IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ACCURATE PREDICTION OF COVID-19 USING DNA SEQUENCES THROUGH MACHINE LEARNING CLASSIFIER

Dr. Jai Ruby MCA., M.Phil. Ph.D and P. Aarthi
Department of Computer Applications, Sarah Tucker College, Thirunelveli-7.

ABSTRACT

According to the World Health Organization (WHO), the corona virus (COVID-19) pandemic is putting even the best healthcare systems across the world under tremendous pressure. The early detection of this type of virus will help in relieving the pressure of the healthcare systems. Chest X-rays has been playing a crucial role in the diagnosis of diseases like Pneumonia. As COVID-19 is a type of influenza, it is possible to diagnose using this imaging technique. With rapid development in the area of Machine Learning (ML), there had been intelligent systems to classify between Pneumonia and Normal patients. This work proposes the machine learning-based logistic regression classification algorithm. Bioinformatics and genomic signal processing use computational techniques to solve various biological problems. They aim to study the information allied with genetic materials such as the Deoxyribonucleic Acid (DNA), the Ribonucleic acid (RNA), and the proteins. Fast and precise identification of the protein coding regions in DNA sequence is one of the most important tasks in analysis. Existing Digital Signal Processing (DSP) methods provide less accurate and computationally complex solution with greater background noise. Hence, improvements in accuracy, computational complexity, and reduction in background noise are essential in identification of the protein coding regions in the DNA sequences. In this work, a new DSP based method is introduced to detect the protein coding regions in DNA sequences. Here, the DNA sequences are converted into numeric sequences using electron ion interaction potential (EIIP) representation. In this work, we use DNA sequence of the corona virus. The test is conducted using the data bases available in the National Centre for Biotechnology Information (NCBI) site.

1. INTRODUCTION

The biological activity of every living organism is controlled by billions of individual cells. The control centre of each cell is the Deoxyribonucleic Acid (DNA) which contains a complete set of instructions needed to direct the functioning of every each one of the cells. The chemical composition of the DNA is the same for all living organisms. The DNA of every living organism contains four basic nucleotide bases: Adenine, Cytosine, Guanine, and Thymine, usually abbreviated using the symbols A, C, G and T respectively

Deoxyribonucleic acid (DNA) technologies have been widely used in genetic engineering, forensics, and anthropology. We can see that the size of the databases storing DNA, RNA, and amino-acid sequences is increasing exponentially (Matsumoto et al., 2000). As an example, the lengths of the 24 chromosomes in human are found to have 50 to 250 million base pairs (Human Genome Project Science, http://www.ornl.gov/sci/techresources/ Human_Genome/ project/ info.shtml).

The following properties have been identified in many sequences and formed the basis for all DNA compression algorithms.

• DNA sequences contain repeated substrings. Repeated substrings in DNA sequences are often longer than linguistic texts.

• DNA sequences contain repeated palindromes.

• DNA sequences contain repeated reverse complements.

Genes are sequences of base pairs that contain instructions to produce proteins. They are also related to heredity. The areas of the DNA that contain genes are thus called coding areas, while the remaining parts are called non-coding areas. In higher-level eukaryotes, genes are usually spliced up into alternating regions of exons and introns. The introns are non-coding DNA and are cutout before the messenger ribonucleic acid (mRNA) leaves the nucleus for the ribosome - where the protein specified by the mRNA is synthesized. The information in the mRNA thus represents the exons which are then used to make proteins. Apart from the four bases, another unknown element is found in the exons. The unknown base also participates in transforming the information within the cells, represented by an alphabet N. The DNA sequence is then represented by a set of {A, C, G, T and N} nucleotides. It is in the form of a double helix held together by hydrogen bonds. The nucleotides (A,T) and (C, G) are

