Active regulatory regions prediction using Deep Learning and Bayesian Optimisation

Francesco Stella
Department of Computer Science
University of Milan, Milan, Italy
francesco.stella@studenti.unimi.it

***Abstract*—Active regulatory regions prediction is a central task in Bioinformatics and it is crucial to understand gene expression and the connection with diseases such as cancer or Alzheimer. In this work a comparison between different models, specifically a Feed-Forward Neural Network (FFNN), a Convolutional Neural Network (CNN) and a Multi-Modal Neural Network (MMNN) is performed and a detailed description of the preprocessing, data analysis, training and evaluation procedures is given.**

1. Introduction

This work aims at devising and evaluating three Deep Learning models for predicting active regulatory regions for the GM12878 cell-line, using a statistically robust methodology. After a very brief introduction to the problem and related works in section 2, an introductory description of the approaches used for model and feature selection is given in section 3, involving Bayesian Hyperparameter Optimisation (BHO) and the BorutaSHAP algorithm. Then, in section 4, the experimental setup is discussed, more precisely the used datasets are presented together with the methods for their retrieval and the steps followed for data preprocessing, data analysis, training, validation and evaluation are discussed in detail. The results of the training process are presented in section 5 and final conclusion are given in section 6.

2. Related works and state-of-the-art

In recent years, Deep Learning methods have been widely adopted for a variety of Bioinformatics tasks, with promising results. Li et al. [1] used supervised deep learning to detect enhancers and promoters in the genome and to distinguish between their active or inactive state. They retrieved the annotations for promoters and enhancers from FANTOM dataset [1] and other genome-wide feature data from ENCODE dataset [2], configuring the task as a classification problem and using a feed-forward neural network (FFNN). A key role for their models' performance is played by the Randomised Deep Feature Selection (RDFS) and the Random Forest (RF) models used for feature selection, which allowed their model to obtain promising results above 0.80 auPRC (area under the Precision-Recall Curve) for six cell lines in detecting active enhancers and promoters from exons and other unknown regions. Bu et al. [2] developed a CNN capable of predict active super-enhancers genome-wide. Super-enhancers are clusters of active enhancers that have a key role in the development of diseases including tumors. Their model essentially consists of a variable number of convolutional layers (specifically two, three or four layers) followed by a fully-connected layer and an output layer. The best performing configuration is that with four convolutional layers, which achieves a precision of 0.92 (0.90 F1-score).
Fang et al. [3] compared classical Machine Learning models, specifically a linear Support Vector Machine (SVM), a SVM with Radial Basis Function (RBF) kernel and a Random Forest, with Deep Learning models, more precisely with a FFNN and a CNN. Their goal was to find the most suitable model to functionally annotate the regulatory non-coding genome, such as enhancers. Fang et al. showed that random forest has similar perfomance with SVM, while SVM with RBF kernel outperforms both these models and the best model found is the CNN.

3. Models

Three models are trained and evaluated on the datasets, more precisely a Feed-Forward Neural Network (FFNN), a Convolutional Neural Network (CNN) and a Multi-Modal Neural Network (MMNN). Before training, Bayesian Hyperparameter Optimisation [4] (BHO) is performed on FFNN and CNN in order to explore hyperparameter space and perform model selection. More precisely, structural hyperparameters (number of layers of the network or number of units of a specific layer) are used for tuning the networks and the best configurations are adopted for the actual training. In the following, a brief introduction to BHO and BorutaSHAP feature selection algorithm given.

3.1. Bayesian Hyperparameter Optimisation

Bayesian Hyperparameter Optimisation is an approach to optimisation typically used when dealing with objective functions that are non-convex or costly to evaluate [4],

1. FANTOM: https://fantom.gsc.riken.jp/
2. ENCODE: https://www.encodeproject.org/

although it can present issues when dealing with high-dimensional spaces [5]. Compared to other common techniques, such as Grid Search or Random Search, the strength of BHO resides in the possibility to guide the search toward promising hyperparameter configurations. BHO entails the formulation of a *surrogate model* of the unknown objective function, that is an estimate of the original function obtained by computing, typically, the *posterior mean function* of a Gaussian Process (GP). A Gaussian Process is a generalisation of the Multivariate Gaussian distribution, more precisely it is an infinite-dimension stochastic process in which any finite combination of dimensions is a Gaussian distribution [6]. Hence, GPs are distributions over functions and are completely specified by their *mean* and *covariance functions*, the latter being typically the *squared exponential function* (also known as RBF kernel). The actual objective function we want to optimise belongs to the family of functions sampled from a GP, which incorporates our prior knowledge about the problem. The goal consists in estimating the posterior probability of a model (or *hypothesis*) M given the evidence (or *observations*) E that we collect during the search in the hyperparameter space. Equation 1 shows that the posterior probability of M given E is proportional to the *likelihood* of E given M multiplied by the prior of M:

$$P(M|E) \propto P(E|M)P(M) \tag{1}$$

There are two main choices that must be performed during BHO, namely the choice of the prior (as previously discussed) and the choice of an *acquisition function* [7]. The acquisition function guides the search, specifying the next points in which to sample the surrogate model. The idea is to find a compromise between *exploration*, i.e. sampling in correspondence of high uncertainty, and *exploitation*, i.e. sampling in correspondence to high/low values of objective function. There are different acquisition function typically used, the main ones being *Probability of Improvement (PI)*, *Expected Improvement (EI)* and *Upper Confidence Bound (UCB) / Lower Confidence Bound (LCB)*. PI can be computed analytically in the case of GP, as well as EI, with the former used to maximise the probability of improving and the latter used to maximise the EI over the best current value. Finally, UCB and LCB have a tunable parameter k that balances exploitation and exploration. In all cases, noise is assumed to be present in each observation. More details will be discussed in appendix A.
In this project, keras_tuner [1] is used to perform BHO for the FFNN and the CNN models, more precisely to select the number of units in the Dense layers (for both FFNN and CNN), the number of filters and the size of the kernel (for CNN). Moreover, during BHO, the inclusion of Dropout and Batch Normalisation layers is chosen for both models, with the Dropout value being selected from a predefined interval. Table 1 shows some of the main parameters used for the Bayesian Optimisation procedure.

