CONSTRUCTION OF AN OPTIMIZED MULTI-STAGE HIGH-THROUGHPUT VIRTUAL SCREENING PIPELINE FOR LONG NON-CODING RNA'S

A Thesis

by

MANASA GADIYARAM

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2022

Major Subject: Electrical Engineering

ABSTRACT

Long non-coding RNA's(lncRNA's) are a type of RNA transcripts with a length of more than 200 nucleotides which cannot be translated into proteins. The study of lncRNAs is extremely important since it has been discovered that a wide range of biological processes are affected by them, such as epigenetic regulation, metabolic processes, chromosome dynamics and cell differentiation.This work investigates the classification of lncRNA's from protein coding transcripts(PCT's) using a multi-stage high throughput virtual screening(HTVS) pipelne. Each stage of the pipeline is a support vector machine(SVM) model.

Various features associated with RNA's in general have been calculated. These features are divided into three groups- sequence based, secondary structure based and physicochemical property based. These features were first calculated and analysed in a method called LncFinder. Support vector machines have been trained on these features on the basis of complexity and time taken for calculation. Support-vector machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

These SVM's have then been arranged on an HTVS pipeline as different stages of the pipeline. The pipeline has been optimized using an optimization framework, for determining the screening thresholds of each stage of the HTS pipeline. The final number of lncRNA's obtained can then be further used for drug discovery purposes.This multi-stage classification process significantly reduces the effective selection cost per potential candidate and make the HTS pipelines less sensitive to their structural variations.

DEDICATION

To my parents, grandparents, friends and research group members.

ACKNOWLEDGMENTS

I am extremely thankful to my advisor Dr. Byung-Jun Yoon for guiding me through out the project and for trusting me with working on cutting edge technology. Under his guidance, I had the opportunity to explore different areas of my major which I never knew existed.

I am also thankful to my committee members- Dr. Xiaoning Qian, Dr. Krishna Narayanan and Dr. Arul Jayaraman for their quick responses and help through out the course of this project. Their timely responses made the completion of this project smoother.

My research group members, especially Narges Zarnaghinaghsh and Hyun-Myung Woo have constantly guided me and helped me use resources at High Performance Research Computing(HPRC) and take knowledge from parts of their own research work.

Finally, I would like to thank Katie Bryan from the Electrical and Computer Engineering Graduate Office for quickly replying to my emails and making sure that the administration process for completing my thesis was done properly.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Byung-Jun Yoon [advisor], Dr. Xiaoning Qian and Dr. Krishna Narayanan of Department of Electrical and Computer Engineering and Dr. Arul Jayaraman of the Department of Chemical Engineering.

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing (HPRC).

All the work conducted for the thesis was completed by the student independently

Funding Sources

No other outside source of funding was provided.

NOMENCLATURE

A	Adenine
b	Bulge
C	Cytosine
CPAT	Coding Potential Assessment Tool
CPC	Coding Potential Calculator
EIIP	Electron-ion interaction pseudo-potential
FFT	Fast Fourier Transform
G	Guanine
h	Hairpin
HPRC	High Performance Research Computing
HTS	High Throughput Screening
HTVS	High Throughput Virtual Screening Pipeline
l	Loop
LncRNA	Long Non- coding Ribo-nucleic acid
ORF	Open Reading Frame
PLEK	Predictor of Long non-coding RNAs and messenger RNAs based on an improved k-mer scheme
P-U	Paired-Unpaired sequence
s	Stem
SNR	Signal to Noise Ratio
SS	Secondary Structure
SSE	Secondary Structure Elements

SVM	Support Vector Machine
TAMU	Texas A & M University
U	Uracil

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

Long non-coding RNAs (lncRNAs), a form of transcripts which are longer than 200 nucleotides and not able to encode proteins within the intracellular space, has been the focus of research in the past several years. Several research suggest that more than 80 percent of the human genome has biochemical functions, while less than 2 percent of the genome may be translated into proteins. Furthermore, as much as 70 percent of the non-coding sequences are transcribed into lncRNAs. All those figures endorse that lncRNA's include masses of information expecting our exploration.

Lots of proof [8] have indicated that lncRNAs play an influential role in numerous complicated human diseases consisting of lung cancer, Alzheimer diseases and cardiovascular diseases. Databases like LncRNADisease and Lnc2Cancer have gathered heaps of experimentally proven relations among lncRNAs and diseases, that have additionally showed the intimate connections among lncRNAs and diseases.This makes lncRNA identification a fundamental step of lncRNA research. Many methods and tools have been developed using machine learning techniques like Coding Potential Calculator(CPC), Coding Potential Assessment Tool(CPAT), Predictor of long-noncoding RNAs and messenger RNAs based on an improved k-mer scheme(PLEK) etc.

All of the above mentioned tools and methods heavily rely on features derived from the sequence. The features used by them include Open reading frame(ORF) length, ORF coverage, sequence length, k-mer frequencies, count of stop codon and many more.The essence of these kinds of features is to evaluate the differences in intrinsic composition between lncRNAs and mRNAs or PCT's[9]. The problem with sequence based features is that the sequence compositions varies from species to species, and thus these methods provide very unstable performances on different species.

Among the contemporary methods, [10] LncFinder is a tool that shines the most. It not only uses sequence based features but also incorporates secondary structure based features and physicochemical based features. This inherently makes it usable to different types of species as well as less prone to variations in sequence data.

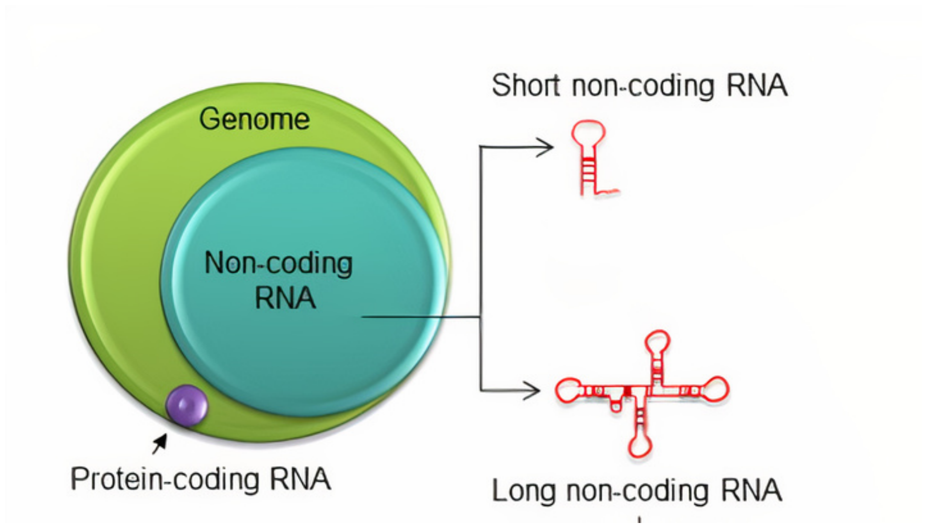


Figure 1.1: LncRNAs and PCTs in total Genome.[1]

Compared with existing lncRNA identification tools, LncFinder has the following merits:

1. LncFinder utilizes features from three different categories: intrinsic composition of sequence, multi-scale structural information and physicochemical property based on EIIP and fast Fourier transform (FFT).

2. The machine learning model[11] used by LncFinder is SVM. This classifier is not only simple to implement but also gives high accuracy.

These merits makes LncFinder an ideal candidate to calculate the various features. Our work performs part of what LncFinder does- we calculate the final features used by that tool and train multiple SVM models [8].

HTS pipelines[7] consist of various stages, each of which is associated with either an experimental or computational platform. These platforms filter the samples from different perspectives from other platforms. A previous stage evaluates the samples through the associated an experimentally or computationally efficient platform and passes only the samples whose score exceeds a certain threshold to the next stage associated with a experimentally or computationally expensive platform for a more sophisticated evaluation. In this way, HTS pipelines help to obtain desirable candidates through a series of selection processes. The final candidates obtained after the last stage can then further be used to perform sophisticated research. In our case, potential lncRNAs obtained after the last stage can be used for drug discovery processes or other research involving detection of cancer. Since HTS pipelines give such valuable information, they tend to be extensively used as the first stage of any research process.

Thus constructing an HTVS pipeline can narrow down the search space from infinite number of samples to desired number of candidates. This makes it important to determine how different stages are placed after one another. The most optimal way of placing the multiple stages will give maximum number of desirable candidates. This optimization can help to determine the optimum threshold of each stage for the candidates to pass through. Another thing to consider is also the number of stages needed- this can depend on the problem statement we are dealing with and the features that we can extract from our dataset[2].

Combining LncFinder method and the HTVS pipeline is the essence of our project. LncFinder trains its SVM model after going through a series of feature selection processes. We take that knowledge of final features calculated from them and train four SVM models. These models are all trained with features with different level of complexity of calculation. These models are then placed in the HTVS pipeline as stages and optimization is performed where a hard budget constraint is selected and classification is done such that that budget is respected. The final stage of the pipeline contains all the features thus making it the most expensive as well as most accurate stage.

1.1 Our Approach

This project focuses on performing classification of Long non-coding RNA's from Protein coding transcripts using a multi-stage high throughput pipeline. Each stage of the pipeline is a machine learning model. For this project, we have chosen a simple model named Support Vector Machine(SVM). SVM's are supervised learning models that take a set of training examples, each marked as belonging to one of two categories. The SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. SVM[12] maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In this project, we have trained four SVM models with different features extracted from the lncRNA and PCT sequences. These sequences were taken from GENCODE dataset. The features are grouped into three types- sequence based features, physicochemical features and secondary structure based features[5]. The sequence based features were calculated by taking the Open Reading Frame(ORF) of each sequence and then performing further calculations on that. The sequence based features taken are- ORF length, ORF coverage, Logarithmic distance of ORF's w.r.t lncRNA's, Logarithmic distance of ORF's w.r.t PCT's and total Logarithmic distance. The physicochemical features taken were- Quantile statistics (Q1, Q2, maximum value and minimum value), signal to noise ratio, signal at N/3 position. For the secondary structure features, we first calculated the secondary structure of each sequence and derived three sequences from each structure- acguD sequence, acguACGU sequence and PU sequence. The secondary structure features taken from them were- Logarithmic distance of acguD w.r.t lncRNA's, Logarithmic distance of acguD w.r.t PCT's and total Logarithmic distance of acguD sequences, Logarithmic distance of acguACGU w.r.t lncRNA's, Logarithmic distance of acguACGU w.r.t PCT's and total Logarithmic distance of acguACGU sequences, Minimum Free Energy(MFE) of sequences and Paired-Unpaired(PU) ratio.

Each SVM model was trained by selecting features based on the level of complexity and time taken to calculate. The features which were easily calculable were used to train a model and placed at the initial stages of the pipeline. The final stage of the pipeline was trained on all the features we calculated, which meant that it included the features which took the most time to calculate. This final stage also gave the most accurate results for classification. Keeping the final stage fixed, the other three stages were optimized using an optimization framework to decide the order at which the stages should be placed. The optimized order gave the best classification results with maximum number of desirable candidates at the end of the fourth stage.

