How cost sensitive learning impact on the prediction of CRR in K52 cell line

Sara Manfredi

September 2022

1 Introduction

The scope of the project is to understand if the binary prediction of the activity versus the inactivity of cis-regulatory regions, that is an unbalanced classification problem(see Figure1), can be enhanced by the use of the cost sensitive learning (it assigns a higher cost to misclassification of the minority class during the training of the algorithm).

Cis-regulatory regions (CRR) are inside the non-protein-coding DNA (98% of the total DNA sequences) and play a crucial role in precise control of the gene expression. Promoters and enhancers act via complex interactions across time and space in the nucleus to control when, where and at what magnitude genes are active. CRRs, through interactions with proteins such as histones and sequence-specific DNA-binding transcription factors (TFs), help specify the formation of diverse cell types and respond to changing physiological conditions. While promoters specify and enable the positioning of RNA polymerase machinery at transcription initiation sites, enhancers modulate the activity of promoters from linearly distal locations away from transcript initiation sites.[7].

Using CAGE(Cap Analysis of Gene Expression), the FANTOM5 Consortium has identified an atlas of transcriptionally active promoters[6] and a permissive set of transcriptionally active enhancers characterized by bidirectional eRNAs [4] across hundreds of human cell types and tissues[4].

In this project, we'll try to predict the activity of enhancers and promoters conceptualizing it as a binary classification problem using the data provided both by ENCODE[5], that provides epigenomic features, and by the UCSC repository dataset, that provides sequence data obtained from the human reference genome.

Thus, for each cis-regulatory element (identified by its position inside the genome), we have two different representations: the (1) a set of numeric features suitable for training FFNN models and (2) nucleotide sequences to be processed with CNN models[3].

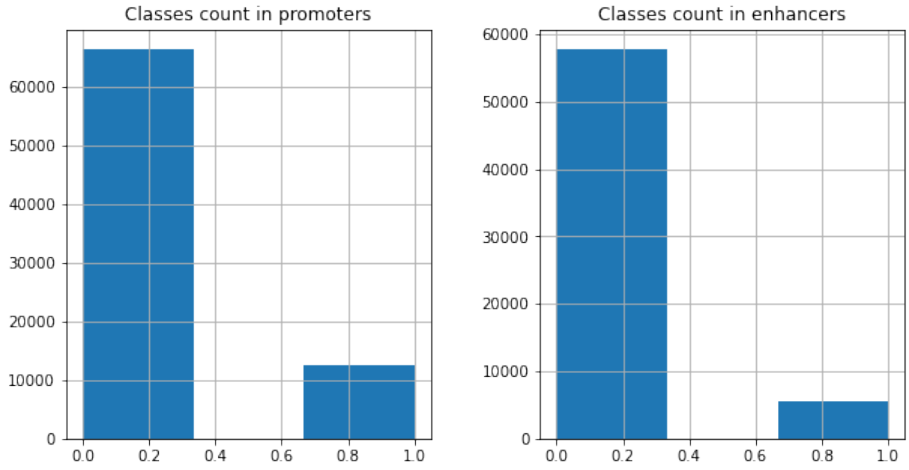

Figure 1: Class unbalance

2 Models

The scope of the project is to understand if the use of the cost sensitive learning inside four type of Neural Network results in better results.

For doing so, it has been developed a basic version for all the networks described in this section and the same version has been used for the cost sensitive learning so that the only difference is the use of a different cost function. The models used in the project are a Feed Forward Neural Network, a Convolutional Neural Network and two different types of Multi Modal Neural Networks. In this section are described the details of the networks.

The implementation of the models has been done with Keras and Tensorflow.[1]

2.1 Feed Forward Neural Network

The FFNN has been used to digest the epigenomic data. The layers of the FFNN can be found in Table 1. In particular, it has been used a ReLU activation function for all the Dense hidden layers. The ReLU activation function is a common choice for the hidden layers of a FFNN since this function is very close to be linear and so it preserve many of the properties that make linear models easy to optimize with gradient-base methods and also properties that make linear models good generators. A sigmoid activation function is used for the output layer since a binary prediction is required. To avoid overfitting the Dense layers have been spaced out by Dropout layers with 0.3 probability of dropping. The nadam optimizer[2] has been used with default values and the loss

[1] https://keras.io/

[2] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Nadam

Base-FFNN			
Layer Type	Units	Activation	Percentage
Dense	64	ReLU	-
Dense	32	ReLU	-
Dropout	-	-	0.3
Dense	32	ReLU	-
Dropout	-	-	0.3
Dense	1	Sigmoid	-

Table 1: Layers of the FFNN.

