deepBreaks: a Machine Learning Tool for Identifying and Prioritizing Genotype-phenotype Associations

Mahdi Baghbanzadeh
https://orcid.org/0000-0002-1878-2691
Tyson Dawson
The George Washington University
Bahar Sayoldin
The George Washington University
Todd Oakley
University of California
Keith Crandall
George Washington University https://orcid.org/0000-0002-0836-3389
Ali Rahnavard (rahnavard@gwu.edu)
The George Washington University https://orcid.org/0000-0002-9710-0248

Article

Keywords:

Posted Date: February 1st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2534899/v1

Additional Declarations: There is **NO** Competing Interest.

deepBreaks: a machine learning tool for identifying and prioritizing genotype-phenotype associations

Authors: Mahdi Baghbanzadeh[1], Tyson Dawson[1], Bahar Sayoldin[1], Todd H. Oakley[2], Keith A. Crandall[1], Ali Rahnavard[1,*]

Affiliations:
[1]Computational Biology Institute, Department of Biostatistics and Bioinformatics, Milken Institute School of Public Health, The George Washington University, Washington, DC 20052
[2]Ecology, Evolution, and Marine Biology, University of California, Santa Barbara, California 93106

*Correspondence to rahnavard@gwu.edu

Key points:

- **Generality:** *deepBreaks* is a new computational tool for identifying genomic regions and genetic variants significantly associated with phenotypes of interest.
- **Validation:** A comprehensive evaluation of *deepBreaks* performance using synthetic data generation with known ground truth for genotype-phenotype association testing.
- **Interpretation:** Rather than checking all possible mutations (breaks), *deepBreaks* prioritizes statistically promising candidate mutations.
- **Elegance:** User-friendly, open-source software allowing for high-quality visualization and statistical tests.
- **Optimization:** Since sequence data are often very high volume (next-generation DNA sequencing reads typically in the millions), all modules have been written and benchmarked for computing time.
- **Documentation:** Open-source GitHub repository of code complete with tutorials and a wide range of real-world applications.

Table of contents

Abstract

Sequence data (e.g., nucleotides or amino-acids) are critical for advancing our understanding of biology. However, there are many challenges with investigating and analyzing sequencing data and genotype-phenotype associations, including non-independent observations, large noise components, nonlinearity, colinearity, and high dimensionality. Therefore, machine learning (ML) algorithms are well suited for analyzing sequence data as they can capture nonstructural patterns of relationships with the biology of interest as genotype-phenotype associations. Nevertheless, flexible and user-friendly implementations of ML approaches are lacking, especially ones that take advantage of the unique features of high-volume DNA sequence data. Here, we present *deepBreaks*, a generic approach with unified data analysis steps that identify the most important positions in sequence data that correlate with phenotypic traits of interest. *deepBreaks* compares the performance of multiple ML algorithms and prioritizes the most important positions based on the best-fit models. *deepBreaks* is open-source software with documentation available online at https://github.com/omicsEye/deepBreaks.

Main

We developed a generic and computationally optimized tool, namely *deepBreaks*, to identify and prioritize important sequence positions in genotype-phenotype associations. Our approach is as follows: first, we prepare a training dataset based on the provided raw sequencing data. Second, we fit multiple machine learning algorithms and, based on their cross-validation score, select the best model and use the model to predict the phenotype of interest based only on its provided sequence and then try to interpret this model in order to find the most discriminative positions of the sequence. By doing this, we not only assess the possibility of the predictability

of the phenotype based on the sequences but also use the most accurate predictive models to find out and prioritize the positions in the sequence that are more predictive.

Advancements in sequencing and computational technologies have provided researchers with large-scale data, and the development of tools for analyzing such data is growing[1]. A great challenge in developing predictive sequence-to-phenotype models is to deal with linear and non-linear effects and consider the whole sequence simultaneously, which makes machine learning algorithms suitable to address these problems[2]. Decoding and interpreting results from predictive models are essential to make biological inferences[3,4]. Fitting machine learning and deep learning models on sequence data to model traits has been studied in various frameworks ranging from predicting drug resistance[5–7] to cancer detection[8,9]. *deepBreaks* is developed to rank the performance of machine learning models that best fit the data and then, based on those models, prioritize and report the most discriminative positions of the sequence with respect to a given phenotype of interest. Early efforts in this field were implemented by proposing a Bayesian method and utilizing all the marker data simultaneously to predict the phenotype[10]. Machine learning approaches for genotype-phenotype associations have evolved, and some support more effective and reproducible use of multivariate genotype data for the prediction of quantitative traits[11]. Tools such as KOVER predict phenotypes based on reference-free genomes using k-mers (short DNA sequences) as the features, chi-square tests for filtering redundant features prior to modeling[12], and for the features that have exact equal values, assign the same importance score[12]. Studies have attempted various approaches for predicting phenotypes based on the sequence, including using dense neural networks[13], convolutional neural networks[14,15], and ensemble learners[14]. Comparing the outcome of different machine learning algorithms shows that there is no universally best predictive algorithm for the diversity of genotype-phenotype studies[13]. To find the best model that is suited for a given set of data, researchers compared several models and made their inferences about the feature importances only based on the best model[16].

In this paper, we have evaluated the performance of the *deepBreaks* approach on simulated data and assessed its performance in finding the important variables in a dataset with the ground truth. We have also applied *deepBreaks* to multiple datasets to show its wide applications and power to detect the most important positions in both nucleotide and amino acid sequence data. In the methods section, we elaborate on the steps that *deepBreaks* takes to prepare the data, fit models to the data, and interpret the results. *deepBreaks* is a generic software that can be applied in sequence-to-phenotype studies to show the feasibility of first predicting the phenotype based on a sequence and then determining what are the most discriminative parts of the sequence in predicting the phenotype.

