Prediction of the regulatory regions active in the cell line "A549" through deep learning methods

Vincenzo Conversano
Department of Computer Science
University of Milan, Milan, Italy

1 INTRODUCTION

This paper belongs to the Bioinformatics field and presents a solution for predicting the active regulatory region in the immortalized cell line "A549". Since 98% of the genome is composed of non-coding regions (i.e. regions that do not contain instructions for creating proteins) in the past these parts were called "junk" DNA [1]. During the last decade, it was shown that non-coding regions are extremely important, also for understanding connections with human diseases. Specifically, the E. P. consortium stated in [2] that 80% of the human genome has one or more biochemical functions.

A Cis-Regulatory Region is just a region of the genome that regulates the expression of another region in the genome. We have different types of regulatory regions:

- Enhancers: they are distal (up to 1 Mbp) from the gene and can be located before and after it. These sequences can be bounded by proteins to enhance the transcription process of that gene.
- Silencers: they are distal from the gene and can be located before and after it. These sequences are responsible for reducing the transcription process.
- Promoters: these sequences are attached to the side of the gene. Relevant proteins (such as RNA polymerase and transcription factors) will activate the promoter and initiate the transcription of that gene. The resulting transcription produces an RNA molecule (such as mRNA)

An important Bioinformatics problem in the last 20 years has been the detection of Cis Regulatory Regions (CRRs) and the prediction of their activity. Indeed, there are many works in the field. However, most of them consider either epigenomic data from ENCODE [2] either Sequence data of nucleotides.

Even though from the FANTOM5 [3] dataset we have access to 250 cell lines labels relative to Cis-Regulatory Regions, we only have a limited set of epigenomic data available from ENCODE [2]. This work will exclusively consider "A549" cell line but can be easily extended to other six cell lines available in epigenomic_dataset [1].

The A549 cell line was established in 1972 by D.J. Giard, et al., through an explant culture of adenocarcinomic lung tissue of a 58-year-old Caucasian male. This adenocarcinomic cell line is categorized as a non-small-cell lung carcinoma (NSCLC), which tends to be less aggressive and spread less quickly than small cell lung carcinoma (SCLC) but proves to be more common, accounting for 85-88% of all cases of lung cancer.

Motivated by the above, this work a) addresses the problem of classifying Cis Regulatory regions (promoters and enhancers), b) considers both Epigenomic and Sequence data, c) is based exclusively on reliable and publicly available data ensuring reproducibility, d) optimizes a series of classifiers to the specific problem, the learning of which does not require enormous data quantities and e) proposes a Multi-Modal framework combining the benefits of heterogeneous types of data.

[1]https://github.com/AnacletoLAB/epigenomic_dataset

2 MACHINE LEARNING MODELS

This section describes the considered models used for the classification tasks. Aiming at a Multi-Modal framework able to benefit from the two modalities of data, we started considering two commonly used Deep Learning models: Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN). And then concatenated them in a Multi-Modal Neural Network (MMNN).

Since MLPs are suitable for tabular data, we employed them for processing Epigenomic data. On the other hand, CNNs work well with data that has a spatial relationship and is non-tabular. Thus, we employed them for processing Sequence data encoded in the one-hot format.

As a plus, we considered a Random Forest [4] working on Epigenomic features and compared its performances with the Neural Networks.

2.1 Multilayer Perceptron

A Multilayer Perceptron network employs multiple perceptrons distributed in a multi-layered scheme where the technique of Backpropagation is applied to update the weights and learn the characteristics of the training set [5]. The layers that are usually found in MLPs are:

- Input Layer [2]: this layer is composed of as many input neurons as the number of features. The goal of this layer is to bring the inputs into the system.
- Dense Layer [3]: the actual computation of the MLP happens in these layers. It is composed of n hidden neurons and each one performs a weighted summation of its inputs and forward the output to neurons in the successive layer.
- Dropout Layer [4]: this layer randomly shut down some outputs of the previous layer - during the training phase - in order to cut down the network complexity. It is considered one of the most basic techniques to avoid overfitting in neural networks.
- Activation Layer [5]: this layer applies a specified activation function to the output of the previous layer. In this work, we considered: a) ReLU function for the hidden layers b) Sigmoid function for the output layer.

2.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a classical type of Artificial Neural Network whose model is inspired by the human visual cortex. The layers that are commonly found in a CNN are:

- Input Layer [6]: this layer is composed of as many input neurons as the number of features. The goal of this layer is to bring the inputs into the system.

[2]https://www.tensorflow.org/api_docs/python/tf/keras/layers/InputLayer
[3]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
[4]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
[5]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
[6]https://www.tensorflow.org/api_docs/python/tf/keras/layers/InputLayer

- Conv1D Layer [7]: this layer is the hallmark of CNNs. It creates a convolution kernel that is convolved with the input of the layer over a single dimension.
- MaxPool1D Layer [8]: this layer downsamples the input representation by taking the maximum value over a spatial window of the specified size.
- AveragePooling1D Layer [9]: this layer downsamples the input representation by taking the average value over a spatial window of the specified size.
- Activation Layer [10]: this layer applies a specified activation function to the output of the previous layer. In this work we considered: a) ReLU function for the hidden layers b) Sigmoid function for the output layer .

2.3 *Multi-modal Neural Network*

A Multi-modal Neural Network (MMNN) is just a neural network that combines data coming from different modalities (in this case DNA Sequence data and Epigenomic data). It does the job by truncating the output layers of two parallel Neural Networks (in this case an MLP and a CNN) and by merging them exploiting a Concatenate[11] Layer. Since this kind of network exploits the two sources of information we are expected to get more reliable results.

2.4 *Random Forest*

The specific classifier creates multiple weak-learners (that demonstrate high variance but low bias) and combine them in order to robustify the achieved predictions [4]. The Random Forest algorithm fits a plethora of Decision Trees (DT), typically on different bootstrap samples, while each tree elaborates on a random subset of features.

2.5 *Models hyper-parameters*

The weights of the MLPs and the filters values of the CNNs are learnt by the learning algortihm directly from data, exploiting back-propagation and a Gradient Descent algorithm (in this work, Nadam [6]), in order to minimize a loss function (in this work, Binary Cross Entropy). In addition to these parameters, Neural Networks have a lot hyper-parameters that are not learnt from the data and one should properly tune.

The hyper-parameters that a generic Neural Network has are:

- Depth of the network: total number of layers.
- Architecture of the network: how layers are organized in the network.
- Number of units: it is the number of perceptrons of a Dense layer.
- Number of filters: it is the number of filters of a Convolutional layer.
- Dropout rate: rate to which a Dropout layer discards information in order to avoid overfitting.
- Learning rate: it is a scalar that determines how much we are adjusting the weights at each step.
- Decay: it modifies the learning rate over time.

On the other end, the typical parameters that we can find in a Random Forest are:

- N-estimators: the number of DTs.
- Max depth: the maximum depth of each DT. It should be noted that growing very deep trees can lead to overfitting.

[7]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D
[8]https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool1D
[9]https://www.tensorflow.org/api_docs/python/tf/keras/layers/AveragePooling1D
[10]https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
[11]https://www.tensorflow.org/api_docs/python/tf/keras/layers/concatenate

Details about the parameterization approaches and the architectures we used are described in the subsection 3.5.

3 EXPERIMENTAL SETUP

This section describes a) the considered tasks; b) how is the dataset composed; c) the data preprocessing step; d) correlations and data distributions; e) the feature selection step; f) data visualization; g) the creation of the holdouts; and h) how results have been studied.

3.1 *The considered tasks*

In this work we will develop models that given a region of DNA (described by Epigenomic data and Sequence data from HG38 dataset) are able to discriminate between:

- Active enhancers vs inactive enhancers.
- Active promoters vs inactive promoters.

