Prediction of regulatory regions in a specific cell line using Deep Learning Techniques

1st Giuseppe Bonura
University of Milan
Department of Computer Science
giuseppe.bonura@studenti.unimi.it

January 15, 2022

1 Introduction

In a very short interval, "bioinformatics" has become an extremely active research field. Although it began with sequence comparison (which is a sub branch of the study of the non randomness of DNA sequences), it now encompasses a far wider spread of activity, which truly epitomizes modern scientific research. It's highly interdisciplinary field mainly involving molecular biology and genetics, computer science, mathematics and statistics. It uses computer programs for a variety of applications, including determining gene and protein functions, establishing evolutionary relationships, and predicting the three-dimensional shapes of proteins.
In this project, the focus is mainly on the DNA sequence and the human genome, in fact it's purpose is to analyze regulatory regions in **H1** cell line and determine which regions are active and which are not.

1.1 What are regulatory regions?

Less than 2/3% of the DNA is believed to encode useful information. The remain percentage sometimes called "junk" DNA, has no apparent function. Of the 2/3% of functional DNA, the majority specifies genes—regions of the DNA comprising two parts, a regulatory region or promoter and a coding region. The regulatory region is partly responsible for specifying the conditions under which the gene product is produced, or the degree to which it's produced. The coding region specifies the functional molecular product or products often a protein, but sometimes an RNA (ribonucleic acid). For protein-coding genes, the coding region specifies the amino-acid sequence of the protein. However, the protein is not constructed directly from the DNA. Instead, the DNA is transcribed into an RNA intermediate, called messenger RNA (mRNA), which is then translated into a protein.

In Section 2 we will provide an overview of the models evaluated on the aforementioned tasks, along with the corresponding explored hyper-parameter spaces. In Section 3 we will discuss details pertaining to the execution of the experiments and to the data. Finally, in Section 4 we will present the results of the experiments and their implications.

2 Models

In this work, deep neural networks are build for binary classification useful for predicting the activity of regulatory regions in the H1 cell line. They have been made three different types of deep neural network architectures, and the adjustment of parameters in each neural networks is a very complex and expensive task, for this reason, in order to have a sufficiently truthful model with consistent parameters, it was used KerasTuner, which is an easy-to-use, scalable hyperparameter optimization framework that solves the pain points of hyperparameter search.

- Multilayer perceptron or Feed Forward Neural Network
- Convolutional Neural Network
- Multimodal Neural Network

The supervised approach was used for all models. That supervised learning algorithms are, roughly speaking, learning algorithms that learn to associate some input with some output, given training set of examples of inputs x and outputs y. In many cases the outputs y may be difficult to collect automatically and must be provided by a human "supervisor". Let's see in detail of each model.

2.1 Artificial Neural Network

Artificial neural networks are popular machine learning techniques that simulate the mechanism of learning in biological organisms. The human nervous system contains cells, which are referred to as neurons. [1] Generally artificial neural networks are very simple and consist of only one layer between the network input and output. Similarly, it's architectural simplicity doesn't allow us to approximate very complex mathematical functions. In this regard, what are known as deep neural networks have been introduced, i.e. neural networks that have a greater number of 2 hidden layers between the input and output layers. These models are called **feed-forward** because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no cycles or loops in the network. [2] All neural networks used in this work are classified as a feed-forward neural network.

2.2 Multilayer perceptron

Multilayer neural networks contain multiple computational layers; the additional intermediate layers (between input and output) are referred to as hidden layers because the computations performed are not visible to the user. The specific architecture of multilayer neural networks is referred to as feed-forward networks, because successive layers feed into one another in the forward direction from input to output. The default architecture of feed-forward networks assumes that all nodes in one layer are connected to those of the next layer. Therefore, the architecture of the neural network is almost fully defined, once the number of layers, the number/type of nodes in each layer and type of activation function have been defined. [1]

FFNN Summary			
N.Layers	Layers Type	N.Neurons	Activation
Input	Input	58	-
1,2	Dense	160	ReLu
3,4,5,6	Dense	208	ReLu
Output	Dense	1	Sigmoid

Table 1: FFNN Architecture.

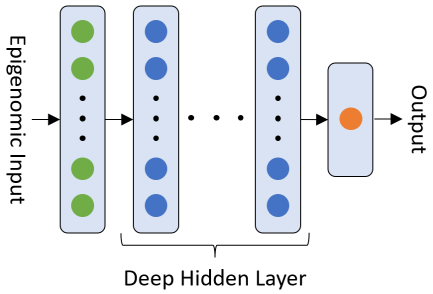

Figure 1: FFNN Network Diagram

The hyperparameter space used for the tuner the FFNN are:

Parameter Name	Hyperparameter Values	Result
num_layers	m=2 M=6 s=1	6
n_neurons0	m=32 M=256 s=32	160
n_neurons1	m=16 M=256 s=16	208
learning_rate	{1e-2, 1e-4}	1e-2

Table 2: FFNN Hyperparameter Space.

2.3 Convolutional Neural Network

Convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data,which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. [2] We will note that within the model of this CNN there will not only be the convolution layer, but there will be more layers that we will briefly describe:

- **Input layer:**[1] This layer represent the entry point of a neural network. Its shape should be equal to the shape of the inputs. In our case the shape is determined by the window size of nucleotides considered.
- **Conv1D:**[2] This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs.
- **Batch Normalization:**[3] This layer applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. The activation function is applied after each layer of batch normalization.
- **Activation Function:**[4] An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. The ReLu function, acronym for rectifier linear unit, was used for this model. It's a very simple function to calculate: it flattens the answer to all negative values to zero, while leaving everything unchanged for values that are equal or greater than zero.
- **Dropout:**[5] The dropout is a technique that allows the deactivation of a percentage of neurons within each hidden layer, thus modifying the topology (in terms of neurons). For each epoch of training, one chooses (randomly) which neurons to keep and which ones to discard and the resulting network is trained. The procedure is then repeated, keeping and discarding different neurons at each epoch: once the network is deemed ready, the original network is taken and the weights coming out of the previously deactivated neurons are adjusted.

[1] https://www.tensorflow.org/api_docs/python/tf/keras/layers/InputLayer
[2] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv1D
[3] https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization
[4] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
[5] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

- **MaxPooling1D:**[6] This layer allows us to reduce the dimensionality of the input by concentrating everything in a single sample and aggregating the weight values using a specific function, in this case the maximum value.
- **GlobalMaxPooling1D:**[7] This layer is similar to MaxPooling1D, but it aggregates an entire feature map in one value.
- **Dense:**[8] Dense layer is a standard fully connected layer.

In this project it has been used providing in input layer a sequences of data in matrix form. Let's see in detail the architecture of the CNN designed.

CNN Summary				
N.Layers	Layers Type	Filters Shape	Kernel Size	Activation
Input	IN	[(256, 4)]	-	-
1	C1D, BN, RL, DR	[(256, 4)]	8	ReLu
2	C1D, BN, RL, DR, MP1D	[(242, 64)]	8	ReLu
3	C1D, BN, RL, DR	[(120, 32)]	2	ReLu
4	C1D, BN, RL, DR, MP1D	[(119, 32)]	2	ReLu
5	C1D, BN, RL, DR	[(59, 32)]	2	ReLu
6	GAP1D, BN, DR	32	-	-
7	DN	80	-	-
Output	OUT	1	-	Sigmoid

Table 3: CNN Architecture.

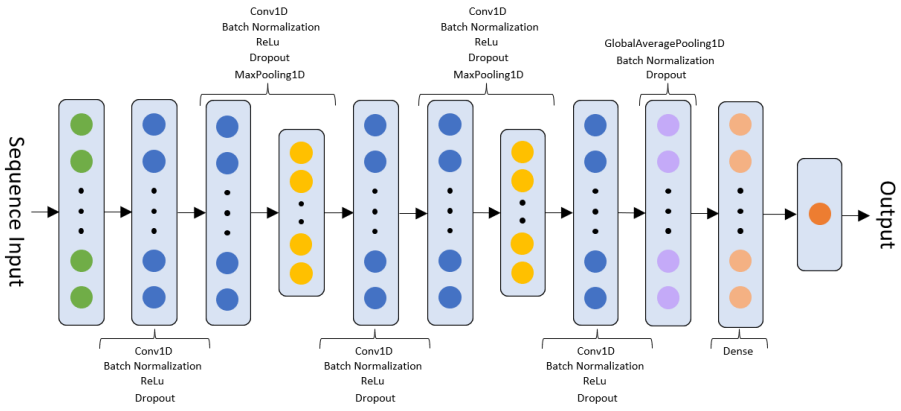

Figure 2: CNN Network Diagram

The hyperparameter space used for the tuner of the CNN are:

[6] https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool1D
[7] https://www.tensorflow.org/api_docs/python/tf/keras/layers/GlobalMaxPool1D
[8] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

Parameter Name	Hyperparameter Values	Result
num_conv_layers	m=2 M=8 s=1	5
kernel_size_0	m=5 M=8 s=1	8
n_filter_0	m=32 M=128 s=32	64
drop_rate_input	m=0 M=0.5 s=1	0.19335805735959433
drop_rate_out	m=0 M=0.5 s=1	0.39499479777555135
kernel_size_1	m=2 M=10 s=1	2
n_filter_1	m=16 M=128 s=16	32
n_filter_output	m=16 M=128 s=16	80
learning_rate	{1e-2, 1e-4}	1e-2

Table 4: CNN Hyperparameter Space.

2.4 Multimodal Neural Network

Multimodal neural networks are precisely networks that give us the possibility of combining two or more deep neural networks, in order to exploit the potential of the latter. In the real world, information comes from heterogeneous sources represented in different ways. Obviously this difference in data representation provides different statistics and properties. The main objective of multimodal networks is therefore to exploit as much as possible the properties and benefits provided by each single neural network. In the project it was therefore planned to bring together the result of the two networks in a junction layer followed by a dense layer. This approach will therefore help us in increasing the degree of prediction of the task by combining epigenomic data, managed by the FFNN network, and genomic data (sequence data), managed by the CNN network.

MMNN Summary				
N.Layers	Network Type	Layers Type	N.Neurons	Activation
Input 1	FFNN	-	-	-
Input 2	CNN	-	-	-
1	-	Concatenate	-	-
2	-	Dense	64	ReLu
Output	-	Dense	1	Sigmoid

Table 5: MMNN Architecture.

The hyperparameter space used for the tuner of the MMNN are show in Table 6. There are also some parameters common to all three networks visible in the Table 7.

