Aus dem Zentrum für Medizinische Forschung
der Medizinischen Fakultät Mannheim
(Direktor: Prof. Dr. med. Norbert Gretz)

Advanced hierarchical learning approach for microRNA and target prediction

Inauguraldissertation
zur Erlangung des akademischen Grades
Doctor scientiarum humanarum (Dr. sc. hum.)
der
Medizinischen Fakultät Mannheim
der Ruprecht-Karls-Universität
zu
Heidelberg

vorgelegt von
Alisha Parveen

aus
New Delhi, India
2019

Dekan: Prof. Dr. med. Sergij Geordt
Referent: Prof. Dr. med. Norbert Gretz

TABLE OF CONTENTS

ABBREVIATIONS

ACC	Accuracy
AGO-1	Argonaute-1
AHDR	Advanced hierarchical deep-rooted network
AI	Artificial intelligence
ANN	Artificial neural network
AUC	Area under curve
BNB	Bernoulli naïve bayes
CNN	Convolutional neural network
CPU	Central processing unit
DFFN	Deep feedforward network
DL	Deep learning
DRN	Deep residual network
DT	Decision tree
GAN	Generative adversarial network
GPU	Graphics processing unit
LASSO	Least absolute shrinkage and selection operator
LR	Logistic regression
MCC	Matthews correlation coefficient
miRNAs	microRNAs
ML	Machine learning
OS	Operating system
PPV	Positive predictive value
ReLu	Rectified linear unit
RNN	Recurrent neural network
RNA pol II	RNA polymerase II
RISC	RNA induced silencing complex
SVM	Support vector machine
TPR	True positive rate
TNR	True negative rate

1. INTRODUCTION

1.1 microRNAs

microRNAs (miRNAs) are short endogenous non-coding RNA molecules of 17-24 nucleotides in size [1]. miRNAs have important regulatory roles in animals, plants, and viruses [1]. They regulate post-transcriptional expressions of genes by aligning to different regions of miRNA targets [1]. miRNAs are involved in several biological processes [1] as summarized in Figure 1. miRNAs play regulatory roles during cellular processes and differentiation events [1]. In a few years, miRNAs and gene regulation have impacted virtually every field of biology [1].

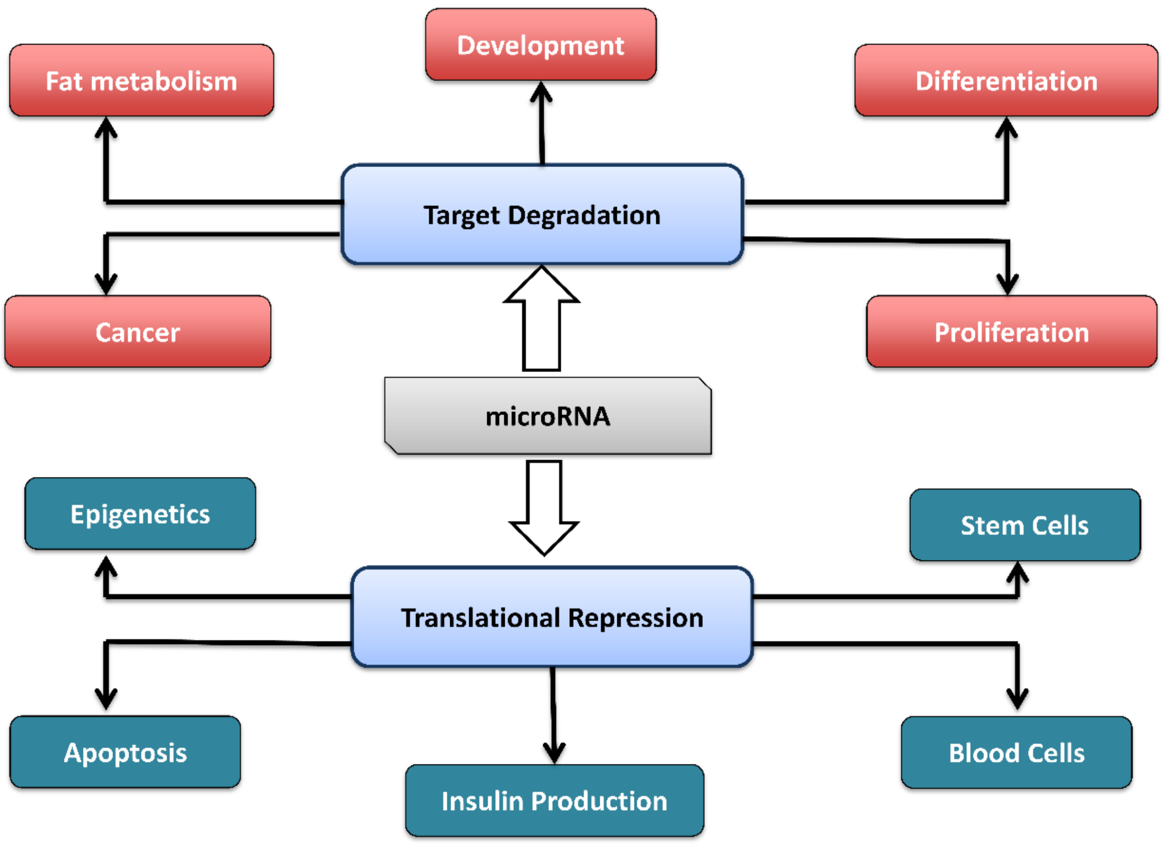

Figure 1. Overview of miRNA gene regulation in different cellular processes. miRNA binds to the target gene, which leads to target degradation and translational repression, which involved in different biological processes.

The RNA polymerase II (RNA pol II) plays a crucial role in the miRNA biogenesis [1, 2]. This enzyme transcribes miRNA genes as long primary transcripts, which are known as pri-miRNAs (Figure 2). While Drosha (ribonuclease III enzyme) processes pri-miRNA into stem-loop precursor miRNAs (pre-miRNAs) [1, 2], the exportin V (ds-RNA binding protein dependent on the RanGTP) transports them from nucleus into the cytoplasm [1, 2].

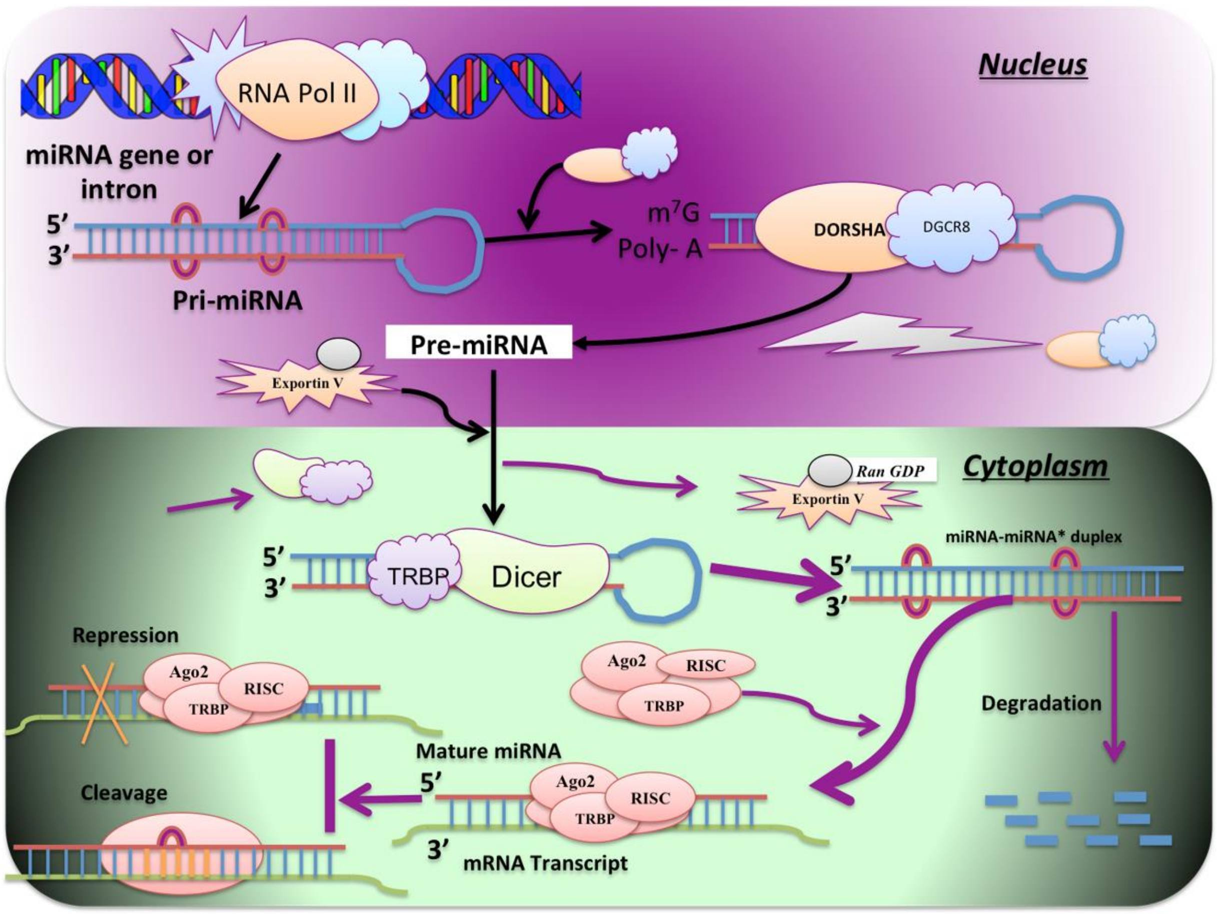

Figure 2. Overview of miRNA biogenesis. miRNA gene is processed into the nucleus by RNA pol II to generate pri-miRNA. Dorsha slices pri-miRNA into pre-miRNA and exports it into the cytoplasm using Exportin V. Dicer removes the loop-like structure from pre-miRNA and converts into a double-stranded mature sequence. In the end, the mature miRNA sequence along with RISC molecules undergoes gene regulation by translational repression or cleavage. This figure is adopted from Trionfini and Benigni [2].

In the cytoplasm, the maturation phase begins with Dicer (ATP dependent RNase III protein) that recognizes pre-miRNA and processes it into 21 base pair long miRNA-miRNA* duplex structure [1, 2]. The duplex structure consists of an antisense strand that contains the G:U wobble base pair, mismatches and unpaired at 5' end or sense strand [1, 2]. The duplex structure loads on argonaute-1 (Ago-1) protein along with the RNA-induced silencing complex (RISC) and this complex structure guides to the target mRNA that leads to gene silencing under three different processes - degradation, translational repression, and mRNA destabilization [1, 2].

Since miRNA are major regulators of cellular physiological processes, they also play important roles in human diseases including cancer, cardiovascular disease, diabetes, kidney disease, and neurological disorders [3]. oncomiRs are miRNAs with roles in cancer [3] while some miRNAs that suppress cancer are known as tumor suppressor miRNAs (Figure 3).