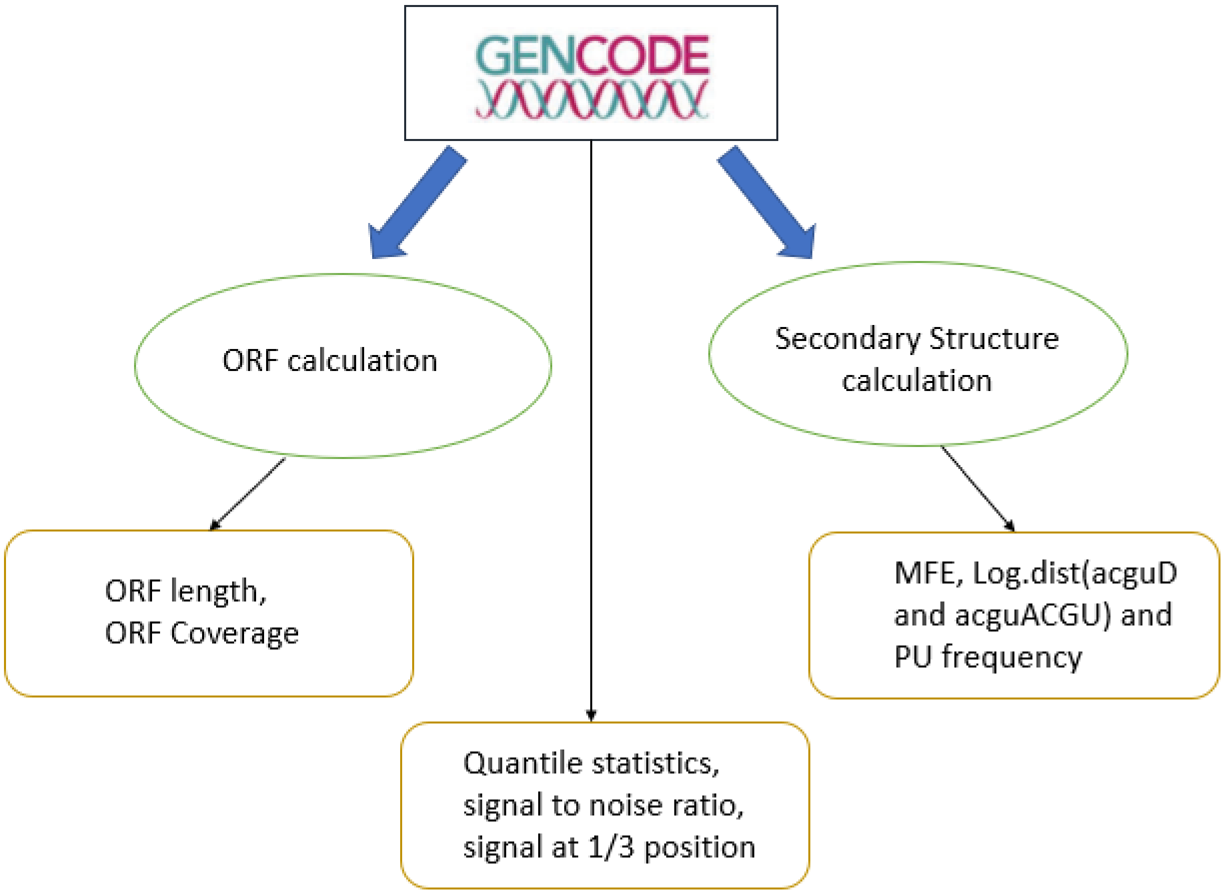


Figure 1.2: Overview of feature calculation

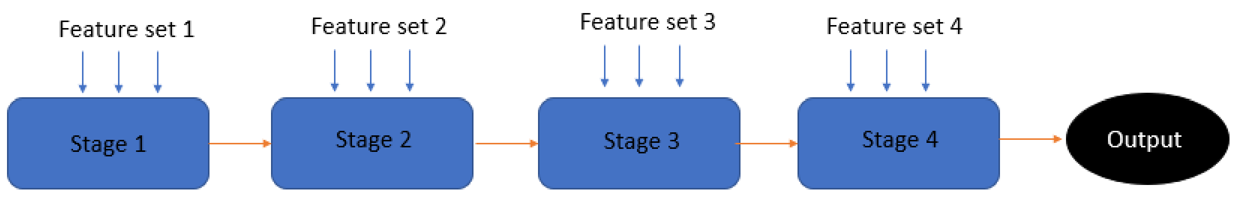


Figure 1.3: Overview of HTVS pipeline

Figure 1.2 shows the overview of how the final features are calculated from the GENCODE dataset. The final calculated features are then grouped according to complexity of calculation and time taken for calculation. Figure 1.3 shows each stage being trained with different set of features. The output after stage 4 gives the final candidates of lncRNAs.

2. LITERATURE SURVEY

2.1 Understanding different types of RNA sequences

Any RNA sequence can be defined in terms of its intrinsic properties. These properties, when quantified can be used to derive features to uniquely define each sequence. A typical RNA sequence looks like the following-

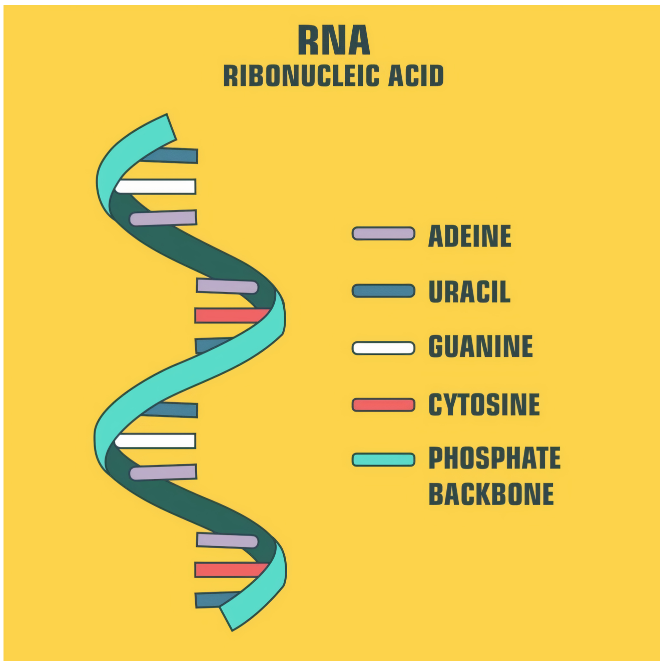


Figure 2.1: Typical RNA strand[2]

Some RNA strands can get translated to proteins. Such RNA's are called as Protein Coding Transcripts(PCTs). These amino acids or proteins are further helpful for various bodily functions and reactions which are needed for the survival of the living organisms. Following is the process followed by PCTs to get proteins or amino acids-

DNA gets transcribed to RNAs which further get translated to amino acids. But, this does not happen to all RNAs. There are different types of RNAs like- short non-coding RNAs, long

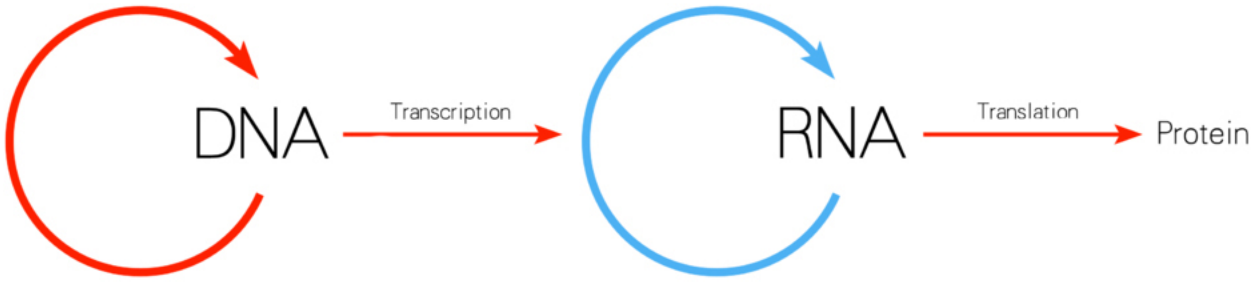


Figure 2.2: Transcription and Translation process[3]

non-coding RNAs, messenger RNAs etc. A major portion of these various types of RNAs do not get translated to proteins. These types of RNAs are required to be studied in depth since they participate in various bodily reactions as well as have shown to have a major impact in various autoimmune diseases. In this study,we focus on long non-coding RNAs(lncRNAs). There are various markers using which an lncRNA can be identified. These markers, as explained above, can be used as features. This study focuses on three types of such features- sequence based, secondary structure based and physicochemical based features. All these features are explained in detail in the following sections.

2.2 Features for RNA sequences

LncRNAs and PCTs can be defined by many different types of features. In this study, we focused our research to only three such types.They are based on a method named LncFinder. Many studies have also used these features since they give the most information about each sequence which is essential for a meaningful classification. Both types of RNAs are found in almost all living organisms like humans, cats, zebras, wheat[2] etc.

2.2.1 Sequence based features

Many previous studies have shown lncRNAs and PCTs have a different distribution of adjoining bases. The bases in RNAs are Adenine(A), Uracil(U), Guanine(G) and Cytosine(C). The difference in their distribution helps in differentiating them. Following are the various sequence based features-

2.2.1.1 Open Reading Frame Length

An Open Reading Frame(ORF)[12] is a part or a length of a DNA or RNA sequence which starts at a start codon and ends at a stop codon. The start codon, in terms of RNA, can be defined as the nucleotide group "AUG". The stop codon refers to the groups "UAA", "UAG" and "UGA". The length between the start and stop codon can be used to say that a translation can happen. The translation is not guaranteed but the ORF gives a sub-sequence where it can happen, if at all it happens. The length of that ORF can be very helpful to calculate. This is because, lncRNAs are non protein coding- so they tend to have smaller ORF lengths while protein coding transcripts have longer lengths. An interesting thing to observe is the length of the ORF can be divided by three. This is because any algorithm which tries to find the start and stop codons in a sequence, searches for it by taking three nucleotides as a group. Thus, the length is always divided by three. A sequence can have multiple ORF's. Usually, The ORF with the maximum length has the highest possibility of getting translated into proteins. The following figure shows ORF in an example sequence-

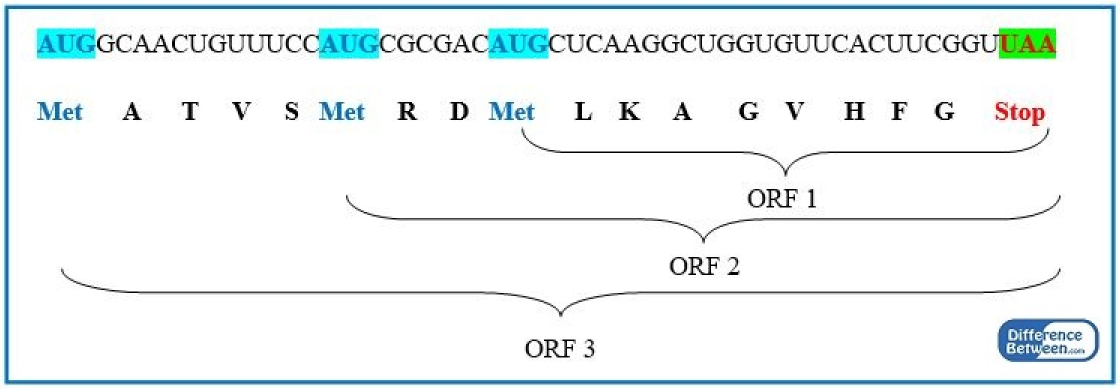

AUGGCAACUGUUUCCAUGCGCGACAUGCUCAAGGCUGGUGUUCACUUCGGUUAA
Met A T V S Met R D Met L K A G V H F G Stop
ORF 1
ORF 2
ORF 3
Difference Between.com

Figure 2.3: Example of searching ORF's[4]

2.2.1.2 Open Reading Frame Coverage

Open Reading Frame Coverage is nothing but the ratio of the length of a particular ORF with the total length of the sequence. Along with ORF length, the coverage gives a more comprehensive view of an ORF. It becomes substantially easier to compare coverage of different ORFs of a sequence than the actual length of an ORF when we want to select the maximum length ORF. Also, when comparing between ORFs of different sequences, the coverage tends to give a better understanding as each sequence is of different lengths and thus a ratio provides a better view.

2.2.1.3 Logarithmic Distance

Logarithmic distance is more of a scheme than a feature in this case. This scheme further gives three more features. In this scheme, each sequence, be it lncRNA or PCT is compared to a bank of known lncRNA and PCT sequences. Essentially, the distance between each sequence and a known database of lncRNAs and PCTs is calculated. This gives us an estimate of how close it is to either type of RNAs. The smaller the number, the more likely it is to be of that type. There are many databases than provide such banks-GENCODE is a prime example. In this project too, we have used data from GENCODE. Other databases include University of California, Irvine database and ENSEMBL which provide data for human, mouse, zebrafish, wheat and many more[2]. Finally, the ratio between the two distances is calculated to give the final Logarithmic distance[13]. Thus, we get three features from this scheme. The formula to calculate the Logarithmic distance is given as-

$$logdist.LNC = \frac{1}{n} \sum ln \frac{freq.seq(i)}{freq.lnc(i)}, i = 1, 2, 3...4^k \tag{2.1}$$

$$logdist.PCT = \frac{1}{n} \sum ln \frac{freq.seq(i)}{freq.pct(i)}, i = 1, 2, 3...4^k \tag{2.2}$$

$$logdist.Ratio = \frac{logdist.LNC}{logdist.PCT} \tag{2.3}$$

The numerator in Eq. (2.1) and Eq. (2.2) gives the frequencies of each type of k-mer encountered in the sequence. The sequence can either be the complete RNA sequence or the maximum length ORF of an RNA sequence. Similarly in the both the equations, the denominator is the frequencies of each type of k-mer for the sequences in each bank or database. The summation is done for all possible combinations in a k-mer for 4 nucleotides. There have been many studies, exploring different types of k-mers. Essentially the value of "k" can be anything- ranging from a 4 to the complete length of the sequence. But, after extensive research, these studies have shown that optimum results are obtained for k=6. Thus, we have taken k as 6 in our study too. Thus, the above three equations give three different features for each sequence.

2.2.1.4 Other sequence based features

There have been a huge number of studies which researched on the different types of features which can be extracted from the sequence of an RNA. This is because, one single sequence tends to provide a plethora of information, thus enhancing our understanding of RNA sequences. The most popular feature to be calculated is the length of the entire sequence.

Another feature that is usually calculated is the "G+C" content. It is nothing but the percentage of bases in an RNA strand which are either Guanine(G) or Cytosine(C). Another important feature that can be calculated is the codon bias. Codon usage bias refers to the phenomenon where specific codons are used more often than other synonymous codons during translation of genes, the extent of which varies within and among species. Each codon is a group of three nucleotides[14]

Hexamer score is also a popular feature to be calculated in many studies.It refers to the relative degree of hexamer usage bias in a particular sequence. A positive value would indicate a coding region while a negative value would indicate a non-coding region.

Finally, just like the Logarithmic distance, there exists a Euclidean distance. It is calculated in a similar way, where each sequence is compared to a bank of lncRNAs and PCTs and finally a ratio is taken. The only difference being the formula to calculate it. The formulae are given as-

$$Eucdist.LNC = \sqrt{\sum (freq.seq(i) - freq.lnc(i))^2}, i = 1, 2, 3...4^k \tag{2.4}$$

$$Eucdist.PCT = \sqrt{\sum (freq.seq(i) - freq.pct(i))^2}, i = 1, 2, 3...4^k \tag{2.5}$$

$$Eucdist.Ratio = \frac{Eucdist.LNC}{Eucdist.PCT} \tag{2.6}$$

Thus, just like the logarithmic distance, the above scheme also gives three separate features using the three formulae.