complement pairs. Each nucleotide in one DNA strand always binds to its complementary nucleotide in another strand. The two strands are biologically similar to each other. Thus, only one strand needs to be encoded while another strand can be deduced from this strand. DNA sequences are not random sequences. They contain long-term repetitions in which the sub sequences are similar to each other. The following properties have been identified in many sequences and formed the basis for all DNA compression algorithms . DNA sequences contain repeated substrings. Repeated substrings in DNA sequences are often longer than linguistic texts. DNA sequences contain repeated palindromes. DNA sequences contain repeated reverse complements.
The non-coding area sometimes called as junk DNA contains redundant repeating sequences. The functions of the repetitive non-coding areas are not completely understood. Although the major attention has been on the coding areas, it has been shown that the non-coding areas could be performing some important function, and hence should not be ignored. Repetitive structures have been implicated in various diseases and genetic disorders.
There are plenty of specific types of data that need to be compressed, for ease of storage and communication. Among them are texts (such as natural language and programs), images, sounds, etc. In this paper, we focus on the compression of a specific kind of text only, namely genetic sequences. We consider primarily DNA and RNA sequences and discuss briefly proteins at the end of the paper. The deoxyribonucleic acid (DNA) constitutes the physical medium in which all properties of living organisms are encoded. The knowledge of its sequence is fundamental in molecular biology. Important molecular biology databases (EMBL, GenBank, DDJB) are developed around the world to store nucleotide sequences (DNA, RNA) and amino-acid sequences of proteins. It is well known that their size increases nowadays exponentially fast. Not as big yet as some other scientific databases, their size is in hundreds of GE [Kam91].

The compression of genetic sequences constitutes, therefore, a very challenging task. DNA and RNA sequences can be considered as texts over a four-letter alphabet, namely (A, C, G, T) (note that T is replaced with U in the case of the RNA). For complete genomes, these texts can be very long. The human genome, for instance, contains three billion characters over twenty-three pairs of chromosomes. It contains all the genetic material of human beings. Only about ten percent of its sequence contains genes. The rest is considered to be noncoding. No complete sequences of that size are known nowadays. The longest complete sequence now available in the third chromosome of yeast [Oli92]. It contains more than three hundred thousand characters and happens to be very resistant to compression. Finally, proteins can be seen as texts over a twenty-letter alphabet, namely the twenty amino-acids used as monomers. They are much shorter than DNA sequences. On average, their size is about one thousand characters.

2. LITERATURE REVIEW

1. Cross chromosomal similarity for DNA sequence compression , Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, published July 14, 2008

They study cross-chromosomal similarity for DNA sequence compression. The length and location of similar repeated regions among the sixteen chromosomes of *S. cerevisiae* are studied. It is found that the average percentage of similar sub sequences found between two chromosome sequences is about 10% in which 8% comes from cross-chromosomal prediction and 2% from self-chromosomal prediction. The percentage of similar sub sequences is about 18% in which only 1.2% comes from self-chromosomal prediction while the rest is from cross-chromosomal prediction among the 16 chromosomes studied.

2. Analysis of cross sequence similarities for DNA multiple sequence compression, Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, June, 2010

Current DNA compression algorithms rely on finding repetitions within the DNA sequence so that similar sub sequences can be encoded by referencing to each other. We explore similarities between different chromosomes of the sequence 'Saccharomyces cerevisiae'. These similarities are characterised by the existence of similar subsequences among different chromosomes. The longer the similar sub sequences are, the higher the cross-similarities are. This implies that it would be advantageous to compress two or more chromosome sequences together so that similar sub sequences found between multiple chromosome sequences can be encoded together.

3. A Simple and Fast DNA Compressor, G. Manzini and M. Rastero 2004

In this paper they consider the problem of DNA compression. It is well known that one of the main features of DNA sequences is that they contain substrings which are duplicated except for a few random mutations. For this reason most DNA compressors work by searching and encoding approximate repeats. We depart from this strategy by searching and encoding only exact repeats. However, we use an encoding designed to take advantage of the possible presence of approximate repeats. Our approach leads to an algorithm which is an order of magnitude faster than any other algorithm and achieves a compression ratio very close to the best DNA compressors.

