Bioinformatics project: active regulatory regions prediction using ML

Arianna Converti
https://github.com/AriannaConverti97/Bioinformatics_project

January 24, 2023

1 Introduction

Bioinformatics is an interdisciplinary field of biology, computer science, mathematics, and other. It is used to analyze and interpret biological data [5].

One of the fundamental pillars of biology is DNA.
The DNA is a polymer composed of two nucleotide chains that contains biological information. Each nucleotide is composed of one sugar called deoxyribose, a phosphate group and one of nucleobases (adenine, guanine, cytosine and thymine). The two chains are joined together with hydrogen bonds, according to base pairing rules (A-T and C-G). This bond makes the transcription process easy.

The transcription is the process of copying a segment of DNA into RNA, regulated by the two regulatory cis-regions: Enhancers and promoters. Promoters are a sequence of DNA (100-1000 base pairs) to which proteins bind that initiate transcription of a RNA. Instead, enhancers are a short (50-150 base pairs) region of DNA that can be bonded by proteins to increase the transcription of a particular gene.
The transcription is very important because the information transcribed into RNA can or not (about 98%) encode proteins, one of the basic ingredients for life.

This work focused on analyzing the regulatory regions in 'HEK293' cell line of the human genomic sequence 'hg38' and determining which regions are active and which are not.

Human Embryonic kidney 293 cells[6] (HEK293) is a cell line derived from an aborted fetus or human embryonic kidney cells grown in tissue culture taken from a female fetus in 1973. These cells are easy to grow in a culture and for this reason are used in many experiments that involve vaccine production and to study the protein expression.

2 Models

In this project, we want to apply some Neural Networks to predict the activity of regulatory regions in 'HEK293'. We use three different architectures:

- Perceptron
- Multi layer perceptron
- Convolutional neural network
- Multimodal neural network

2.1 Perceptron

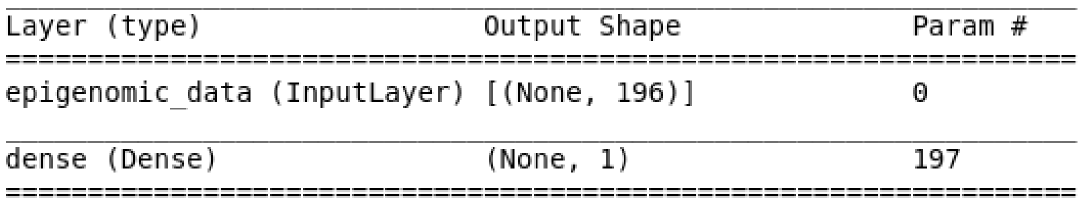
A perceptron [4] is the first network created by McCulloch and Pitts in 1943. It solves problems of classification data only if this data are linearly separable.

It classified data if the sum of products of weights of the input layers with the input data are greater than a threshold.

$$o = \begin{cases} 1 & if \sum w_i * x_i \\ 0 & otherwhise \end{cases} \tag{1}$$

In this way, we have created a model with two models:

- Input layer: its shape is equal to the number of the features in the epigenomic data;
- Dense layer: a fully connected layer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
epigenomic_data (InputLayer) [(None, 196)]             0
_________________________________________________________________
dense (Dense)                (None, 1)                 197
=================================================================
```

Figure 1: Perceptron structure

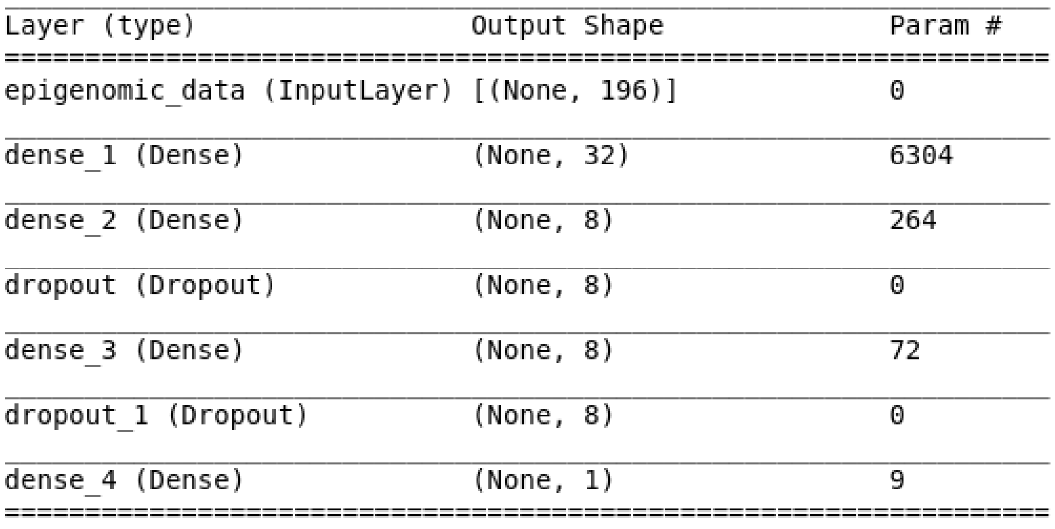
2.2 Multi Layer Perceptron

A Multi Layer Perceptron (MLP) [4] is a feed forward Neural Networks i.e an acyclic graph when the information travels in one way . It is used for predicting data that are not linear separable.

This NN is composed of several layers linked together: the first one is called input layers, the last one is called output layers and in between there are one or several hidden layers. For training MLP uses backpropagation.

The architecture of the MPL used in this work is composed of the following layers:

- Input layers and output layers are the same of the Perceptron;
- Dropout layers: it used to prevent overfitting. The randomly sets input units to 0 in each step of the training.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
epigenomic_data (InputLayer) [(None, 196)]             0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                6304
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 264
_________________________________________________________________
dropout (Dropout)            (None, 8)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 72
_________________________________________________________________
dropout_1 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 9
=================================================================
```

