

# TAD detection with machine learning classifiers

Albert Lidel

March 23, 2021

## 1 Introduction

Within mammalian organisms, the genomic material in form of DNA can be divided into functional compartments called genes, which are the basic hereditary unit. Those genes can be transcribed into RNA and later translated into proteins [5][9]. The hereditary material of an organism can be understood as the set of genes, which is read and analyzed with common DNA-sequencing methods [11]. The process of transcribing genes is regulated, hence the analysis of DNA as a set of genes, is not enough, to get the full picture of how the organism will develop with a given genome [9]. Which genes are transcribed at a given point and time is called gene expression [9]. Determining gene expression is an important and ongoing research topic in the field of Molecular Biology. This is, because it allows for better understanding of how organisms develop certain traits and how cells specialize into different functionality [3]. It also helps in understanding the origin of certain diseases [3][9]. One aspect of how the cell regulates its expression of genes is by the spatial formation of its genetic material [3]. Chromosomes will entangle both within themselves, as well as with each other, forming a complex spatial structure within the nucleus. This is accomplished with protein-DNA complexes, which uses proteins, the so called histones, to bind two positions within the DNA strands together. This complex is called chromatin. It serves multiple purposes, both during cell division and outside of it. Those include reinforcement of DNA, gene expression and dense packaging. At the smallest scale, DNA is packaged within a nucleosome. Those nucleosomes can form higher structures. Other proteins, such as cohesin and CTCF also play a role. The aforementioned sequencing methods are oblivious to those structures. For this reason, sequencing methods for capturing the spatial structure of the genome have been developed. The set of those methods is called Chromosome Conformation Capture. While they share common steps in their pipeline, the goal of these and their scopes vary [14]. We will look at Hi-C, which belongs to this set of methods. The pipeline of Hi-C crosslinks DNA at a location of close spatial proximity of two strings. The DNA will then be sheared at this location, ligated together and then sequenced. This read is then called a di-tag or sometimes chimeric read, since it is build from two fragments

which were fused together because of close proximity and are not necessarily adjacent in the original DNA string [29]. Hi-C, as all Chromosome Conformation Capture methods, is a high-throughput approach [3]. Di-tags sequenced via Hi-C can be mapped against a reference genome to link those tags to their two specific locations [29]. With this information a so called Hi-C Matrix or contact matrix can be constructed. In this matrix, one data point corresponds to the number of interactions between two locations on the reference genome. The data is put together into bins of adjacent locations, to reduce the size of the matrix [26]. DNA sequenced with Hi-C allows for analysis of a spatial structure called a topological associated domain, in short TAD. TADs are formed mainly by CTCF, but other proteins like cohesin and histones also play a role [3]. It is expected, that they have an impact in gene expression [3]. Interactions are much more common between loci inside a domain, than with outside loci. With this knowledge and a contact matrix with the number of interactions between two loci, TADs and their boundaries can be computed by a statistical approach [26]. A boundary of a TAD is defined as either the start or end gene position of a domain [26]. Numerous programs with different approaches for identifying boundaries and their respective TADs have been proposed [33]. These programs are often referred to as TAD-Callers [33]. The types of detected TADs can be either disjunctive, overlapping or hierarchical [33]. One of those programs is hicFindTads, an algorithm which uses a sliding window of varying sizes. It computes a score based on the number of interactions for each bin and then searches for local minima to place disjunctive TAD boundaries [26].

## 2 Task

In this work, a Machine Learning approach for TAD-Calling will be implemented. To this end, existing Machine Learning Models will be used to predict TAD boundaries from an input Hi-C Matrix. The existing models will be classifiers from the sklearn library in the python programming language. Different approaches and classifiers for this problem will be tested and a few preprocessing steps to prepare and balance the Hi-C input data will be integrated. The classification problem will consist of two classes, namely "boundary" and "non-boundary". If a boundary is the start or end of a TAD will not be distinguished and if the resulting TADs will be disjunctive, overlapping or hierarchical might be dependent on the training set. At first, the behaviour of the hicFindTads TAD-Caller will be tried to emulate. Later, the quality of the prediction against a CTCF protein track will be tried to be maximized and will also be tested against the outcome of hicFindTads. A few other TAD-Callers will also be tested.

## 3 Methods

### 3.1 Hi-C Matrix

As an input for our pipeline Hi-C contact matrices were used. Those contain the number of interactions between bin  $i$  and bin  $j$  at position  $(i, j)$ , resulting in a  $n \times n$  matrix,

where  $n$  is the number of bins of the matrix. Typical bin size values range from 1 kilobases to 10 megabases, usually being referred to as the resolution of the matrix. Kilobases and megabases in this context refer to the amount of DNA base pairs, i.e. Adenine-Thymine, Guanine-Cytosine and reverse, in a bin. This matrix has an identity diagonal, where position  $(i, i)$  refers to the number of interactions, a given bin has with itself. The two halves above and below this diagonal are symmetrical to each other, i.e.  $I(i, j) = I(j, i)$ , where  $I(x_1, x_2)$  is the number of interactions at position  $(x_1, x_2)$ . All matrices are sorted by position. Multiple chromosomes can be contained in one matrix, in which case it is grouped by chromosome in addition to being sorted by the position on the chromosomes. The number of interactions in the matrix is a measured value by a Chromosome Conformation Capture technique, and may be affected by artefacts and varying coverage on different regions [31].

### 3.2 Normalization

A Hi-C contact matrix can be normalized to improve the comparability of matrices of different origins and different read coverage [31]. Range normalization, sometimes referred to min-max normalization is one of these techniques. With range normalization, data in a set  $x_{in}^i \in X_{in}$  is scaled to a given interval  $[r_{min}, r_{max}]$ , which can be seen in equation 1.

$$x_{norm}^i = \frac{x_{in}^i - \min(X_{in})}{\max(X_{in}) - \min(X_{in})} * (r_{max} - r_{min}) + r_{min} [21] \quad (1)$$

Another normalization option is obs/exp. With this option, an expected matrix is computed from the observed input. This can be seen in the following equation 2, where  $M$  is the Hi-C Matrix,  $i$  and  $j$  valid indices of this matrix and  $diag(i, j)$  the diagonal between indices  $i$  and  $j$ .

$$exp_{i,j} = \Sigma(diag(i, j)) * \Sigma(row(j)) * \Sigma(row(i)) / \Sigma(M) [31] \quad (2)$$

A list of matrices  $M$  can also be scaled to the matrix of the lowest read coverage. This matrix is defined as the matrix  $m_i$  with the lowest sum of its elements  $s_i$ . All matrices in the list are scaled using a scaling factor from this lowest sum, as seen in equation 3. This is done to make matrices of different origin comparable.

