Predicting active regulatory regions using Deep Learning

Adem Baran Orhan[a]

2nd-year Undergraduate Erasmus Student -V08934

[a] *University of Milan, Via Festa del Perdono, 7, 20122 Milano*

Abstract

As a second-year undergraduate student in Computer Science, I tried my best to complete this project. Please consider all results are not accurate.

Using deep neural networks I predicted regulatory regions in the human cell line, specifically the "H1" cell line, for it is active or inactive. I used several neural network methods in the project like Perceptron, CNN, FFNN, and MMNN. Prediction results were obtained by comparing models by visualizing them by bar plots.

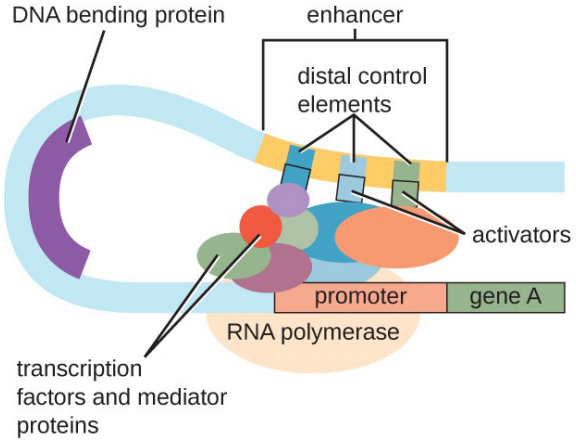


Figure 1: CRRs, through interactions with proteins and transcription factors, help specify the formation of diverse cell types and to respond to changing physiological conditions.

1. Introduction

Gene expression is the process by which the information encoded in a gene is used to either make RNA molecules that code for proteins or to make non-coding RNA molecules that serve other functions.[1] Cis Regulatory Regions(CRRs) in a genome regulate gene expressions.

Promoters and enhancers in the non-coding regulatory regions have some roles, such as Enhancers modulating the activity of promoters from linearly distal locations away from transcript initiation sites, and promoters specify and enable the positioning of RNA polymerase machinery at transcription initiation sites.

The project aims to use different machine learning methods to predict which regions are active or inactive in cis-regulatory regions. For the project, We will focus on a specific cell line which is H1 from the Human genome (which will be used as HG38). H1 is one of the most commonly used Human Embryonic stem cell lines.

Identifying active cis-regulatory regions in the human genome is critical for understanding gene regulation and assessing the impact of genetic variation on phenotype[2]

2. Models

First, we understand that our task includes binary classification.

We will use four Neural Network models: Perceptron, FFNN, CNN, and MMNN. Further in the project, we will compare the models' performance.

We used NADAM for the optimizer and binary cross-entropy for the loss function on each model.

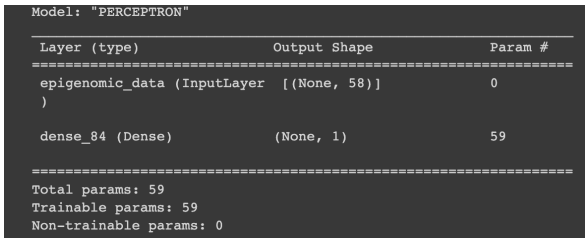
2.1. Perceptron

Perceptron is a basic binary classification algorithm invented back in the 1950s. When we use Perceptron during the project and the report, We mean multilayer perception, which has input, output, and multiple hidden layers. By backpropagation, it adjusts the weights in the neural network to minimize cost.

2.2. Feed Forward Neural Network

A feedforward neural network (FNN) is an artificial neural network wherein connections between the nodes do not form a cycle [4].

In our model, we used three layers with 64,32,16 neurons with RELU, and the last layer(output layer) used Sigmoid as an activation function with only



```
Model: "PERCEPTRON"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 epigenomic_data (InputLayer  [(None, 58)]             0
 )

 dense_84 (Dense)            (None, 1)                 59

=================================================================
Total params: 59
Trainable params: 59
Non-trainable params: 0
```

Figure 2: Perceptron hyper-parameters summary.