1. https://keras.io/api/keras_tuner/tuners/bayesian/

Parameter	Value	Description
acquisition function	UCB	acquisition function to maximise
alpha	1e-4	expected amount of noise in the sampled observations
beta	2.6	balancing between exploration and exploitation
max_trials	15 (CNN) 20 (FFNN)	maximum number of model configurations to test

TABLE 1: Main parameters used for BHO.

3.2. Feature Selection methods

Feature selection plays a central role in a machine learning task, having an important effect on the final performances of the model and on the time requirements of training. The goal is to remove redundant and irrelevant features. Several approaches have been studied over the years for performing feature selection, however there are some advantages and disadvantages to take into account for each method [8], [9]. Broadly speaking, feature selection methods can be grouped into three main classes, namely *Filter Methods*, *Wrapper Methods* and *Embedded Methods*, although other approaches including (among others) hybrid and ensemble methods have been proposed too. Filter Methods entail the ranking of the features through a score obtained upon the execution of some statistical tests and then use a threshold to select a subset of features. To this class belong, for example, χ^2, Fisher, Mann-Whitney U and Pearson correlation tests, which are simple and fast but they cannot capture feature interactions since they are *univariate* methods. Alongside these methods, there are also *multivariate* tests including Mutual Information Feature Selection (MIFS), minimal Redundancy Maximal Relevance (mRMR) or Conditional Mutual Information Maximisation (CMIM), that remove redundant features, thus improving model's performance. Furthermore, exhaustive methods have been developed (e.g. BOOST, FastEpistasis, TEAM) to find all feature interactions, however they are restricted to pairwise interactions and they cannot remove redundancy [9].
Wrapper Methods use the performance of the selected algorithm to find the best subset of features on which train the model. Major disadvantages of these methods involve the dependency on the specific chosen algorithm, hence they not guarantee to work if another algorithm is chosen after the feature selection task, and the heavy computational load. However, they typically result in higher performance than filter methods [9].
Finally, Embedded Methods integrate the feature selection task in the algorithm itself and the internal parameters of the model are adjusted during training together with the weights given to each feature.

3.3. BorutaSHAP algorithm

In this work, an algorithm named BorutaSHAP[3] and belonging to the family of wrapper methods is chosen for

3. https://github.com/Ekeany/Boruta-Shap

feature selection. This algorithm is based on the concept of *shapley value*, term coined by Shapley [10] that lays its foundations on cooperative game theory and also used in *Explainable ML* to provide an explanation of the predictions of a model. More precisely, considering a prediction task and a set of features $\mathcal{X}$, the shapley value of a feature $x \in \mathcal{X}$ is a measure of the contribution of the feature itself on the prediction. Shapley values can be defined using two approaches, namely a *permutation-based approach* and a *subset-based approach*. The former considers all possible permutations $p \in \mathcal{P}$ of the features and adds a new $x_i \in \mathcal{X}$ to a set $\mathcal{S}$ in the order provided by p. The marginal contribution of x_i to the set $\mathcal{S}$ is considered before adding x_i to this set. Then, the average marginal contribution of x_i across all permutations is defined as its shapley value. This algorithm can be formalised as shown in equation 2, in which *v* is a value function of a subset of features.
While the first approach has a complexity of O(M!), the second one has a complexity of O(2^M) and is represented in equation 3. Time complexity is one of the main disadvantages of using shapley values, hence an approximation is typically used entailing the uniform sampling of permutations from $\mathcal{P}$ [11].

$$\phi_i(v) = \mathop{\mathbb{E}}_{O \sim \pi(M)} [v(pre_i(O) \cup \{i\}) - v(pre_i(O))] \tag{2}$$

$$\phi_i(v) = \mathop{\mathbb{E}}_{S} \left[\frac{2^{M-1}}{M} \binom{M-1}{M}^{-1} (v(S \cup \{i\}) - v(S)) \right] \tag{3}$$

In BorutaSHAP, the so called SHAP (SHapley Additive exPlanations) values are used to assess the relevance of the features. SHAP values were first proposed by Lundberg et al. [12] and they are key enablers of explainability in ML, since they can be used with any ML model to achieve local explainability (at the instance level). Starting from the expected prediction of the model (base value), SHAP values estimate the contribution of each feature to the actual predicted value as a conditional expectation given that feature. Figure 1 should clarify this statement. Essentially, BorutaSHAP is a variation of the classic Boruta algorithm [13], that is also a wrapper method that uses Random Forests to perform feature selection, leveraging on a statistical test to assess whether a given feature is more relevant than a random guess. Further details about BorutaSHAP together with the retrieved feature importance values are presented in appendix B.

4. Experimental setup

The experimental setup is described in the following, starting with the presentation of the datasets in section 4.1, followed by the preprocessing procedure discussed in section 4.2. Then, data analysis and visualisation are addressed in 4.3, followed by the K-Fold Cross Validation (K-Fold CV) algorithm 4.4 and training task 4.5.

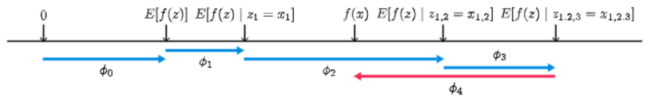


Figure 1: SHAP values representation, from [12]. The order in which feature contributions are considered matters and all the orderings (or a subset) should be considered.

4.1. Datasets

Three datasets are considered for the retrieval of epigenomic and sequence data, together with the labels associated to the enhancers and promoters:

- **ENCODE Dataset**: ENCODE (ENCyclopedia Of DNA Elements [14]) is an international consortium whose goal is to provide a comprehensive collection of functional elements of the human genome together with two levels of annotation of experimental data;
- **UCSC Genome Browser**: is an online tool that provides a means to visualise all the 23 chromosomes of the human genome at different scales [15], hosted by the University of California, Santa Cruz, that also played a key role in the development of the ENCODE project and still supports it.
- **FANTOM5**: Functional ANnoTation Of the Mammalian genome [16] is a worldwide project whose goal is to identify all functional elements in the mammalian genomes. This project was started at the research and development institute of RIKEN, Yokohama, in 2000. It currently consists of six editions and the fifth, which is used in this work, systematically investigated the set of genes present in all the human body's cell types. The researchers first used Cap Analysis of Gene Expression (CAGE) to map transcriptome and regulatory data together with a number of cancer cell lines, then they performed RNA analysis considering different cell types.