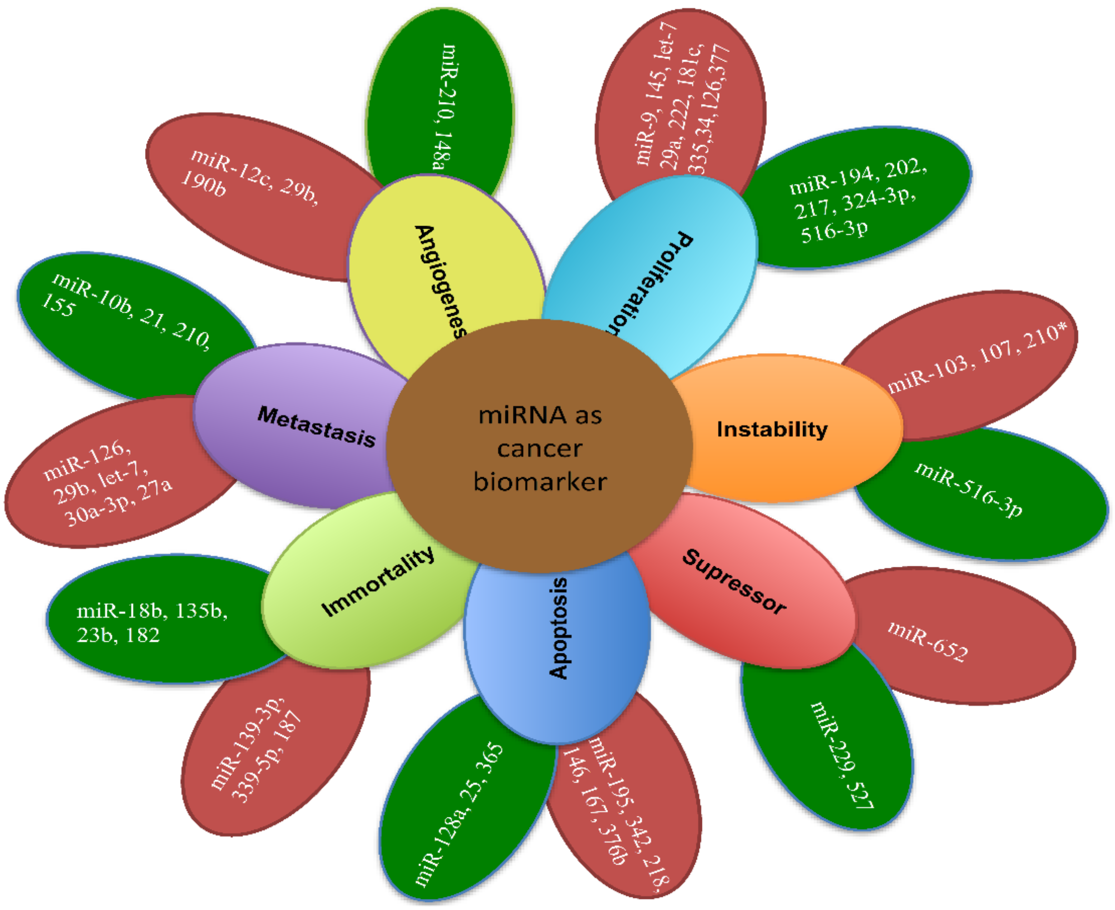

Figure 3. Application of miRNA in different biological processes. Each colorful petal is a different process of diseases or stages in cancer whereas green petal indicates upregulation processes and red petal indicates downregulation processes.

Table 1 summarizes roles of miRNAs in other human diseases by either up- or down-regulation.

Table 1. miRNA regulation involved in other human diseases.

Disease	**miRNA**	**Reference**
Cardiac hypertrophy	miR-23a, miR-23b, miR-24, miR-195, miR-199a, and miR-214 (**Upregulate**).	4
Alzheimer	miR-9, miR-128a, miR-125b (**Upregulate**).	5
Down syndrome	miR-99a, let-7c, miR-125b-2, miR-155 and miR-802 (**Upregulate**).	6
Systemic lupus erythematosus	miR-189, miR-61, miR-78, miR-21, miR-142-3p, miR 342, miR-299-3p, miR-198 and miR-298 (**Upregulate**).	7
	miR-196a, miR-17-5p, miR- 409-3p, miR-141, miR-383, miR- 112, and miR-184 (**Downregulate**).	
Psoriasis	miR-203 (**Upregulate**).	8
Rheumatic arthritis	miR-155, miR-146 (**Upregulate**).	9

1.1.1 *Measurable characteristics of miRNA target*

a) Seed sequence of miRNA

Seed sequence is a conserved heptamer region, which is located in 2-8 nucleotides from the starting point at the 5’ end of miRNA [1]. Generally, miRNAs are supposed to bind in the 3' untranslated region (3' UTR) region of the target gene for translational regulations [1]. Alignment between miRNA-targets follows the process of Watson and Crick mechanism in different architectures [10] such as canonical (Figure 4a) and non-canonical (Figure 4b-f) binding.

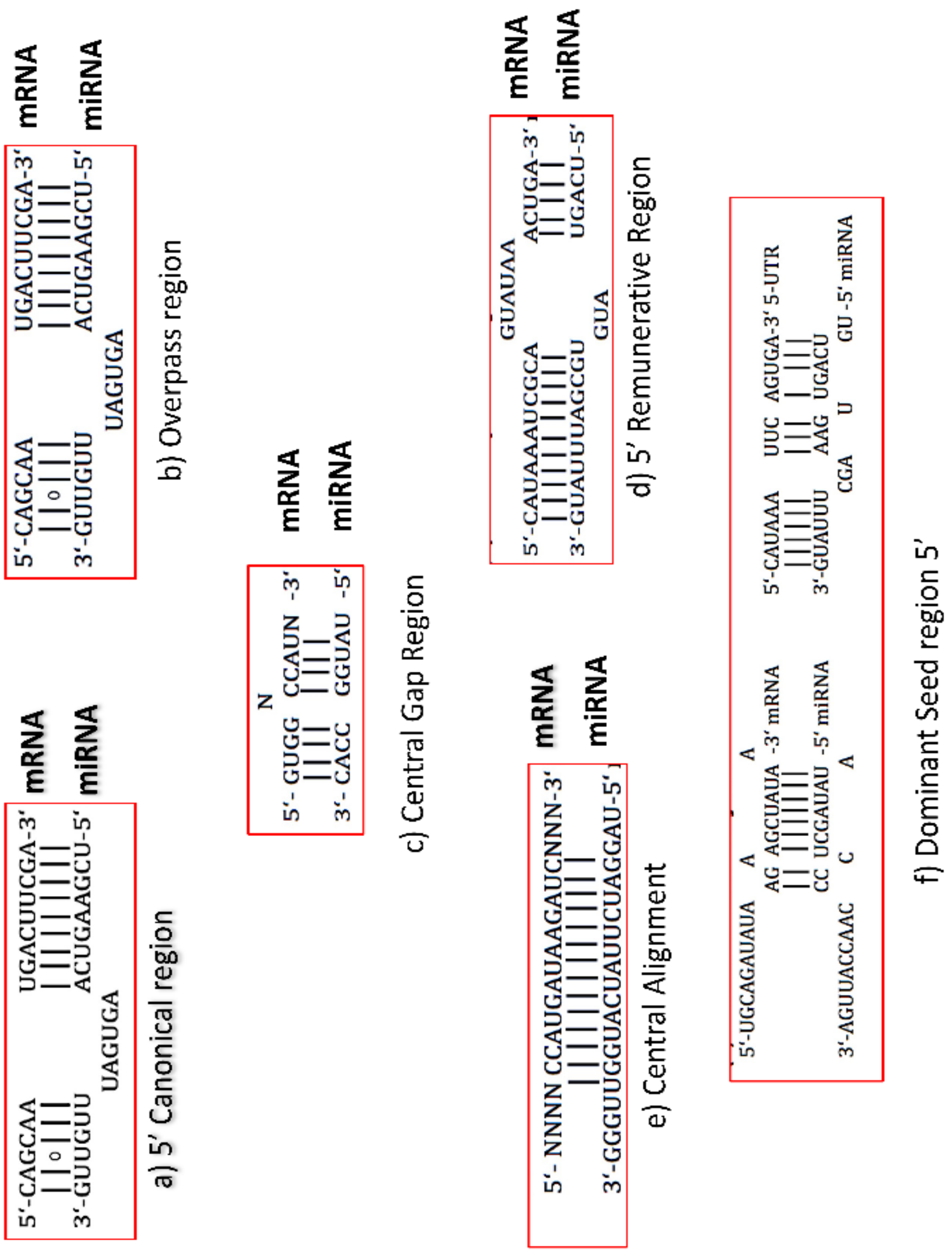

Figure 4. Seed type in alignment between miRNA and target gene. This figure is adopted from Dweep *et al.* [10].

b) Target site conservation

Sequence conservation is interpreted by the maintenance of nucleotides primarily in the 3' UTR, the 5' UTR, and the miRNA while considering multi-species alignment like human, chimpanzee, mouse, chicken, and fishes [10]. During miRNA target prediction, this conservational analysis hints for a functional miRNA target [10]. Hence, the conservation of the target site plays an important role in computing miRNA target interaction.

c) Free energy of miRNA-mRNA duplex structure

Gibbs free energy (or thermodynamics) measures the stability of a biological system and it is designated as ΔG [10]. The negative ΔG is indicative of less energy, hence higher stability. This applies to the prediction of the target site, as the hairpin loop has higher energy (positive ΔG) while the stem region has lower ΔG, during miRNA-target hybridization [10]. Therefore, the overall ΔG assists in miRNA target prediction.

d) Site accessibility

Site accessibility is an ability of a miRNA to locate and hybridize with a target [10]. It portrays an energy-based measure, which essentially evaluates the potential of a given target site to be single-stranded and thus accessible for the miRISC protein for binding [10]. The mRNA secondary structure then unfolds as the miRNA completes binding to a target [10]. Hence, the predicted amount of energy required to make a site accessible to a miRNA can be evaluated for computing actual target sites [10].

e) Other characteristics

There are other features to build the classifier model more effective for target gene prediction namely "AU content" in the binding site of the target gene [11], "G:U wobble" in seed match refers to the pairing of "G" with "U" instead of "C" nucleotide [11]. Additionally, the probability of multiple miRNA binding to a specific target is called multiple target abundance [11]. The presence of asymmetric-symmetric stem and the loop structure infers the binding of miRNA sequence to the target gene [11]. Computing free energies of duplex structures within each seed and out-seed regions are also essential [11].

1.1.2 *Computational target prediction resources*

Target prediction is a procedure in which miRNA complementarity aligns to mRNA target [11]. Single miRNA can bind to several mRNAs at different positions [11]. Hence, there are massive challenges in predicting miRNA-mRNA interactions as this process is not fully understood [10]. There are several types of experimental approaches for target prediction (microarray, pSILAC, real time PCR, western blot, and luciferase reporter assay), however, these approaches are time-consuming and expensive [10]. Hence, several bioinformatic tools have been developed to overcome these issues. Herein, a summary of top tools for miRNA target prediction is provided (Table 2).

Table 2. Overview of tools for miRNA target prediction.