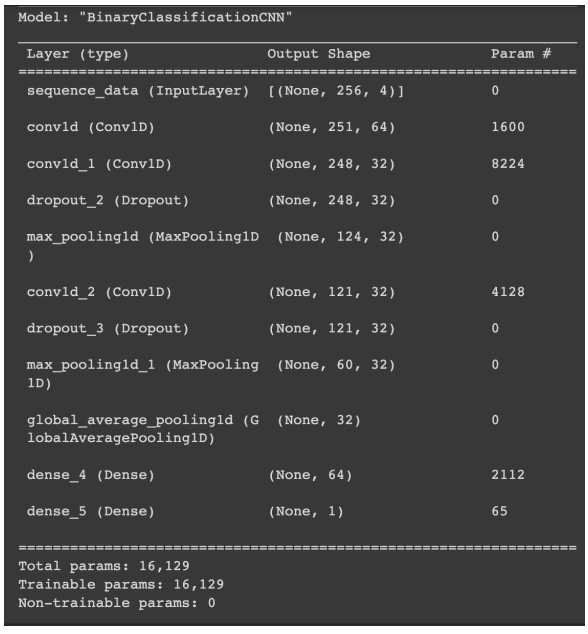
one neuron. Used NADAM as an optimizer and binary cross-entropy for loss function.

2.3. Convolutional Neural Network

Convolutional neural networks are a promising tool for solving the problem of pattern recognition. Most well-known convolutional neural network implementations require a significant amount of memory to store weights in learning and to work.[5]

In this model, we started with the 1D Convolution layer and the Max Pooling layer, then the 1D Convolution layer, followed by the Max Pooling layer, and lastly Global Average Pooling and Dropout layers. Again output layer with Sigmoid activation function.

Used NADAM as optimizer and binary cross entropy for loss function.



```
Model: "BinaryClassificationCNN"
_________________________________________________________________
 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 (MaxPooling  (None, 60, 32)           0
 1D)

 global_average_pooling1d (G  (None, 32)               0
 lobalAveragePooling1D)

 dense_4 (Dense)             (None, 64)                2112

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 16,129
Trainable params: 16,129
Non-trainable params: 0
```

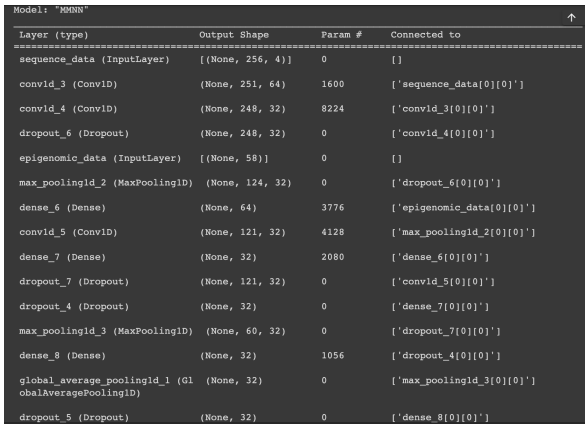
Figure 3: CNN hyper-parameters summary.

2.4. Multi Modal Neural Network

Multimodal learning is a good model to represent the joint representations of different modalities. [6]

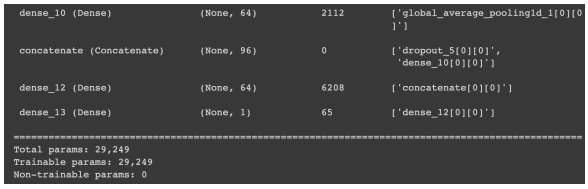
As name suggests I combined two different models for MMNN. Our models main purpose will be combining 2 different data in our project this is Sequence data and Epigenomic Data.

