 0.61 seconds to run 699 test genes and achieving a 99.1% accuracy.

REFERENCES

[1] K. B. Mohammed, S. V. Boyapati, M. D. Kandimalla, M. B. Kavati, and S. Saleti, "A comparative analysis of the evolution of dna sequencing techniques along with the accuracy prediction of a sample dna sequence dataset using machine learning," in 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS), pp. 1–5, 2023.

[2] S. Juneja, A. Dhankhar, A. Juneja, and S. Bali, “An approach to dna sequence classification through machine learning,” Aug 2022.

[3] S. Solis-Reyes, “Dna sequence classification: It’s easier than you think: An open-source k-mer based machine learning tool for fast and accurate classification of a variety of genomic datasets,” 2018.

[4] M. Uddin, M. K. Islam, M. R. Hassan, F. Jahan, and J. H. Baek, “A fast and efficient algorithm for dna sequence similarity identification,” Aug 2022.

[5] B. E. Blaisdell, “Effectiveness of measures requiring and not requiring prior sequence alignment for estimating the dissimilarity of natural sequences,” Dec 1989.

[6] C. Burge, A. M. Campbell, and S. Karlin, “Over- and under- representation of short oligonucleotides in dna sequences.,” Feb 1992.

[7] E. Asgari, K. Garakani, A. C. McHardy, and M. R. K. Mofrad, “MicroPheno: predicting environments and host phenotypes from 16S rRNA gene sequencing using a k-mer based representation of shallow sub-samples,” Bioinformatics, vol. 34, pp. i32–i42, 06 2018.

[8] P. Ng, “dna2vec: Consistent vector representations of variable-length k-mers,” 2017.

[9] N. S. Chauhan, “DNA sequence dataset, v1,” 2020. https://www.kaggle.com/datasets/nageshsingh/dna-sequence-dataset.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[11] Y. Yuan, L. Wu, and X. Zhang, “Gini-impurity index analysis,” IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3154–3169, 2021.

[12] T. Rumpf, A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, and L. Pluemer, “Early detection and classification of plant diseases with support vector machines based on hyperspectral reflectance,” Computers and Electronics in Agriculture, vol. 74, no. 1, pp. 91–99, 2010.

[13] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, “Multinomial naive bayes for text categorization revisited,” in AI 2004: Advances in Artificial Intelligence (G. I. Webb and X. Yu, eds.), (Berlin, Heidelberg), pp. 488–499, Springer Berlin Heidelberg, 2005.

[14] A. C. Gay Thome, “Svm classifiers – concepts and applications to character recognition,” Nov 2012.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, pp. 8024–8035, Curran Associates, Inc., 2019.