Epigenomic data, consisting in enhancer and promoter regions for the GM12878 cell-line, is retrieved from ENCODE using the Python package *epigenomic_dataset* [4] that provides also label values retrieved from FANTOM5. The epigenomic_dataset package also provides a means to process the regulatory regions with a window of a given size and with a metric chosen between *mean* and *max*. A window size of 256 is chosen, together with the *mean* metric, which is typically more robust, since it reduces the effect of the noise on the data. Different available configurations, including window sizes of 64, 128, 512 or 1024 and the use of the *max* metric should be tested, however due to time constraints only the aforementioned configuration is used. The UCSC Genome Browser provides the human

4. https://github.com/AnacletoLAB/epigenomic_dataset

genome from the HG38 assembly, which is retrieved using the package *ucsc_genomes_downloader*[5].

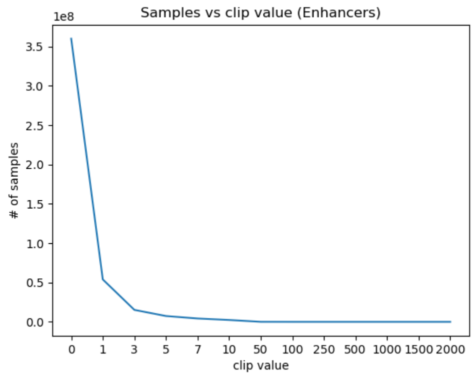
4.2. Data preprocessing

Preprocessing plays a central role in the training of a model, determining the final performance of the model itself. The following steps are followed, in the order: constant feature removal, data imputation (i.e. replacing null values), labels clipping, data scaling, highly correlated features removal and uninformative features removal. Having retrieved epigenomic data, with a window size of 256 and using the mean metric, constant values are searched for to be removed, however none of them is found. Then, data imputation is executed using the KNNImputer implementation from Scikit-Learn[6], that uses the K-NN algorithm to fill missing values in the dataset (with the mean of the neighbours). Subsequently, labels clipping is performed in order to establish an upper value for the ground truth values, that may be affected by undesired phenomena such as noise or other complications in the acquisition of the data. All values exceeding the threshold are modified to match the threshold itself. Figure 2 shows the number of samples varying with the clipping values (or thresholds) for both enhancers and promoters. A value of 3 is used for the enhancer labels, while a value of 50 is used for the promoter labels.

Then, another important preprocessing step that can improve both the training efficiency and the performance of the model is performed, namely the feature scaling. Two types of scaling are performed based on the *density* of the features, defined as the ratio between non-sparse features on the total. Robust scaling is used when this ratio is greater than or equal to 0.85 and consists in removing the median of the data and scaling according to the interquartile range, which makes this approach robust to outliers. On the contrary, the maximum absolute scaler scales the features individually, limiting their maximum value to 1.0 and avoiding to modify the original sparsity of the data.

4.3. Data analysis

After preprocessing, data analysis is performed following a sequence of steps. First, Pearson and Spearman tests are executed on enhancers and promoters data in order to detect correlations among features and outputs. The implementations provided by scipy[7] [8] are used. More precisely, both tests are executed on the feature set and highly correlated features (correlation threshold > 0.95) are removed, while correlated features ($0.8 \leq$ correlation threshold ≤ 0.95) are retained and some of them are visualised using pairplots, shown in figure 3. Also, correlations between features and outputs are tested, however none of them is found with these two tests. The major difference between Pearson and



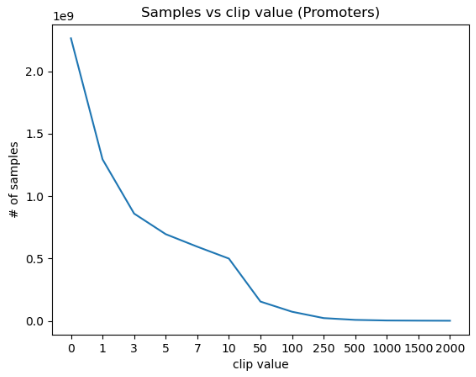


Figure 2: Labels clipping values for enhancers and promoters.

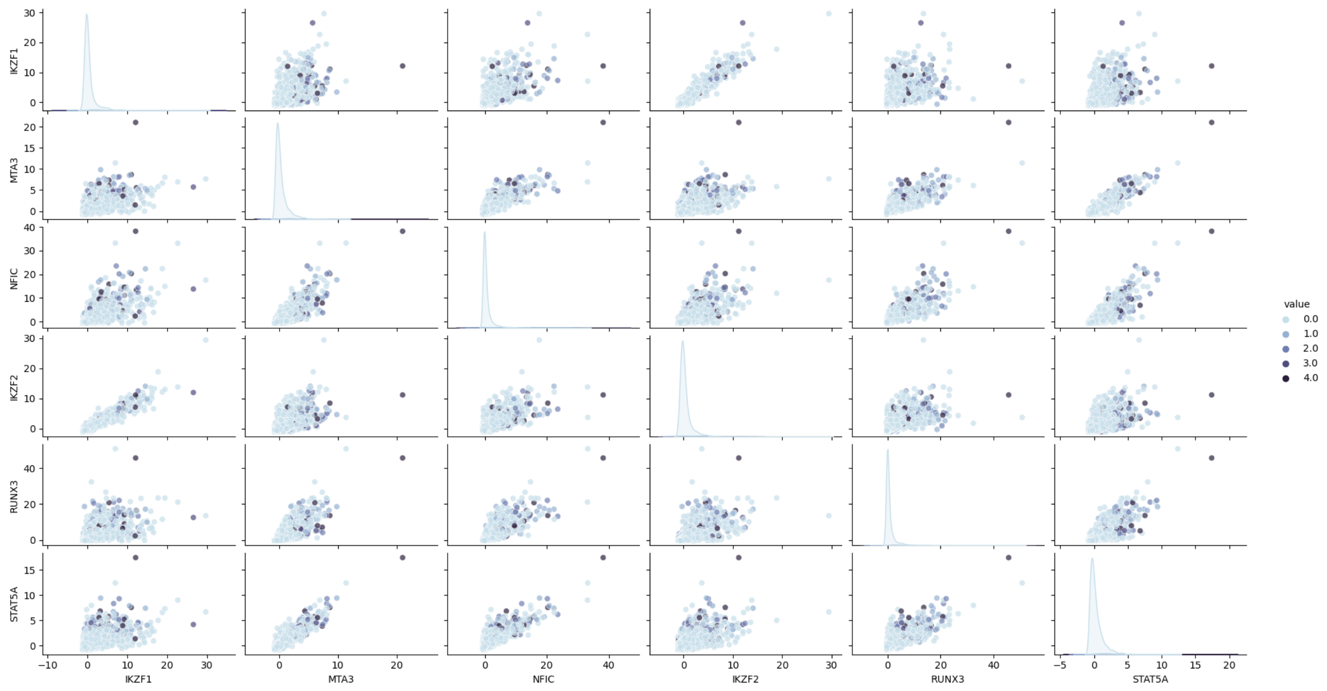
Spearman tests is that the former is only capable of detecting linear relationships, while the latter can find nonlinearities, at the cost of larger execution time (for further details, read appendix C).

