APPLICATION OF CLASSIFICATION TECHNIQUES ON HELICOBACTER PLYORI DATASET

Marvin Kika

CONTENTS

INTRODUCTION

Helicobacter Pylori is a bacterium that infects the mucus lining of the stomach and duodenum, leading to peptic ulcers, gastritis, and duodenitis. It is the only known microorganism that can thrive in the highly acidic environment of the stomach. The helical shape of the bacteria is thought to have evolved to penetrate and colonize the mucus lining. Studies for sequencing the genome of Helicobacter pylori have been completed, and several strains are known.

Our dataset contains information about helicobacter pylori, including nucleotide sequences and the genotype of the bacteria. Each sample of the dataset consists of 201 attributes. The first 200 are numerical attributes and represent a sequence of encoded

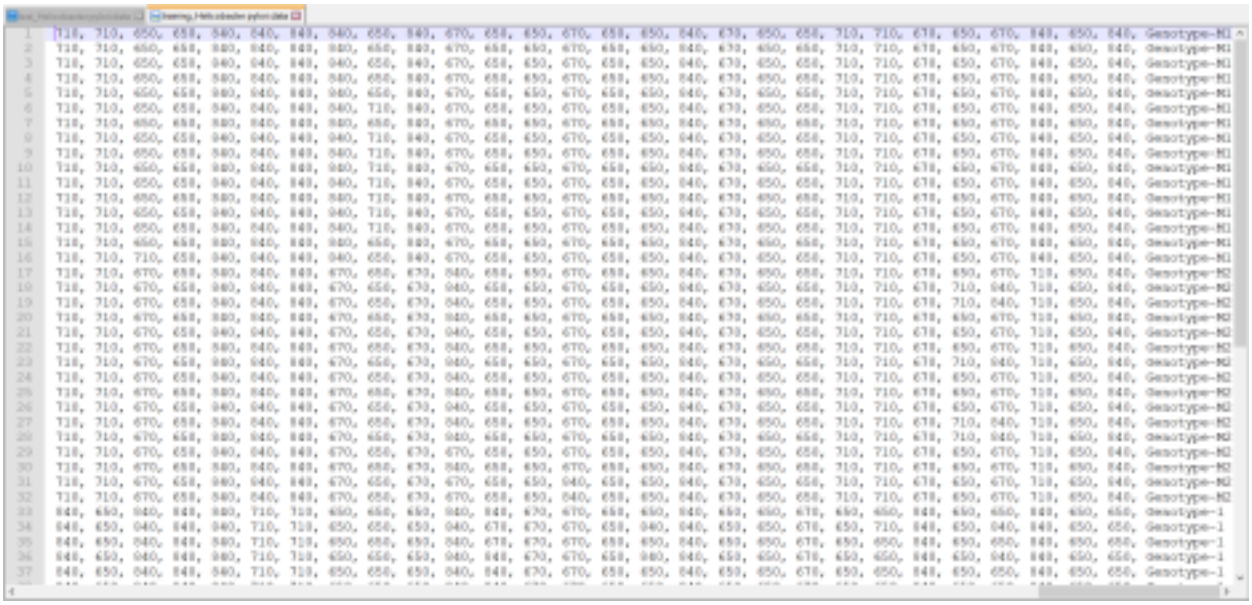
nucleotides. The last attribute is a categorical attribute which represents the genotype of the bacteria. The genotype is the target attribute, and it can have 4 possible values : Genotype M1, Genotype M2, Genotype 1, Genotype 2.

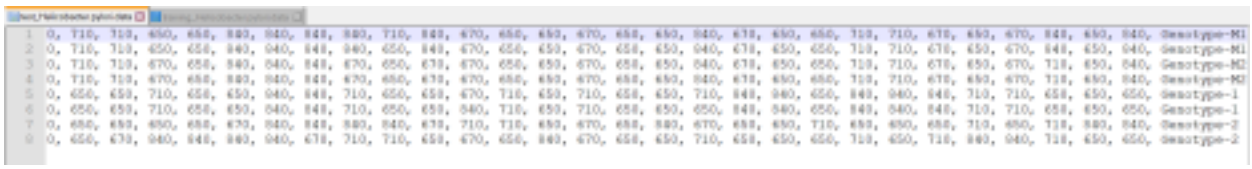
The factors of ASCII codes for every nucleotide type are used to convert them into numerical values:

- ❑ Adenine Nucleotide (A) : 65 x 10 = 650
- ❑ Guanine Nucleotide (G) : 71 x 10 = 710
- ❑ Thymine Nucleotide (T) : 84 x 10 = 840
- ❑ Cytosine Nucleotide (C) : 67 x 10 = 670

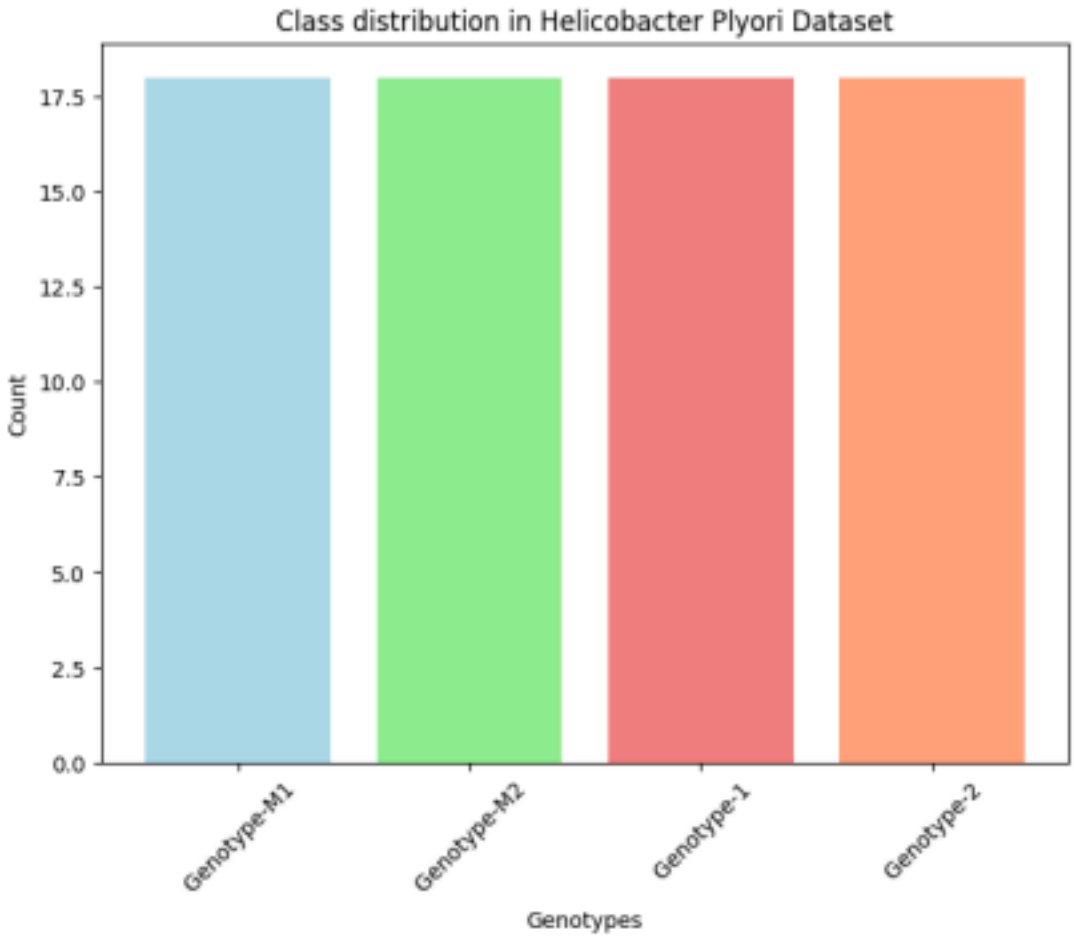
The dataset contains 72 records in total. It is balanced, with each of the 4 genotypes having 18 samples.
Training file contains 64 samples (16 for each genotype).
Testing file contains 8 samples (2 for each genotype).

Helicobacter Pyori dataset training file.





Helicobacter Pyori dataset testing file.



MATERIALS AND METHODS

Our project consists of training and applying multiple machine learning and deep learning algorithms to the dataset.The goal is to identify the most effective method for classifying the genotype of the Helicobacter pylori bacteria.