Tools	Types of feature	Algorithm	Organism
miRanda [12]	Seed match, conservation, and free energy	Local alignments of miRNA:UTR, assessing the thermodynamic folding energy of a miRNA:UTR duplex	Mammals, flies,and worms.
TargetScan [13]	Seed match and conservation	Predicted targets by either the predicted efficacy of targeting (context+ scores) or the probability of conserved targeting (P_{CT}) based on conservation.	Mammals, flies, and worms
MiRTarget2 [14]	Seed types, base composition, and secondary structure	Support vector machine	Mammals and birds
RNA22 [15]	Seed match and free energy	Use pattern discovery to identify target islands and evaluate the free energy of paired target islands and candidate miRNAs	Mammals, flies, and worms.
TargetMiner [16]	Seed match, conservation, free energy, site accessibility, target-site abundance	SVM based classifier identifying potential seed sites between a user-provided miRNA and mRNA of choice.	Any
SVMicro [17]	Seed match, conservation, free energy, site accessibility and target-site abundance	SVM based learning classifier.	Any
PITA [18]	Seed match, conservation, free energy, site accessibility and target-site abundance	Potential site extracted by seed match criteria, undergoes site accessibility by computing a free energy score. In the end, target-site abundance is considered by combining the site accessibility scores for the same miRNA to identify a total interaction score for the miRNA and UTR.	Mammals, flies, and worms.
RNAHybrid [19]	Seed match, free energy, and target-site abundance	Based on free energy by assigning P-value to potential miRNA:mRNA.	Any

Although there are several computational approaches for miRNA target prediction, however, these methods have various limitations such as low accuracy of predictors and less sensitivity for the prediction [10]. Hence, these prediction algorithms are not enough to predict the putative target interactions [10]. Additionally, there are still many unknowns and uncertainties in interactions among these molecules.

1.2 Machine learning technique

Artificial Intelligence (AI) is a wide research area focusing on the development of computational methods to mimic human cognitive behavior and intelligence [20]. Machine Learning (ML) is the sub-field of AI, where computational algorithms optimize performance using the user-provided training dataset and generates a predictive model [20]. ML is widely applied in different areas such as probability, statistics, signal processing, computational mathematics, philosophy, control systems theory, cognitive psychology, biology, economics, and others [21]. The ML is broadly classified into two main learning algorithms such as supervised and unsupervised learning (Figure 5).

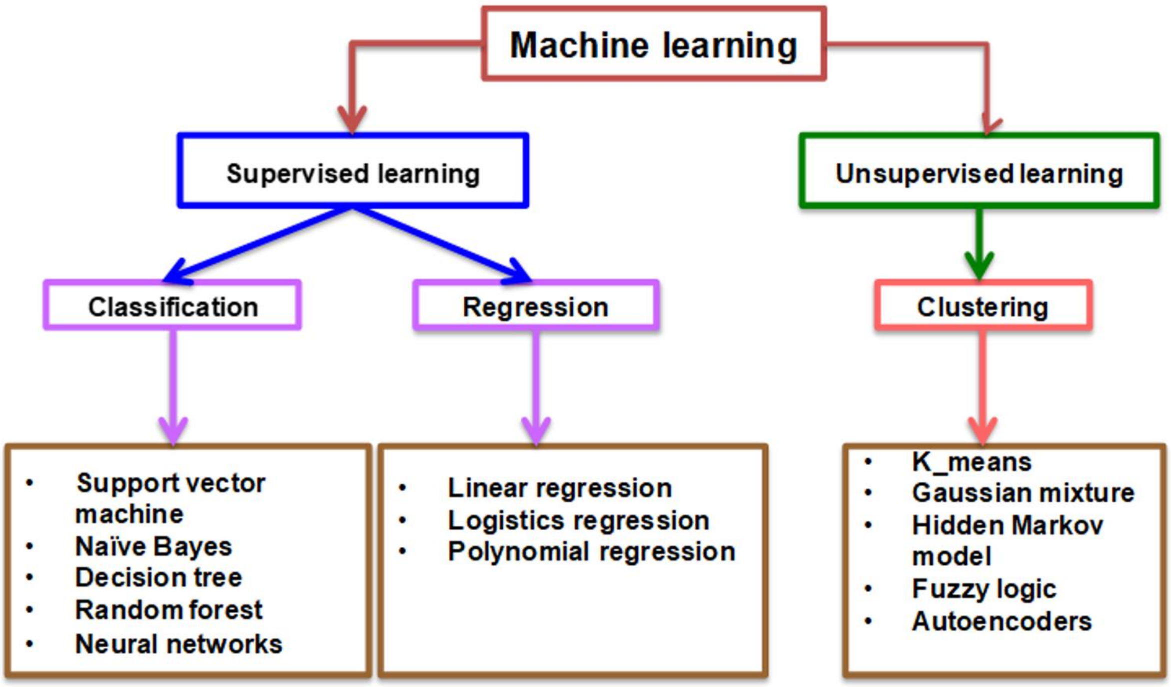

Figure 5. Hierarchical tree of ML. There are two types of learning (supervised and unsupervised) and these learning methods have their own algorithms (clustering is the method of unsupervised whereas classification and regression is the method of supervised). Each method has its corresponding learning algorithms for training.

Unlike traditional algorithms, ML algorithms allow the computer to train on the input data and these algorithms use statistical analysis for predictions [21]. Therefore, ML allows computers building models from sample data to automate decision-making processes based on the input data [21].

1.2.1 *Supervised learning*

Supervised algorithms train mapping functions from the input dataset (a) to the output (Z) by the user-provided training dataset and their corresponding output [22].

$$Z = f(a)$$

The goal is to approximate the mapping function so well that when you have new test data (a) that you can predict the output variables (Z) for that data [22]. Supervised learning further grouped into classification and regression methods [22]. Regression is known to predict a continuous value that tries to predict the continuous output value [22] whereas classification predicts class labels (for instance, 0 class label for non-targets and 1 class label for targets) of new observations [22].

1.2.2 *Unsupervised learning*

Unsupervised learning is used for the training of a model without any prior information about the output dataset [23]. The main objective of unsupervised learning is the construction of a model from the underlying structure or distribution in data [23]. Clustering is the main method of an unsupervised learning algorithm [23].

1.2.3 *Artificial neural network*

Artificial neural network (ANN) is one of the most popular ML algorithms, which is inspired by the neuronal activity of the human brain [24]. ANN consists of highly connected mathematical elements to process information from dynamic state response to input [24]. An artificial neuron is the simplest unit in the neural network, which is known as perceptron and typically correlated with brain neuron cell [24] (Figure 6). A typical ANN is consisting of an input node in the directed graph, which is associated with their corresponding weights that creates connections between output and input nodes [24] (Figure 6).

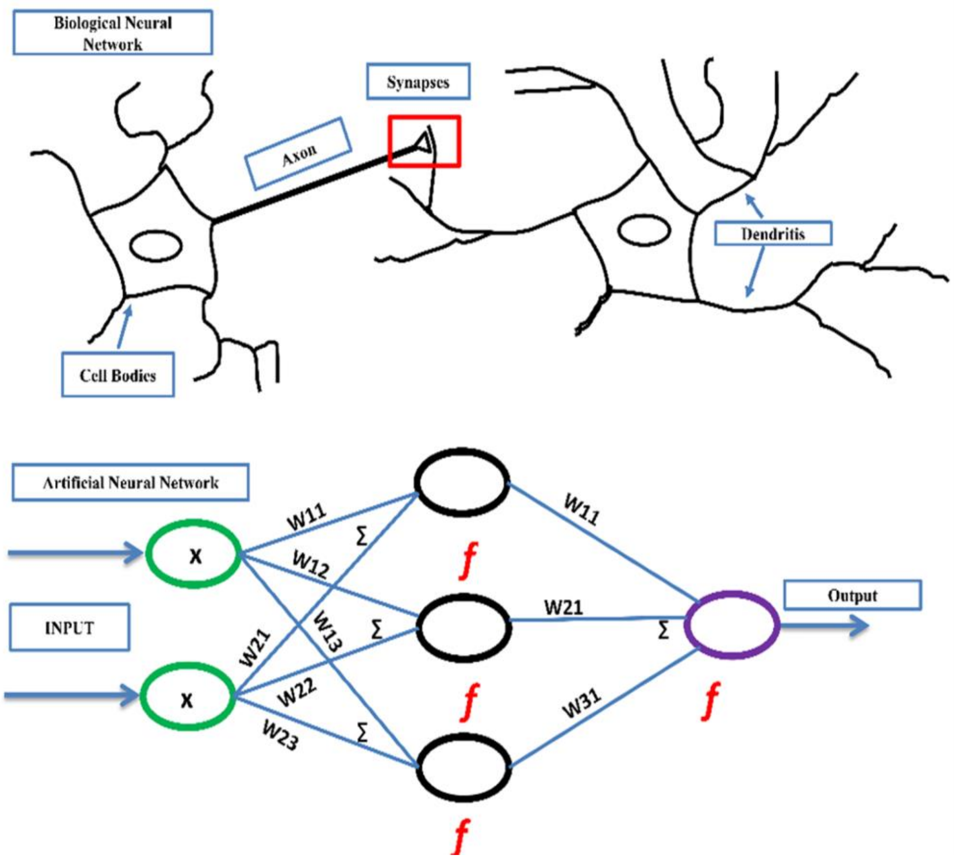

Figure 6. Neural network. The artificial neural network is the graphical representation of a biological neural network and works in the same process of transmitting information from the cell body to synapses.

The ANN receives input in the form of a sequence pattern and an image vector with their corresponding weights. In the ANN, a weight contains information in solving a problem at each node [25]. Each input node multiplied by their corresponding weights and summed up inside the ANN structure [25]. If the sum of weighted input corresponds to zero, bias value is added to make the output a non-zero whereas if weighted input corresponds to a non-zero value, the threshold value is set up and weighted value passes through an activation function to get the desired output [25].

1.2.4 *Deep Learning*

Deep Learning (DL) is the subset of the ML-based approach for extracting features and learning patterns from a large dataset by employing multilayered network [26]. As genomic datasets are increasing day by day, extraction of important information has become the most challenging task in bioinformatics. DL is being used to overcome the problem in handling big datasets [26] (Figure 7). In general, DL has two features: (a) multiple hidden layers (nonlinear processing units), and (b) supervised or unsupervised learning of feature presentations on each layer [27].