Figure 2: Multi layers perceptron structure

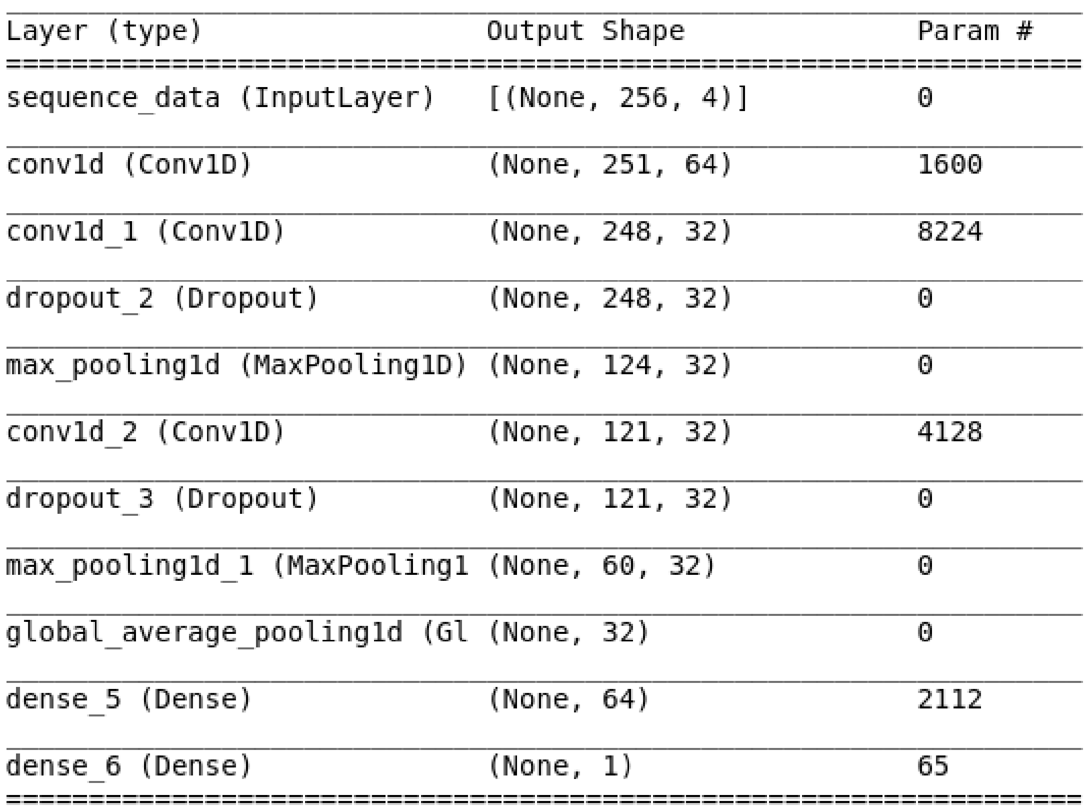
2.3 Convolutional Neural Networks

A convolutional Neural Network [4] is a deep neural network that applies convolutional operations. This operation consists of a shift of a convolutional kernel (matrix) along all the input dimensions and calculates the product. After this operation we obtain the features map that is used by the next layers.

Other operations that these networks made are pooling that reduces the dimension of data by extracting the max value of the cluster in the feature map.

The architectures of the CNNs built in this works is composed by the following layers:

- Input layers: the shape of the nucleotides considered.
- Conv1D: create a convolutional kernel and produce a tensor output.
- Dropout: The same layers of the precedent networks
- Max Pooling: it applies the pooling operation.
- Global Max Pooling: the same of the Max pooling but it aggregates an entire map in one value.
- Dense: fully connected layers.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequence_data (InputLayer)   [(None, 256, 4)]          0
_________________________________________________________________
conv1d (Conv1D)              (None, 251, 64)           1600
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 248, 32)           8224
_________________________________________________________________
dropout_2 (Dropout)          (None, 248, 32)           0
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 124, 32)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 121, 32)           4128
_________________________________________________________________
dropout_3 (Dropout)          (None, 121, 32)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 60, 32)            0
_________________________________________________________________
global_average_pooling1d (Gl (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
```

Figure 3: Convolutional Neural Networks structure

2.4 Multimodal Neural Networks

Multi-Modal Neural Network, that training the two previous models on their respective data, aims at solving the task concatenating them.

3 Experimental setup

3.1 The considered task

The experiment consists of studying two tasks:

- the active enhancers vs inactive enhancers

- active promoters vs inactive promoters

in the ‘HEK293’ cell lines, on the HG38 dataset.

The experiment is performed on “MSI NVIDIA GTX 1660 super” and all the neural networks are implemented using Keras[9]/TensorFlow2[8].

3.2 Datasets

The main datasets used in this projects are:

- the epigenomic data from ENCODE Encyclopedia of DNA Elements. It offers a selection of human and mouse genomes.
- labels from FANTOM5, reports the transcription events on human and mouse genome.
- genomic sequence from UCSC Genome Downloader. It is a genome browser created by the University of California, Santa Cruz.

To retrieve the data we used this libraries created by Luca Cappelletti:

- epigenomic_dataset: a data wrapper which download enhancers and promoters;
- ucsc_genomes_downloader: which download the reference genome;
- keras_bed_sequence: to bed the reference genome with respect to the epigenomic data.

