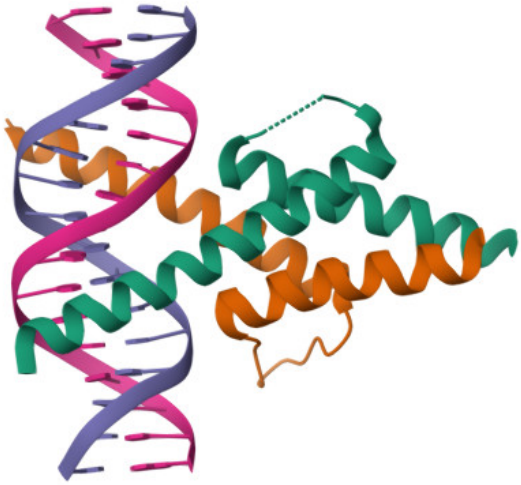
Leveraging Machine Learning For BHLH Protein Analysis & Prediction

Quantitative & Financial Modeling

Computational Biology



Professor:
Achraf EL ALLALI

Performed by:
Ilham LAGHRISSI & Brahim MOKNASSI

February 4, 2024

Introduction :

Plants, such as the model organism Arabidopsis thaliana, heavily depend on a diverse array of proteins to control their growth, central to these processes are the basic helix-loop-helix transcription factors, a sizable family with pivotal roles in various activities. This study delves into the intriguing realm of bHLH factors in Arabidopsis, utilizing machine learning to unlock their mysteries.

BHLH factors stand out due to their distinctive protein structure, characterized by a DNA-binding basic region and a helix-loop-helix (HLH) domain crucial for dimerization. Their capacity to form associations, either with themselves or other protein types, enables them to finely regulate the expression of specific genes, influencing diverse cellular functions.

This research harnesses the power of machine learning, a robust computational tool, to investigate the characteristics of bHLH factors in Arabidopsis. Through the analysis of a comprehensive dataset encompassing DNA sequences, functional annotations, and expression levels, the machine learning model learns to discern unique patterns within the bHLH family. This allows the model to accurately identify novel members of this essential DNA-binding group.

The significance of this study extends beyond bHLH factors. To evaluate the model's versatility, it undergoes testing on datasets associated with other gene families. This step explores whether the bHLH-trained model can adapt and apply its knowledge to classify genes beyond its initial training scope. Such adaptability could open doors for broader applications in DNA-related gene family classification across diverse biological systems.

This report delves into the technical details of the project, explaining the methodology used, including DNA data preprocessing and machine learning algorithm selection. Furthermore, it presents the results of the classification tasks, emphasizing the model's performance in both DNA-binding and non-DNA-binding gene classification.

By deciphering the regulatory roles of bHLH factors through machine learning, this study contributes to a deeper understanding of plant gene regulation, with a focus on the DNA level.

Preprocessing :

Reading Fasta Files:

This part of the code reads genetic sequences from Fasta files (fasta_file and bhlh_file). The read_fasta_file function parses the files, creating lists of dictionaries (fasta_sequences and bhlh_sequences) to store sequence information.

Calculating k-mer Distribution:

The code calculates the distribution of k-mers, or subsequences of length k, for each genetic sequence in fasta_sequences. The results are stored in the result list of dictionaries, capturing the sequence ID, k-mer distribution, and BHLH family indicator.

Creating DataFrame and CSV File:

This section converts the result list of dictionaries into a Pandas DataFrame. The DataFrame is then exported to a CSV file ('kmer_distribution_final_results2.csv') for further analysis and persistence.

Reading Data with Pandas:

Pandas is utilized to read the CSV file ('kmer_distribution_final_results2.csv') into a DataFrame named data. This step facilitates data manipulation and analysis in a tabular format.

Data Cleaning and Label Encoding:

The code performs data cleaning operations on the DataFrame, dropping unnecessary columns ('ID') and filling any missing values with zeros. Additionally, categorical data is transformed into numerical format using label encoding.

Train-Test Split:

The data is split into training and testing sets using train_test_split from sklearn.model_selection. This step is crucial for training machine learning models on a subset of the data and evaluating their performance on a separate test set.

Machine learning :

Confusion metrics :

In the realm of machine learning, particularly in the context of statistical classification, a confusion matrix, also referred to as an error matrix, is a specific tabular representation designed to visualize the performance of an algorithm, typically one employed in supervised learning.

Each row of the matrix corresponds to instances in an actual class, while each column represents instances in a predicted class, or vice versa. The name "confusion matrix" originates from its ability to easily highlight instances where the system confuses or mislabels two classes, indicating common errors in classification.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

From the values gotten in the confusion matrix we can calculate the accuracy, the precision, the recall and the F1 score :

⋆ **Accuracy:** tells us how close the measured value is to a known value:

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

⋆ **Precision:** tells about how accurate the model is in terms of those which were predicted positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

⋆ **Recall:** it calculates the number of actual positives the model was able to capture after labeling it as positive (true positive).

$$\text{Recall} = \frac{TP}{TP + FN}$$

⋆ **F1 score:** it gives a balance between precision and recall.

$$\text{F 1} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a machine learning algorithm used for classification and regression tasks. It operates on a lazy learning principle, memorizing the training dataset and predicting based on similarities during testing.

For classification, it employs majority voting. The choice of distance metric, such as Euclidean or Manhattan, is crucial, with 'k' being a key parameter that influences performance.

In out case, we want to see the accuracy of the KNN model and it's proficiency in classifying proteins in our dataset and compare those results to other models to see the best classifier that align with classification of proteins in such datasets.