Base-CNN				
Layer Type	Units	Activation	Kernel/Pool size	Percentage
Conv1D	64	ReLU	6	-
Conv1D	32	ReLU	4	-
Dropout	-	-	-	0.3
MaxPool1D	-	-	2	-
Conv1D	32	ReLU	4	-
Dropout	-	-	-	0.3
MaxPool1D	-	-	2	-
GlobalAveragePooling1D	-	-	-	-
Dense	64	ReLU	-	-
Dense	1	Sigmoid	-	-

Table 2: Layers of the CNN.

function used is the binary cross entropy.

2.2 Convolutional Neural Network

The CNN has been used for the sequence data because they have been created to deal with sequences; to try to predict patterns. In this case we'll try to find motifs, sequences of nucleotides that correspond to binding site of TF. The layers of the CNN can be found in Table 2. In particular, it has been used a ReLU activation function for all the Conv1D hidden layers. To avoid overfitting the Conv1D layers have been spaced out by Dropout layers with 0.3 probability of dropping. After each Dropout layer a MaxPool1D layer has been used to aggregate the results. The kernel size choice has been done based on the assumption that the genomic motifs that characterize regulatory region have a size of 20 nucleotides at most[1]. At the end of the convolutional part, we use a GlobalAveragePooling1D instead of a Flatten layer because it has less paramenters and so it is less prone to overfitting.

2.3 Multi Modal Neural Networks

In the project there are two type of MMNN; both of them take as input both the epigenomic data and the sequence data and they both use a concatenation of the FFNN and the CNN described above. The first implementation take the results of the two NN already processed and then apply a common layer to produce the result, while the second has the same layers but they are trained together. From now on the first will be called MMNN and the second BoostedMMNN.

2.4 Cost Sensitive Learning

In order to use the cost sensitive learning, we passed to the fit function[3] of every model a *class_weight* dictionary computed using the method *compute_class_weight*[4] and specifying a *class_weight='balanced'*.

3 Experimental Setup

In this section it can be found the considered task(3.1), the datasets used and how to download them (3.2), the pre-processing techniques used (3.3), the data visualization techniques used (3.4), how the holdout have been carried out (3.5) and how the results have been measured (3.5). The experiments have been implemented using a Colab machine with a GPU NDIVIA Tesla T4.

3.1 The considered task

The task of the project is to understand if the use of a cost sensitive learning[2] can make a difference in the prediction of active versus inactive cis-regulatory regions.

3.2 Datasets

3.2.1 Epigenomic dataset

The epigenomic dataset are taken from the ENCODE project atlas and are downloaded using the *epigenomic_dataset*[5] library. This library already do some pre-processing. In particular we are collecting the data for the cell line K562[6], with window size equal to 256 nucleotides, we are downloading the data already binarized and we are specifing the values of the TPMs[7] in the following way:

- For the enhancers:
  - *min_active_tpm_value=0*

[3] https://www.tensorflow.org/api_docs/python/tf/keras/Model
[4] https://scikit-learn.org/stable/modules/generated/ sklearn.utils.class$_w eight.compute_class_weight.html$
[5] https://github.com/AnacletoLAB/epigenomic_dataset
[6] https://en.wikipedia.org/wiki/K562_cells
[7] https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

- *max_inactive_tpm_value=0*

- For the promoters:
  - *min_active_tpm_value=5*
  - *max_inactive_tpm_value=0*

This setup for the TPMs removes a significant portion of the promoters samples and does not take into account the issue of predicting values just above and just below the selected threshold for the enhancers but is in line with the work done here[7].

3.2.2 Sequence data

The genomic sequences can be retrieved from the UCSC Genomes browser[8] using the *ucsc_genomes_downloader*[9] simply specifying the version of the human genome we want to download, in this case *Hg38*. In order to use this information in our neural networks, we need to follow the steps:

1. Extract the bed indices equal to the one of the epigenomic data
2. Get the sequence of the nucleotides that corresponds to the bed indices this information in its one-hot-encoded version. For doing so we used the library *keras_bed_sequence*[10]

3.3 Data pre-processing

The epigenomic data before been used in the models have been pre-processed in the following manner:

- Rate between features and samples
- Detection of NaN values
- Data Imputation
- Drop of constant feature
- Normalization

For the genomic data, being a sequence of nucleotides and not features, is not possible to perform these kinds of pre-processing.