2.2.2 Secondary Structure based features

Secondary structure plays an important role in some biological functions and is thought to be more conserved than the primary sequence. However, structural information is rarely used to predict lncRNA. To examine the characteristics of this category, here we present a multi-scale secondary structure that represents structural information of RNA sequences at three levels: stability, secondary structural elements (SSE) combined with pairing conditions, and structural nucleotide sequences. The following sections explain different secondary structure based features-

2.2.2.1 Low Scale Features

Minimum Free Energy (MFE)[13] is a basic structural overview that shows the stability of RNA structure. In this case, it can be termed as a low-scale feature. There are few unstable lncRNAs, but on average lncRNAs are less stable than mRNAs. Many studies have used this as a leverage for classification purposes. the following boxplot clears the picture even better. The boxplot was calculated using the secondary structure of the GENCODE data for lncRNAs. It gives a rough estimation on how lncRNAs are different from mRNAs.

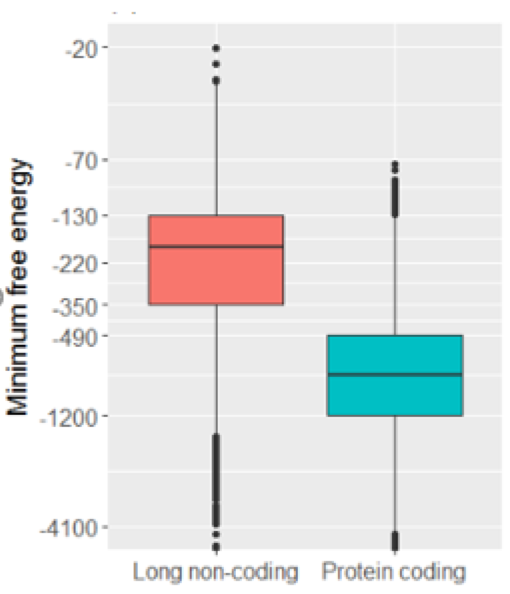


Figure 2.4: Boxplot for comparing MFE values[5]

2.2.2.2 Medium Scale Features

For this type of features, the secondary structure of the sequence is derived. From this structure, we further calculate multi-scale structures[13]. To get a complete understanding, we take seq[n] as an RNA sequence of length N, and the nucleotide to be denoted as lowercase a,c,g,u.

Let SS[n] be the secondary structure sequence of seq[n], with SS[n] being defined using a dot-bracket notation SS[n]($\in$ {·.·, ·(·, ·)·}).

To depict the basic components of an RNA sequence, following SSEs are taken- stem(s), bulge(b), loop(l) and hairpin(h). For these Secondary structure elements, their frequency of occurrence is calculated.

Further, the dot bracket notation is used to derive the Paired-Unpaired sequence where the following is used-

$$\text{Paired- Unpaired Seq[n]} = \begin{cases} U, & \text{if } SS[n] = \cdot \\ P, & \text{if } SS[n] \neq \cdot \end{cases}$$

Replacing nucleotides of seq[n] with corresponding SSEs, the first secondary structure-derived sequence—SSE full sequence (SSE.Full Seq) is obtained. Taking the continuous identical SSE as single SSE, another sequence—SSE abbreviated sequence (SSE.Abbr Seq)—is obtained. Together the P-U sequence, SSE Full sequence and SSE-Abbr sequence form the medium scale features.The medium scale features can be summarized in the following figure-

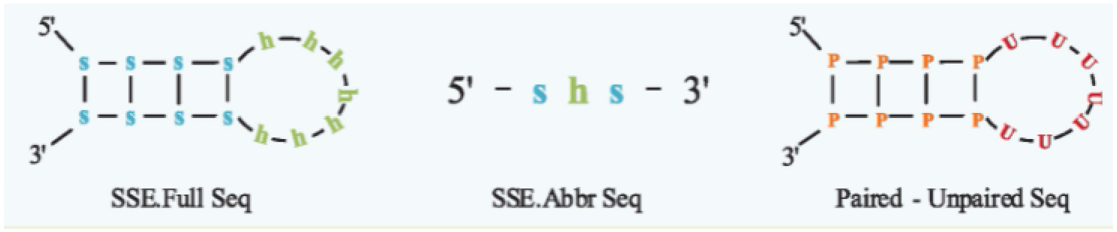


Figure 2.5: Medium Scale Features[6]

2.2.2.3 High Scale Features

Using the definition of P-U sequence, we can further calculate three more secondary structure derived sequences- acguD, acguS and acgu-ACGU[2]. The "D" stands for Dot and the "S" stands for Stem. The following formulae are used to get the desired sequences-

$$\text{acguD Seq[n]} = \begin{cases} D, & \text{if } SS[n] = . \\ seq[n], & \text{if } SS[n] \neq . \end{cases}$$

$$\text{acguS Seq[n]} = \begin{cases} seq[n], & \text{if } SS[n] = . \\ S, & \text{if } SS[n] \neq . \end{cases}$$

$$\text{acgu-ACGU Seq[n]} = \begin{cases} A, & \text{if } seq[n] = a \bigwedge SS[n] \neq . \\ C, & \text{if } seq[n] = c \bigwedge SS[n] \neq . \\ G, & \text{if } seq[n] = g \bigwedge SS[n] \neq . \\ U, & \text{if } seq[n] = u \bigwedge SS[n] \neq . \\ seq[n], & \text{if } SS[n] = . \end{cases}$$

From the secondary sequences above, further calculations can be done. For example- k-mer frequencies for each sequence can be calculated, euclidean and logarithmic distances can be calculated. The euclidean and logarithmic distances can be calculated by finding the secondary structures of the sequences and comparing it to the secondary structure sequences of the lncRNAs and PCTs from the database bank. The high scale features can be summarized in the following figure-

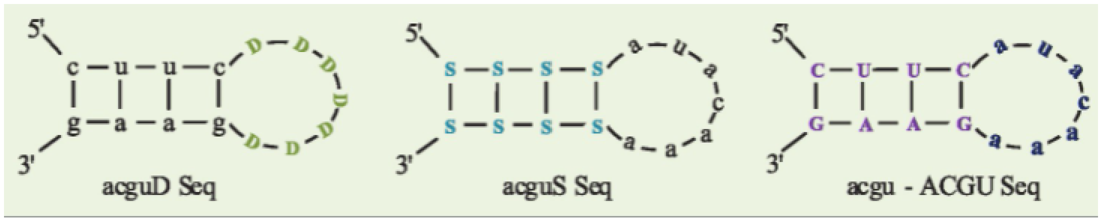


Figure 2.6: High Scale Features[6]

2.2.3 Electron-Ion Interaction Pseudopotential based features

The physicochemical or the psuedopotential(EIIP) properties of a sequence also give valuable information. EIIP was initially used to locate exons. Each nucleotide has its own EIIP value[5]. EIIP values denote the energy of the delocalized electrons of a nucleotide. For any RNA sequence, the EIIP values are as follows-

Nucleotide	EIIP Value
a	0.1260
c	0.1340
g	0.0806
u	0.1335

Table 2.1: EIIP values for each nucleotide.

Using the EIIP values, we can substitute each nucleotide with its EIIP value. By doing so, we obtain a unique EIIP sequence for each sequence. An example of an EIIP sequence is as follows-

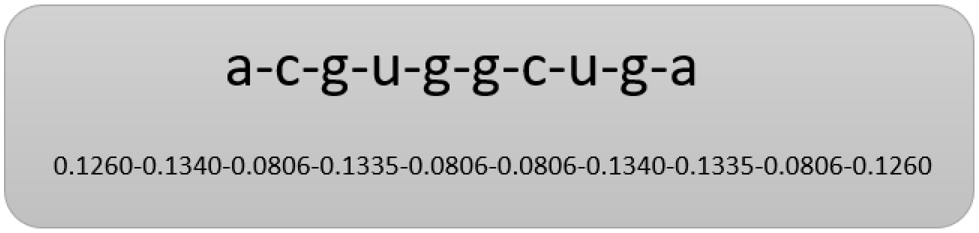


Figure 2.7: EIIP sequence on an example RNA sequence

Since now we have a numerical sequence for each RNA sequence, we can perform certain calculations to get features. The features explained in further sections-

2.2.3.1 Power Spectrum

Every EIIP indicator sequence helps to generate its respective power spectrum. Let the EIIP sequence be denoted by $X_e[n]$.

Applying FFT on this sequence, we get-

$$X_e[k] = \sum_{n=0}^{N-1} X_e[n] \exp(-j\frac{2\pi kn}{N}), \tag{2.7}$$

We can then calculate the corresponding power spectrum-

$$S_e[k] = |X_e[k]|^2 \tag{2.8}$$

This power spectrum can further be used to calculate average power. The formula for average power is as follows-

$$\overline{E} = \frac{\sum_{k=0}^{N-1} S_e[k]}{N} \tag{2.9}$$

This can further be used to locate the signal at $\frac{1}{3}$ position and calculate the signal to noise ratio. The signal at $\frac{1}{3}$ position is denoted as $S_e[\frac{N}{3}]$. The formula for Signal to Noise ratio (SNR) is-

$$SNR = \frac{S_e[\frac{N}{3}]}{\overline{E}} \tag{2.10}$$

It has been observed that PCTs have a higher value of signal at 1/3 position, signal to noise ratio and average power as compared to lncRNAs. Previous studies have extensively used these factors for effective classification.

2.2.3.2 Quantile statistics

For every sequence of numbers in the power spectrum, the quantile statistics can be calculated. They are the values at the 25th percentile, 50th percentile and 75th percentile. They are denoted as Q1, Q2 and Q3 respectively. The maximum and minimum values of the sequence are also considered. Many studies[9] have taken the top 10 values or the top 10% of the entire sequence and then calculate the quantile statistics. This helps to get a comprehensive view of the sequence as we mostly care about the values which have a high impact on defining the sequence. Because signals from mRNA are generally stronger than signals from lncRNA, protein-encoding transcripts tend to have higher quantile statistics than lncRNA. EIIP-based features consider the physicochemical as well as 3-base periodicity properties of protein-coding sequences, and it has shown to present robust results on many non-model data sets. To get a complete understanding on how EIIP features can be effective in classification of lncRNAs and PCTs, the following boxplots are presented. These boxplots are were generated on data downloaded from GENCODE to help give a rough estimate-

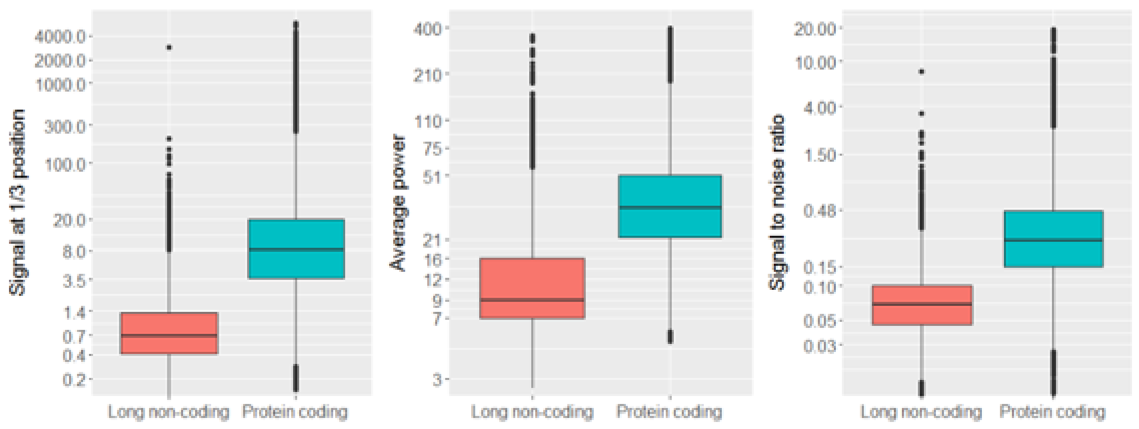


Figure 2.8: Boxplots for Signal at 1/3 position, Average Power and Signal to Noise Ratio[2]

2.3 High Throughput Screening Pipelines

A High Throughput Screening Pipeline(HTS) is a structure which has multiple stages, where each stage filters the input with respect to some inherent conditions of that stage. The next stage receives the data from the previous stage to be further filtered and finally after N number of stages, the output is taken. This output is considered to be the purest with the most number of probable candidates which can further be used for different types of experiments. An extremely simple version of an HTS is shown in the following figure-

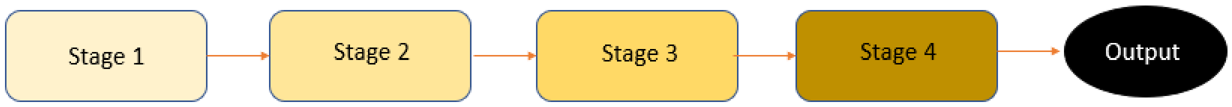


Figure 2.9: A simplified High throughput Screening Pipeline

Usually, as the stages progress, the complexity of each stage increases, thus the conditions to be satisfied also become stringent. A better outlook would be to get a mathematical understanding of stages.