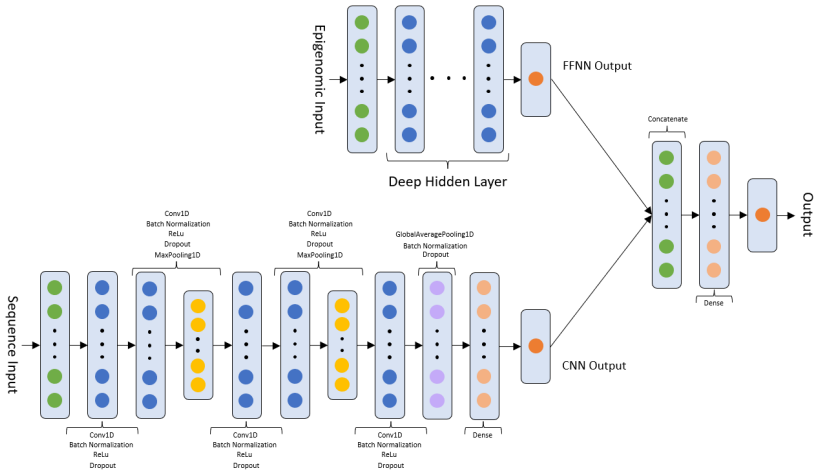

Figure 3: MMNN Network Diagram

Parameter Name	Hyperparameter Values	Result
n_neurons_concat	m=32 M=256 s=32	64
learning_rate	{1e-2, 1e-4}	1e-2

Table 6: MMNN Hyperparameter Space.

Hyperparameter	Values
Loss Function	Binary Crossentropy
Optimizer	Nadam
Epochs	1000
Early Stopping Patience	2
Early Stopping Min Delta	0.001

Table 7: Common Hyperparameter Value.

3 Experimental setup

The experiment was performed on two machine, and consist in the study of two tasks:

- active enhancers vs inactive enhancers (AEvsIE)
- active promoters vs inactive promoters (APvsIP)

in the H1 [3] cell line, on the HG38[9] dataset.

[9] https://genome-euro.ucsc.edu/cgi-bin/hgGateway?redirect=manual&source=genome.ucsc.edu

3.1 The considered tasks

The first and main machine used for performed the task is Colab[10] and the second is a personal notebook with this HW and SW characteristics:

- Intel i5-8250U
- 16 GB of Ram
- Windows 10 Home x64
- Editor Vs-Code[11] 1.63.2
- Python Virtual Env With v3.7.4[12]

The two tasks considered aim to study the activity of enhancer and promoter regions in the H1 cell line. Each task is executed using a single fix dimension of window sizes set to: 256bs (given the limited resources in the development environment, a higher number of windows would have caused the instance to time out and not complete tasks). The chosen window size determines the relative number of nucleotides associated with the enhancer or promoter region considered. All the project phases have been divided into three macro areas. In a first phase, a binary exploration of the data was performed in order to determine the appropriate threshold values for the assigned cell line. In the second stage the hyperparameters were tuned (using Keras Tuner[13]) for each network model in personal computer, then identified and cached the results. The third e last phase is the training of the model, that was performed on Colab with feature selection using boruta with the maximum number set to 30 iterations to avoid execution time too long and sending Colab into time-out without having completed the tasks in development.

3.2 Dataset source

The data used for the experiment are divided into:

1. Epigenomic Data
2. Sequence Data
3. Labels

Epigenomic data was retrieved from ENCODE[14], that is a consortium formed by research groups, whose collaboration aims to create a complete list of functional elements in the human genome, including elements that act at the protein

[10] Free hosted Jupyter notebook service by Google, that providing computing resources including GPUs. https://research.google.com/colaboratory/faq.html
[11] https://code.visualstudio.com/
[12] https://www.python.org/downloads/release/python-374/
[13] https://keras.io/keras_tuner/
[14] https://www.encodeproject.org/

and RNA levels, and regulatory elements that control cells and circumstances in which a gene is active.
The sequence data was recovered from the UCSC Genome Browser[15]. This is a web site where members of the International Human Genome Project consortium give the possibility to publicly access and download the data of the human genome giving the possibility to have a graphic visualization tool.
As for the labels used for training and testing of neural networks, the data used was taken from FANTOM5. FANTOM5 is the evolution of the FANTOM[16] project, that is an international project, born around the 2000s aiming at identifying all functional elements in mammalian genomes.
Fortunately, the hard work to retrieve the epigenomic data from ENCODE, retrieving the labels from FANTOM and querying the terabytes of bigwig files has already been done using an HPC. Some python packages have been made available which are useful for speeding up the execution of the project.

- epigenomic_data[17]: Is a package developed by the PhD students of Anacleto Lab[18]. This package is used to download epigenomic data related to labels, already pre-processed for specific window size, from ENCODE and FANTOM5.
- ucsc_genomes_downloader[19]: Is a package useful to retrieve the human genome HG38 from the UCSC Genome Browser.
- keras_bed_sequence[20]: Is a package useful to select and one-hot encode the specific sequences related to the regions and the window sizes under study.
- keras_mixed_sequence[21]: Is a package useful to organize and feed the data during training and testing phase of deep neural network models.

3.2.1 Binary Exploration

Before to retrieve the data and convert the real values of the labels to a binary values, was necessary to make a binary exploration for determine the most appropriate value for threshold. The values reported in the data retrieved from FANTOM5 are in TPM (Transcripts Per Kilobase Million). Without going in deep, the value of the TPM, is the activation value of a given region that is measured in a dynamic system that evolves over time, and at the moment of the measurement it's practically killing the cell, so it's difficult to understand how this value was either evolving, meaning it was going up, so the region of was turning on, or it was going down, meaning the region was turning off. Therefore

[15] https://genome-euro.ucsc.edu/index.html
[16] https://fantom.gsc.riken.jp/
[17] https://github.com/AnacletoLAB/epigenomic_dataset
[18] https://anacletolab.di.unimi.it/
[19] https://github.com/LucaCappelletti94/ucsc_genomes_downloader
[20] https://github.com/LucaCappelletti94/keras_bed_sequence
[21] https://github.com/LucaCappelletti94/keras_mixed_sequence

Index	Task	Region	Cell	Tr	Sample	Pos	Neg	Mean
0	AE-IE	Enhancers	H1	0	63285	3038	60247	0.048
1	AE-IE	Enhancers	H1	0.25	63285	3038	60247	0.048
2	AE-IE	Enhancers	H1	0.5	63285	3038	60247	0.048
3	AE-IE	Enhancers	H1	0.75	63285	715	62570	0.011
4	AE-IE	Enhancers	H1	1	63285	715	62570	0.011
5	AP-IP	Promoters	H1	0	99881	35191	64690	0.352
6	AP-IP	Promoters	H1	0.25	99881	35191	64690	0.352
7	AP-IP	Promoters	H1	0.5	99881	35191	64690	0.352
8	AP-IP	Promoters	H1	0.75	99881	35191	64690	0.352
9	AP-IP	Promoters	H1	1	99881	25559	74322	0.255

Table 8: Imbalance between positives and negatives samples as the threshold varies.

the values obtained are a screen shoot at a given moment for a given cell, so using one threshold rather than another is very complicated. For this reason the exploration on various threshold values will help us to determine it having particular care also on the imbalance some data.
In a jupyter called Data_Analysis[22] is reported the binary exploration based on the entire dataset and not on training data. Meanwhile, in Table 8 is shows the result on the measurement of positive and negative sample as the threshold varies. There is evidence of how when the threshold changes there is a variation between positive and negative values. In general, it's possible to note that the number of positive and negative samples as regards the promoters doesn't generate an imbalance of the classes, this due to the fact that during the transcription phase these DNA sequences are located at the beginning of the gene and the latter it also contains the nucleotide from which RNA synthesis begins. On the other hand, it happens differently for enhancers that being located hundreds or thousands of nucleotides further ahead are difficult to intercept, and in fact, also in a not very marked way, we have a slight imbalance of about 5% of the positive values compared to the negative ones.

Initially they were established as threshold values 0, 0.25 and 0.5, and analyzing the mean for both regions no variation was visible (Index 0, 1 and 2 for enhancers and index 5,6,7 and 8 for promoters in Table 8). To try to identify the point in which there was a variation (useful to determine the activation of the region), were downloaded float epigenomic data in order to perform a visual exploration using a histogram between 0 and 1.

By displaying the histogram shown in the Figure 5, and the red boxes of row 3 in Table 8, it's possible identify graphically and numerically that a threshold < 0.75 can be used for the task of the active enhancers vs inactive enhancers.

[22] https://github.com/bonurag/bioinformatics_project/blob/master/Data_Analysis.ipynb

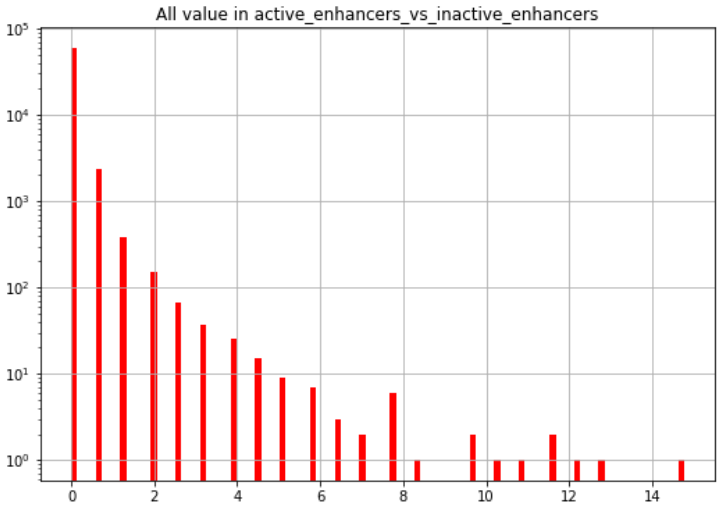

Figure 4: All TPM value AE-IE task.

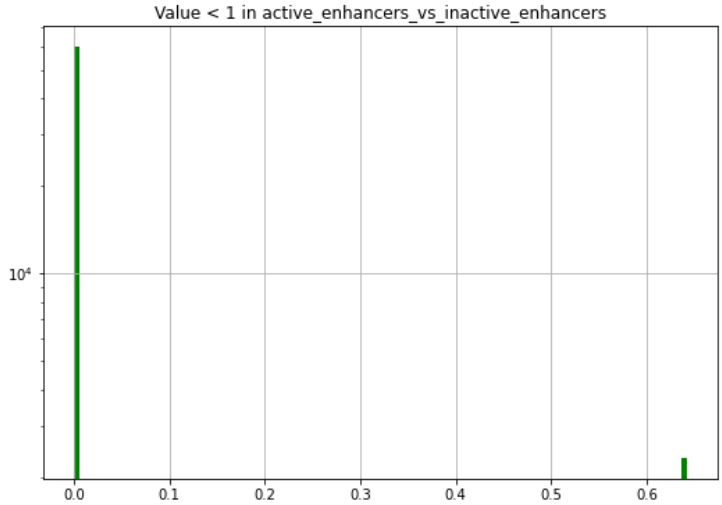

Figure 5: AE-IE TPM value in range 0-1.

