Bioinformatics project: active regulatory regions prediction using ML

Naomi Demolli

Abstract

Gene expression is regulated by noncoding regions of DNA called cis-regulatory (CRRs). The prediction of tissue-specic active regulatory regions in the human genome is a relevant problem in Bioinformatics and Molecular Biology. In this project I developed deep neural models able to predict whether a regulatory regions are active or not in the A549 cell line.

The performance obtained depends on the model; considering only the AUPRC metric, which is more suitable when dealing with unbalanced data sets, the performance do not appear to be particularly high.

In the end, the optimization performed on the hyperparameters of FFNN did not make relevant improvements to the classification task.

1. Introduction

Gene expression is the complex process by which the cell transforms the information contained in its DNA into molecules that perform functions necessary for metabolic and structural activities. Gene expression is regulated by cis-regulatory non-coding regions of DNA, such as enhancers and promoters. Promoters are sequences of nucleotides, located near a gene, which represent the starting point of the transcription. Enhancers are DNA sequences that help induce transcription of a gene, but they're far from the gene itself (this makes more difficult the identification of these regions).

The goal is to predict which regions are active or inactive in a cell line, this may be useful due to the critical role they play in controlling transcription (CRRs are involved in the development of different cell types/tissues and several lines of research show that genetic variants in regulatory regions may be deleterious or directly involved in genetic diseases in the timing and intensity of gene expression during the cell life [1]).

There are two tasks performed on the data:

- Active enhancers vs inactive enhancers (AE vs IE)
- Active promoters vs inactive promoters (AP vs IP)

Once the data were obtained and preprocessed, I turn the problem into a classification task and define fixed learning models to solve it, such as Perceptron, FFNN and CNN. After that, an attempt is made to optimize the hyperparameters of the FFNN model, in order to verify an increase in performances.

2. Models

In several studies it has been verified how the use of deep learning methods can improve our knowledge regarding the genomic locations of cis-regulatory regions [2]. The initial goal of the project is to make a comparison of the performances of 5 fixed models, Perceptron, CNN, FFNN, MMNN, and boostedMMNN

For each model I use a binary cross entropy loss due to the fact that it is a binary classification task.

2.1. Perceptron

Perceptron is a linear classifier used in supervised learning: each input is multiplied by its weight, the resulting values are summed to then apply an activation function (such as Sigmoid).

2.2. Feed Forward Neural Network

Neural networks are a broad class of predictors capable of performing different and complex tasks.

Layer (type)	Output Shape	Param #
epigenomic_data (InputLayer)	[(None, 44)]	0
dense_2 (Dense)	(None, 1)	45

Figure 1. perceptron summary

They are composed of interconnected processing units of the form $g(x) = \sigma(w^T \dot{x})$ with σ usually nonlinear activation function. In order to estimate how valid the classification made by the network is, it is necessary to define a cost function $\ell(y, \hat{y})$; improving net performance, by modifying its weights and bias, is a problem of minimizing the cost function $\ell(y, \hat{y})$. A commonly used optimization algorithm in neural networks is the gradient descent algorithm or one of its variants (in this project I use Nadam). A technique called backpropagation is used to compute gradients.

A neural network has a set of hyperparameters to be defined before learning (such as number of layers, number of neurons, learning rate etc).

In this project I referred to the architecture decided in paper [1], in fig 2 the architecture used.

Layer (type)	Output Shape	Param #
epigenomic_data (InputLayer)	[(None, 44)]	0
dense_4 (Dense)	(None, 16)	720
dense_5 (Dense)	(None, 4)	68
dense_6 (Dense)	(None, 2)	10
dense_7 (Dense)	(None, 1)	3

Total params: 801
Trainable params: 801
Non-trainable params: 0

Figure 2. FFNN summary

2.3. Convolutional Neural Network

CNN are a family of neural networks widely used in the field of computer vision. A convolutional neural network initially focuses on features simple features and gradually extrapolates increasingly complex features that combined return the output. Convolutional layers are the fundamental layers of this type of network; they contain a set of filters (kernels) that convolve with the image and create an activation maps.