Results

Methods overview

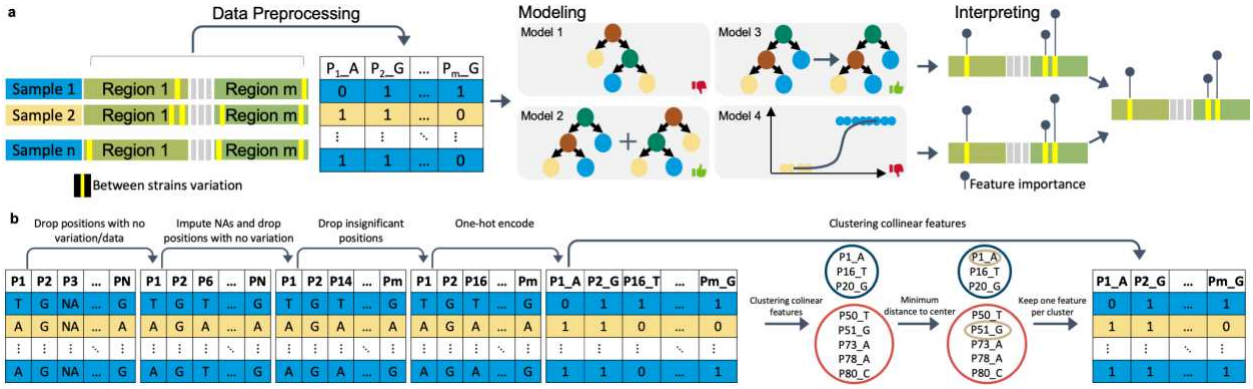

Figure 1: *deepBreaks* overall workflow. a) *deepBreaks* begins with sequencing data organized in a multiple sequence alignment (MSA) format. Sequences can be nucleic acids or amino acids. The phenotype of interest is also a required input parameter. Preprocessing steps are conducted to recode the sequencing data and phenotype into a format usable by the machine learning models. A modeling step follows in which various models are attempted and ranked. The best model is probed to identify the positions which best predict phenotype. These results are then merged and presented to the user as visualizations and interpretable tables. **b)** The illustrated preprocessing steps implemented in the *deepBreaks* pipeline are essential for the approach efficiency by summarizing positions used in analyses.

The input data of *deepBreaks* is a Multiple Sequence Alignment (MSA) file containing$X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, $i \in \{1,2,\ldots,n\}$, *n* sequences of length m and a phenotypic metadata, with a vector of size *n*, and p_is, as phenotypes which are related to the i^{th} sequence. x_{ij} (the j^{th} element of the sample i) can be a subset of $\{A, T/U, C, G\}$, or any one-letter character representing an amino acid. Phenotypes (p_is) can be continuous measures such as height, BMI, categorical, or binary variables such as obese/healthy weight/underweight, antibacterial resistance/sensitivity, or mild/severe cases of a disease. *deepBreaks* has three phases: i) preprocessing, ii) modeling, and iii) interpreting (**Fig. 1a**). In the data preprocessing phase, illustrated in **Fig. 1b**, we deal with imputing missing values, ambiguous reads, dropping zero-entropy columns, clustering correlated positions (features), and dropping redundant features which do not carry a significant amount of information in association with the phenotype under study. To keep track of positions before starting to drop the columns, all of them (by default) are named from p_1 to p_m. The names of the columns (positions) in the dataset are fixed, and dropping certain columns does not change the position names in a sequence. To identify the colinearity of features, we cluster features based on their pairwise distances. Then we cluster them using the density-based spatial clustering of applications with noise (DBSCAN) method[17] algorithm and take the feature that is the closest to the center of the cluster as the representative of the cluster in the training data set[18]. Two sets of models for continuous or categorical phenotypes are incorporated in our training model phase, and a complete list of these models with their default parameters is available in the Methods section. For model comparison, *deepBreaks* by default uses a 10-fold cross-validation approach and ranks the models based on their average cross-validation score. The default performance metrics for

regression and classification that *deepBreaks* uses are mean absolute error (MAE) and F-score, respectively. A complete list of other performance metrics is available in the Methods section. For interpreting the contribution of the positions in the sequence to the predictive models, we use the feature importance and coefficients. These importance values are then scaled to 0 and 1 (maximum importance). We also consider the same importance values for features that have been clustered together. We provided a detailed elaboration of the pipeline in the Methods section.

Simulation Study

In the data preprocessing steps, we first, drop the redundant positions based on the p-values of a statistical test, and then to deal with the colinearity of features, we calculate the distance between features and then cluster them into groups of features and select one of them as the representative of the group. We designed a simulation study to assess the effects of these steps and the changes in the threshold for p-value and distance metric on the performance of three different models (Adaboost, Decision Tree, and Random Forest) and their ability to estimate the true effect size of the groups of features in different datasets.
Each data set is simulated based on this formula:

$$y = X\beta + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2)$$

$$X_{n\times m} = [X_{n\times informative} \quad X_{n\times redundant}]$$

$$\beta_{m\times 1} = \begin{bmatrix} \beta_{informative\times 1} \\ \mathbf{0} \end{bmatrix}, \qquad \beta_{informative_i} \sim N(0, 100)$$

We first created a data matrix of size $X_{n\times m}$ with binary data, then we selected a subset of its features as informative. The corresponding regression coefficients of these selected informative features were then sampled from a normal distribution and the rest was considered zero. We call this data set, the initial dataset which has no colinear features. Then, based on the initial dataset, we created datasets with colinear features. To add a colinear feature, we randomly selected one of the informative features, and then replaced 40% of its values with random samples from a binary distribution. We repeat this process 5 times and each time use the data set created in the last step as the initial data set. So, we start with data with no colinear features and gradually increase the number of colinear features. **Fig. 2** illustrates three steps of the simulation on a sample dataset with 10 features (4 informative, 6 redundant).

Figure 2: **Creating training datasets with different levels of multicolinearity.** The first dataset has independent features and only some of them are informative. Then, we start to replace the redundant features with samples from the informative features to add colinear features to the training dataset. For example, we took a sample from f1 and replaced it with f5.

The correlations between sampled features in the dataset are a random number between 0.6 to 1 (exactly the same). With each data frame, we train models with different combinations of thresholds for correlation (no clustering, 0.7, and 0.9) and different thresholds for p-value (no filtration and 0.2) and then run 5 times repeated 10-fold cross-validation on the data set. Finally, each data set with each of the combinations results in 50 performance metrics for the model and one value for the correlation between the estimated effect size and the true effect size. We used two different datasets as the initial datasets. One has 1000 samples, and 1000 features with 10 informative features, and the other have 1000 samples and 2000 features with 20 informative features.

We can see that in **Fig. 3**, the performance of the models stays in the same range, and preprocessing methods do not affect the predictive performances. However, if we do not cluster the colinear features together, we can see that the ability of the models to estimate the true effect sizes decrease significantly (**Fig. 3a-b, e-f**). The complete code for simulation and visualization of the results is available at https://github.com/omicsEye/deepbreaks/simulation.

