Original Research Article

Protecting genetic genealogical databases from identical-by-state probing attacks: a machine learning-based approach

ABSTRACT

Identical-by-state (IBS) probing is a way of attacking a public genealogy database to discover the identities of people who share specific qualities. The attacker creates an IBS-inert DNA sequence (IBS-inert-DNA-sequence) and combines it with a sequence containing the trait of interest. The final sequence matches people with genomic areas similar to the trait. To prevent attacks, it was hypothesized that the design of IBS-inert-DNA-sequences is based on the principle that they are susceptible to detection by skilled machine learning systems, because the attacker purposely creates an IBS-inert-DNA-sequence which is structurally dissimilar to real DNA sequences. Thedataset consisted of real DNA (from the UCI Machine Learning Repository's splice junction gene sequences dataset) and computer-generated sequences. Eighteen non-identical Random Forest classifier models (RF-models) were created to determine the best configurations for discriminating between real and computer-generated sequences. The findings revealed that an optimized RF-model combined with *k*-mer and *n*-gram values of 2 each, resulted in the most performant model, with accuracy, sensitivity, specificity, false positive rate, Matthew's correlation coefficient, and area under the receiver operating characteristic curve values of 88.3%, 84.8%, 91.8%, 8.2%, 0.768 and 0.958, respectively. A decline in performance was linked to an increase in *k*-mer size.

Keywords: Identical-by-state, attack, genealogical databases, random forest

1. INTRODUCTION

Direct-to-consumer genetic testing (DTC-GT) services provide genetic information to individuals without clinical or research intervention[1]. To do this, the DTC-GT services send genetic testing devices to their clients, who return them with biological samples usually consisting of saliva. After the biological samples have been processed, genetic informationabout each client is obtained and uploaded to specialized databases. The clients can then proceed to download their genetic data files (GDF) which contain their

raw genetic data[2].According to Curnutte and Testa[3], the DTC-GT services justify their actions by claiming that consumers have a right to personal information, especiallythat which can help them learn more about their genealogies and improve their health.

When these clients obtain their genetic data, many of them proceed to volunteer the data to third party services like openSNP, DNA.Land and GEDmatch which run genealogical databases. The reasons for sharing their genetic data with others on online platforms are varying and at times, very personal. As an example, Edge and Coop[4] explained that adoptees and individuals who were conceived using donated sperm cells may want to find, or to be found by their blood relatives.Many of these genealogical databases allow just about anyone to upload unauthenticated GDFs. In some cases, when genetic matches are found, the uploader is supplied with both the names and the contact details of the matches[4]. This has certain advantages to bodies such as law enforcement agencies which aim at quickly identifying the relatives of an unknown suspect whose DNA sample was recovered from a crime scene[5].

The existence of these large human genetic databases and the fact that the identities of genetic matches are revealed to any uploader presentadvantages and disadvantages. Haeusermann *et al*. [1]explained that the availability of large genetic datasets could help scientists to study human diseases better, increase public involvement in science and establish participant-driven research initiatives. They also outlined some negative consequences. For example, given the immutability of genetic data, obtaining and disclosing the genetic characteristics of an individual may be very harmful to the concerned, as abuse, stigmatization and risks of discrimination in education, insurance and employment could ensue. Carmi[6] correctly stated that loss of privacy may further extend from an individual to their relatives.

In order to address the above-mentioned negative aspects, it is important to know how individuals in genealogical databases who possessmedically sensitive genetic markers can be identified. Edge and Coop [4] and Ney *et al*.[2]explaineddifferentmeans by which an ‘adversary’ may obtain information from a genealogical database, especially information which the ‘adversary’ is not supposed to have. In their paper, Edge and Coop [4]demonstrated an approach named identical-by-state (IBS) probing, through which an adversary can obtain information about individuals whopossess a specific disease allele. Identical-by-state segments are (near) identical DNA segments which are shared by different individuals.

In IBS probing, the adversary uploads a DNA sequence consisting of an allele of interest surrounded by fake genotype data. This fake genotype data is designed such that it has a low possibility of sharing an IBS segment with anyone in the database (in other words, it is IBS inert). Due to this design, any hits would most likely reveal individuals who possess the allele of interest.