The retrieved data is described in the following chart 4 and 1.

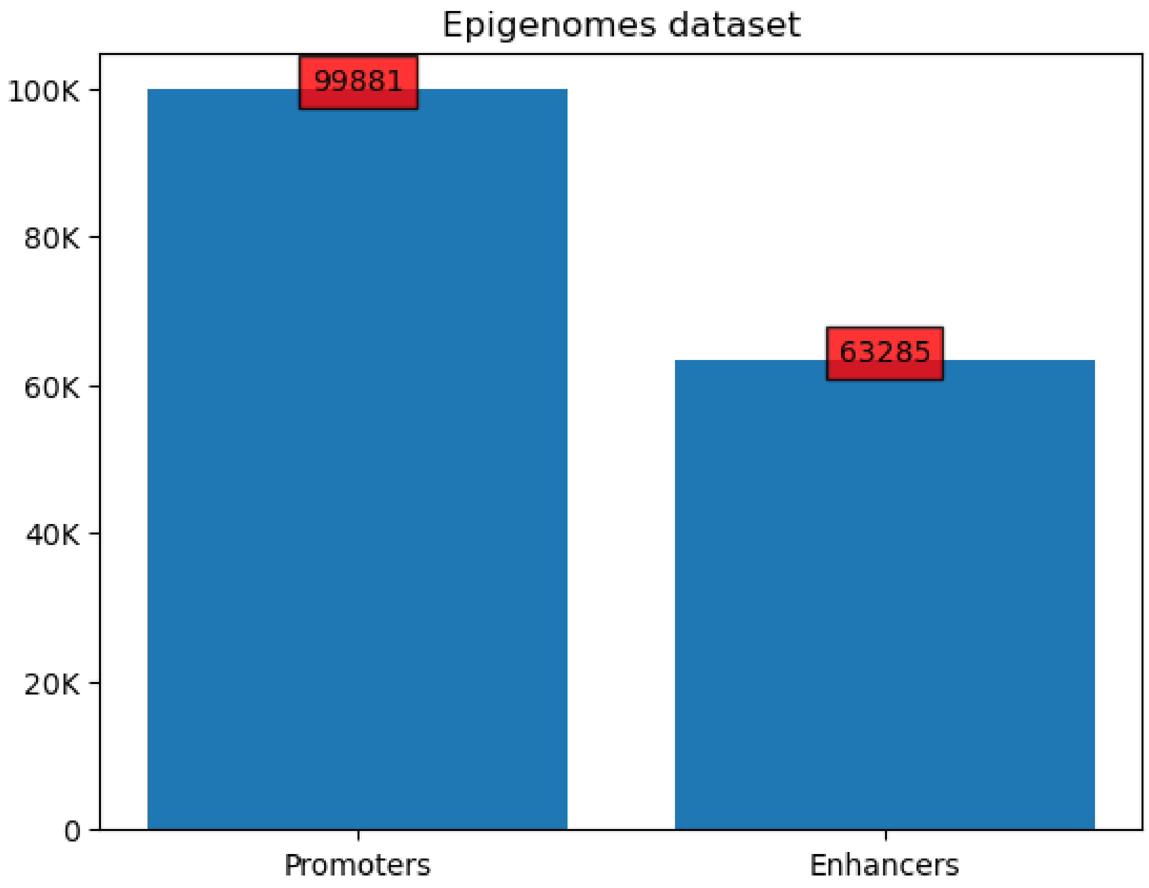


Figure 4: Epigenomes dataset

3.3 Data pre-processing

The data pre-processing is composed of five task:

- Rate between features and samples
- Nan detection and data imputation
- Constant features
- Normalization
- Class balance

3.3.1 Rate between features and samples

First of all, we analyze the rate between features and samples. If this rate is lower than one, it will provoke an overfitting problem. According to the Tab 1, both rates coming to promoters data and enhancer data are greater.

	Enhancers	Promoters
Number of samples	62385	99881
Number of features	196	196
Number of Nan	1	0
Rate between samples and features	510	323

Table 1: Information about the datasets.

3.3.2 Nan detection and Data Imputation

Mainly, the Nan-values are provoked by human error. They can provoke several problems: first of all we don't know how to use them, and secondly the machine can't use them for the computation.

In this task we want to check if the number of Nan-values are lower than 5%. And if this occurs, it is enough to impute the missing values with KNN Imputer. Otherwise, we drop this data but we lose information.

As we can see from Tab 1, we have only one missing value in epigenomic data. We used a KNN Imputer to compute a value to replace the missing value based on the 5 neighbor values.

3.3.3 Constant features

We look for features that have the same constant values and we drop them, because they don't add any significant information.

Fortunately, in this dataset we don't find any.

3.3.4 Normalization

Normalization is a method used to compare two or more features in the same range of values. This is done because features with different ranges of values can increase the learning complexity and to avoid this, we scale the values with robust Z scoring.

We use a robust statistic because it is resistant to error and outliers [28].

The robust Z scoring instead of subtracting the mean and dividing by standard deviation (which are heavily influenced by outliers), subtracts the median and divides by the interquartile range, as shown in the formula.

$$Z(x) = \frac{x - Q_2}{Q_3 - Q_1} \tag{2}$$

The result of this operation is to have all the features in the same range values.

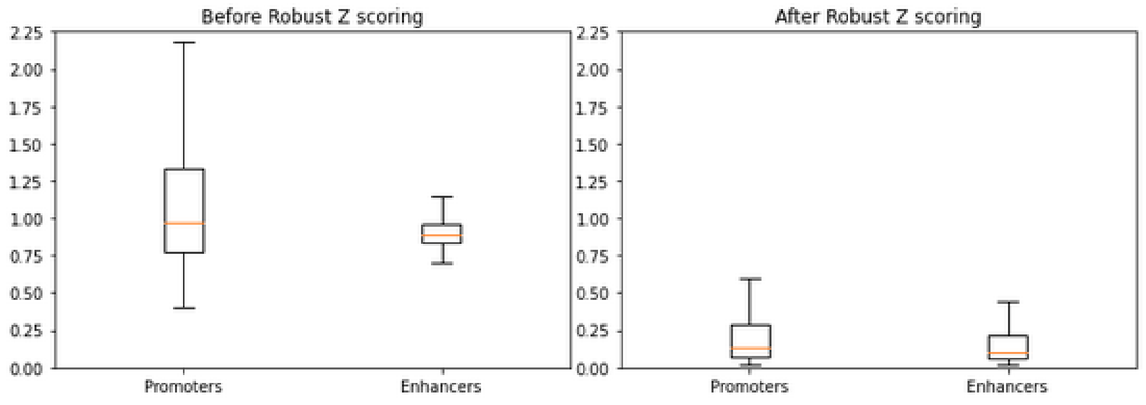


Figure 5: How to change the mean of data of both datasets.

3.3.5 Class balance and binarization