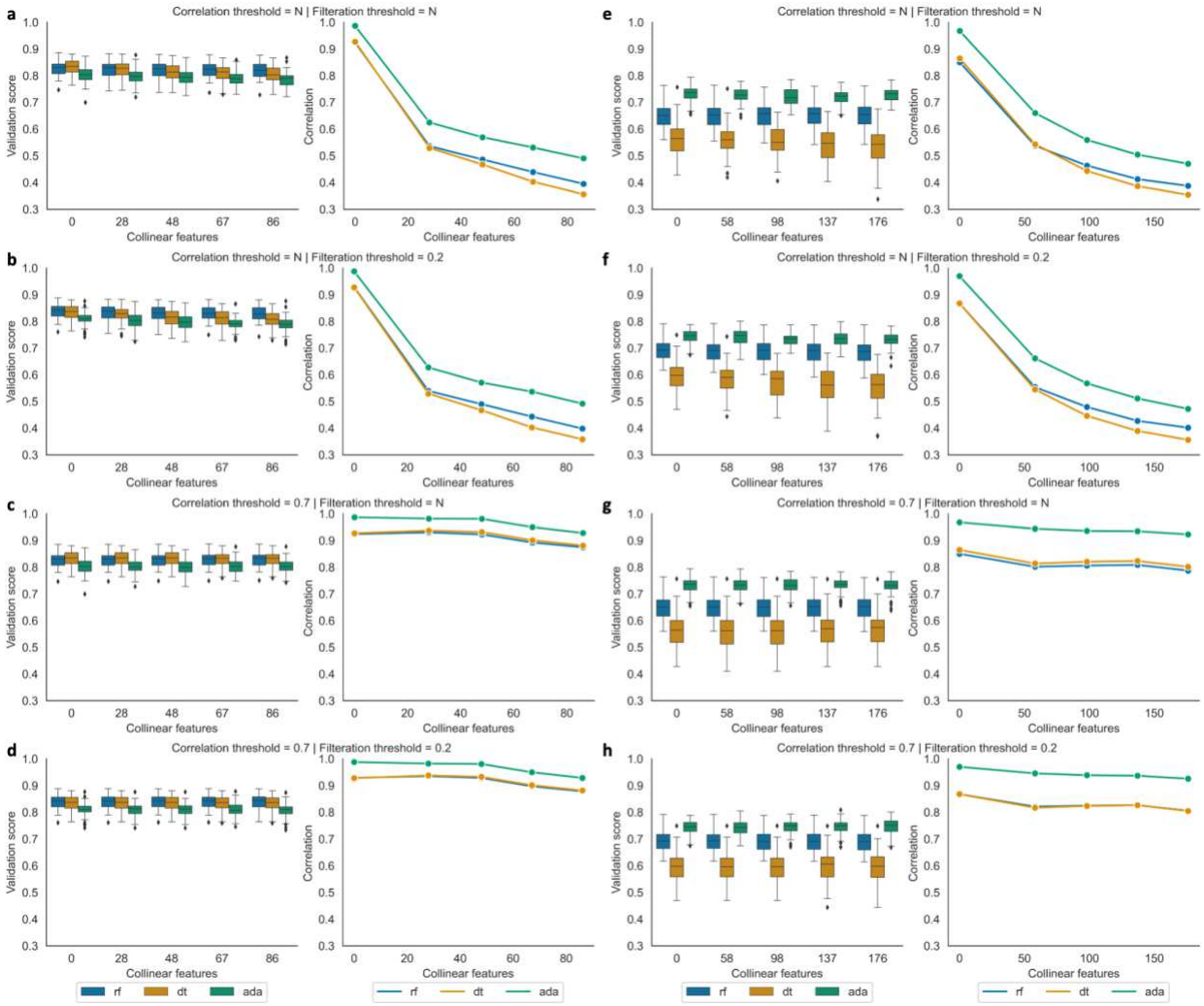

Figure 3: **A simulation study. a-d** results of simulation on a dataset with 1000 samples, 1000 features with 10 informative features. **e-h** results of simulation on a dataset with 1000 samples, 2000 features with 20 informative features. boxplots show the validation scores of each model with different levels of colinearity (x-axis) and line plots show the correlation between the true effect sizes and the predicted effect sizes by models. colinearity (x-axis) zero means the starting dataset. **a,e**, using no clustering and no filtration. **b,f**, using no clustering but filtering redundant features with a p-value of 0.2. **c,g**, using clustering with a threshold of correlation 0.7 and no filtration. **d,h**, using clustering with a threshold of correlation of 0.7 and filtering redundant features with a p-value of 0.2.

deepBreaks identifies amino acids associated with color sensitivity

Opsins are genes involved in light sensitivity and vision, and when coupled with a light-reactive chromophore, the absorbance of the resulting photopigment dictates physiological phenotypes like color sensitivity. We analyzed the amino acid sequence of rod opsins because previously published mutagenesis work established mechanistic connections between 12 specific amino acid sites and phenotypes[19]. Therefore, we hypothesized that machine learning approaches could predict known associations between amino acid sites and absorbance phenotypes. We identified opsins expressed in rod cells of vertebrates (mainly marine fishes) with absorption spectra measurements (λ_{max}, the wavelength with the highest absorption) (Fig. 6a). The dataset

contains 175 samples of opsin sequences including samples with experimental mutations. We next applied deepBreaks on this dataset to find the most important sites contributing to the variations of λ_{max}. We found sites 37, 39, 50, 83, 124, 127, 137, 158, 165, 173, 225, 261, 292, and 299 to be important in terms of affecting the λ_{max} (Fig. 6b). Some of these sites are known from published mutagenesis experiments[19] to strongly affect λ_{max} (**Fig. 4d**). **Fig. 4c** illustrates the effects of mutations in positions 261 and 292 of the sequences.