2.3.1 Mathematical representation of HTS pipeline

Mathematically, The HTS pipeline can be summarized in the following diagram-

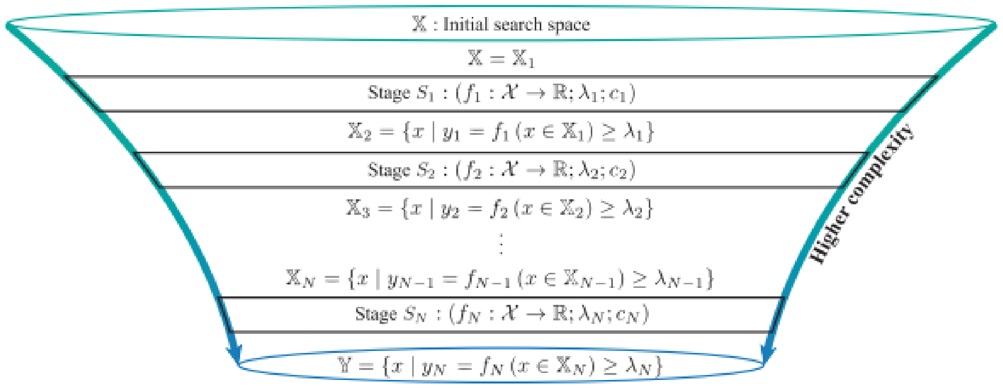

$\mathbb{X}$: Initial search space
$\mathbb{X} = \mathbb{X}_1$
Stage $S_1 : (f_1 : \mathcal{X} \to \mathbb{R}; \lambda_1; c_1)$
$\mathbb{X}_2 = \{x \mid y_1 = f_1(x \in \mathbb{X}_1) \geq \lambda_1\}$
Stage $S_2 : (f_2 : \mathcal{X} \to \mathbb{R}; \lambda_2; c_2)$
$\mathbb{X}_3 = \{x \mid y_2 = f_2(x \in \mathbb{X}_2) \geq \lambda_2\}$
$\vdots$
$\mathbb{X}_N = \{x \mid y_{N-1} = f_{N-1}(x \in \mathbb{X}_{N-1}) \geq \lambda_{N-1}\}$
Stage $S_N : (f_N : \mathcal{X} \to \mathbb{R}; \lambda_N; c_N)$
$\mathbb{Y} = \{x \mid y_N = f_N(x \in \mathbb{X}_N) \geq \lambda_N\}$
Higher complexity

Figure 2.10: Mathematical representation of HTS pipeline[7]

Let X be the initial search space or the input data. Let c_i be the cost per sample to evaluate at each stage S_i.

1. The first stage S_1 contains a function f_1, where $X = X_1$ is taken as the input. A threshold λ_1 is also taken. Then after performing f_1 at a cost c_1 per sample, we get a set of output named X_2. This X_2 satisfies the condition that $x \in X_1 \geq \lambda_1$.

2. The second stage has the function f_2 with its own threshold value λ_2. Again the function f_2 is applied at cost c_2 on input X_2 to get the output X_3.

3. X_3 forms the input for the next stage. This continues until the last stage is reached.

4. The last stage has the threshold as λ_N. This value is always fixed and decided by experts or some previous calculations on the input data.

The cost per sample is the amount of time taken to compute or process a sample at a particular stage. Thus, the cost per sample varies according to the input sequence and the function applied at each stage. In general, the cost c_i is proportional to the predictive accuracy. As the stages have been arranged on the basis of increased complexity, the cost per stage also increases as we reach the last stage.

The cost is determined from some previous calculations while determining the function of each stage. For HTS pipelines, the objective is to determine the threshold values λ_i of each stage.

The goal of the HTS pipeline is also to maximize the cardinality of the output or the potential candidate set Y i.e. $max|Y|$. We want maximum candidates so that we have a maximum sized pool for further research to take place[7].

Under the assumption that the pipeline S has been reasonably constructed, the scores of the samples X across all stages $S_1, S_2, S_3, ..., S_N$ is represented as a joint score distribution denoted as $f_S(y_1, y_2, y_3, ...y_N)$.

The reward function $r(\lambda)$ according to the lambda values $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]$ of the stages $S_1, S_2, ...S_N$ is as follows-

$$r(\lambda) = \int \cdots \int_{[\lambda_N ..., \lambda_1]}^{\infty} f_S(y_1, y_2, y_3, ...y_N)\, dy_1 \ldots dy_N \tag{2.11}$$

From above equation, it can be observed that $r(\lambda)$ is proportional to the number of potential samples through the pipeline S as the more the number of input samples, more will be the length of each y_i, thus increasing the number of potential samples.

2.3.2 Relation between Correlation of Stages and Potential Candidates

In a previous study[7], an analytical analysis was performed on a 4-stage HTVS pipeline. It was observed that if the correlation between the prior stages and the last stage was higher, then the potential candidates were detected much earlier, implying usage of less computational resources. With a lower correlation between stages, the candidates were detected only after a higher use of resources. The following graph summarizes it-

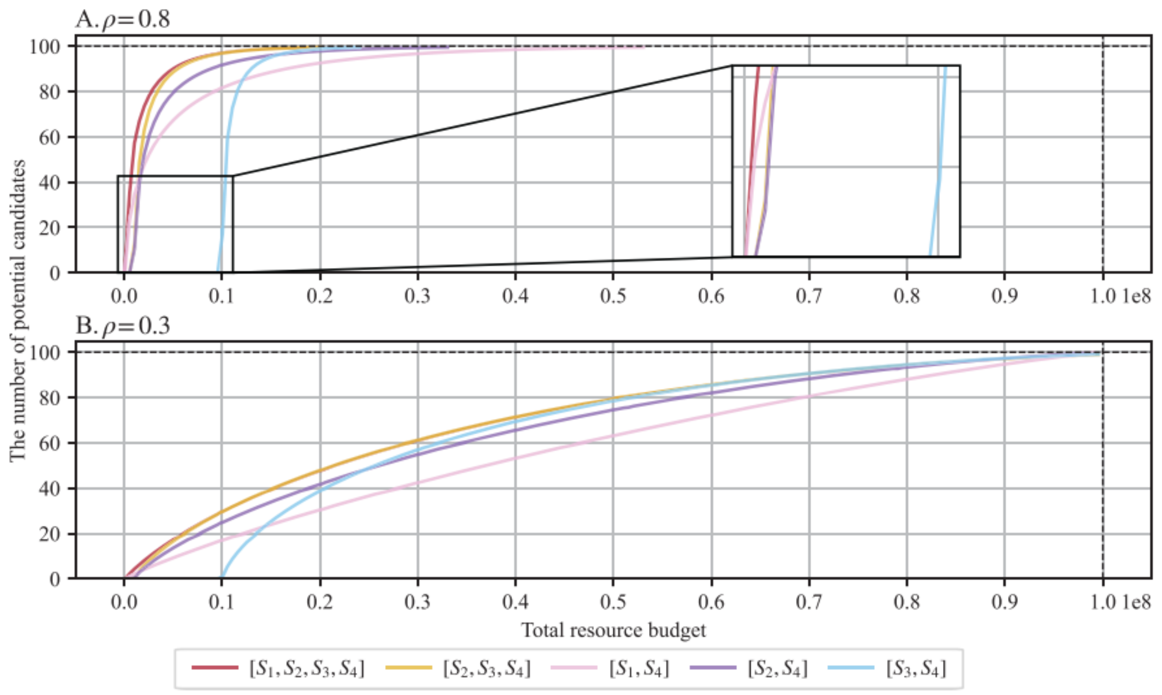


Figure 2.11: Candidates vs Computational cost w.r.t correlation[7]

2.3.3 Optimization of HTS pipeline under fixed computational budget

Let the total computational budget be C. Since the end goal is to maximize the number of potential candidates, which is in turn proportional to $r(\lambda)$, the logical thing would be to develop a constraint based on $r(\lambda)$. Let the optimal screening thresholds be denoted as $\psi^* = [\lambda_1^*, \lambda_2^*, ...\lambda_{N-1}^*]$ of prior stages S_i, where $i = 1, 2, ...N - 1$[7].

The optimum screening thresholds ψ^* can be found out by solving the following optimization problem-

$$\psi^* = \arg\max_{\psi}^{N-1} r([\psi, \lambda_N]) \tag{2.12}$$

$$\text{s.t.} \sum_{i=1}^{N} c_i |X_i| \leq C \tag{2.13}$$

here X_i is the number of samples that have passed the previous stages $S_1, S_2, ..., S_{i-1}$. It is defined as-

$$|X_i| = |X| \int \cdots \int_{[\lambda_{i-1} \ldots, \lambda_1]}^{\infty} f_{S1:i-1}(y_1, y_2, y_3, \ldots y_{i-1})\, dy_1 \ldots dy_{i-1} \tag{2.14}$$

2.3.4 Using HTS pipelines for other biological purposes

This project is focused on using a High Throughput Pipeline for screening or filtering purposes. But the usage of an HTS pipeline is not limited to that. A lot of studies have used HTS pipelines for virus detection. Many HTS pipelines[8] are much more complicates than a simple 4 stage structure. They may have various conditions such as-if one stage is not giving desired results or following a particular condition, then a different stage would be applied and the output would then be provided to the next stage. Such complex pipelines are very useful in RNA and DNA sequencing. A pipeline within a pipeline is also a viable HTS pipeline. More complex the problem, more intertwined stages will exist in the pipeline.

HTS pipelines are immensely used by various biotechnology companies for drug discovery too[3]. They act as a tool for running millions of biological or chemical tests in a much shorter time than physical experimentation. Currently, HTS pipelines are being used to develop vaccines for COVID-19.

2.4 Other Technologies and Methods

This project has used the features developed and used by an existing method named Lncfinder. But, the detection of long non-coding RNA has been a huge area of interest in the last decade. This has led to many different methods being developed.

One method lncADeep[14] uses the same sequence based features as the LncFinder but also calculates other features on the Longest Coding Sequence(LCS) of each sequence. It takes into consideration the Fickett-Nucleotide score as well as the HMMER Index and the EDP of LCS. Along with secondary structures, it also considers Hydrogen Bonding and Van der Walls properties. It had a very high accuracy score of 0.977 where it used the Deep Neural network (DNN) to identify lncRNAs.

Another method lncRNA-MDeep[10] incorporates one-hot encoding of transcript sequences along with the sequence based features. It uses a combined model of Deep Neural Network(DNN) and Convolutional Neural Network(CNN) to classify lncRNAs from PCTs. It has an accuracy of 0.9312.

A method named lncRNAnet[6] uses all the features of lncRNA-MDeep along with one hot encoding of the longest ORF region of each sequence. This comprehensive set of features is used to train a combination of a Convolutional Neural Network and Recurrent Neural Network(RNN) along with a Bucketing phenomenon in the initial stages for classification. The accuracy achieved is 0.812.

Our method differs from all the above since we are using a connected set of stages, each stage being an SVM model to get a pipeline which effectively generates maximum potential candidates of lncRNAs.

3. MATERIALS AND METHODS

3.1 Dataset

The dataset used in this project is the GENCODE-Human-lncRNA-Release38 dataset from GENCODE website[15]. It is a fasta file with over 40,000 Long non-coding RNA sequences with their annotation. For PCTs, GENCODE-Human-PCT-Release38 dataset from the same website is used. It is also a fasta file with over 100,000 PCT sequences along with their annotation. For the calculation of Log Distance, we required a standard set of lncRNAs and PCTs different from the above sequences which would act as the bank against which the Log Distance for each sequence would be calculated. So, from the GENCODE website, we took GENCODE-Human-lncRNA-Release20 dataset and GENCODE-Human-PCT-Release20 dataset for PCTs and lncRNAs respectively. They were named as lncRNA reference dataset and PCT reference dataset. The lncRNA reference dataset containe over 19,000 sequences while the PCT reference dataset contains over 89,000 sequences. For the purpose of this project, we have taken only a portion of sequences from each dataset. That has been summarized in the table below-

Dataset	Sequences available	Sequences Used
lncRNAs	48,752	4,062
PCTs	106,143	4,082
lncRNA reference	19,835	2,203
PCT reference	89,828	2,245

Table 3.1: Dataset Overview

3.2 Feature Calculation and Model Training

In this project, we have calculated the three groups of features- sequence based, EIIP based and secondary structure based features. Since we are following the findings of the method LncFinder, we have only calculated those features that they have used in their final model. After the calculation of features, we have divided them into four groups depending on the level of complexity and the time taken to calculate them. Based on those four groups, we trained four SVM models, all of them using a linear kernel.

Among the sequences used from the complete dataset of lncRNAs and PCTs-the total which amounts to 8144 sequences, 70% of these were used for training the SVM models and the remaining 30% was used for testing purposes. This 30% also acts as the input data for the HTVS pipeline optimization. The following table summarizes the distribution of training and testing data-

Sequence type	Training data	Testing data
lncRNAs	2,523	1,522
PCTs	3,165	917
Total	5,688	2,439

Table 3.2: Training and Testing data

The following sections explore the feature calculation and model training methods discussed above-

3.2.1 Model trained on EIIP features

3.2.1.1 EIIP features calculation

Electron-ion interaction pseudopotential (EIIP) gives the value of the energy of delocalized electrons of a nucleotide. So, each sequence has its own unique sequence of EIIP values using which the following features were calculated. The EIIP features are calculated as-

1. Calculate FFT of each numerical sequence per RNA sequence.

2. Calculate the Power Spectrum and ‘Average Power from FFT.

3. Calculate signal at $\frac{N}{3}$ position and Signal to noise ratio.

4. Arrange the Power Spectrum sequence in ascending order and select the top 10 values to ensure that we take the most impactful numbers which define the sequence.