$$S = \{s_i | s_i = \Sigma(m_i), m_i \in M\} \quad (3)$$

$$s_{min} = \min(S) \quad (4)$$

$$m'_i = \frac{m_i}{s_i / s_{min}} \quad (5)$$

### 3.3 Boundary and protein data

As has been previously stated, boundaries are described as the start and end genes of a given TAD. Those genes were expressed as positions on the matrix for a given chromosome. For simplicity, disjunctive boundaries were assumed. The used input data only contained disjunctive data. Protein track files for Hi-C inputs were also included. Specifically CTCF track files were utilized, since a correlation between high amounts of the CTCF protein and boundaries for TADs has been proposed [3]. Information about domains and proteins was not directly stored within our Hi-C Matrices. Instead both were stored in a so called BED file (browser extensible data), which can be understood as a tab-delimited text file [24]. A BED file consists of three required columns and nine optional columns, as can be seen in figure 1 [24]. At least the start and end position and the chromosome, on which this data point resides, must be specified. Some of the used protein files had custom columns for the measured peak value and other information, which were not part of the BED standard, but were instead in the narrowPeak format [12]. An overview of the columns for this format can be seen in table 2. Those protein and domain files were intersected, to filter out low quality boundaries, for which no peak could be found.

Table 1: BED file format [24]

column	description
chrom	chromosome name
chromStart	start position on chromosome
chromEnd	end position on chromosome
name	custom name or .
score	display value
strand	+, - or . for orientation
thickStart	start position for display
thickEnd	end position for display
itemRgb	RGB display value
blockCount	number of sub-elements
blockSizes	size of sub-elements
blockStarts	start of sub-elements

### 3.4 Visualization

A matrix can be visualised with the help of a colourmap. In figure 1, a sample matrix is displayed, with the identity diagonal over the points  $(i, i)$ . For better visualization of TADs, the matrix can be rotated by  $45^\circ$  and the triangle below the identity diagonal be discarded. Then, boundary information can be inserted into the plot and a protein track can be aligned below the matrix. Those boundaries can be visualised disjunctive, like in figure 2 with a triangle shaped bracket. Pyramid-shape structures in the matrix of high interaction count can be made out [31].

Table 2: narrowPeak file format [12]

column	description
chrom	chromosome name
chromStart	start position on chromosome
chromEnd	end position on chromosome
name	custom name or .
score	display darkness
strand	+, - or . for orientation
signalValue	measurement of this region
pValue	statistical significance (p-value)
qValue	statistical significance (q-value)
peak	point source for peak

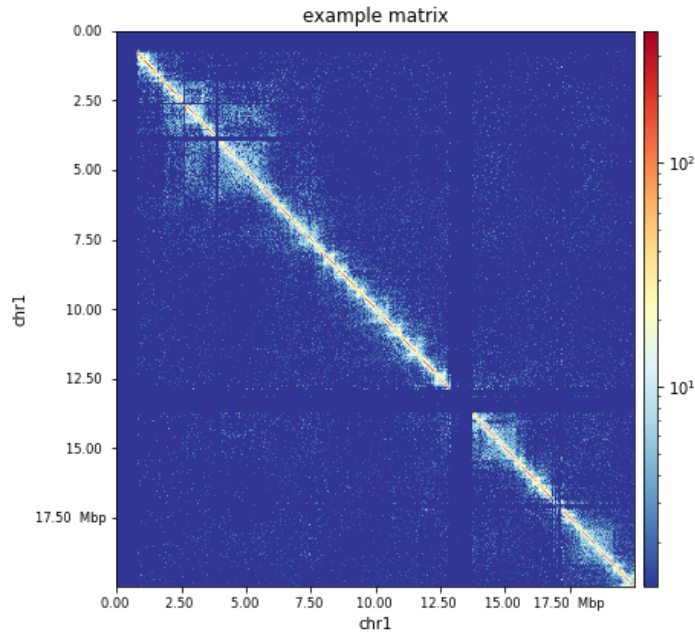


Figure 1: A sample Hi-C Matrix of a human chromosome 2 (high interaction areas bright) [27]

### 3.5 Statistical Classification and Machine Learning

Classification in the field of Statistics is the discrimination of items into different classes with respect to observed features. In this case a procedure was build by training a mathematical model. Which was taught to discriminate between items with preset classes. This is referred to as Supervised Learning. Machine Learning is an umbrella term for procedures that learn from a given set of examples and will later be able to make

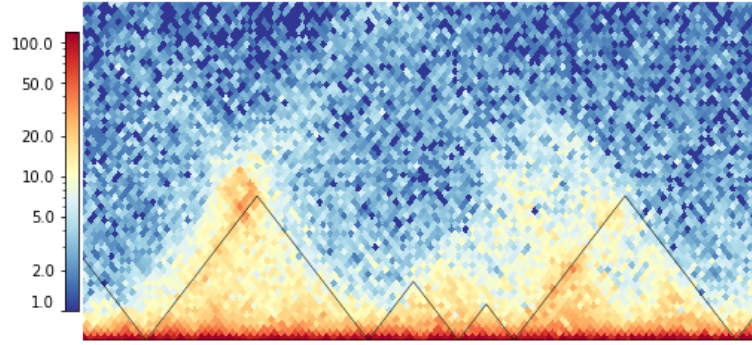


Figure 2: A sample Hi-C Matrix of a human chromosome 1, position 8mb to 9mb (high interaction areas in orange) with boundary information [27]

predictions from its findings [16]. The learned examples are referred to as the training set, whereas a set which will be predicted to validate the model is called the test set [22]. In this work, the pair of training and test set was called input set. Every example in the input set has the same set of attributes, which are referred to as features. Those features are used to describe the example for the classification procedure [16]. Supervised Learning a subset of Machine Learning [16]. Within the scope of this paper, data was discriminated between two classes, namely boundary and non-boundary. A number of different models, also called classifiers, were tried on our data. The main approach in this work was to make use of Ensemble Learning. Ensemble Learning is a general technique, where multiple models are combined to improve performance [10]. Since within the scope of this work it was necessary to train models on multiple datasets, the classifiers needed to be able to be partially fitted [22]. Models can need fine tuning to better perform on the given classification tasks. The parameters for the model and the pre-cleaning steps, which have an impact on classification performance are called hyper-parameters. The performance of different sets of values for these parameters is often tested and the best set is later used for the classifier in production [22].