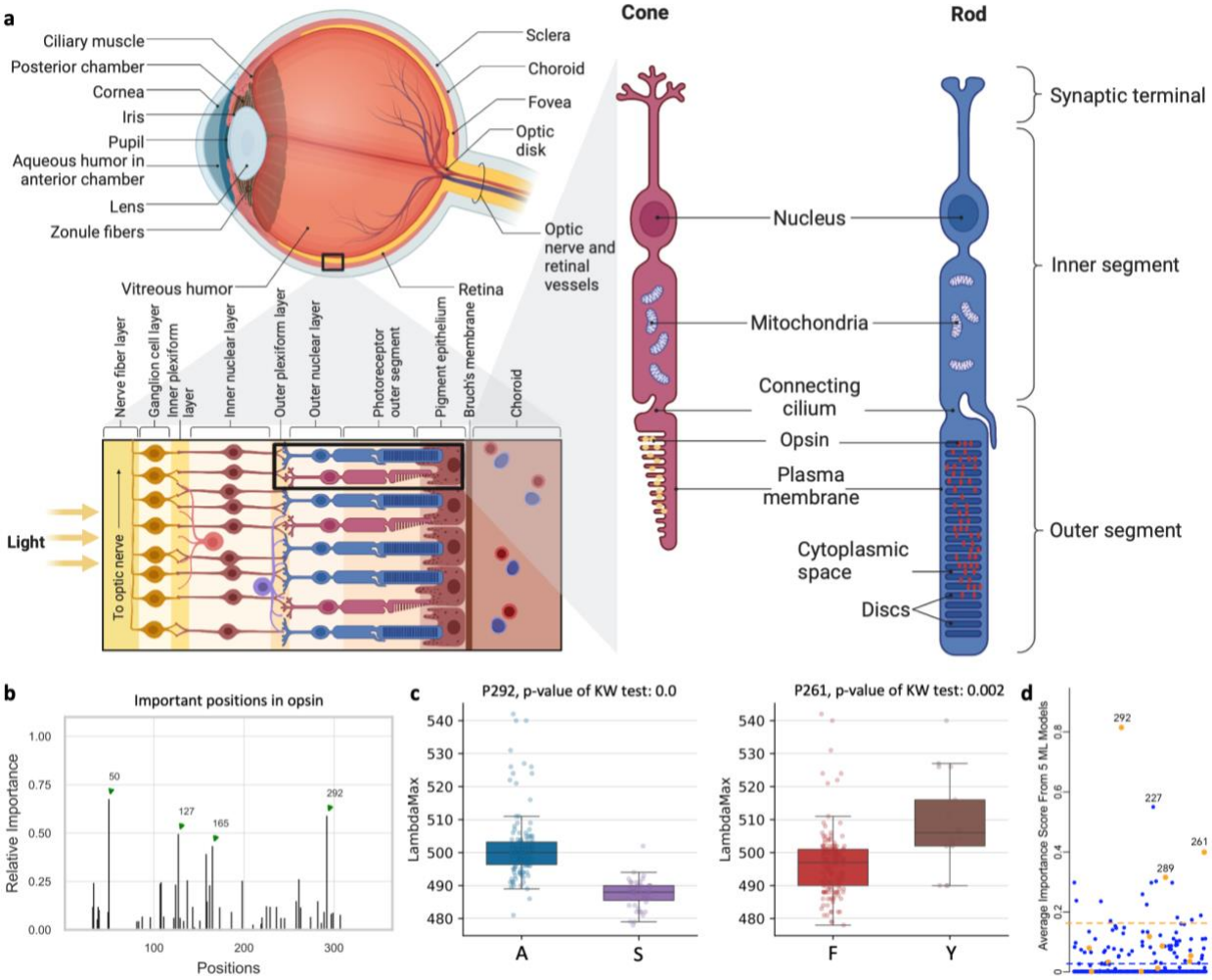

Figure 4: Anatomy of the eye involved in light sensing. a, the overall view from a vertebrate eye, structure of the retina, and position of opsins in rod and cone cells. **b**, important positions in opsin amino acid sequences. **c**, functional changes in positions 292 and 50 as exemplary. **d**, the importance score of experimentally validated positions, yellow color points, and novel positions below points above the importance score threshold colored as a dashed line.

deepBreaks identifies HIV regions with potentially important functions

Subtypes of the human immunodeficiency virus type 1 (HIV-1) group M are different in the envelope (Env) glycoproteins of the virus. These parts of the virus are displayed on the surface of the virion and are targets for both neutralizing antibody and cell-mediated immune responses[20]. The third hypervariable domain (V3) of HIV-1 *gp120* is a cysteine-bounded loop structure usually composed of 105 nucleotides and labeled as the base (nu 1:26 and 75:105), stem (nu 27:44 and

54:74), and turn (nu 45:53) regions[20]. Among all of the hyper-variable regions in *gp120* (V1-V5), V3 is playing the main role in the virus infectivity[21]. Here we use *deepBreaks* to identify regions in the V3 loop that are important in terms of associating the V3 sequences to subtypes B and C. We used the Los Alamos HIV Database[22] (www.hiv.lanl.gov) to gather the nucleotide sequences of the V3 loop of subtypes B and C. We then dropped the repeated samples from the same patients and the final dataset contained 35,424 sequences with a combination of 24,042 (67.87%) sequences of subtype B and 11,382 (32.13%) sequences of subtype C. The maximum length of the sequences was 105 nucleotides. Three distinct communities (clusters) with potentially different biological functions were detected using the *omeClust*[23] based (a zoom-out approach) on V3 loop distances between samples suggesting there are variations in the V3 loop with potential functions(**Fig. 5a**). *deepBreaks* is a zoom-in approach to identifying important mutations (**Fig. 5b-d**). The most important changes are in the stem and turn parts of the V3 loop (**Fig. 5c**). Having a T in position 39 is more prevalent in subtype B, and subtype C mostly has a G in this position (**Fig. 5d**).

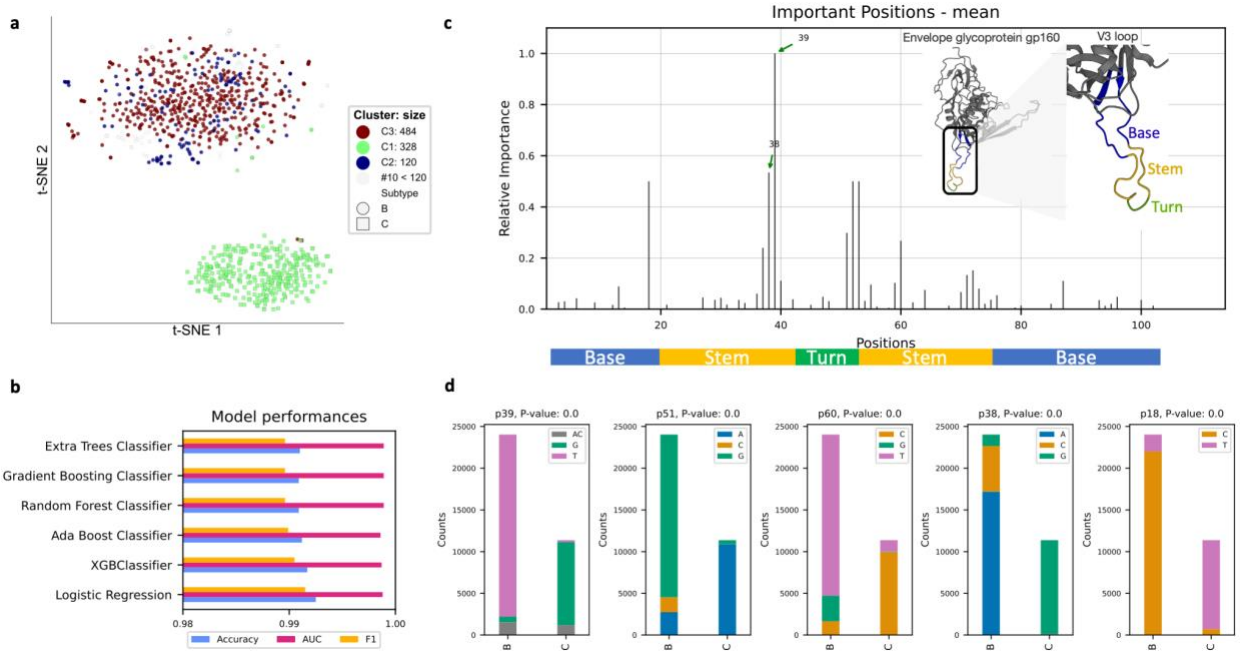

Figure 5: Classification of HIV-1 subtypes B and C based on nucleotide sequences of the V3 loop. **a**, cluster analysis of the sequences with ground truth labels from the Los Alamos National Lab database. **b**, results of 10-fold cross-validation of top 5 classification models, trained to predict the subtypes of the HIV-1 based on the V3 loop. **c**, important positions reported by *deepBreaks* based on the results of the top three models labeled with the sections of the sequence. **d**, stacked bar plots of the top 5 positions that contribute to the classification models.