5. Calculate the Quantile statistics of these top 10 values in the power spectrum of every sequence- Q1(25th percentile), Q2(50th percentile), maximum value and minimum value.

The following block diagram explains the feature calculation-

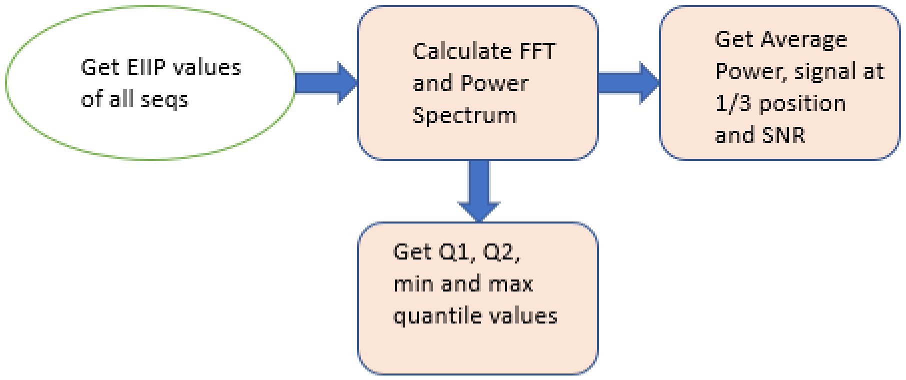

Get EIIP values of all seqs
Calculate FFT and Power Spectrum
Get Average Power, signal at 1/3 position and SNR
Get Q1, Q2, min and max quantile values

Figure 3.1: Block diagram for EIIP features calculation

The total time taken to calculate all the EIIP features for training and testing sequences is 20 seconds. Thus, the average time taken to calculate EIIP features for one sequence is 0.0024621445 seconds.

EIIP features showcase the difference between an lncRNA sequence and PCT sequence very profoundly. The following boxplots are a testament to that-

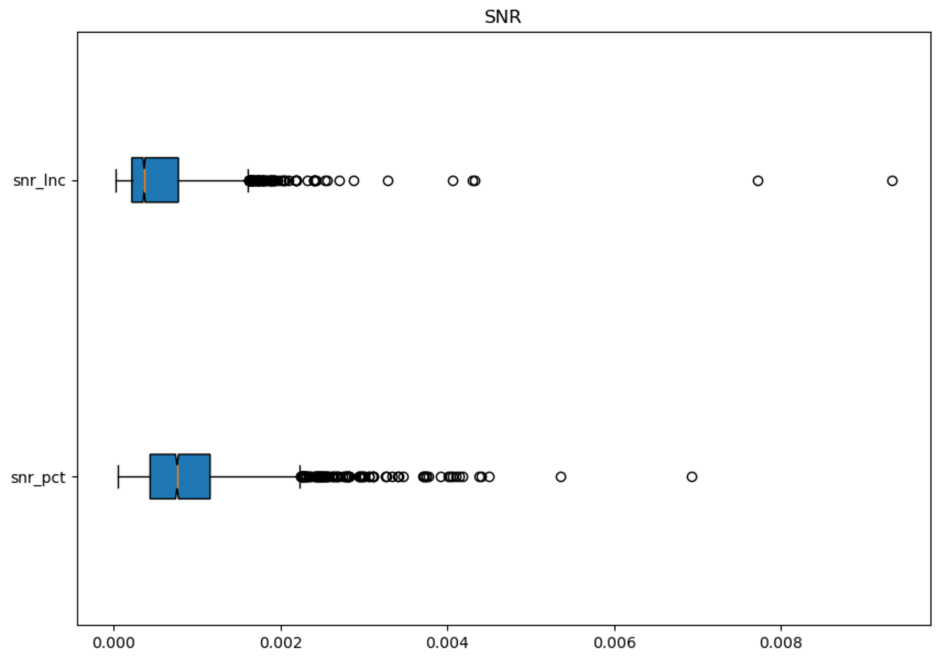


Figure 3.2: Boxplot for SNR

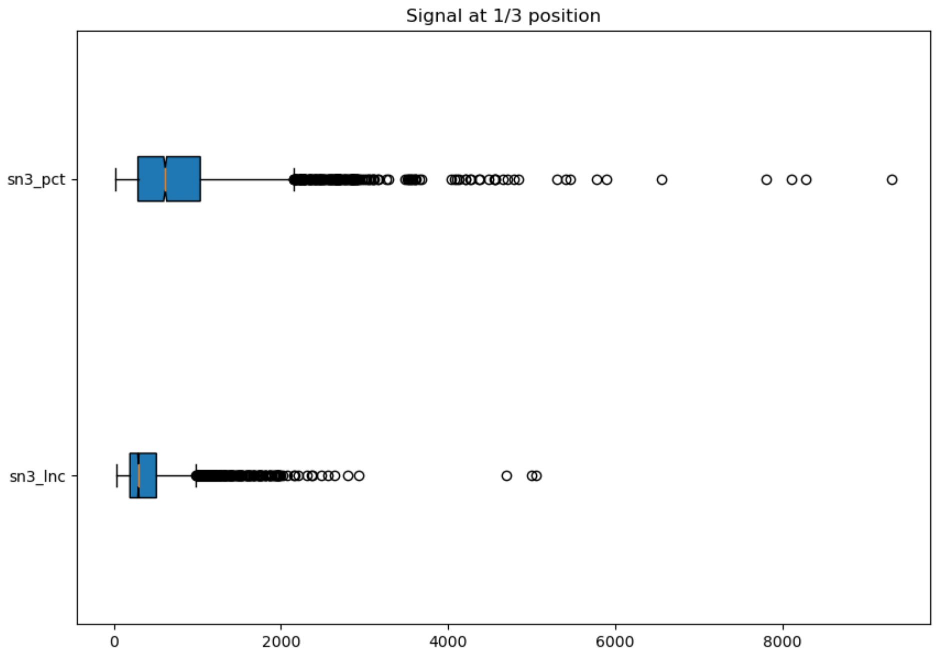


Figure 3.3: Boxplot for Signal at 1/3 position

3.2.1.2 Model 1 training

An SVM model with a linear kernel was trained on 5,123 samples which contained a mixture of lncRNA and PCT sequences. The features were all EIIP based features and the total number of features was six. The model completed its training in 1.3925 seconds. The following table summarizes the accuracy, sensitivity and specificity of the model obtained-

Model type	Accuracy	Sensitivity	Specificity
Model 1	0.6478	0.7148	0.6161

Table 3.3: Parameters for Model 1

3.2.2 Model trained on EIIP and ORF features

3.2.2.1 EIIP and ORF features calculation

The EIIP features are directly taken from the calculation done for Model 1. For ORF features, we consider the ORF length and ORF coverage of every sequence. Following steps are used to calculate them-

1. For each sequence, get all possible ORFs by fixing the start codon as "AUG" and stop codons as "UAA", "UAG" and "UGA". A sequence can have any number of ORFs. Consider all three cases- taking the complete sequence, without the first nucleotide and without the first two nucleotides. This ensures that all possible ORFs are calculated.

2. Calculate the length of each ORF per sequence and select the ORF with maximum length. This forms the ORF length feature

3. Divide each ORF length per sequence with the length of the sequence. This gives the ORF Coverage feature.

The following block diagram explains the feature calculation more precisely-

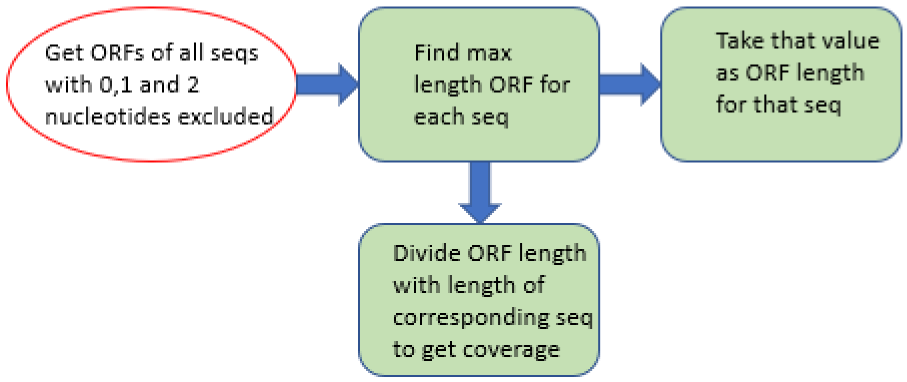

Get ORFs of all seqs with 0,1 and 2 nucleotides excluded
Find max length ORF for each seq
Take that value as ORF length for that seq
Divide ORF length with length of corresponding seq to get coverage

Figure 3.4: Block diagram for ORF features calculation

The total time taken to calculate the ORF features for training and testing sequences is 35 seconds. Thus the average time taken per sequence to get its ORF features is 0.0043078 seconds.

It has been observed that since lncRNAs do not translate to proteins, their ORF lengths are usually smaller than PCTs. This is because longer the length of ORFs, more is the probability that it will get translated to proteins. The following scatter plots for ORF length and ORF coverage give an idea about it-

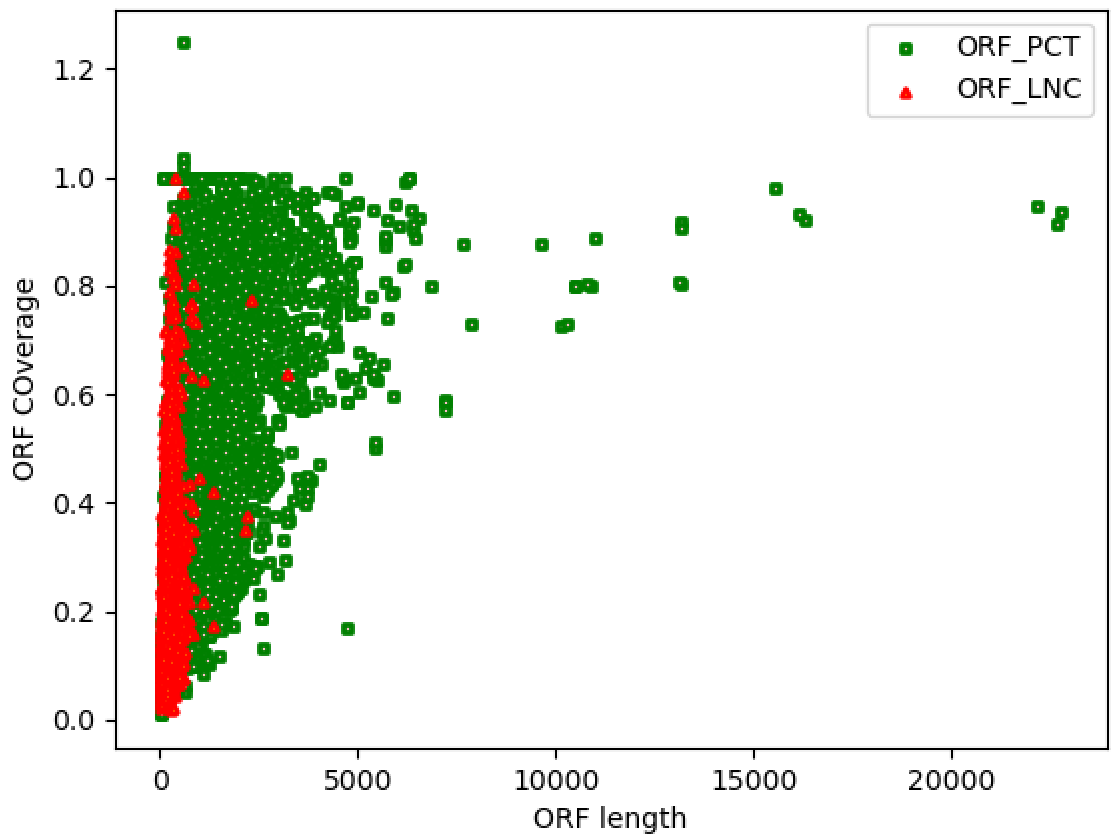


Figure 3.5: Scatter plot for ORF length and ORF coverage

3.2.2.2 Model 2 training

An SVM model with a linear kernel was trained on 5,123 samples of lncRNAs and PCTs. The features used were EIIP features and ORF features. The total number of features used were eight. The model completed its training in 1.7249 seconds. The following table summarizes the accuracy, sensitivity and specificity of the model obtained-

Model type	Accuracy	Sensitivity	Specificity
Model 2	0.8241	0.9516	0.7561

Table 3.4: Parameters for Model 2

3.2.3 Model trained on EIIP and Sequence based features

3.2.3.1 EIIP and Sequence based features calculation

Sequence based features are made of ORF features and Logarithmic distance features with respect to ORFs of sequences. The EIIP and ORF features are borrowed from Model 2. For calculating the Logarithmic features, the following steps apply-

1. Calculate hexamer frequencies of all possible hexamers of all the ORFs and the lncRNA and PCT reference sequences.

2. Calculate average hexamer frequencies of each hexamer of the lncRNA and PCT reference sequences.

3. Divide the the hexamer frequency of each hexamer per sequence with the corresponding hexamer in the lncRNA and PCT reference sequences.

4. Calculate the natural logarithm of each value. If a value is 0, then skip calculating the natural logarithm-instead count the number of zeros.

5. Add all non-zero values and subtract the number of zeros. Then divide each of them by the corresponding sequence length.

6. This gives two features- log distance with respect to lncRNA and log distance with respect to PCT.

7. Now, calculate the ratio of log distance w.r.t lncRNA and log distance w.r.t PCT for each sequence. This gives the final Log distance for ORF feature.