As for the FFNNs, the value of hyperparameters must first be decided for CNNs.

In this project I used the architecture specified in the paper [1], fig. 3 shows the architecture used.

Layer (type)	Output Shape	Param #
sequence_data (InputLayer)	[(None, 256, 4)]	0
conv1d_2 (Conv1D)	(None, 251, 64)	1600
max_pooling1d_1 (MaxPooling 1D)	(None, 125, 64)	0
conv1d_3 (Conv1D)	(None, 123, 128)	24704
global_average_pooling1d_1 (GlobalAveragePooling1D)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 10)	1290
dense_7 (Dense)	(None, 1)	11

Total params: 27,605
Trainable params: 27,605
Non-trainable params: 0

Figure 3. CNN summary

2.4. Multi-Modal Neural Network

The information in real world usually comes as different modalities. Due to the distinct statistical properties of different information resources, it is very important to discover the relationship between different modalities. Multimodal learning is a good model to represent the joint representations of different modalities.

3. Method

In order to identify active cis-regulatory regions, I initially use models such as FFNN and CNN and MMNN. For each model I develop a *base version* and then I compare their performances on the task.

Subsequently I develop an *optimized version* only for FFNN (due to time and computational constraints) in which the selection of hyper-parameters is made by Bayesian optmization. The hyper-parameters chosen for optimization are shown in Fig. 4. I finally verify if the perfomance of model that have hyper-parameters tuned is better.

All models are developed using Keras with the TensorFlow backend.

3.1. Holdout

All classification tasks, performed by *base model*, are executed using 50 stratified hold-outs, each

Hyper - parameters	Values
num_units_1	16, 32, 64, 128, 256
num_units_2	4, 8, 16, 32 64
num_units_3	2, 4, 8, 16
dropout-rate_1	0.0 to 0.5, step = 0.1
dropout-rate_2	0.0 to 0.5, step = 0.1
dropout-rate_3	0.0 to 0.5, step = 0.1

Figure 4. FFNN hyper-parameters

composed by splitting the dataset in a training part (80% of samples) and in a test part (20 % of samples). Stratified hold-outs preserve the percentage of samples for each class, useful in case of unbalance data.

For hyper-parameter optimization, 25 stratified holdouts were generated. Each holdout was split into a training set (80% of samples) and a test set, each training set is split in turn into a sub-training set and a validation set (20%); the Bayesian optimization is performed on 7 trials, each model is trained on the sub-training set and tested on the validation test. The best model is re-trained on the whole training set and evaluated on the test set. Subsequently it's calculated the average performance of the top 25 models and is compared with the average performance of the *baseline model* .

Initially, val_accuracy was chosen as the metric to evaluate the best model during Bayesian optimization. Subsequently, either because of the unbalanced dataset or because there was no improvement in performance, another optimization was performed using AUPRC as a metric

4. Dataset

In this project I exploit data from ENCODE project and FANTOM consortium. The first one was launched by the US National Human Genome Research Institute (NHGRI) in September 2003 and aims to identify functional elements in the human genome. FANTOM (Functional ANnoTation Of the Mammalian genome) is an international research consortium led by RIKEN focused on functional annotation of mammalian genomes and characterization of transcriptional regulatory networks

The work to retrieve the epigenomic data from ENCODE and the labels from FANTOM is already done by Luca Cappelletti, indeed we can exploit the epigenomic_dataset package [3] to automatically download the data.

Even though from the FANTOM5 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. In this project I focus on a single cell line, A549. A549 cells are adenocarcinomic human alveolar basal epithelial cells, and constitute a cell line that was first developed in 1972 by D. J. Giard, et al. through the removal and culturing of cancerous lung tissue in the explanted tumor of a 58-year-old caucasian male.

I get the labels for the machine learning task from FANTOM dataset, the dataset has the structure of a simple BED file format (with also the optional field, strand) in which the first 4 columns are the coordinates of the region, while the last one is the label.

I get the epigenomic data from ENCODE project, as for the labels, the dataset has BED columns, the other columns are the names of 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.