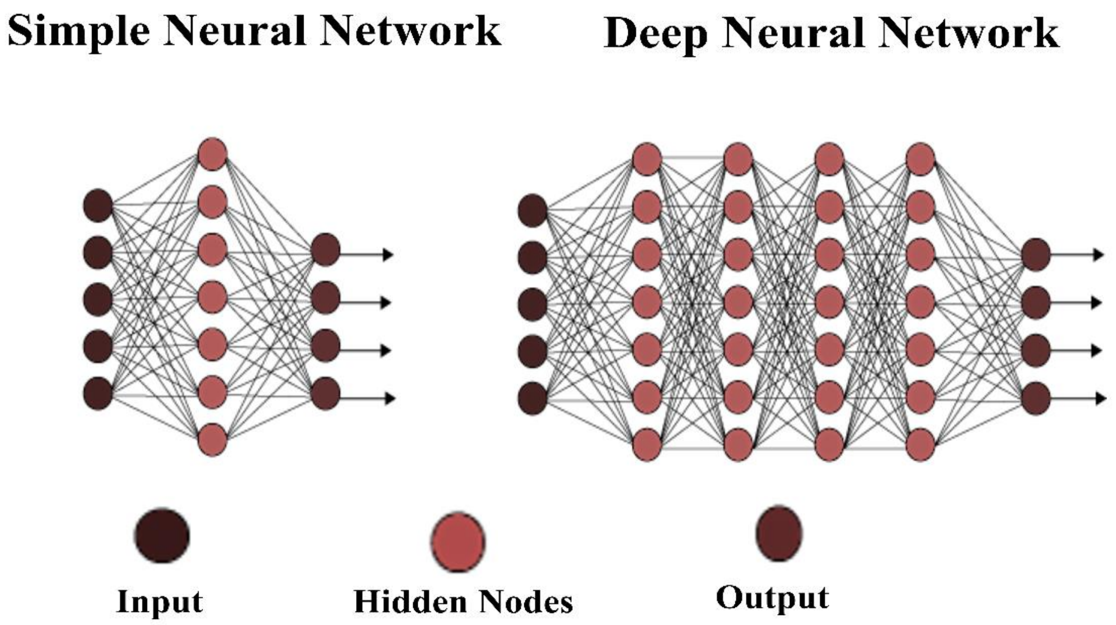

Figure 7. Representation of deep neural network and simple neural network. A deep neural network is an advanced hierarchical technique of simple neural network to make more condense analyses of data. This figure is adopted from [26].

Applications of DL have been primarily focused on image recognition, video, and sound analyses, as well as natural language processing; it also opens doors in life sciences [27]. In the ANN, the network consists of neurons (layers) that are interconnected in adjacent layers to each other [27]. As the nonlinear processing units increase, the network architecture becomes deeper and complex [27]. The DL approach consists of six popular network architecture such as convolutional neural networks, recurrent neural networks, autoencoders, deep residual networks, and deep feedforward networks [28].

a) Convolutional neural network

Convolutional neural network (CNN) is the most popular algorithm in image recognition and natural language processing [29]. The CNN architecture consists of four stages:

- Receiving an input layer from data [29].
- Convolutional layer undergoes preprocessing to reduce the sensitivity of the filters by reducing the noise and other variations [29].
- In the activation layer, preprocessed input signals pass from one layer to another [29].
- In the last stage, all the layers of the network are connected with every neuron from a preceding layer of the neurons from the subsequent layer [29].

b) Recurrent neural network

Recurrent neural network (RNN) is an advanced form of DL technique based on both feedforward and feedbackward network [30]. The RNN approach is mainly used in text data, speech recognition, prediction problems in both classification and regression [30].

c) Autoencoder

Autoencoder is based on the principle of backpropagation in an unsupervised learning environment [31, 32]. This technique has similarities with principal component analysis (PCA) [31, 32]. Autoencoders are easy to train on specialized input data as it does not require any new preprocessing step for the data [31, 32].

d) Deep residual network

Deep residual network (DRN) is an intriguing network consisting of multiple residual layers [33]. Each residual layer consists of sets of activation functions along with their corresponding weights. However, the rectified linear unit (ReLu) is an activation function, which is applied to each layer. This increases their accuracies with the least number of weights [33].

e) Deep feedforward network

Deep feedforward network (DFFN) is a supervised acyclic directed graph that trains from the input dataset. The aim of DFFN is to minimize the error on the prediction task [26, 34]. DFFN is comprises of three layers: an input layer, a sequence of hidden layers, and an output layer. Each layer consists of several nodes and their corresponding weight. At each layer, activation functions are applied to the "sum of weighted inputs" and pass the result to all nodes of the next layer [26, 34]. Mathematically, DFFN information passes through the hierarchical composition of the functions being evaluated from input (x) layer, through the intermediate computations used to define function (f), and generate output (Y) [26, 34].

$$Y = f(x) \approx f*(x)$$

DFFN defines a mapping Y= f (x; θ) and learns the value of the parameters (θ) that result in the best function approximation f ∗ (x) and it is evaluated at different (x) instances (i.e., expected outputs) [26, 34]. These models are called feedforward because information flows through the function being evaluated from (x), through the intermediate computations used to define (f), and finally to the output (Y) [26, 34]. Each interconnection between nodes in each layer is represented by different activation functions [26, 34]. The overall hidden layers along with their corresponding weights are known as the depth of the network [26, 34]. Therefore, "DL" terminology arose from this technique.

1.3 Keras

Keras is an upgraded python library for profound learning with Theano and TensorFlow at the backend [35]. Keras can run using both the central processing unit (CPU) and graphics processing unit (GPU) [35]. Keras has following properties:

(a) Keras is user-friendly application, which reduces the cognitive load [35].

(b) It offers reliable and basic API, which provides actionable hints against errors made by the user [35].

(c) Keras is highly modular in nature. It uses completely configurable modules like neural layers, cost capacities, enhancers, introduction schemes, activations functions, and regularization plans [35]. These modules can easily be configured and modified based on the requirement for construction of new models.

(d) Keras is highly flexible package and one can easily either add new modules or remove existing modules, this feature makes Keras suitable for advanced searches [35].

1.4 Aims of the study

With the discovery of miRNA, the search for the target genes and their role in different cell systems began [1]. A purely experimental approach for scanning target genes and their binding sites is a time-consuming and also an expensive approach [3]. Another possibility is a purely computer-assisted analysis method at the sequence level to find potential interaction partners and their binding sites. In recent years, many prediction programs for miRNA gene interactions have been developed, which started with low accuracy. Through further knowledge in miRNA biology, the accuracy of these programs could be further improved. The main goal of this work is to further improve the accuracy of the prediction of possible target genes using the latest machine learning methods.

The main objectives of this study are:

- to construct a novel deep learning-based approach for increasing the accuracy for miRNA target prediction within the whole human genome.
- to apply an optimized approach to this novel deep learning-based approach for target prediction of large datasets.
- to validate the potential predicted interactions with validated interactions.
- to identify potential candidate miRNAs on top mutated cancer genes.

2. MATERIAL AND METHODS

2.1 Material

When creating sets for model training, it is important not to introduce any biases that could be considered by the model to distinguish between positive and negative instances. This section describes the development of positive and negative datasets for model training and testing.

2.1.1 *Data Collection*

Using the human reference genome assembly version GRCh38.p10 as standard, mature miRNA and mRNA were extracted from *miRBase* v21 [36] and *Ensembl* [37], respectively. To validate the performance accuracy of the classifier model, a verified dataset was extracted from *miRTarBase* v7.0 [38], which is a comprehensive repository of experimentally supported miRNA targets [38]. The *miRTarBase* v7.0 [38] incorporates published information from 8510 research articles resulting in 422,517 experimentally supported miRNA target gene interactions [38]. For each interaction, *miRTarBase* v7.0 [38] also provides direct experimentally verified evidences, such as reporter gene assay, and/or indirect experimental evidences such as a microarray [38].

2.1.2 *Workspace*

A CPU was used with specification as following - Intel® Core™ i7-6800K CPU @ 3.40GHz x 12 with 62.8GB memory, disk 424.6GB and OS type 64-bit.

2.1.3 Platform used

Python version 3.0+ was used for DL training using the following packages:

(a) SciPy or sklearn package [35] (version 0.19.0).

(b) NumPy library [35] (version 1.12.1).

(c) Pandas library [35] (version 0.21.0).

(d) Matplotlib library [35] (version 2.0.2).

(e) TensorFlow [35] (version 1.0.0).

2.2 Dataset generation

This section describes the construction of positive and negative datasets for the training of the model.

2.2.1 *Positive dataset*

A positive dataset was developed by scrutinizing the high-throughput miRNA-target using Watson and Crick method (Figure 8). Seed alignments were obtained from *miRanda* [12] and *TarPmiR* [39] prediction tools. Generated alignments were verified by *TarBase* [40] and *miRTarBase* [38] databases. This dataset was subsequently used for filtering (together with the negative sets) and further training of the classifier model.

2.2.2 *Negative dataset*

A negative dataset was constructed from the binding site region of the human transcript sequence by randomly mutate with a frequency probability of 0.95 with the help of a *python scripting* (Appendix 8.1). Therefore, these mutated regions hinder the binding of the seed region. Generated negative dataset was verified by *TarBase* [40] database as described in Figure 8.

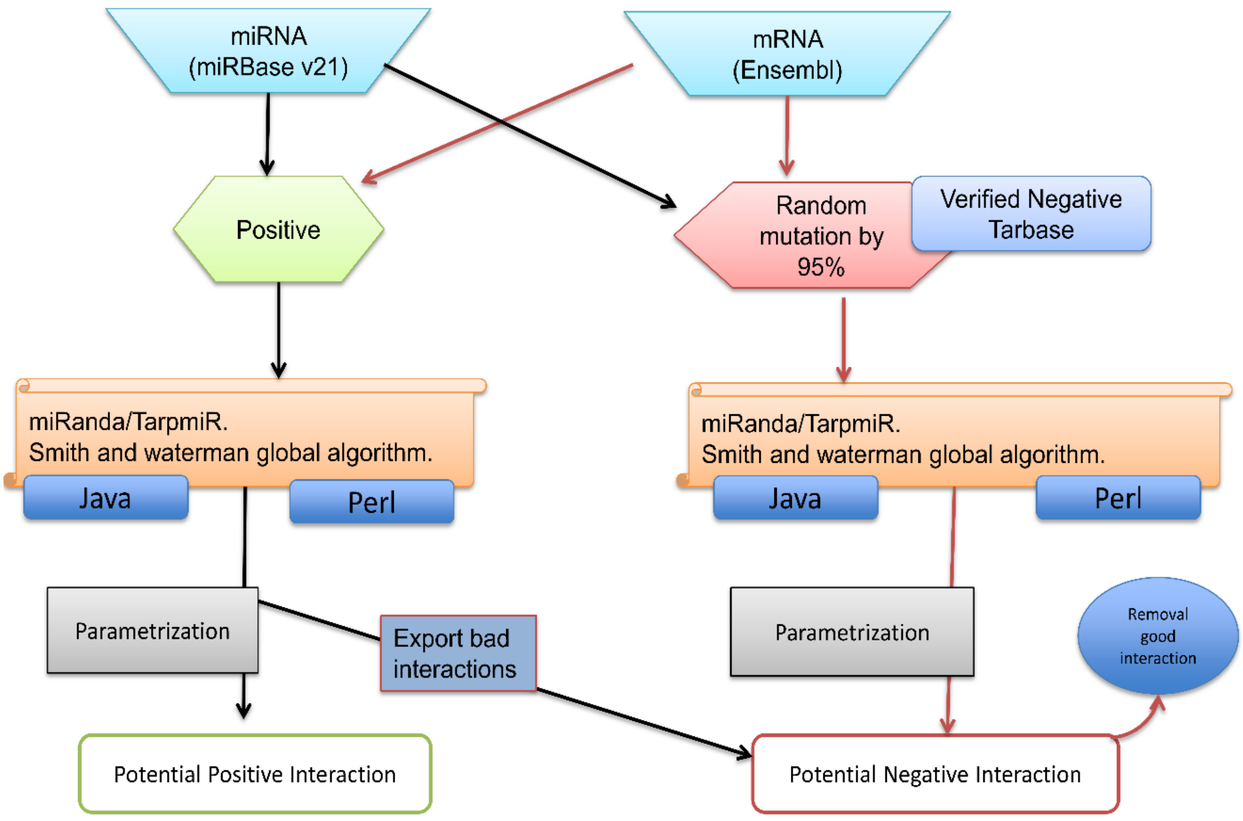

Figure 8. Generation of the two-class label dataset for training. miRNA is downloaded from *miRBase* v21 [36] and mRNA downloaded from *Ensembl* [37] for interaction predictions. Putative features were identified of each interaction. In the parametrization step, a potential two-class (negative and positive) label datasets were generated for training of the classifier model.