The following block diagram gives a better understanding-

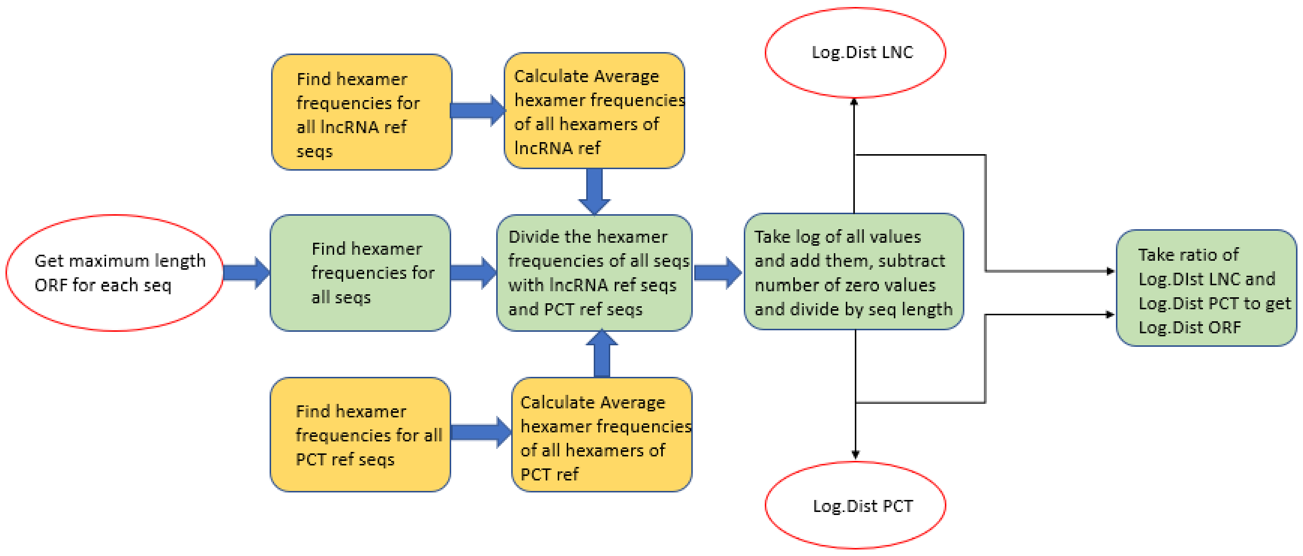

Find hexamer frequencies for all lncRNA ref seqs
Calculate Average hexamer frequencies of all hexamers of lncRNA ref
Log.Dist LNC
Get maximum length ORF for each seq
Find hexamer frequencies for all seqs
Divide the hexamer frequencies of all seqs with lncRNA ref seqs and PCT ref seqs
Take log of all values and add them, subtract number of zero values and divide by seq length
Take ratio of Log.Dist LNC and Log.Dist PCT to get Log.Dist ORF
Find hexamer frequencies for all PCT ref seqs
Calculate Average hexamer frequencies of all hexamers of PCT ref
Log.Dist PCT

Figure 3.6: Block Diagram for Log Distance ORF

The total time to calculate the Log Distance for ORF features is 6.5 hours. This includes calculating hexamer frequencies for reference frequencies. Thus, the avearge time per sequence to get these features is 2.72 seconds.

3.2.3.2 Model 3 training

An SVM model with a linear kernel was trained on 5,123 samples containing lncRNAs and PCT sequences. The features used were EIIP and sequence based features. Thus, the total number of features used were eleven. The model completed its training in 0.8285 seconds. The table below summarizes the model parameters-

Model type	Accuracy	Sensitivity	Specificity
Model 3	0.8228	0.9320	0.7609

Table 3.5: Parameters for Model 3

3.2.4 Model trained on EIIP, Sequence based and Secondary Structure based features

3.2.4.1 EIIP, Sequence based and Secondary Structure based features calculation

Secondary structure features are calculated by first obtaining the secondary structure and minimum free energy of each sequence and then calculating the log distance of acguD and acguACGU sequences. Minimum Free Energy for both types of sequences can be visualized as followed-

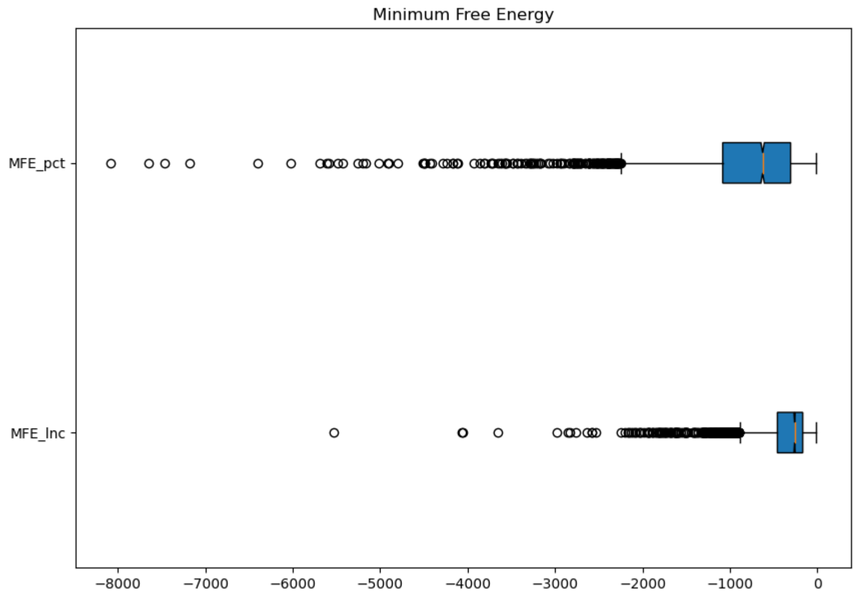


Figure 3.7: Boxplot for Minimum Free Energy

MFE for PCTs is always lower than lncRNAs- which makes it a distinguishable feature. The EIIP and sequence based features are borrowed from Model 3. The following steps explain the calculation of secondary structure based features-

1. Run the RNAFold algorithm from the ViennaRNA package and get the secondary structures of sequences in dot-bracket notation and the minimum free energy of each sequence.

2. Convert the dot-bracket sequences to PU sequences and calculate the PU frequency.

3. Convert the dot-bracket sequences to acguD and acguACGU sequences.

4. Calculate the log distance of acguD and acguACGU sequences by following the steps in calculating the log distance of ORFs. Here, the entire sequence will be considered.

The following block diagram gives a better representation of the above steps-

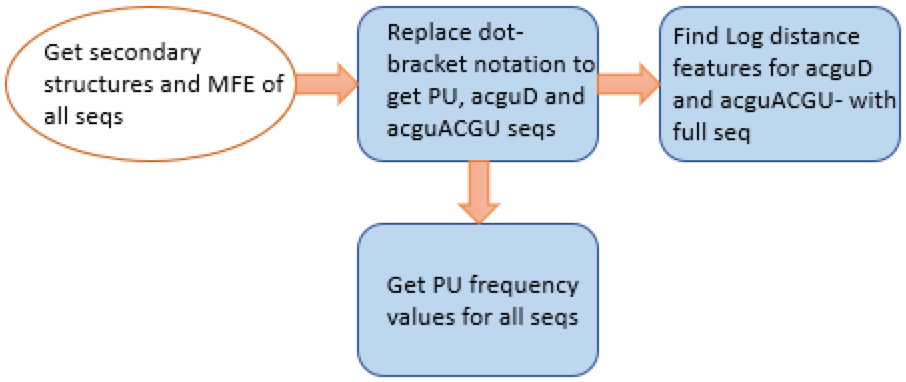

Get secondary structures and MFE of all seqs
Replace dot-bracket notation to get PU, acguD and acguACGU seqs
Find Log distance features for acguD and acguACGU- with full seq
Get PU frequency values for all seqs

Figure 3.8: Block Diagram for Secondary Structure based features

The total time taken to calculate the secondary structure based sequences is 18 hours. Thus, the time taken for each sequence is 12.6 seconds.

3.2.4.2 Model 4 training

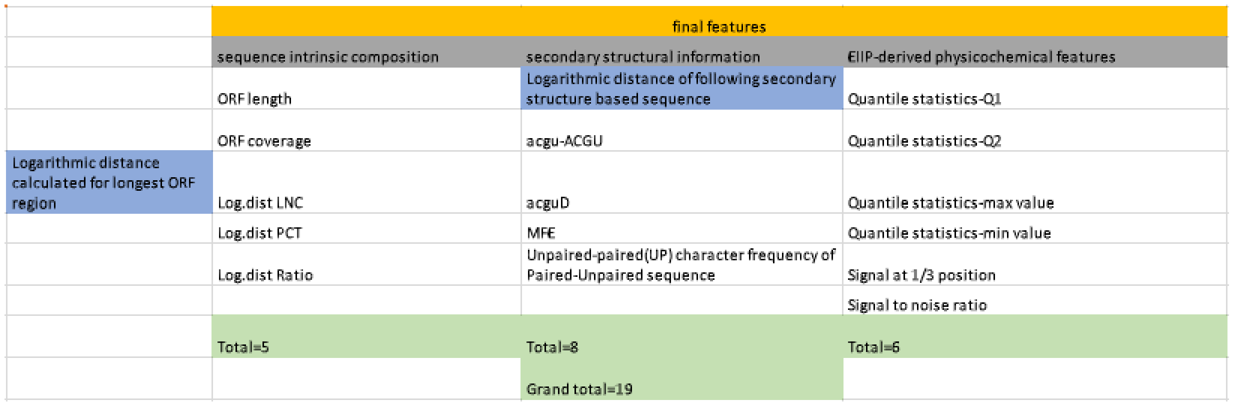
An SVM model with a linear kernel was trained on 5,123 samples containing lncRNAs and PCTs. The features used were EIIP, sequence based and secondary structure based features. Total features used were nineteen. The model completed the training in 1.0429 seconds. The model parameters are summarized below-

Model type	Accuracy	Sensitivity	Specificity
Model 4	0.8265	0.9356	0.7643

Table 3.6: Parameters for Model 4

3.2.5 Summary of Features

The following snapshot gives a summary of all the features used in training all 4 SVM models.



	final features		
	sequence intrinsic composition	secondary structural information	EIIP-derived physicochemical features
	ORF length	Logarithmic distance of following secondary structure based sequence	Quantile statistics-Q1
	ORF coverage	acgu-ACGU	Quantile statistics-Q2
Logarithmic distance calculated for longest ORF region	Log.dist LNC	acguD	Quantile statistics-max value
	Log.dist PCT	MFE	Quantile statistics-min value
	Log.dist Ratio	Unpaired-paired(UP) character frequency of Paired-Unpaired sequence	Signal at 1/3 position
			Signal to noise ratio
	Total=5	Total=8	Total=6
		Grand total=19	

Figure 3.9: Summary of features

The time taken for calculation of features for each model and time taken for training of each model is summarized below-

Model type	No. of features	Time taken for calculation/seq(s)	Time taken for training(s)
Model 1	6	0.0024621445	1.3925
Model 2	8	0.0043078	1.7249
Model 3	11	2.72	0.8285
Model 4	19	12.6	1.0429

Table 3.7: Summary of feature calculation time(s) and model training time(s)

3.3 HTVS Pipeline Optimization

3.3.1 Cost Selection for each model/stage

Each model has its own cost associated with it to carry out computation. We have allocated the cost for each model by taking the time taken for features of each model to be calculated. The cost in milliseconds has been summarized below-

Model type	Cost for each model(ms)
Model 1	0.00139501905028
Model 2	0.0017322870267
Model 3	0.0035501488598
Model 4	0.013654243837

Table 3.8: Computational cost for each model(ms)

3.3.2 Selection of λ_4

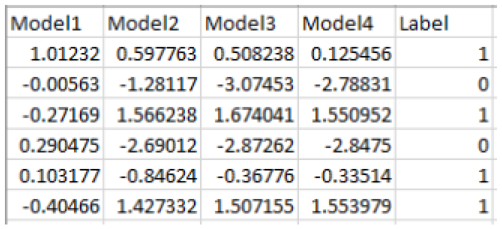
λ_4 is the threshold value for the last stage-in our case the fourth stage. As observed from the scores calculated for each sample, the last model-Model 4 has scores such that all of those related to lncRNA sequences have a value above 0.1. Thus, we have chosen its value to be 0.1.

3.3.3 Score and Correlation Calculation

All the models were used to predict the values of sequences from testing sequences. This resulted in hard values or either 1 or 0. These values were used to used to calculate the accuracy, sensitivity and specificity of models as shown in the previous section.

For each model, the predictive scores for test sequences were calculated. Since we have used SVM models, we calculated these scores using the decision_function method available in "sklearn" package.