3.2 *The dataset*

The dataset has been created by aggregating data coming from three sources of information:

- FANTOM5 [3] provided **Cis regulatory regions labels** and **regions' coordinates** in BED format[7].
- ENCODE [2] provided **epigenomic data** for the considered Cis regulatory regions. Specifically, they used the CAGE protocol [8] and for each target (protein or gene) they measured the level of CAGE expression in Tags per million.
- UCSC genome browser [9] [10] provided **Nucleotide sequences** for the promoters and enhancers regions.

FANTOM [3] is an international research consortium focused on functional annotation of mammalian genomes and characterization of transcriptional regulatory networks [11]. In FANTOM5 the consortium has unraveled the biggest collection of promoters and enhancers in mammalian primary cells.

The goal of ENCODE [2] is to build a comprehensive parts list of functional elements in the human genome, including elements that act at the protein and RNA levels, and regulatory elements that control cells and circumstances in which a gene is active.

UCSC genome browser [9] [10] is an interactive website offering access to genome sequence data from a variety of vertebrate and invertebrate species and major model organisms, integrated with a large collection of aligned annotations.

In order to retrieve data from these sources we used the following python packages:

- epigenomic_dataset[12]: *bigWig* files are generally used to store common ENCODE data (for CAGE experiments it is the most common format). These files contain a lot of None Values and therefore they are very noisy. The epigenomic_dataset package solves this problem by averaging these values and returns already pre-processed epigenomic data.
- ucsc_genomes_downloader[13]: this package is useful for rapidly download the genome of the provided assembly from UCSC. We will be interested in retrieving the human genome sequence (HG38).

3.3 *Data Pre-processing, Correlations, and Distributions*

The data pre-processing step is one of the most important steps in a common Machine Learning pipeline. In this work, we considered the following sub-steps.

[12]https://github.com/AnacletoLAB/epigenomic_dataset
[13]https://github.com/LucaCappelletti94/ucsc_genomes_downloader

3.3.1 NaN Values Imputation: Since the epigenomic data is obtained by averaging genomic regions from *bigWig* files, if we have a NaN value in the epigenomic_dataset, this means that for the considered region of 256 nucleotides (or other window sizes) we have no values available. For imputing these values we used a KNN imputer [14], with $k = 5$, exploiting its sklearn implementation.

3.3.2 Data Normalization: Performing data normalization is an essential step before performing feature selection and dimensionality reduction. Since the Epigenomic data is full of noise, we decided to use the robust variant of Z-score normalization. In this variant, we subtract the median from the data and divide by the inter-quartile range value. In this way our data will have $median = 0.0$ and $iqr = 1.0$.

3.3.3 Labels Clipping: Labels of Cis Regulatory Regions can be noisy. This is usually caused by errors in the CAGE experiments. Hence, to avoid outliers bias our analysis and the models learning we decided to clip all the label values to the first integer greater than the 0.99 quantile.

3.3.4 Labels Binarization: The task of predicting the level of activation of an enhancer/promoter region is a regression task. Indeed the level of activation of a region is a real value expressed in TPMs. However, methods that consider the task exclusively as a regression one are limited in terms of performance evaluation. Thus, the need of choosing the right cut-off for defining a region active arose. Many researchers addressed the problem but there is not a definite answer yet.

The FANTOM5 authors suggested using a threshold of 1 TPMs for both promoters and enhancers, after having dropped all rows that have a label between 0 and 1. This setup removes a portion of the samples, but removes the issue of the threshold. However, in the cell line A549 this approach generates an enhancer dataset with an Imbalance Ratio ($\frac{regions_-}{regions_+}$) of 270.60. With such a high rate there is not much we can do, thus we decided to avoid dropping values and consider still a threshold of 1 TPMs for promoters and a tentative threshold of 0.0 TPMs for enhancers, in order to have - as shown in Figure 1 - a more learnable enhancer dataset ($\frac{regions_-}{regions_+} = 16.56$). On the other hand, the promoter data has an Imbalance Ratio of 3.9.

Moreover, in this work, Set balancing approaches such as SMOTE [12] were not considered because, on one hand, we consider the procedure of biological data generation too complex to be mimicked by oversampling approaches, on the other hand, through undersampling we would discard the vast majority of datapoints. Indeed, it has been shown by Cappelletti et al. in [13] that Set Balancing in a very similar setting leads to over-optimistic results.

3.3.5 Feature Correlations: It's very common in epigenomic data to find features that a) are not correlated with the output label; b) are correlated with other features (features' redundancy).

Firstly, we studied the correlations of the features with the output by using Pearson (studying linear correlations) and Spearman (studying monotonic correlations) tests. From this step, we found out that the sets of features: {'CBX2', 'KDM5A', 'ZC3H11A'} and {'FOSB', 'ZC3H11A'} are not correlated respectively with the label values of enhancers and promoters regions, with a statistical significance level of 0.01 i.e. ($p - value < 0.01$).

Secondly, we studied the correlations between features, again, by using Pearson (studying linear correlations) and Spearman (studying monotonic correlations) tests. From this step, we did not find any pair of features that are extremely correlated with each other ($correlation > 0.85$) , with a statistical significance level of 0.01 i.e. ($p - value < 0.01$).

Last but not least, we run the MIC test [14] on the sets of features found in step one in order to discover any non linear correlation of those features with the output and avoid discarding important features. According to the MIC test, none of those features exhibit any non linear correlation with the output, thus we proceeded to discard them.

Despite the features seems to be not highly correlated with each other, we decided to visually confirm that by plotting the scatter plots of the 3 most correlated features pairs for both promoters (Figure 2) and enhancers (Figure 3). The majority of the plots have the typical 'L' shape that suggests no correlation between the features pair. However, there are many pairs, such as (SMC3, CTCF), that seem to be correlated, these are pairs that have a moderately high correlation score but not greater than 0.85.

3.3.6 Features Distributions: The distributions of the 5 most dissimilar features are shown in Figure 4 and Figure 5.

3.3.7 Feature Selection: Selecting the right subset of features is an important step to a) reduce the computational complexity; b) reduce the noise and create cheaper models; c) obtain more interpretable features.

The most common approaches to feature selection are Wrapper and Filters. Filters methods, such as minimum Redundancy Maximum Relevance Feature Selection (mRMR) [15], are based on mutual information and statistical tests (t-test, F-test) and are particularly light computationally. Specifically, mRMR method select features incrementally by requiring that they are maximally dissimilar to each other (minimum redundancy) and they provide the maximal mutual information with the target output (maximal relevance). The only con of mRMR is that we need to specify the number k of features to be selected.

On the other hand, Wrapper methods, such as Boruta [16], exploit a learning algorithm (in Boruta case, Random Forests) to give importance to a certain combination of features directly by using accuracy. In general, these methods are heavier computationally.

In this work we used both Boruta and mRMR, the considered implementations respectively are BorutaPy[15] and mrmr_selection[16]. Specifically, we run a 5 split StratifiedShuffleSplit[17] cross validator to create 5 random stratified folds and then run on each one Boruta and mRMR (with $k = |BorutaSet|$) algorithms. In promoters data, Boruta algorithm considered relevant all the features over the 5

[14]https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

[15]https://github.com/scikit-learn-contrib/boruta_py

[16]https://github.com/smazzanti/mrmr

[17]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffle

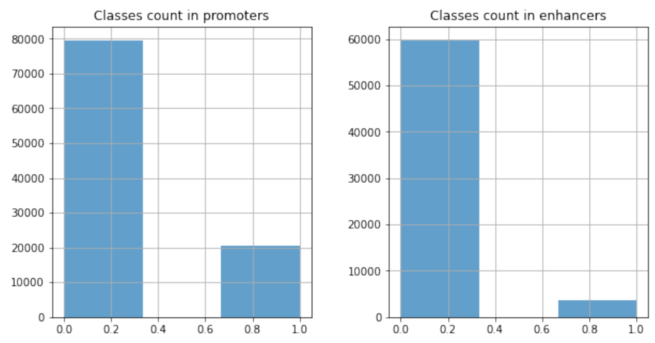

Fig. 1: Classes count.