Novel insights of niche associations in the oral microbiome

Microbial species tend to adapt at the genome level to the niche in which they live. We hypothesize that genes with essential functions change based on where microbial species live. Here we use microbial strain representatives from stool metagenomics data of healthy adults from the Human Microbiome Project[24]. Each microbial strain representative is a concatenation of marker genes using the StrainPhlAn tool[25]. The input for *deepBreaks* consists of 1) an MSA file

with 1006 rows, each a representative strain of a specific microbial species, here *Haemophilus parainfluenzae*, with 49839 lengths of only marker genes used by strainPhlAn; and 2) labels for *deepBreaks* prediction are body sites from which samples were collected: buccal mucosa, supragingival plaque, and tongue dorsum. *deepBreaks* predicts an influential mutation at location 2282 in the PARA_08970 gene (**Fig. 6a**). Location wise buccal mucosa, and supragingival plaque are closer[26] and have similar mutation rates compared to tongue dorsum (**Fig. 6b**). The mutation is located in the PARA_08970 gene, membrane-spanning protein in TonB-ExbB-ExbD complex, which can have a function related to *Haemophilus parainfluenzae* clades in different oral sites. This suggests environmental conditions such as pH and temperature in oral sites cause microbial species mutation to adapt to the niche they live.

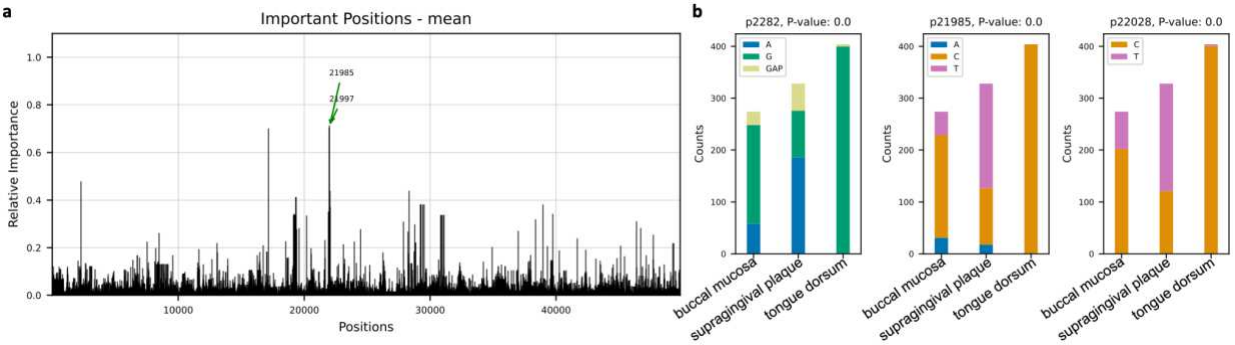

Figure 6: Mutations in *Haemophilus parainfluenzae* from human oral systems are associated with sampling sites. a, aggregated importance of positions across the top three models. **b**, stacked barplot of two top positions showing the frequency of nucleotides in each position for different niches.

deepBreaks reveals important SARS-CoV-2 regions associated with Alpha and Delta variants

Variants occur with new mutations in the virus genome. Most mutations in the SARS-CoV-2 genome do not affect the functioning of the virus. However, mutations in the spike protein of SARS-CoV-2, which binds to receptors on cells lining the inside of the human nose, may make the virus easier to spread or affect how well vaccines protect people. Other mutations may lead to SARS-CoV-2 being less responsive to treatments for COVID-19[27]. Variants of SARS-CoV-2 have been categorized into multiple variants, but based on their effect on public health and five of these — Alpha, Beta, Delta, Gamma, and Omicron — have been labeled as variants of concern and associated with enhanced transmissibility and increased virulence[28,29]. We used the publicly available data from GSAID[30] and obtained 10,000 sequences of spike protein region for SARS-CoV-2 samples of the Alpha variant — one of the first variants of concern identified by the WHO — and 10,000 sequences of the spike protein region for SARS-CoV-2 samples of the Omicron variant — one of the newest variants of concern identified by the WHO. We used MAFFT algorithm[31] with PAM 200[32] to align these sequences. The final data set after dropping the replicates was consist of 9863 sequences of the Alpha variant and 9618 sequences of Omicron variant (19481 total). Then, we used *deepBreaks* to analyze the data and find the most important (predictive) positions in these sequences in terms of classifying the variants (**Fig. 7a**). The mutations in this part of the sequence were highly correlated and happened almost

concurrently. We have shown 6 of the positions with mutations in these sequences and their detailed changes in **Fig. 7b**.

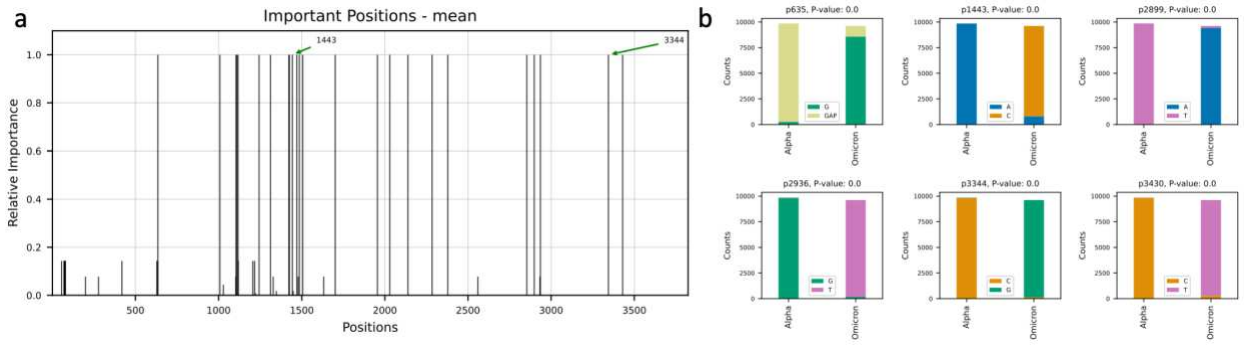

Figure 7: Classifying the SARS-CoV-2 variants based on the spike protein sequences. a, important positions in the spike protein (S) of SARS-CoV-2 in terms of predicting Alpha and Omicron variants. **b**, details of how the mutations appear in variants.