In their proposed countermeasures, Edge and Coop [4]made a suggestion (countermeasure 8) which triggered this study: uploads with evidence of IBS-inert segments should be identified and blocked. To our knowledge, we are the first to specifically consider machine learning as a potentially efficient genetic database protection tool against IBS probing attacks. We hypothesized that the same principle behind the design of IBS-inert DNA sequences makes these sequences prone to detection by skilled machine learning systems. Since the attacker purposefully creates an IBS-inert DNA sequence and designs it to be structurally dissimilar to real DNA sequences, skilled machine learning algorithms can greatly exploit this dissimilarity in order to differentiate real DNA sequences from fake ones(which are purposely designed not to structurally resemble real DNA sequences).The aim of this study was to test the above-mentioned hypothesis.

2. MATERIALS AND METHODS

2.1.System specification

The in-house scripts used for this research were written in Python 3 with system specification as follows: 64-bit operating system, x64-based processor, 8GB RAM, intel CORE i5 processor. After the results were obtained, both the scripts and the research datasets were pushed to a GitHub repository. The datasets and source code are found at https://github.com/Enowtakang/study-ml-0322.

2.2. Dataset preparation

A dataset containing labeled examples of real DNA sequence samples and those generated by the computer (to serve as fake genetic data) was prepared, and the skill of a chosen machine learning model in differentiating the real from the fake DNA sequence samples was tested. The real DNA sequence samples were obtained from the splice junction gene sequences dataset available at the popular UCI machine learning repository [7].

Upon downloading the dataset (splice.data) and initially inspecting it in notepad, it was found to have three columns. It was then loaded into a pandas data frame using the read_csv function which is suitable for reading files with coma-separated values. Since the dataset lacked headers, arbitrary letters were provided as names for the first and second columns (*a* and *b* respectively) while the third column containing the 3,190real DNA sequence samples,each of sixty nucleotides long, was named *dna_sequence*. Next, an extra column (*dna_label*) was added to the data frame. This column was then filled with ones (1s) using the pandas Series function and a *for* loop. The labels (1s) indicated that each DNA sequence sample was real. The *dna_sequence* and the *dna_label* columns were further isolated to form a new data frame.

To create a data frame containing generated DNA sequences, an empty pandas data frame was first created. Next, a function to generate one DNA sequence was defined. Using the sequence generation function and list comprehension, a list of 3,190 DNA sequences sixty nucleotides-long were generated. Using a *for* loop and list comprehension, the list of generated DNA sequences was stored in pandas Series and further inserted into the first column of the empty data frame using the pandas insert function. During insertion, it was named *dna_sequence*.Just as it was done in the first data frame, an extra column (*dna_label*) was added to this data frame containing the generated DNA sequences and filled with zeros (0s) to signify that the DNA sequence examples were not real. The two data frames containing only labeled real and generated DNA sequences were merged using the pandas concat function and saved to a coma separated value (CSV) file. This CSV file was later reloaded and the whitespace before each of the real DNA sequence samples was removed. Finally, the data frame ready for machine learning binary classification was exported as an excel file. This final dataset (research_data.xlsx) was perfectly balanced, with two classes (0 and 1) used to label all the 6,380 DNA sequence instances as either generated or real DNA sequence samples.

2.3. Data preprocessing

In most machine learning algorithms that use numerical data rather than categorical data (contain letters rather than numbers), preprocessing data is the most important stage[8]. Since the research dataset was categorical, it had to be converted to a numerical form. A process known as *k*-merization was usedto convert each DNA sequence into a concatenated succession of *k*-mer subsequences. A *k*-mer is a

subsequence of length *k*. For instance, the DNA sequence ATGC would have four 1-mers (A, T, G, C), three 2-mers (AT, TG, GC), two 3-mers (ATG, TGC) and one 4-mer (ATGC). 2-mers, 4-mers and 8-mers were used during this study. *k*-merization produced a feature set that was further transformed into numerical data using Scikit-Learn's CountVectorizerwhich converted the feature set into a sparse matrix. 2-gram, 4-gram and 8-gram specifications were used during this study, instead of the 1-gram default value for the ngram_range parameter.