Any value above < 0.75 would only increase the offset between positive and negative values. For AE-IE task, the value 0 identified by the green row in the Table 8 has been chosen as the threshold.

By displaying the histogram shown in the Figure 7, and the red boxes of row 9 in Table 8, it's possible identify graphically and numerically that the reason for the lack variation of the average with the first threshold values chosen (0, 0.25,

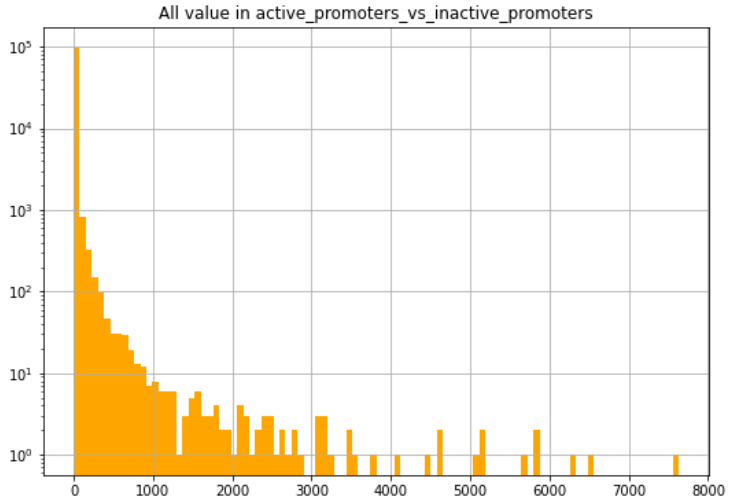

Figure 6: All TPM value AP-IP task.

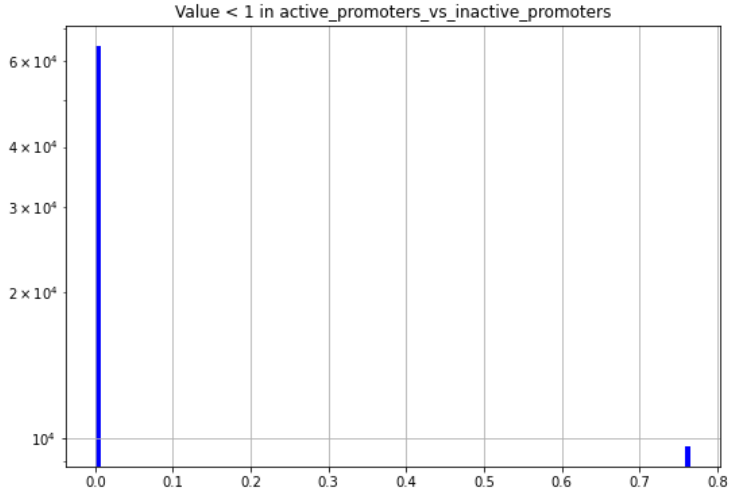

Figure 7: AP-IP TPM value in range 0-1.

0.5). In fact, it emerges from the graphic representation that for values greater than 0 there is a real gap. For AP-IP task, the value 1 identified by the blue row in the Table 8 has been chosen as the threshold.

In some real world datasets, and especially biological and medical datasets, we often encounter a strong imbalance between classes. The more unbalanced the classes, the harder it becomes for us to create good models to predict them. It's therefore important to know immediately if the classes are more or less

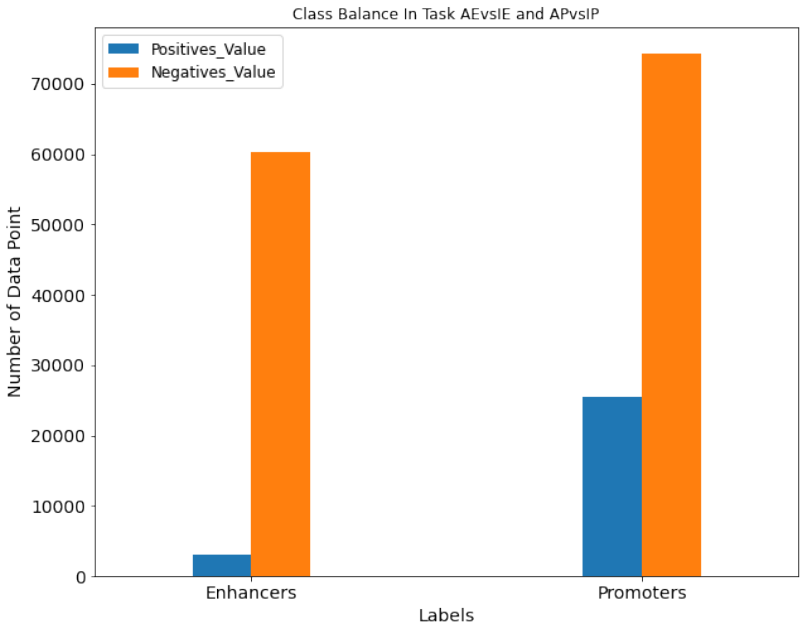

Figure 8: Class Balance Between AE-IE and AP-IP Task.

Region	Positives Value	Negatives Value
Enhancers	3038	60247
Promoters	25559	74322

Table 9: Numerical data between positives and negatives samples.

balanced or we are in a highly unbalanced scenario. For this reason after this binary exploration, useful for identifying the thresholds, we can see in Figure 8 the unbalancing of the classes for the two tasks.

3.3 Data pre-processing

To use sequence data, categorical values need to be converted to a BED[23] format. This is done through one-hot encoding, using the keras_bed_sequence python package, which allows us to obtain a matrix representation starting from the sequence of the human genome (Example AAATGG...TCA...TCCAT). Finally, after data retrieval and generate sequence data, we can move on data elaboration section.

[23] https://www.ensembl.org/info/website/upload/bed.html

3.4 Data Elaboration

Data preprocessing is an important step in machine learning projects. Specifically it refer to manipulation or dropping of data before it's used in order to ensure or enhance performance. The steps performed are:

1. Rate between features and samples
2. Nan detection and imputation
3. Constant detection
4. Drop Constant features
5. Feature Scaling
6. Feature Correlations and Distributions
7. Feature Selection

3.4.1 Rate between features and samples

In some datasets, there are more features than samples. This can introduce problems such as a particularly strong overfitting of the model. For this reason it's good practice to check if the datasets have a rate between features and samples greater than one. In a study case that has been analyzed, and in the datasets in use, it can be seen from the Table that the rate is well above 1.

Region	Total Sample	Features	Features Rate
Enhancers	63285	58	1091.12
Promoters	99881	58	1722.09

Table 10: Rate Between Features and Samples.

3.4.2 Nan detection and imputation

From a theoretical and practical point of view, the presence of NaN (Not a Number) values or missing values within a dataset is a problem. Theoretically these values cannot be interpreted, while from a practical point of view their presence can cause errors during the execution of the algorithms, or provide incorrect data when evaluating some metrics. In order to deal with them, a k-Nearest Neighbors (k is a parameter that can be varied) Imputer[24] is used to impute the missing values. The latter used by default, a euclidean distance metric that supports missing values is used to find the nearest neighbors. Each sample's missing values are imputed using the mean value from nearest 5 (Five is a default value) neighbors found in the training set.

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

3.4.3 Constant Detection

We can also insert within this section the task of detecting constant features inside a dataset. Surely their presence doesn't involve important problems like those that are instead introduced by missing values, but their search and elimination is also part of the data cleaning. In general, it's not very common to find constant features inside the datasets, and most of the time it's a symptom that indicates that something is wrong in the data itself, but by not providing added value to each sample they can be eliminated. In the dataset in use for this project and for the specific cell line, no constant features to be eliminated have been identified.

3.5 Feature Scaling

Feature scaling, or normalization, is a method used to achieve comparable features across their range of values. In general, in this type of situations where the features aren't normalized, the network and the models in general should be able to adapt by changing the weights of the network, but what we want is to obtain weights as optimized as possible to better manage the label in exit and not to self-normalize the weights of the network, making learning more difficult and longer, among other things. In linear classifiers, for example, if a characteristic takes on high values, it's importance is greater in forecasting than another characteristic with lower values. We want to avoid this and other common problems with unsized features. To obviate the above, a Robust Scaler[25] is used, which as the name suggests is a robust tool, especially towards the outliers (which are the ones we want to eliminate) which allows us to normalize the data by subtracting the median and dividing by the standard deviation within an interquartile[26] range between 0.25 and 0.75. Therefore, given a Gaussian distribution, we will remove the initial e part and the tail of the distribution, in order to avoid bringing values that tend towards infinity into the dataset. It should also be remembered that this normalization problem has a very high impact in networks, or in any case in models that include layers where the sum of the various features is performed. Unlike in models such as decision trees, the problem doesn't exist.

3.6 Data Correlations and Distributions

3.6.1 Feature Correlation

Another parameter that we can evaluate is that of correlation. The latter can be applied with respect to output labels or between features. What are we going to determine with the correlation? Correlation is a specific analysis that measures the strength of the association between two variables and the direction of the relationship. In terms of the strength of the relationship, the value of the

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

[26] https://it.wikipedia.org/wiki/Scarto_interquartile

correlation coefficient varies between +1 and -1. When it's close to 1 it means that there is a strong positive correlation whereas when the coefficient is close to -1, it means that there is a strong negative correlation. Finally, coefficients close to 0 mean that there is non linear correlation, but this does not mean that there is no correlation. For this reason, in addition to the application of the Pearson[27] and Spearman[28] indices, the MIC[29] (maximum coefficient of information) which is a measure based on non-parametric statistics was used. It helps us determine the strength of the linear or non-linear association between two variables X and Y. Given a certain correlation threshold, we will then determine what are the features not correlated with the output labels, and if present they can also be eliminated.