4. Universal Intelligent Data Compression Systems: A Review, Ahmed Kattan, 2010
Researchers have classically addressed the problem of universal compression using two approaches. The first approach has been to develop adaptive compression algorithms, where the system changes its behaviour during the compression to fit the encoding situation of the given data. The second approach has been to use the composition of multiple compression algorithms. Recently, however, a third approach has been adopted by researchers in order to develop compression systems: the application of computational intelligence paradigms. This has shown remarkable results in the data compression domain improving the decision making process and outperforming conventional systems of data compression

5. Pattern recognition Using Genetic Algorithm, Majida Ali Abed, Ahmad Nasser Ismail and Zubadi Matiz Hazi, 2010.

The recognition processes is among the many intelligent activities of the human brain system. This paper is concerned with the Pattern recognition (isolated Arabic characters) using genetic algorithm to satisfy a successful recognition operation. The unknown character is read from a file and many operations will perform on it to manipulate it and extract its features, to compare these features with saved template's features .The ratio of successful was over 95%. The proposed system has been implemented and tested on Delphi 6 environment

6. A new challenge for compression algorithms: Genetic Sequences, S. Grumbach and F. Tahi, 1994
they analyze in some detail the properties of the sequences, which cause the failure of classical algorithms. We then present a lossless algorithm, biocompress-2, to compress the information contained in DNA and RNA sequences, based on the detection of regularities, such as the presence of palindromes. The algorithm combines sub stituational and statistical methods, and to the best of our knowledge, leads to the highest compression of DNA. The results, although not satisfactory, give insight to the necessary correlation between compression and comprehension of genetic sequences.

7. A universal algorithm for sequential data Compression, J. Ziv and A. Lempel, 1977
A universal algorithm for sequential data compression is presented. Its performance is investigated with respect to a non probabilistic model of constrained sources. The compression ratio achieved by the proposed universal code uniformly approaches the lower bounds on the compression ratios attainable by block-to-variable codes and variable-to-block codes designed to match a completely specified source.

8. Universal data compression algorithm based on approximate string matching,
I. Sadeh, 1996
A practical source coding scheme based on approximate string matching is proposed. It is an approximate fixed-length string matching data compression combined with a block-coder based on the empirical distribution. A lemma on approximate string matching, which is an extension of the Kac Lemma, is proved. It is shown, based on the lemma, that the deterministic algorithm converts the stationary and ergodic source, *u*, into an output process *v*, and under the assumption that *v* is a stationary process, after the scheme has run for an infinite time, the optimal compression ratio $R(D)$ is achieved.

9. A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison, X. Chen, S. Kwong and M. Li 1999

We present a lossless compression algorithm, GenCompress, for genetic sequences, based on searching for approximate repeats. Our algorithm achieves the best compression ratios for benchmark DNA sequences. Significantly better compression results show that the approximate repeats are one of the main hidden regularities in DNA sequences. We then describe a theory of measuring the relatedness between two DNA sequences. Using our algorithm, we present strong experimental support for this theory, and demonstrate its application in comparing genomes and constructing evolutionary trees.

3. METHODOLOGY

This project has following modules

Input DNA sequence

Preprocessing

- Remove unwanted character
- Count base

Feature Extraction

- Unigram
- Bigram
- Trigram
- Quadgram

Classification

The description of the modules are below

Input DNA sequence

Considering the five bases of DNA sequences {A, C, G, T and, N}, 2 bits are not enough to store / represent each base. However, if one applies, the standard compression tools, usually they fail to achieve compression, because, they are all designed for English text compression. It is well known that DNA sequences do not present the same regularities as found in linguistic texts. To design a good DNA compressor, one must take advantage of the regularities found in this kind of data.

Preprocessing

- Remove unwanted character
  To remove the unwanted characters like N.
- Count base
  To count the A,C,T,G character. For demonstrating the methodology used in this compressor (PRDNAC - Pattern Recognition based DNA Sequence Compressor),
  let us consider the following DNA sequence having 128 base pairs.

AGTCAGTCCTGAAAGCACCTAAGCCGAATCC
ANTACNTACCCGTCCGTANTTTTTAATTTTTNACC
GTTGCCTCCACTGACTGACGAACGTNCGAACTGA
TNGTATNGCTAAATNGCTAAAATCGNTN.