2.4. Sequence cluster visualization

This was used to visualize the intra-label and inter-label relationships between the DNA sequences. To do this, Scikit-Learn's train_test_split method was first used to separate the feature set (X) and the label set (y_data) into training and test batches. Specifically, twenty percent of the X and y_data values were kept for model testing. The random_state parameter was given a numerical value (42) to ensure that if the code was rerun, the exact train-test split would be obtained, without changing the results. The stratify parameter was given the value y_data (the label set variable name). This ensured that the training and test datasets contained examples of each class in the same proportions as in the original dataset. It is important to note that this specification would be of paramount value if the classes in the original dataset were imbalanced.

After splitting the data into training and test batches, Scikit-Learn's truncated singular value decomposition (TruncatedSVD) method was used to reduce the multi-dimensional training feature set to two dimensions using its n_components parameter. This method was used for dimensionality reduction because unlike principal component analysis (PCA) which centers the data before computing singular value decomposition, truncated SVD does not center the data, making it good for working with the sparse matrices earlier produced using Scikit-Learn's CountVectorizer. Scikit-Learn's t-distributed stochastic neighborhood embedding (TSNE) method was also used to reduce the multi-dimensional training feature set to two dimensions using its own n_components parameter. In addition, its perplexity parameter was set to 100 and its random_state parameter was set to 42. Finally, sequence cluster visualizations with truncated SVD and t-SNE were generated with Matplotlib's pyplot method.

2.5. Random Forest classifier

Random Forest (RF) classifier is an ensemble classifier that uses randomness to create a group of independent and non-identical decision trees[9]. Each tree is first constructed using a random bootstrapped sample of the training data. Then, the RF model only considers a small subset of features for training at each split of the tree. This reduces variance, which helps to increase generalization. The final classification is determined by merging the results of the decision trees that have received the highest number of votes. When used on large dimension data, the RF bagging approach can effectively reduce the danger of overfitting. As a result, RF can handle a large data set with a lot of dimensions[10].

2.6. Hyperparameter tuning

Even though Scikit-Learn implements pragmatic default parameters for the RF model, these default parameters are not guaranteed to be optimal for all problems.The most important hyperparameters of the RF model were optimized before training using Scikit-Learn. They included the number of trees in the forest, the minimum number of data points placed in a node before splitting it, the minimum number of data points allowed in a leaf node, the maximum number of features considered for splitting a node, the maximum number of levels in each decision tree and the method for sampling data points (with or without replacement). In order to narrow the search for the best hyperparameter values, we first evaluated a wide range of values for each hyperparameter using Scikit-Learn's RandomizedSearchCV method by specifying a grid of hyperparameter ranges and randomly sampling 100 candidates from the grid, performing 3-fold cross-validation with each combination of values. The best parameters are shown in Table 1. After narrowing down the range for each parameter, we used Scikit-Learn's GridSearchCV to explicitly specify every combination of settings to try. GridSearchCV evaluates every combination possible instead of randomly sampling from a distribution. A 3-fold cross-validation was fitted for each of the 288 possible combinations totaling 864 fits. The best results are also shown in Table 1.

Table 1. Output from hyperparameter tuning

	Hyperparameter					
Search method						
	n estimators	min samples split	min samples leaf	max features	max depth	bootstrap
default	10	2	1	'auto'	None	True
RandomizedSearchCV	1400	5	1	'sqrt'	800	False
GridSearchCV	1100	3	2	3	700	False

2.7. Model performance evaluation

The RF classifier was applied on the dataset 18 non-identical times.Each time, a distinct configuration consisting of one value eachfrom the k-mer (2-mer, 4-mer and 8-mer), n-gram (2-gram, 4-gram and 8-gram) and hyperparameter tuning decision (tuned model and default model)options was used. From eachof the resulting confusion matrices, true positive (TP) and true negative (TN) numbers of truly identified generated and real DNA sequences were obtained for each triplet combination. Also, false-positive (FP) and false-negative (FN) numbers of falsely identified generated and real DNA sequences were obtained for each triplet combination.