Convolutional Neural Networks (CNN), on the other hand, are able to take into account the spatial regions, so the data were transformed into sequential type data. Therefore I have two different representations for each cis-regulatory region: one with a set of numerical features to train models such as Perceptron and FFNN and the other as a sequences of nucleotides to feed the CNN.

5. Binarization of the labels

I need to binarize values of labels to have a classification task. First of all, I plot the distributions of label for promoters and enhancers

For both promoters and enhancers the distribution is unbalanced (almost exponential), moreover the ranges of values are very different from each other [fig. 5]

In order to binarize the labels there are several approaches:

- the FANTOM suggested threshold at 1TPMs for both promoters and enhancers, this is good for promoters with an unbalance of 0.204, but on this cell line it would cause a high unbalance of 0.0037 for enhancers [fig. 6]

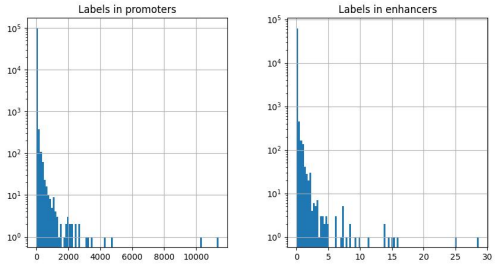

Figure 5. Distributions of labels without binarization

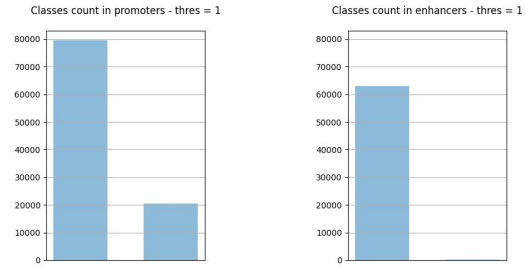

Figure 6. Class count with 1TPMs threshold both for promoters and enhancers

- another approach suggested by FANTOM authors is setting threshold at 1TPMs, dropping values between 0 and 1 for both promoters and enhancers. This reduces the unbalance both for promoters and enhancers, respectively at 0.255 and 0.0038. Focusing on the highest unbalance, namely that of enhancers, this decrease is not enough to remove part of the samples from the dataset, in addition the improvement for promoters' labels is small [fig 7]

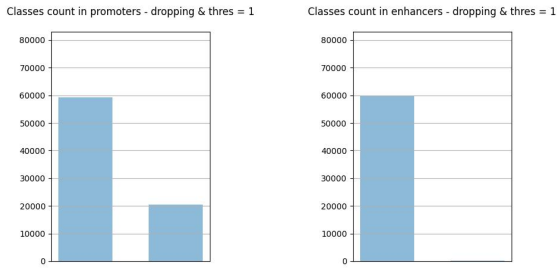

Figure 7. Class count with 1TPMs threshold and drop values both for promoters and enhancers

- in the end, I decided to choose two different values for the thresholds - remembering that the ranges are very different - and not to drop values between 0 and 1: for promoters I left the suggested threshold of 1TPMs and thus an unbalance of 0.204, for enhancers I chose threshold at 0TPMs - almost all values of enhancers labels are 0 (third quartile is 0) - with an unbalance of 0.057.

  (Warning: I'm not focusing on the biological meaning of my choices, the threshold are chosen to obtain a dataset useful for learning)

6. Pre-processing

First thing to do is check the rate between number of samples and features in the dataset: in the case of promoters is 2080.9, while 1318.3 for enhancers.

The second step is imputation of missing values:

- for promoters only the 0.007% is NaN values, the sample with most values has 2 NaN values out of 48 and the feature with the most missing values has 189 NaN values out of 99881

- for enhancers only the 0.001% is NaN values, the sample with most values has 1 NaN values out of 48 and the feature with the most missing values has 32 NaN values out of 63285

There are several approaches to replace NaN values, I decided to use nearest neighbors imputation: each missing feature is imputed using values from k nearest neighbors that have a value for the feature. The features of the neighbors are averaged uniformly, K is an hyperparameter and is set to 5 in this project.