Discussion

In this study, we provided an integrated generic approach to find the most discriminative changes in a sequence in association with a given phenotype of interest. Our approach is based on first training accurate machine learning models and then using their information to interpret the predictive power of each position in the sequence data. However, having an accurate predictive model for sequence-to-phenotype studies is a challenging task. One of the major challenges in training an accurate model is rooted in the high-dimensional MSA files with lengthy sequences and a limited number of samples, known as the curse of dimensionality[33]. The other tremendous challenge for training and interpreting the models is the colinearity between positions of an MSA file which have a negative effect on the performance of the models[18]. We showed that by implementing multiple filtration methods in the data preprocessing step, *deepBreaks* not only finds the most accurate model based on the given data but also allows for the interpretation of the trained models. To justify our approaches in terms of dealing with the redundancy of features and multicollinearity, we also conducted a simulation study on multiple datasets with different levels of colinear features. The results of these simulation studies showed that our method not only helps in reducing the dimensionality of the data by clustering the colinear features and dropping the redundant features but also assists the models in being able to estimate the importance of features with the presence of different levels of multicollinearity.

We also evaluated the performance of *deepBreaks* on real data with 5 different datasets. *deepBreaks* pointed out the important positions in the sequences of each of the data sets which have been mentioned in the literature without any prior knowledge of the problem and only based on the provided samples. Moreover, in each study, important positions were reported that have not been mentioned in the literature before, opening new topics for further research. Finally, by applying *deepBreaks* in different scenarios ranging from predicting a continuous phenotype, such as light sensitivity with amino acid sequences of opsins, to categorical phenotypes, such as different niches of *Haemophilus parainfluenzae* based on its genome sequence, and finding significant results, we showed its wide applicability. To facilitate the

usage of this tool, we have provided it as an open-source python library with various sample codes and tutorials for both installation and usage at https://github.com/omicsEye/deepbreaks.

Data Availability

All of the data used in this study and the Jupyter notebooks that were used to produce the results are available at https://github.com/omicsEye/deepbreaks.

Acknowledgments

This work was supported by the National Science Foundation grant DEB-2028280 and DEB-2109688 to AR and KAC and IOS-1754770 to THO.

Author contributions

A.R., M.B., and K.A.C. conceived the method; M.B. implemented the software; M.B., B.S., and T.D. tested and packaged the software and evaluated the performance; M.B. and A.R. provided online documents and software. M.B., A.R., B.S., T.D., and T.H.O. analyzed application datasets. M.B., A.R., B.S., T.D., T.H.O., and K.A.C. drafted the manuscript. All authors discussed the results and commented on the paper.

Competing interests

The authors declare no competing financial interests.

Methods

Approach

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks[34]. Supervised learning, which is a branch of ML, aims to find a function f that maps input data to output variable y through a training set of $t = \{(X_1, p_1), (X_2, p_2), \ldots, (X_n, p_n)\}$. A supervised learning algorithm takes X_i as input and produces $f(X_i)$ as an estimation for the p_i. Supervised learning algorithms are designed to enhance their performance by minimizing the distance $||f(X_i) - p_i||$ [35]. In our case, $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, $i \in \{1,2,\ldots,n\}$ are sequences of length m (m nucleotides or amino-acids) and p_is are phenotypes related to the i^{th} sequence. For example x_{ij} (the j^{th} element of the sample i) can be a subset of $\{A, T/U, C, G\}$, or any amino acids. Phenotypes (p_is) can also be a continuous measure such as height, BMI, or categories such as obese/not obese, antibacterial resistance, or mild/severe cases. Assuming that raw data (sequences and phenotype) are provided, actions are still required to prepare the data for machine learning algorithms. *deepBreaks* has three phases: i) preprocessing the data, ii) fitting models to preprocessed data and comparing them, iii) interpreting the results of top predictor(s) by reporting tables and visualizations.

Preprocessing

Data preprocessing is a fundamental step for any ML algorithm. Sequence data may contain missing values, ambiguous reads, zero-entropy columns, correlated positions (features), and redundant features which do not carry a significant amount of information in association with the phenotype under study. The *deepBreaks* pipeline for preprocessing starts with dropping columns in the dataset that contain missing values over a certain threshold. The default threshold is 80% of the number of samples. So, if we have 1000 samples while preserving the position names, we drop all the positions that have over 800 missing values from the training set (this value can be changed based on the user preference). Dropping the zero-entropy (constant) features from the dataset is the next step. It is worth mentioning that before starting to drop the columns, all of them (by default) are named from p_1 to p_m. The names of the columns (positions) in the dataset are fixed, and dropping certain columns does not change the position names in a sequence. For the remaining missing values, if the number of missing values are above a certain threshold (default 15%) we impute them by the term 'GAP', and if they are below that threshold we use the mode (most frequent) of reads in each position to impute the missing values. Some columns may also have ultra-rare cases which have a share of below 2% of the reads in a position. These values are also replaced by the mode of that column. After this step, as we modified the reads, positions are again checked for entropy, and positions with zero entropy will be dropped from the training set. The next step is to perform chi-square (categorical phenotype) or Kruskal-Wallis (continuous phenotype) tests to reduce the number of positions in the training data set and drop the redundant ones. We use these statistical tests to assess the significance of each position by running tests on all the positions against the phenotype one by one. Those features where the p-value of their test against the phenotype is less than a threshold[36] (default p-value = 0.25) will be dropped. A list of all features and their test p-values will be provided as a report to the user. As we consider each position in the sequences as a feature of our training dataset, we need to check for colinearity between our predictive variables, as it can cause issues for parameter estimation[18]. To check for the relationship between positions, we first one-hot encode the features and drop one feature of each position to avoid adding colinear features to the training dataset. Then, we use a distance function that calculates the pairwise distances between features. The list of available metrics are spearman correlation[37], hamming[38], jaccard[39], normalized mutual information[40], adjusted mutual information[41], and adjusted Rand score[42]. The result of this step is a symmetric distance matrix with values between 0 (being exactly the same) to 1 (uncorrelated). We then use this symmetric matrix of distance values and feed it into the density-based spatial clustering of applications with noise (DBSCAN) method[17] for clustering the features based on their pairwise distances. The reason behind this is to cluster the features that provide the same information[18]. After that, we select one feature from each cluster as the representative of that cluster and drop the rest of the features in that cluster from the training set. Although we drop the rest of the features in each cluster except the representative, we keep the information of the members of the clusters for interpretation after the modeling step. The default parameters of the DBSCAN are epsilon (distance between centers) equal to 0.2 and the minimum points for a cluster are equal to 2.

Models

We use different sets of models for continuous or categorical phenotypes. For continuous phenotypes, we fit linear regression, Ridge Regression[43], Lasso Regression[44], Bayesian Regression, Lasso Least Angle Regression[45], Huber Regressor [46], Extremely Randomized Trees (Extra Trees)[47], Extreme Gradient Boosting (xgboost)[48], Light Gradient Boosting Machine (lightgbm)[49], Random Forest[50], Decision Tree[51], and AdaBoost[52]. For problems with a categorical phenotype, we use Extra Trees, xgboost, lightgbm, Random Forest, Decision Tree, AdaBoost, Gradient Boosting, and Logistic Regression. For all of the above-mentioned models, we use the default hyperparameters from the scikit-learn library in python[53] and a grid search (expandable by user preference) parameter set that is provided in the documentation. For model comparison, *deepBreaks* by default uses a 10-fold cross-validation approach and ranks the models based on their average cross-validation score. K-fold cross-validation is a resampling method that partitions the whole dataset into k separate equal-size parts and then uses k-1 parts for training the model and 1 part for testing the performance. This process is repeated k times, and the average score of all k models is called the cross-validation score (**SFig. 2**).