Feature Extraction

Feature extraction is a type of dimensionality reduction where a large number of pixels of the image are efficiently represented in such a way that interesting parts of the image are captured effectively.

Unigram, Bigram, Trigram

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. Using Latin numerical prefixes, an n-gram of size 1 is referred to as a "**unigram**"; size 2 is a "**bigram**" ; size 3 is a "trigram".

Scan the sequence and identify the repeating patterns. The repeating patterns are coded according to their uniqueness. In this case, the following patterns are identified: AG, AA, AC, AT, AN, CA, CC, CT, CG, GT, GA, GC, TC, TG, TA, TT, TN, NT, AGT,

Pattern Id	Pattern
P1	TTTTT
P2	AGTC
P3	AAGC
P4	TCCA
P5	NTAC
P6	CCGT
P7	AATC
P8	AN
P9	AA
P10	CG
P11	NG
P12	TN
P13	GT
P14	AT

AAA, AAG, AGC, ACC, AAT, ATC, ANT, CCT, CTG, CAC, CTA, CCG, CGA, CCA, CGT, GTC, GAA, GCC, GTA, TCC, TGA, TAA, TAC, TTT, NTA, AGTC, AAGC, AATC, CTGA, CTAA, CGAA, CCGT, GTCC, TCCA, TTTT and TTTTT. Among the above patterns some of them are reverse of it such as the pattern AGTC is reverse of CTGA, ACCT is the reverse of TCCA and CGAA is reverse of AAGC and so on. To achieve a higher compression ratio, this compressor finds the longest exact repeats. Then, the patterns are coded in the following way: if the pattern is not an existing one from n then it is coded as Pi+1 and the symbol table is generated. The Table 1 represents the symbol table required for coding the above mentioned sequence after eliminating the redundant patterns. The number of bits required to represent a pattern is determined by satisfying the condition that for any ‘ n’ , number of pattern formation $\leq 2n$, possible cases. In addition to it, one more bit is used for indicating the reverse.
Once the symbol table is generated with the identified repeating patterns, the coded table for the sequence is constructed. The coded table is given in Table 2. Then, the work file will have the following entries to represent the sequence. P2 P2 CTGA P3 ACCT P3 CGAA P4
P5 P5 P6 P6 P8 P1 P9 P1 NA P6 TGCC P4 CTGA CTGA CGAA P10 P12 CGAA CTGA P12 P13 P14 P11 P7 P14 P11 P7 AATC GN P12.
The pseudo code for extracting DNA sequences with pattern matching and genetic programming features are given below:
1. Read the Sequence and form the symbol table having the pattern lists.
2. Determine the number of bits required for representing the patterns.
3. Using the symbol table generated in step 1, form a coded table that contains the sequence of recognized patterns along with their positional values.
4. Construct another coded table to represent the patterns and their occurrences.
5. Form the work file to represent the sequence of patterns by using symbol and coded tables.
6. Apply the above procedure for the pattern of patterns recursively till the optimum value is achieved.
7. Update the work file with the indices and headers to retrieve the patterns.

Dataset
GenBank (http://www.ncbi.nlm.nih.gov/Genbank/) The GenBank database is maintained by the National Center for Biotechnology Information (NCBI), USA. The current release is Release 147 (20 April 2005), and contains similar number of sequence entry and nucleotide bases as in EMBL.

Classification

Logistic Regression is a popular statistical model used for binary classification, that is for predictions of the type *this or that, yes or no, A or B,* etc. Logistic regression can, however, be used for multiclass classification, but here we will focus on its simplest application.