Accuracy (ACC), sensitivity (SN), specificity (SP)[11], and Matthews correlation coefficient (MCC) [12]were computed with the following equations:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Sensitivity = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{TN+FP}$$

$$MCC = \frac{((TP*TN)-(FP*FN))}{\sqrt{((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))}}$$

Another quantitative measure of great importance was the false positive rate (FPR)[13]which summarized how often a real DNA sequence prediction was made when in fact the outcome was a generated DNA sequence. The false positive rate was computed with the following equation:

False positive rate= $\frac{FP}{FP+TN}$ = 1 –*Specificity*

Finally, Scikit-Learn's roc_auc_score method was used to compute the area under the receiver operating characteristic (ROC) curvefor each of the models resulting from all 18 configurations. Scikit-Learn's roc_curve method was used to visualize the skills of the best and worst performing configuration by plotting their ROC curves.

3. RESULTS

3.1 Sequence clusters

The SVD scatterplot (Figure 1a) shows that on an intra-label scale, both the real DNA sequences (blue points) and the generated DNA sequences (red points) cluster around each other appreciably well. However, the real DNA sequences cluster better than the generated DNA sequences since the lose points in the real DNA sequences lie at the periphery of the fairly elliptical blue cluster, while the lose points in the generated DNA sequences cause the overall cluster shape to unevenly stretch out, significantly affecting its core elliptical nature.The TSNE scatterplot (Figure 1b) shows that on an inter-label scale, the real DNA sequences and those generated do not form separate clusters. In fact, there is significant overlapping of both red and blue clusters, with the red cluster virtually lying over the blue one.

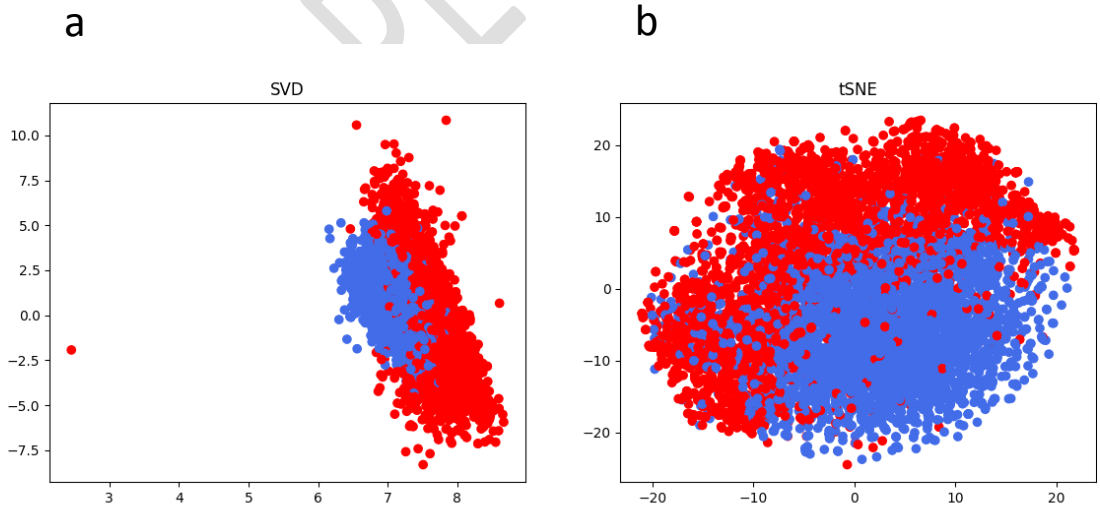


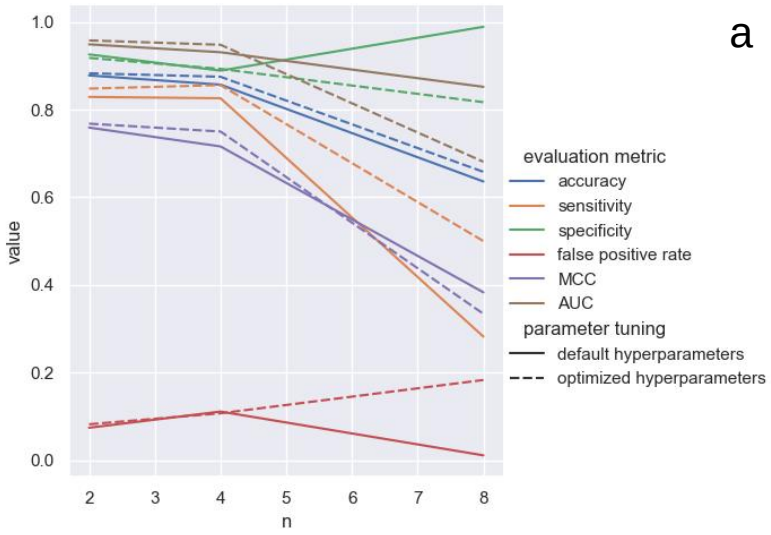
Figure 1. SVD (a) and TSNE (b scatterplots of both real DNA sequences (blue points) and generated DNA sequences (red points).