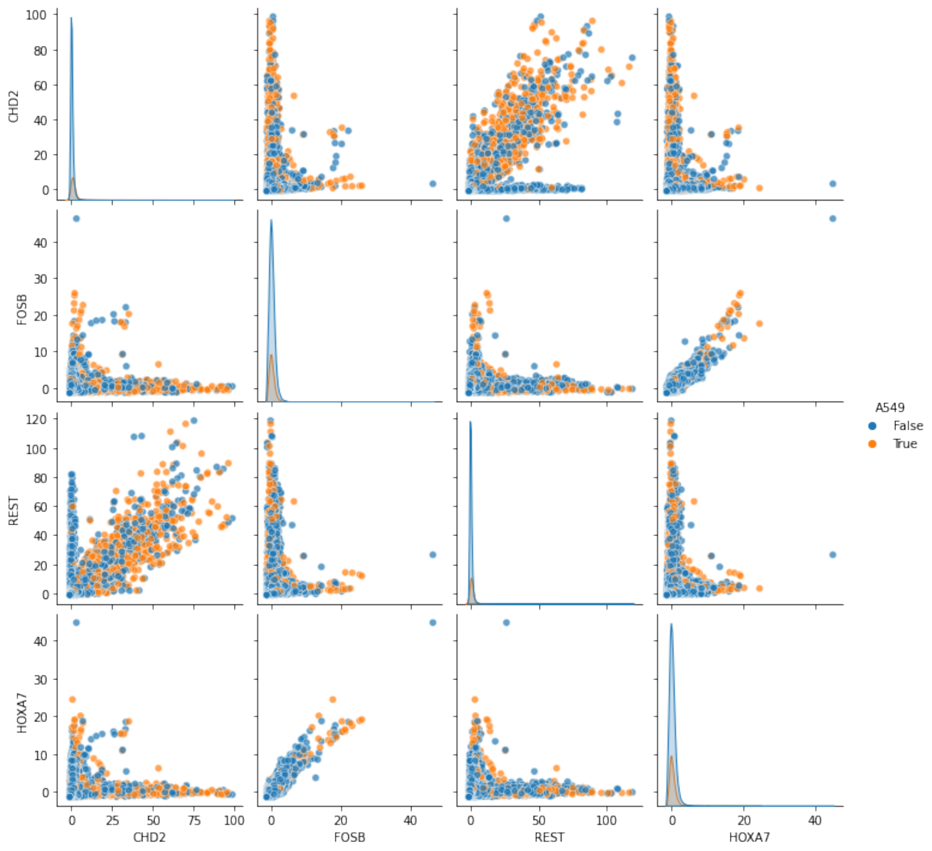

Fig. 2: Scatter plots of the three most correlated feature pairs in promoters' regions.

stratified holdouts, thus, mRMR algorithm will also do that since $k = |BorutaSet|$. We proceeded using the set of stable features as it is. In enhancers data, the sets of selected features by Boruta and mRMR are stable (35 features are always selected by both methods) over the 5 stratified holdouts. Then, we considered the intersection set stable and used it as selected features.

3.4 *t-SNE and PCA based Data Visualization*

To visualize data we obviously need to use a decomposition technique that reduces the feature space to a 2 dimensional space. To do so we resort on t-SNE and PCA techniques and used their sklearn implementations[18] [19]. The data visualizations are shown in Figure 6 and Figure 7. It is worth observing that a) no cluster appear from the decompositions, b) sequence data decomposition appears more "mixed" than the epigenomic one, c) in this context t-SNE works better than PCA.

3.5 *Models Parameterization*

This section describes the parameterization process of the Models. Specifically, we used two main approaches: 1) Using Fixed Models; 2) Using Meta Models.

3.5.1 Fixed Models: In this work, we decided to employ the architectures and the hyper-parameters setting proposed in [17] and [18], with which authors addressed the task of predicting labels for Cis-regulatory regions in GM12878, HepG2, K562 cell lines. The architectures and hyper-parameters settings are shown in Tables III II, whereas the settings of MMNN are not shown because its architecture is just the union of the MLP and the CNN with a last layer for merging, while its learning parameters are the same as the CNN.

TABLE I: Fixed MLP parameters.

Layer	Units	Activation
Dense	16	ReLU
Dense	4	ReLU
Dense	2	ReLU
Output	1	Sigmoid

Learning parameter	Value
learning rate	0.5
decay	0.1
batch size	32
optimizer	SGD
Max. no. of epochs	64

3.5.2 Meta Models: We tuned the Models architectures and hyper-parameters using BayesianOptimization [19] for the Neural Networks and GridSearch for the Random Forests. In order to do that properly we realized an External 5 fold Monte Carlo Cross-Validation and tuned the hyperparameters on an internal sub-split of the training set (more details in sub-section 6.). In this way, we do not reach the optimal models (for doing that we would need a Nested CV) but we get statistically reliable results. The tools we used for doing tuning are 1) keras-tuner for BayesianOptimization [20] and 2) hypopt for GridSearch [21].

[18]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[19]https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[20]https://keras.io/keras_tuner/
[21]https://github.com/cgnorthcutt/hypopt

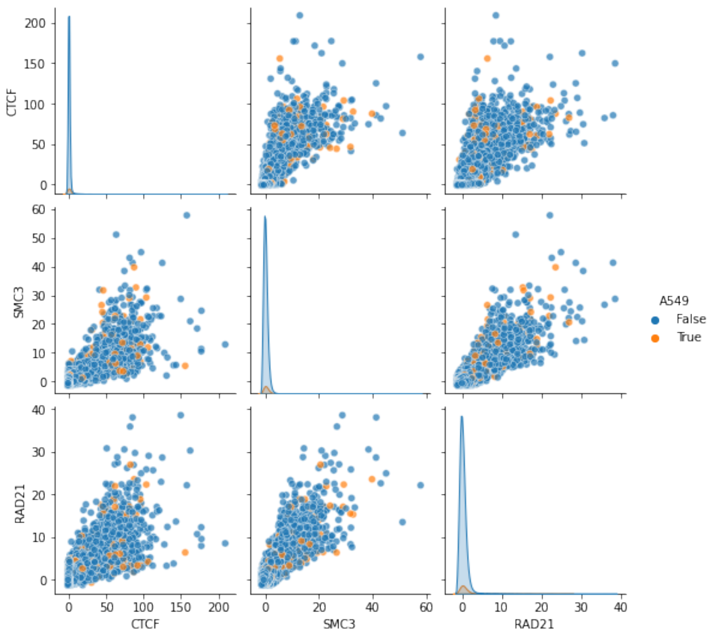

Fig. 3: Scatter plots of the three most correlated feature pairs in enhancers' regions.

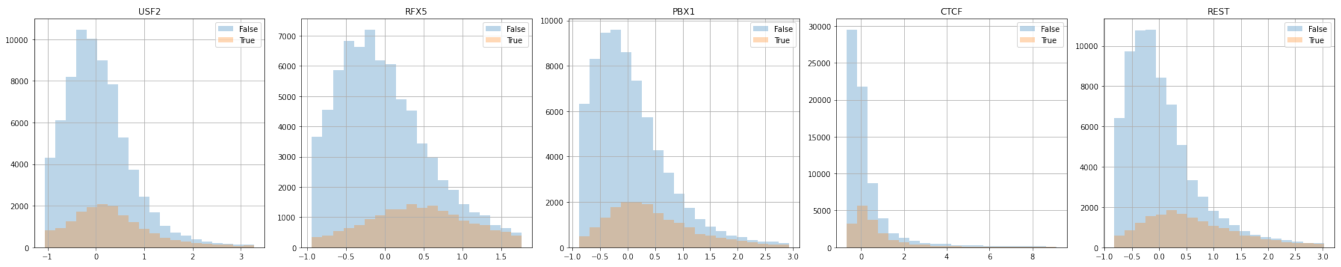

Fig. 4: Feature distributions in promoters regions.