The general workflow is:

1. get a dataset
2. train a classifier
3. make a prediction using such classifier

Logistic regression hypothesis

The logistic regression classifier can be derived by analogy to the *linear regression hypothesis which is*:

$h_\Theta(x)= \Theta^T x$

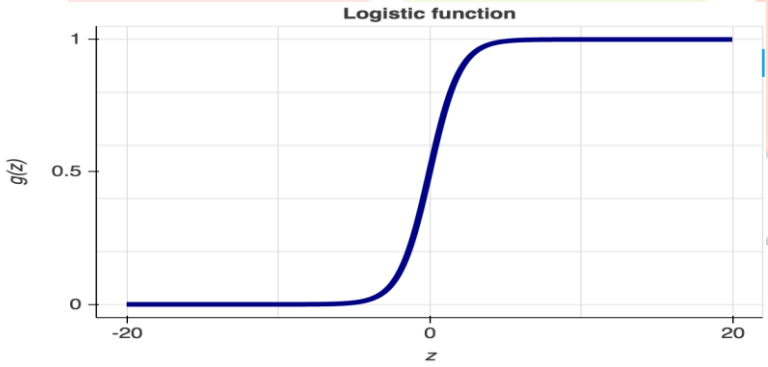
The result is the logistic regression hypothesis:

$$h_\Theta(x) = \frac{1}{1+e^{-\Theta Tx}}$$

Logistic regression hypothesis

The function **g(z)** is the **logistic function**, also known as the *sigmoid function.*

The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5.



Logistic function

Logistic regression decision boundary Since our data set has two features: height and weight, the logistic regression hypothesis is the following:

$h_\Theta(x)=g(\Theta_0+ \Theta_1x_1+ \Theta_2x_2)$

The logistic regression classifier will predict "Male" if:

$\Theta_0+ \Theta_1x_1+ \Theta_2x_2 > 0$

This is because the logistic regression "*threshold*" is set at g(z)=0.5, see the plot of the logistic regression function above for verification.

For our data set the values of θ are:

$$\Theta = \begin{pmatrix} 0.06924 \\ -0.49269 \\ 0.19834 \end{pmatrix}$$

To get access to the θ parameters computed by scikit-learn one can do:

 # For theta_0:print(fitted_model.intercept_)**# For theta_1 and theta_2:**print(fitted_model.coef_)

With the coefficients at hand, a *manual* prediction (that is, without using the function **clf.predict()**) would simply require to compute the vector product

$\boldsymbol{\Theta^T x}$ and to check if the resulting scalar is bigger than or equal to zero (to predict Male), or otherwise (to predict Female).

A visualization of the *decision boundary* and the complete data set can be seen here:

As you can see, above the decision boundary lie most of the **blue points that correspond to the Male class**, and below it all the **pink points that correspond to the Female class**.

Also, from just looking at the data you can tell that the predictions won't be perfect. This can be improved by including more features (beyond weight and height), and by potentially using a different decision boundary.

Logistic regression decision boundaries can also be non-linear functions, such as higher degree polynomials. Computing the logistic regression parameter

The scikit-learn library does a great job of abstracting the computation of the logistic regression parameter θ, and the way it is done is by solving an optimization problem.

Let's start by defining the logistic regression cost function for the two points of interest: y=1, and y=0, that is, when the hypothesis function predicts Male or Female.

$$\text{Cost}(h_\Theta(x),y) = \begin{cases} -\log(h_\Theta(x)) & \text{if } y=1 \\ -\log(1-h_\Theta(x)) & \text{if } y=0 \end{cases}$$

Then, we take a convex combination in **y** of these two terms to come up with the logistic regression cost function:

$J(\Theta)=-[y \log(h_\Theta(x))+(1-y)\log(1-h_\Theta(x))]$

$J(\Theta)=-[y \log(h_\Theta(x))+(1-y)\log(1-h_\Theta(x))]$

Logistic regression cost function

The logistic regression cost function is convex. Thus, in order to compute θ, one needs to solve the following (unconstrained) optimization problem:

$$\min_\theta J(\theta)$$

There is a variety of methods that can be used to solve this unconstrained optimization problem, such as the 1st order method *gradient descent* that requires the gradient of the logistic regression cost function, or a 2nd order method such as *Newton's method* that requires the gradient and the Hessian of the logistic regression cost function — this was the method prescribed in the scikit-learn script above.