2.3 Feature identification for training

In this section, features of miRNA target interactions were divided into three categories such as thermodynamic features, structural features, sequence features, and other features that comprise of energy, binding position, binding sites, accessibility, PyloP flanking and stem regions, and binding probability. A total of 96 features were chosen for the miRNA target prediction model. This section briefly describes feature generation.

a) Seed Scanning: miRNA target interactions are a measure of complementary alignment between the miRNA and the targets. Seed scanning algorithms like *miRanda* [12] algorithm search for initial hits, which can contain a canonical and non-canonical interaction verified by *TarPmiR* [39] algorithm to find perfect seed interaction.

b) Conservations: Conserved sequences are indicative of functional importance upon the feature selection. *TarPmiR* [39] *or ElMMo* [41] have been successfully applied to reduce the number of false positive predictions and it is verified by both *miRanda* [12] and *average PhyloP* [42] from Vienna package [42] for the generation of the conservational score.

c) Free energy: Thermodynamic energy of the duplex secondary structure is commonly utilized by programs such as *RNAduplex* [42] from Vienna package [42]. Further, it was verified by using both *miRanda* [12] and *TarPmiR* [39].

d) Accessibility: Accessibility of binding site in the 3' UTR is critical because miRNAs are assembled in the RISC protein [10]. Several miRNA target prediction tools have been using accessibilities like *miRanda* [12]. Further, it was verified by *RNAplfold* [42] from Vienna package [42].

e) Structural pairing pattern: RNA structure is a critical descriptor involved in target interactions in the seed region [43]. There are 58 instances of structural patterns of interaction such as the number of pairing in seed region, number of loops in symmetric or asymmetric way, number of bulges in seed in symmetric or asymmetric patterns, number of loops in regions outside the seed segment in symmetric or asymmetric way, number of bulges in regions outside the seed segment in symmetric or asymmetric patterns, and distance between start sequence in the seed region to the paired sequence at 5' start of region outside the binding region [43] (Appendix 8.2).

f) Sequence descriptors: Sequences are divided into two regions: (a) the binding region between miRNA and mRNA and (b) the region outside binding region [43]. In both regions, base pairings have been extracted between miRNA-mRNA namely AU content, wobble pairing, and other pairings to check the frequency of nucleotides at each position [43]. These descriptors were calculated by *Java scripts* (Appendices 8.3 and 8.4).

g) Motif related features: This is a probabilistic pairing measurement at a different position of miRNA sequence [39]. At each position, if a pairing between sequences is matching are called as “a” else “e”. Calculated Motif features of interaction of each sequence are different, which is influencing the dimension of the vector space. This probabilistic parameter is calculated by:

$$\frac{m}{e} = \frac{1}{l}\sum_{i=0}^{l} \log\ Q_i$$

Where l is the miRNA sequence length and Q_i is the probability at the position an in i_{th} miRNA sequence. This mathematical representation was calculated by *TarPmiR* [39].

2.4 Pre-processing of the interaction descriptors

In the filtering phase, parameters were applied either in the form of stricter energy-based filtering or quality-based filtering. Hence, the following criteria were used for a good dataset:

a) Replacement of strings values of interactions namely 3’UTR, CDS, 5’UTR and promoter to integer values because the classifier is trained by using integer or floating data points.

b) Handling of missing values is dependent on the problem of the trained classifier [44]. To construct the proper training dataset, this study used imputation and removing unreliable data points from the dataset to handle missing values [44].

c) Not all interactions are energetically in favorable conditions. Therefore, data points are removed based on energy score to set stringent interactions in both datasets and thus decreases the false positive interactions (< -10 *Kcal/mol* for positive and > -9 *Kcal/mol* the for the negative set) [45].

d) Generated interactions have higher than 80% sequence identities in the positive dataset and less than 79% in the negative dataset [46].

e) Conservation score was evaluated from the phylogenetic analysis of all species. High conservation scores (>=150) were removed to decrease negative interactions form positive class or vice-versa [46].

f) In the training phase of the classifier model, the class labels are referred to as seed (i.e., “1” is for the positive group of the dataset and “0” is for the negative group of the dataset).

2.5 AHDR approach

The AHDR approach is based on DFFN approach (section 1.2.4), which is written in python. The input dataset was composed of miRNA target interactions along with their corresponding potential characteristics, which was used to train the AHDR approach. The dataset was split into three sets: training (60% of the dataset), test (20%), and validation (20%). AHDR approach consists of multi-layered architecture with an input layer (an input dataset), a sequence of hidden layers (7 hidden layers are used), and an output layer. Each layer consists of several numbers of nodes and weight (it is connection between a node of one layer to the node of the other layer). Different activation function was applied to each layer of the network (Figure 6). It introduces non-linear properties to the network architecture that converts an input signal of a node to an output. The input dataset passed through each node of all hidden layers, which increases accuracy of each interaction. In order to compare the performances of four different classifier algorithms namely decision tree (DT) [45], Bernoulli naïve bayes (BNB) [43], logistic regression (LR) [46], and support vector machine (SVM) [39] with optimal parameters were considered using scikit package in python. DT is a tree-based model where leaves are the potential solutions and nodes are the decision-making points. LR is the logistic model where the dependent variables are in a binary form. SVM is a non-probabilistic linear classifier where data points are linearly separated by a hyperplane. BNB is a graph-based classification algorithm, where probabilities are assigned to each class. DT is the tree-like graph model where leaves represents the possible solutions of the problem whereas nodes represents the point of the decision. LR is the predictive analysis when the dependent variable is binary in nature. SVM is a discriminative classifier formally defined by a separating hyperplane. BNB classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the bayes theorem.

2.6 Statistical measures

To evaluate the performance of each classifier model, the following statistical measures were used such as accuracy (ACC), area under curve (AUC), sensitivity, specificity, F-score, and Matthews correlation coefficient (MCC), using sklearn package in python. These measures were calculated using true positive (TP), false positive (FP), true negative (TN), and false positive (FP) as defined by equations:

$$ACC = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

$$Sensitivity = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$Specificity = \frac{\mathrm{TN}}{\mathrm{TN + FP}}$$

$$MCC = \frac{\mathrm{TP.TN - FP.FN}}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$F - score = \frac{2PPV.TPR}{(PPV + TPR)}$$

Sensitivity and specificity are mathematical functions that measure the quality of binary classifier model. The ability of algorithms to identify the true targets were analysed by sensitivity, whereas the probability of algorithms that correctly returns a target that is not regulated by the miRNA were analysed by specificity. MCC is a recognized measure that evaluates the quality of binary classifiers between true targets and false targets. F-score is the weighted harmonic mean of both sensitivity and specificity of the test model.

3. RESULTS

To accomplish the aims of this study, a new DL-based framework was developed for miRNA target prediction and it was named as advanced hierarchical deep-rooted (AHDR) framework. Furthermore, the performance of AHDR was compared with other existing prediction tools using a testing dataset. In the end, the AHDR was employed for identification of possible miRNAs interacting with top mutated oncogenes for further validation.

3.1 AHDR framework

Figure 9 illustrates the AHDR framework. In this framework, the initial data is in the form of miRNA and targets derived from *miRBase* [36] and Ensembl [37] and these were filtered out using *miRanda* [12] and *TarPmiR* [39]. Further, the generated interactions were verified by *TarBase* [40].

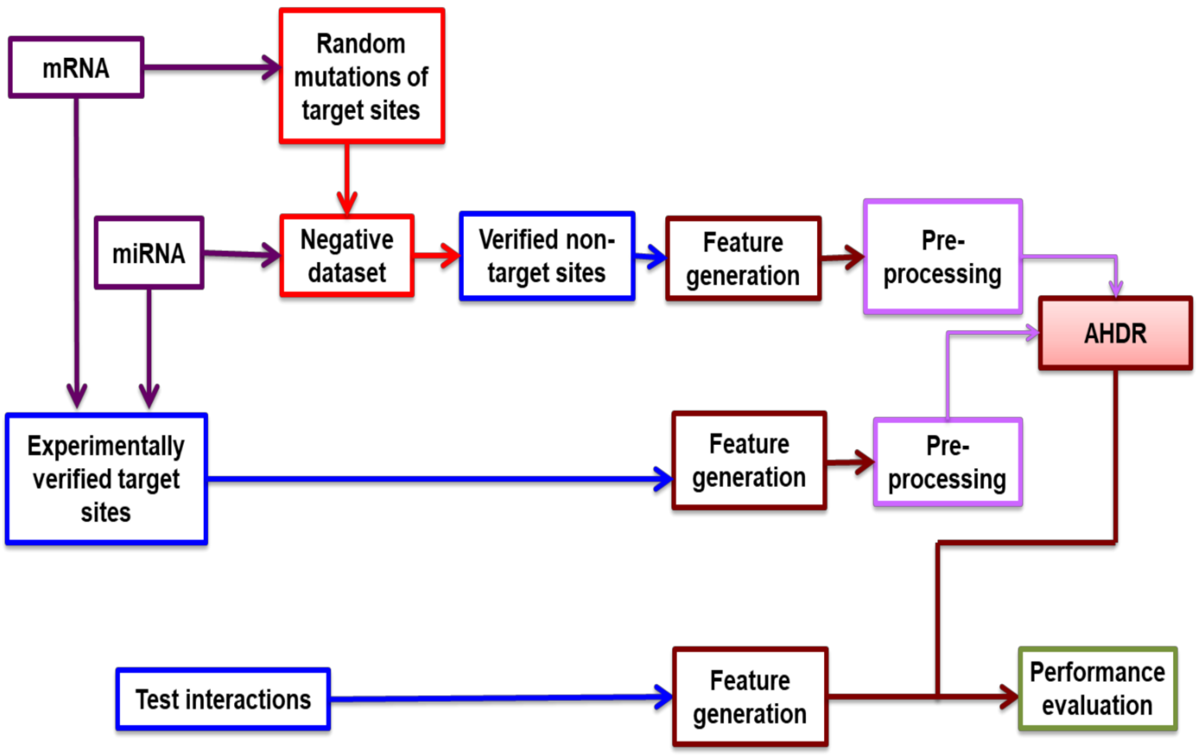

Figure 9. Overview of AHDR framework to elucidate the potential interactions targets of miRNA.

To calculate the interaction descriptors, both flanking regions of seed and their interactions were considered. The generated putative features were applied to build the AHDR model after pre-processing. The test dataset was composed of miRNAs and transcripts, which was used for the detection of the potential target sites (PTS). Finally, the performance evaluation of the AHDR was carried out using statistical tests.