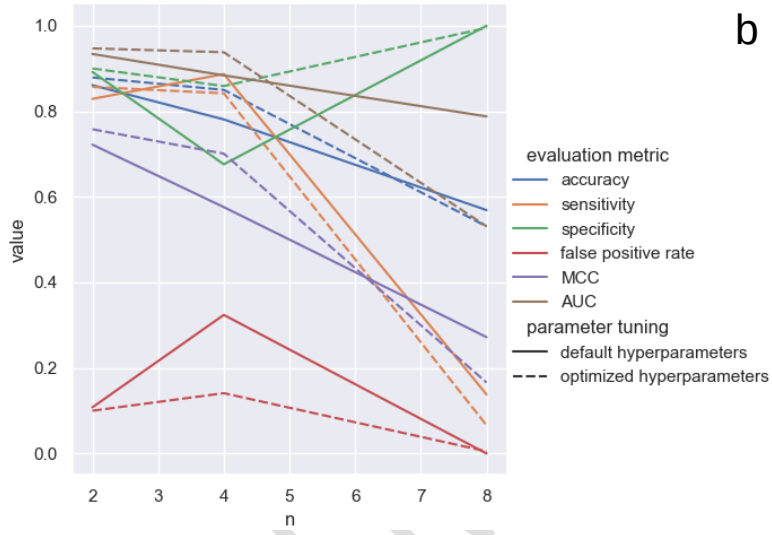
3.2. Hyperparameter tuning output

Table I shows that there are differences between the default hyperparameter values and final tuned hyperparameter values. For example, the default minimum number of data points placed in a node before splitting is 2, while the optimal number is 3. Also, there is no default maximum number of levels in each tree whereas the optimal maximum number of levels in each tree is 700.

3.3 Comparing model performance under different configurations

As stated earlier, the RF classifier was applied on the dataset, a total of 18 non-identical times. Each time, a distinct configuration consisting of one value, each from the*k*-mer (2-mer, 4-mer and 8-mer), n-gram (2-gram, 4-gram and 8-gram) and hyperparameter tuning decision (tuned model and default model) options was used.Figure 2shows the variation in RF classifier performance under the different configurations. Since the FPR is calculated directly from the model specificity metric, their trends correlated for all*k*-mer values. For example, at a *k*-mer size of 2, model performance for all evaluation metrics dropped with increasing *n*-gram values, with the exceptions of both the pre-optimization specificity metric which improved between *n*-gram values of 4 and 8 and the pre-optimization FPR which also improved between *n*-gram values of 4 and 8. At a *k*-mer size of 4, the overall trend was mostly similar to that of *k*=2. However, unlike the results for *k*=2, the numerical differences between the different performance metrics were wider. Also, the post-optimization specificity metric followed a similar trend to that of the pre-optimization specificity metric. At a *k*-mer size of 8, the numerical differences between the different performance measures were widest. Also, the drop in model performance metrics was more noticeable between *n*-gram values of 2 and 4. Comparatively slight drops in model performance metrics were noticed between *n*-gram values of 4 and 8, except for pre-optimization specificity values which rather increased slightly, and pre-optimization AUC values which dropped more steeply.


a
value
n
evaluation metric
accuracy
sensitivity
specificity
false positive rate
MCC
AUC
parameter tuning
default hyperparameters
optimized hyperparameters


b
value
n
evaluation metric
accuracy
sensitivity
specificity
false positive rate
MCC
AUC
parameter tuning
default hyperparameters
optimized hyperparameters