3.3.1 Rate between features and samples

We check that the rate between features and samples in the epigenomic data is not greather than 1; otherwise we could have a feature that separate the outputs/labels/classes just by chance. The ratio for the enhancers is *0.0067* and for the promoters is *0.0054*.

[8] https://genome.ucsc.edu/index.html
[9] https://github.com/LucaCappelletti94/ucsc_genomes_downloader
[10] https://github.com/LucaCappelletti94/keras_bed_sequence

3.3.2 NaN Detection

We check if the epigenomic data have some NaN values; we also check if some epigenomic feature or some part of the chromosome contains only, or a big fraction of NaN values, in that case we can delete them. It's not our case because there are some NaN values but are not concentrated in one feature or in one example.

3.3.3 Data Imputation

Since we find out that there are some NaN values, we impute them using the KNNImputer[11], where imputation for missing values is done using k-Nearest Neighbors.

3.3.4 Constant features

The aim of this step is to check if there are some features that have the same value for all the examples, and so that they don't bring any information to distinguish between the classes and we can drop them; in the case under question, there aren't.

3.3.5 Normalization

The normalization of the features is performed using RobustScaler[12], a scaler that use statistics robust to outliers.

3.4 Data correlations and distributions

In this step we want to understand if there are some features that are correlated with each other, that is we don't lose any information if we remove one of them. In order to carry out this test, we used the Pearson correlation coefficient[13], that measures the linear relationship between two datasets, and the Spearman correlation coefficient[14], that measures the monotonicity of the relationship between two datasets, both with *p-value = 0.01* and *correlation-threshold = 0.95*.

The results show that in the promoter the feature POLR2G is highly correlated with the feature *RBFOX2* with correlation = 0.96 and the feature CBFA2T3 with TCF12 with correlation 0.951. Since the correlation for both pairs is just slightly over the correlation threshold, we have decided to not considered it statistically valuable and so we won't drop any feature. In the enhancers none pair was found as correlated under the same values for the *p_value* and *correlation-threshold*.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html
[12] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
[13] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html
[14] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

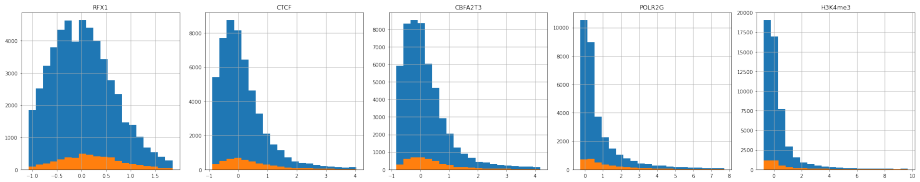

Figure 2: Most different features in enhancers

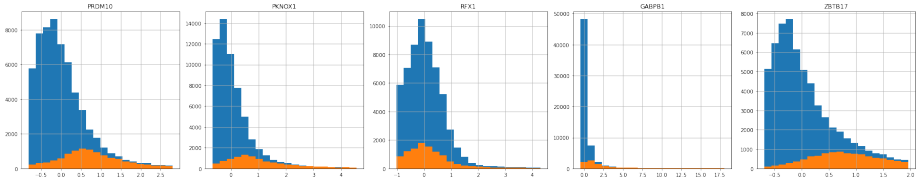

Figure 3: Most different features in promoters

3.5 Most differentiable features

In order to understand if we could use a linear classifier to resolve the classification problem we plot the features that are more distant from the others (the euclidean distance is used in the calculus[15]) and we search for a feature that has a clear distintion between the classes. Since such feature is not present, as we can see in Figure1 for enhancers and in Figure2 for promoters, it should be a problem where the Neural Networks can bring something important to the table.

3.6 Data Visualization

Since the datasets are composed in one case of several features, 429 for the epigenomic data and of 256 nucleotides one-hot-encoded for the sequence data, it's not possible to visualize the examples and their relative classes as they are. In order to visualize them, we use two techniques of data decomposition: Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE). Both the techniques reduce the features of the datasets to 2 and the results are shown in Figure3 and Figure4.