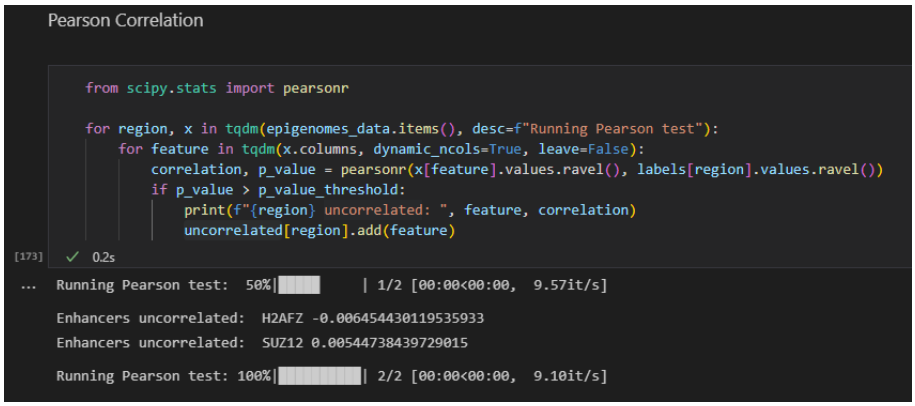

Figure 9: Pearson Correlation.

Once the correlation steps with the output have been completed, it can be seen on Figure 11 that the features identified by the **H2AFZ**, **UZ12** labels are not related to the output labels and therefore can be eliminated. Obviously their presence does not involve problems during the training of the models, but when we have a very high number of features, their elimination can greatly facilitate the training phase.
The next step was to identify and drop features that have extremely high Pearson correlation values (greater than 0.90 or even than 0.99) with each other. These features often don't add meaningful information when removed. If we have to choose to remove a feature, we will remove the one with lower entropy. In information theory, the entropy[30] of a message source is the average information contained in each sent message. The information contained in a message is the greater the less likely it was. A foregone message, which has a high probability of being emitted by the source, contains little information, while an

[27] https://it.wikipedia.org/wiki/Indice_di_correlazione_di_Pearson
[28] https://it.wikipedia.org/wiki/Coefficiente_di_correlazione_per_ranghi_di_Spearman
[29] https://en.wikipedia.org/wiki/Maximal_information_coefficient
[30] https://it.wikipedia.org/wiki/Entropia_(teoria_dell%27informazione)

Figure 10: Spearman Correlation.

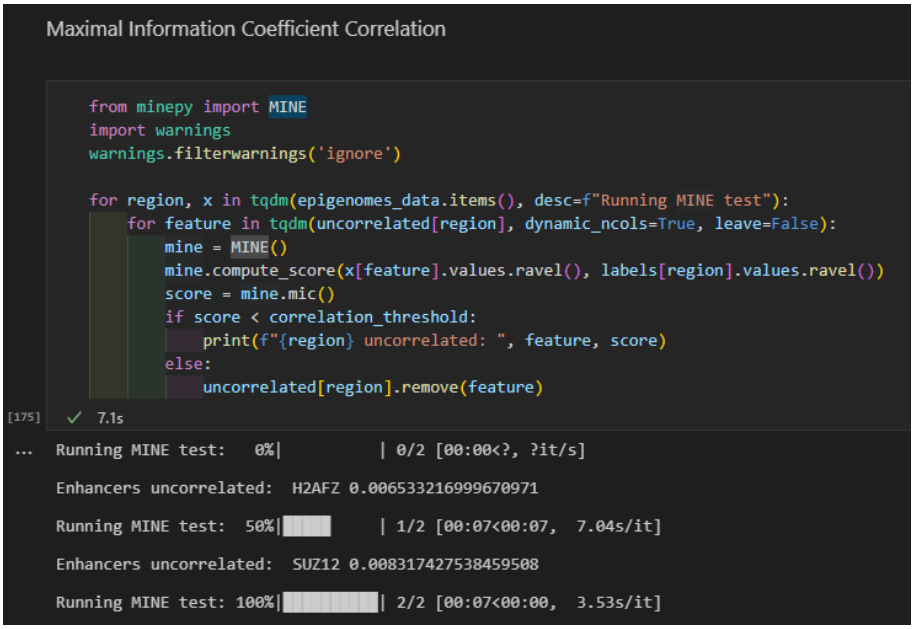

Figure 11: Maximal Information Coefficient Correlation.

unexpected, unlikely message contains a large amount of information.
We show the top five (where possible) most correlated features in promoters and enhancers in Figure 12 and Figure 13 respectively. Blue square they indicate the samples of the inactive regions, while the orange diamonds indicate the active regions. Since no features highly correlated with statistical significance are found, we will keep all the features.
This plot allows us to graphically identify the most correlated features (green circles), which are those that are positioned on the diagonal or antidiagonal, vice

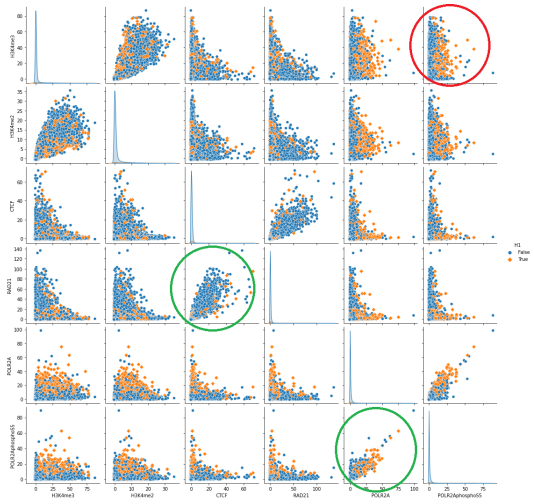

Figure 12: Most Correlated Features In Enhancers Region.

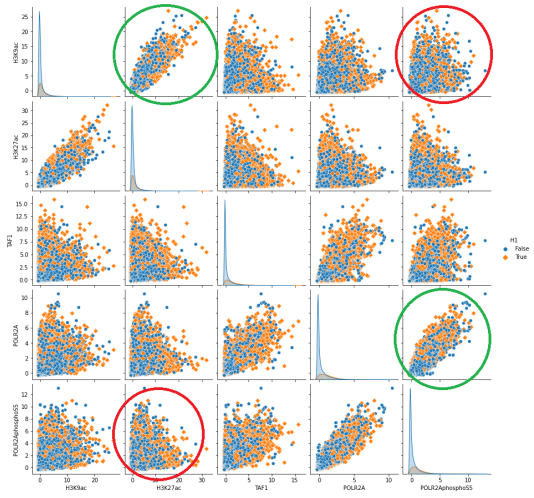

Figure 13: Most Correlated Features In Promoters Region.

versa all the others, that in general the samples tend to accumulate towards one of the two axes, are considered unrelated (red circles). In the main diagonal is possible to see the distribution of a single feature. In general for the inactive value there are a lot of sample most close to zero, while for the active regions we have a small Gaussian distribution.

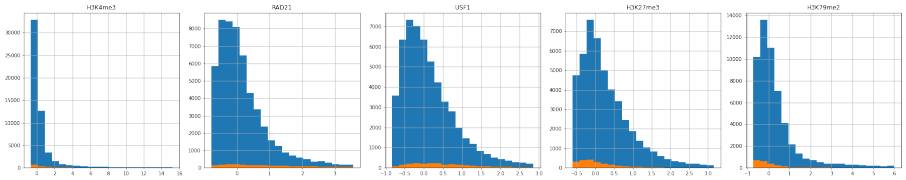

Figure 14: Top 5 Features Distribution In Enhancers Region.

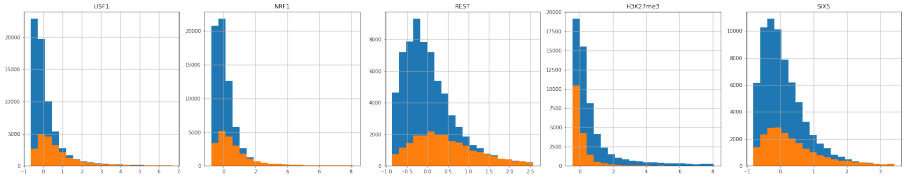

Figure 15: Top 5 Features Distribution In Promoters Region.

3.6.2 Feature Distributions

Also, we want to show the distributions of the epigenomic features. Since there are 58 features, and we cannot show all the distributions, we select the top 5 most different features (5 for promoters and 5 for enhancers), using the pairwise euclidean distance between the features, and represent their distributions using histograms on a log scale. Figure 14 shows the top 5 most different features for promoters and Figure 15 shows the top 5 most different features for enhancers. Analyzing the distribution of features is a way to also understand how separable the classes in question are. Furthermore, given the alignment between them, it's possible to determine which features are particularly informative and relevant in the training phase and which are not.

3.7 Feature Selection

Feature selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Both correlation with output and correlation between features can be used for the selection of features, but there are numerous techniques for running feature selection. In this project we will use Boruta[31] using the python package Boruta_py[32]. Boruta is based on statistical tests performed on the performances calculated through a random forest classifier. The random forest gives the importance of each characteristic, replacing it with a random set of values. Comparing the importance of the feature between the feature itself and the set of values that replaced it, if the result is the same it means that the

[31] https://www.jstatsoft.org/article/view/v036i11
[32] https://github.com/scikit-learn-contrib/boruta_py

feature in question is not useful at all. It's a very expensive technique, which is why during the realization of the project it was performed with a maximum number of iterations equal to 30.

3.8 Data Visualization

In this section, we explore two techniques used for data transformation and visualization. Specifically, most of the datasets used in bioinformatics, but in machine learning in general, have a number of features that far exceed the 3 dimensions of space, with which it's possible to perform the projection inside a cube, otherwise we would fall back into hyperspace. For this reason these techniques allow us to reduce the dimensionality while preserving the maximum amount of information.

- PCA: It's a technique aimed at deriving, starting from a set of correlated numerical variables, a smaller set of "artificial" orthogonal variables. The reduced set of linear orthogonal projections (known as "principal components" or "principal components", "PC") is obtained by linearly combining the original variables in an appropriate manner.

- t-SNE: t-SNE is a statistical method for visualizing high-dimensional data by giving each data point a location in a two or three dimensional map. It's a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. [4]

In Figure 17, 18, 19, 20, 21, 22 shows the results of t-SNE applied to epigenomic and sequence data in both promoter and enhancer regions with different value of perplexity[33], while in the Figure 16 the representation of the same data performed with the PCA algorithm with a number of components equal to 2.

3.9 Holdouts

In order to evaluate the models we executed 10 holdouts, each of which, after shuffling the data, split the data sets in a training set that containing 80% of the data and in a test set containing the remaining 20% of the data. In doing so, we preserved the stratification of the classes in both the training and test sets. This subdivision approach into train data and test data is necessary to verify the generalization capacity of the trained model (train data) on unseen data (train data).