Performance of KNN

Metric	Value
Accuracy	0.7105
Precision	1.0000
Sensitivity/Recall	0.4054
F1 Score	0.5769

Table 1: Performance Metrics for KNN Algorithm

Support Vector Machines (SVM)

A support vector machine (SVM) is a supervised learning algorithm used for many classification and regression problems, including signal processing medical applications, natural language processing, and speech and image recognition.

SVM aims to find a hyperplane that maximizes the margin between data points of different classes. In practical scenarios where complete separation is not possible, the algorithm allows a soft margin, permitting a small number of misclassifications.

Performance of SVM

Metric	Value
SVM Accuracy	0.8421
SVM Precision	0.9310
SVM Sensitivity/Recall	0.7297
SVM F1 Score	0.8182

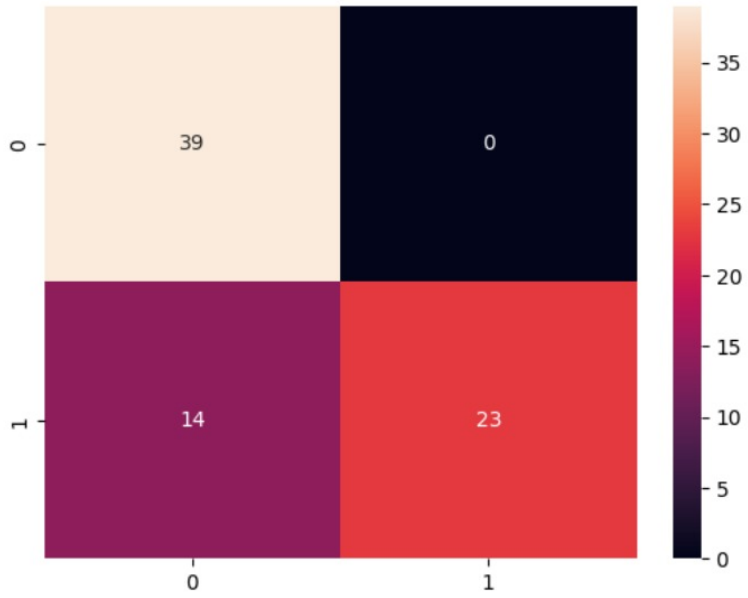
Table 2: Performance Metrics for SVM Algorithm

Random Forest (RF)

The Random Forest Algorithm is a widely used supervised machine learning method employed for both Classification and Regression tasks in the field of Machine Learning. This algorithm leverages the concept of an ensemble, resembling a forest with numerous trees. The strength of the algorithm lies in its robustness, which increases with the greater number of trees it incorporates. Specifically, Random Forest builds a classifier composed of multiple decision trees, each trained on different subsets of the given dataset.

The algorithm enhances predictive accuracy by aggregating the predictions from these individual trees. The core idea is rooted in ensemble learning, a strategy that combines multiple classifiers to address intricate problems and elevate the overall performance of the model.

Performance of RF



Metric	Value
Accuracy	0.8768
Precision	1.0
Sensitivity/Recall	0.7216
F1 Score	0.7667

Table 3: Performance Metrics for RF algorithm

We can easily observe that the random Forest is the algorithm that gives us best performance, a higher accuracy of 87% and higher recall with 72% that's why we will classify our data using the random forest classifier in feature selection as in the cross validation part, and that will be the object of the next section.

Results & Discussion :

As previously mentioned, the random forest model has consistently demonstrated superior accuracy and precision, making it our preferred choice for both feature selection and cross-validation. Our approach involves training the model with 80% of the dataset and testing it with the remaining 20%. However, due to the substantial size of our data file, the feature selection process, which involves traversing all features to identify the most relevant ones, is time-consuming, but, we have initiated the code execution, and as soon as results become available, we will provide updates on our findings.

Despite the ongoing execution of the code for feature selection, we anticipate longer processing times due to the extensive dataset. To mitigate this, we have also implemented another classification algorithm, k-nearest neighbors (KNN), specifically for feature selection. While the KNN algorithm requires a considerable amount of time, it has yielded results. Thus, our strategy involves leveraging both KNN and random forest algorithms—using KNN for efficient feature selection and random forest for robust cross-validation.

This dual-algorithmic approach allows us to balance computational efficiency with the accuracy and precision benefits offered by the random forest model in the context of cross-validation and the results are as follows:

	10	50	100	150	200	250	1000
Accuracy Metrics	Accuracy: 0.59 Precision: 0.50 Recall: 0.42 F1 Score: 0.46	Accuracy: 0.63 Precision: 0.57 Recall: 0.40 F1 Score: 0.47	Accuracy: 0.67 Precision: 0.64 Recall: 0.46 F1 Score: 0.54	Accuracy: 0.66 Precision: 0.63 Recall: 0.42 F1 Score: 0.50	Accuracy: 0.69 Precision: 0.70 Recall: 0.42 F1 Score: 0.52	Accuracy: 0.67 Precision: 0.68 Recall: 0.40 F1 Score: 0.50	Accuracy: 0.69 Precision: 0.78 Recall: 0.34 F1 Score: 0.47

Table 4: Table of the Accuracy Metrics for each number of best features

As illustrated in the accuracy metrics, the highest accuracy values are observed when the number of best features is set at 200 and 1000. However, it's noteworthy that the precision of the model with 1000 features surpasses that of the model with 200 features.

Despite our dataset encompassing approximately 4000 features, indicating a potential for improved performance, the limiting factor lies in the computational time required for processing.

From these findings, we can deduce that achieving the best accuracy, at 67%, is contingent on the careful selection of the number of features. The comparison between 200 and 1000 features highlights the nuanced relationship between accuracy and precision, emphasizing the trade-off that exists in optimizing model performance. While our data suggests the possibility of enhanced accuracy, the practical consideration of computational resources becomes crucial for real-world applications.