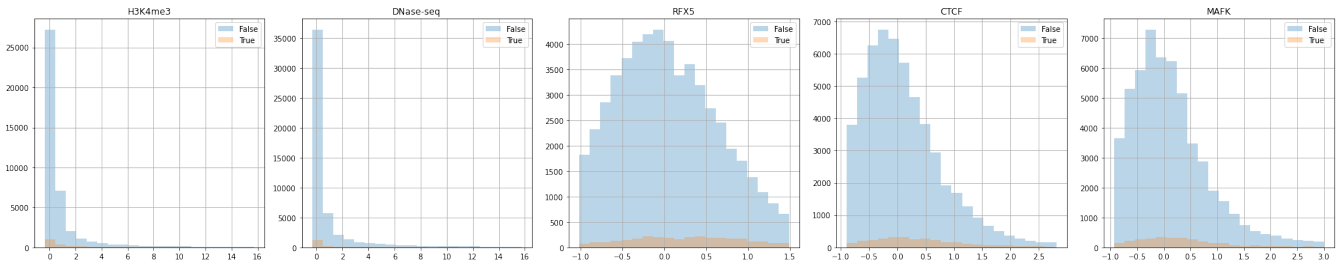

Fig. 5: Feature distributions in enhancers regions.

3.6 Holdouts

Fixed Models are evaluated through a Monte Carlo cross validation technique (10 holdouts), using in each holdout 80% of data for training and 20% of data for testing.

On the other hand, for Hyper Models we realized an external Monte Carlo cross validation (5 holdouts) and an tuned the hyperparameters on a subsplit of the training set. Specifically, at each iteration the data is splitted in the following way:

- 65 % of data for training
- 15 % of data for validation (sub-split of training)
- 20 % of data for testing

Hyper Models are optimized on the validation set and then evaluated on the test set. In order to obtain, statistically correct results and at the same time do not depend on the specific training set we should run a Nested Cross Validation procedure, but this is not considered in this work because it is too computationally heavy.

The Holdouts have been realized using StratifiedShuffleSplit.

3.7 Result Analysis

The metrics used to evaluate Models performances were chosen taking into account the highly imbalanced settings in which we are. Indeed, metrics such as Accuracy or Specificity are not considered. Hence, we used:

- F1-Score: harmonic mean of the specificity and recall;
- AUPRC: Area Under Precision Recall Curve;
- AUROC: Area Under Receiver Operating Characteristic Curve.

Then we plotted the average performances of all the Models and compared them with each other.

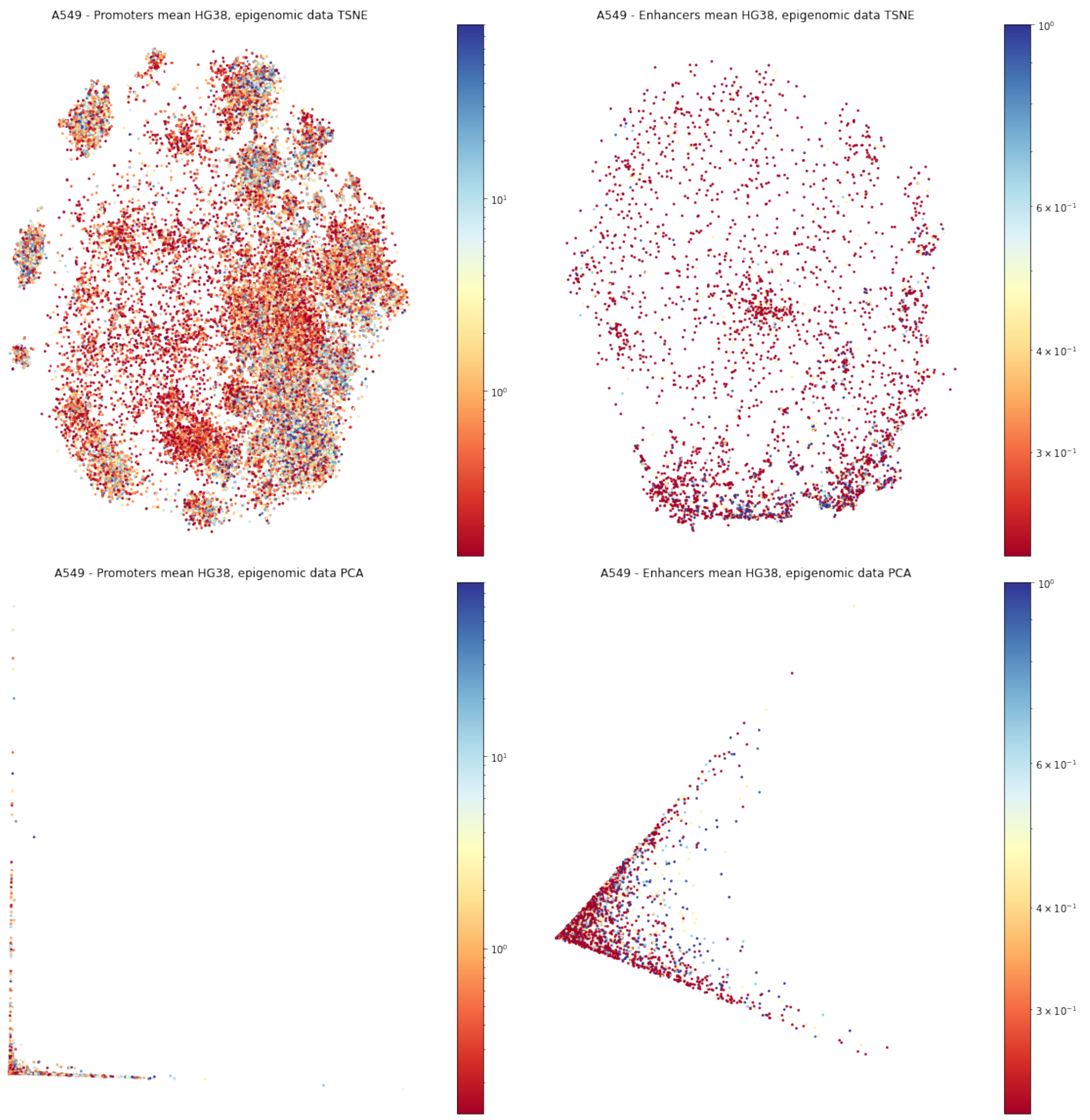

Fig. 6: Epigenomic Data visualizations of A549 cell line.

TABLE II: Fixed CNN parameters.

Layers	Type	Units	Kernel	Activation	Notes
3	Conv1D	64	5	ReLU	-
1	MaxPool1D	-	-	-	size=2
3	Conv1D	128	3	ReLU	-
1	MaxPool1D	-	-	-	size=2
3	Conv1D	128	3	ReLU	-
1	GlbAvgPool1D	-	-	-	-
1	Dropout	-	-	-	$P = 0.1$
2	Dense	10	-	ReLU	-
1	Dropout	-	-	-	$P = 0.1$
1	Output	1	-	Sigmoid	-

Learning parameter	Value
learning rate	0.002
batch size	256
optimizer	Nadam
Max. no. of epochs	100

TABLE III: Fixed RF parameters.