### 3.5.1 Decision Tree Classifier

As the name suggest, the Decision Tree Classifier internally relies upon a single Decision Tree comprising of nodes, some of which are leaves. When training the classifier, new nodes are build with the help of a single feature per node. The currently looked at samples will be split by inventing a rule based on the single feature dividing them into subsets. This is done at multiple nodes. On predicting, which class an example belongs to is decided by reaching a specific leaf. A classifier based on a single Decision Tree is often not sufficient on its own, mostly because it tends to overfit on the training set [22].

### 3.5.2 Random Forest Classifier

The Random Forest Classifier belongs to the Ensemble methods and contains multiple Decision Trees. These are trained using a subset of the training samples. Additionally, the splitting of these samples at nodes can be controlled. Depending on the implementation, the output of each Decision Tree by is combined by the average of the weighted output of each tree instead or using majority voting [22]. Overall, the overfitting problem of the Decision Tree Classifier is reduced by introducing randomness and using multiple trees [22].

### 3.5.3 Bagging Classifier

The Bagging Classifier too belongs to the set of Ensemble methods. Instead of only using internal Decision Trees, the Bagging Classifier is agnostic towards its internal classifiers, hence a wide range of different types can be used. Sklearn recommends using so called strong internal classifiers, which are those, who could be used as a full classifier on their own. Like the Random Forest, the usage of multiple internal classifiers reduces the overfitting problem and improves prediction quality [22] [15]. The training of this model is done by so called bootstrapping, where random subsets of the training data are drawn for each internal one. Here, the prediction is also achieved by combining the output [22]. Bagging allows for a wide range of hyper-parameters to control how many features and samples are drawn.

### 3.5.4 Adaptive Boosting Classifier

Boosting is another way, as opposed to bagging, to combine multiple classifiers to improve performance. As opposed to the later, Maclin et al. did not find a general improvement, but calls the application of this technique more situational [15]. Also, boosting relies on weak internal models, which could not be used as a functioning model on its own [22]. In the scope of this work, shallow Decision Trees in an Adaptive Boosting model were used. These shallow Decision Tree are often called stumps, since they only consist of one splitting node and leaves [6]. The Adaptive Boosting model was first introduced by Freund et Schapire [7]. With the training set  $(X, Y)$  with  $x_i \in X$  being the feature and  $y_i \in Y$  being the label of example  $i$ , the model assigns equal weights  $d_t^i$  for every example. Then, at step  $t$ , a new weak classifier is build which implements a hypothesis  $h_t$  for the data with the distribution  $D_t$ . The weak classifier either allows for weights internally or it gets a subset of the data chosen by its weights. This hypothesis needs not to be perfect, it just needs to be better than randomly assigning examples to classes. The error of these hypothesis or classification  $\epsilon_t$  is denoted as the sum of misaligned examples with weights, as in equation 6. A weight for the next step at  $i$  will be updated using the old weight, the overall weight distribution at time  $t$ , whether the hypothesis was correct for  $i$  and the error  $\epsilon_t$ . This update method will assign higher weights for misclassified samples. This will lead to over-representation of difficult to classify examples. When predicting, the result will be a majority vote. Adaptive Boosting is often shortened verbally to AdaBoost [7].

$$\epsilon_t = \sum_{h_t(i) \neq y_i} d_t^i [7] \quad (6)$$

$$d_{t+1}^i = \text{update}(d_t^i, D_t, \epsilon_t, h_t(i)) [7] \quad (7)$$

### 3.5.5 Other methods

Depending on the implementation, non-numerical values in numerical features may not be accepted. This can make a pre-cleaning step necessary, where a strategy is enacted to get rid of those values. This is being referred to as imputing. In this work, non-numeric values were replaced with a constant number [22]. The Support Vector Classifier was used as a comparison to other classifiers, since it is often used in classification problems [22]. Principal Component Analysis was done for visualization of datasets and during initial testing [22]. The Receiver Operating Statistic and feature importance plots were used for the validation of classifiers [22].

### 3.6 Feature Selection

The features for our models were taken from Hi-C contact matrices. To this end, the neighbourhood of every bin was selected. This neighbourhood contains every pair of interactions between two bins up to a certain distance  $d$  from the observed bin  $b$ , as in equation 8. Graphically, this leads to a square being selected over the diagonal of the matrix for every bin, with the selected bin being at the center. Since the matrix is symmetrical, selecting the triangle over the diagonal as can be seen in figure 3 is sufficient, which corresponds to only selecting the Interactions between  $i$  and  $j$ , where  $i \leq j$  as in equation 8.

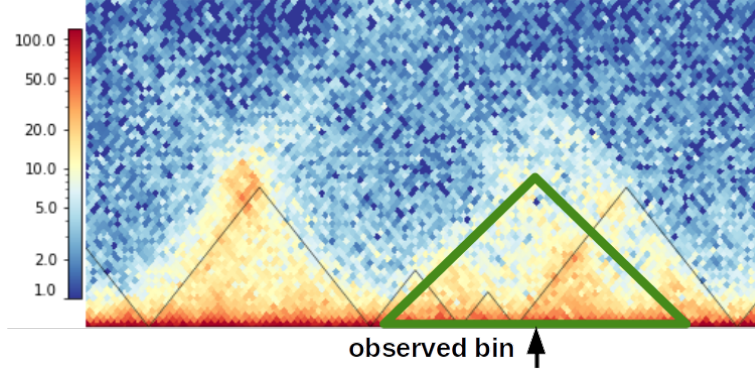


Figure 3: figure 2 with feature selection [27]



$$indices_b = [b - d, b + d] \quad (8)$$

$$features_b = \{I(i, j) | i \in indices_b, j \in indices_b, i \leq j\} \quad (9)$$

### 3.7 Resampling methods

Generally a Hi-C Matrix contains more non-boundary positions than boundary positions [31]. This makes our input dataset imbalanced, where one class has more examples than the other one. Training Machine Learning models on imbalanced datasets can skew models towards overemphasizing the majority class and leads to overall worse performance [10]. Resampling methods were used to allow for meaningful classification. These methods take an imbalanced dataset and try to balance them into a dataset containing the same number of examples for every class [10]. In this work, four classes of resampling methods were tested. Oversampling methods try to balance the dataset by creating examples for the minority classes. Undersampling methods with prototype selection try to choose examples of the majority class and discard the rest. Those with prototype creation, try to synthesize examples from the majority class. Combined ones try to combine both undersampling and oversampling techniques [13]. An overview of the resampling methods that were used, can be seen in table ?? . The most basic one is Random Undersampling, in which examples are selected randomly from the majority class. Cluster Centroids is the only prototype generation method, that we tested. It resamples the given dataset by generating  $n$  samples, where  $n$  is the size of the minority class, of each majority class. Those samples are generated by clustering the majority class via KMeans Clustering, computing  $n$  clusters with one center each [13]. In the method "Edited Neares Neighbours", samples are selected via local criteria. Those criteria denote, whether or not samples with similar features are classified equally. If not, they are pruned from the dataset. When used as a resampling method, only samples of the majority classes are omitted. Multiple derivatives of this method exist, like Repeated Edited Nearest Neighbour [13], but here, only the basic method was tested. Synthetic Minority Oversampling Technique synthesizes samples for the minority classes and was first introduced in Chawla et al.[4]. To this end, neighbours of a sample are taken and combined to produce similar examples locally. Those neighbours are randomly chosen [4]. SMOTEENN combines Edited Nearest Neighbours and Synthetic Minority Oversampling Technique [13].