In addition, it must be check the presence of features whose values are constant: these features don't provide any information to the target feature and are redundant data available in the dataset. In the epigenomic data of cell line A549 there are no costant features for either promoters or enhancers.

In the end, it is necessary to normalize the data: I used a better version of Z-score named Robust Scaler that scales features using statistics that are robust to outliers (this Scaler removes the median and scales the data according to the quantile range)

7. Correlations

Once the pro-processed dataset is obtained, it is needed to verify the correlation between features and output and the correlation between features.

Correlation is a statistical measure that tells us about the association between the two variables, it describes how one variable behaves if there is some change in the other variable. I used two different correlation coefficient, Pearson and Spearman.

- Pearson coefficient measures the linear correlation between to variables (it has a value between +1 and -1, with +1 total positive linear correlation, 0 no linear correlation and -1 negative linear correlation)
- monotonic function (perfect Spearman correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other)

For promoters the features EHMT2, FOSB and RNF are considered unrelated to the output for Pearson, the feature ZFP36 is evaluated uncorrelated with output for Spearman. For enhancers, both Pearson's test and Spearman's test identify 2 unrelated features with the output: ZC3H11A and CBX2 are not linearly correlated with output and features CBX2 and KDM5A are not correlated with output for Spearman's test. I then proceeded to eliminate the features not correlated to the output.

I used same correlation tests in order to identify interrelated features, no single feature turns out to be extremely correlated with another.

In figure 20 it's possibile to visualize the most correlated features for enhancers, it also worth notice the presence of class imabalance.

In figure 21 it's possibile to visualize also the most correlated features for promoters.

8. Feature selection

Features selection is a fundamental step in many machine learning pipelines, I would like to select only the relevant features from the whole set. In this project I used Boruta.

The algorithm Boruta uses a wrapper approach built around a random forest classifier (Boruta is a god of the forest in the Slavic mythology).

The importance measure of an attribute is obtained as the loss of accuracy of classification caused by the random permutation of attribute values between objects. Then the average and standard deviation of the accuracy loss are computed. Alternatively, the Z score computed by dividing the average loss by its standard deviation can be used as the importance measure, unfortunately the Z score is not directly related to the statistical significance of the feature importance returned by the random forest algorithm. Since we cannot use Z score directly to measure importance, we need some external reference to decide whether the importance of any given attribute is significant. The importance of a shadow attribute can be nonzero only due to random fluctuations. Thus the set of importances of shadow attributes is used as a reference for deciding which attributes are truly important. [4]

Neither for promoter nor enhancers Boruta algorithm considers necessary to eliminate some features.

9. Visualization

To better understand the distribution of the data, Principal Component Analysis (PCA) was used. PCA helps simplify data exploration and visualization, it's a technique used to identify a smaller number of uncorrelated variables known as principal components from a larger dataset.

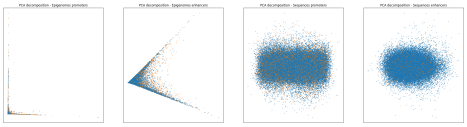

Figure 8. PCA decomposition for epigenomic and sequential data

As can be see in image 8 the sequential data are not particularly separable, it can be assumed that the classification will be particularly difficult and thus the performance will not be optimal. Epigenomic data are slightly more separable, but we do not observe a distinct clustering of the data

10. Results

10.1. Results for baseline models

To analyze the performance of the models, I consider three metrics: Area Under The Receiver-Operating Curve (AUROC), Area Under the Precision-Recall Curve(AUPRC) and accuracy

We can observe that the accuracy values, in fig 9, turn out to be high for each model; this can be misleading. In fact, taking into consideration the other two metrics, we observe a general drop in performance. This is probably due to class unbalance: accuracy is