In my model FFNN and CNN combined.



```
Model: "MMNN"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 sequence_data (InputLayer)  [(None, 256, 4)]          0         []

 conv1d_3 (Conv1D)           (None, 251, 64)           1600      ['sequence_data[0][0]']

 conv1d_4 (Conv1D)           (None, 248, 32)           8224      ['conv1d_3[0][0]']

 dropout_6 (Dropout)         (None, 248, 32)           0         ['conv1d_4[0][0]']

 epigenomic_data (InputLayer)  [(None, 58)]            0         []

 max_pooling1d_2 (MaxPooling1D)  (None, 124, 32)       0         ['dropout_6[0][0]']

 dense_6 (Dense)             (None, 64)                3776      ['epigenomic_data[0][0]']

 conv1d_5 (Conv1D)           (None, 121, 32)           4128      ['max_pooling1d_2[0][0]']

 dense_7 (Dense)             (None, 32)                2080      ['dense_6[0][0]']

 dropout_7 (Dropout)         (None, 121, 32)           0         ['conv1d_5[0][0]']

 dropout_4 (Dropout)         (None, 32)                0         ['dense_7[0][0]']

 max_pooling1d_3 (MaxPooling1D)  (None, 60, 32)        0         ['dropout_7[0][0]']

 dense_8 (Dense)             (None, 32)                1056      ['dropout_4[0][0]']

 global_average_pooling1d_1 (Gl  (None, 32)            0         ['max_pooling1d_3[0][0]']
 obalAveragePooling1D)

 dropout_5 (Dropout)         (None, 32)                0         ['dense_8[0][0]']
```

Figure 4: MMNN hyper-parameters summary.



```
 dense_10 (Dense)            (None, 64)                2112      ['global_average_pooling1d_1[0][0
                                                                 ]']

 concatenate (Concatenate)   (None, 96)                0         ['dropout_5[0][0]',
                                                                  'dense_10[0][0]']

 dense_12 (Dense)            (None, 64)                6208      ['concatenate[0][0]']

 dense_13 (Dense)            (None, 1)                 65        ['dense_12[0][0]']

==================================================================================================
Total params: 29,249
Trainable params: 29,249
Non-trainable params: 0
```

Figure 5: MMNN hyper-parameters summary continued.

3. Experimental Setup

For the genomic sequence in this project, UCSC Genome Browser was used.[7] More specifically, we are predicting and developing neural networks on the HG38 dataset in the project. Since it is hard to compute all the dataset cell lines, we are interested in only the H1 cell line.

3.1. Considered Task

The project's main task is to develop deep neural models to predict specific non-coding regions active or not in the H1 cell line. Our window size is considered 256 nucleotides in the project. For sequence data, we have FFNN, CNN, and MMNN, and for epigenomic data, we have Perceptron, CNN, FFNN, and MMNN.

3.2. The Dataset

UCSC Genome browser provides the nucleotide sequence. In the project HG38 alias human genome sequences used. For retrieving the sequence

ucsc_genome_downloader package used.

FANTOM5 provides the cell lines labels relative to Cis-regulatory regions in BED format.

ENCODE provide the Epigenomic Data for regions which we are using. We have preprocessed epigenomic data such as A549, H1, HEK293, GM12878, HepG2, K562, and MCF-7. Since the epigenomic_dataset package was already preprocessed, there was no need to preprocess the ENCODE data.

3.3. Data Pre-processing

Before feature selection or visualization, preprocessing of data is crucial. Missing values identified, imputed with proper imputation technique. With the preprocessing, we expect better and more high-quality predictions.

3.3.1. Rate between features and samples

The first-rate between features and samples checked. The rate between features and samples is more than 1000. The result shows us we need to make a feature selection to get smaller value with more informative features.

3.3.2. NaN values

In the dataset, we have 20 NaN values, 19 from promoters and one from enhancers. Several techniques for the Nan value imputation include meaning, mode, and median imputations. Nearest Neighbour Imputation used in the imputation. KNN imputation with neighbors equal to 5(k=5) was chosen for the imputation.

3.3.3. Normalization

Normalization of the data also has an important role since we know that we may have too much noise in our data. Instead of Z scoring normalization, a robust scaler was used.

This Scaler removes the median and scales the data according to the quantile range. The IQR ranges between the first quartile (25th quantile) and the third quartile (75th quantile).[8]

3.3.4. Binarizing Labels

For binarizing, the data threshold methods are used. After several attempts to understand which one is giving better results decided to take two different thresholds for both promoters and enhancers. Promoters threshold set to 1 and enhancers set to 0 without dropping values.