After the correlation tests, the BorutaSHAP algorithm is executed to select features. This algorithm is a wrapper method that can use tree-based predictors to perform feature selection. More precisely, a Random Forest Regressor provided by XGBoost[9] is used together with K-Fold Cross Validation (with K=3, see sec. 4.4) to select features. BorutaSHAP uses this model to estimate SHAP values and thus the relevance of each feature. Two examples of feature importance plots retrieved during the execution of the algorithm are given in appendix B. A key aspect that led to the choice of the BorutaSHAP implementation for feature selection is that it provides the reliability of the SHAP metric for estimating feature relevance together with the GPU support, which drastically reduces computation time. Upon the end, the number of retained features is 73 for the enhancers task and 100 for the promoters task.

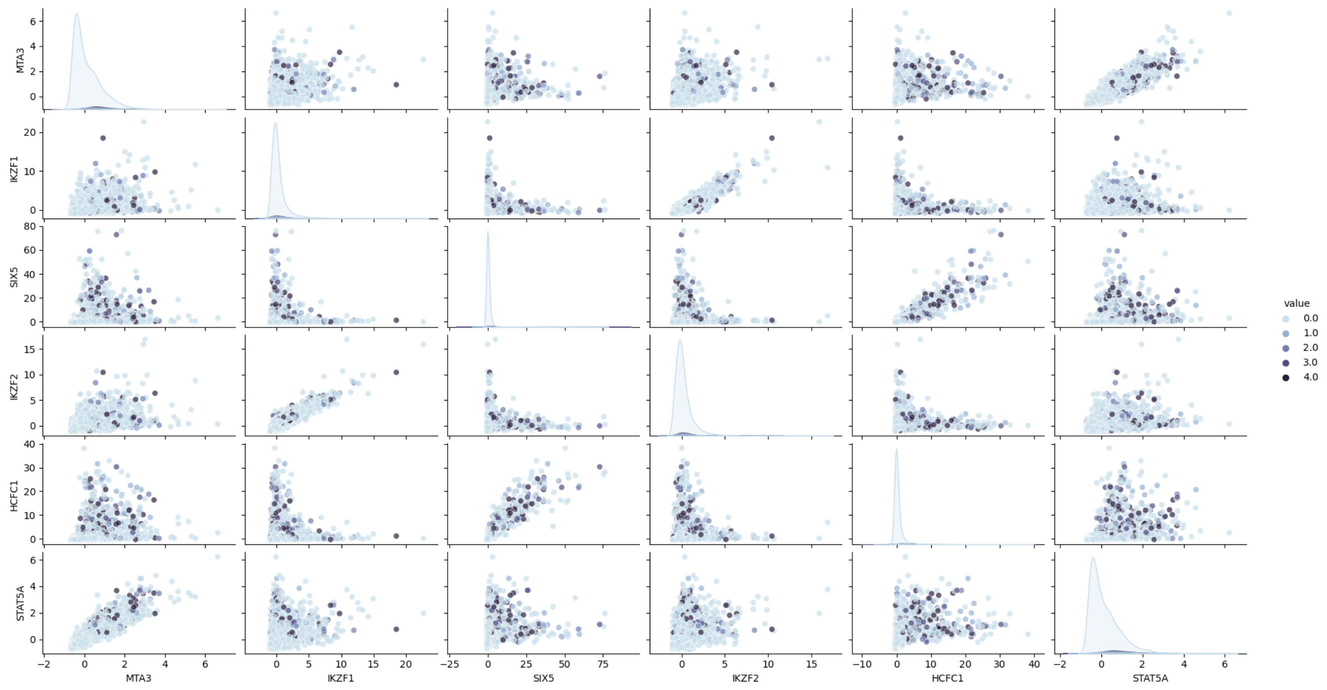
Data visualisation through Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) algorithms is also performed on the features, figure 4 shows the results for both enhancers and promoters and both sequence and epigenomic data. PCA is a widely used

5. https://github.com/LucaCappelletti94/ucsc_genomes_downloader
6. KNNImputer - sklearn
7. Pearson Test - scipy
8. Spearman Test - scipy

9. XGBoost Random Forests



(a) Some of the most correlated enhancer features.



(b) Some of the most correlated promoter features.

Figure 3: Pairplots representing correlated features for the enhancers and promoters prediction tasks.

deterministic dimensionality reduction technique, whose goal is to reduce the dimensionality while preserving the most information. This is achieved by performing a linear transformation of the original components into a lower-dimensional space, following the ordering dictated by the variance associated to each one of them: the component with the largest associated variance is first projected, followed by the second and so on. In this project, two principal components are used in order to visualise the data with a two-dimensional plot. Despite its ease of use and relatively low computation time, PCA may not be the best choice to analyse data in some cases, since it is mainly concerned with preserving large pairwise distances and this is not always the best strategy for visualising data, especially in presence of complex manifolds. A non-deterministic and non-linear approach, represented by t-SNE, preserves local structures, i.e. small pairwise distances, thus providing better visualisation capabilities. Essentially, given N datapoints, t-SNE tries to build a simmetric probability distribution, shown in equation 4, that is proportional to the similarity between these datapoints. The probability distribution is a gaussian, whose amplitude is adapted to the density of the points by means of a parameter called *perplexity*, that defines the number of neighbors to consider, with larger values of perplexity making the algorithm less sensitive to local structures. Then, a map to a lower-dimensional space is computed by minimising the Kullback-Leibler divergence, represented in equation 5, using gradient descent and the similarity between datapoints in the new space is modeled by a t-Student distribution. Given all these computations, t-SNE is of course more expensive, hence an implementation leveraging GPU acceleration[10] is used in this project.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{4}$$

$$KL(P||Q) = \sum_{i \neq j} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{5}$$