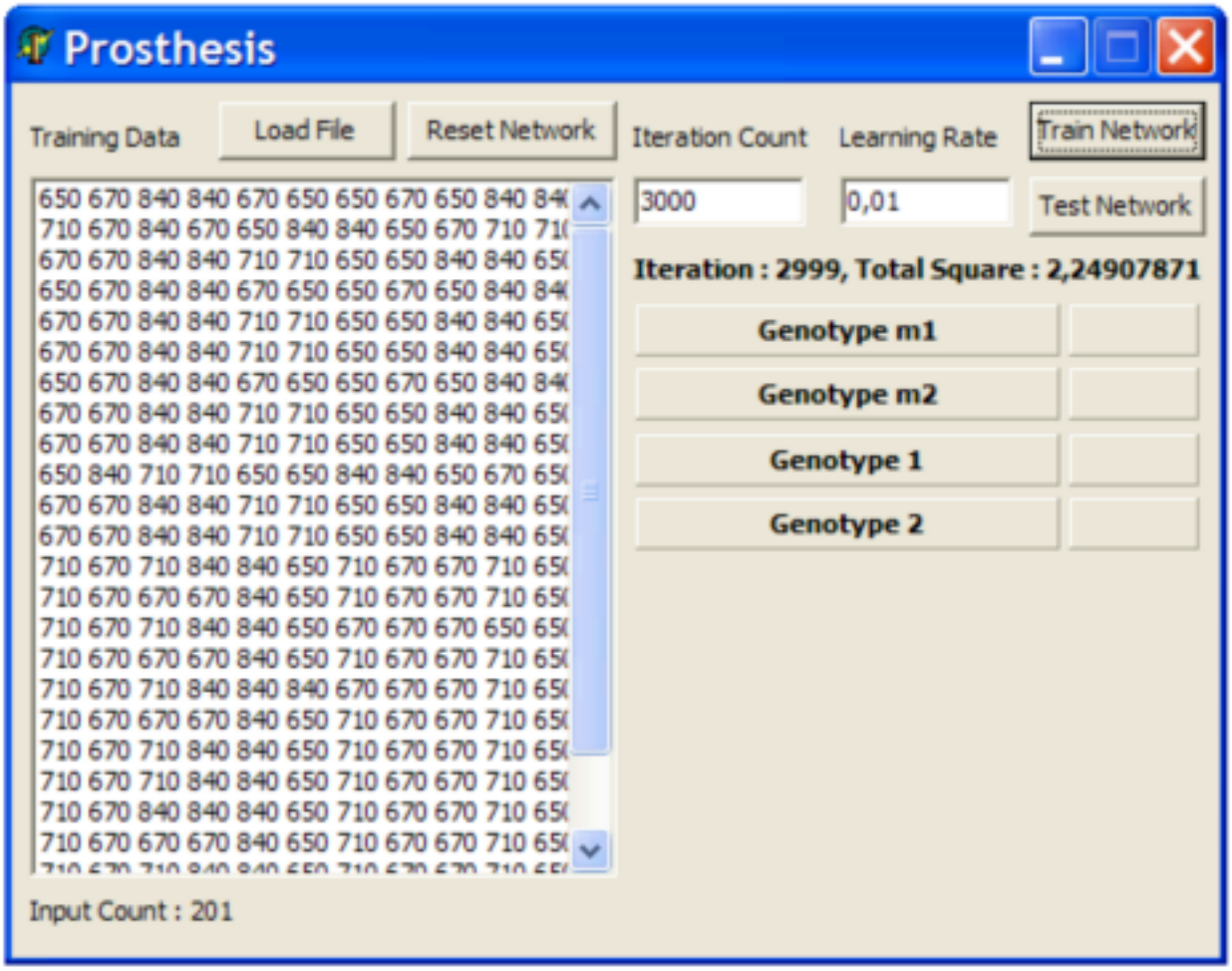
To implement the methods, we used Python in Google Colab. Normalization techniques, encoding, dimensionality reduction, different test/train splits of the data and parameter tuning were tested in order to determine which techniques are beneficial for this dataset, to compare models and to achieve high classification accuracy.