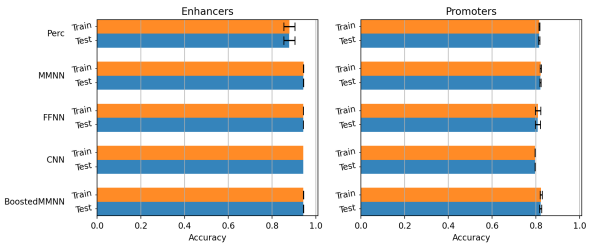

Figure 9. accuracy

the fraction of predictions our model got right, if the model starts classifying each sample as belonging to the majority class, I might have a very high accuracy but a low AUROC or AUPRC

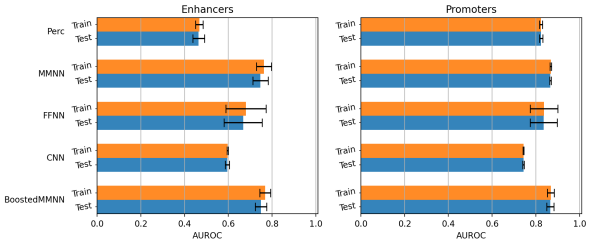

Figure 10. AUROC

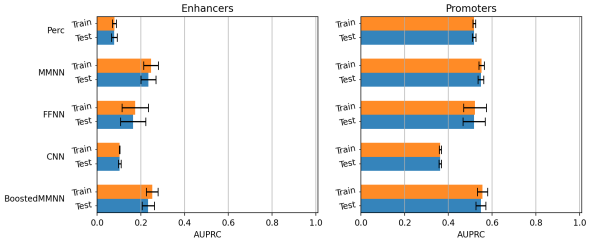

Figure 11. AUPRC

In general the performance of the models is better for the IP vs. AP task than for the IE vs. IE task, this is probably due to the fact that the classes result more unbalanced in enhancers. No strong overfitting occurred in any model.

Perceptron seems to have the worst performance in all three metrics; it is probably too simple model for the two tasks considered.

The FFNN model tends to have a high standard deviation even though the performances were calculated on 50 holdouts, this suggests that the model does not always converge to the same minimum point. This could be a problem; I would like to find a more stable model.

As seen in the dataset analysis, the sequential data were not very separable; in fact, it can be verified by looking at the metrics above that CNN performance is not particularly good

In general the boostedMMNN and MMNN seem to be the most suitable models for solving the tasks considered

I later used the Wilcoxon test to compare different models trained on the same data considering task IP vs. AP separately from task IE vs. IE.

I obtained that the performance of MMNN and BoostedMMNN are statistically identical for each metric and in each task.

In general the performance of Perceptron and CNN remains the worst, a comparison of the two showed that they are statistically different but which is better or worse depends on the metric considered and the task

10.2. Results of tuned models

Bayesian optimization was performed on the hyper-parameters of the FFNN. Initially, the metric considered to choose the best model was the val_accuracy.

Barplots summarizing the results can be seen in the figures 12, 13, 14, 15. Against expectations, the peformances of the optimized model turn out to be no better than the base model.

I thought that since the dataset was unbalanced, the choice of val_accuracy as a metric might be misleading. Therefore, I re-performed another optimization using as the metric for choosing the best model, the greater AUPRC

Barplots summarizing the results can be seen in the figures 16, 17, 18, 19. Even changing the metrics considered, the performance of the optimized model does not improve and is the same, in some cases worse, than the performance of the base model.

References

[1] Cappelletti, L. et al. (2020). *Bayesian Optimization Improves Tissue-Specific Prediction of Active Regulatory Regions with Deep Neural Networks*. In *Rojas, I., Valenzuela, O., Rojas, F., Herrera, L., Ortuño, F. (eds) Bioinformatics and Biomedical Engineering. IWBBIO 2020. Lecture Notes in Computer Science(), vol 12108. Springer, Cham.*

[2] Li, Y., Shi, W. and Wasserman. *Genome-wide prediction of cis-regulatory regions using supervised deep learning methods*. In *BMC Bioinformatics 19, 202 (2018)*