For the case of gradient descent, the search direction is the negative partial derivative of the logistic regression cost function with respect to the parameter θ:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}(h_\theta(x) - y)\, x$$

Partial derivative of the logistic regression cost function.

In its most basic form, gradient descent will iterate along the negative gradient direction of θ (known as a *minimizing sequence*) until reaching convergence

Iterate until convergence

{

$\Theta_j = \Theta_j - \alpha(h_\Theta(x)-y)x$

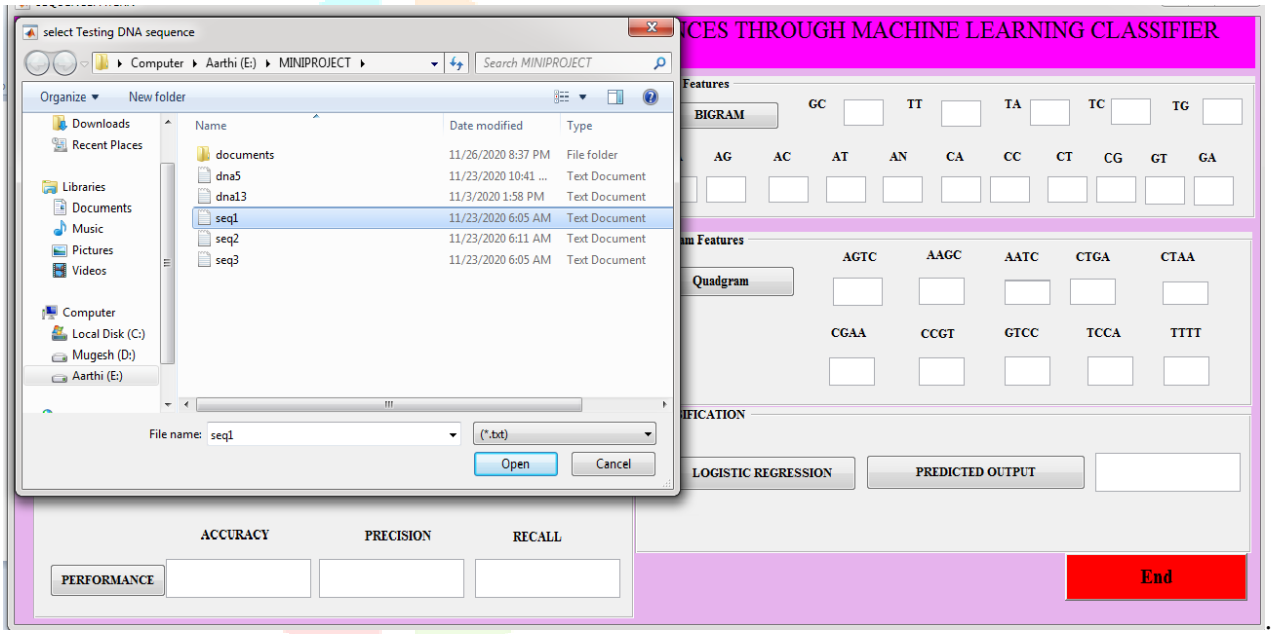
}

Prototype of gradient descent

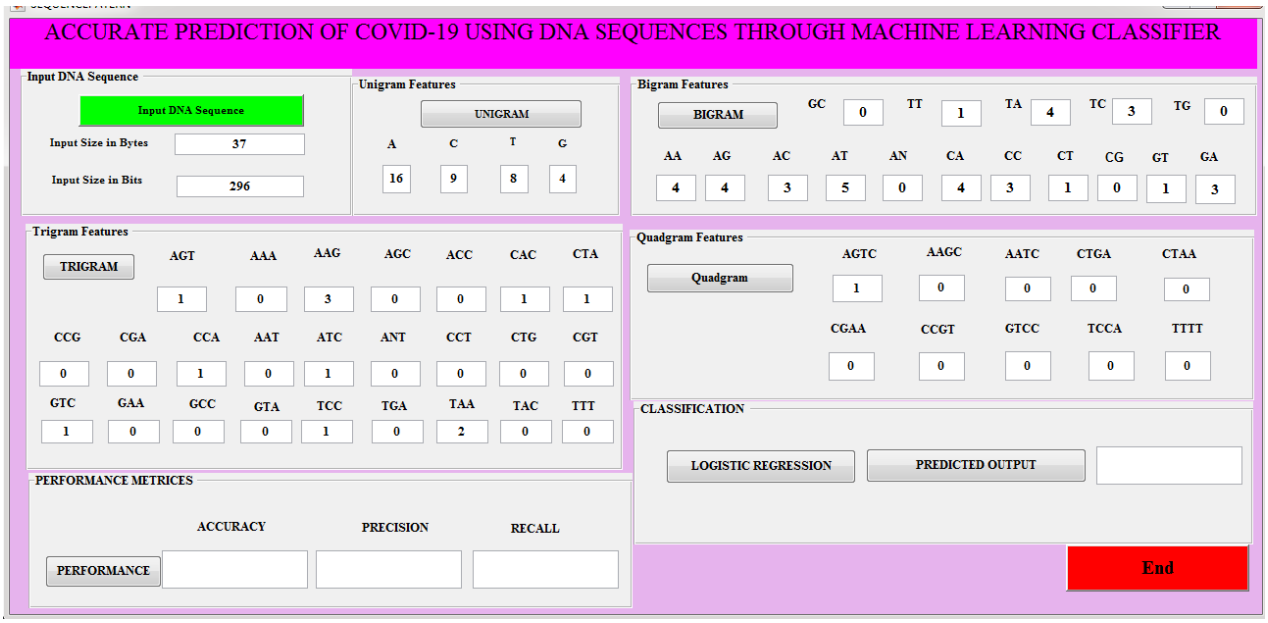
Notice that the constant α is usually called the *learning rate* or the *search step* and that it has

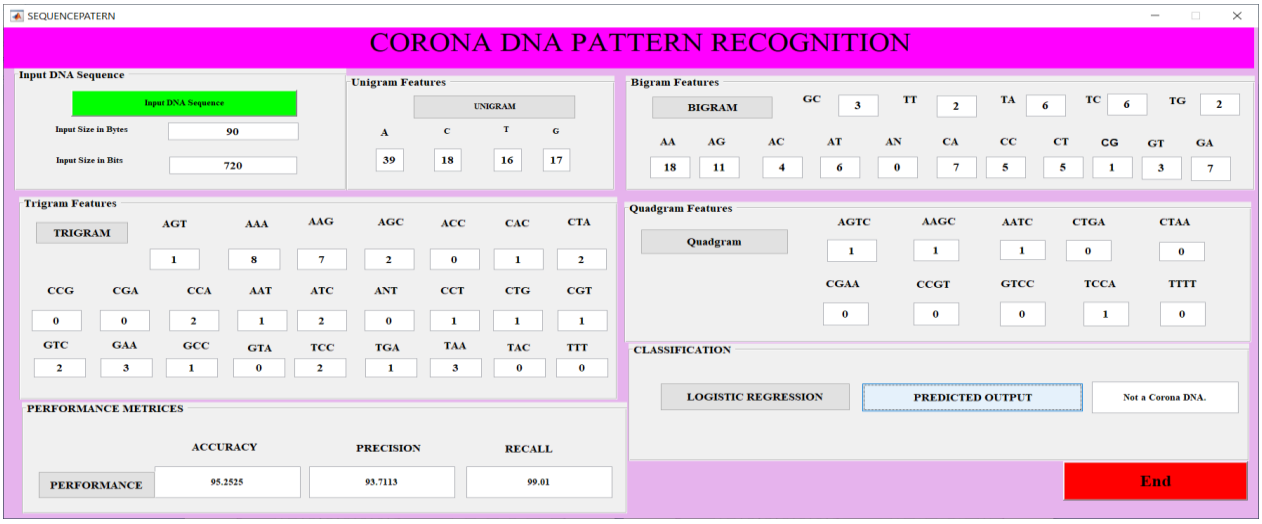
to be carefully tuned to reach convergence.