STATE OF THE ART SCIENTIFIC PAPERS

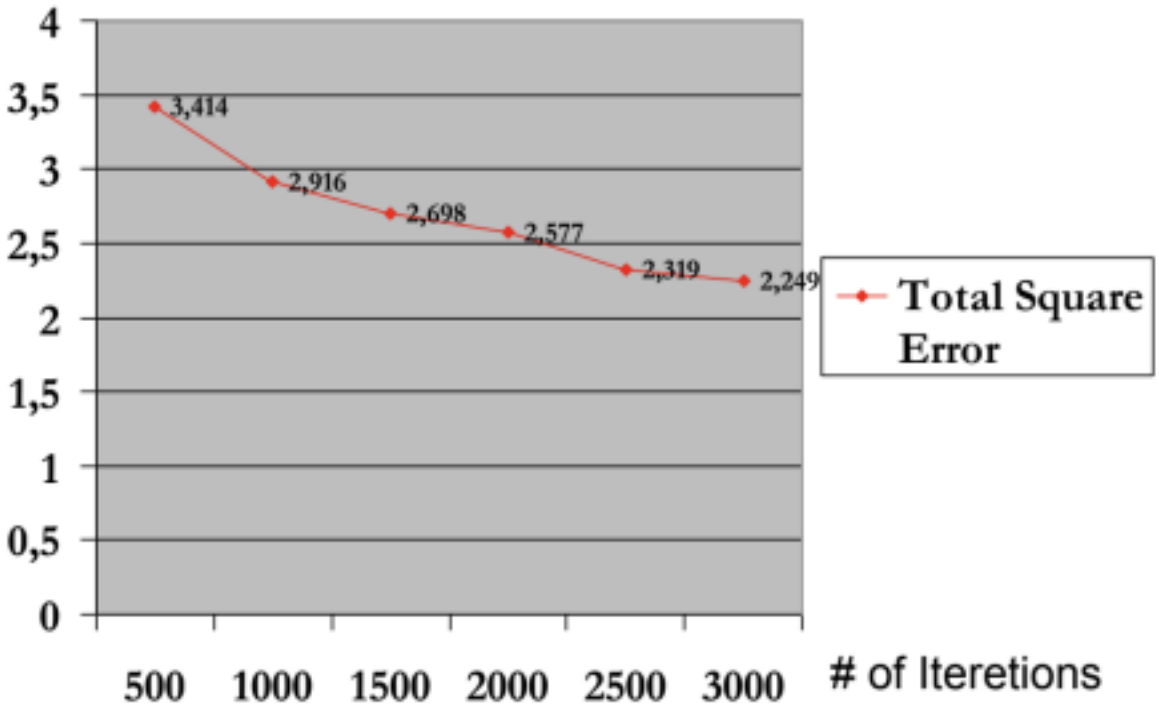
There were two scientific papers published for our dataset, by Prof. Dr. Bekir Karlik.

The first paper, named “A neural network approach for classification of Helicobacter Plyori based on national strains”, implemented a neural network approach with a

multi-layered perceptron (MLP). The total square error was reduced from 3,414 to 2,249 with 3000 training iterations, with a learning rate of 0,01. The encoded nucleotide values were used as inputs of the ANN, which contained 200 nodes in the input layer. The output layer contained 4 nodes, 1 for each genotype of the bacteria. The number of hidden layers and the accuracy of the model were not specified in the paper. The total square error values were plotted after training the model with 3000 iterations, with a learning rate of 0.01. The square error was reduced from 3,414 to 2,249.