The default performance metrics for regression and classification that *deepBreaks* uses are Mean Absolute Error (MAE) and F-score.

$$MAE = \frac{\sum_{i=1}^{n} |f(X_i) - p_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}, F - score = \frac{2}{recall^{-1} + precision^{-1}}$$

The default list of metrics that deepBreaks reports are provided in the documentation and the user and provide predefined custom metrics or a set o metrics from the scikit-learn library in python[53].

Interpretation

For interpreting the contribution of sequence positions to the predictive models, we use the feature importance, coefficients, and weights as different algorithms have different kinds of output. For xgboost[54] and lightgbm[55] the reported feature importance represents the number of times a feature appears in a tree. For AdaBoost, random forest, decision tree, extra tree, and gradient boosting the importance of a feature is its Gini importance which is computed as the normalized total reduction of the criterion brought by that feature[53].

If we have N samples that reaches the node j of a tree and G be the impurity of the node j, the importance of node j, ion_j is calculated as follows:

$$ion_j = N_j G_j - N_{left\ child\ node(j)} G_{left\ child\ node(j)} - N_{right\ child\ node(j)} G_{right\ child\ node(j)}$$

Based on this, the feature importance value of the i^{th} feature is:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ion_j}{\sum_{k \in all\ nodes} ion_k}$$

And then we can normalize the feature importance value for the i^{th} feature by dividing it with the sum of the importance of all the features:

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j}$$

These calculations are just based on one tree and in other tree-based algorithms such as random forest, AdaBoost, and extra tree, the final importance of a feature is its average over all of the fitted trees. For the linear models, the regression coefficients or weight is considered as the feature importance[53].
In the preprocessing phase, we one-hot encoded the positions into training features, so, each position (based on the number of its unique characters), is transformed into one or more than one feature. For example, if position pi consists of $\{A, T, C\}$, its features are pi_A and pi_T in the form of one-hot encoded features (we drop pi_C to avoid colinearity). These features have separate importances and we sum the absolute value of their importance and the importance of position pi is the sum of importances of the feature pi_A and pi_T. After that, to normalize the importances, all the importances will be divided by their maximum value. We consider zero importance for all the positions that have been dropped during the preprocessing steps. For all those positions that were in the same group and dropped based on their distance values and DBSCAN clustering, we consider the same feature importance value. *deepBreaks* by default considers the top three best-fitted models, but the user can change this number. We then calculate the average importance value for each feature over the top selected models. *deepBreaks* creates reports of the models and merged results separately.

deepBreaks Output

deepBreaks creates reports of 1) p-values from statistical tests for each position against a phenotype, 2) the related distance matrix, 3) clusters of correlated positions, 4) a table of fitted models with their performance metrics, 5) feature importance values for each of the top models and their merged results, 6) plots for importance values based on individual models and merged results 7) box plot (continuous phenotype) or stacked bar plot (categorical phenotype) for most discriminative positions.
In the data preprocessing we use NumPy[56], Pandas[57] and SciPy[58] python libraries. For model comparisons and cross-validation pipeline, we use scikit-learn[53], xgboost [48], and lightgbm[49]. Visualizations generated by *deepBreaks* make use of the seaborn[59] and matplotlib[60] libraries in python. We used *deepBreaks 1.0.1* for all the applications and evaluations in this manuscript.

References

1. Ritchie, M. D., Holzinger, E. R., Li, R., Pendergrass, S. A. & Kim, D. Methods of integrating data to uncover genotype–phenotype interactions. *Nat. Rev. Genet.* **16**, 85–97 (2015).
2. Moore, J. H., Asselbergs, F. W. & Williams, S. M. Bioinformatics challenges for genome-wide association studies. *Bioinformatics* **26**, 445–455 (2010).
3. Doshi-Velez, F. & Kim, B. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv [stat.ML]* (2017).
4. Leung, M. K. K., Delong, A., Alipanahi, B. & Frey, B. J. Machine Learning in Genomic

Medicine: A Review of Computational Problems and Data Sets. *Proc. IEEE* **104**, 176–197 (2016).

5. Yang, Y. *et al.* Machine learning for classifying tuberculosis drug-resistance from DNA sequencing data. *Bioinformatics* **34**, 1666–1671 (2018).
6. Hadikurniawati, W., Anwar, M. T., Marlina, D. & Kusumo, H. Predicting tuberculosis drug resistance using machine learning based on DNA sequencing data. *J. Phys. Conf. Ser.* **1869**, 012093 (2021).
7. Adam, G. *et al.* Machine learning approaches to drug response prediction: challenges and recent progress. *NPJ Precis Oncol* **4**, 19 (2020).
8. Wan, N. *et al.* Machine learning enables detection of early-stage colorectal cancer by whole-genome sequencing of plasma cell-free DNA. *BMC Cancer* **19**, 832 (2019).
9. Kurian, B. & Jyothi, V. L. Breast cancer prediction using an optimal machine learning technique for next generation sequences. *Concurrent Eng.: Res. Appl.* **29**, 49–57 (2021).
10. Lee, S. H., van der Werf, J. H. J., Hayes, B. J., Goddard, M. E. & Visscher, P. M. Predicting unobserved phenotypes for complex traits from whole-genome SNP data. *PLoS Genet.* **4**, e1000231 (2008).
11. Guzzetta, G., Jurman, G. & Furlanello, C. A machine learning pipeline for quantitative phenotype prediction from genotype data. *BMC Bioinformatics* **11 Suppl 8**, S3 (2010).
12. Drouin, A. *et al.* Predictive computational phenotyping and biomarker discovery using reference-free genome comparisons. *BMC Genomics* **17**, 754 (2016).
13. Montesinos-López, A., Montesinos-López, O. A., Gianola, D., Crossa, J. & Hernández-Suárez, C. M. Multi-environment Genomic Prediction of Plant Traits Using Deep Learners With Dense Architecture. *G3* **8**, 3813–3828 (2018).
14. Ma, W. *et al.* A deep convolutional neural network approach for predicting phenotypes from genotypes. *Planta* **248**, 1307–1318 (2018).
15. Liu, Y. *et al.* Phenotype Prediction and Genome-Wide Association Study Using Deep

Convolutional Neural Network of Soybean. *Front. Genet.* **10**, 1091 (2019).