[3] Cappelletti, L. *github.com/AnacletoLAB/epigenomic_dataset*

[4] Kursa, M. B., and Rudnicki, W. R. (2010). *Feature Selection with the Boruta Package*. In *Journal of Statistical Software, 36(11), 1–13*

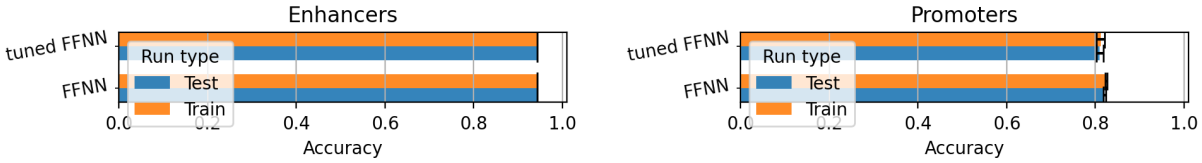

Figure 12. tuning by max val_accuracy

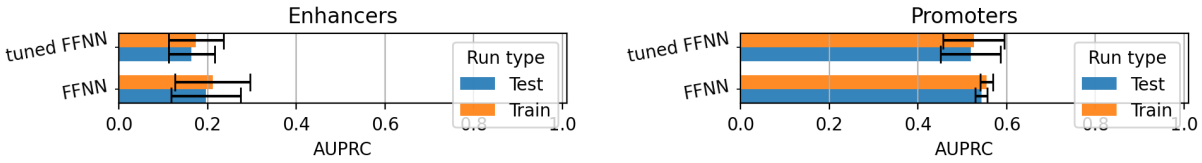