4.4. K-Fold Cross Validation

K-Fold Cross Validation is a technique used for training and validating ML algorithms, typically used in presence of small datasets due to its capability to effectively use limited data. However, K-Fold CV is also suitable in presence of larger dataset, typically providing more robust metrics with respect to classic holdouts strategies and, despite its greater expensiveness with respect to simpler methods, in this projects the 3-Fold Cross Validation algorithm is used to splits the dataset. More precisely, the dataset is split into 3 equal-sized folds and each one of them is used in turn as the validation split, while the other 2 are used for the actual training. In general, a larger number of splits leads to a better validation procedure, typically 10 is a good number, however due to limited time this value is set to 3 in the present project. This approach is used first for feature selection and then for the training of the three models.

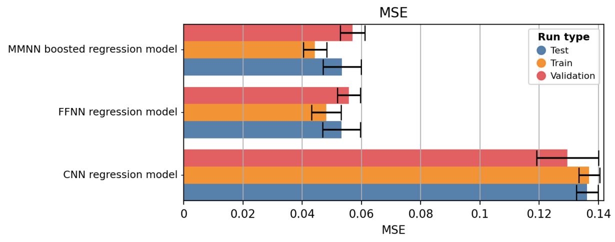
10. tsne-cuda implementation

4.5. Training

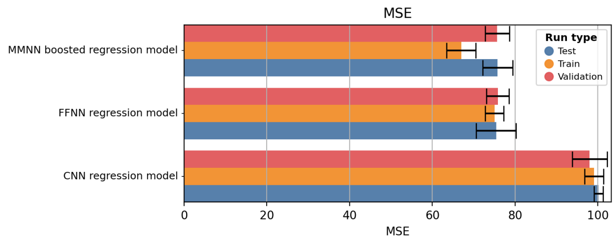
The training of the models is preceded by BHO (except for the MMNN), in which the best configuration found for the hyperparameters is used for the actual training. The Keras-Tuner[11] framework is used to search in the hyperparameter space, more precisely two hypermodels are defined for the FFNN and the CNN architectures, with the search space involving structural elements (such as the number of layers) and regularisation elements (the Dropout value). For this first optimisation phase, a simple train-test split of the dataset is used, with two thirds of the elements devoted to the training and one third to the testing. This choice can be improved by using K-Fold cross validation, however, since it is more time consuming, this simpler procedure is chosen. On the other hand, after the selection of the best models through BHO, the training and validation are performed using 3-Fold CV on the 90% of the dataset and the evaluation is performed on the remaining 10% of the data, which is the test set. The overall training history for the three models is reported in figures 5, 6 and 7. For the other histories, please check the project repository.

5. Results

After the training procedure, the performance of the models are plotted using the package barplots[12] and the results are shown in figure 8, in which mean squared error is shown for each of the splits of the dataset, namely train, validation and test, for both the tasks related to enhancers and promoters.



(a) Mean squared error metric for each model trained on the enhancers task, with a window size equals to 256.



(b) Mean squared error metric for each model trained on the promoters task, with a window size equals to 256.

Figure 8: Barplots containing the Mean Squared Error estimated during the training of the three models.

11. https://keras.io/keras_tuner/
12. https://github.com/LucaCappelletti94/barplots

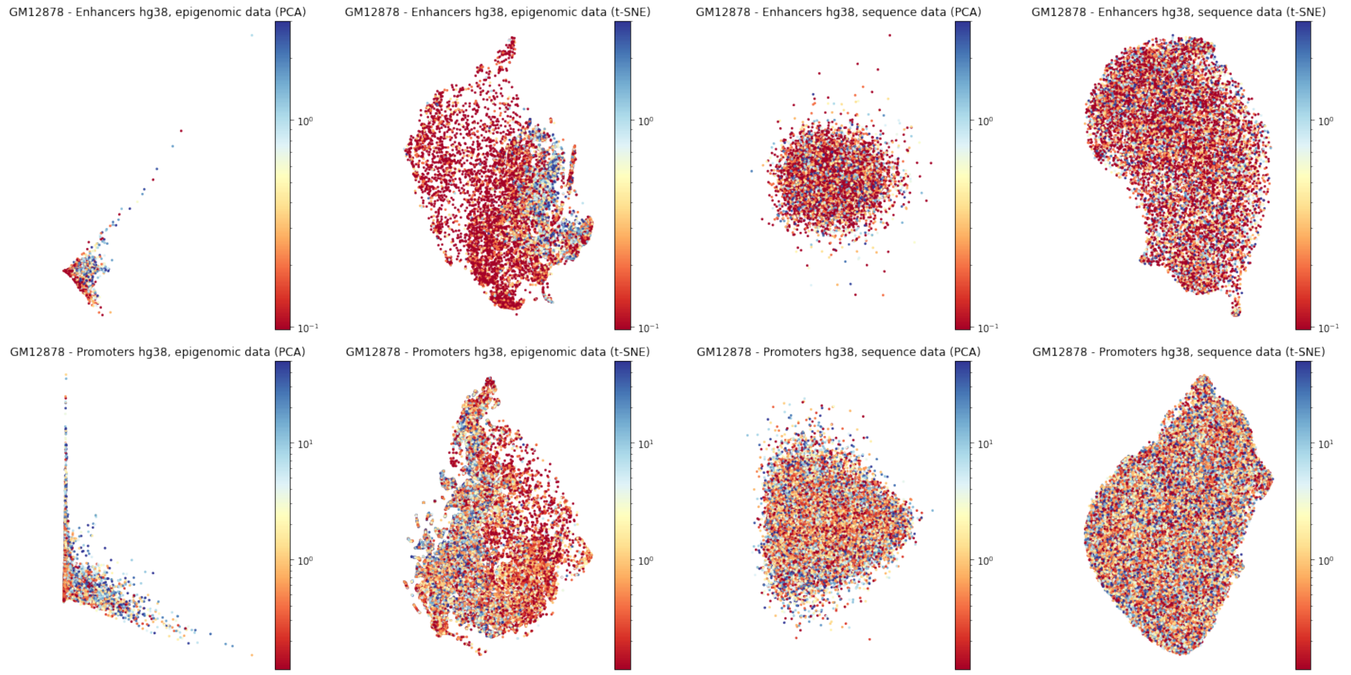
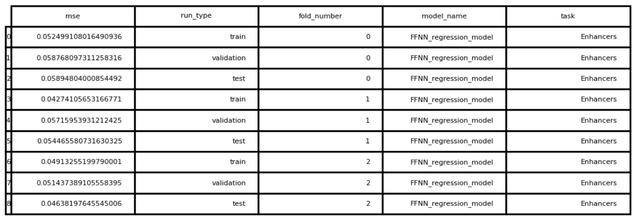
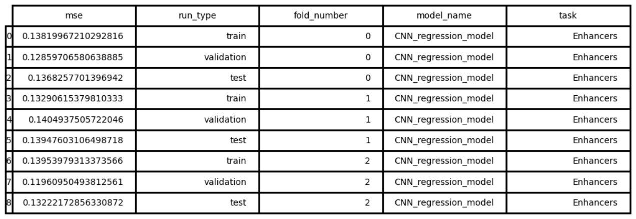


