Bioinformatics Project Report

Marco Lassandro ID. 945907

1 INTRODUCTION

The project consists on the application of supervised machine learning methods for the prediction of active and inactive cis-regulatory regions, including both enhancers and promoters, for a given cell line, in particular for the cell line H1. Much research is devoted to the identification of cis-regulatory elements and to their cell-specific activation status. Such studies are essential to dissect the mechanisms underlying the modulation of gene expression and to understand the functional impact of genetic variants on human diseases [2].

In the current state of art many published works have been carried out to study this type of problems, here some articles related in particular on the prediction of active states in CRRs by using machine learning algorithms:

- In [2] they have investigated the capacity of deep learning models to separate enhancers and promoters and to distinguish them from other regions in and between activity states. For each experiment they have conducted a MLP model search from no hidden layers up to three hidden layers with maximally 256, 128, and 64 units in the first, second, and third hidden layers. The deep model takes experimentally derived features over genomic regions as inputs and outputs class labels of these regions with probabilities. They also have been provided some experiments to access the utility of DNA sequences for the recognition of promoters and enhancers, and the discrimination between active and inactive regulatory sequences using string sequence kernels;
- In [1] they provided additional evidence that Feed Forward Neural Networks (FFNN) trained on epigenetic data and one-dimensional convolutional neural networks (CNN) trained on DNA sequence data can successfully predict active regulatory regions in different cell lines. Moreover, they showed that model selection by means of Bayesian optimization applied to both FFNN and CNN models can significantly improve deep neural network performance, by automatically finding models that best fit the data;
- in [3], based on rich data resources such as the Encyclopedia of DNA Elements (ENCODE) and the Functional Annotation of the Mammalian Genome (FANTOM) projects, they have used and compared six machine learning (ML) methods (SVM, MLP, 1-NN, NB, RF, Sequential minimal optimization (SMO)), for the identification of enhancer and promoter regions in the human genome, given 7 different cell line data;

In this project, the models used are based on three types of neural networks: Feed Forward Neural Network (FFNN), Convolutional Neural Networks (CNN) and Multi-Modal Neural Networks (MMNN).

After the pre-processing and visualisation of the data, for each task, a model selection phase, has been conducted in order to get, for each type of neural network, the model that best fit the given tasks; selected models have been evaluated with different metrics and statistical tests have been applied on the experiments in order to make a reliable comparison of the results obtained.

2 EXPERIMENTAL SETUP

All the experiments has been run in a COLAB environment with a GPU/TPU accelerator, for the construction and the training of the NN models it has been used the keras library.

2.1 Dataset source

Two types of dataset have been used to address the proposed tasks, in the case of FFNN models the dataset used consists of epigenomic data related to defined regulatory regions, created with the ChIP-sequencing method. Such data has been retrieved by querying the ENCODE database, where ENCODE is a portal offering an immense amount of experimental results in the molecular biology field. In practice for each CRRs there is a set of numeric features that correspond to the genes that code for the proteins we are measuring the activation of, the float values represent the amount of interaction measured in that specific region.

The labels about the activity status of the cis-regulatory regions, have been derived using thresholds on tags per million (TPM) values of the Cap Analysis of Gene Expression (CAGE) data set, from FANTOM5 database, where FANTOM is a worldwide collaborative project aiming at identifying all functional elements in mammalian genomes.

To retrieve the epigenomic data from ENCODE and retrieve labels from FANTOM, avoiding to query terabites of bigwig files, the epigenomic_dataset library has been used in order to automatically download the complete dataset.

In the case of CNN models the data used, for each cis-regulatory region, was a sequence of nucleotides, extracted from the "UCSC genomic browser" (an on-line, and downloadable, genome browser hosted by the University of California), that after retrieved, they are represented by one-hot encoded sequences.

Also in this case to retrieve the sequences and match them with the labels derived from the FANTOM database it has been used the epigenomic_dataset library.
For all the tasks it has been considered the same window size for the genomic regions, set to 256, and the genome hg38 assembly as reference genome.