For binarize, task FANTOM5 authors suggested that threshold at 1TPM for both promoters and enhancers. After trying this step on the data, we obtain 0.01 unbalance for enhancers.

Same settings with dropping the values between 0 and 1 is still better for the promoters but for the enhancers we have 0.01 unbalance.

Finally I tried different threshold and obtained better result for enhancers which is 0.04. Threshold at 1 for promoters and 0 for enhancers.

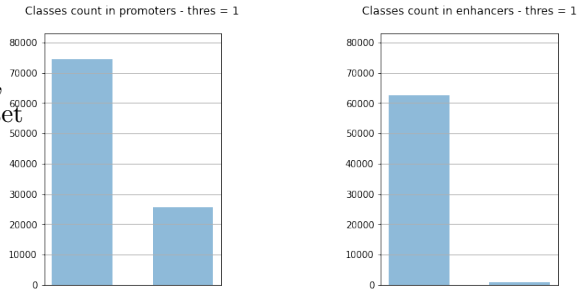


Figure 6: Class count for the threshold at 1, .

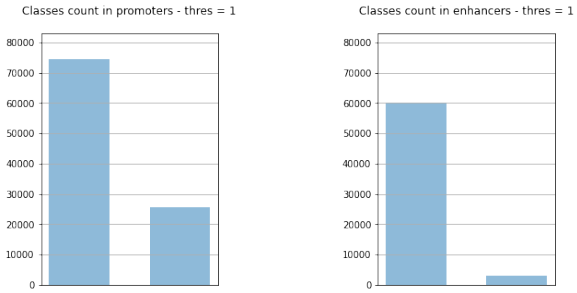


Figure 7: Class count for Promoters (threshold=1) and enhancers(threshold=0), .

3.4. Data Correlations and Distributions with Visualization

Correlation is a relation existing between phenomena or things or between mathematical or statistical variables[9].

The project considered correlation as a feature correlation. Dropped the features which do not have any correlation with the output.

3.4.1. Feature Correlations

For examining the feature correlation, we took the statistical significance level at 0.01, which we used as the p-value threshold during the code.

Used both Linear correlation and Monotonic correlation functions alias as Pearson and Spearman. As a result we obtain a list of uncorrelated cell lines 'enhancers': 'RNF2', 'SUZ12', 'promoters': 'H3K79me2', 'RAD21'

To compute non-linear correlations, the Maximal information coefficient measures the strength of the linear or non-linear association between two variables X and Y. Since this method is expensive to compute, it is given only uncorrelated lines.

For the extremely correlated features again I examine the same process with correlation threshold 0.95 and p value threshold 0.01. In this step none of the features seemed to be extreme correlate.

For better understanding how my features correlate with each other, I used scatter plot. With the figures some of pairs doing L shape. This

means they have rare correlation. Also some of the features looks like have better correlation but not more than 0.95 ratio.