Strong imbalance in datasets can provoke a problem in the learning phase [29]. If the ratio between positive and negative examples is too low (or high), the classifier will predict the most common class.

According to the chart 6, this rate is higher than 1/10 so the class is approximately balanced.

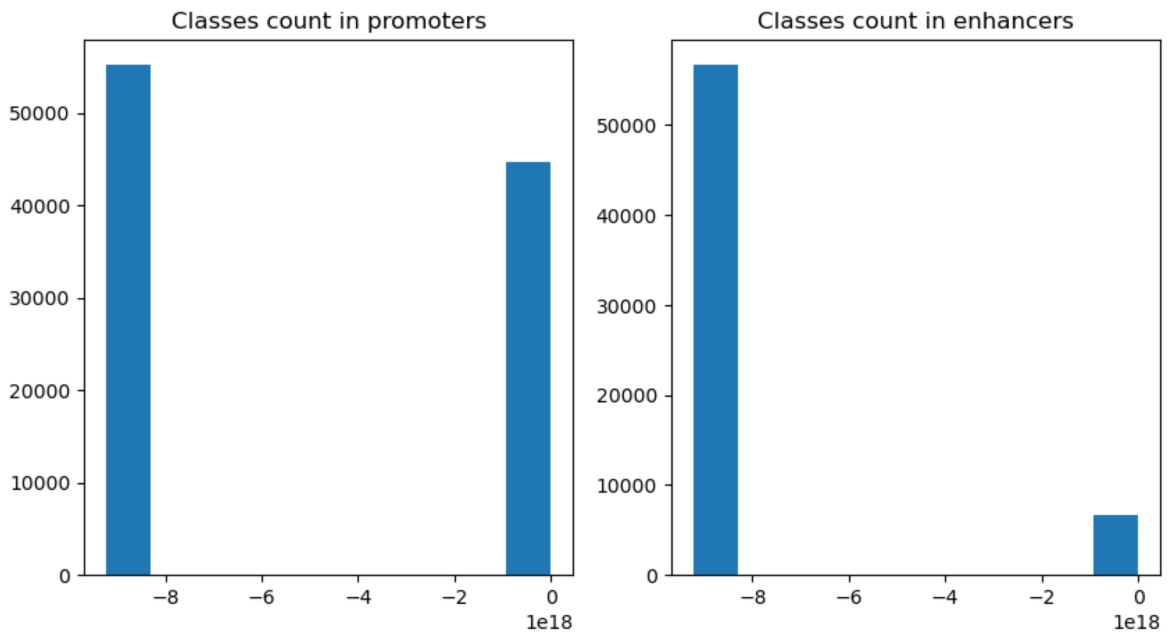


Figure 6: Class Balance

3.4 Data correlation and distributions

In this chapter, we want to do this task:

- Features distributions
- Correlation with output
- Correlation with features

3.4.1 Features distributions

Since the features are 196 we decide to visualize the most different features, using the euclidean distance.

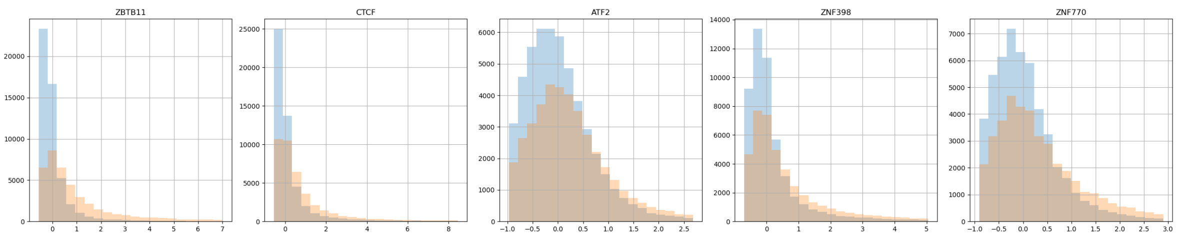


Figure 7: Top 5 different features from promoters.

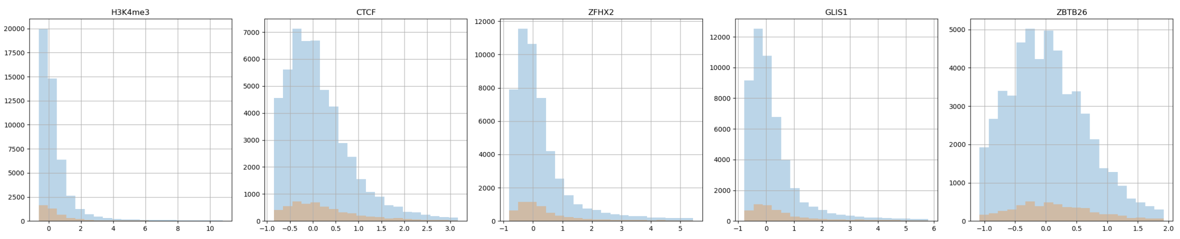


Figure 8: Top 5 different features from enhancers.

3.4.2 Correlation with output

We want to know if there are any features that don't have any correlations with his output. Figure 9 show some examples of correlations. If this happens we decide to drop it because they are useless and the dimensionality decreases.

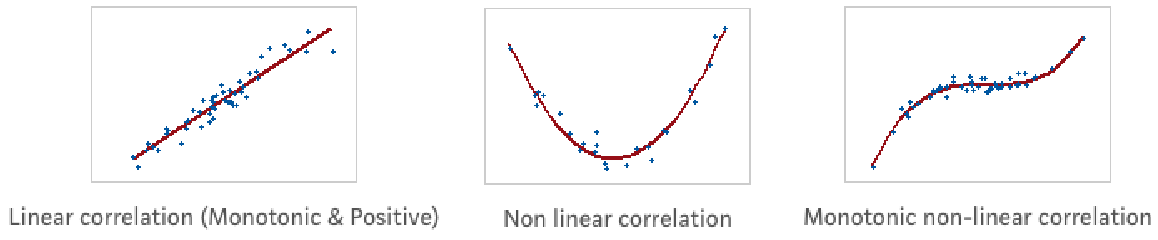


Figure 9: An example of correlation

We start with the linear correlation measured by Pearson coefficent [10]. It is the ratio between the covariance of two variables and the product of their standard deviations; thus, it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1. If the result is near to 0 the correlation is weak, otherwise if it is near to 1 or -1 the correlation is positively high or negatively high.

The second correlation we want to find is the monotonic correlation measured by the Spearman coefficient [11]. It is defined as the Pearson coefficient between the rank variables and in mathematics this is known as a weak order. The coefficient will be high, when observations have a similar rank, otherwise it will be low.

The last correlation is the non linear correlation measured by Maximal information coefficient[13]. A non linear relation is like a circular or polynomial function. This coefficient has a high computational cost.

At the end of this computation, we drop 21 features correlated with output in promoters data and 43 features in enhancers data.

3.4.3 Correlation with features

We have done the same thing to calculate the correlation between the features, but we haven't any significant correlation.

We visualize the most correlated features according to the Pearson coefficient, using the pairplot method (figure 10 and figure 11). More the data is near the diagonal the more the correlation is high.