As we could expect, the visualization of the sequence data doesn't show any correlation with the labels while in the epigenomic data, especially for the promoters, we can see some part of the graph where one of the two class is predominant.

[15] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html

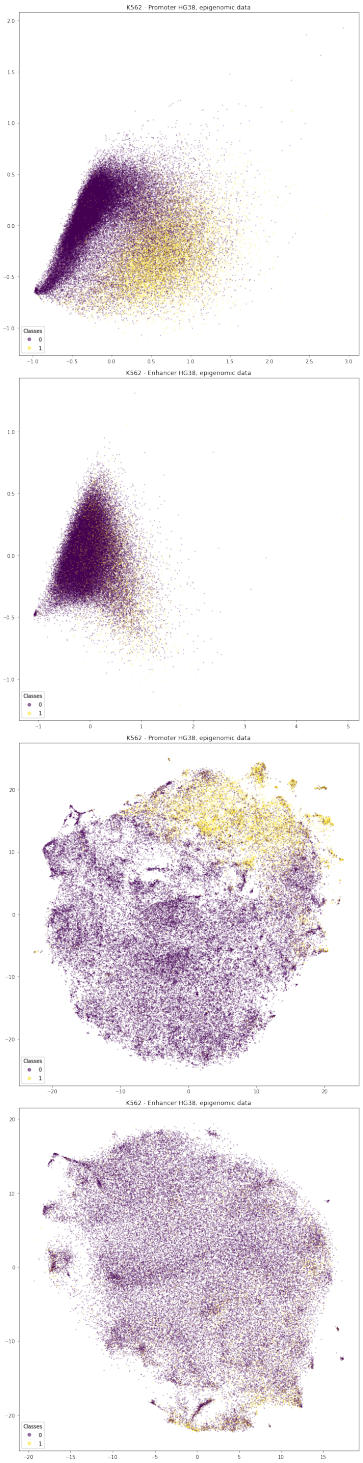

Figure 4: PCA and t-SNE visualization for the epigenomic data

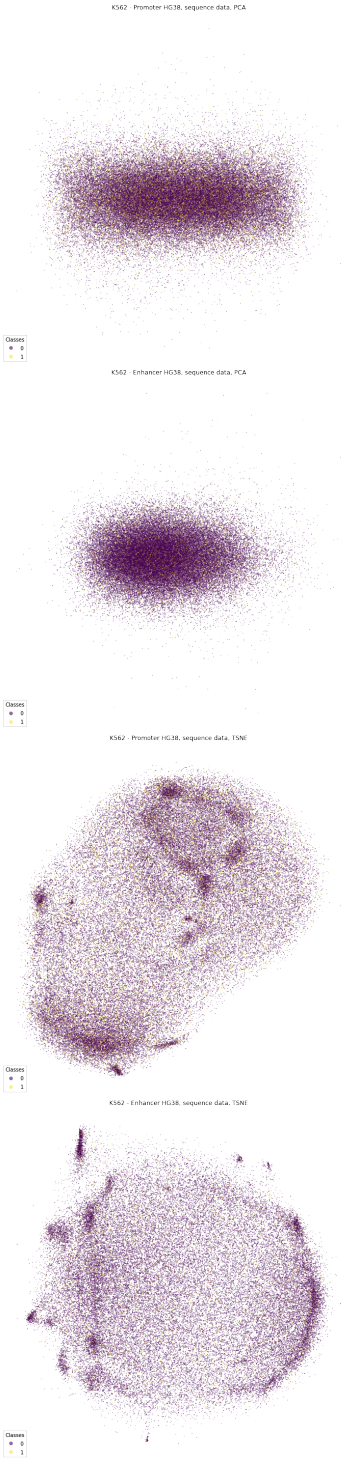

Figure 5: PCA and t-SNE visualization for the sequence data

3.7 Holdouts

In order to evaluate the model, we create a generator of stratified holdouts[16], that guarantees the same distribution of the output classes within the training and test holdouts.

3.7.1 Feature Selection With The Output

Inside each holdout we performed a feature selection with the output using Boruta[17] and a Random Forest Classifier[18] in order to reduce the number of features in the epigenomic data. Boruta execute statistical tests on the performance of a Random Classifier: it tells us the importance of each feature replacing it with a random set of values; if the importance of a feature is the same of the random set of values, we can say that the feature is not useful.