Figure 4: Application of PCA and t-SNE algorithms for dimensionality reduction of enhancers and promoters and both epigenomic and sequence data.



	mse	run_type	fold_number	model_name	task
0	0.052499108016490936	train	0	FFNN_regression_model	Enhancers
1	0.058768097311258316	validation	0	FFNN_regression_model	Enhancers
2	0.05894804000854492	test	0	FFNN_regression_model	Enhancers
3	0.04274105653166771	train	1	FFNN_regression_model	Enhancers
4	0.05715953931212425	validation	1	FFNN_regression_model	Enhancers
5	0.054465580731630325	test	1	FFNN_regression_model	Enhancers
6	0.04913255199790001	train	2	FFNN_regression_model	Enhancers
7	0.051437389105558395	validation	2	FFNN_regression_model	Enhancers
8	0.04638197645545006	test	2	FFNN_regression_model	Enhancers

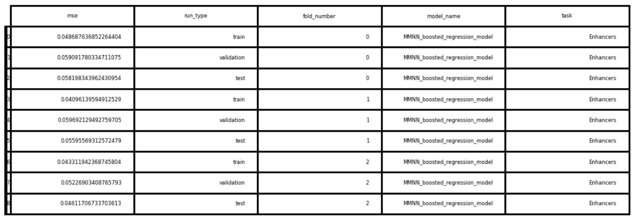
Figure 5: Training history of the FFNN model for the enhancers task.



	mse	run_type	fold_number	model_name	task
0	0.13819967210292816	train	0	CNN_regression_model	Enhancers
1	0.12859706580638885	validation	0	CNN_regression_model	Enhancers
2	0.1368257701396942	test	0	CNN_regression_model	Enhancers
3	0.13290615379810333	train	1	CNN_regression_model	Enhancers
4	0.1404937505722046	validation	1	CNN_regression_model	Enhancers
5	0.13947603106498718	test	1	CNN_regression_model	Enhancers
6	0.13953979313373566	train	2	CNN_regression_model	Enhancers
7	0.11960950493812561	validation	2	CNN_regression_model	Enhancers
8	0.13222172856330872	test	2	CNN_regression_model	Enhancers

Figure 6: Training history of the CNN model for the enhancers task.

Moreover, the Wilcoxon signed-rank test is performed in order to statistically compare the results. More precisely, the metrics related to each fold of the training history of the models are grouped by model and tested pairwise. The Wilcoxon test computes the differences between the single observations to assess whether the null hypothesis, stating that the two samples come from the same distribution, must



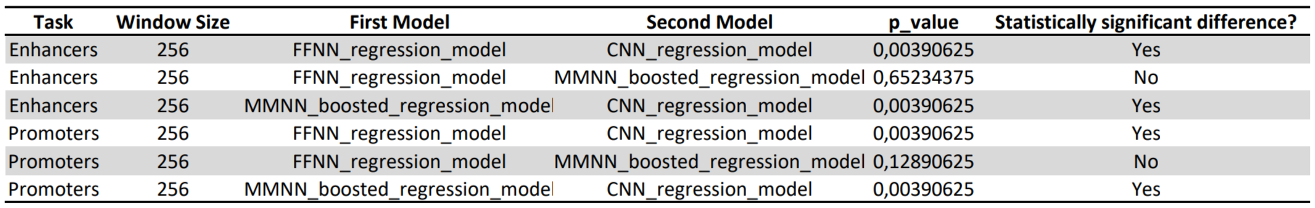
	mse	run_type	fold_number	model_name	task
0	0.048687636852264404	train	0	MMNN_boosted_regression_model	Enhancers
1	0.059091780334711075	validation	0	MMNN_boosted_regression_model	Enhancers
2	0.058198343962430954	test	0	MMNN_boosted_regression_model	Enhancers
3	0.04096139594912529	train	1	MMNN_boosted_regression_model	Enhancers
4	0.059692129492759705	validation	1	MMNN_boosted_regression_model	Enhancers
5	0.05595569312572479	test	1	MMNN_boosted_regression_model	Enhancers
6	0.043311942368745804	train	2	MMNN_boosted_regression_model	Enhancers
7	0.05226903408765793	validation	2	MMNN_boosted_regression_model	Enhancers
8	0.04611706733703613	test	2	MMNN_boosted_regression_model	Enhancers

Figure 7: Training history of the MMNN model for the enhancers task.

be rejected (p-value $\leq$0.05). If this is not the case (p-value $>$ 0.05), then it assumed that there is no statistical difference between the models' performances. From the figure 8 it seems clear that CNN performs worse than the other models and the Wilcoxon test confirms this assumption. Figure 9 shows the results of this test.