### 3.8 Label Noise

An input set for training and testing can contain so called label noise. In noisy datasets, examples are misclassified, where they are assigned to classes, to which they do not actually belong. Common Machine Learning models for classification usually assume a perfect input set and will be impacted severely by imperfect ones [8]. The paper "Classification of in the Presence of Label Noise" [8] surveys multiple approaches to learning with noisy data and calls Label Noise a big issue in many Machine Learning

Models. Two approaches, that were presented in this paper were used by us to improve performance. The first approach was the testing of a label-noise-resistant model. In this work, a Bagging Classifier was used, which belongs to this set of models [8]. This is done by combining the output of multiple internal classifiers [8]. Another approach we took, was to take the methodology described in the Confident Learning paper by Northcutt et al. [17]. This work tried to estimate an input set without misclassified labels from one with noisy labels. The authors claim, that it is a general approach, which does not need a specific estimator to function and does not need hyperparameter tuning. To this end, from the training set, a joint distribution between the true labels and the known erroneous labels is estimated by computing a confident joint. From this confident joint, the set of erroneously assigned labels is estimated. In a last step, the data is pruned and reweighted using the confident joint and the set of erroneously assigned labels [17]. Another paper by the same authors is "Learning with Confident Examples: Rank Pruning for Robust Classification with Noisy Labels" [19]. This technique can be used on binary classification problems, when one class does not contain falsely assigned labels. For our case, this was not integrated, as it was not appropriate. But it might be done in the future, if a class with only boundaries or one with only non-boundaries can be produced.

### 3.9 Easy Ensemble Classifier

In their paper "Exploratory Undersampling for Class-Imbalance Learning", Liu et al. introduced two classifiers for imbalanced learning [10]. One of those two models is the Easy Ensemble Classifier. This classifier is in actuality a Bagging Classifier, with internal Adaptive Boosting Classifiers, which in turn use internal weak Decision Tree Classifiers. Additionally, Random Undersampling is performed by the Easy Ensemble Classifier beforehand. This leads to a model, which can work with an imbalanced dataset [13] [10]. As has been previously stated, Bagging is also a strategy for working with noisy datasets [8]. This classifier was used as our default classifier in this paper.

### 3.10 HiCExplorer

HiCExplorer is a command line suite of tools for working with Hi-C Data [31]. It is written in the python programming language. The most important method used in this work was hicFindTADs, which we used by default for our training data and for evaluation. Other tools used, were hicAdjustMatrix for cutting matrices, hicTransform and hicNormalize for normalization, pyGenomeTracks and hicPlotMatrix for plotting and hicInfo for getting basic information about a matrix. Also, some code snippets for ingesting and preparing of Hi-C Matrices were taken from HiCExplorer [31]. hicFindTADs is a subprogram of HiCExplorer. Here, from an input Hi-C Matrix TADs can be called. hicFindTADs first computes a z-score matrix, based on the local contact frequency up to a maximum bin distance  $d$ , similar to our feature selection method. So for every bin  $i$ , the corresponding interactions  $I(x_1, x_2)$  where  $x_1, x_2 \in [i - d, i + d]$  are considered. For each bin in the matrix a so called TAD-separation score is calculated, which corresponds to the mean of the values in this matrix within the interval  $[i - d, i + d]$ . In contrast to

our selection method, this is done for varying distances, which can be set via parameters. These scores are compared to the neighbourhood and local minima identified. These minima are flagged as boundaries, if they pass as statistical significant [25].

### 3.11 Alternative TAD Callers

For Validation of our model we also tested it on TAD Callers other than HiCEXplorer. To this end we consulted the work "Comparison of computational methods for the identification of topologically associating domains" of Zufferey et al.[33] and picked TAD Callers, which like hicFindTADs also compute disjunctive TADs. We were unsuccessful in running SpectralTAD on our data. ClusterTAD is a unsupervised learning approach to TAD calling. Internally, it relies on a number of clustering methods such as K-Means, Expectation Maximization and Hierarchical Clustering. The method can be preset, if this is not done, ClusterTAD will test multiple methods on the input data. It will also vary the number of clusters K, which can alternatively set by a parameter. It was available via a packaged JAR file and called via a python script [20]. rGMAP uses a Gaussian Mixture model to classify interactions inside a domain and interactions between domains. Then a proportion test is used to find blocks of high interaction count and TADs are called using those two forms of information. It was available to us as a library in the R language and called via a R script [32].

### 3.12 Python Libraries and Platform

For the implementation of our Machine Learning Pipeline, the python libraries scikit-learn (sklearn for short), imbalanced-learn (imblearn) and cleanlab were imported. For all classifiers, their implementation in the scikit-learn library was used. All classifiers needed to implement the fit and predict methods, as can be seen in table 3 and need to inherit from sklearn.BaseEstimator. For the directly called classifiers it was necessary to implement a partial fit, which allows to fit an existing classifier on additional data. In the context of sklearn, this means, that the classifier must allow a so called warm start. The partial fit was implemented by adding additional internal classifiers via the get\_params and set\_params methods [22]. An official implementation of the Easy Ensemble Classifier exists within the imblearn library, which internally relies upon the implementation of the Bagging Classifier and the AdaBoost Classifier of the sklearn library [13]. This pipeline was rebuild following the source of the official version and using the same implementations of the sklearn library. The imblearn library provided all resampling methods [13]. The methodology of Confident Learning was integrated from the Cleanlab library. This library provided Confident Learning as an out-of-box pipeline called Learning with Noisy Labels. As has been previously stated, this pipeline does not rely on hyper-parameters and works as a wrapper for any given classifier [18]. Parallelization options were provided by all libraries [22] [18] [13]. An overview of the python libraries in our project can be found in table 8. During initial testing the Galaxy instance of the University of Freiburg [2] was used. This included TAD calling with hicFindTADs, as well as matrix normalization and cutting. Galaxy is a platform for

Table 3: methods and attributes of our classifiers

<b>method</b>	<b>description</b>
fit	train model on data
predict	predict from passed set
get_params	get parameter of estimator
set_params	set parameter of estimator
<b>attribute</b>	<b>description</b>
warm_start	flag for partial fit
n_jobs	thread count for parallelization
n_estimators	number of internal classifiers

Data-Science Tasks within the field of Biomedical research [2]. It was accessed via its web interface. For all other purposes, local computation resources were used. This included a local installation of the Anaconda platform [1] and its associated Jupyter Notebooks [23] for development and testing. For the TAD-Caller rGMAP, a local R distribution was used and ClusterTAD was called via its packaged JAR version.