2.2 Considered Tasks

Given the cell line **H1**, four tasks has been proposed:

- **Task 1**: Inactive Promoters vs Active Promoters (TPM $\geq$ 1.0) (IP vs AP ((TPM $\geq$ 1.0))))
- **Task 2**: Inactive Promoters vs Active Promoters (TPM $\geq$ 0.5) (IP vs AP ((TPM $\geq$ 0.5))))
- **Task 3**: Inactive Enhancers vs Active Enhancers (TPM $\geq$ 1.0) (IE vs AE ((TPM $\geq$ 1.0)))
- **Task 4**: Inactive Enhancers vs Active Enhancers (TPM $\geq$ 0.5) (IE vs AE ((TPM $\geq$ 1.0)))

2.3 Models

For each task three types of neural network models (FFNN, CNN and MMNN) have applied and evaluated. The choice of the architecture for the FFNN and CNN models and the settings of the learning hyper-parameters has been carried out through a **Bayesian optimization approach** at each task, in this way it has been avoided the use of a grid search approach for the model selection and tried to follow what has been done in [1]. Moreover, for each task, MMNN models have been constructed by chaining the last hidden layers, of the CNN and FFNN models chosen by the bayesian optimization procedure.

2.3.1 Bayesian Optimization

What has been used for model selection step is the **gp_minimize** function of the **scikit-optimize** library. The idea is to approximate a given function using a Gaussian process. In other words the function values are assumed to follow a multivariate gaussian. The covariance of the function values are given by a GP kernel between the parameters. Then a smart choice to choose the next parameters to evaluate can be made by the acquisition function.

In practice, the **gp_minimize** function has been used to choose the network architecture and the value of the learning parameters to minimize the **validation loss** of the validation set obtained by splitting the given dataset, with no feature selection step, in 80% as training set and the remaining 20% as test set, by using the **StratifiedShuffleSplit** function of the sklearn library. For each task the number of evaluations performed by the bayesian optimization function has been 40, the acquisition function parameter has been set to "gp_hedge" so that the acquisition function used to select the parameters on the next iteration are probabilistically choosen by the gp_minimize, three functions could be selected: lower confidence bound, negative expected improvement and negative probability of improvement.

2.3.2 Models Architectures

In the case of the FFNN models the hyper-parameter spaces defined for the Bayesian Optimization procedure can be shown in the following table:

Layers	**Hyper-parameter Space**	**Activation**
No. of Dense Layers	{1, 2, 3}	
No. of units layer 3	{128, 256, 512}	ReLU
No. of units layer 2	{64, 128, 256}	ReLU
No. of units layer 1	{16, 32, 64, 128}	ReLU
Learning parameters		
learning rate	[1e-06, 0.1]	
l2 regularizer	[1e-06, 0.1]	
batch size	128	
weight value estimation	Adam	
max. no. epochs	100	

Table 1: Hyperparamters spaces defined for the model selection step

For each task, the architectures and the learning parameters used for the FFNN model can be shown in table below:

FFNN	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Architecture				
No. of Dense Layers	2	1	2	3
No. of units layer 3				128 (ReLU)
No. of units layer 2	256 (ReLU)		128 (ReLU)	128 (ReLU)
No. of units layer 1	64 (ReLU)	128 (ReLU)	32 (act. ReLU)	16 (ReLU)
No. of units of output layer	1 (sigmoid)	1 (sigmoid)	1 (sigmoid)	1 (sigmoid)
Learning parameters				
learning rate	1e-03	1e-03	0.01	0.001
l2 regularizer	1e-06	1e-06	1e-06	0.001
batch size	128	128	128	128
weight value estimation	Adam	Adam	Adam	Adam
max. no. epochs	100	100	100	100

Table 2: FFNN models selected by the Bayesian Optimization procedure

In the case of CNN models the hyper-parameter spaces defined for the Bayesian Optimization procedure follow what has done in [1], but with an additional search on the probability of the dropout layers:

Layers	Type	Units	Kernel	Activation	Notes
3	1D Convolutional + Batch Norm	64	5	ReLU	-
1	Max Pooling 1D	-	-	-	Size 2
1	1D Convolutional + Batch Norm	{32, 64, 128}	{5, 10}	ReLU	-
1	Max Pooling 1D	-	-	-	Size 2
1	Flatten	-	-	-	
1	Dense	{16, 32, 64}	-	ReLU	
1	Dropout	-	-	-	[0.1, 0.5]
1	Dense	{16, 32, 64}	-	ReLU	
1	Dropout	-	-	-	[0.1, 0.5]
1	Dense	1	-	Sigmoid	-

Table 3: Spaces of the hyperparameters selected for the model selection of CNN architecture, here the learning parameters has been fixed based on what has been done in [1]

For each task, the CNN models used can be seen in this table:

Layers	Type	Units	Kernel	Activation	Notes
3	1D Convolutional + Batch Norm	64	5	ReLU	-
1	Max Pooling 1D	-	-	-	Size 2
1	1D Convolutional + Batch Norm	T1: 32	T1: 10	ReLU	
		T2: 32	T2: 5	-	-
		T3: 32	T3: 5	-	-
		T4: 32	T4: 5	-	
1	Max Pooling 1D	-	-	-	Size 2
1	Flatten	-	-	-	
1	Dense	T1: 16	-	ReLU	-
		T2: 32	-	ReLU	-
		T3: 32	-	ReLU	-
		T4: 32	-	ReLU	-
1	Dropout	-	-	-	T1: 0.5
		-	-	-	T2: 0.4
		-	-	-	T3: 0.5
		-	-	-	T4: 0.5
1	Dense	T1: 64	-	ReLU	-
		T2: 16	-	ReLU	-
		T3: 16	-	ReLU	-
		T4: 16	-	ReLU	-
1	Dropout	-	-	-	T1: 0.5
		-	-	-	T2: 0.4
		-	-	-	T3: 0.5
		-	-	-	T4: 0.5
1	Dense	1	-	Sigmoid	-

CNN Learning parameters	Values
learning rate	0.002
batch size	128
weight value estimation	Nadam
max. no. epochs	100

Table 4: CNN models selected by the Bayesian Optimization procedure

The MMNN models have been obtained by chaining the hidden layers of the CNN and FFNN models selected by the model selection step, the optimizer used for the MMNN models has been the **nadam optimizer** with a learning rate of 0.001.

2.4 Data pre-processing

For the epigenomic data, the pre-processing steps used has been:

- a normalization of the data, by using the RobustScaler function of the sklearn python library, in order to make a normalization not affected by outliers;
- an application of an imputation procedure in order to replace the possible missing values in the data, for such step it has been used the KNNImputer function of the sklearn library, where each sample's missing values are imputed using the mean value from n_neighbors nearest neighbors found in the training set, in the project the n_neighbors parameter has been set to the default value that is 5, with as distance metric the eucledian distance.

It has found that the total number of missing values in the promoters epigenomic dataset, for the cell line H1, are 19; instead for the enhancers epigenomic dataset the the total number of missing values were 1.

For the DNA sequence dataset just a one-hot encode of the single nucleotides has been performed.

2.5 Imbalance of the classes

The imbalance between the active and inactive classes on each task can be seen in the following table, the imbalance of the classes is very strong for the tasks related to the enhancers regions:

Tasks	**Imbalance Inactive/Active**	**No. samples**
IP vs AP (TPM $\geq$ 1.0)	0.74 / 0.26	99881
IP vs AP (TPM $\geq$ 0.5)	0.65 / 0.35	99881
IE vs AE (TPM $\geq$ 1.0)	0.99 / 0.01	63285
IE vs AE (TPM $\geq$ 0.5)	0.95 / 0.05	63285

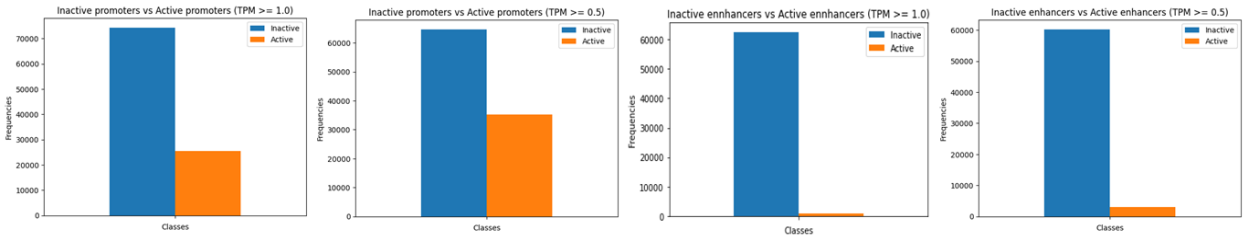


Figure 1: Bar plots of the imbalance between classes for each task

2.6 Data correlations and distributions

In the case of epigenomic data, correlations has been computed first between the features and the output feature, then between the features themself.

2.6.1 Correlations between features and output

For the correlations between features and output, after the application of the Pearson Correlation method a Spearmen Correlation method has been applied, in order to get which were the features with the lowest linear and monotonic correlation index with respect to the output feature, with related p-value; anyway the most uncorrelated features have been kept in order to be analyzed by the feature selection procedure explained on the "Feature Selection" section.