Learning parameter	value
max_depth	5
class_weight	balanced_subsample

Aiming at a reliable comparison, the distributions of the metrics - obtained by the different algorithms/approaches - in the various holdouts were compared using Wilcoxon Test. If the test produces a $p - value < 0.01$ - where 0.01 is the threshold we considered for statistical significance - we can reject null-hypothesis and thus one of the two compared algorithms/approaches performed better. Specifically, we applied the test to 1) All Models pairs combinations for every task; 2) Models working with Feature Selection vs Models working on the original set; 3) Fixed Models vs their corresponding Hyper Model.

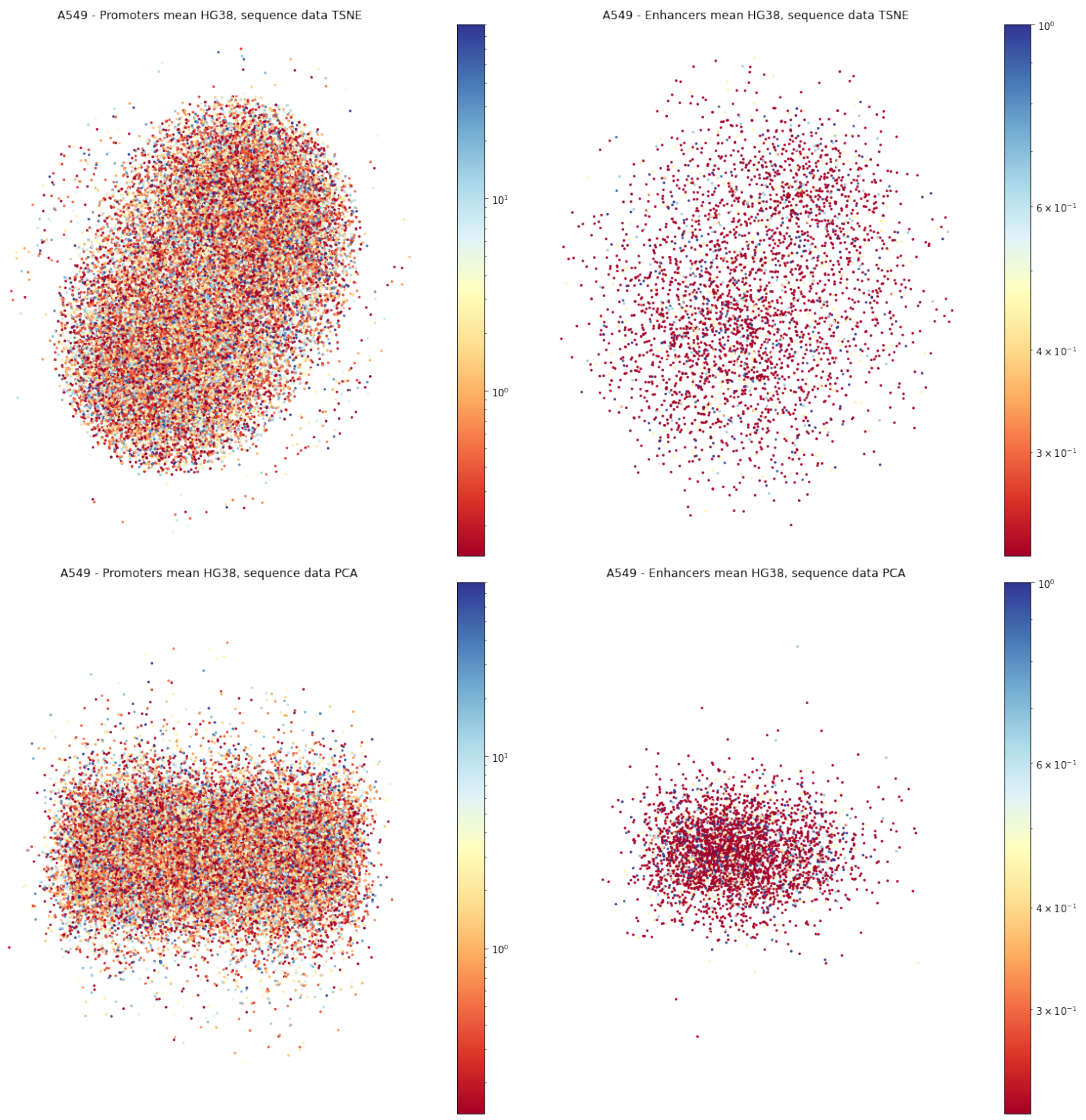

Fig. 7: Sequence Data visualizations of A549 cell line.

TABLE IV: Meta MLP parameters.

Layer	Search space	Activation
No. of layers	1,2,3,4,5	-
Dense	32,..,128	ReLU
Dropout	0.0,...,0.9	ReLU
Output	1	Sigmoid
Learning parameter	**Search space**	
learning rate	0.00001,..,0.001	
decay	0.0	
batch size	32	
optimizer	Nadam	
Max. no. of epochs	1000	

4 RESULTS

In this section, we report the experimental results and discuss comparisons between Models. In order to have a general idea of their performances we showed in Figures 8, 9, 10, 11, 12, 13 how each one performed on average on Train and Test holdouts.

First, we observe that the Fixed CNN is over-fitting in each task, this may be due to two not mutually exclusive reasons: 1) the network is too complex for the task 2) the information contained in Sequence data is not sufficient to determine the corresponding label.

Second, we observe that the Fixed Random Forest (working only with Epigenomic Data) is able to compete against MMNN and - as confirmed by Wilcoxon tests (Tables VII VIII IX) - in some situations it outperforms the other models. This suggests us that classical Machine Learning models should be considered for the tasks, earning more interpretable results and models that are easier to tune with respect to Neural Networks.

Thirdly, we observe that Models working with Feature Selected Sets are not distinguishable from Models not using Feature Selection (Table XIII). This may be the result of the fact that the starting Sets of features were not big and thus most of features were kept by Feature Selection algorithms.

Fourthly, we observe that there are no MetaModels emerging

TABLE V: Meta CNN parameters.

Layer	Search space	Activation
No. of layers	1,3,5	-
Conv1D	-	ReLU
Conv1D	filters: 32,...,128	-
Conv1D	kernel: 3,...,6	-
Dropout	P: 0.3,...,0.9	ReLU
Max/AvgPool	2	-
GlobalAvgPool	2	-
No. of layers	1	-
Dense	units: 16,..,64	ReLu
Dropout	P: 0.3,...,0.9	ReLU
Output	units: 1	Sigmoid

Learning parameter	Search space
learning rate	0.00001,..,0.001
decay	0.0
batch size	32
optimizer	Nadam
Max. no. of epochs	1000

TABLE VI: Meta MMNN parameters.

Layer	Search space	Activation
No. of layers	1,3,5	-
Conv1D	-	ReLU
Conv1D	filters: 32,...,128	-
Conv1D	kernel: 3,...,6	-
Dropout	P: 0.3,...,0.9	ReLU
Max/AvgPool	2	-
GlobalAvgPool	2	-
No. of layers	1	-
Dense	units: 16,..,64	ReLu
Dropout	P: 0.3,...,0.9	ReLU
Output	units: 1	Sigmoid
No. of layers	1,2,3,4,5	-
Dense	units: 32,..,128	ReLU
Dropout	P: 0.0,...,0.9	ReLU
Output	units:1	Sigmoid
Concatenate	-	-
Dense	units: 16,..,128	ReLU

Learning parameter	Search space
learning rate	0.00001,..,0.001
decay	0.0
batch size	32
optimizer	Nadam
Max. no. of epochs	1000

from Wilcoxon Tests, meaning that they are all comparable in their performances.