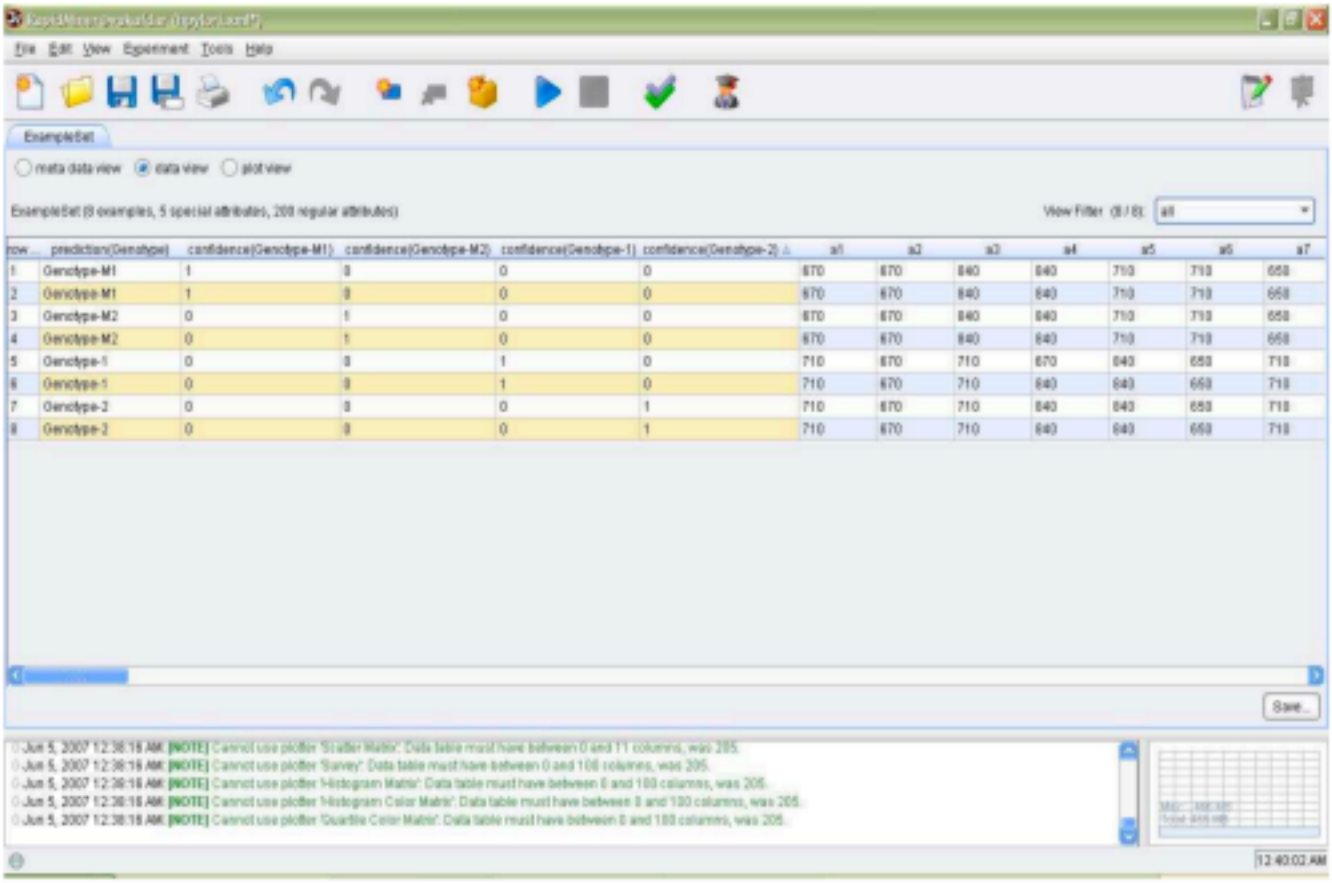
Implementation of the model [Karlik B. et. al., 2008]



Total square error curve [Karlik B. et. al., 2008]

The second paper “Classification of Helicobacter Plyori according to national strains using bayesian learning”, suggested the implementation of Naive Bayes classifier for this particular dataset.

The model achieved an Accuracy of 1.0 when classifying the testing set.



Screenshot of the testing phase [Karlik, B. , 2007]