For each task, the top five uncorrelated features w.r.t the output feature are show below, with Pearson and Spearman correlation methods:

Table 5: IP vs AP (TPM $\geq$ 1.0)

Feature	**Pearson**	**p-value**
H3K79me2	-0.025	3.45e-15
REST	0.045	7.73e-47
H2AFZ	-0.06	1.15e-84
H3K4me2	0.064	2.70e-90
RFX5	0.072	1.71e-114

Table 6: IP vs AP (TPM $\geq$ 0.5)

Feature	**Pearson**	**p-value**
H3K79me2	-0.011	0.001
REST	0.044	2.95e-44
H2AFZ	-0.054	4.32e-66
RFX5	0.064	3.55e-120
MAFK	0.085	1.77e-159

Table 7: IE vs AE (TPM $\geq$ 1.0)

Feature	**Pearson**	**p-value**
RNF2	0.003	0.49
SUZ12	-0.01	0.14
H3K79me2	0.01	0.055
H3K27me3	-0.01	0.00
H3K4me1	0.02	3.11e-05

Table 8: IE vs AE (TPM $\geq$ 0.5)

Feature	**Pearson**	**p-value**
SUZ12	0.01	0.17
H2AFZ	-0.01	0.10
H3K27me3	-0.01	0.01
RNF2	0.014	0.00
H4K20me1	0.03	5.30e-16

Table 9: IP vs AP (TPM $\geq$ 1.0)

Feature	**Spearman**	**p-value**
H2AFZ	-0.07	1.26e-115
RNF2	-0.09	1.04e-162
H3K79me2	-0.09	8.19e-171
SUZ12	-0.16	0.0
MAFK	0.16	0.0

Table 10: IP vs AP (TPM $\geq$ 0.5)

Feature	**Spearman**	**p-value**
H2AFZ	-0.07	7.53e-99
RNF2	-0.07	9.76e-100
H3K79me2	-0.09	2.16e-188
SUZ12	-0.15	0.0
MAFK	0.17	0.0

Table 11: IE vs AE (TPM $\geq$ 1.0)

Feature	**Spearman**	**p-value**
SUZ12	-0.006	0.08
RNF2	0.01	0.02
SIX5	0.01	0.0
H3K4me1	0.02	1.6e-06
H2AFZ	-0.02	3.24e-08

Table 12: IE vs AE (TPM $\geq$ 0.5)

Feature	**Spearman**	**p-value**
SUZ12	0.00	0.36
H2AFZ	-0.01	0.0
RNF2	0.02	8.06e-05
SIX5	0.02	8.05e-05
H3K79me2	-0.02	1.10e-06

2.6.2 Correlations between features

For the correlations between features the linear correlation has been checked through the Pearson Correlation method, from the result obtained we can say that no features have a significant correlation between each other.

The most five correlated features, in absolute value, with the Pearson Correlation method can be seen in the following tables:

Table 13: IP vs AP features

Feature 1	**Feature 2**	**Pearson**	**p-value**
POLR2A	POLR2AphosphoS5	0.93	0.0
H3K27ac	H3K9ac	0.90	0.0
TAF1	POLR2A	0.89	0.0
SUZ12	RNF2	0.85	0.0
POLR2A	TAF7	0.83	0.0

Table 14: IE vs AE features

Feature 1	**Feature 2**	**Pearson**	**p-value**
POLR2A	POLR2AphosphoS5	0.88	7.53e-99
H3K4me3	H3K4me2	0.88	9.76e-100
CTCF	RAD21	0.86	2.16e-188
SUZ12	RNF2	0.82	0.0
SUZ12	H3K27me3	0.84	0.0