3.2 Feature description of miRNA target interaction

The first step in the DL method is a feature selection. During this study, a total of 95 features were obtained after the pre-processing of the dataset (section 2.4). These 95 features were divided into three different categories as structural, sequence, and thermodynamics features (summarized in Table 3). In the first category, the miRNA target duplex structure is divided into two subdivisions, the alignment (seed) region (5’ until eight nucleotides of the miRNA) and outside alignment (outseed) region (3’ remainder). For both subdivisions the following descriptors are extracted to generate 57 structural descriptors of interactions: (a) number of symmetric loops with lengths 1–7 and those with lengths >7 (eight descriptors), (b) number of asymmetric loops (loops with unequal numbers of unpaired bases on the both strands), (c) number of bulges of lengths 1–7 and those with lengths > 7 (eight descriptors), (d) distance from the start of the seed (the 3’ end) to the first paired base of the 5’ start of the outside alignment part, (e) number of loops (unpaired bases opposite each other between paired bases), (f) number of bulges (inserts on one strand between paired bases), (g) number of paired bases (bp), and (h) number of asymmetric loops with lengths 1–7 and those with lengths >7 (eight descriptors) as summarized in Table 3A.

Table 3. Summary of generated features of miRNA-target interaction.

S.No.	Features	Definition	Source code
A) STRUCTURAL FEATURES			
1	bul_seed_len1	Bulges present in seed region with length 1.	Perl
2	bul_seed_len6	Bulges present in seed region with length 6.	Perl
3	sym_seed_loop_len3	Symmetric loop in seed region with length 3.	Perl
4	sym_outseed_loop_len2	Symmetric loop in seed region with length 2.	Perl
5	sym_outseed_loop_len5	Symmetric loop in seed region with length 5.	Perl
6	asym_seed_loop_len_gt7	Asymmetric loop in seed region with length greater than 7.	Perl
7	asym_seed_loop_len7	Asymmetric loop in seed region with length 7.	Perl
8	bul_seed_len_gt7	Bulges present in seed region with length greater than 7	Perl
9	bul_seed_len2	Bulges present in seed region with length 2.	Perl
10	bul_outseed_len7	Bulges present in out_seed region with length 7.	Perl
11	asym_outseed_loop_len6	Asymmetric loop in out_seed region with length 6.	Perl
12	asym_outseed_loop_len4	Asymmetric loop in out_seed region with length 4.	Perl
13	sym_outseed_loop_len4	Symmetric loop in out_seed region with length 4.	Perl
14	bul_seed_len7	Bulges present in seed region with length 7.	Perl
15	bul_seed_len5	Bulges present in seed region with length 5.	Perl
16	5'nloops	Loops at 5’ end.	Perl
17	bul_outseed_len5	Bulges present in out_seed region with length 5.	Perl

18	bul_outseed_len6	Bulges present in out_seed region with length 6.	Perl
19	sym_outseed_loop_len1	Symmetric loop in out_seed region with length 1.	Perl
20	asym_outseed_loop_len1	Asymmetric loop in out_seed region with length 1.	Perl
21	asym_seed_loop_len5	Asymmetric loop in seed region with length 5.	Perl
22	asym_outseed_loop_len5	Asymmetric loop in out_seed region with length 5.	Perl
23	sym_seed_loop_len_gt7	Symmetric loop in seed with length greater than 7	Perl
24	asym_outseed_loop_len_gt7	Asymmetric loop in out_seed region with length greater than 7.	Perl
25	sym_outseed_loop_len3	Symmetric loop in out_seed with length 3.	Perl
26	sym_seed_loop_len7	Symmetric loop in out_seed with length 7.	Perl
27	asym_seed_loop_len2	Symmetric loop in out_seed with length 2.	Perl
28	5'nblgus	Number of bulges at 5' end.	Perl
29	sym_outseed_loop_len6	Symmetric loop in out_seed with length 6.	Perl
30	bul_outseed_len2	Bulges present in out_seed region with length 2.	Perl
31	sym_seed_loop_len6	Symmetric loop in seed with length 6.	Perl
32	sym_outseed_loop_len7	Symmetric loop in out_seed with length 7.	Perl
33	seed_nblgus	Number of bulges in seed	Perl
34	d5	Distance from 5'.	Perl
35	sym_seed_loop_len2	Symmetric loop in seed with length 2.	Perl
36	bul_outseed_len3	Bulges out_seed region with length 3.	Perl
37	asym_outseed_loop_len7	Asymmetric loop in out_seed region with length 7.	Perl
38	bul_outseed_len1	Bulges out_seed region with length 1.	Perl
39	asym_outseed_loop_len2	Asymmetric loop in out_seed region with length 2.	Perl
40	bul_outseed_len4	Bulges out_seed region with length 4.	Perl
41	sym_outseed_loop_len_gt7	Symmetric loop in out_seed with length greater than 7.	Perl
42	sym_seed_loop_len4	Symmetric loop in out_seed with length 4.	Perl
43	sym_seed_loop_len5	Symmetric loop in out_seed with length 5.	Perl
44	asym_seed_loop_len3	Asymmetric loop in seed region with length 3.	Perl
45	seed_nloops	Number of loops in seed.	Perl
46	bul_seed_len4	Bulges seed region with length 4.	Perl
47	sym_seed_loop_len1	Symmetric loop in seed with length 1.	Perl
48	asym_seed_loop_len1	Asymmetric loop in seed region with length 1.	Perl
49	5'nasymloops	Asymmetric loops in 5'end.	Perl
50	bul_outseed_len_gt7	Bulges seed region with length greater than 7.	Perl
51	asym_seed_loop_len6	Asymmetric loop in seed region with length 6.	Perl
52	seed_nasymloops	Total number of asymmetric loops in seed region.	Perl
53	asym_outseed_loop_len3	Asymmetric loop in out_seed region with length 3.	Perl
54	bul_seed_len3	Bulges seed region with length 3.	Perl
55	asym_seed_loop_len4	Asymmetric loop in seed region with length 4.	Perl
56	Bul_outseed_len1	Bulges in outseed region with length 1	Perl
B) SEQUENCE FEATURES			
1	POS7	Pairing at position 7 in outseed region.	Java

2	Total_TA	TA pairing in outseed region.	Java
3	Total_Mismatces	Total mismatches in out_seed region.	Java
4	AlignLen	Alignment length.	Java
5	Tot_mismatch	Total mismatches in seed region.	Java
6	Total_AU	Presence of AU_content out_seed region.	Java
7	POS9	Pairin g at position 9 in outseed region.	Java
8	AU_content	Presence of AU_content seed region.	Java
9	POS5	Pairing at position 5 in outseed region.	Java
10	Tot_ug	Total GU present in seed region.	Java
11	POS4	Pairing at position 4 in outseed region.	Java
12	POS8	Pairing at position 8 in outseed region.	Java
13	POS3	Pairing at position 3 in outseed region.	Java
14	POS10	Pairing at position 10 in outseed region	Java
15	Tot_Othermismat	Number of other mismatches in seed region.	Java
16	POS1	Pairing at position 1 in outseed region.	Java
17	Tot_Mat	Number of matched in seed.	Java
18	POS2	Pairing at position 2 in outseed region.	Java
19	Total_GC	Number of GC in out_seed region.	Java
20	Tot_gc	Number of GC_content in seed.	Java
21	POS6	Pairing at position 6 in outseed region.	Java
22	Total_GU	Number of GU pairing in out_seed.	Java
C) OTHER FEATURES			
1	LCP	Length of the longest consecutive pairs.	TarPmiR
2	m/e	Probability pairing at different position of miRNA.	TarPmiR
3	PS3’	Difference of paired positions between the seed region and the miRNA 3’ end region.	TarPmiR
4	BRL	Binding region length	TarPmiR
5	align5'	Alignment at 5’ end.	Perl
6	Score	Conservation score.	miRanda
7	binding_probability	Probability of interactions.	TarPmiR/ViennaRNA
8	Binding_pos	Position region of interaction.	miRWalk
9	PLC	Position of the longest consecutive pairs.	TarPmiR
10	NP	Number of pairings in target site.	TarPmiR
11	P3’	Pairing at 3’end.	TarPmiR
12	Seed	Seed region.	miRanda
13	Numirna	Number of miRNAs.	Perl
14	PyloP_Flanking	Flanking conservation	TarPmiR/ PhyloP
15	PhyloP_Stem	Stem conservation	PhyloP
16	Energy	Thermodynamic energy	miRanda/TarPmiR/RNAduplex
17	Accessibility	Measurement of target gene open for miRNA.	miRanda

In the second category, pairing information of the duplex alignment for the first 10 nucleotides is encoded as categorical variables. These variables are namely, a) GC pair, b) AU pair, c) G:U wobble, d) mismatch and e) gap of duplex structure. Furthermore, the pairing information is summarized over the alignment region, the 3′ region of the miRNA, and the total miRNA region. This includes a total of GC matches (Tot_GC) in the seed region, whereas alignment outside the seed region named as Total_GC (Table 3B). In the third category, remaining descriptors are taken from other predictor algorithms (Table 3C).

3.3 Feature selection for miRNA target prediction

Feature selection is a process for the selection of an optimal number of variables, which is responsible for the model construction [47]. It is a pivotal aspect that makes the model more efficient by eliminating redundant variables and shrinking the size [47].

For the selection of optimal features, five algorithms of feature selections were applied to the generated 95 features (Table 3) for the training purpose. These five algorithms are the least absolute shrinkage and selection operator (LASSO), randomized logistics, chi-square, variance threshold, and decision tree-based selection. LASSO gave 19 putative descriptors while the variance threshold has selected 70 descriptors (Figure 10).

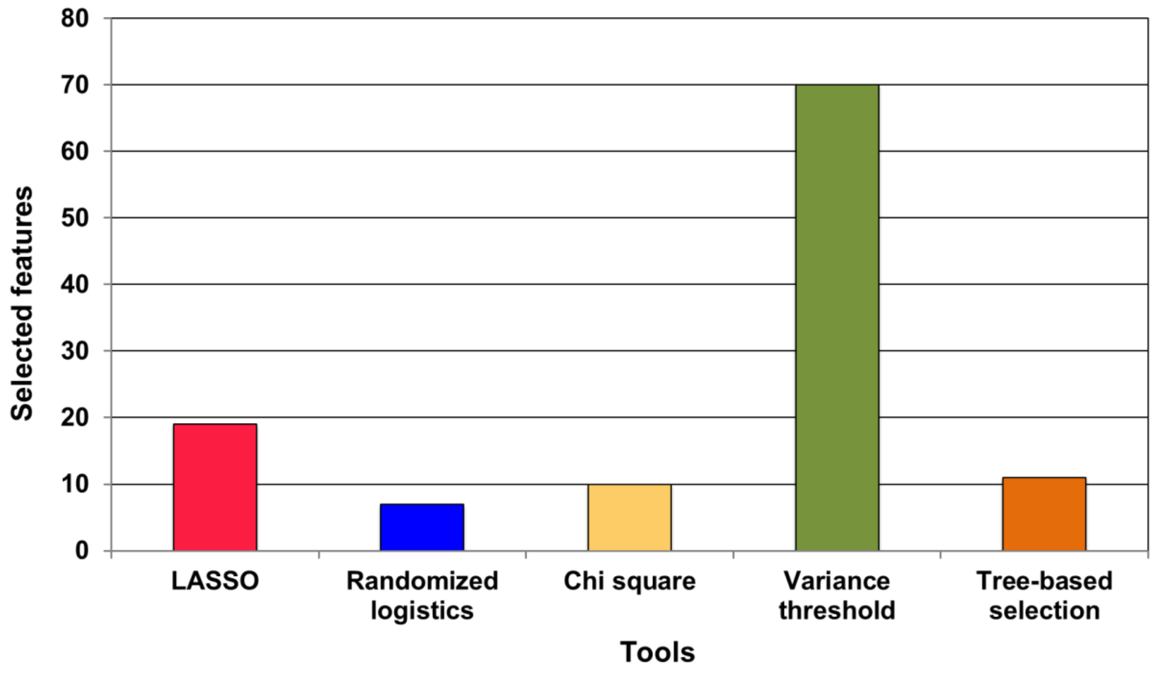

Figure 10. Overview of feature selection by different feature selection algorithms. For the selection of relevant features, five different approaches have been applied. These are LASSO (red box), randomized logistics (blue), Chi square (yellow), variance threshold (green), and tree-based approach (orange box).