## 4 Results

In this work, the previously described methods were implemented into a python pipeline. Then, input datasets were selected and TADs were called using hicFindTADs or alternative TAD Callers. These called TADs, together with aforementioned datasets were used for the training and testing of said pipeline. A default configuration was found for this pipeline, which can be seen in table 5. Then, variations of these input parameters were applied and tested. These variations are explained in table 6. In a last step, input data was filtered with the help of a CTCF protein track. Then, the pipeline was tested with this filtered data and the overlap with the protein track computed and the called TADs visualized.

### 4.1 Implementation and Integration into HiCExplorer

Our pipeline was implemented into a library python file and two accessing programs. In the library file, most of the functionality was implemented. Here, to a given input Hi-C Matrix or a list of matrices, feature selection, normalization and resampling can be done. Optionally, cleanlab can be used in addition. After these pre-cleaning steps, a given classifier can be trained on the data or TADs can be called. A pipeline to ingest domain files in BED format and protein files in narrowPeak format was also implemented. These files can be intersected, to filter out any low-quality TADs by checking against a threshold value for protein peaks. Access of this library was divided into aforementioned programs, namely into a training part and a predict part. Into the training program, all necessary modes for training classifiers were integrated. The first mode is train\_test, which allows the testing of a new classifier with a single input set. On this set, a train-test

Table 4: datasets used in this work

name	type	genome	resolution	format
Rao2014 [27].	Hi-C Matrix	human GM12878	10 kb	cool
dm [30]	Hi-C Matrix	drosophelia melangaster	10 kb	h5
CTCF_Rao2014 [28]	protein track	human GM12878	n.a.	narrowPeak

split is performed and with the help of a domain file this classifier is trained and a classification report is produced. With the `train_new` mode, a new classifier can be defined and initially trained on a given input set and an associated domain file. By default, an Easy Ensemble Classifier will be trained, but the user may specify a custom sklearn classifier and may also pick a resampling method from a list or pass any imblearn implementation of these methods. An additional set of hyper-parameters can be passed, which includes the size of the feature selection and the imputing value. The output will be a newly trained classifier in form of a pickled file. With the `train_existing` mode, a pickled classifier can be passed to train it on additional input matrices with associated domain files. The output will again be a pickled classifier in form of a file. Before using a classifier for TAD calling, it is possible to check the performance of this classifier with the help of the `predict_test` mode. Here, a given classifier with an input set can be tested and the output compared against a passed domain file. The output will be a classification report denoting the performance of the classifier. For all modes, a protein file in the narrowPeak format and a threshold value can be passed to filter the passed domain file. The actual TAD calling is provided by the second program. Here, a given input set can either be predicted by provided default classifier, or a classifier in file form may be passed. The output will be a domain file with the predicted boundaries. For all nodes mentioned, the resolution and normalization method of all sets both for training and predicted sets must match.

## 4.2 Datasets

For testing and fitting our pipeline, a couple of datasets were used from multiple sources. Our main dataset, both for explorative testing and training of our pipeline, was the Hi-C Matrix of the human genome from Rao et al. [27]. Another dataset was the genome of a *Drosophila Melanogaster* (a type of fly) from a Galaxy Tutorial [30]. For our matrices protein track files were found, where possible. As previously stated, TAD boundaries for our input sets were produced using hicFindTADs or our alternative TAD-Callers.

## 4.3 Exploration of parameters for hicFindTADs

With the goal to providing a first baseline for our classifier, hicFindTADs was used to call TADs on the Rao2014 matrix. Hyper-parameters of hicFindTADs were varied and the output was checked visually for the best result. This was repeated for every input dataset. The results can be seen in table 5. These results were compiled into domain files and associated protein files were found.

Table 5: Default parameters for hicFindTADs and our classification

<b>hicFindTADs</b>	
<b>parameter</b>	<b>value</b>
min. win. size	30000
max. win. size	100000
step size	10000
corrections	FDR
q-value	0.03
threshold	0.01
<b>classification</b>	
<b>parameter</b>	<b>value</b>
resampling method	Random Undersampling
classifier	Easy Ensemble Classifier
number of estimators	90
distance	20
resolution	10kb
normalization	obs/exp
impute value	-1
use cleanlab	no

#### 4.4 Exploration of classifiers, resampling methods and parameters

For the decision on which model will be used, a number of different classifiers were initially tested for their performance on the Rao2014 dataset and the domain file provided by hicFindTADs. Due to its abundance in literature, the Random Forest Classifier was tested at first, which yielded results, which were not significant enough to further explore this classifier. Then, other Machine Learning approaches were tested, namely boosting and bagging. Boosting alone, in form of a AdaBoost Classifier also underperformed on the given dataset. Approaches with Bagging on Decision Trees or Support Vector Classifier were also abandoned. Initially very promising results yielded the combination of boosting and bagging in form of the Easy Ensemble Classifier. Due to this results, this classifier became the default from there on. Also, as a baseline, some initial parameters were set 5. The default resampling method of Random Undersampling was varied. The baseline method of each class of resampling methods, namely oversampling, undersampling prototype creation and undersampling prototype selection, was tested. Promising results beyond Random Resampling was given by the prototype creation method Cluster Centroids. The results of these tests can be seen in table 6 test 1 - 9, where chr1 of Rao2014 served as the test set and the remainder of the genome as the training set. Some hyperparameters of our pipeline were explored, to get some insights and exclude some ranges of parameters for later tuning. The results can be seen in table 6 test 13 - 22. Different normalization methods were tested, namely range normalization