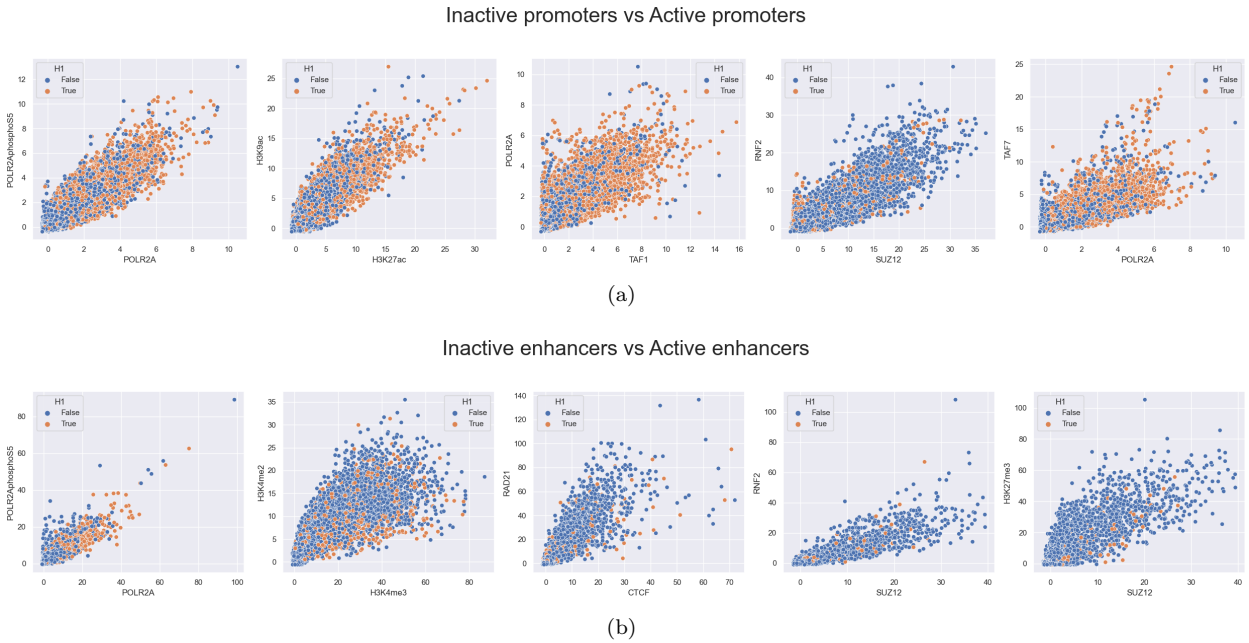


Figure 2: Scatter plot of the top five most correlated pairs of features: (a) Promoters dataset features correlation (b) Enhancers dataset features correlation