Lastly, we observe that Hyperparameters Optimization procedure can improve Models performances, confirming the results showed in [17]. As showed in Figures 8 9 10 11 12 13 the Meta Models are almost all-ways above Fixed Models. However, the reader should be aware that we did not perform Wilcoxon tests on these distribution because of the different choices in the holdout realization. Moreover, it is interesting noticing that the MetaCNNs are not overfitting in contrast to FixedCNNs.

5 CONCLUSIONS

This article described a Multi Modal framework combining the advantages of individual data modalities. After extensive experiments, it was shown that Bayesian Optimization can improve Models performances, and considering the encouraging results obtained by Random Forests it is worth carrying out more experiments exploiting classical Machine Learning Models.

Importantly, the present experimental set-up is based on a publicly available dataset, while the results are fully reproducible and available at [22].

[22]https://github.com/vincenzoconv99/

REFERENCES

[1] S. Ohno, "So much'junk'dna in our genome," in *Evolution of Genetic Systems, Brookhaven Symp. Biol.*, 1972, pp. 366–370.

[2] E. P. Consortium *et al.*, "An integrated encyclopedia of dna elements in the human genome," *Nature*, vol. 489, no. 7414, p. 57, 2012.

[3] R. P. C. DGT, F. Consortium *et al.*, "A promoter-level mammalian expression atlas," *Nature*, vol. 507, no. 7493, pp. 462–470, 2014.

[4] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998. [Online]. Available: https://doi.org/10.1109/34.709601

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: http://www.nature.com/articles/323533a0

[6] T. Dozat, "Incorporating nesterov momentum into adam technical report," 2015.

[7] U. G. Browser, "Frequently asked questions: Data file formats," https://genome.ucsc.edu/FAQ/FAQformat.htmlformat1.

[8] T. Shiraki, S. Kondo, S. Katayama, K. Waki, T. Kasukawa, H. Kawaji, R. Kodzius, A. Watahiki, M. Nakamura, T. Arakawa *et al.*, "Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage," *Proceedings of the National Academy of Sciences*, vol. 100, no. 26, pp. 15 776–15 781, 2003.

[9] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler, "The human genome browser at ucsc," *Genome research*, vol. 12, no. 6, pp. 996–1006, 2002.

[10] J. Navarro Gonzalez, A. S. Zweig, M. L. Speir, D. Schmelter, K. R. Rosenbloom, B. J. Raney, C. C. Powell, L. R. Nassar, N. D. Maulding, C. M. Lee *et al.*, "The ucsc genome browser database: 2021 update," *Nucleic Acids Research*, vol. 49, no. D1, pp. D1046–D1057, 2021.

[11] Nature, "The fantom5 project," https://www.nature.com/collections/jcxddjndxy.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[13] L. Cappelletti, A. Petrini, J. Gliozzo, E. Casiraghi, M. Schubach, M. Kircher, and G. Valentini, "Bayesian optimization improves tissue-specific prediction of active regulatory regions with deep neural networks," in *International Work-Conference on Bioinformatics and Biomedical Engineering*. Springer, 2020, pp. 600–612.

[14] D. Albanese, M. Filosi, R. Visintainer, S. Riccadonna, G. Jurman, and C. Furlanello, "Minerva and minepy: a c engine for the mine suite and its r, python and matlab wrappers," *Bioinformatics*, vol. 29, no. 3, pp. 407–408, 2013.

[15] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[16] M. B. Kursa and W. R. Rudnicki, "Feature selection with the boruta package," *Journal of statistical software*, vol. 36, pp. 1–13, 2010.

[17] L. Cappelletti, A. Petrini, J. Gliozzo, E. Casiraghi, M. Schubach, M. Kircher, and G. Valentini, "Boosting tissue-speci
c prediction of active cis-regulatory regions through deep learning and bayesian optimization techniques," in *BMC Bioinformatics*. BioMed Central, 2022.

[18] Y. Li, W. Shi, and W. W. Wasserman, "Genome-wide prediction of cis-regulatory regions using supervised deep learning methods. biorxiv," 2016.

[19] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

TABLE VII: Results of Wilcoxon test on Fixed Models (active enhancers vs inactive enhancers using Feature Selection).

Distribution 1	Distribution 2	p value	Statistical Difference
FFNN F1-Score	RF F1-Score	0.0019	RF outperforms FFNN
FFNN AUPRC	RF AUPRC	0.0019	RF outperforms FFNN
FFNN AUROC	RF AUROC	0.0019	RF outperforms FFNN
FFNN F1-Score	CNN F1-Score	0.0019	CNN outperforms FFNN
FFNN AUPRC	CNN AUPRC	0.16	Indistinguishable
FFNN AUROC	CNN AUROC	0.003	CNN outperforms FFNN
FFNN F1-Score	MMNN F1-Score	0.007	MMNN outperforms FFNN
FFNN AUPRC	MMNN AUPRC	0.0019	MMNN outperforms FFNN
FFNN AUROC	MMNN AUROC	0.0019	MMNN outperforms FFNN
RF F1-Score	MMNN F1-Score	0.0019	RF outperforms MMNN
RF AUPRC	MMNN AUPRC	0.005	MMNN outperforms RF
RF AUROC	MMNN AUROC	0.0019	MMNN outperforms RF
RF F1-Score	CNN F1-Score	0.0019	RF outperforms CNN
RF AUPRC	CNN AUPRC	0.0019	RF outperforms CNN
RF AUROC	CNN AUROC	0.0019	RF outperforms CNN
MMNN F1-Score	CNN F1-Score	0.005	MMNN outperforms CNN
MMNN AUPRC	CNN AUPRC	0.0019	MMNN outperforms CNN
MMNN AUROC	CNN AUROC	0.0019	MMNN outperforms CNN

TABLE VIII: Results of Wilcoxon test on Fixed Models (active enhancers vs inactive enhancers not using Feature Selection).

Distribution 1	Distribution 2	p value	Statistical Difference
FFNN F1-Score	RF F1-Score	0.0019	RF outperforms FFNN
FFNN AUPRC	RF AUPRC	0.0019	RF outperforms FFNN
FFNN AUROC	RF AUROC	0.0019	RF outperforms FFNN
FFNN F1-Score	CNN F1-Score	0.0076	CNN outperforms FFNN
FFNN AUPRC	CNN AUPRC	0.10	Indistinguishable
FFNN AUROC	CNN AUROC	0.013	Indistinguishable
FFNN F1-Score	MMNN F1-Score	0.017	Indistinguishable
FFNN AUPRC	MMNN AUPRC	0.0019	MMNN outperforms FFNN
FFNN AUROC	MMNN AUROC	0.0019	MMNN outperforms FFNN
RF F1-Score	MMNN F1-Score	0.0019	RF outperforms MMNN
RF AUPRC	MMNN AUPRC	0.19	Indistinguishable
RF AUROC	MMNN AUROC	0.0019	MMNN outperforms RF
RF F1-Score	CNN F1-Score	0.0019	RF outperforms CNN
RF AUPRC	CNN AUPRC	0.0019	RF outperforms CNN
RF AUROC	CNN AUROC	0.0019	RF outperforms CNN
MMNN F1-Score	CNN F1-Score	0.02	Indistinguishable
MMNN AUPRC	CNN AUPRC	0.0019	MMNN outperforms CNN
MMNN AUROC	CNN AUROC	0.0019	MMNN outperforms CNN

TABLE IX: Results of Wilcoxon test on Fixed Models (active promoters vs inactive promoters).