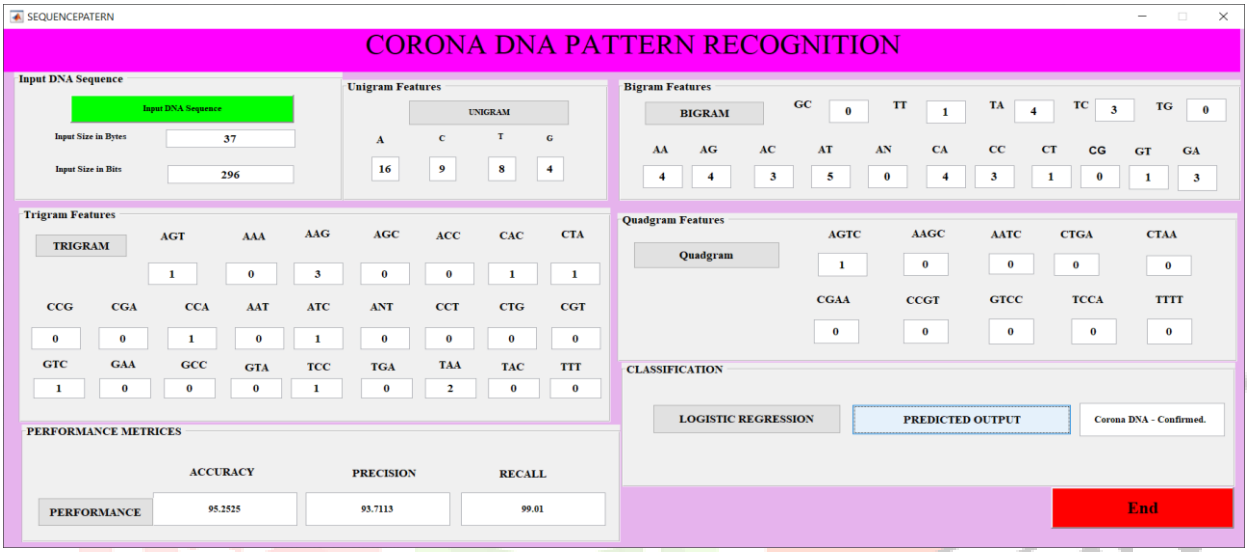
4 RESULT



The above image shows the dataset file selection by the user







5. CONCLUSION

In this work, we have presented the use of machine learning classifiers for the effective classification of COVID-19. The proposed methodology is trained on two publicly available datasets and has outperformed across all the classes. Machine learning algorithm is applied for final classification leading to the best result obtained by Logistic Regression with the Accuracy, Sensitivity, Specificity of 0:973, 0:974, 0:986. Therefore, this approach of using Corona DNA sequence and computer-aided diagnosis can be used as a massive, faster and cost-effective way of screening. Also, it brings down the time for testing drastically. To make a clinically effective prediction of COVID-19, training with more massive datasets and testing in the field with a larger cohort can be immensely useful.

6. REFERENCE

1. Cross chromosomal similarity for DNA sequence compression , Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, published July 14, 2008
2. Analysis of cross sequence similarities for DNA multiple sequence compression, Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, June, 2010
3. A Simple and Fast DNA Compressor, G. Manzini and M. Rastero 2004
4. Universal Intelligent Data Compression Systems: A Review, Ahmed Kattan, 2010
5. Pattern recognition Using Genetic Algorithm, Majida Ali Abed, Ahmad Nasser Ismail and Zubadi Matiz Hazi, 2010.
6. A new challenge for compression algorithms: Genetic Sequences, S. Grumbach and F. Tahi, 1994
7. A universal algorithm for sequential data Compression, J. Ziv and A. Lempel, 1977
8. Universal data compression algorithm based on approximate string matching, I. Sadeh, 1996
9. A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison, X. Chen, S. Kwong and M. Li 1999
10. DNA Compression Challenge Revisited, B. Behzadi and F. Le Fessant, June 2005.