Figure 13. tuning by max val_accuracy

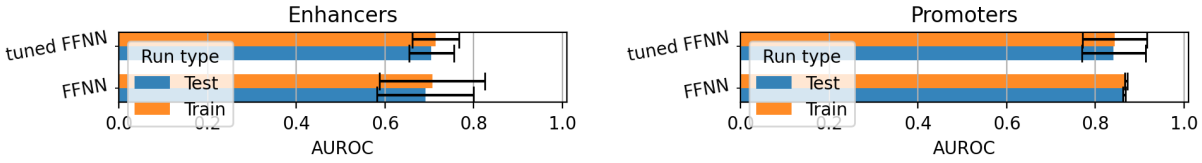

Figure 14. tuning by max val_accuracy

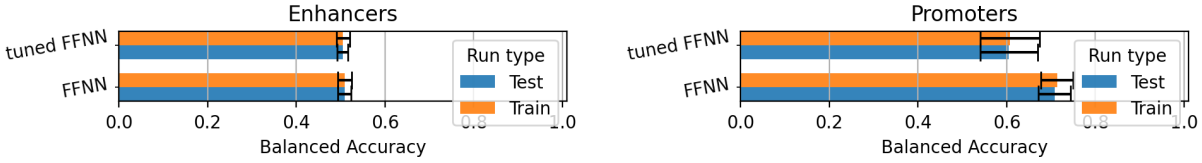

Figure 15. tuning by max val_accuracy

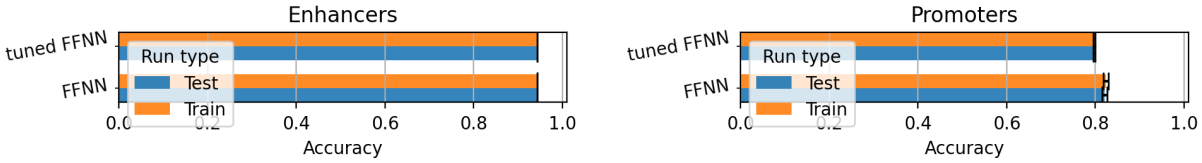

Figure 16. tuning by max auprc

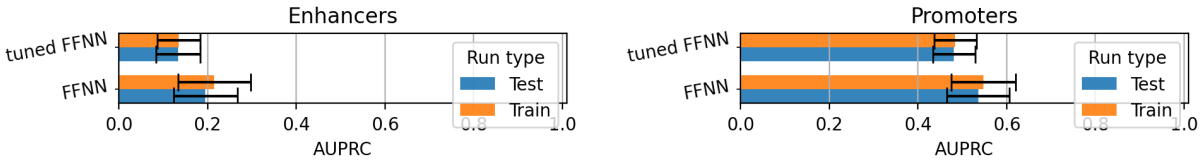

Figure 17. tuning by max auprc