16. Lee, Y.-C. *et al.* Using Machine Learning to Predict Obesity Based on Genome-Wide and Epigenome-Wide Gene-Gene and Gene-Diet Interactions. *Front. Genet.* **12**, 783845 (2021).
17. A Density-Based Algorithm for Discovering Clusters in Large. *https://www.aaai.org › KDD › 1996 › KDD96-037https://www.aaai.org › KDD › 1996 › KDD96-037*.
18. Dormann, C. F. *et al.* Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography* **36**, 27–46 (2013).
19. Yokoyama, S., Tada, T., Zhang, H. & Britt, L. Elucidation of phenotypic adaptations: Molecular analyses of dim-light vision proteins in vertebrates. *Proc. Natl. Acad. Sci. U. S. A.* **105**, 13480–13485 (2008).
20. Lynch, R. M., Shen, T., Gnanakaran, S. & Derdeyn, C. A. Appreciating HIV type 1 diversity: subtype differences in Env. *AIDS Res. Hum. Retroviruses* **25**, 237–248 (2009).
21. Felsövályi, K., Nádas, A., Zolla-Pazner, S. & Cardozo, T. Distinct sequence patterns characterize the V3 region of HIV type 1 gp120 from subtypes A and C. *AIDS Res. Hum. Retroviruses* **22**, 703–708 (2006).
22. Compendium, H. Foley B, LT, Apetrei C, Hahn B, Mizrachi I, Mullins J, Rambaut A, Wolinsky S & Korber B, Eds. *Biophysics Group, Los Alamos National Laboratory …*.
23. Rahnavard, A. *et al.* Omics community detection using multi-resolution clustering. *Bioinformatics* **37**, 3588–3594 (2021).
24. Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature* **486**, 207–214 (2012).
25. Truong, D. T., Tett, A., Pasolli, E., Huttenhower, C. & Segata, N. Microbial strain-level population structure and genetic diversity from metagenomes. *Genome Res.* **27**, 626–638 (2017).
26. Lloyd-Price, J. *et al.* Strains, functions and dynamics in the expanded Human Microbiome

Project. *Nature* **550**, 61–66 (2017).

27. Lauring, A. S. & Malani, P. N. Variants of SARS-CoV-2. *JAMA* (2021) doi:10.1001/jama.2021.14181.
28. Aleem, A., Akbar Samad, A. B. & Slenker, A. K. Emerging Variants of SARS-CoV-2 And Novel Therapeutics Against Coronavirus (COVID-19). in *StatPearls* (StatPearls Publishing, 2022).
29. Tracking SARS-CoV-2 variants. https://www.who.int/activities/tracking-SARS-CoV-2-variants.
30. Khare, S. *et al.* GISAID's Role in Pandemic Response. *China CDC Wkly* **3**, 1049–1051 (2021).
31. Katoh, K., Rozewicki, J. & Yamada, K. D. MAFFT online service: multiple sequence alignment, interactive sequence choice and visualization. *Brief. Bioinform.* **20**, 1160–1166 (2017).
32. Dayhoff, M., Schwartz, R. & Orcutt, B. 22 a model of evolutionary change in proteins. *Atlas of protein sequence and structure* **5**, 345–352 (1978).
33. Clarke, R. *et al.* The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nat. Rev. Cancer* **8**, 37–49 (2008).
34. Hierons, R. Machine learning. Tom M. Mitchell. Published by McGraw-Hill, Maidenhead, U.K., International Student Edition, 1997. ISBN: 0-07-115467-1, 414 pages. Price: U.K. £22.99, soft cover. *Software Testing, Verification and Reliability* vol. 9 191–193 Preprint at https://doi.org/10.1002/(sici)1099-1689(199909)9:3<191::aid-stvr184>3.0.co;2-e (1999).
35. Nordhausen, K. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition by Trevor Hastie, Robert Tibshirani, Jerome Friedman. *International Statistical Review* vol. 77 482–482 Preprint at https://doi.org/10.1111/j.1751-5823.2009.00095_18.x (2009).
36. Hosmer, D. W., Jr, Lemeshow, S. & Sturdivant, R. X. *Applied Logistic Regression*. (John

Wiley & Sons, 2013).

37. Coefficient, S. R. C. In The Concise Encyclopedia of Statistics. Preprint at (2008).
38. Hamming Distance. in *Encyclopedia of Biometrics* (eds. Li, S. Z. & Jain, A.) 668–668 (Springer US, 2009).
39. Hancock, J. M. Jaccard Distance (Jaccard Index, Jaccard Similarity Coefficient). in *Dictionary of Bioinformatics and Computational Biology* (2014).
40. Kvalseth, T. O. Entropy and Correlation: Some Comments. *IEEE Trans. Syst. Man Cybern.* **17**, 517–519 (1987).
41. Vinh, N. X., Epps, J. & Bailey, J. Information theoretic measures for clusterings comparison. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* Preprint at https://doi.org/10.1145/1553374.1553511 (2009).
42. Morey, L. C. & Agresti, A. The Measurement of Classification Agreement: An Adjustment to the Rand Statistic for Chance Agreement. *Educ. Psychol. Meas.* **44**, 33–37 (1984).
43. Hoerl, A. E. & Kennard, R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **12**, 55–67 (1970).
44. Santosa, F. & Symes, W. W. Linear Inversion of Band-Limited Reflection Seismograms. *SIAM J. Sci. and Stat. Comput.* **7**, 1307–1330 (1986).
45. Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. Least angle regression. *aos* **32**, 407–499 (2004).
46. Huber, P. J. & Ronchetti, E. M. Robust Statistics. *Wiley Series in Probability and Statistics* Preprint at https://doi.org/10.1002/9780470434697 (2009).
47. Geurts, P., Ernst, D. & Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **63**, 3–42 (2006).
48. Chen, T. & Guestrin, C. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* Preprint at https://doi.org/10.1145/2939672.2939785 (2016).

49. Ke, Meng, Finley & Wang. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.*
50. Breiman, L. Random Forests. *Mach. Learn.* **45**, 5–32 (2001).
51. Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. *Classification and Regression Trees*. (Chapman & Hall/CRC, 2017).
52. Freund, Y. & Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. System Sci.* **55**, 119–139 (1997).
53. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **12**, 2825–2830 (2011).
54. *xgboost: Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM) Library, for Python, R, Java, Scala, C++ and more. Runs on single machine, Hadoop, Spark, Dask, Flink and DataFlow*. (Github).
55. *LightGBM: A fast, distributed, high performance gradient boosting (GBT, GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks*. (Github).
56. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357–362 (2020).
57. McKinney, W. Data Structures for Statistical Computing in Python. in *Proceedings of the 9th Python in Science Conference* (SciPy, 2010). doi:10.25080/majora-92bf1922-00a.
58. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (2020).
59. Waskom, M. seaborn: statistical data visualization. *J. Open Source Softw.* **6**, 3021 (2021).
60. Hunter. Matplotlib: A 2D Graphics Environment. **9**, 90–95 (2007).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- deepBreaksSupplement.pdf