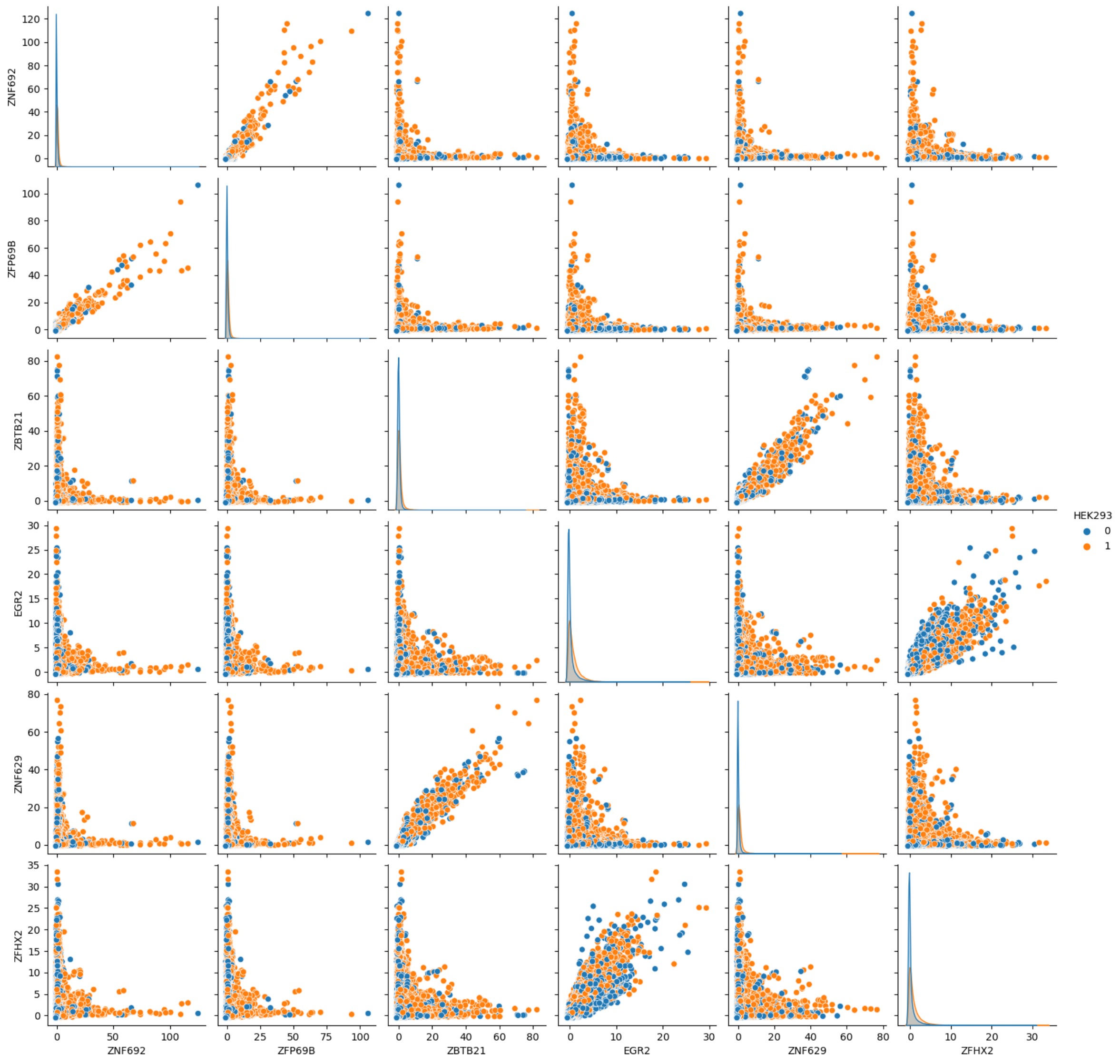


Figure 10: Most correlated features from promoters epigenomes

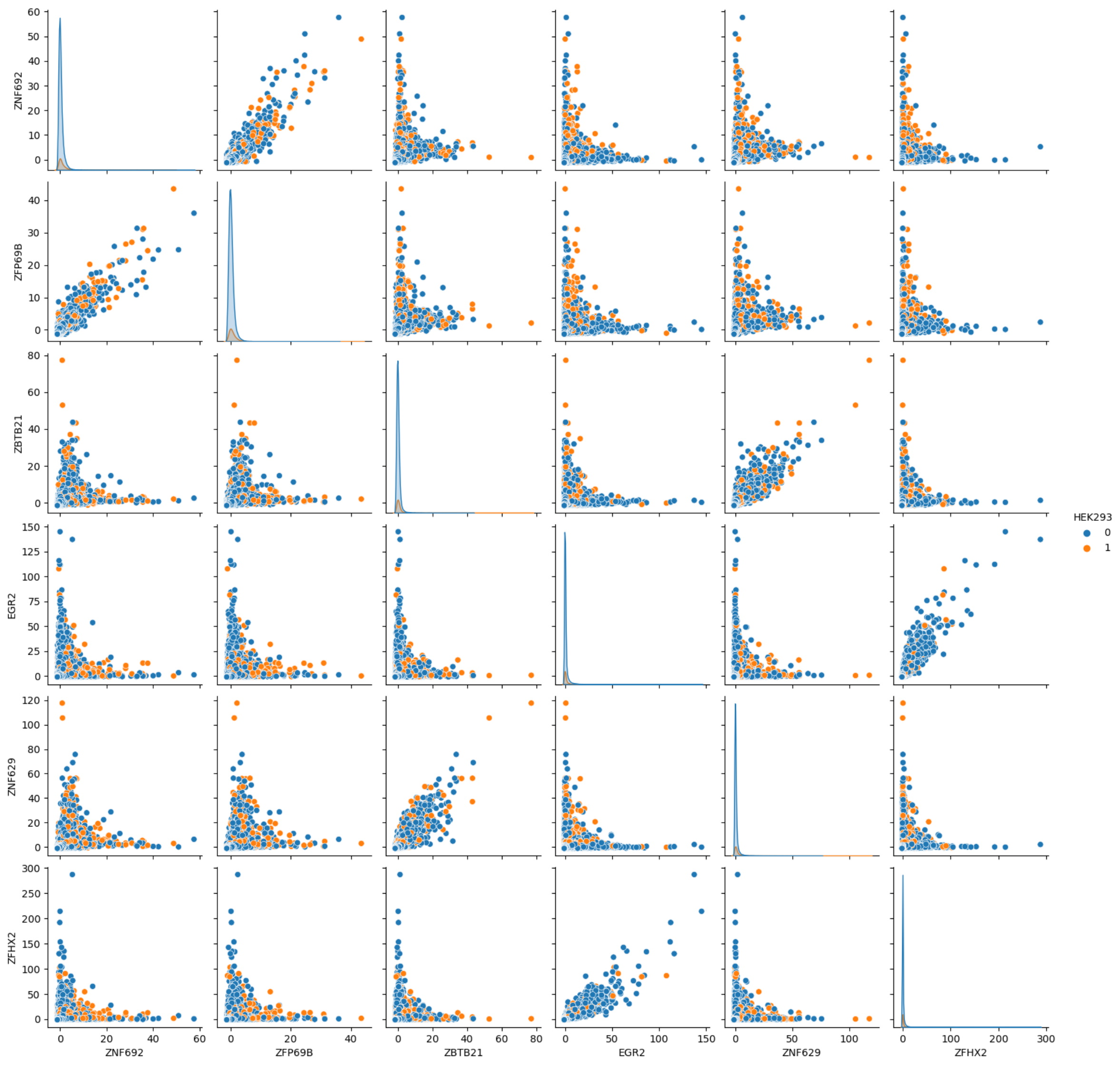


Figure 11: Most correlated features from enhancers epigenomes

3.5 Data visualization

Decomposition is useful to reduce the dimensionality of the datasets. We apply three methods:

- PCA [14]
- UMAP [15]
- t-SNE [16][17]

The figure 12 shows the mapping of the features into a 2-d plane. We color in purple the data labeled active and red the inactive ones.

3.5.1 PCA

Principal component analysis is a linear dimensionality reduction algorithm for analyzing large datasets containing a high number of dimensions/features. We want to visualize if the dataset is divided in clusters.

Both PCA graphs don't say much about the features, so we can proceed with the other two methods.

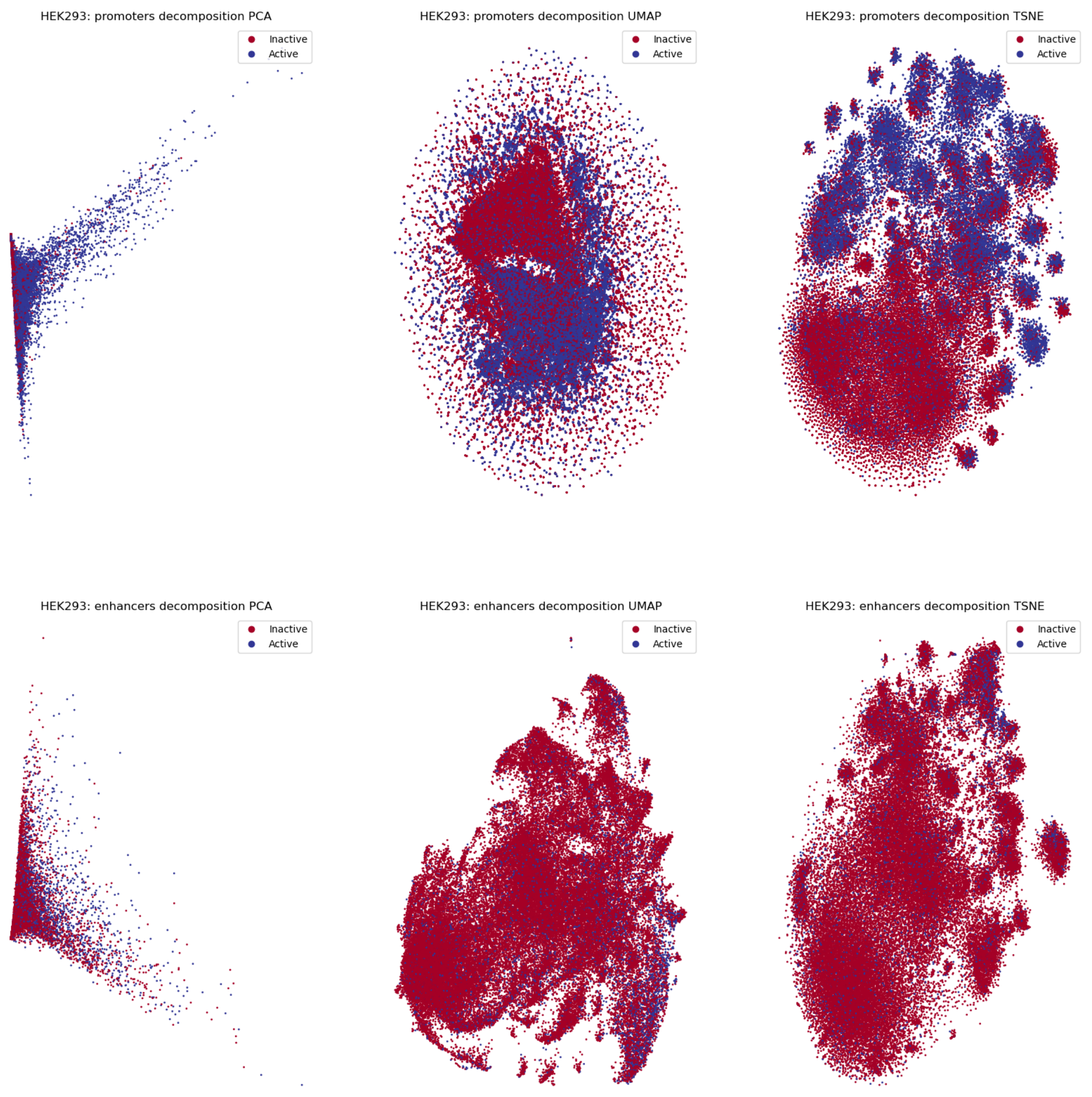


Figure 12: Data visualization

3.5.2 UMAP

Uniform Manifold Approximation and Projection is a dimensionality reduction method that creates low dimensional representations of the datasets, which can be used as input to simple models, reducing the computational time.

In this chart, we can find that promoters can be easily separated in two clusters, however in the enhancers data there is no way to visualize the cluster.

3.5.3 t-SNE

t-distributed stochastic neighbor embedding is another technique for visualizing high-dimensional data, among PCA this method nonlinear dimensionality reduction.

According to the chart, the promoters are divided into two clusters: the first one is on the bottom and is composed mainly of inactive promoters, and the second one is on the top of the charts and is composed of different clusters of active promoters.

Otherwise, for the enhancers data we can find many clusters composed mainly by inactive enhancers.

3.6 Holdouts

Holdouts is a method that splits a dataset into a 'train' and 'test' set. The training set is what the model is trained on, and the test set is used to see how well that model performs on unseen data.

In this work we perform only two holdouts, in which 80% of data for training and the remaining 20% of the data for testing. The splits are made by random.

3.7 Result analysis

To evaluate the model's performance we use get_complete_binary_metrics library from extra_keras_metrics by Luca Cappelletti. It returns a set of metrics that includes all binary metrics.

In this work we want to analyze:

- Accuracy calculate the percentage of correct predictions[20].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