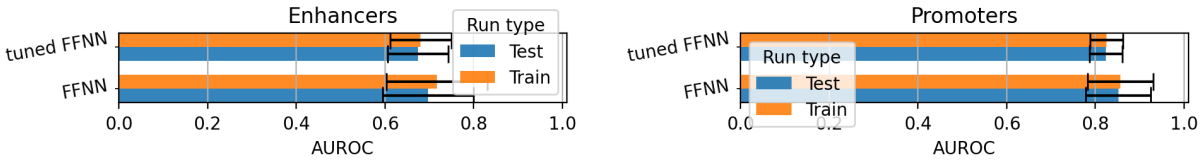

Figure 18. tuning by max auprc

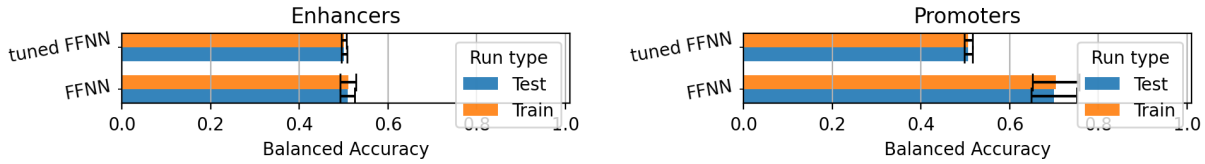

Figure 19. tuning by max auprc

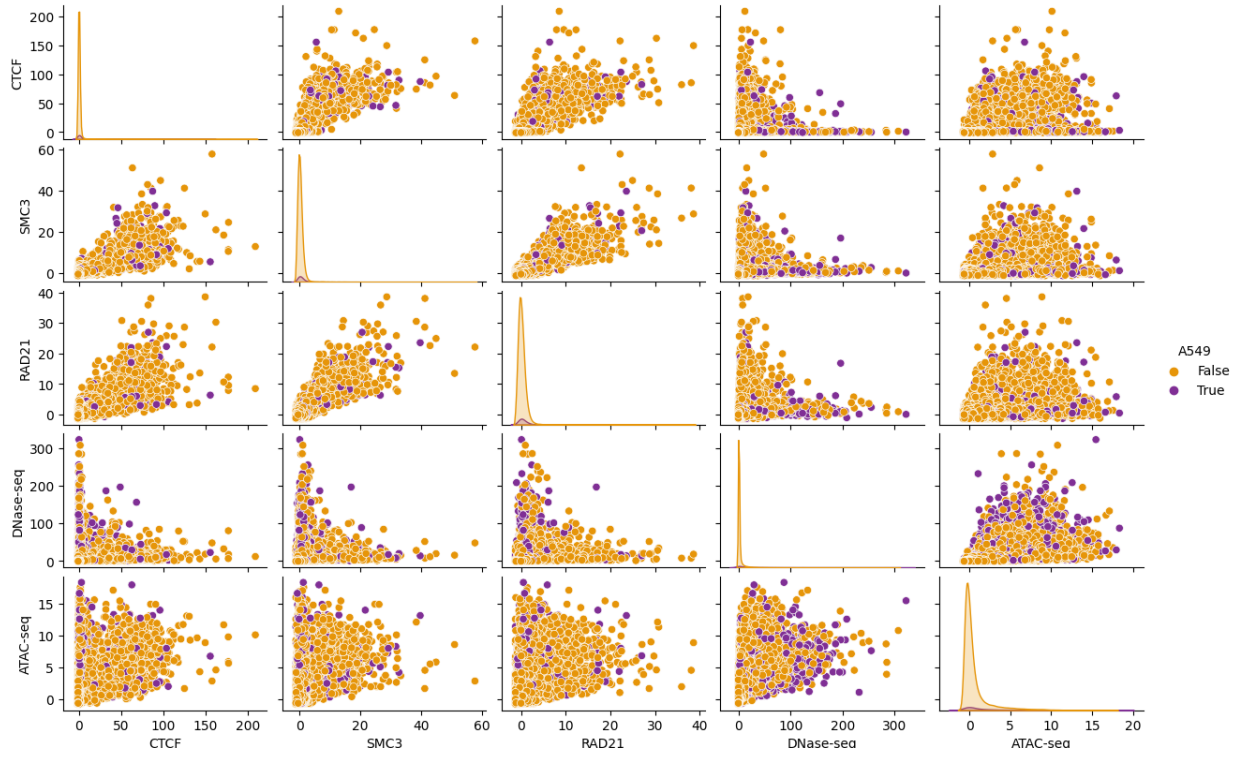

Figure 20. most correlated features for enhancer

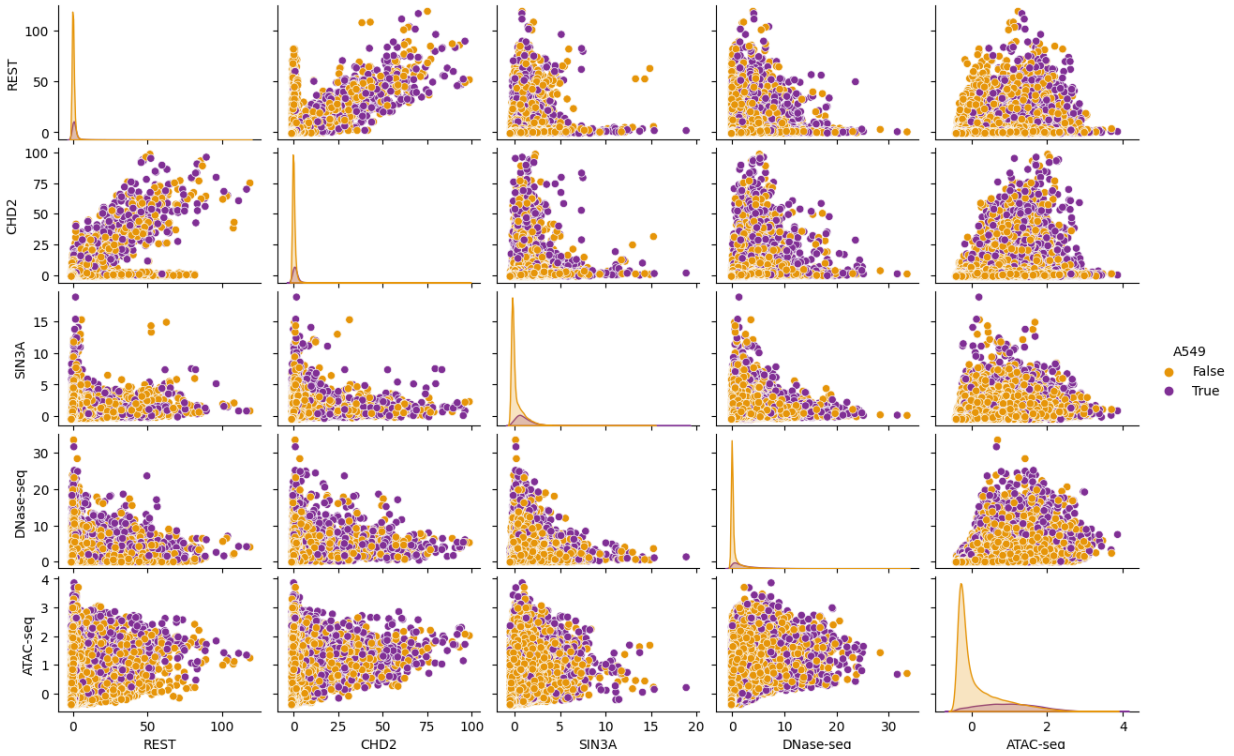

Figure 21. most correlated features for promoters