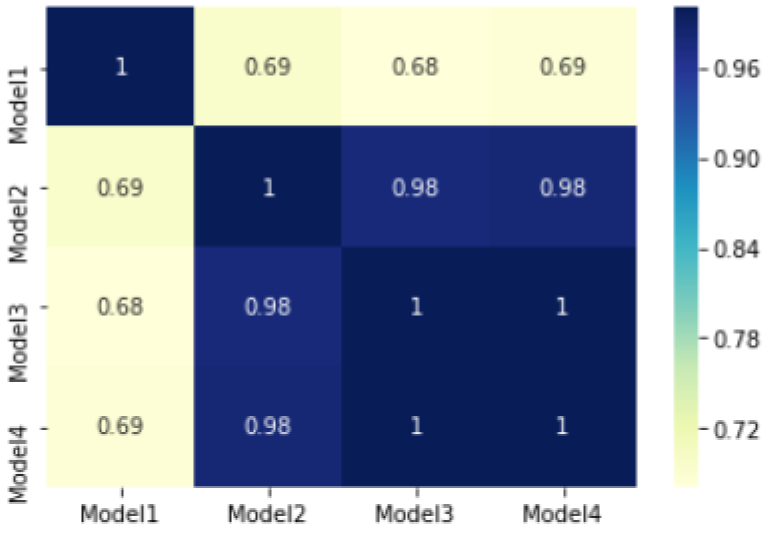
The snapshot of scores of some sequences is shown below-



Model1	Model2	Model3	Model4	Label
1.01232	0.597763	0.508238	0.125456	1
-0.00563	-1.28117	-3.07453	-2.78831	0
-0.27169	1.566238	1.674041	1.550952	1
0.290475	-2.69012	-2.87262	-2.8475	0
0.103177	-0.84624	-0.36776	-0.33514	1
-0.40466	1.427332	1.507155	1.553979	1

Figure 3.10: Snapshot of scores of samples for all models

Using the scores predicted above, we calculate correlation using the Pearson's correlation function. This helps to see how correlated the models are to each other. The following heat-map helps to visualize that-



```
        Model1    Model2    Model3    Model4
Model1  1.000000  0.694526  0.681374  0.685802
Model2  0.694526  1.000000  0.978197  0.980394
Model3  0.681374  0.978197  1.000000  0.999246
Model4  0.685802  0.980394  0.999246  1.000000
```

Figure 3.11: Correlation of models with each other

According to the heat-map, Model 3 and Model 2 are the most correlated to the Model 4. Also, Model 4 has the highest complexity because it considers all possible features.

The last stage of the pipeline has been fixed as Model 4. This is because it gave the highest value in terms of accuracy and specificity, thus it has the probability to produce maximum number of potential candidates. Our goal in this project is to develop an optimized HTVS pipeline with maximum four stages. The optimized pipeline may also show some good results for a smaller number of stages. Thus, it is important to have a starting point on how the stages should be arranged so that the optimization process can begin from there. According to this paper, after performing optimization on a 4-stage pipeline with simulated data, it was observed that when all the stages were arranged in the increasing order of correlation value w.r.t to the final stage, it reached the maximum number of desired candidates much earlier with less computational cost. Thus, we have also arranged our models with increasing complexity and correlation with the last stage as our initial pipeline.

3.3.4 Pipeline Optimization

As described earlier, the aim is to estimate the values of lambdas of earlier stages by taking into consideration various computational budgets. The objective function with the cost constraints was formalised in Eq. 2.12. Thus, for each permutation of stages, we place a constraint on computational cost and get the values of lambdas. We also observe the number of samples obtained at the end of the last stage for each computational cost at every permutation.

The following block diagram gives a better overview-

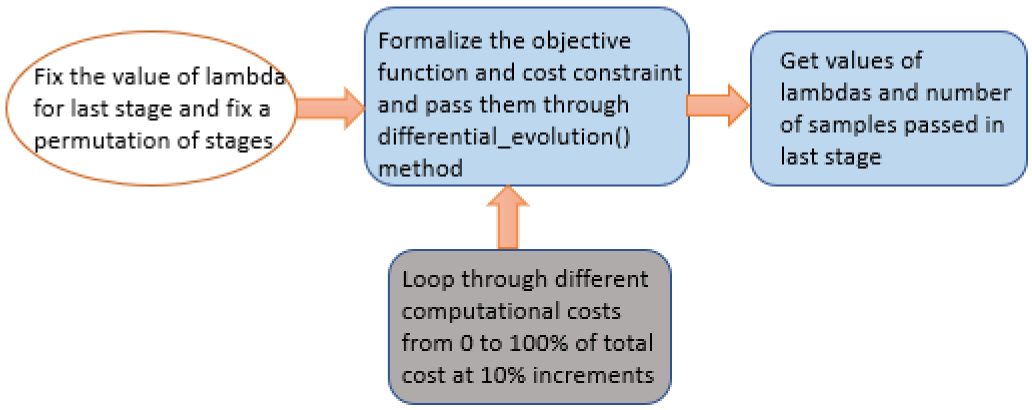

Fix the value of lambda for last stage and fix a permutation of stages
Formalize the objective function and cost constraint and pass them through differential_evolution() method
Get values of lambdas and number of samples passed in last stage
Loop through different computational costs from 0 to 100% of total cost at 10% increments

Figure 3.12: Block Diagram of Optimization Process

Additionally, a baseline pipeline has also been defined which is used to compare with the optimized pipeline and check if how better our pipeline is. In all cases, our optimized pipelines have worked better than the baseline pipelines.

3.3.4.1 Results of Optimization

After running the optimization code from author for 2,439 samples among which 1,595 were positive samples of lncRNAs, the following results were obtained. The computation cost was looped through 0 to cost of stage 4. Keeping the last stage as fixed with Model 4, all permutations were taken-

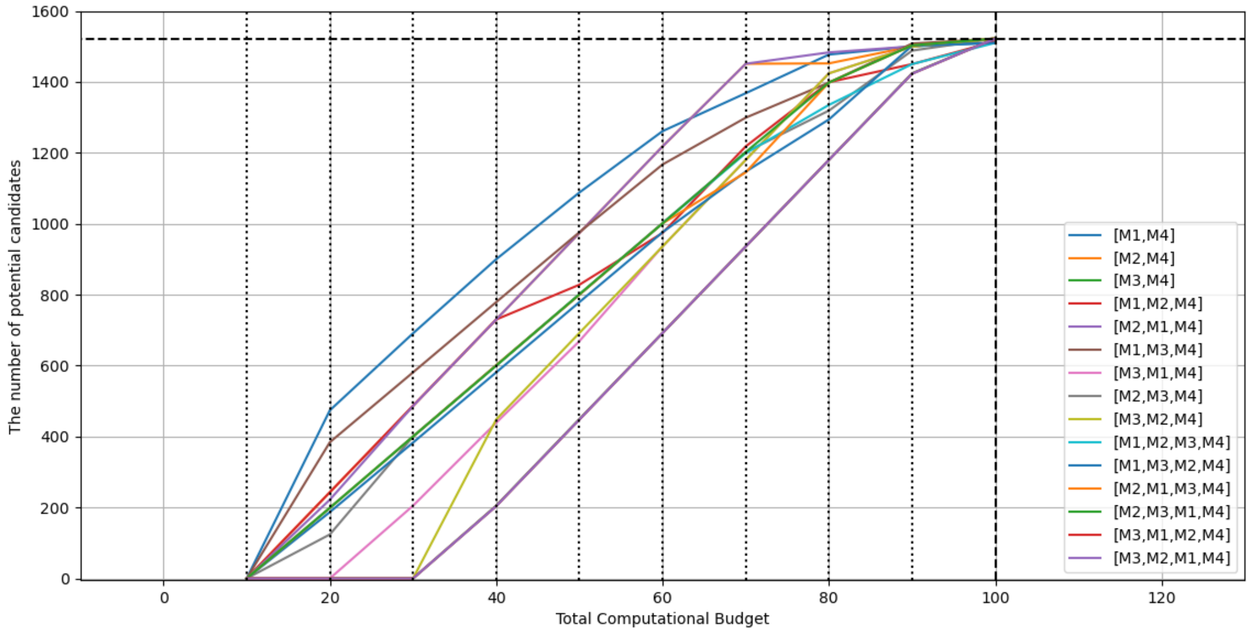


Figure 3.13: Computational Cost versus potential candidates

Taking Model 1 as M1, Model 2 as M2, Model 3 as M3 and Model 4 as M4, The above graph has been tabulated as follows-

Stages	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
[M1, M4]	0	474	691	900	1088	1261	1368	1477	1500	1510
[M2, M4]	0	243	486	730	974	1218	1451	1452	1512	1522
[M3, M4]	0	0	0	205	449	692	936	1180	1424	1522
[M1, M2, M4]	0	243	487	730	828	974	1218	1399	1450	1514
[M2, M1, M4]	0	222	486	730	974	1218	1451	1483	1500	1522
[M1, M3, M4]	0	384	581	779	976	1167	1299	1398	1508	1552
[M3, M1, M4]	0	0	205	439	669	936	1180	1424	1500	1522
[M2, M3, M4]	0	124	400	601	800	1002	1202	1319	1488	1522
[M3, M2, M4]	0	0	0	448	692	936	1180	1424	1499	1522
[M1, M2, M3, M4]	0	199	400	600	801	1001	1199	1335	1450	1510
[M1, M3, M2, M4]	0	187	383	581	779	976	1147	1294	1502	1510
[M2, M1, M3, M4]	0	199	400	601	800	1001	1145	1398	1502	1522
[M2, M3, M1, M4]	0	199	400	600	801	1001	1202	1398	1503	1522
[M3, M1, M2, M4]	0	0	0	204	448	692	936	1180	1423	1522
[M3, M2, M1, M4]	0	0	0	204	448	692	936	1180	1423	1522

Table 3.9: Computational cost vs Potential Candidates

3.3.4.2 Discussion

The simulations were run by setting the computational cost and λ_4. Then after deciding the order of the stages and taking individual score of them, we started our simulation. As seen from the graph and table, the cost varied from 10% of total cost to total cost. The total number of true lncRNAs taken is 1,522. We have passed a total of 2,435 sequences. Thus based on the results obtained, we made the following observations-

1. 10% of total computational cost is not enough for any permutation of stages to give any potential candidates.

2. The study from which the optimization method was followed, it was observed that correlation between stages was directly proportional to the slope of graph containing the potential number of candidates. Thus in our case, Model 2 and Model 3 are highly correlated to Model 1. Thus, we

can see that the stages which contain only Model 2, Model 3 and Model 4, be it any combination-especially the combinations [M2, M4], [M3, M2, M4], [M2, M3, M4] and [M3, M4], has a steep slope and identifies 1,595 samples at 100% computational cost. The steep slope ensures that they identified more candidates with less computational cost much as compared to the other stages.

3. Model 1 is least correlated to Model 4.This leads to not detecting 1,595 sequences even at 100% cost utilisation for stages involving Model 1.

4. Model 3 is more correlated to Model 4 as compared to Model 2, bust also computationally more expensive. Thus when we compare the pipelines ([M3, M1, M4],[M1, M3, M4]) and [M2, M1, M4], the latter has a steeper slope, thus getting more candidates for lesser cost. It means that if stages with higher complexity are placed as prior stages, then we get less candidates. These also show that when stages that are highly correlated to the last stages are placed as initial stages, our number of potential candidates increases.

5. Pipelines [M2, M1, M3, M4] and [M2, M3, M1, M4] almost give same number of candidates, with the latter being slightly better. Both pipelines have the worst performance among all the 4-stage pipelines.

6. Thus, overall we can observe that among the 3-stage pipelines, [M2, M1, M4] and among the 4-stage pipeline, [M2, M3, M1, M4] work the best.

7. Pipelines with M1 as the first stage work best when presented with a low computational cost. AT 20% cost, all those pipelines give a higher number of potential candidates than others.

8. All pipelines starting with either M1 or M2 can identify almost 75% candidates at 60% cost.

9. Even though we started from the pipeline [M1, M2, M3, M4] and our best pipelines can detect 1,595 sequences, the difference is not so stark-being only 6% different from other pipelines.

10. At 90% cost, all stages could get the same number of potential candidates. The maximum differences was about 3.6%, which is not a big number.As the cost increases, this difference reduces even more.

The table obtained shows the total number of sequences obtained at the end of Stage 4 of each pipeline for different cost. But, It is always better to consider how good are the values that we got. Thus, we calculated the accuracy, precision, recall and F-1 score of all pipelines.

The following graph gives a view on accuracy's obtained-

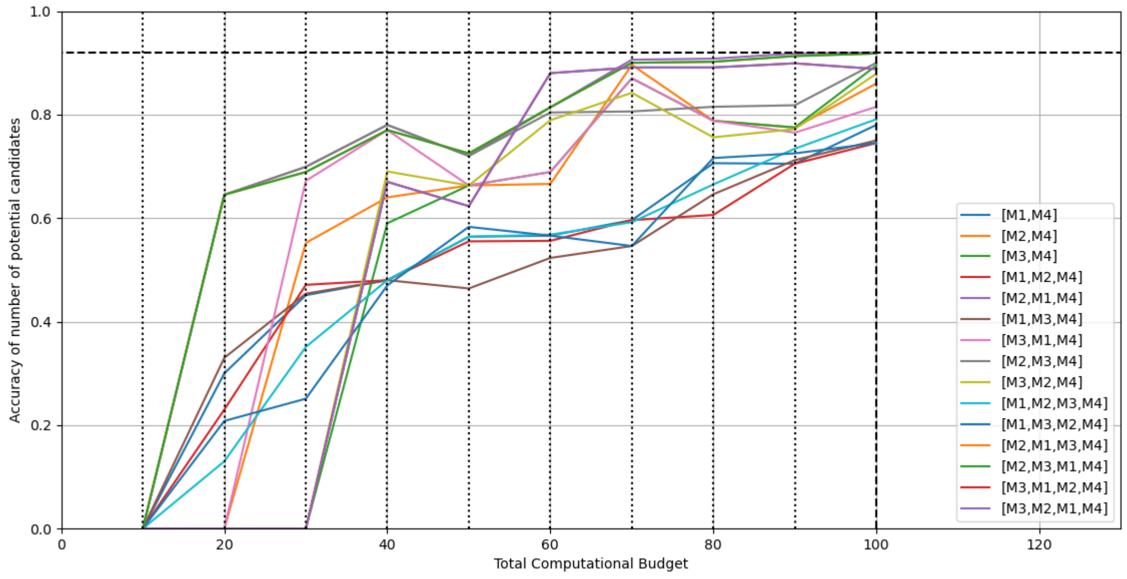


Figure 3.14: Accuracy-Computational Cost versus potential candidates

The pipelines [M2, M1, M4] and [M2, M3, M1, M4] have shown the highest accuracy at almost all costs. The stages involving M1 as first stage have the worst performance overall. Thus, both 3-stage and 4-stage pipelines work good enough with respect to accuracy.The last two permutations and [M2, M1, M3, M4] and [M2, M3, M1, M4], have the same accuracy. Pipelines starting with M2 and M3-especially the 4-stage pipelines, have also shown a higher accuracy-probably due to the higher complexity and better classification by them. Overall, at 90% cost, a good percentage of candidates are rightly detected by almost all pipelines.