and obs/exp normalization. Also, the maximum distance of the feature selection and tried different ranges for the range normalization were varied. Imputing values were also varied. In a last test, LearningWithNoisyLabels of the cleanlab library was used. Do note, that this procedure assumes a perfect test set, which was not given in our case [18]. As an additional insight, Principal Component Analysis was performed in the feature selection step. While this approach was abandoned during initial testing, the result of it can be seen on a subset of the data in figure ?? with three components. As can be seen, no clear distinction between boundaries and non-boundaries can be recognized. Also no clear clusters seem to form. Do note, that the implementation of Principal Component Analysis of the sklearn library has no knowledge about the classes of the data. For further validation of the Easy Ensemble Classifier, the Receiver Operating Statistic was plotted. This can be seen in figure 5. This plot shows a clear advantage of this classifier over random assignment. Also, in figure 6, the importance of features was displayed. Here, the distance was reduced to 15 for better visualization. Every bar in this plot denotes to a feature in the training set. The height of each bar is equal to the importance of this feature for classification. Hence, how big the value of this features has an impact on the class of an example. Bars in red were interactions on the identity diagonal of the matrix. Importance of features seem not to be uniformly distributed. While there were some features in red, which seem to be overemphasized, this does not seem to be a general rule.

#### 4.5 Testing on different datasets and TAD Callers

To check, whether our pipeline can generalize from the provided examples, our classifiers were trained on the Rao2014 dataset and later tested on the matrix of *Drosophila Melanogaster*. The results can be seen in table 6 test 10 - 12. The prediction quality dropped compared to the tests within the Rao2014 dataset itself, but remained clearly over random chance. This was true for all tests, including variations of the resampling method, although with worse prediction performance. As an additional validation of our model, it was tested against other TAD callers. As has been previously mentioned, two alternative TAD Callers were used. Instead of producing a domain file for the input set with hicFindTADs, instead this set was produced with another TAD Caller. This means, that the classifier was trained on the behaviour of the chosen TAD Caller and then tested if it can reproduce this behaviour. This was done for rGMAP and ClusterTAD. The classifier was able to reproduce the behaviour of rGMAP well within the Rao2014 dataset, but performance dropped significantly when trained on the Rao2014 dataset and tested on the dm dataset. When using ClusterTAD as a TAD Caller, only an accuracy of 0.867 could be achieved inside the Rao2014 dataset. On the dm dataset, the classification performance dropped to the performance of random chance. The testing of SpectralTAD failed. The results can be seen in table 6 test 23 - 26.

#### 4.6 Validation against a CTCF track and Visualization of TADs

In a last step, TADs were called using aforementioned classifier on a matrix, like it would be used in production. Here, instead of using the obs/exp normalization, range normalization

Table 6: condensed results table. If not otherwise noted, default parameters were used

#Test	parameter variations	training set	test set	accuracy
1	None	Rao2014	Rao2014	0.917
2	Random Forest;ENN	Rao2014	Rao2014	0.506
3	Random Forest;SMOTE	Rao2014	Rao2014	0.554
4	Random Forest;Cluster Centroids	Rao2014	Rao2014	0.661
5	Edited Nearest Neighbour	Rao2014	Rao2014	0.682
6	SMOTE	Rao2014	Rao2014	0.836
7	Cluster Centroids	Rao2014	Rao2014	0.881
8	Bagging Classifier (SVC);Cluster Centroids	Rao2014	Rao2014	0.62
9	Bagging Classifier (Decision Tree);Cluster Centroids	Rao2014	Rao2014	0.722
10	None	Rao2014	dm	0.808
11	Cluster Centroids	Rao2014	dm	0.736
12	use cleanlab	Rao2014	dm	0.771
13	distance = 4	Rao2014	Rao2014	0.886
14	distance = 5	Rao2014	Rao2014	0.917
15	distance = 10	Rao2014	Rao2014	0.892
16	distance = 15	Rao2014	Rao2014	0.931
17	distance = 20	Rao2014	Rao2014	0.917
18	distance = 25	Rao2014	Rao2014	0.914
19	range normalization (0-10)	Rao2014	Rao2014	0.92
20	range normalization (0-1)	Rao2014	Rao2014	0.92
21	impute value = 0	Rao2014	Rao2014	0.917
22	use cleanlab = True	Rao2014	Rao2014	0.871
23	use input data build with ClusterTAD	Rao2014	Rao2014	0.867
24	use input data build with ClusterTAD	Rao2014	dm	0.447
25	use input data build with rGMAP	Rao2014	Rao2014	0.958
26	use input data build with rGMAP	Rao2014	dm	0.671



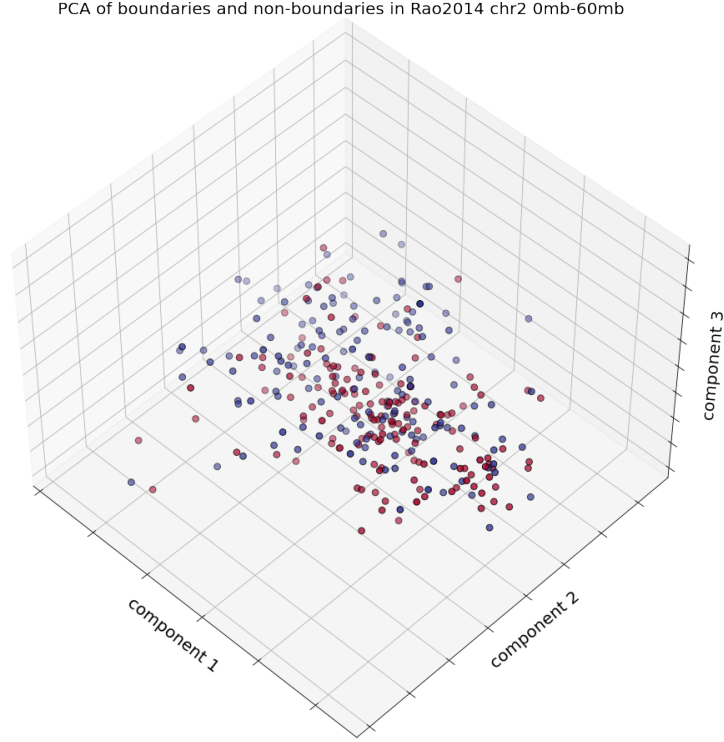


Figure 4: selection of boundary (blue) and non-boundary (red) data points on chromosome 2 of Rao 2014 [27] clustered with Principal Component Analysis with 3 components

was used and the parameter distance was reduced to 15, since those parameters seemed to perform better. Instead of an unfiltered domain file from hicFindTADs, the classifier was trained on a domain file, which was intersected with a CTCF protein track. Only boundaries with a CTCF peak on the same position were considered. After TAD Calling, the resulting domain files were tested against the same CTCF track and the percentage of matched boundaries were written into table 7. An additional step was introduced, which filtered out called boundaries at positions, where a boundary was already called in the vicinity of 50kb. As can be seen, about 50% of the boundaries computed by our classifier had a CTCF peak at their position. Due to the difference in number of boundaries called between our classifier and hicFindTADs, it was difficult to compare outcomes. Instead, a subset of the computed TADs were visualized in figure 7 for hicFindTADS, figure 8 for the classifier without cleanlab used and figure 9 for the classifier with cleanlab used.