3.8 Results Analysis

In order to evaluate the Models with or without the use of the cost sensitive learning, we have created the confusion matrices of both the models and we have computed the Wilcoxon test[19] with $p\_value = 0.01$ on the metrics useful to evaluate unbalanced data:

- Recall
- AUROC
- AUPRC
- F1-score
- Balanced accuracy

The selection of the metrics is based on[8].

4 Results

In Figures 5,6,7,8 we can see the confusion matrices of the different models for the promoters while in Figures 9,10,11,12 the confusion matrices for the enhancers.

The results of the Wilcoxon test are summarized on Table from 3 to 10.

We can conclude that for the FFNN and CNN the cost sensitive version outperform the base version for all the metrics considered and indeed were the models most simple and with worst results, while concerning the MMNN and the BoostedMMNN the cost sensitive learning doesn't bring much difference in the results especially in the promoters were, as we can see also from the Figure 3, are the most simple task.

[16] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
[17] https://github.com/scikit-learn-contrib/boruta_py
[18] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[19] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

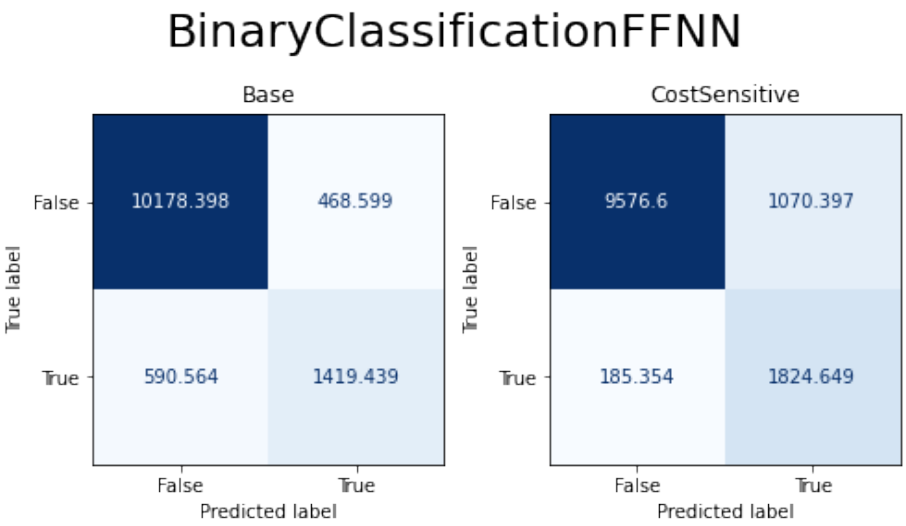

Figure 6: Confusion matrices of the FFNN with and without cost sensitive learning for the promoters.

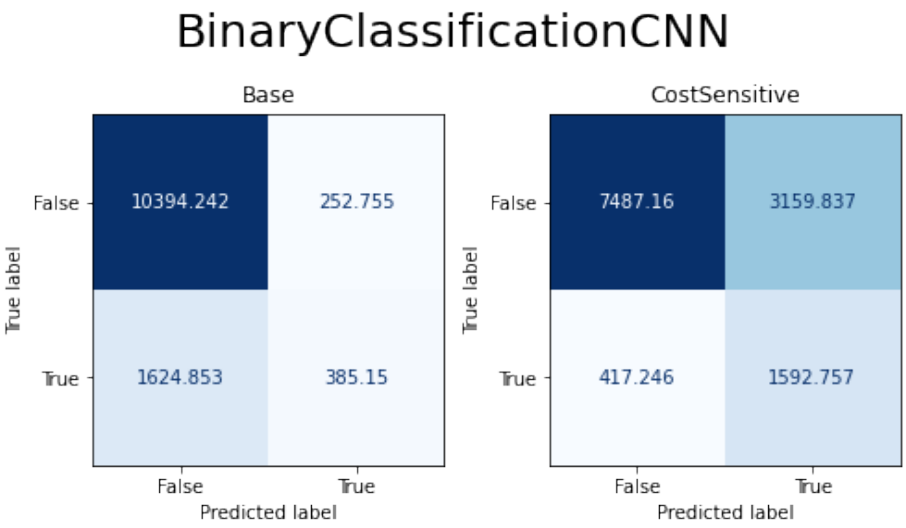

Figure 7: Confusion matrices of the CNN with and without cost sensitive learning for the promoters.