[33] https://en.wikipedia.org/wiki/Perplexity

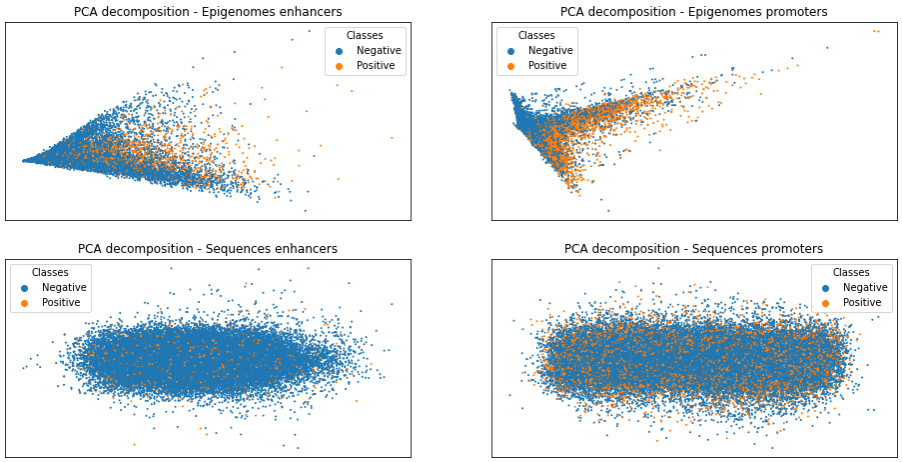

Figure 16: PCA Applied To Epigenomic And Sequence Data.

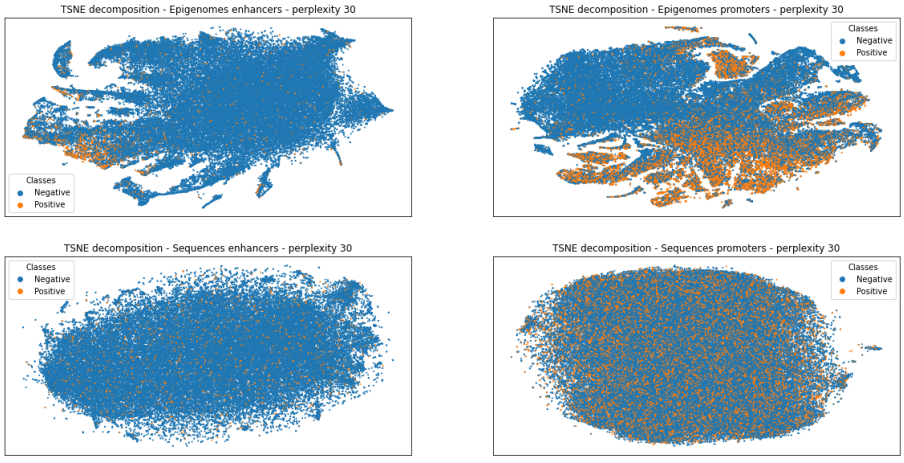

Figure 17: t-SNE With 30 Perplexity

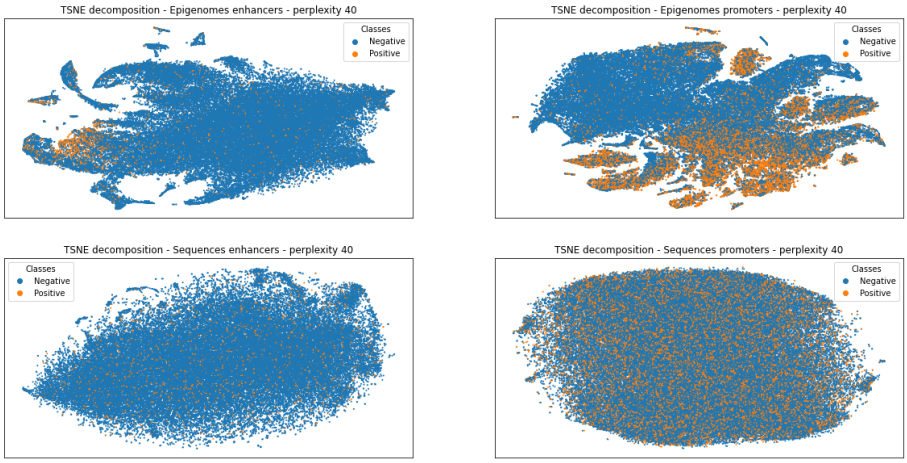

Figure 18: t-SNE With 40 Perplexity

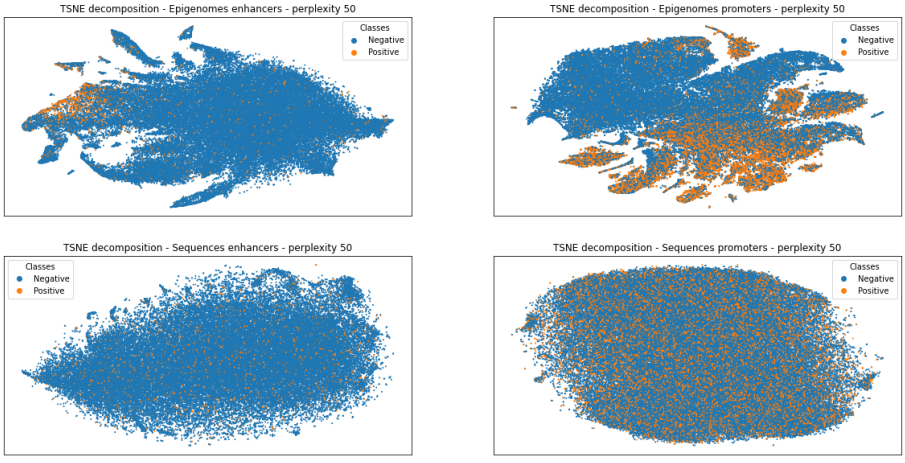

Figure 19: t-SNE With 50 Perplexity

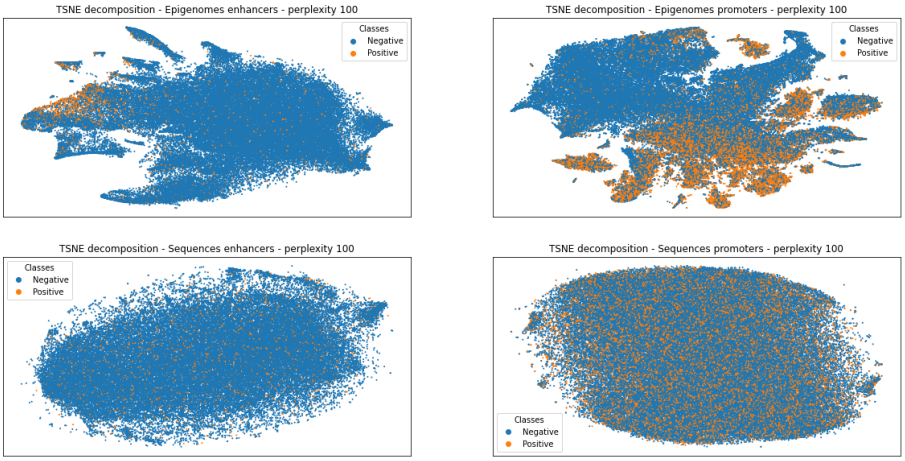

Figure 20: t-SNE With 100 Perplexity

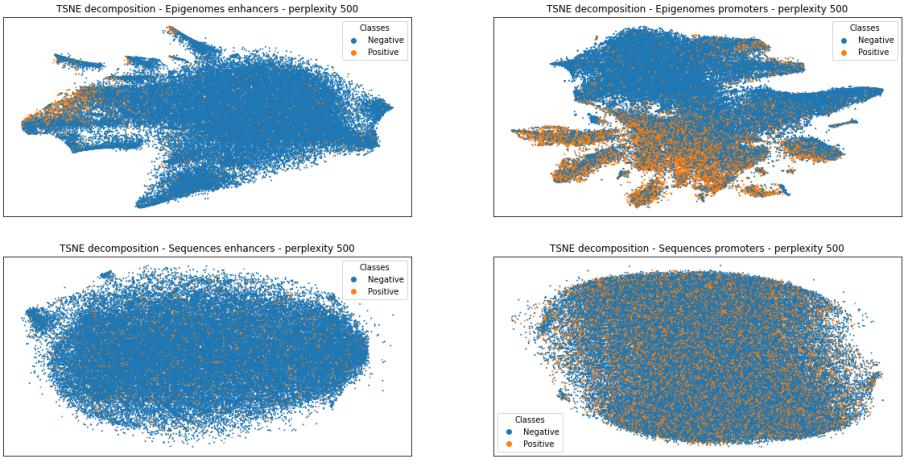

Figure 21: t-SNE With 500 Perplexity

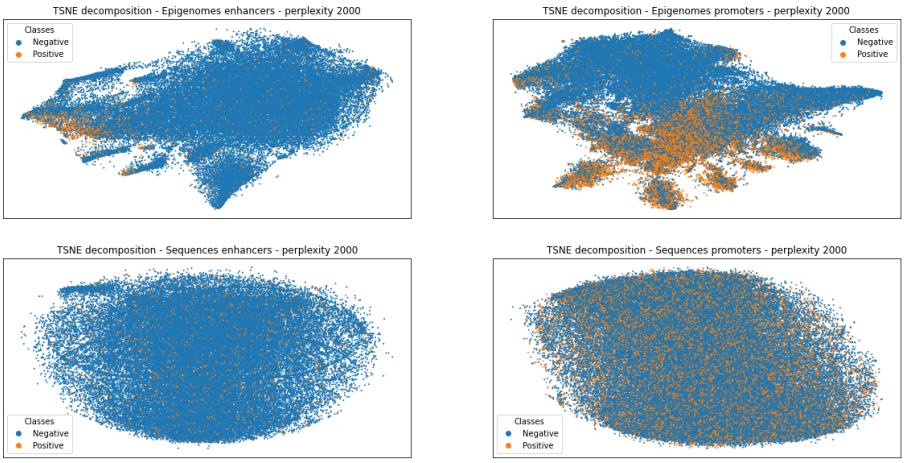

Figure 22: t-SNE With 2000 Perplexity

3.10 Metrics

Each model was evaluated according to mainly three metrics: area under the precision recall curve (AUPRC), area under the receiver operating characteristic curve (AUROC) and Accuracy (Loss plots are put in it only for give context in some point of results). Given the imbalance of the classes, the mean AUPRC over the test sets was considered the most informative predictor of the quality of the model.
To gauge the existence of a statistical difference between models under comparison, we used the Wilcoxon signed-rank test. The Wilcoxon signed-rank [5] test tests the null hypothesis that two related paired samples come from the same distribution. In particular, it tests whether the distribution of the differences of two vector x - y is symmetric about zero for same sample. We considered a p-value lower than 0.01 (one error every 100) as a sensible threshold to claim that the compared models were statistically different.