- Precision is the fraction of relevant instances among the retrieved instances[21].

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

- Recall is the probability of a positive test result, conditioned on the individual truly being positive[22].

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

- Specificity is the probability of a negative test result, conditioned on the individual truly being negative[23].

$$Specificity = \frac{TN}{TN + FP} \tag{6}$$

- Balanced accuracy is the arithmetic mean of recall and specificity[24].

$$Balanced_accuracy = \frac{recall + specificity}{2} \tag{7}$$

- AUROC is the area under the curve of plotted True positive rate and False positive Rate [25].

- AUPRC is the area under the curve of plotted recall with respect to precision [26].

- F1 Score keeps the balance between precision and recall [27].

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

4 Result

Firstly we plot the loss, which for binary classification shows how close the prediction probability is closed to the true values. According to the chart 13, both for enhancers and promoters data the CNN loss is the best.

We then look at the AUROC, we can see that the MMNN scored an average of 0.71 for AEvsIE and of 0.89 for APvsIP, according to figure 14. This is the best outcome.

The only model that performs poorly is the CNN on the epigenomic data. We are not impressed because, like the decomposition shown, the task is hard to accomplish.

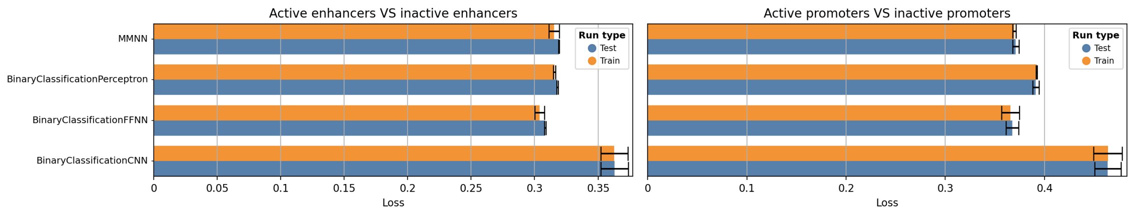


Figure 13: Loss

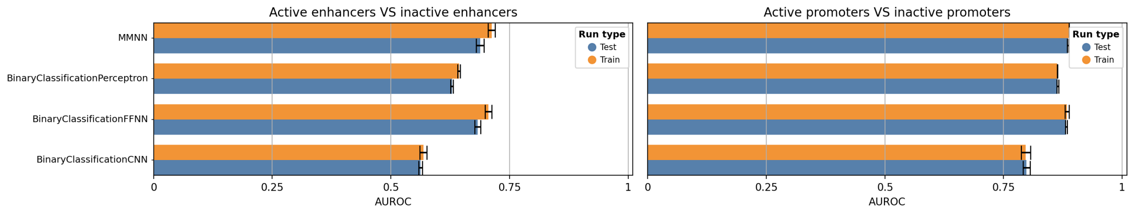


Figure 14: AUROC

5 Future developments

First of all we need to repeat the experiment on a more powerful machine.

We will introduce the features selection that find the relevant features for use in model construction. This technique is useful because it drops the redundant or irrelevant features without removing information. The best features selected algorithms are Random Forest and Boruta. It trains a random forest classifier and applies a feature importance metric to evaluate the importance of each feature. This method will be applied during the holdout split. // Of course if we do this, we can help the model to discriminate better.

Another interesting thing is to compare the models trained with the entire datasets and the models trained with the selected data.