2.6.3 Data distributions visualization

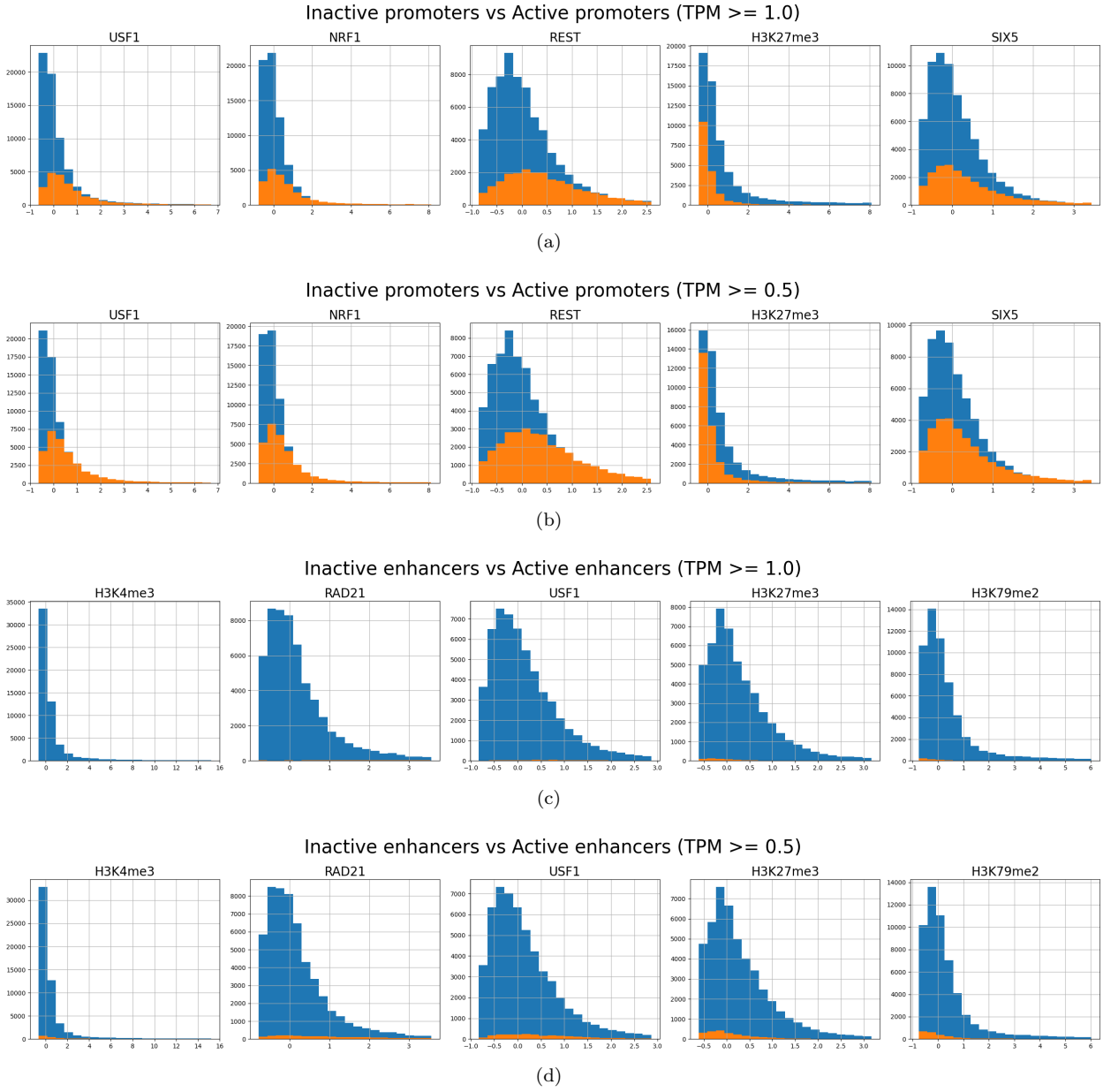


Figure 3: Histogram plot of distributions for the top five different features in the promoters and enhancers epigenomic dataset of the cell line H1: (a) Task 1 (b) Task 2 (c) Task 3 (d) Task 4

As can be seen there aren't distributions with visible separated classes, some distributions present some little offset between the classes.

2.7 Data visualization

2.7.1 Visualisation of the epigenomic data with PCA

Here we can see a visualisation of data at each task, for epigenomic data a PCA procedure has been applied to obtain the first two principal components, in the case of DNA sequence data an MFA decomposition has been applied.

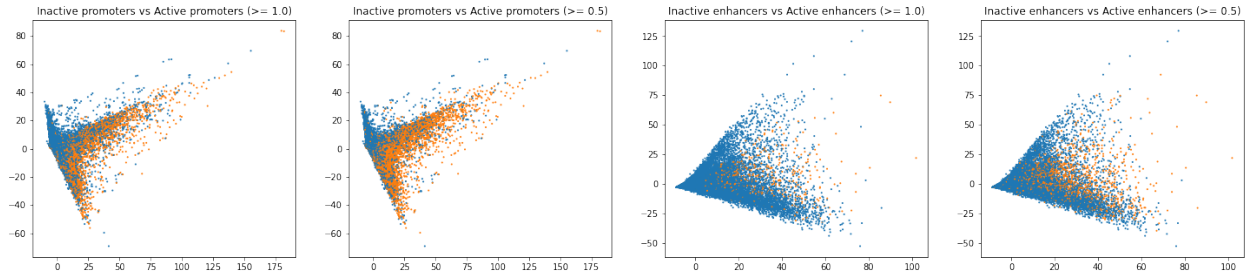


Figure 4: Decomposition in the first two principal components of epigenomic data, given the cell line H1, using PCA

2.7.2 Visualisation of the DNA sequence data with MFA

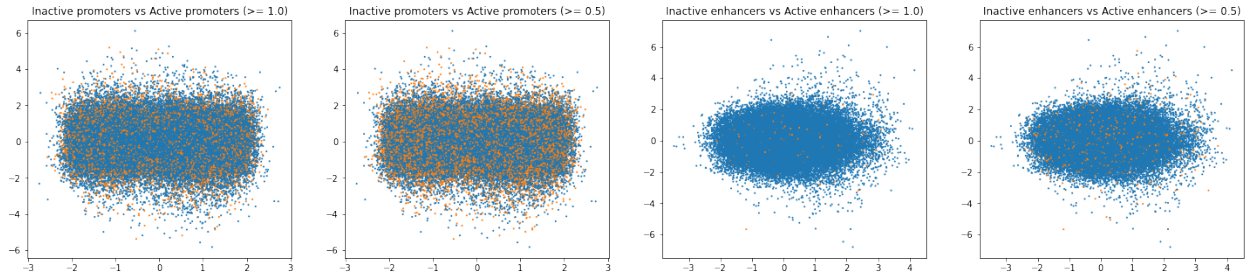


Figure 5: Decomposition in the first two principal components of DNA sequence data, given the cell line H1, using MFA

2.8 Feature Selection

In the evaluation of the FFNN models a feature selection procedure has been applied over the epigenomic data by using the python implementation of Boruta, a wrapper around a Random Forest classifier algorithm, that tries to find all features carrying information usable for prediction. The feature importances are derived from the Gini impurity function, then it decides what are the features that should be rejected at each iteration.

The feature selection procedure has been applied first on the whole dataset and then has been executed in each of the holdout generated, using the training data of the holdout. Main parameters set on the BorutaPy function:

Parameter	**Value**	**Notes**
estimator	RandomForrestClassifier	A supervised learning estimator, with a 'fit' method that returns the feature_importances_ attribute
max_depth	5	max depth of the ensemble of trees generated by the estimator
alpha	0.05	Level at which the corrected pvalues will get rejected in the correction steps.
max_iteration	200	The number of maximum iterations to perform.
n_estimators	'auto'	the number of estimators is determined automatically based on the size of the dataset.