6. Conclusion

This project explores the task of predicting active enhancers and promoters for a specific tissue, namely the GM12878 cell-line, through Deep Learning methods. The focus is on building a statistically robust methodology for performing the task and comparing the results from different models. However, there are several aspects that are simplified and can be improved: for example, more window sizes different from 256 can be tried to preprocess epigenomic data, more splits for the K-Fold CV procedure can be



Task	Window Size	First Model	Second Model	p_value	Statistically significant difference?
Enhancers	256	FFNN_regression_model	CNN_regression_model	0,00390625	Yes
Enhancers	256	FFNN_regression_model	MMNN_boosted_regression_model	0,65234375	No
Enhancers	256	MMNN_boosted_regression_model	CNN_regression_model	0,00390625	Yes
Promoters	256	FFNN_regression_model	CNN_regression_model	0,00390625	Yes
Promoters	256	FFNN_regression_model	MMNN_boosted_regression_model	0,12890625	No
Promoters	256	MMNN_boosted_regression_model	CNN_regression_model	0,00390625	Yes

Figure 9: Results of the Wilcoxon test.

generated and a more exhaustive Bayesian search can be performed on the hyperparameters space.

References

[1] Y. Li, W. Shi, and W. Wasserman, "Genome-wide prediction of cis-regulatory regions using supervised deep learning methods," *BMC Bioinformatics*, vol. accepted, 05 2018.

[2] H. Bu, J. Hao, Y. Gan, S. Zhou, and J. Guan, "Deepsen: a convolutional neural network based method for super-enhancer prediction," *BMC Bioinformatics*, vol. 20, 12 2019.

[3] C.-H. Fang, N. Theera-Ampornpunt, M. Roth, A. Grama, and S. Chaterji, "Aikyatan: mapping distal regulatory elements using convolutional learning on gpu," *BMC Bioinformatics*, vol. 20, 10 2019.

[4] J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio, "Algorithms for hyper-parameter optimization," 12 2011.

[5] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh, "High dimensional Bayesian optimization with elastic Gaussian process," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2883–2891. [Online]. Available: https://proceedings.mlr.press/v70/rana17a.html

[6] E. Brochu, V. Cora, and N. Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 12 2010.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," 2012.

[8] P. Kumbhar and M. P. Mali, "A survey on feature selection techniques and classification algorithms for efficient text classification," 2016.

[9] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan, "A review of feature selection methods for machine learning-based disease risk prediction," *Frontiers in Bioinformatics*, vol. 2, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbinf.2022.927312

[10] L. S. Shapley, *A Value for N-Person Games*. Santa Monica, CA: RAND Corporation, 1952.

[11] B. Rozemberczki, L. Watson, P. Bayer, H.-T. Yang, O. Kiss, S. Nilsson, and R. Sarkar, "The shapley value in machine learning," 2022. [Online]. Available: https://arxiv.org/abs/2202.05594

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4768–4777.

[13] M. B. Kursa and W. R. Rudnicki, "Feature selection with the boruta package," *Journal of Statistical Software*, vol. 36, no. 11, p. 1–13, 2010. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v036i11

[14] "Encode project." [Online]. Available: https://www.encodeproject.org/

[15] [Online]. Available: https://genome.ucsc.edu/

[16] D. Pr, "Fantom5." [Online]. Available: https://fantom.gsc.riken.jp/5/

[17] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *ArXiv*, vol. abs/1802.03888, 2018.

Appendix

1. Further details on GPs' RBF kernel and activation functions

When performing Bayesian Optimisation with a GP prior, the RBF kernel is typically used due to its smoothness, although it may not always be the best choice, since the differences of all the features x_i have the same effect on the covariance [6]. The standard version of the RBF kernel can be formalised as shown in equation 6.

$$k(x_i, x_j) = \exp\left(-\frac{1}{2\theta^2}||x_i - x_j||^2\right) \tag{6}$$

The first of the commonly used acquisition functions is the Probability of Improvement (PI), shown in equation 7, that evaluates the surrogate model in the point most likely to improve the best value found so far, indicated with x^+. $\Phi(\cdot)$ is the normal cumulative distribution function, while μ and σ are, respectively, the mean and the variance of the model at the considered point, x. The major drawback of the PI function is that it is purely exploitative, subsequently, in order to mitigate this tendency a parameter ξ added.

$$PI(x) = \mathcal{P}(f(x) \geq f(x^+) + \xi) = \Phi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right) \tag{7}$$

The second acquisition function is the Expected Improvement (EI), which quantifies the improvement with respect to the best point found so far and samples the point that maximises the expected improvement, thus solving issues such as underexploration or blocking in a local minimum that affect the PI function. Equation 8 shows the generalised form of the EI function, where $\phi(x)$ and $\Phi(x)$ represent, respectively, the probability density function and the cumulative distribution function.

$$EI(x) = \begin{cases} (\mu(x) - f(x^+) - \xi)\Phi(Z) + \mu(x)\phi(Z), if \sigma(x) > 0 \\ 0, if \sigma(x) = 0 \end{cases} \tag{8}$$

The last commonly used acquisition function is the confidence bound criteria, which uses a non-negative parameter k as a multiplier for the variance in the formulas indicated in 9 (Lower Bound Criteria) and 10 (Upper Bound Criteria).