4 Results

1. Once all the phases of the project were completed, we are created some graphs for help to understand the results obtained in a more concrete way. Specifically, it's possible to see a series of barplots based essentially on the metrics indicated in the section 3.10. For each task the results were grouped by model and if they had more or less used the feature selection. All the graphs referring to this section are those shown in the Figures 23, 24, 25, 26, 27, 28, 29, 30.

2. In addition to the use of graphs only, an appropriate function based on the Wilcoxon test was used through which it's possible to determine in a

formal way if one model was better than the other respect to some metrics. The results obtained are shown in the table 11.

Task	Model	Metrics	P_Value	Result
Enhancers	BinaryClassificationFFNNV1	AUPRC	0.845	ST
Enhancers	BinaryClassificationFFNNV1	AUROC	0.921	ST
Enhancers	BinaryClassificationFFNNV1	Accuracy	0.275	ST
Enhancers	BinaryClassificationCNNV1	AUPRC	0.556	ST
Enhancers	BinaryClassificationCNNV1	AUROC	0.625	ST
Enhancers	BinaryClassificationCNNV1	Accuracy	1.0	ST
Enhancers	MMNNV1	AUPRC	0.921	ST
Enhancers	MMNNV1	AUROC	0.695	ST
Enhancers	MMNNV1	Accuracy	0.838	ST
Enhancers	BoostedMMNNV1	AUPRC	0.492	ST
Enhancers	BoostedMMNNV1	AUROC	0.769	ST
Enhancers	BoostedMMNNV1	Accuracy	0.695	ST
Promoters	BinaryClassificationFFNNV1	AUPRC	0.193	ST
Promoters	BinaryClassificationFFNNV1	AUROC	0.019	ST
Promoters	BinaryClassificationFFNNV1	Accuracy	0.037	ST
Promoters	BinaryClassificationCNNV1	AUPRC	0.130	ST
Promoters	BinaryClassificationCNNV1	AUROC	0.130	ST
Promoters	BinaryClassificationCNNV1	Accuracy	0.556	ST
Promoters	MMNNV1	AUPRC	0.431	ST
Promoters	MMNNV1	AUROC	0.431	ST
Promoters	MMNNV1	Accuracy	0.232	ST
Promoters	BoostedMMNNV1	AUPRC	0.921	ST
Promoters	BoostedMMNNV1	AUROC	0.845	ST
Promoters	BoostedMMNNV1	Accuracy	0.232	ST

Table 11: Results For Wilcoxon Test Applied On Model With And Without Feature Selection.

3. Further graphical representation was used to point out once again, how the use of the boruta algorithm for this specific cell line, not only with a maximum number of iterations set to 30 it was not able to eliminate features, but that for the purpose of identifying an optimal value of AUPRC it was not supportive but added complexity and lengthened the execution times.

5 Conclusion

A conclusion, by analyzing the plots obtained we can draw conclusions, referring in a precise way to the results shown for the AUPRC metric, which turned out

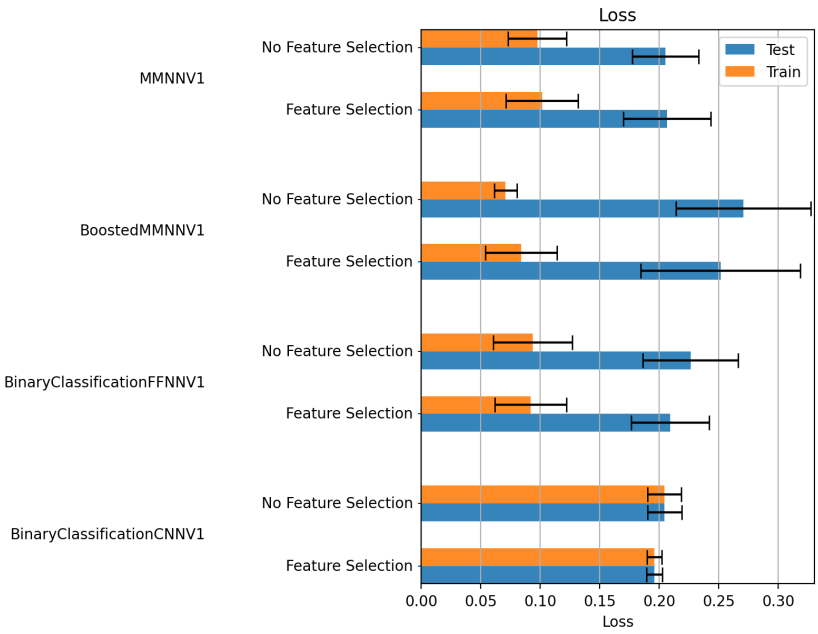

Figure 23: Loss Enhancers

to be the most reliable together with the AUROC, when dealing with problems with unbalanced classes in which you are very interested in finding the positive examples. Surely among all the models, the one that performed best for both tasks was the Multimodal Neural Network, which as described combines the properties of the models making the most of the peculiarities of each of them. It's also visible from these graphs that the help of the feature selection did not affect the performance trend, and it's also necessary to make a note, as regards the CNN model, the differentiation between feature selection and no features selection it's only for the convenience of using the tool for the creation of plotbars, in fact with this model no feature selection has been made, since it uses the sequence data as the first thing, and as a second boruta it has been applied only to epigenomic data, also because applying it to the sequence data is a very complex task that was analyzed in a study [6] conducted by the research group of Anacleto Lab [34]. Probably an easy way to conduct a feature selection on the sequence data and then get different results would have been to vary the windows size, but system limitations and the high amount of task execution time was not explored. It's also possible (as initially expected) to see how the imbalance of the samples related to the enhancers turned out to be crucial for the model training phase, the few samples make learning difficult and tend to overfit the model as can be seen by comparing Figure 26 and Figure 24 where we

[34] https://anacletolab.di.unimi.it/

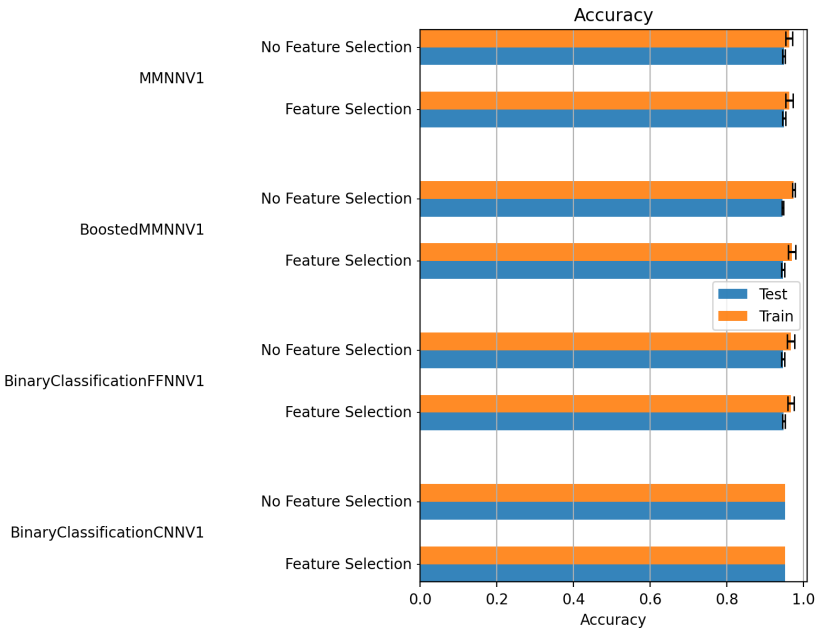

Figure 24: Accuracy Enhancers

have a very high accuracy and also a fairly low loss Figure 23 but a practically almost zero AUPRC. On the other hand, the task relating to promoters is totally different, with a certainly higher number of samples, and a tolerable imbalance between positives and negatives that allows us, with the same metrics 30, 28, 27 viewed, to assert that the models have provided a much better response. Obviously there are some models, such as convolutional neural networks that are very complex to optimize and that in general are used in tasks that generally involve images. For educational and dissemination purposes, we also add to the document two screens relating to the histories of the best AUPRC for the two tasks.

6 Future Developments

We can certainly consider the task analyzed in this problem open and ready to be improved by introducing some corrections and insights. First of all we can try to use meta models based on performing Bayesian algorithms to try to find the best possible model to approach the data in use. Vary the type of models used by, for example, combining in a multimodal network other types of models that are not based on networks but for example on binary trees, such as random forest to see how it behaves with such a complex task. Another thing that we can evaluate while using the convolutional neural network is the

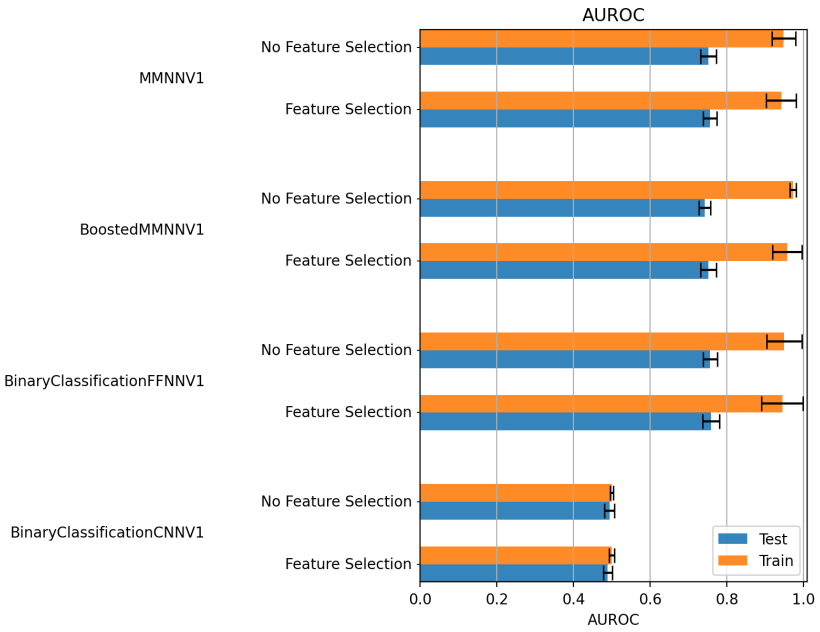

Figure 25: AUROC Enhancers

combination of the sequence data of the chromosome name with the aid of an embedding layer[35]. It's possible to use techniques such as those introduced by the HyperSMURF [7] algorithm to try to obtain better results in the case of unbalanced classes such as in the enhancers task.