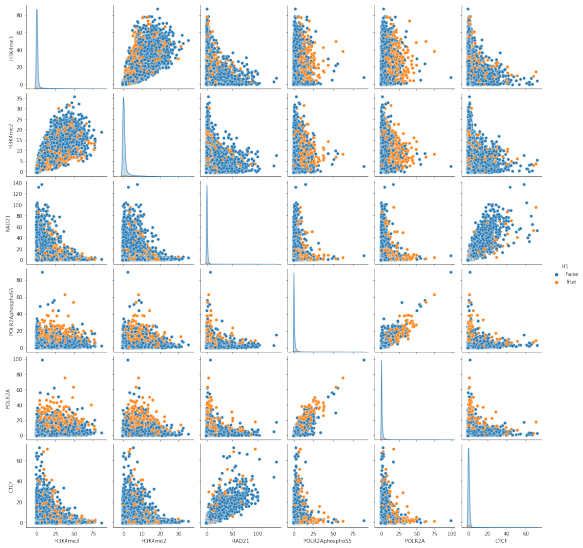


Figure 8: Most correlated features from enhancers region

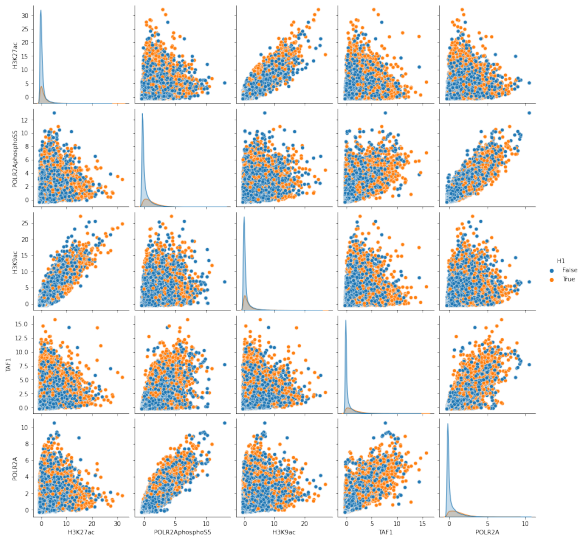


Figure 9: Most correlated features from promoters region

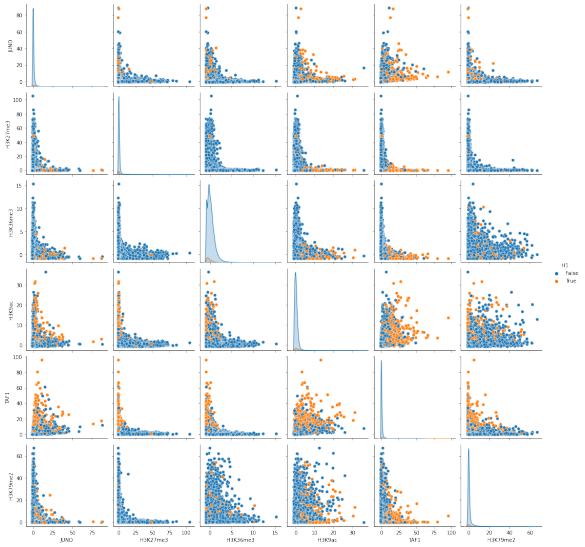


Figure 10: Least correlated features from enhancers region

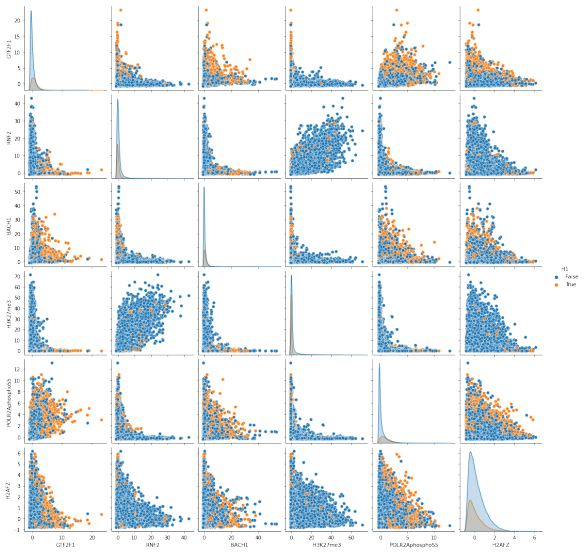


Figure 11: Least correlated features from promoters region

3.4.2. Feature Distribution

Most of the time understanding distributions, using histograms is not a bad idea. Since it is hard to show all the distribution I select 10 most different feature. And selected distance measure as euclidean.

For reducing the impact of the outliers I use quantile values as 0.05 and 0.95 and filtered them before 0.01 and 0.99 percentile

Our distribution on the labels or samples are exponentially and also understand that ranges are too different.