In contrast, randomized logistics yielded only seven descriptors. In this study, the LASSO algorithm was used for the training of the classifier. LASSO is a powerful method for selecting relevant features because it performs two main steps, regularization and variable selection [48]. In the regularization step, it penalizes coefficients of regression variables shrinking them to zero. In the variable selection step, features with a non-zero coefficient after the regularization are selected to be part of the model [48]. Therefore, the main goal of the LASSO is the reduction of the prediction error (Figure 11).

LASSO has selected 19 putative features with their corresponding embedded score as shown in Figure 10. These features are: a) alignment length in seed (AlignLen), b) energy, c) conservation score, d) total matching in seed (Tot_Mat), e) GC count in seed (Tot_gc), f) total mismatch in outseed (Tot_mismatch), g) total AU count in outseed (Total_AU), h) total GC count in outseed (Total_GC), i) alignment at 5' (align5'), j) binding region length (BRL), k) distance from 5' (d5), l) the longest consecutive pairings (LCP), m) motif (m/e), n) mRNA, o) miRNA, p) number of pairing (NP), q) number of paired bases (numirna), r) pairings in 3'end (P3'), and s) position of the longest consecutive pairings (PLC). The majority of these features (selected by the LASSO) are sequence-based features (6 out of 19) and other types of features (10 out of 19).

Based on the embedded score, top four features (out of 19 selected features) are energy (0.99), P3` (0.86), m/e (0.81), and miRNA (0.69), while the embedded score remained 0.63 from 10th-19th selected feature (Figure 10). Since thermodynamic energy, pairing at 3', and m/e are top features based on the embedded score, however, other feature selection algorithms did not pick these features. An example of the input dataset for the training of the AHDR classifier is shown in appendix 8.5.

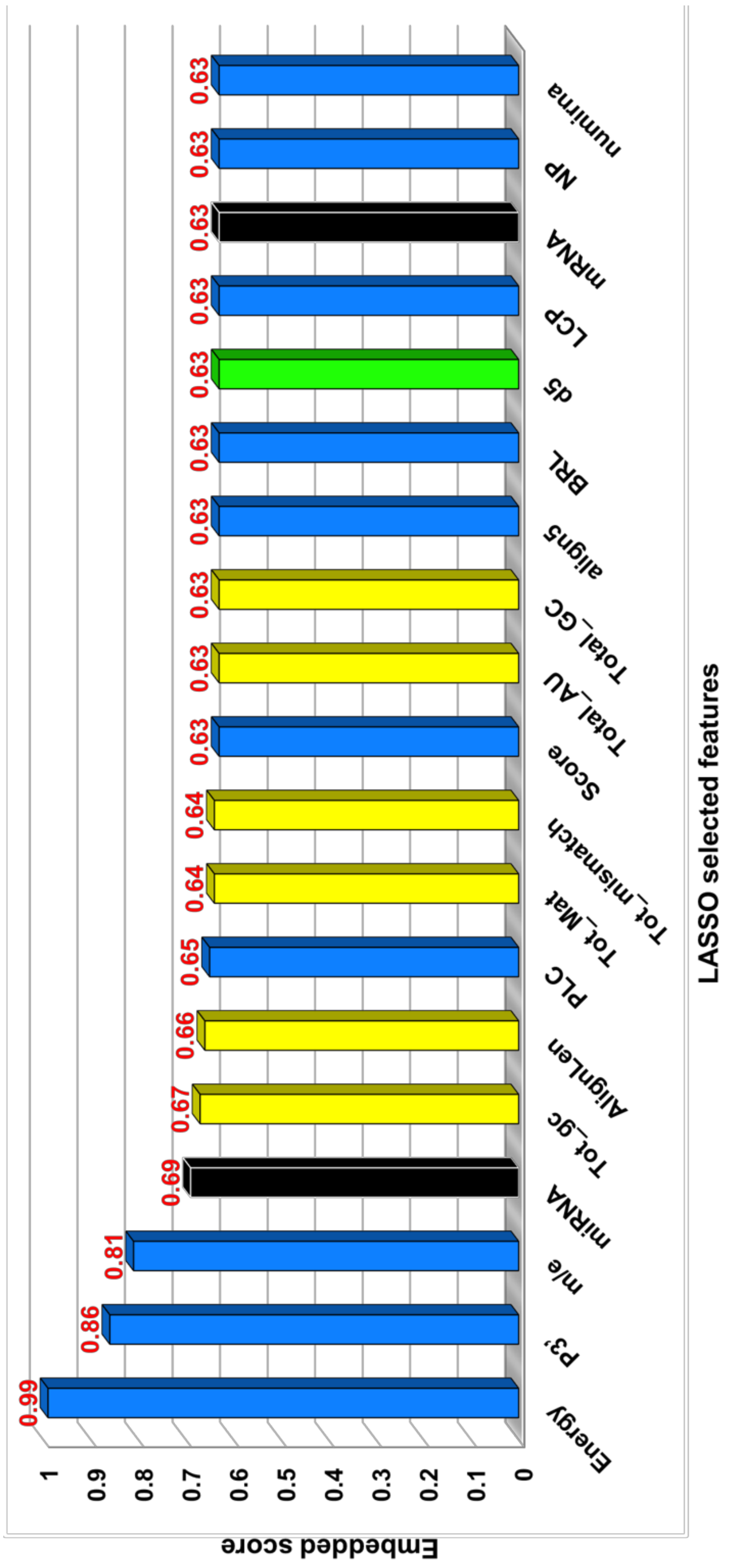

Figure 11. Embedded score of selected potential feature generated by the LASSO algorithm. Top two selected categories are sequence and other features (Table 3). Different colors design types of features like black, yellow, green, and blue for miRNA/mRNA, sequence, structure, and other features, respectively.

3.4 Hyperparameter optimization

The optimization of hyperparameters of the classifier is crucial for the DL approach. It generates the best network architecture by maximizing the accuracy of the validation dataset through optimal hyperparameters. AHDR is based on DFFN architecture. It comprises several hyperparameters, including the number of hidden (*noh*) layers, the number of nodes (*non*) in each layer, a pre-training rate (*ptr*), a pre-processing method (*ppmd*) and the batch size (*bs*) through dropout approach (Table 4). ReLu activation function is used for eight layers, while the sigmoid activation function is applied at the 9th layer (Appendix 8.6).

The training epochs are set to 16 for the AHDR approach. Therefore, the AHDR architecture resulted in [9, 64, 128, 128, 64, 64, 64, 64, 32, 16] as given in Table 4, where 9 represented the *noh* followed by nodes at each layer and 16 represented the number of units (node) of the output layer. AHDR was composed of nine dense hidden layers while the output layer comprised 16 sigmoid output nodes. The shape of the AHDR was consistent with its intended functionality with three types of layers as following:

(a) The first layer increases the dimensionality of the prediction problem allowing the representation of the dataset in a more complex dimension (over-completion). This layer does not necessarily improve the efficiency of AHDR.

(b) Middle layers (from one to six) aim to identify the relevant features represented in the data; they correspond to the first half of a network. These layers were pre-trained to learn the features that are the most representative of miRNA target interactions.

(c) The last three layers were responsible for classifying the features learned by AHDR, and it followed the typical shape of the DFFN classification network.

Table 4. The optimal hyperparameters in the AHDR model architecture.

Hyperparameters	**Model Structure**
Number of hidden (*noh*) layer	9
Number of node (*non*) in each layer	64, 128, 128, 64, 64, 64, 64, 32, 16
Computational parameters (weights)	3584, 8256, 8320, 16512, 8256, 4160, 4160, 4160, 2080
Pre training rate (*ptr*)	0.001
Batch size (*bs*)	128
Preprocessing method (*ppmd*)	Label encoder

The selection of optimized hyperparameters of other ML algorithms such as AHDR, DT, BNB, LR, and SVM are summarized (Table 5).

Table 5. The optimized hyperparameters of each machine learning methods.

Algorithm	Package	Optimized hyperparameters
AHDR	Keras	lr = 0.01, hl = 9, loss = 'binary_cross entropy', hln = [128, 64, 128, 128, 64, 64, 64, 32, 16], optimizer = 'rmsprop', Output layer activation function = 'sigmoid'.
DT	Scikit-learn	class_weight = balance, max_depth = None (default).
BNB	Scikit-learn	Alpha = 1.0.
LR	Scikit-learn	Penalty = L2, solver = Liblinear.
SVM	Sklearn	Gamma = Auto, kernel = rbf, degree = 3.

The parameter alpha of BNB is optimized in the range of 0 (worse) to 1 (best) with an interval of 0.1 and is set to 1.0. In the DT, the max_depth of the tree is set to default means nodes are expanded until all leaves (output nodes) are pure or until all leaves contain less than min_samples_split (default =2) samples. While the class weight of the tree is set to be balanced because it automatically adjusts weights inversely proportional to class frequencies in the input data (Table 5). The parameters of the LR such as penalty is set to regularization (L2), as it handles both dense and sparse input dataset, whereas the solver is set to liblinear, as it supports both L1 and L2 regularization. The number of optimized parameters of SVM includes gamma selects auto_deprecated (uses 1/n features i.e., no explicit value of gamma was passed), kernel selects radial basic functions (rbf), and degree is set to 3 (Table 5).

3.4 Performance evaluation of other ML methods

Generally, four evaluation metrics are used, namely AUC, ACC, true positive rate [TPR, sensitivity/recall], and false positive rate [TNR, specificity] in the statistical evaluation [49]. The AUC value ranges between 0.5 to 1 where 0.5 denotes a bad prediction algorithm and 1 denotes a good algorithm. Calculations of ACC, TPR, and TNR were as follows:

$$ACC = (TP+TN)/(TP+TN+FP+FN)$$

$$TPR = TP/(TP+FN)$$

$$TNR = TN/(TN+FP)$$

where TP, FP, TN, and FN represent true positive, false positive, true negative, and false negative, respectively. In two class label prediction, the outcomes are labeled either as positive or negative [50]. TP is the output where both its validated and predicted labels are positive, FP involves a predicted label being positive and the validated label being negative, TN involves the predicted and validated labels both being negative, and FN involves the predicted label being negative and the validated label being positive [49].