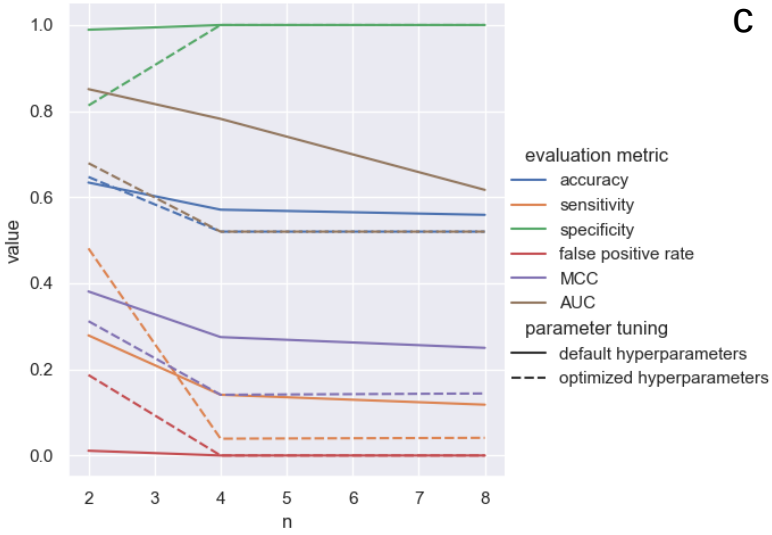

c
value
n
evaluation metric
accuracy
sensitivity
specificity
false positive rate
MCC
AUC
parameter tuning
default hyperparameters
optimized hyperparameters

Figure 2. Performance analyses of RF classifier under varying pairwise combinations of k-mer (2-mer, 4-mer, 8-mer) and n-gram (2-gram, 4-gram, 8-gram) values before and after model hyperparameter optimization; a, b & c show variations in model evaluation metrics under constant k-mer values of 2, 4 and 8.

The general trend which characterized model performance with increasing *k*-mer valueswas a drop in model performance metrics. Also, model optimization improved model performance, with the exception of performances at k=2 and *n*-gram between 4 and 8, where model optimization rather decreased model performance. From the numerical results of model performance in Table 2 of Appendix1, the model configuration with the best performance metrics hada 2-mer, 2-gram and optimized hyperparameters configuration. The model with the worst performance had an8-mer, 8-gram and optimized hyperparameters configuration. Figure 3 shows the ROC curves of the best (AUC=0.958) and worst (AUC=0.520) RF runs. Since the dataset had perfect class balance, there was assurance that the curves were not presenting an optimistic picture of the model. The ROC curve is a plot of the FPR (x-axis) versus the hit rate (y-axis).

Lower false positives and higher true negatives are shown by lower values on the x-axis of the plot (Figure 3a). Greater true positives and smaller false negatives are shown by higher values on the y-axis of the plot (Figure 3a). A model whose ROC curve resembles a straight line from the bottom left to the top right of the graph is a naïve model (Figure 3b). Such a model would have a minimum AUC of 0.5, since it would most often predict the most common class.