DATA MANIPULATION

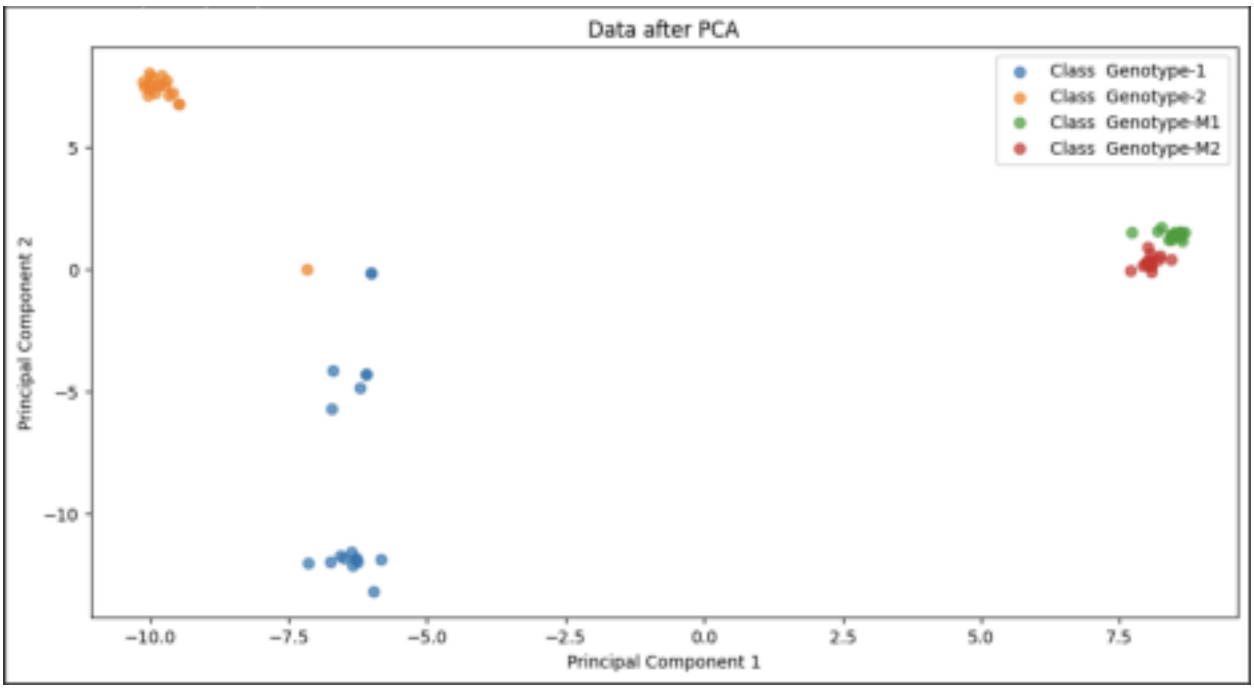
Original dataset had only 8 testing samples. We combined both files together (72 records in total) and performed 80/20 and 70/30 test/train splits, to get more information about the quality of the models we implemented.

We applied One-Hot encoding to the categorical genotype attribute.

We applied 3 different normalization techniques : Z-score, decimal scaling and MinMax normalization.

We applied dimensionality reduction using PCA.

For testing purposes, we applied One-Hot Encoding to the entire dataset when applying KNN with Hemming Distance function.