[35] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding

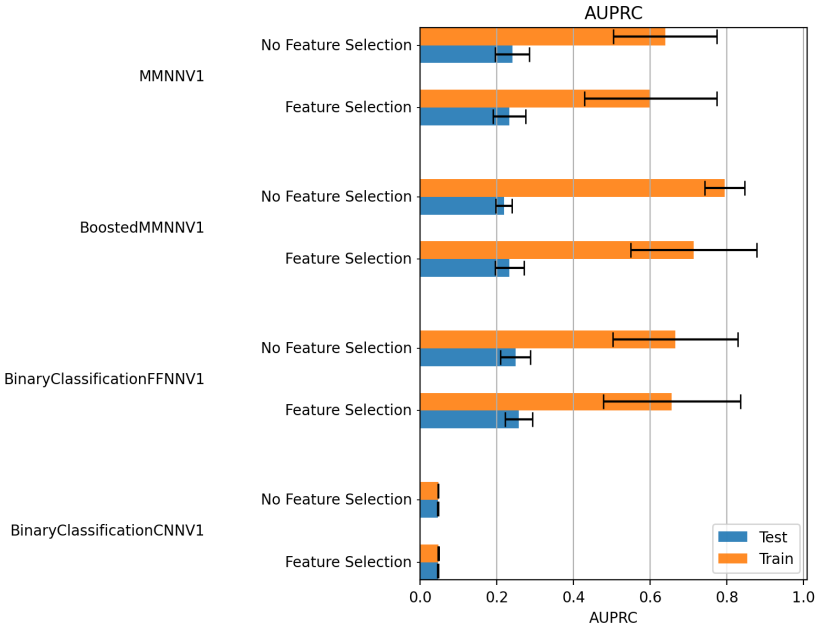

Figure 26: AUPRC Enhancers

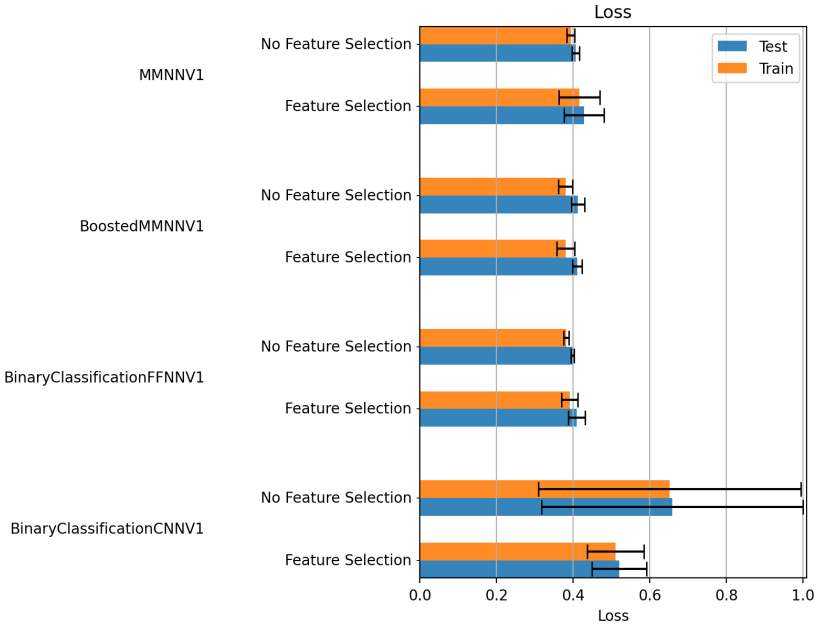

Figure 27: Loss Promoters

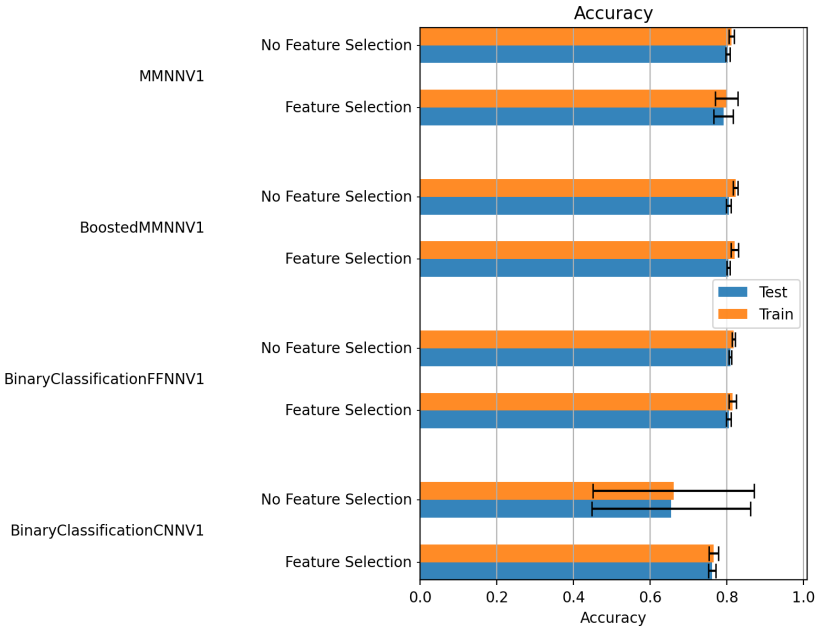

Figure 28: Accuracy Promoters

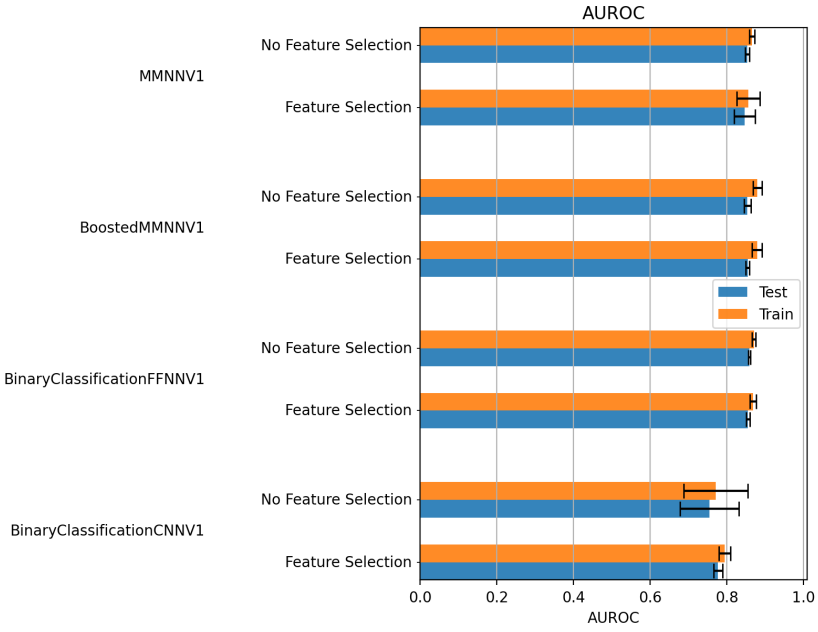

Figure 29: AUROC Promoters

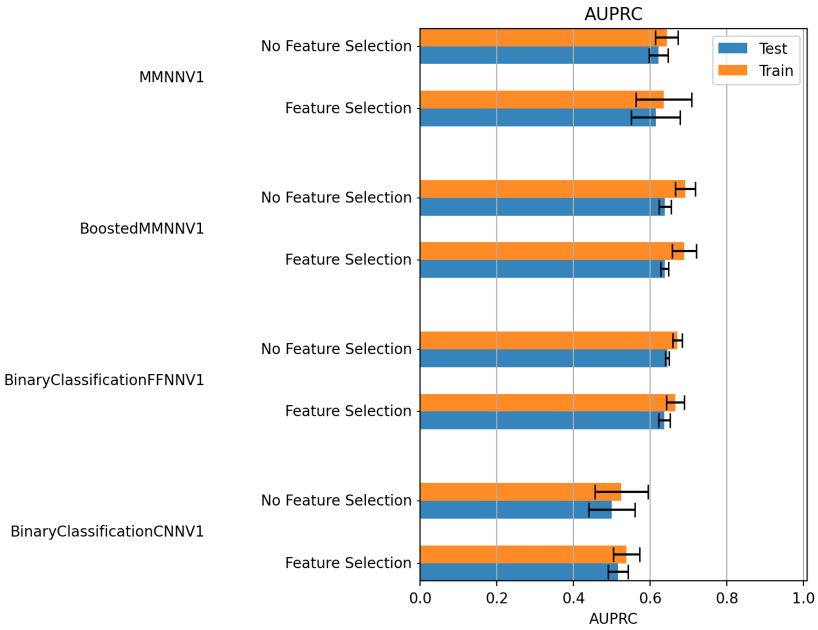

Figure 30: AUPRC Promoters

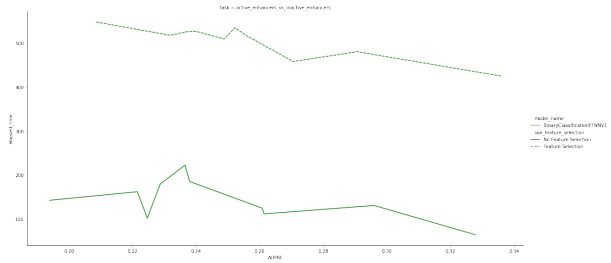

Figure 31: Boruta Elapsed Time VS AUPRC On FFNN In Enhancers Task

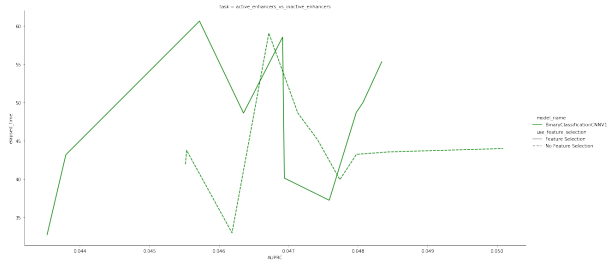

Figure 32: Boruta Elapsed Time VS AUPRC On CNN In Enhancers Task

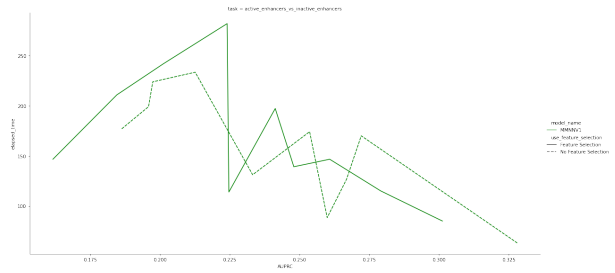

Figure 33: Boruta Elapsed Time VS AUPRC On MMNN In Enhancers Task

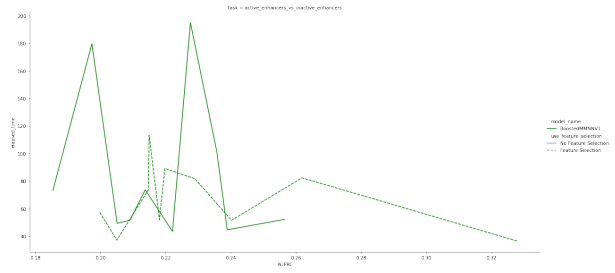

Figure 34: Boruta Elapsed Time VS AUPRC On Boosted MMNN In Enhancers Task

Figure 35: AUPRC Promoters

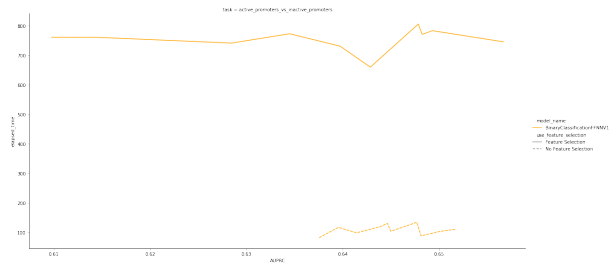

Figure 36: Boruta Elapsed Time VS AUPRC On FFNN In Promoters Task

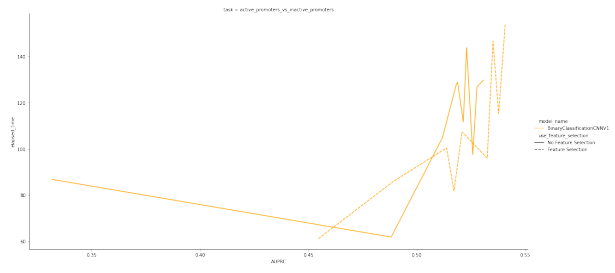

Figure 37: Boruta Elapsed Time VS AUPRC On CNN In Promoters Task

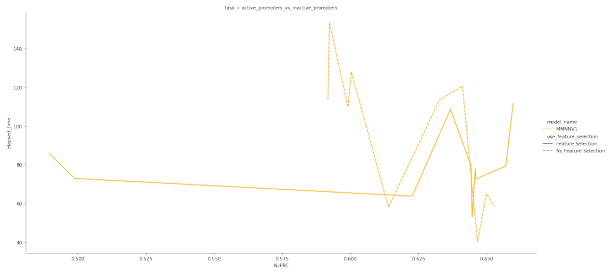

Figure 38: Boruta Elapsed Time VS AUPRC On MMNN In Promoters Task

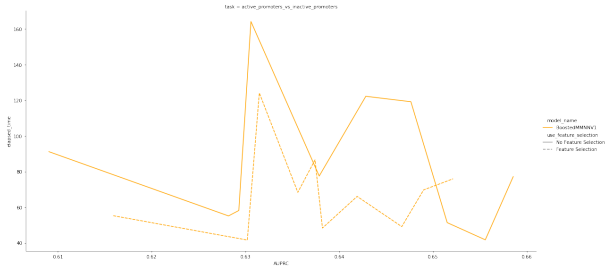

Figure 39: Boruta Elapsed Time VS AUPRC On Boosted MMNN In Promoters Task

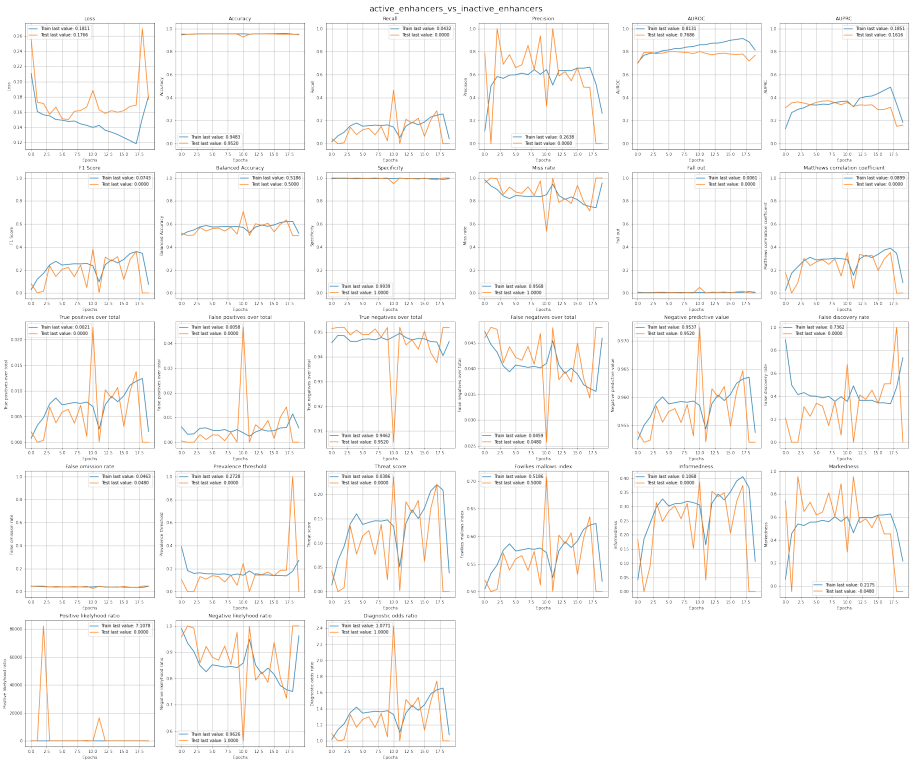

Figure 40: Training Histories Relative At Best AUPRC Value In Enhancers

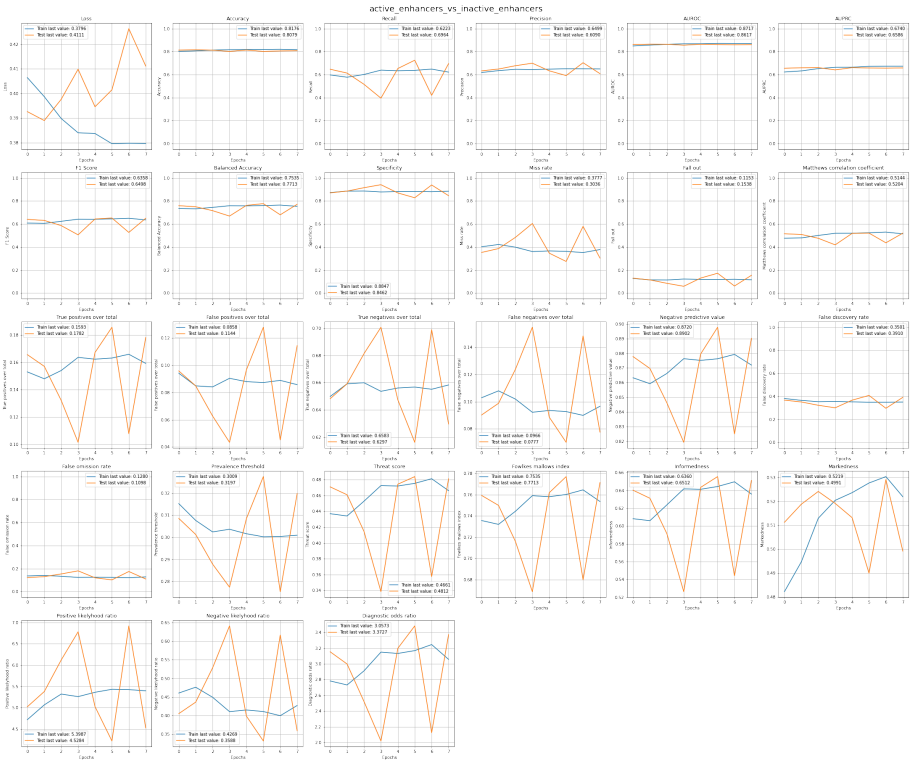

Figure 41: Training Histories Relative At Best AUPRC Value In Promoters

Acronyms

AE-IE active enhancers vs inactive enhancers. 11, 13

AP-IP active promoters vs inactive promoters. 12, 13