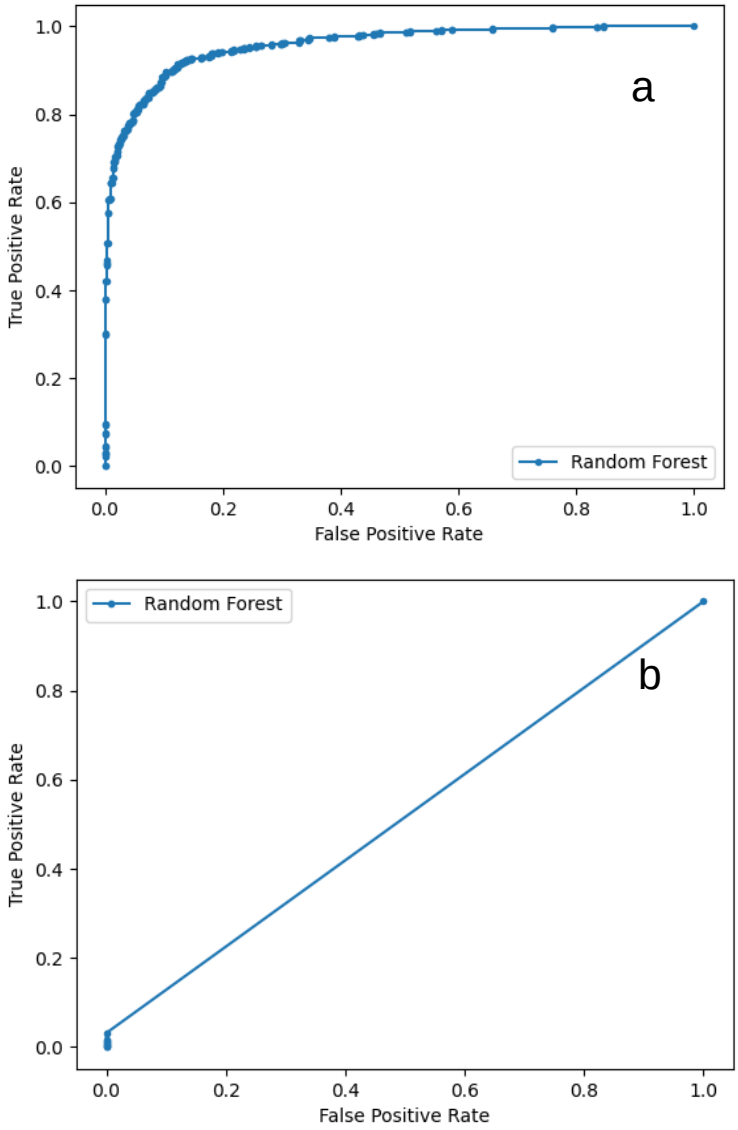


Figure 3. ROC curves of (a) the best (AUC=0.958) and (b) the worst (AUC=0.520) RF runs.

4. DISCUSSION

In his 1970 book entitled *Chance and Necessity: An Essay on the Natural Philosophy of Modern Biology*, the French Nobel Prize laureate Jacques Monod wrote: 'A totally blind process can by definition lead to anything; it can even lead to vision itself.'The generated DNA sequence cluster overlapped with the real DNA sequence cluster most probably because after random generation, there were significant similarities between sequences from both groups. This observation gives an idea about the strength of the RF

classifier used, since the data points were not linearly separable and required a sufficiently sophisticated model to learn them and score high during testing.

Increasing *k*-mer valuesgenerally resulted in poor model performance because smaller k-mer sizes increase the number of features, which in turn provide more detail for the model during learning. A model which learns on more detail would be more capable of predicting the correct class to which an unseen example should belong than a model which learns on less detail. These results confirm those of Alam and Chowdhury[14] who reported that the features they constructed from short k-mers displayed strong discriminatory capacity among their sequence classes.

Thenon-ideal nature of the research dataset was identified as the principal limitation of this study.It was noted thatthe real DNA sequences which were used lacked the degree of structural complexity which is inherent to any human genome sequence that is stored in a GDF, namely retrotransposed elements and DNA transposons like long and short interspersed nuclear elements (LINEs and SINEs), gene components such as introns, exons, promoter elements, frame shifts, splicing signals, and open reading frames, other functional sequences like regulatory elements and non-functional sequences such as pseudogenes, intergenic regions and endogenous retrovirus sequences[15],*inter alia*.Also, the size of the dataset was not ideal. For a robust sequence classifier which can (almost) perfectly differentiate real from unreal DNA sequences, many hundreds of thousands (or even millions) of examples are required, given that the objective is to obtain a model which has the skill level necessary to protect genetic genealogical databases from IBS probing attacks. Lastly, we did not experiment on the classifier performance in the context of actual IBS inert sequences (since we lacked the technological gear to perform the required computationally expensive experiments).

We hope that in future, our idea would be tested in environments which would yield workable versions of our proposed solution, given that it can be directly implemented by the concerned companies without the need to wait for solutions which require inter-organizational cooperation (often made complex by existing policies and bureaucratic bottlenecks).

5. CONCLUSION

With respect to the aim of this study, we believe that we have shown the potential of using machine learning to identify unreal DNA sequences. Machine learning has the potential to identify and block the uploading of IBS-inert (unreal) DNA sequences into public genealogical databases, and also to decontaminate these databases by identifying and removing already-uploaded IBS-inert DNA sequences.