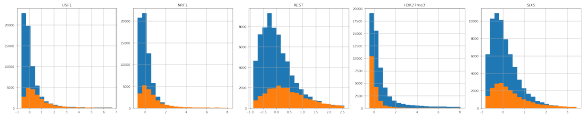


Figure 12: Features Distribution of Promoters with histograms

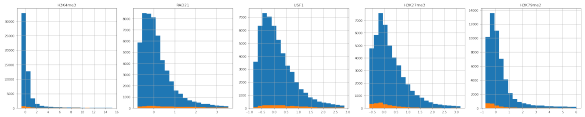


Figure 13: Features Distribution of Enhancers with histograms

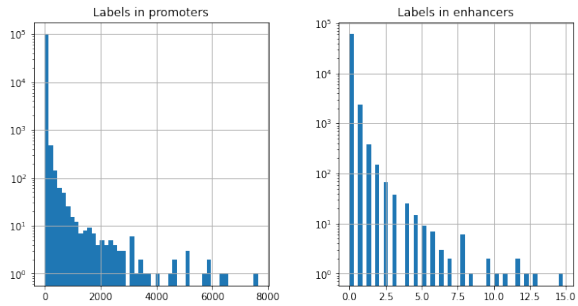


Figure 14: Labels Distribution with histograms

3.5. Data Visualization

Already visualize the features correlations with scatter plot and distribution with histograms. There are also visualizing techniques for high-dimensional data. We will use t-SNE and PCA to reduce the dimension for better visuals.

3.5.1. Visualization using t-SNE and PCA

PCA (Principal component analysis) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.[10]

t-SNE (t-distributed stochastic neighbor embedding) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map. It is 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 nearby points and dissimilar objects model similar objects are modeled by distant points with high probability

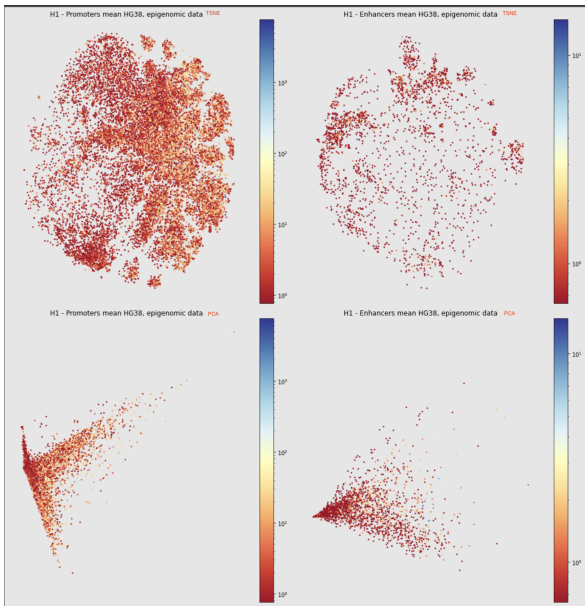


Figure 15: t-SNE and PCA visualization for Epigenomic Data

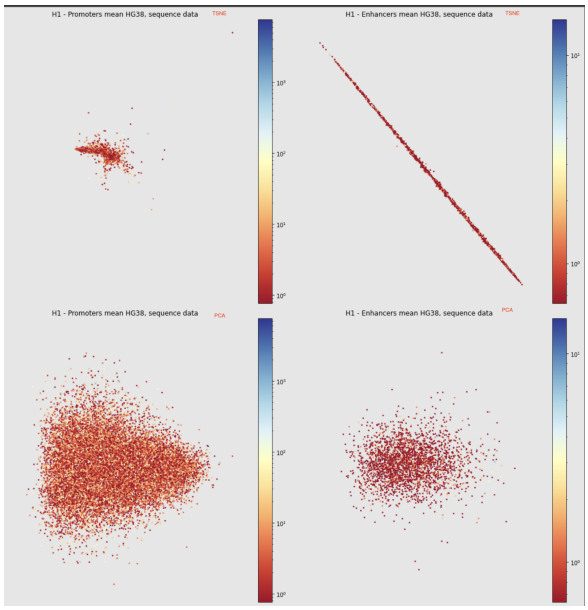


Figure 16: t-SNE and PCA visualization for Sequence Data

From the figures, it is easy to say t-SNE is more mixed and more circular on the epigenomic data, and PCA is more well distributed on the sequence data. In the project perplexity choosed as 50, but If I had not limited hardware perplexity value could be choosen higher values. t-SNE page at scikit learn explaining it more clear.[11]