$$LCB(x) = \mu(x) - k\sigma(x) \tag{9}$$

$$UCB(x) = \mu(x) + k\sigma(x) \tag{10}$$

2. Feature Importance with BorutaSHAP

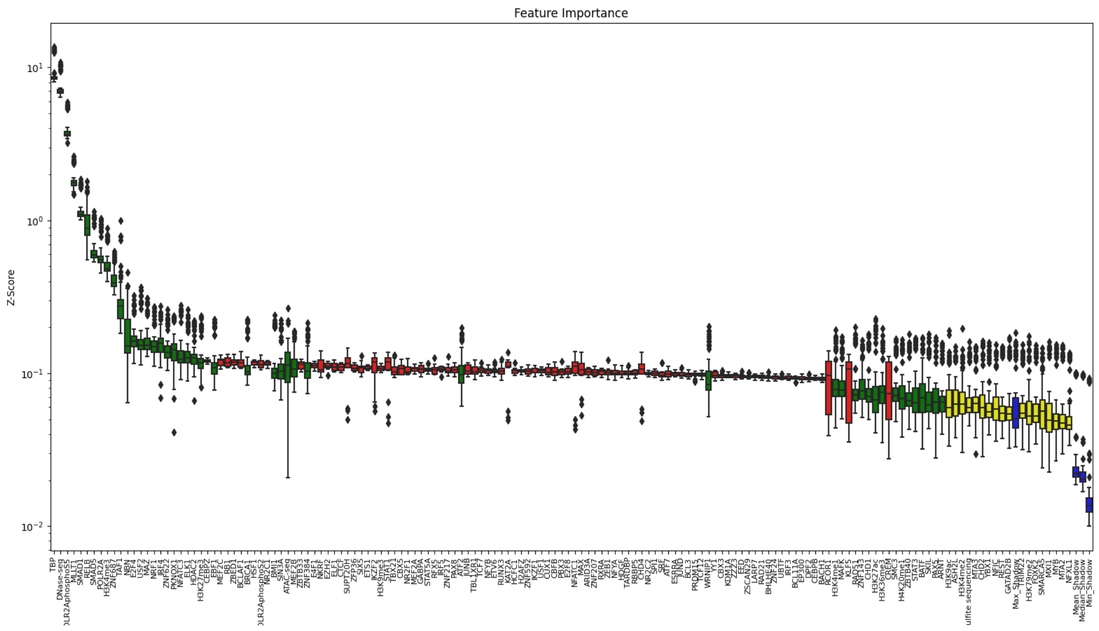
As in the classic Boruta algorithm, BorutaSHAP follows some predefined steps to estimate feature importance. Essentially, each feature is copied at least 5 times [13] in the so called *shadow features*, whose values are randomly shuffled instance-wise in order to remove any correlation with the response. Hence, by definition, the importance of a shadow feature should be zero (actually close to zero, due to random fluctuations) and the set of importances of the shadow attributes is used as reference to assess whether a feature is actually important. Then, in the original Boruta implementation, a RF is executed to estimate the feature importances by means of a value called *z-score*. The maximum z-score among shadow features is used as threshold to find important and unimportant features and, since there may be features with undetermined importance, the procedure is typically repeated multiple times. Time complexity is a major drawback of the Boruta algorithm, since a permutation-based approach is followed and the procedure must be repeated to obtain statistically significant results. There are at least two advantages in using BorutaSHAP, including a reduced time complexity of O(TLD^2), where T is the number of trees, L the maximum number of leaves among the trees and $D = log(L)$ the depth [17] and the possibility to use GPUs to parallelise the computations. Moreover, the SHAP values guarantee the consistency of feature importances, which is also ensured by permutation-based methods, differently from other common approaches such as gain and split count that are not consistent [17]. Two boxplots representing the feature importance estimated by the BorutaSHAP algorithm are shown in figure 10. The other boxplots are provided within the project repository. More precisely, green boxes represent accepted features, red boxes the rejected ones, yellow boxes the tentative features and blue boxes refer to shadow features.

3. Statistical Tests Details

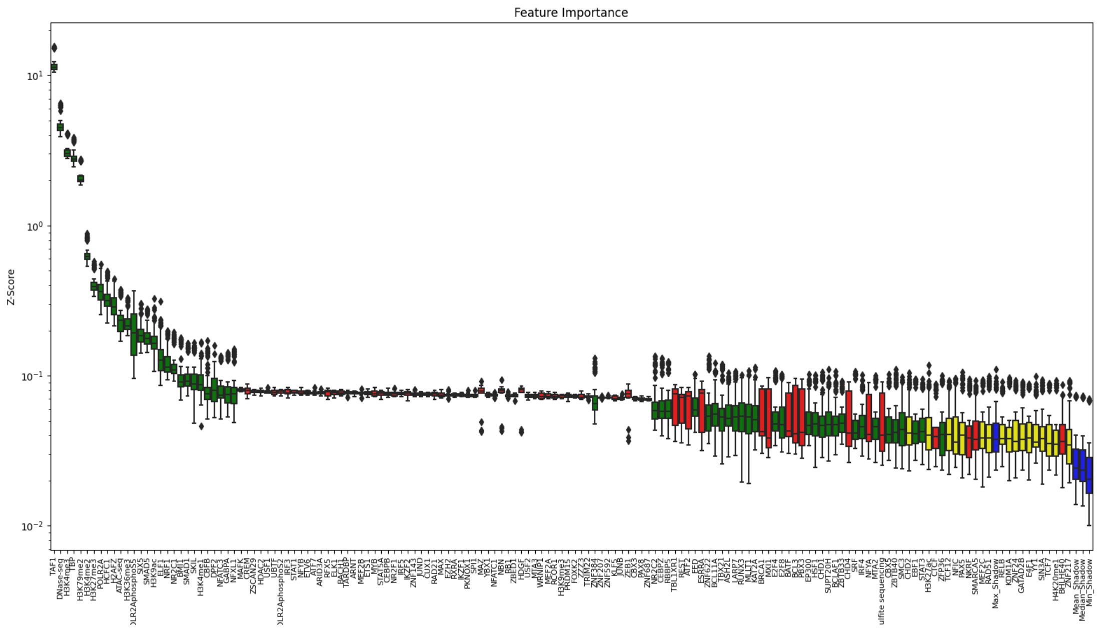
Here, further details are reported for the Pearson and Spearman tests. Pearson's test estimates the so called *Pearson's r* (Pearson product-moment correlation coefficient - PPMCC), that in turn estimates the linear correlation between two sets. Pearson's r can assume values in the range -1 (exact negative correlation, implying that as x increases, y decreases) and +1 (exact positive correlation), with 0 implying no correlation. Essentially, the formula is a ratio between the covariance of the two considered variables and the product of their standard deviations, as shown in equation 11. A null hypothesis stating that the distributions of the samples are uncorrelated and normal is tested, hence a p-value is defined to retain (p>0.05) or reject (p≤0.05) this null hypothesis. Spearman's test estimates the so called Spearman rank-order correlation coefficient, which is a nonparametric measure that describes the monotonicity of a relationship (whether linear or not) and can assume values between -1 and +1 (analogously to the Pearson test). The formula used for computing the coefficient is reported in equation 12, where ρ is the Pearson's r applied to the rank variables R(X) and R(Y). Also in this test the p-value is used to reject a null hypothesis, stating that the Pearson's r is 0.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{11}$$

$$r_s = \rho_{R(X),R(Y)} = \frac{cov(R(X),R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}} \tag{12}$$



(a) Feature importance boxplot for the enhancers, related to the first fold of the training procedure.



(b) Feature importance boxplot for the promoters, related to the first fold of the training procedure.

Figure 10: Boxplots representing the estimated feature importance for the features of the enhancers and promoters.