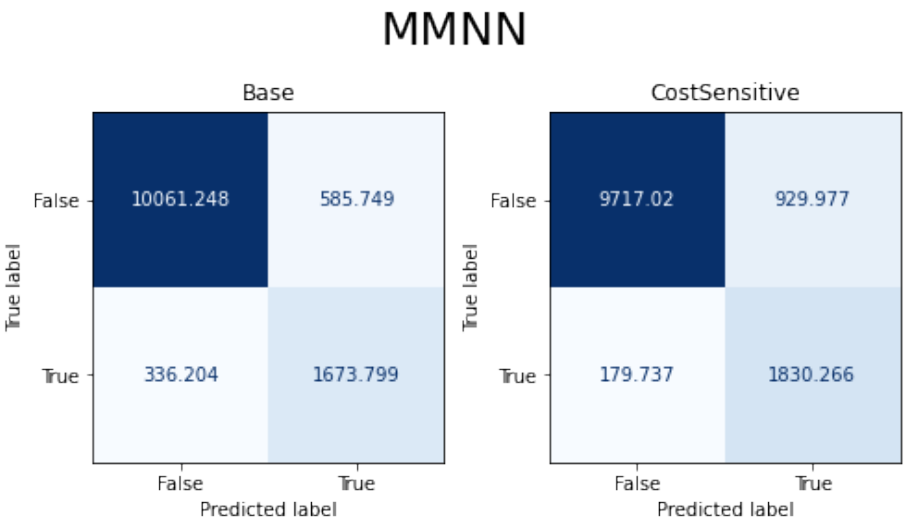

Figure 8: Confusion matrices of the MMNN with and without cost sensitive learning for the promoters.

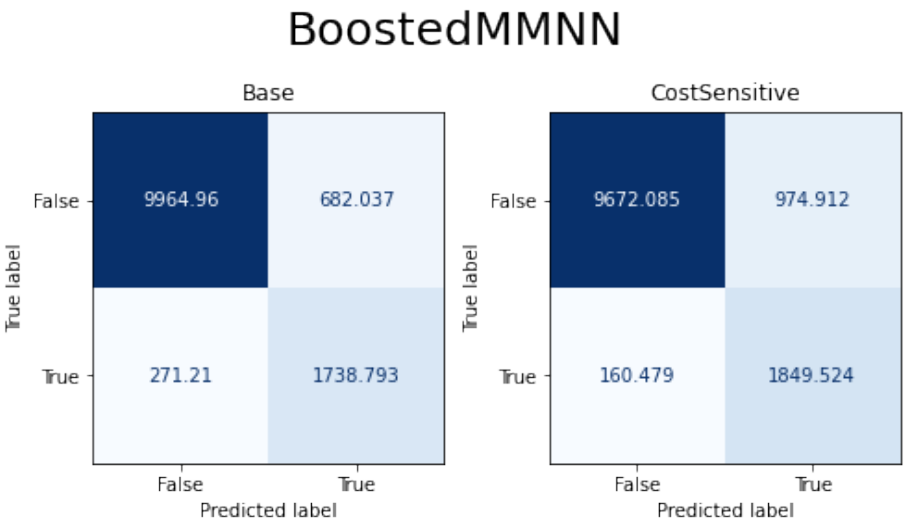

Figure 9: Confusion matrices of the BoostedMMNN with and without cost sensitive learning for the promoters.