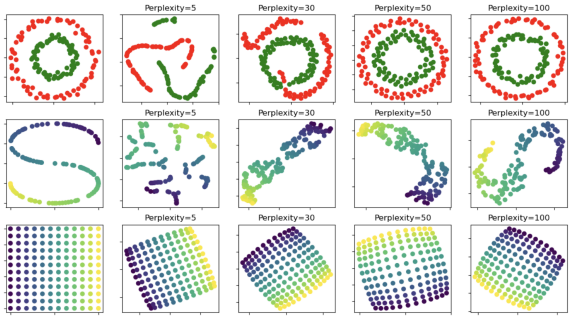


Figure 17: How perplexity can effect

3.6. Holdouts

In the feature selection, we used two options. No feature selection or Boruta feature selection.

Boruta algorithm is a wrapper built around the random forest classification algorithm. The random forest classification algorithm is relatively quick, can usually be run without tuning parameters, and gives a numerical estimate of the feature importance. It is an ensemble method in which classification is performed by voting of multiple unbiased weak classifiers — decision trees. These trees are independently developed on different bagging samples of the training set. 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.

Boruta is based on the same idea, which forms the foundation of the random forest classifier, namely, that by adding randomness to the system and collecting results from the ensemble of randomized samples, one can reduce the misleading impact of random fluctuations and correlations.[12]

My models are trained with two holdouts because we have limited hardware and data split into 80-20 train and test data for each holdout. For splitting, StratifiedShuffleSplit is used from the sklearn model selection library.

3.7. Result Analysis

Metrics and statistical test we used several metrics for evaluating the model's performances. AUROC, AUPRC, F1-score, Accuracy, and Precision

- AUROC: Area Under Receiver Operating Characteristic Curve
- AUPRC: Area Under Precision-Recall Curve
- F1-score: Measure of a test's accuracy.
- Accuracy: How close or far a given set of measurements are to their true value.

After using the above metrics, we use bar plots to see each model's performance.

Metrics and features are compared in the bar plots. In the report, not all the metrics are provided, but if you are interested in seeing all metrics from the project, they are in the Outputs folder on Github.[14]

If the p-value is more significant than the threshold, this means statistically indistinguishable.

4. Results

Under the Result title I will provide bar plots from models. We have opportunity to compare models with the figures.

For AUPRC metric MMNN and BoostedMMNN gived really good results. Considering the accuracy or other metrics for CNN and FFNN concatenation of them are come up with better results. Loss on the CNN model is highest among the other models. And also after lokking all the metrics and comparing them CNN has the worse performance. Reason for that is overfitting or given task is too complex for the neural network.