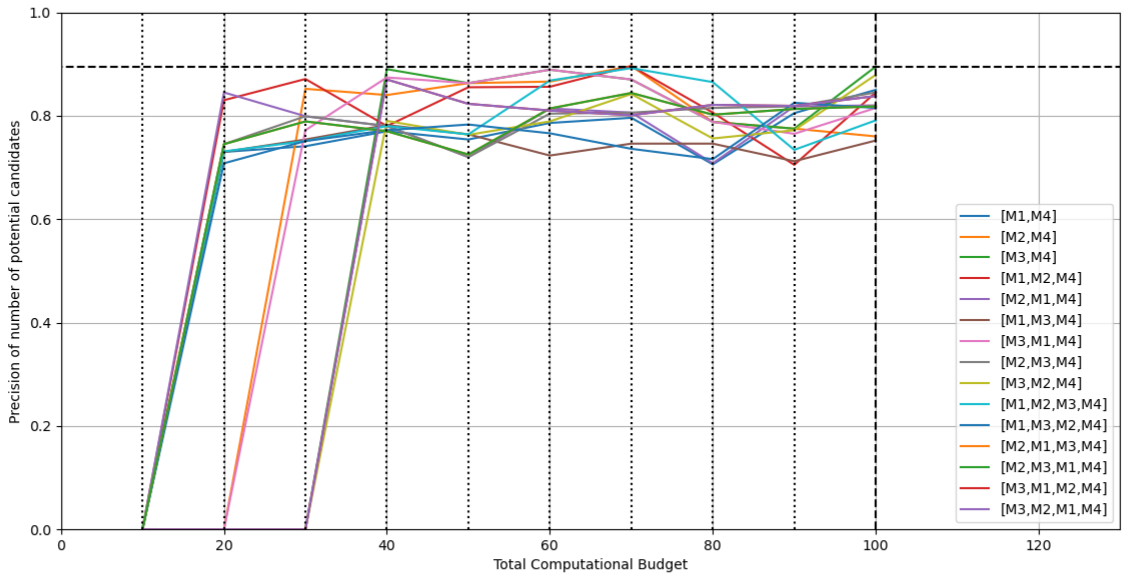


Figure 3.15: Precision-Computational Cost versus potential candidates

From the graph above it can be seen that all the pipelines are precise enough for all computational costs. There is hardly any change regarding that.

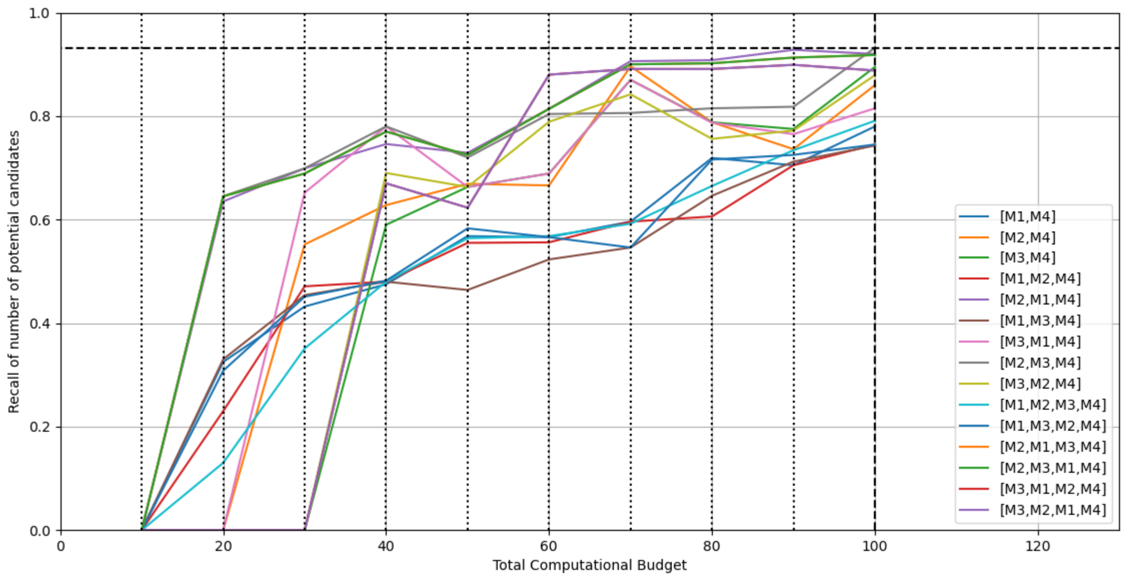


Figure 3.16: Recall-Computational Cost versus potential candidates

As we can see from the graph, the values for recall are very similar to that of accuracy. This works had in hand with the maintained value of precision for all pipelines- meaning the number of false positives are less but overall false negatives are more. Since we want to maximize the number of potential candidates, recall value is a good measure to do that.

Taking the harmonic mean of recall and precision, we get the following graph for F1 scores-

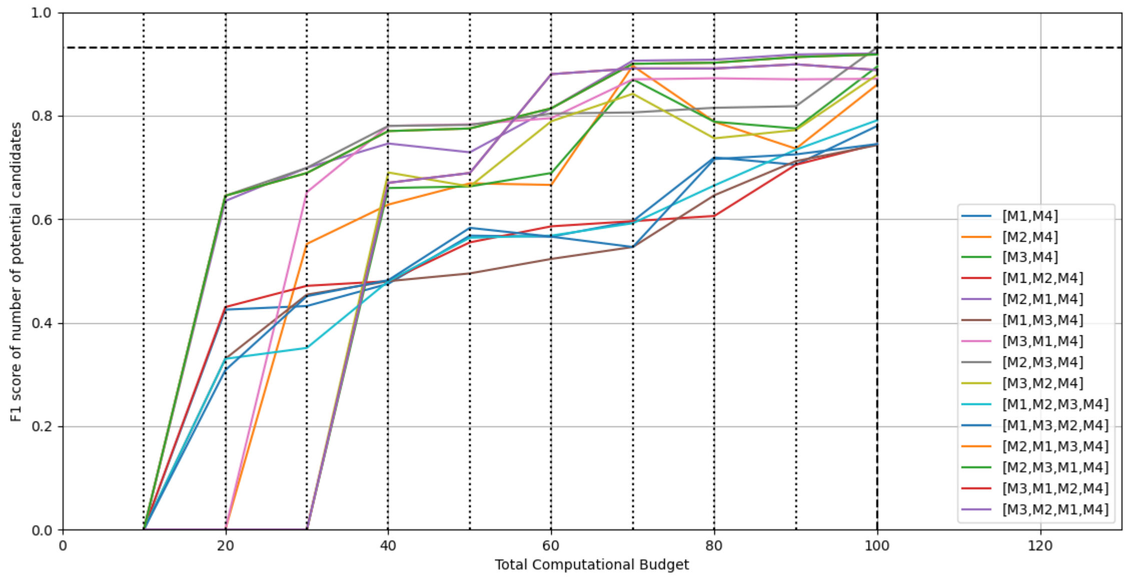


Figure 3.17: F1 score-Computational Cost versus potential candidates

3.4 Code Availability

All codes for reproducing the results can be found at the HTVS pipeline construction for lncRNA classification Github Repoistory.

4. CONCLUSIONS

From this project, we first studied the basic concepts related to RNAs and how we can find different features related to different types of RNAs. We identified three groups of features- sequence based, EIIP based and secondary structure based features.

We have developed four SVM models, each trained with a different set of features defining lncRNAS and PCTs. These models are then arranged in increasing order of correlation between the all prior stages and the last stage. Then, a pipeline optimization framework is applied with a range of fixed computational costs. Each permutation of stages is studied and the total number of sequences as well as the accuracy of each pipeline is reported.

After thorough investigation, pipelines [M2, M1, M4] and [M2, M3, M1, M4] detect the highest number of potential candidates and also have the highest accuracy.

4.1 Future Work

The following future work can be performed-

1. Other models instead of SVM can be used. SVM was considered because it is a simple model, but other Machine Learning models can help to get better accuracy.

2. The dataset size can be increased to get a better perspective.

3. The computational cost restrictions can be softened instead of the hard restrictions taken by us, and instead of just this cost other costs which influence pipelines can be considered. These costs can include factors which can be decided by he user or the company.

REFERENCES

[1] A. Clyde, S. Galanie, D. W. Kneller, H. Ma, Y. Babuji, B. Blaiszik, A. Brace, T. Brettin, K. Chard, R. Chard, *et al.*, "High-throughput virtual screening and validation of a sars-cov-2 main protease noncovalent inhibitor," *Journal of chemical information and modeling*, 2021.

[2] L. Cheng, R. S. Assary, X. Qu, A. Jain, S. P. Ong, N. N. Rajput, K. Persson, and L. A. Curtiss, "Accelerating electrolyte discovery for energy storage with high-throughput screening," *The journal of physical chemistry letters*, vol. 6, no. 2, pp. 283–291, 2015.

[3] Y.-J. Kang, D.-C. Yang, L. Kong, M. Hou, Y.-Q. Meng, L. Wei, and G. Gao, "Cpc2: a fast and accurate coding potential calculator based on sequence intrinsic features," *Nucleic acids research*, vol. 45, no. W1, pp. W12–W16, 2017.

[4] I. Chakraborty and P. Maity, "Covid-19 outbreak: Migration, effects on society, global environment and prevention," *Science of the Total Environment*, vol. 728, p. 138882, 2020.

[5] G. Chen, Z. Wang, D. Wang, C. Qiu, M. Liu, X. Chen, Q. Zhang, G. Yan, and Q. Cui, "Lncrnadisease: a database for long-non-coding rna-associated diseases," *Nucleic acids research*, vol. 41, no. D1, pp. D983–D986, 2012.

[6] L. Wang, H. J. Park, S. Dasari, S. Wang, J.-P. Kocher, and W. Li, "Cpat: Coding-potential assessment tool using an alignment-free logistic regression model," *Nucleic acids research*, vol. 41, no. 6, pp. e74–e74, 2013.

[7] H.-M. Woo, X. Qian, L. Tan, S. Jha, F. J. Alexander, E. R. Dougherty, and B.-J. Yoon, "Optimal decision making in high-throughput virtual screening pipelines," *arXiv preprint arXiv:2109.11683*, 2021.

[8] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles, *et al.*, "The gencode v7 catalog of human long noncoding rnas:

analysis of their gene structure, evolution, and expression," *Genome research*, vol. 22, no. 9, pp. 1775–1789, 2012.

[9] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[10] S. Han, Y. Liang, Q. Ma, Y. Xu, Y. Zhang, W. Du, C. Wang, and Y. Li, "Lncfinder: an integrated platform for long non-coding rna identification utilizing sequence intrinsic composition, structural information and physicochemical property," *Briefings in bioinformatics*, vol. 20, no. 6, pp. 2009–2027, 2019.

[11] Q. Lu, S. Ren, M. Lu, Y. Zhang, D. Zhu, X. Zhang, and T. Li, "Computational prediction of associations between long non-coding rnas and proteins," *BMC genomics*, vol. 14, no. 1, pp. 1–10, 2013.

[12] X. Shi, M. Sun, H. Liu, Y. Yao, R. Kong, F. Chen, and Y. Song, "A critical role for the long non-coding rna gas5 in proliferation and apoptosis in non-small-cell lung cancer," *Molecular carcinogenesis*, vol. 54, no. S1, pp. E1–E12, 2015.

[13] X. Chen, C. C. Yan, X. Zhang, and Z.-H. You, "Long non-coding rnas and complex diseases: from experimental results to computational models," *Briefings in bioinformatics*, vol. 18, no. 4, pp. 558–576, 2017.

[14] D. L. Filer, P. Kothiya, R. W. Setzer, R. S. Judson, and M. T. Martin, "tcpl: the toxcast pipeline for high-throughput screening data," *Bioinformatics*, vol. 33, no. 4, pp. 618–620, 2017.

[15] J. Harrow, A. Frankish, J. M. Gonzalez, E. Tapanari, M. Diekhans, F. Kokocinski, B. L. Aken, D. Barrell, A. Zadissa, S. Searle, *et al.*, "Gencode: the reference human genome annotation for the encode project," *Genome research*, vol. 22, no. 9, pp. 1760–1774, 2012.