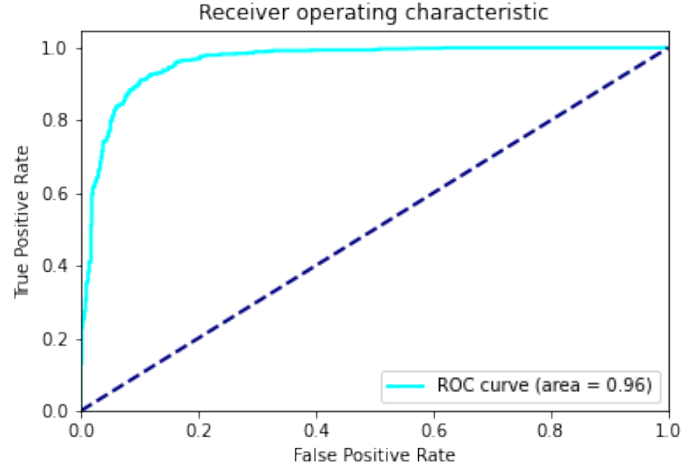


Figure 5: Receiver Operating Statistic for the Easy Ensemble Classifier of test 1

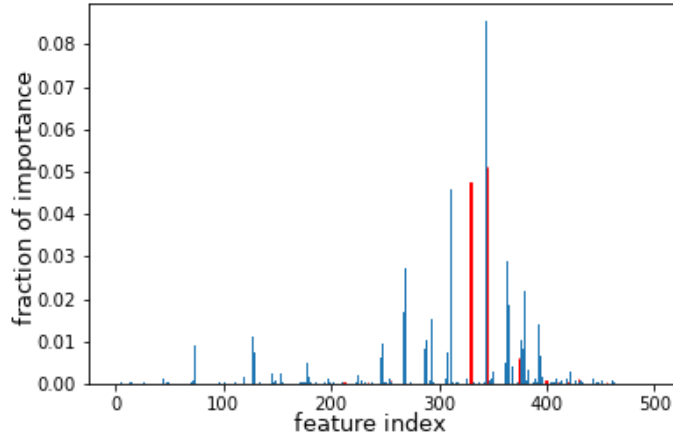


Figure 6: Feature importance of features in test 16. Features on diagonal in red

## 5 Discussion and Conclusion

Our proposed feature selection and choice of input matrices seemed to be sufficient to describe the data. Our testing of resampling strategies with a high sample size of over 10000 samples, showed, that only Cluster Centroids provided a comparable performance during initial testing to Random Undersampling. A functional model in the Bagging Classifier with AdaBoost Classifiers as base estimators was found, which proved to be noise robust and worked well with our imbalanced dataset. Overall, an accuracy score in

Table 7: Called TADs for chr1 of Rao2104 matched with a CTCF narrowPeak track

TAD Caller	matched boundaries	total boundaries	match percentage
hicFindTADs	413	659	62.67%
classifier with cleanlab	624	1199	52.04%
classifier without cleanlab	581	1082	53.70%

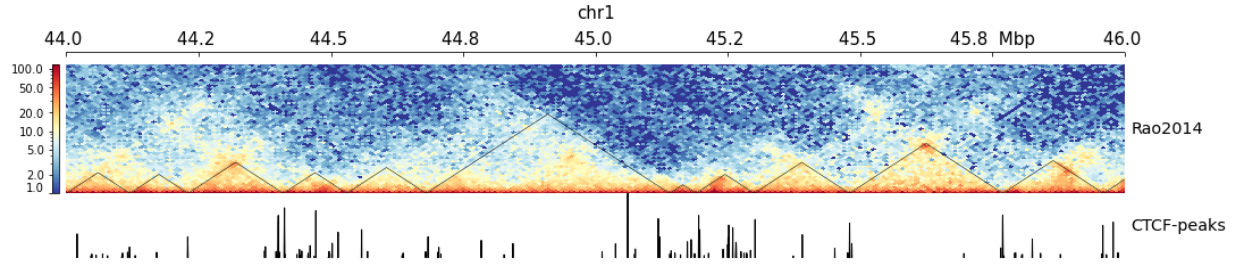


Figure 7: subset of TADs called with hicFindTADs on Rao2014 and CTCF track[28]

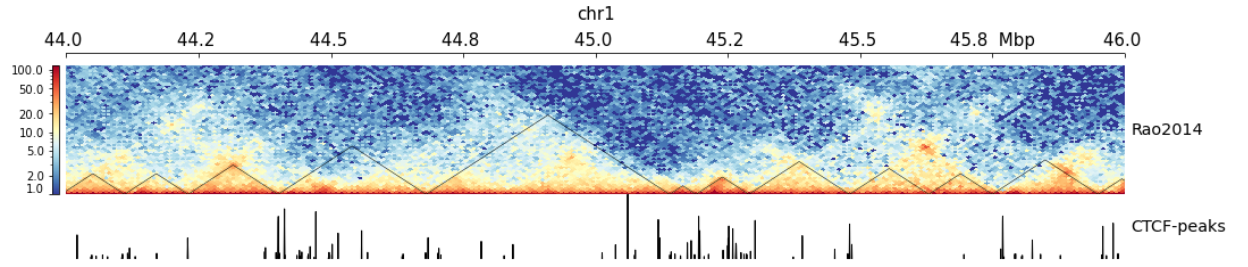


Figure 8: subset of TADs called with our classifier without cleanlab on Rao2014 and CTCF track[28]

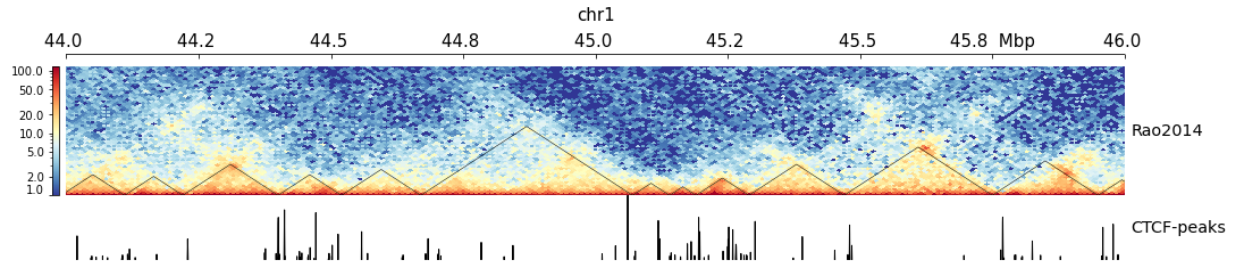


Figure 9: subset of TADs called with our classifier with cleanlab on Rao2014 and CTCF track[28]