Distribution 1	Distribution 2	p value	Statistical Difference
FFNN F1-Score	RF F1-Score	0.0019	RF outperforms FFNN
FFNN AUPRC	RF AUPRC	0.0019	RF outperforms FFNN
FFNN AUROC	RF AUROC	0.0019	RF outperforms FFNN
FFNN F1-Score	CNN F1-Score	0.0019	CNN outperforms FFNN
FFNN AUPRC	CNN AUPRC	0.0019	FFNN outperforms CNN
FFNN AUROC	CNN AUROC	0.0019	FFNN outperforms CNN
FFNN F1-Score	MMNN F1-Score	0.0019	MMNN outperforms FFNN
FFNN AUPRC	MMNN AUPRC	0.08	Indistinguishable
FFNN AUROC	MMNN AUROC	0.08	Indistinguishable
RF F1-Score	MMNN F1-Score	0.0019	RF outperforms MMNN
RF AUPRC	MMNN AUPRC	0.08	Indistinguishable
RF AUROC	MMNN AUROC	0.08	MMNN outperforms RF
RF F1-Score	CNN F1-Score	0.0019	RF outperforms CNN
RF AUPRC	CNN AUPRC	0.0019	RF outperforms CNN
RF AUROC	CNN AUROC	0.0019	RF outperforms CNN
MMNN F1-Score	CNN F1-Score	0.0019	MMNN outperforms CNN
MMNN AUPRC	CNN AUPRC	0.0019	MMNN outperforms CNN
MMNN AUROC	CNN AUROC	0.0019	MMNN outperforms CNN

TABLE X: Results of Wilcoxon test on Meta Models (active enhancers vs inactive enhancers using Feature Selection).

Distribution 1	Distribution 2	p value	Statistical Difference
MetaFFNN F1-Score	MetaRF F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaRF AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaRF AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaMMNN F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaMMNN AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaMMNN AUROC	0.31	Indistinguishable
MetaRF F1-Score	MetaMMNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaMMNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaMMNN AUROC	0.06	Indistinguishable
MetaRF F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaMMNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaMMNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaMMNN AUROC	MetaCNN AUROC	0.06	Indistinguishable

TABLE XI: Results of Wilcoxon test on Meta Models (active enhancers vs inactive enhancers not using Feature Selection).

Distribution 1	Distribution 2	p value	Statistical Difference
MetaFFNN F1-Score	MetaRF F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaRF AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaRF AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaMMNN F1-Score	0.31	Indistinguishable
MetaFFNN AUPRC	MetaMMNN AUPRC	0.81	Indistinguishable
MetaFFNN AUROC	MetaMMNN AUROC	0.81	Indistinguishable
MetaRF F1-Score	MetaMMNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaMMNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaMMNN AUROC	0.06	Indistinguishable
MetaRF F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaMMNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaMMNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaMMNN AUROC	MetaCNN AUROC	0.06	Indistinguishable

TABLE XII: Results of Wilcoxon test on Meta Models (active promoters vs inactive promoters).

Distribution 1	Distribution 2	p value	Statistical Difference
MetaFFNN F1-Score	MetaRF F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaRF AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaRF AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaFFNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaFFNN AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaFFNN F1-Score	MetaMMNN F1-Score	1.0	Indistinguishable
MetaFFNN AUPRC	MetaMMNN AUPRC	1.0	Indistinguishable
MetaFFNN AUROC	MetaMMNN AUROC	0.18	Indistinguishable
MetaRF F1-Score	MetaMMNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaMMNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaMMNN AUROC	0.06	Indistinguishable
MetaRF F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaRF AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaRF AUROC	MetaCNN AUROC	0.06	Indistinguishable
MetaMMNN F1-Score	MetaCNN F1-Score	0.06	Indistinguishable
MetaMMNN AUPRC	MetaCNN AUPRC	0.06	Indistinguishable
MetaMMNN AUROC	MetaCNN AUROC	0.06	Indistinguishable

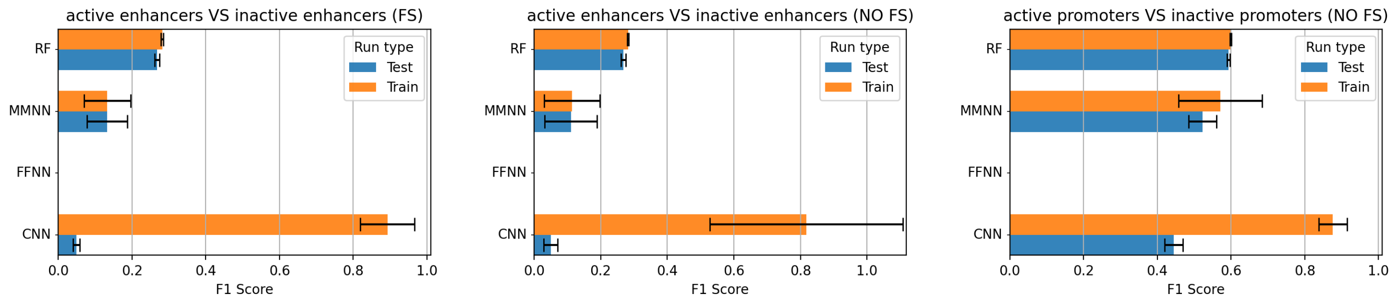

Fig. 8: F1-Score evaluation of Fixed Models over the different tasks.

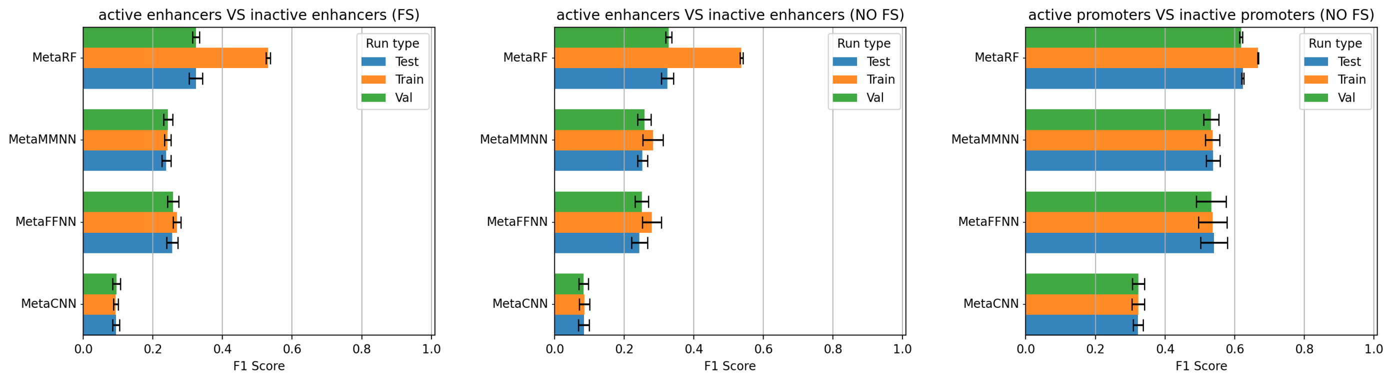

Fig. 9: F1-Score evaluation of Meta Models over the different tasks.

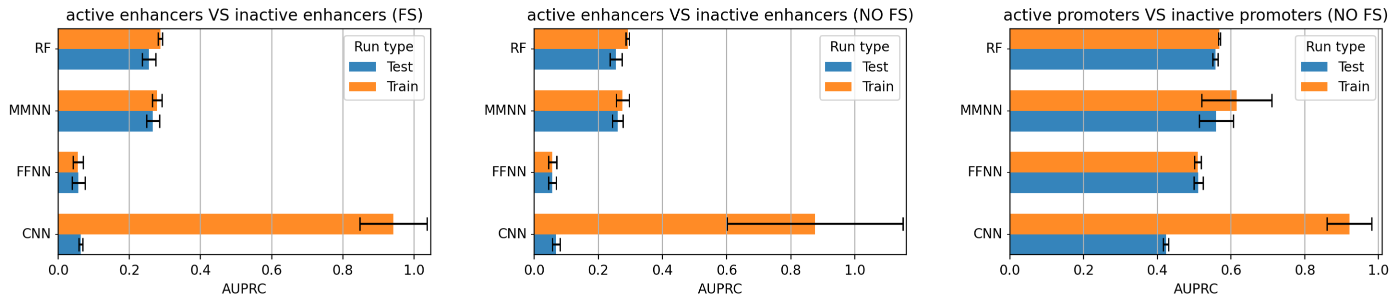

Fig. 10: AUPRC evaluation of Fixed Models over the different tasks.