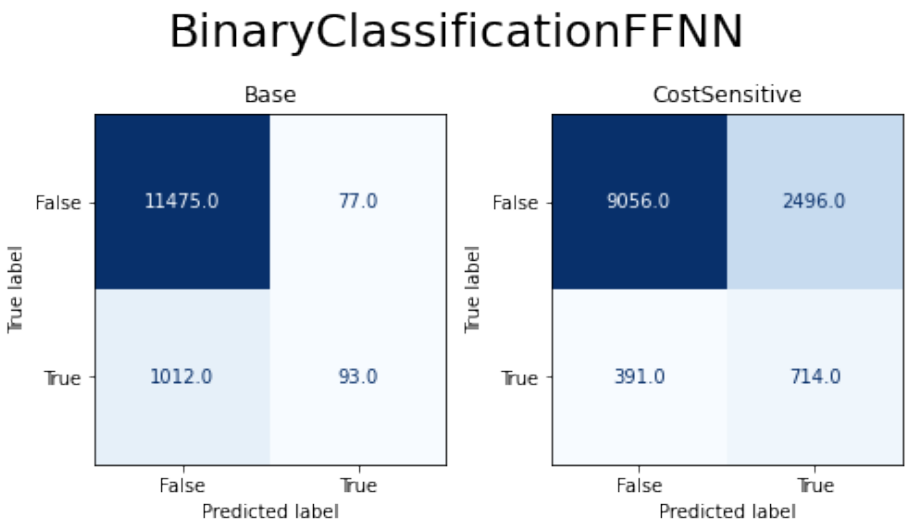

Figure 10: Confusion matrices of the FFNN with and without cost sensitive learning for the enhancers.

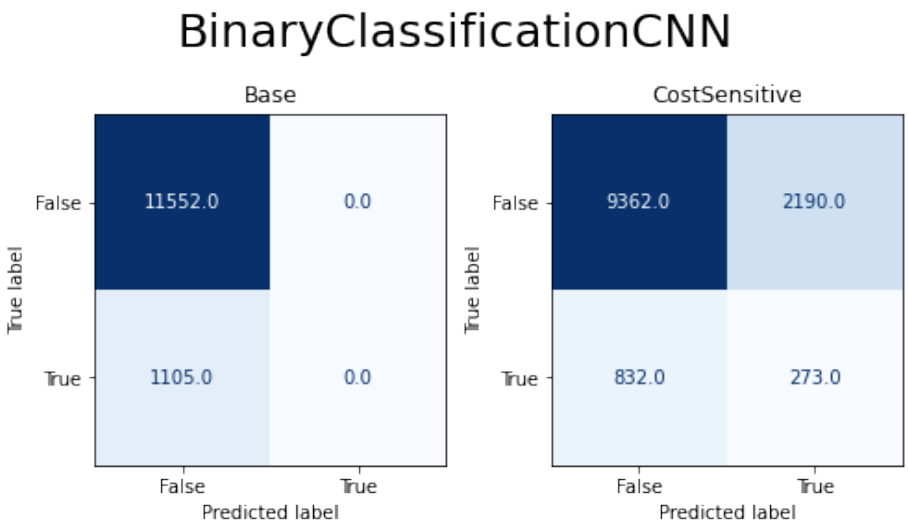

Figure 11: Confusion matrices of the CNN with and without cost sensitive learning for the enhancers.

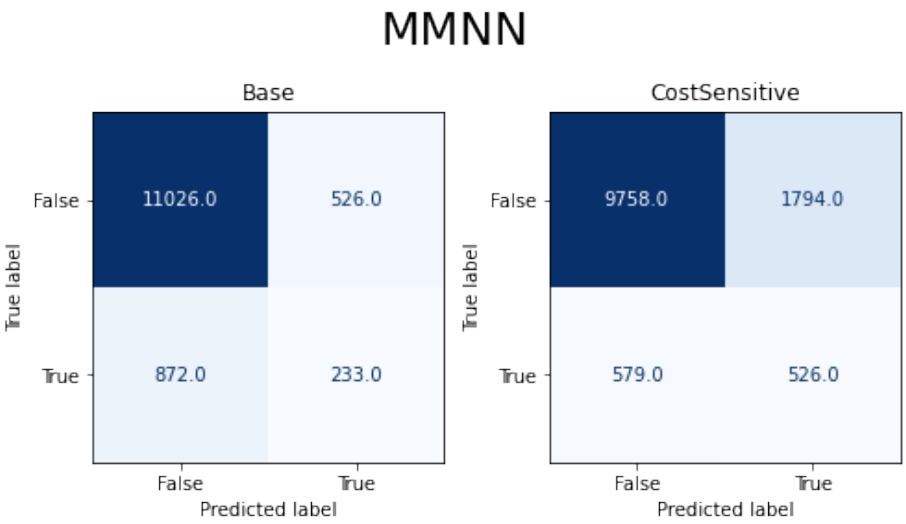

Figure 12: Confusion matrices of the MMNN with and without cost sensitive learning for the enhancers.