Normally, one would like to discuss the threshold used for when we consider a region active or not, but for this cell line was clear that no improvements could be made. Moreover, this kind of problem is an open problem and the FANTOM5 team recommends the thresholds but it will be interesting to try with different threshold to show if something changes.

Lastly, it would be interesting to compare the multi modal performance between different cell lines or adding new models that handle different data that could be useful to the task.

References

[1] Yuli Liu, Python Machine Learning by example. Second Edition, 2019.

[2] Dr.Randal S.Olson, Python Machine Learning, 2015.

[3] Sheldon M. Ross, Probabilità e statistica per l'ingegneria e la scienza. Terza edizione, 2015.

[4] Rudolf Kruse , Christian Borgelt , Frank Klawonn , Christian Moewes , Matthias Steinbrecher , Pascal Held, Computational Intelligence A Methodological Introduction. 2013.

[5] Wikipedia. https://en.wikipedia.org/wiki/Bioinformatics

[6] Wikipedia. https://en.wikipedia.org/wiki/HEK_293_cells

[7] FANTOM: https://fantom.gsc.riken.jp/5/

[8] Tensorflow. https://www.tensorflow.org

[9] Keras. https://keras.io

[10] Pearson coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[11] Spearman coefficient. https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

[12] Ranking. https://en.wikipedia.org/wiki/Ranking

[13] MICE. https%3A%2F%2Fpapers.phmsociety.org%2Findex.php%2Fphme%2Farticle%2Fdownload%2F1625%2F587&usg=AOvVaw30X6K_nkWB5_gWUanQm5iF

[14] PCA. https://en.wikipedia.org/wiki/Principal_component_analysis

[15] UMAP. https://www.diva-portal.org/smash/get/diva2:1619501/FULLTEXT01.pdf

[16] T-SNE. https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction

[17] Tsne. https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

[18] Promoters vs enhancers. https://it.sawakinome.com/articles/biology/unassigned-3998.html

[19] Holdout. https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f

[20] Accuracy. https://en.wikipedia.org/wiki/Accuracy_and_precision

[21] Precision. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[22] Recall. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[23] Specificity. https://en.wikipedia.org/wiki/Sensitivity_and_specificity

[24] Balance accuracy. https://neptune.ai/blog/balanced-accuracy

[25] AUROC. https://it.wikipedia.org/wiki/Receiver_operating_characteristic

[26] AUPRC. https://it.wikipedia.org/wiki/Receiver_operating_characteristic

[27] F1_score. https://en.wikipedia.org/wiki/F-score

[28] Robust scaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[29] Class imbalance. https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/