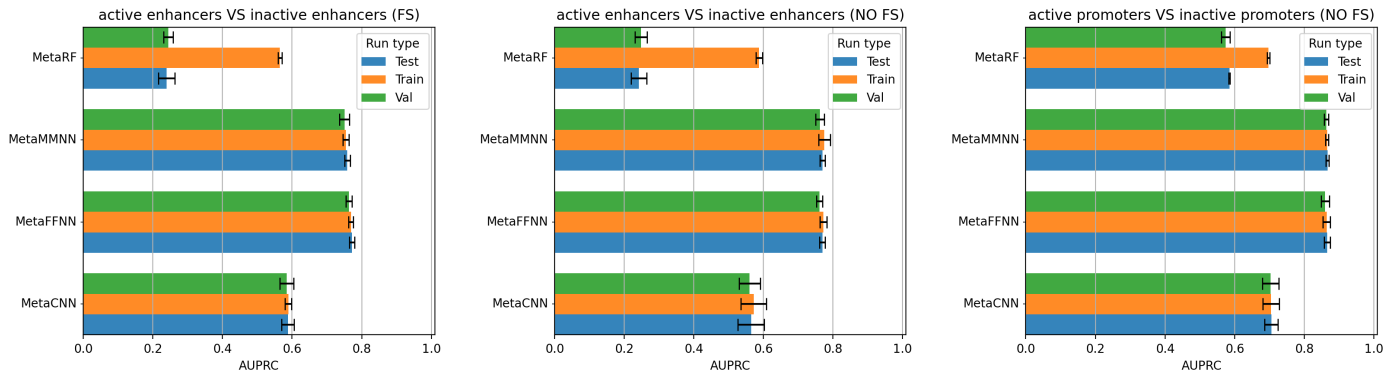

Fig. 11: AUPRC evaluation of Meta Models over the different tasks.

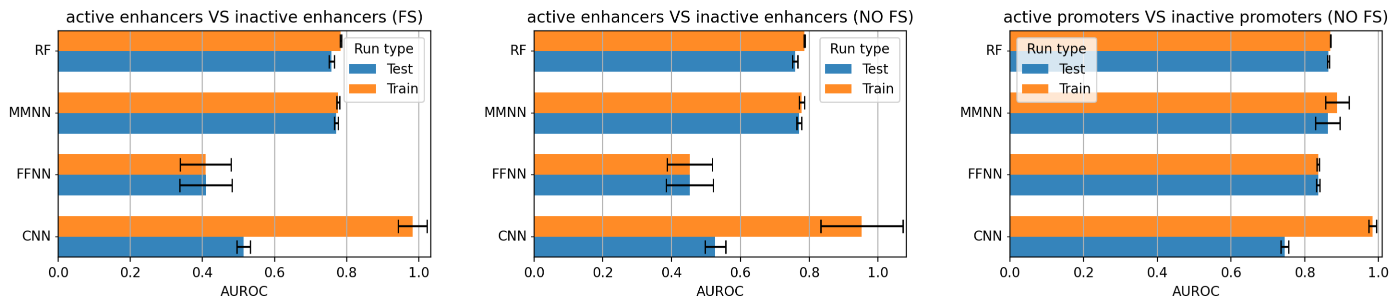

Fig. 12: AUROC evaluation of Fixed Models over the different tasks.

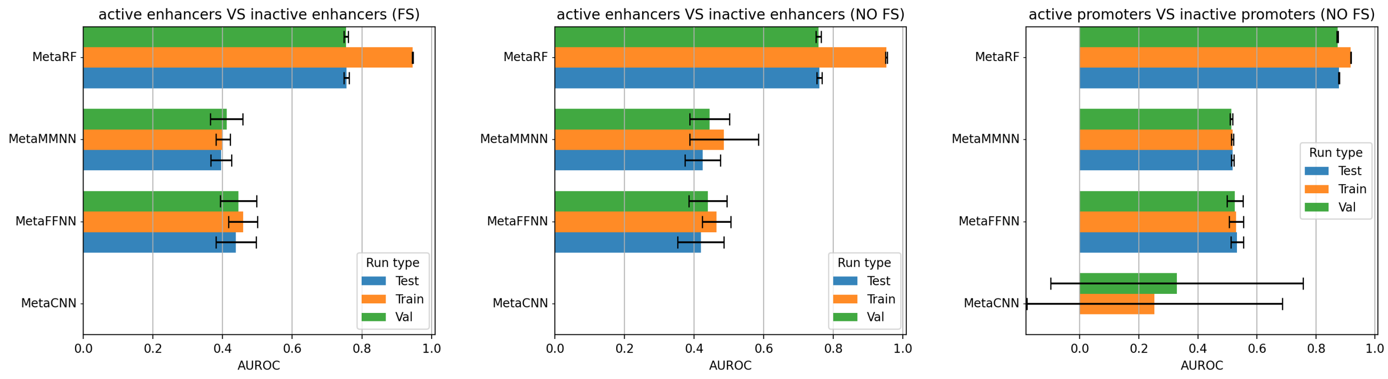

Fig. 13: AUROC evaluation of Meta Models over the different tasks.

TABLE XIII: Results of Wilcoxon test on Feature Selection Models vs Non Feature Selection Models (active enhancers vs inactive enhancers)

Distribution 1	Distribution 2	p value	Statistical Difference
FFNN (FS) F1-Score	FFNN (NO FS) F1-Score	1.0	Indistinguishable
FFNN (FS) AUPRC	FFNN (NO FS) AUPRC	1.0	Indistinguishable
FFNN (FS) AUROC	FFNN (NO FS) AUROC	0.27	Indistinguishable
MMNN (FS) F1-Score	MMNN (NO FS) F1-Score	0.26	Indistinguishable
MMNN (FS) AUPRC	MMNN (NO FS) AUPRC	0.04	Indistinguishable
MMNN (FS) AUROC	MMNN (NO FS) AUROC	0.92	Indistinguishable
RF (FS) F1-Score	RF (NO FS) F1-Score	0.92	Indistinguishable
RF (FS) AUPRC	RF (NO FS) AUPRC	0.55	Indistinguishable
RF (FS) AUROC	RF (NO FS) AUROC	0.55	Indistinguishable
MetaFFNN (FS) F1-Score	MetaFFNN (NO FS) F1-Score	0.18	Indistinguishable
MetaFFNN (FS) AUPRC	MetaFFNN (NO FS) AUPRC	0.43	Indistinguishable
MetaFFNN (FS) AUROC	MetaFFNN (NO FS) AUROC	0.43	Indistinguishable
MetaMMNN (FS) F1-Score	MetaMMNN (NO FS) F1-Score	0.31	Indistinguishable
MetaMMNN (FS) AUPRC	MetaMMNN (NO FS) AUPRC	0.18	Indistinguishable
MetaMMNN (FS) AUROC	MetaMMNN (NO FS) AUROC	0.62	Indistinguishable
MetaRF (FS) F1-Score	MetaRF (NO FS) F1-Score	1.0	Indistinguishable
MetaRF (FS) AUPRC	MetaRF (NO FS) AUPRC	1.0	Indistinguishable
MetaRF (FS) AUROC	MetaRF (NO FS) AUROC	1.0	Indistinguishable