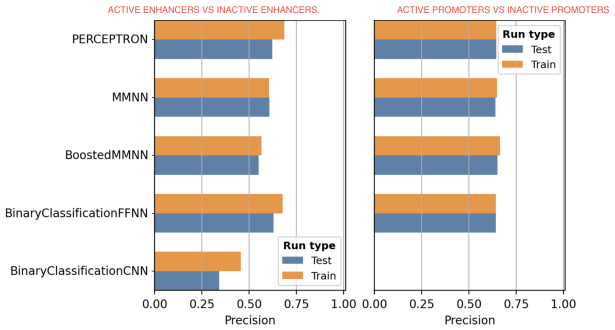


Figure 18: Precision metric for models

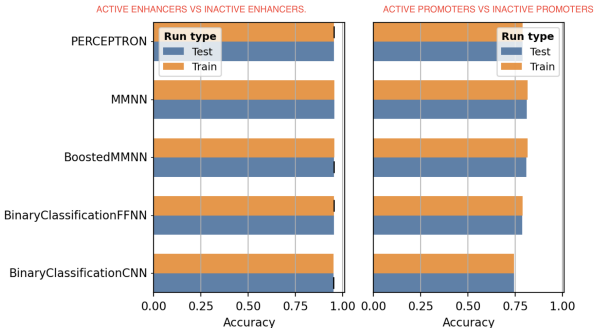


Figure 22: Accuracy metric for models

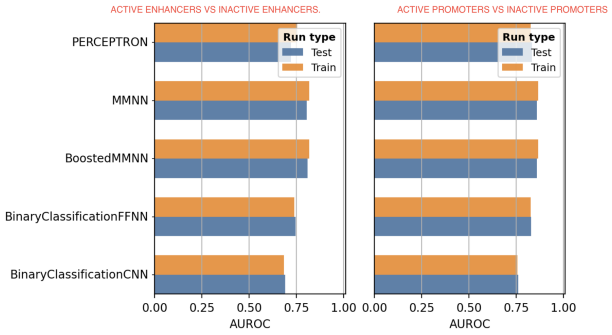


Figure 19: AUROC metric for models

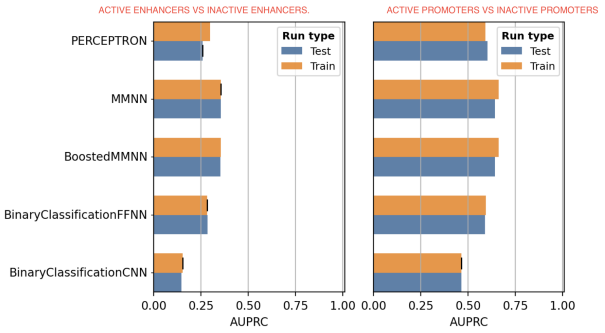


Figure 20: AUPRC metroc for models

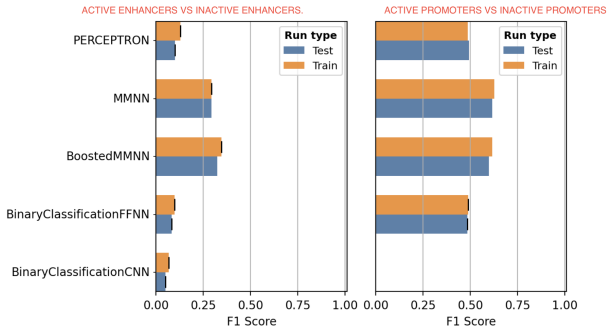


Figure 21: F1 Score metric for models

5. Conclusion

We used multiple Neural Networks during the project. Except my hardware or computational power was not enough to get better results. Already shown in the Genome-wide prediction of cis-regulatory regions using supervised deep learning methods[13], supervised deep learning methods can accurately predict active enhancers and promoters across the human genome. In my case, simple models are given as good results as Multi modal neural networks. We concatenated FFNN and CNN for the multi-modal neural network. From the bar plots, we can understand that CNN's performance is not good. With the project, we understand that a single neural network is not enough and even makes too dire predictions but comparing it with a different neural network such as FFNN and CNN results in higher performance. Since I have limited hardware during the project, It is possible to obtain a better result by, for instance, increasing perplexity or holdout number, more feature selections, and more comparing tests on them.

6. References

[1] https://github.com/LucaCappelletti94/bioinformatics_practice/blob/master/Notebooks/Data

[2] https://www.genome.gov/genetics-glossary/Gene-Expression

[3] https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2187-1

[4] Zell, Andreas (1994). Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. p. 73. ISBN 3-89319-554-8.

[5] https://www.sciencedirect.com/science/article/abs/pii/S0378475420301580?via

[6] https://en.wikipedia.org/wiki/Multimodal_learning

[7] https://genome.ucsc.edu/

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

[9] https://www.merriam-webster.com/dictionary/correlation

[10] https://github.com/LucaCappelletti94/bioinformatics_practice/blob/master/Notebooks/Prediction

[11] https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html#sphx-glr-auto-examples-manifold-plot-t-sne-perplexity-py

[12] https://github.com/LucaCappelletti94/bioinformatics_practice/blob/master/Notebooks/Prediction

[13] https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2187-1

[14] https://github.com/a-baran-orhan/Bioinformatics/tree/main/Outputs