Table 8: used python libraries and their versions

library	version
sklearn	0.23.2
imblearn	0.7.0
cleanlab	0.1.1
hicexplorer	3.5
hicmatrix	14
pybedtools	0.8.1
python	3.7.8
pandas	1.1.0
numpy	1.19.1
matplotlib	3.3.0
scipy	1.5.2

the low 90% range could be achieved without hyper-parameter tuning. A score in the low 80% range could be achieved when testing on a different dataset. Our chosen model can be expanded by introducing additional estimators and training new data on them [22]. We were able to implement our pipeline as a python application and integrate it into HiCEXplorer. The major challenge of our input was label noise introduced by false positives and false negatives. We were not able to produce a input dataset with close to perfect accuracy. Hence, our model can only mimic the behaviour of the tested TAD Callers. This also means, that achieving a score close to 100% is not desirable and might not reflect a better performance. Our classifier seemed to call more TADs than the input data set it was trained with. The output of the TAD Calling had a large intersection with a given CTCF track. The performance of our TAD Caller was difficult to compare to the performance of hicFindTADs due to differing numbers of called TADs.

## References

- [1] Anaconda software distribution, 2020.
- [2] Enis Afgan, Dannon Baker, B  r  nice Batut, Marius van den Beek, Dave Bouvier, Martin   ech, John Chilton, Dave Clements, Nate Coraor, Bj  rn A. Gr  ning, Aysam Guerler, Jennifer Hillman-Jackson, Saskia Hiltemann, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res.*, 46(W1):W537–W544, 2018.
- [3] Boyan Bonev and Giacomo Cavalli. Organization and function of the 3d genome. *Nature Reviews Genetics*, 17:661–678, 10 2016.
- [4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer.

- Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [5] FRANCIS CRICK. Central dogma of molecular biology. *Nature*, 227(5258):561–563, Aug 1970.
  - [6] Akash Desarda. Understanding adaboost, 2019.
  - [7] Yoav Freund and Robert E. Schapire. A short introduction to boosting. 1999.
  - [8] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *Neural Networks and Learning Systems, IEEE Transactions on*, 25:845–869, 05 2014.
  - [9] Jiannan Guo. Transcription: the epicenter of gene expression. *Journal of Zhejiang University SCIENCE B*, 15(5):409–411, May 2014.
  - [10] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. On the class imbalance problem. In *2008 Fourth International Conference on Natural Computation*, volume 4, pages 192–201, 2008.
  - [11] James M. Heather and Benjamin Chain. The sequence of sequencers: The history of sequencing dna. *Genomics*, 107(1):1–8, Jan 2016.
  - [12] W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, Haussler, and David. The human genome browser at ucsc. *Genome Research*, 12(6):996–1006, 2002.
  - [13] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
  - [14] Erez Lieberman-Aiden, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley Bernstein, M. A. Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A. Mirny, Eric S. Lander, and Job Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, 2009.
  - [15] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. *AAAI/IAAI*, 1997:546–551, 1997.
  - [16] D. Michie, D. J. Spiegelhalter, and C.C. Taylor. Machine learning, neural and statistical classification, 1994.
  - [17] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels, 2019.
  - [18] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Cleanlab, 2021.

- [19] Curtis G. Northcutt, Tailin Wu, and Isaac L. Chuang. Learning with confident examples: Rank pruning for robust classification with noisy labels. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, UAI’17. AUAI Press, 2017.
- [20] Oluwatosin Oluwadare and Jianlin Cheng. Clustertad: an unsupervised machine learning approach to detecting topologically associated domains of chromosomes from hi-c data. *BMC Bioinformatics*, 18(1):480, Nov 2017.
- [21] S. Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *CoRR*, abs/1503.06462, 2015.
- [22] 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*, 12:2825–2830, 2011.
- [23] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.
- [24] Aaron R. Quinlan and Neil Kindlon. bedtools, Jan 2021.
- [25] Fidel Ramirez, Vivek Bhardwaj, Laura Arrigoni, Kin Chung Lam, Björn A. Grüning, José Villaveces, Bianca Habermann, Asifa Akhtar, and Thomas Manke. High-resolution tads reveal dna sequences underlying genome organization in flies. *Nature Communications*, 9(1):189, Jan 2018.
- [26] Fidel Ramirez, Joachim Wolff, Björn Grüning, Vivek Bhardwaj, Devon Ryan, and Friederike Dündar. Deeptools/hicexplorer/, 2020.
- [27] Suhas S.P. Rao, Miriam H. Huntley, Neva C. Durand, Elena K. Stamenova, Ivan D. Bochkov, James T. Robinson, Adrian L. Sanborn, Ido Machol, Arina D. Omer, Eric S. Lander, and Erez Lieberman Aiden. A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665–1680, Dec 2014.
- [28] Snyder Stanford University. *Stanford<sub>chipseq</sub>m12878<sub>ctcf</sub>(sc – 15914)<sub>std</sub>*, Mar2021.
- [29] Furlan-Magaril M et al. Wingett S, Ewels P. Hicup: pipeline for mapping and processing hi-c data. 2015.
- [30] Joachim Wolff. Galaxy hi-c training material dm3, Feb 2018.
- [31] Joachim Wolff, Leily Rabbani, Ralf Gilsbach, Gautier Richard, Thomas Manke, Rolf Backofen, and Björn A Grüning. Galaxy HiCEXplorer 3: a web server for reproducible Hi-C, capture Hi-C and single-cell Hi-C data analysis, quality control and visualization. *Nucleic Acids Research*, 48(W1):W177–W184, 04 2020.

- [32] Wenbao Yu, Bing He, and Kai Tan. Identifying topologically associating domains and subdomains by gaussian mixture model and proportion test. *Nature Communications*, 8(1):535, Sep 2017.
- [33] Marie Zufferey, Daniele Tavernari, Elisa Oricchio, and Giovanni Ciriello. Comparison of computational methods for the identification of topologically associating domains. *Genome Biology*, 19(1):217, Dec 2018.