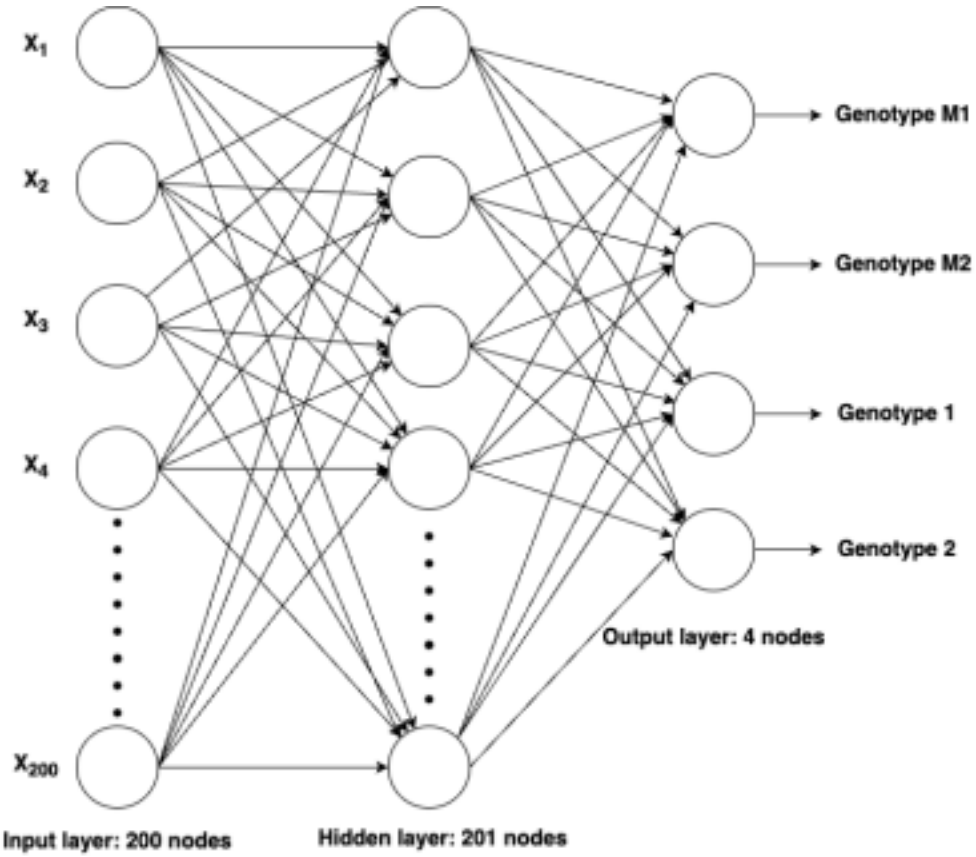


PCA visualization of the dataset

MULTI LAYER PERCEPTRON (MLP)

A Multi-Layered Perceptron with 1 hidden layer was used for backpropagation learning. The input layer contains 200 nodes, because the sequences of encoded nucleotides contain 200 numerical values. The hidden layer contains 201 nodes, 1 more than the input layer. The output layer contains 4 nodes, one for each genotype.

MLP structure of the system:



Multiple executions were conducted with different test/train splits of the data and different activation functions, with 500 training epochs. A learning rate of 0.01, which was used in the first research paper, proved to be too large for our implementation, producing very inconsistent, low accuracy results. Also, the SGD optimizer performed very poorly even with a smaller learning rate, often achieving an accuracy of less than 0.5.

For all cases, Adam optimizer was used with lr=0,001.

	ReLU	Tanh	Sigmoid
Original dataset (64 training instances, 8 testing instances)	1.0	0.875	0.875
Combined dataset (70/30 split)	0.86	0.68	0.68
Combined dataset (80/20 split)	0.86	0.79	0.72

We noticed inconsistent changes in accuracy for multiple executions. The table contains accuracy values that appeared more frequently for the tests we conducted. Inconsistent results may indicate similarities in the sequences of genomes. This idea was also proposed in the first state of the art paper.

Generally, ReLU function produced the most accurate results, reaching an accuracy value of 1.0 in the original dataset split.

RECURRENT NEURAL NETWORK (RNN)

Deep Learning was applied to the dataset by using a Recurrent Neural Network.

The network contained 2 RNN layers with 50 units each, 2 dropout layers and 1 dense layer. Adam optimizer was used, with learning rate=0.001

After 100 training iterations, an accuracy of 0.625 was achieved for the original split of the dataset. Parameter tuning and increasing the number of iterations did not improve the accuracy. The model performed way better for the other two splits, achieving an accuracy of 1.0 for the 80/20 split, and 0.95 for the 70/30. From these results, we can speculate that the model overfits the original training set, therefore it cannot generalize well in the original testing set. A 80/20 split seems to be the most appropriate test/train split for this implementation.

Dataset split	Accuracy
Original	0.625
80/20 train/test split	1.0
70/30 train/test split	0.95

The accuracy was significantly in the original split low because of the small testing dataset. When changing the dataset split, the model performs better due to the fact that it is being tested with more entities, making a failure in classification weigh less in accuracy metric.

It is important to consider that neural networks and Deep learning methods require a large amount of training data to learn significant patterns.

DECISION TREE

Decision tree classifier was applied to the different training splits.

Dataset split	Accuracy
Original	0.875
80/20 train/test split	0.93
70/30 train/test split	0.86

In the original split there was one misclassified record from Genotype 2. After adjusting the split to 80/20, a misclassification occurred with a record from Genotype 1. Further altering the split to 70/30 led to three misclassified records, two from Genotype 1 and one from Genotype 2. Neither normalizing the data nor reducing its dimensions improved the accuracy of the model.

NAIVE BAYES

Naive Bayes classifier was applied to the different train/test splits.

Dataset split	Accuracy
Original	0.875
80/20 train/test split	0.93
70/30 train/test split	0.95

Looking at the data, we got one misclassified record from Genotype 2 in the original split. When we changed the split to 80/20, there was still one misclassification, also from Genotype 2. Even with a 70/30 split, one record from Genotype 2 was misclassified. Neither normalizing the data nor reducing its dimensions changed these results.