REFERENCES

1. Haeusermann T, Fadda M, Blasimme A, Tzovaras BG, Vayena E. Genes wide open: data sharing and the social gradient of genomic privacy. AJOB Empir Bioeth. 2018;9(4):207–221.
2. Ney P, Ceze L, Kohno T. Genotype extraction and false relative attacks: security risks to third-party genetic genealogy services beyond identity inference, 2020;637134336000000000. https://www.sciencegate.app/document/10.14722/ndss.2020.23049 (accessed Mar. 15, 2022).
3. Curnutte M, Testa G. Consuming genomes: scientific and social innovation in direct-to-consumer genetic testing. New Genet. Soc. 2012;31(2):159–181.
4. Edge MD, Coop G. Attacks on genetic privacy via uploads to genealogical databases. eLife. 2020;9:e51810
5. Edge MD, Coop G. How lucky was the genetic investigation in the golden state killer case?" BioRxiv. 2019;531384.
6. Carmi S. The challenges of maintaining genetic privacy. eLife. 2020;9:e54467.
7. UCI Machine Learning Repository Molecular biology (Splice-junction Gene Sequences) Data Set. https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences) (accessed Mar. 15, 2022).
8. Gunasekaran H, Ramalakshmi K, Arokiaraj ARM, Kanmani SD, Venkatesan C, Dhas CSG. Analysis of DNA sequence classification using CNN and hybrid models. *Comput Math Methods Med.* 2021;1835056.
9. Ali J, Khan R, Ahmad N, Maqsood I. Random forests and decision trees. Int. J. Comput. Sci. Issues 2012;9(5):272–278.

10. Bhandari N, Khare SP, Walambe R, Kotecha K Comparison of machine learning and deep learning techniques in promoter prediction across diverse species. Peer J. Comput. Sci 2021;7:e365.
11. Mohamed EA, Rashed EA, Gaber T, Karam O. Deep learning model for fully automated breast cancer detection system from thermograms. PloS One 2022;17(1):e0262349.
12. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta 1975;405(2):442–451.
13. Adeshina YO, Deeds EJ, Karanicolas J. Machine learning classification can reduce false positives in structure-based virtual screening. Proc Natl Acad Sci. 2020;117(31):18477–18488.
14. Alam MNU, Chowdhury UF. Short k-mer abundance profiles yield robust machine learning features and accurate classifiers for RNA viruses. PloS One 2020;15(9):e0239381.
15. Caballero J, Smit AFA, Hood L, Glusman G. Realistic artificial DNA sequences as negative controls for computational genomics. Nucleic Acids Res. 2014;42(12):e99.

Appendix 1

Table 2. Effects of different *n-gram range*&*k-mer* pairwise configurations on Random Forest model performance before and after optimization of model hyperparameters.

	k-mer						
	2-mer		4-mer		8-mer		
n-gram range	default model	optimized model	default model	optimized model	default model	optimized model	
(2, 2)	0.878	0.883	0.861	0.879	0.634	0.647	ACC
	0.829	0.848	0.829	0.857	0.279	0.481	SN
	0.926	0.918	0.892	0.900	0.989	0.813	SP
	0.074	0.082	0.108	0.100	0.011	0.187	FPR
	0.759	0.768	0.722	0.758	0.381	0.312	MCC
	0.949	0.958	0.934	0.947	0.851	0.679	AUC
(4, 4)	0.857	0.875	0.781	0.850	0.571	0.520	ACC
	0.826	0.856	0.887	0.842	0.141	0.039	SN
	0.889	0.893	0.676	0.859	1.000	1.000	SP
	0.111	0.107	0.324	0.141	0.000	0.000	FPR
	0.716	0.750	0.576	0.701	0.275	0.141	MCC
	0.931	0.948	0.884	0.938	0.782	0.520	AUC
(8, 8)	0.636	0.658	0.569	0.531	0.559	0.520	ACC
	0.282	0.500	0.138	0.066	0.118	0.041	SN
	0.989	0.817	1.000	0.995	1.000	1.000	SP
	0.011	0.183	0.000	0.005	0.000	0.000	FPR
	0.383	0.334	0.272	0.166	0.250	0.144	MCC
	0.852	0.681	0.788	0.531	0.617	0.520	AUC