Table 15: Paramters set for the feature selection procedure

Running the feature selection on the whole given epigenomic dataset results that on the 58 features available no features have been rejected on the first two tasks related to the promoters dataset, meanwhile, on the tasks related to the enhancers epigenomic dataset resulted that on the 58 features available 2 features have been rejected.

2.9 Holdouts

All classification tasks are executed using 10 randomly generated holdouts, each composed by splitting the data set into training, containing 80% of samples, and test set, containing 20% of samples. Classification tasks involving model selection were performed using 1 holdout with the same proportion, 80% for the training set and 20% of samples for the test set. The holdouts have been generated using the StratifiedShuffleSplit function of the sklearn library, in order to obtain randomized folds that preserve the percentage of samples for each class.

2.10 Results analysis

Performances are measured by using: the classification accuracy, the Area Under the Receiver-Operating Curve (AUROC) and and Area Under the Precision-Recall Curve (AUPRC); metrics computed over all training and test sets in the 10 holdouts. On FFNN models have been evaluated also the performances over holdouts having data with features selected by the Boruta algorithm.

Performances of the three types of models used on each task were compared by applying Wilcoxon signed rank tests (at 0.01 significance level) to detect statistically significant differences in the mean values of the classifiers' Accuracy, AUPRC and AUROC.

2.11 Results

In all the tasks the CNN models have been outperformed by both the FFNN and MMNN models, such significant differences have been validated by the Wilcoxon test at each task. Between the FFNN and the MMNN mode, in most of the tasks the performances has been resulted statistically indistinguishable: on the second task (IP vs AP (≥ 0.5)), the MMNN model slightly outperformed the FFNN model on the AUPRC metric, on the third task (IE vs AE (≥ 1.0)), the FFNN model outperformed the MMNN model on the AUROC metric.

The results on the accuracy metric are strongly affected by the imbalance of the classes, in particular for the enhancers regions; we can also observe that by the lowering the TPM threshold for the binarization of the activation status there is an improvement on the AUPRC metric results, probably due to the fact that there is a less imbalance of the classes in the data.

IP vs AP ($\geq$ 1.0)	**Accuracy**	**AUROC**	**AUPRC**
CNN	0.74	0.75	0.45
FFNN	**0.81**	**0.86**	0.64
MMNN	**0.81**	**0.86**	**0.65**
IP vs AP ($\geq$ 0.5)	**Accuracy**	**AUROC**	**AUPRC**
CNN	0.70	0.75	0.59
FFNN	**0.79**	**0.85**	0.73
MMNN	**0.79**	0.84	**0.74**
IE vs AE ($\geq$ 1.0)	**Accuracy**	**AUROC**	**AUPRC**
CNN	0.99	0.70	0.07
FFNN	0.99	**0.91**	0.21
MMNN	0.99	0.84	**0.23**
IE vs AE ($\geq$ 0.5)	**Accuracy**	**AUROC**	**AUPRC**
CNN	0.95	0.66	0.24
FFNN	0.95	**0.80**	**0.33**
MMNN	0.95	0.79	0.32

Table 16: Mean results of the classifiers' Accuracy, AUPRC and AUROC

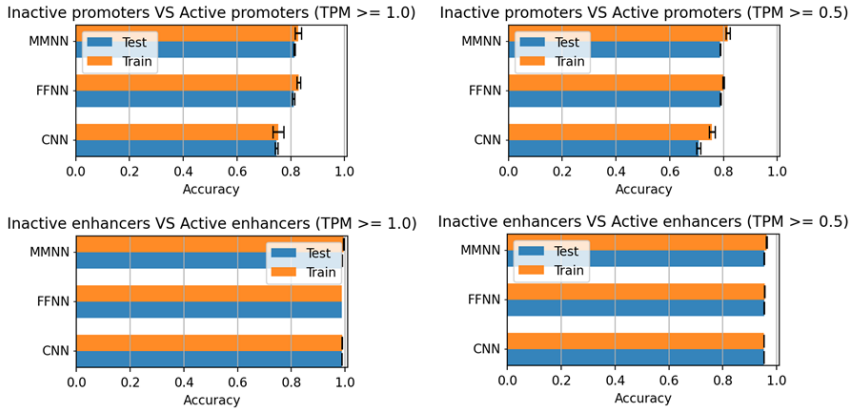
Here we can see the mean values performances obtained by using epigenomic data on FFNN models, with and without the feature selection step on each holdout; no significant differences have been recognized through the Wilcoxon test:

IP vs AP ($\geq$ 1.0)	**Accuracy**	**AUROC**	**AUPRC**
FFNN with no feature selection	0.81	0.86	0.65
FFNN with feature selection	0.81	0.86	0.65
IP vs AP ($\geq$ 1.0)	**Accuracy**	**AUROC**	**AUPRC**
FFNN with feature selection	0.79	0.85	0.73
FFNN with no feature selection	**0.80**	0.85	0.73
IE vs AE ($\geq$ 1.0)	**Accuracy**	**AUROC**	**AUPRC**
FFNN with no feature selection	0.99	0.92	0.20
FFNN with feature selection	0.99	0.92	**0.22**
IE vs AE ($\geq$ 0.5)	**Accuracy**	**AUROC**	**AUPRC**
FFNN with no feature selection	0.95	0.80	0.33
FFNN with feature selection	0.95	0.80	0.33

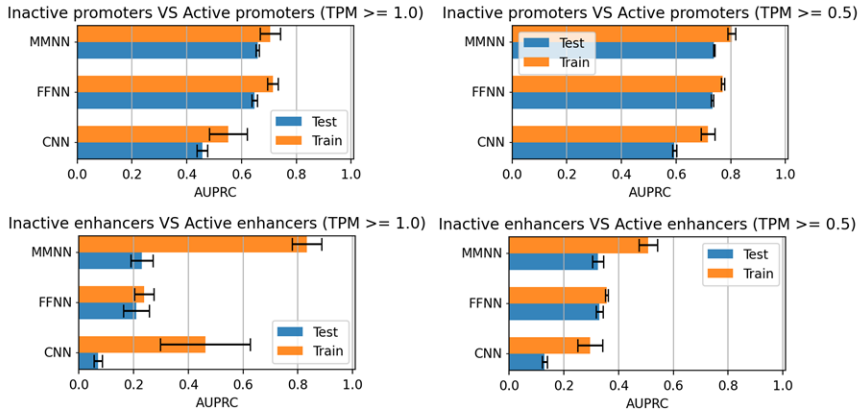
Table 17: Mean results of the FFNN classifiers' Accuracy, AUPRC and AUROC with and without feature selected data

2.11.1 Visualisation of the performances

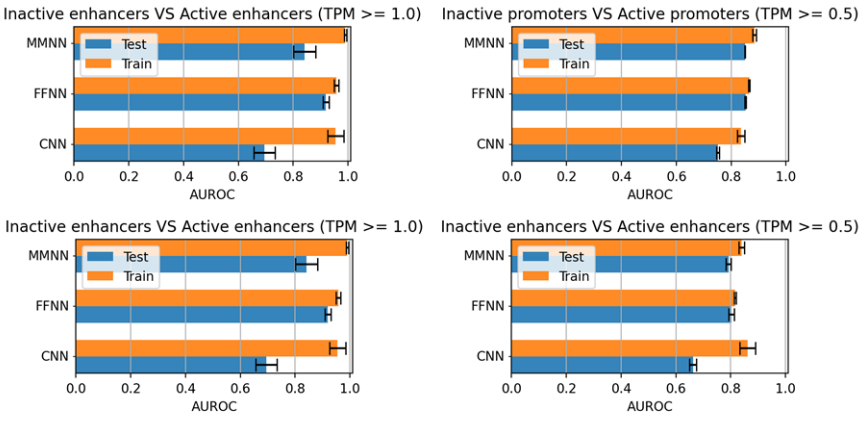
Here we can see a visualisation, by bar plots, of the mean performances obtained also on the training set:



(a)



(b)



(c)

Figure 6: Bar plots of the mean values of the performances obtained by the selected models at each task: (a) Accuracy (b) AUROC (c) AUPRC

References

[1] Luca Cappelletti, Alessandro Petrini, Jessica Gliozzo, Elena Casiraghi, Max Schubach, Martin Kircher, and Giorgio Valentini. Bayesian optimization improves tissue-specific prediction of active regulatory regions with deep neural networks. In Ignacio Rojas, Olga Valenzuela, Fernando Rojas, Luis Javier Herrera, and Francisco Ortuño, editors, *Bioinformatics and Biomedical Engineering*, pages 600–612, Cham, 2020. Springer International Publishing.

[2] Yifeng Li, Wenqiang Shi, and Wyeth W Wasserman. Genome-wide prediction of cis-regulatory regions using supervised deep learning methods. BMC Bioinformatics, 2018.

[3] Melissa Woghiren, Yifeng Li, and Alioune Ngom. Prediction of cis-regulatory genomic elements using machine learning tools.