BN Batch Normalization Layer. 5

bs Base Pair. 8

C1D 1D Convolution Layer. 5

CNN Convolutional Neural Network. 4–6

DN Dense Layer. 5

DR Dropout Layer. 5

ENCODE Encyclopedia of DNA Elements. 8, 9

FANTOM Functional Annotation Of the Mammalian genome. 9

FFNN Feed Forward Neural Networks. 3, 6

GAP1D GlobalAveragePooling1D Layer. 5

HPC High Performance Computing. 9

IN Input Layer. 5

M Max Value. 3, 6, 7

m Min Value. 3, 6, 7

MIC Maximum Coefficient Of Information. 16

MMNN Multi Modal Neural Network. 6, 7

MP1D MaxPool1D Layer. 5

OUT Output Layer. 5

RL Relu Activation. 5

s Step. 3, 6, 7

ST Statistically Indistinguishable. 25

TPM Tags Per Million. 9

UCSC University of California, Santa Cruz. 9

References

[1] C. C. Aggarwal, *An Introduction to Neural Networks*. Springer International Publishing, 2018.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[3] S. P. Hergeth and R. Schneider, "The h1 linker histones: multifunctional proteins beyond the nucleosomal core particle," *EMBO reports*, vol. 16, no. 11, pp. 1439–1453, 2015.

[4] Wikipedia contributors, "T-distributed stochastic neighbor embedding — Wikipedia, the free encyclopedia," 2021. [Online; accessed 6-January-2022].

[5] "wilcoxon." https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html.

[6] L. Cappelletti, T. Fontana, G. W. Di Donato, L. Di Tucci, E. Casiraghi, and G. Valentini, "Complex data imputation by auto-encoders and convolutional neural networks—a case study on genome gap-filling," *Computers*, vol. 9, no. 2, 2020.

[7] M. Schubach, M. Re, P. Robinson, and G. Valentini, "Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants," *Scientific Reports*, vol. 7, 06 2017.