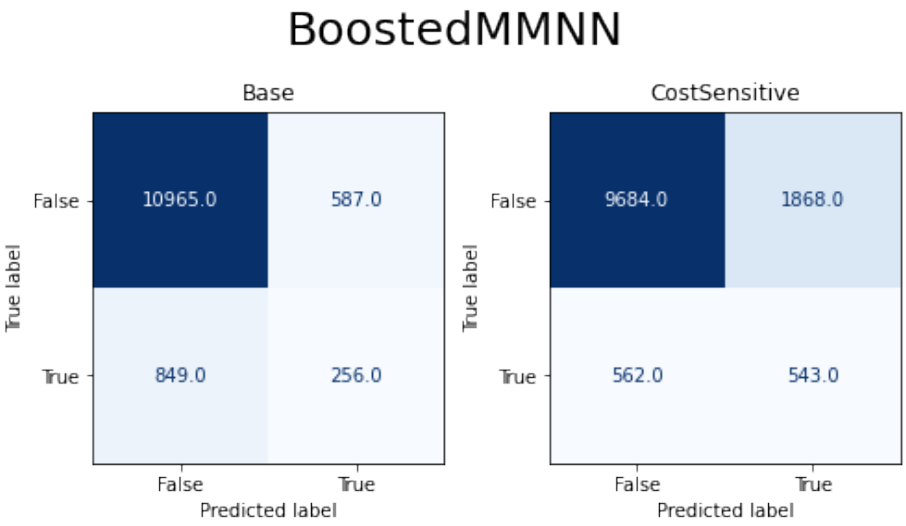

Figure 13: Confusion matrices of the BoostedMMNN with and without cost sensitive learning for the enhancers.

ENHANCERS - FFNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	AUROC
Cost sensitive	AUPRC
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 3: Results of the Wilcoxon test for the Enhancers and the FFNN.

ENHANCERS - CNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	AUROC
Cost sensitive	AUPRC
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 4: Results of the Wilcoxon test for the Enhancers and the CNN.

ENHANCERS - MMNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 5: Results of the Wilcoxon test for the Enhancers and the MMNN.

ENHANCERS - BoostedMMNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 6: Results of the Wilcoxon test for the Enhancers and the BoostedMMNN.

PROMOTERS - FFNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	AUROC
Cost sensitive	AUPRC
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 7: Results of the Wilcoxon test for the PROMOTERS and the FFNN.

PROMOTERS - CNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	AUROC
Cost sensitive	AUPRC
Cost sensitive	f1-score
Cost sensitive	balanced accuracy

Table 8: Results of the Wilcoxon test for the PROMOTERS and the CNN.

PROMOTERS - MMNN	
Best Model	Metric
Base	AUPRC
Base	f1-score
Cost sensitive	recall
Cost sensitive	balanced accuracy

Table 9: Results of the Wilcoxon test for the PROMOTERS and the MMNN.

PROMOTERS - BoostedMMNN	
Best Model	Metric
Cost sensitive	recall
Cost sensitive	balanced accuracy

Table 10: Results of the Wilcoxon test for the PROMOTERS and the BoostedMMNN.

References

[1] Alberts B. et al. *Molecular Biology of the Cell. 4th edition. Short DNA Sequences Are Fundamental Components of Genetic Switches*. Garland Science, 2002.

[2] Jason Brownlee. *How to Develop a Cost-Sensitive Neural Network for Imbalanced Classification*. URL: `https://machinelearningmastery.com/cost-sensitive-neural-network-for-imbalanced-classification/`.

[3] Luca Cappelletti et al. "Luca Cappelletti, Alessandro Petrini, Jessica Gliozzo, Elena Casiraghi, Max Schubach, Martin Kircher, and Giorgio Valentini." In: *Bioinformatics and Biomedical Engineering* (2020), pp. 1–12.

[4] Robin Andersson ans Claudia Gebhard et al. "An atlas of active enhancers across human cell types and tissues". In: *Nature* 507 (2014), pp. 455–461.

[5] The ENCODE Project Consortium. "An integrated encyclopedia of DNA elements in the human genome". In: *Nature* 489 (2012), pp. 57–74.

[6] The FANTOM Consortium, the RIKEN PMI, and CLST (DGT). "A promoter-level mammalian expression atlas". In: *Nature* 507 (2014), pp. 462–470.

[7] Yifeng Li, Wenqiang Shi, and Wyeth W. Wasserman. "Genome-wide prediction of cis-regulatory regions using supervised deep learning methods". In: *BMC Bioinformatics* 19 (2018), pp. 1–14.

[8] Le Wang et al. "Review of classification methods on unbalanced data sets". In: *IEEE Access* (2021), pp. 1–23.