SUPPORT VECTOR MACHINE

SVM classifier was applied to the different train/test splits.

Dataset split	Accuracy
Original	0.875
80/20 train/test split	1.0
70/30 train/test split	0.95

In the original split, there was one misclassified record from Genotype 2. Switching to an 80/20 split produced consistent results, with no misclassifications. Even with a 70/30 split, only one record from Genotype 2 was misclassified. Neither normalizing the data nor reducing its dimensions changed these results. Additionally, both linear and polynomial kernels produced the same results.

LOGISTIC REGRESSION

Logistic Regression classifier was applied to the different train/test splits.

Dataset split	Accuracy
Original	0.875
80/20 train/test split	0.93
70/30 train/test split	0.95

In the initial split, there was one misclassified record from Genotype 2. After changing it to an 80/20 split, one record from Genotype 1 was misclassified. Similarly, with a 70/30 split, there was one misclassified record from Genotype 2. Normalization and dimensionality reduction had no impact on the results.

K-NEAREST NEIGHBORS CLASSIFIER (NON-ENCODED DATA)

Logistic Regression classifier was applied to the different train/test splits.

Dataset split	Accuracy
Original	1.0
80/20 train/test split	1.0
70/30 train/test split	0.95

The table displays the results we got for K=5, using Euclidean Distance as the distance function. With a 70/30 split, there was one misclassified record from Genotype 1. For values of K less than 5, the model achieved perfect accuracy consistently. Normalization and dimensionality reduction had no impact on the results.

KNN + ONE HOT ENCODING + HEMMING DISTANCE

One-Hot Encoding was applied to the data to convert the numeric values into binary vectors. Hamming Distance was used as the distance function, to avoid any unrealistic distances between the encoded attributes.

The model achieved perfect accuracy with every split, every single time.

Increasing the value of K had no effect on the accuracy, since the Hamming Distance between the binary vectors remains the same.

COMPARISON OF METHODS

The table we got summarizes the results we got from all the models we tested.

Method	Accuracy (Original split)	Accuracy (80/20 split)	Accuracy (70/30 split)
MLP (ReLU)	**1.0**	0.86	0.72
MLP (Tanh)	0.875	0.79	0.79
MLP (Sigmoid)	0.875	0.72	0.75
RNN	0.625	**1.0**	0.95
Decision Tree	0.875	0.93	0.86

Naive Bayes 0.875 0.93 0.95 SVM 0.875 1.0 0.95 14

KNN (no encoding, Euclidean dist.,K=5)	**1.0**	**1.0**	0.95
KNN (encoding, Hamming dist.)	**1.0**	**1.0**	**1.0**

The worst result was achieved by RNN in the original split.

The best performing method was KNN, applied to one-hot encoded data using Hamming distance as the distance function.

Conclusions

After a series of experiments conducted on the dataset, it was determined that normalization was not necessary for our analysis. Given that the attributes take one of four discrete values, these features do not exhibit a meaningful continuous scale that would benefit from normalization techniques, which are typically used to standardize ranges in data with variable distances and scales. Additionally, the application of Principal Component Analysis (PCA) was found to be unnecessary in this context. Due to the relatively small size of our dataset, PCA did not yield any performance improvements. However, in scenarios involving much larger datasets, possibly encompassing thousands of entries, the use of PCA could potentially be beneficial. In such cases, PCA can help in reducing dimensionality, thus speeding up processing times and possibly enhancing model performance by focusing on the most relevant features.

Our dataset is very small, that's why deep learning models did not perform well in some cases.

Out of the Machine Learning models we applied, Decision Trees produced more inconsistent results.

All of the mistakes were made when classifying entries of Genotype 1 and 2, indicating that there may be similarities between these genotypes.

KNN classifier achieved perfect results for $K<5$.

KNN with Hamming distance applied to One-Hot encoded data achieved perfect results for any value of K, since the Hamming distance between the binary vectors is always the same.

References:

[1] B. Karlik et. al., Classification of Helicobacter Plyori According to National Strains Using Bayesian Learning, Mathematical and Computational Applications, Vol. 14, No. 3, pp. 241-251, 2009.

[2] B. Karlik A Neural Network Approach for classification of Helicobacter Plyori based on national strains.