In addition, Matthews correlation coefficient (MCC) and F-score were also used to assess the model performance [50]. The calculations of MCC and F-score were as follows:

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$PPV = TP/(TP+FP)$$

$$F = 2PPV.TPR/(PPV+TPR)$$

where PPV is positive predictive value, also known as precision. Essentially, the MCC is a correlation coefficient between the validated label interactions and its predicted labels [49, 50]. The value of MCC ranges between -1 and +1, where a coefficient of +1 represents a perfect prediction, 0 represents random prediction and −1 indicates total disagreement between predicted and true labels. F-score can be interpreted as a weighted average of the PPV and TPR, where an F-score reaches its best value at 1 and the worst at 0 [49, 50].

To evaluate the performance of different ML methods with AHDR, the dataset was divided into three subdivisions consisting of a training set (60% of the original dataset), a validation set (20%) and a test set (20%). The training set was required for the training of the AHDR classifier, the validation set was utilized for the setting of hyperparameters and the test set was used for the evaluating performance of each classifier.

Five commonly used ML classifiers, including LR, BNB, SVM, DT, and AHDR, were considered. Compared with LR and other ML classifier, which is the most frequently used approach in miRNA target prediction, AHDR has the best performance as evident from AUC, ACC, sensitivity, specificity, F-score, and MCC values of 0.98, 0.98, 0.97, 0.90, 0.98, and 0.96 respectively (Figure 12).

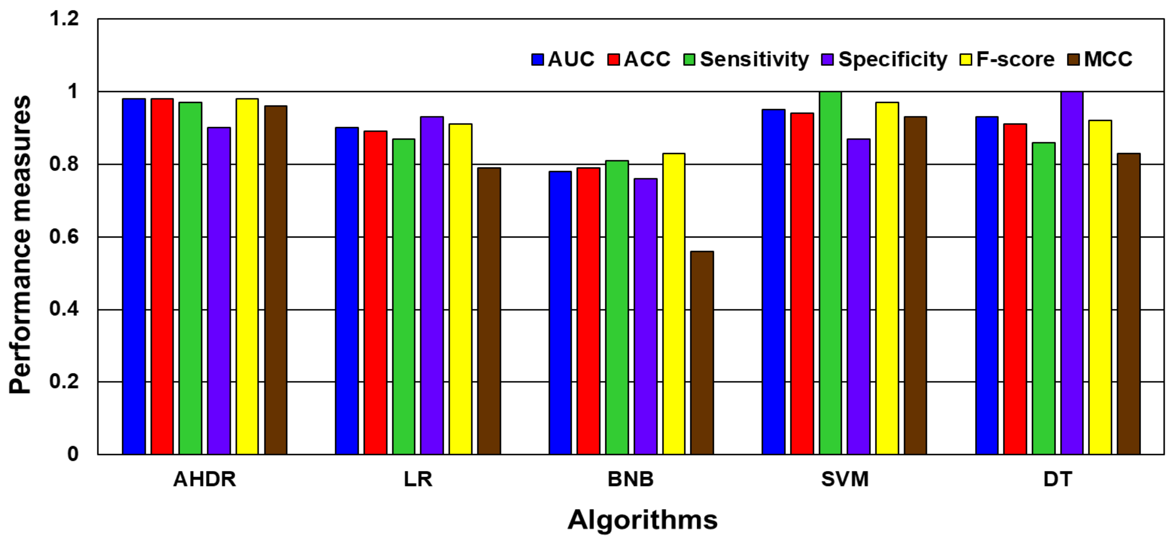

Figure 12. AHDR performance predictions on test dataset with other ML algorithms.

The performance evaluation is indicating that the DL network exhibited improved ability to learn in AUC, ACC, sensitivity, specificity, F-score, and MCC.

To avoid biases, cross-validation is the cutting-edge method in DL, which estimates the skill of the model from new data [24]. In the 10-fold cross-validation, all the known interactions were randomly divided into 10 subsets with equal size. In each fold, one subset was left out as testing samples, and the remaining four subsets were treated as training sets. The entire procedure was repeated until the entire subset was used for training. The average performance of the 10-fold cross-validation was adopted for evaluation [24]. The prediction measures of the 10-fold cross-validation are depicted in Figure 13.

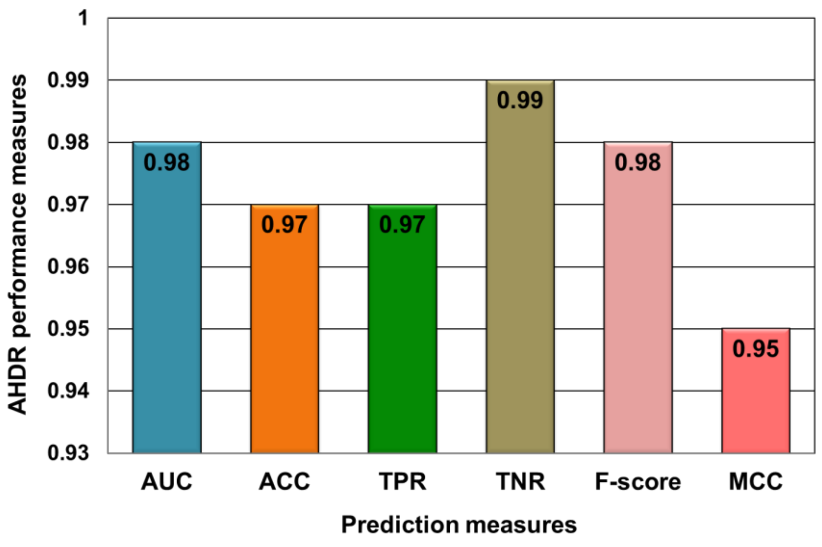

Figure 13. AHDR performance after 10-fold cross-validation. Performance evaluation after 10-fold cross-validation, TNR (brown box plot) outperformed among other performance matrices i.e., the model measures the proportion of actual negatives correctly identified.

After the 10-fold cross-validation, performance measures of AHDR are represented in terms of AUC, ACC, sensitivity, and F-score as 0.98, 0.97, 0.97, and 0.98, respectively (Figure 13). As compared to performance measures in the training phase of AHDR, ACC results in a decrease from 0.98 to 0.97 and MCC value decrease from 0.96 to 0.95 whereas TNR value increases from 0.90 to 0.99. AHDR not only gives accurate predictions on the training data but also gives the best prediction measures from the new data and therefore this procedure avoids overfitting (statistical classifier captures the noise of the data) and underfitting (statistical classifier cannot capture the underlying trend of the data) in the AHDR.

3.5 Performance comparison with other prediction methods

The test dataset was extracted from *miRTarBase* [38] for the comparison of the prediction performance of ADHR with other miRNA target prediction tools [38, 40]. This dataset was composed of 200 verified miRNA and 26315 verified interactions. Eight existing target prediction algorithms were chosen for the comparison of their prediction performance, namely *TargetScan* [13], *miRmap* [51], *MicroT* [52], *comiR* [53], *miRSearch* [54], *miRSystem* [55], *PITA* [18], and *PicTar* [56]. Figure 14 illustrates the performance of different algorithms on the test dataset. Results of the comparison regarding AUC, ACC, sensitivity, and F-score, which is the quality measure of the binary classification (Figure 14).

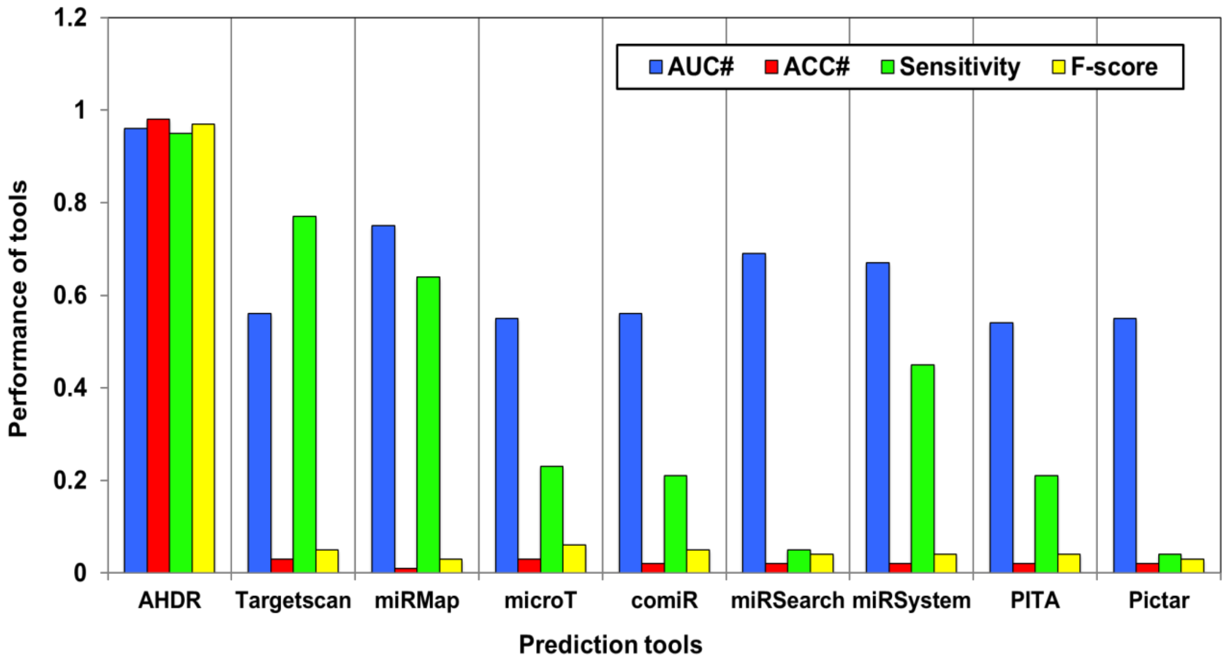

Figure 14. Performance measurement with existing prediction algorithms. AHDR is compared with the other tools in terms of AUC, ACC, sensitivity, and F-score performance matrices. The AUC values range between 0.5 (bad prediction) to 1 (good prediction). F-score measure harmonic mean of sensitivity that lies between 0 (worst score) to 1 (best score). These scores indicating that the AHDR method is the most effective at miRNA-target prediction based on learning approach.

AHDR provides the best AUC among the four predictors (0.96), which is a ~21% increase to the second-best performing predictor, *miRMap* [51] (0.75). The sensitivity of *TargetScan* [13] (0.77) is lower than that of the AHDR (0.95), which indicates the measure of correct miRNA target interactions. ACC parameter confirms the effectiveness of the AHDR (Figure 14).

Instead, the AHDR provides the most balanced results in terms of performance matrices as compared to the other tools (Figure 14). This is evident from its high F-score value (0.97) compared to the other predictors (Figure 14). Overall, the AHDR has outperformed other predictors in terms of performance matrices (Figure 14). The MCC performance method measures the quality of the binary classification in ML (section 3.4). The MCC value of the AHDR shows 0.95 (Figure 15), which is indicative of the best prediction, whereas other existing algorithms have negative values of MCC, ranging from -0.46 (for *TargetScan* [13]) to -0.95 (for *Pictar* [56], Figure 15).

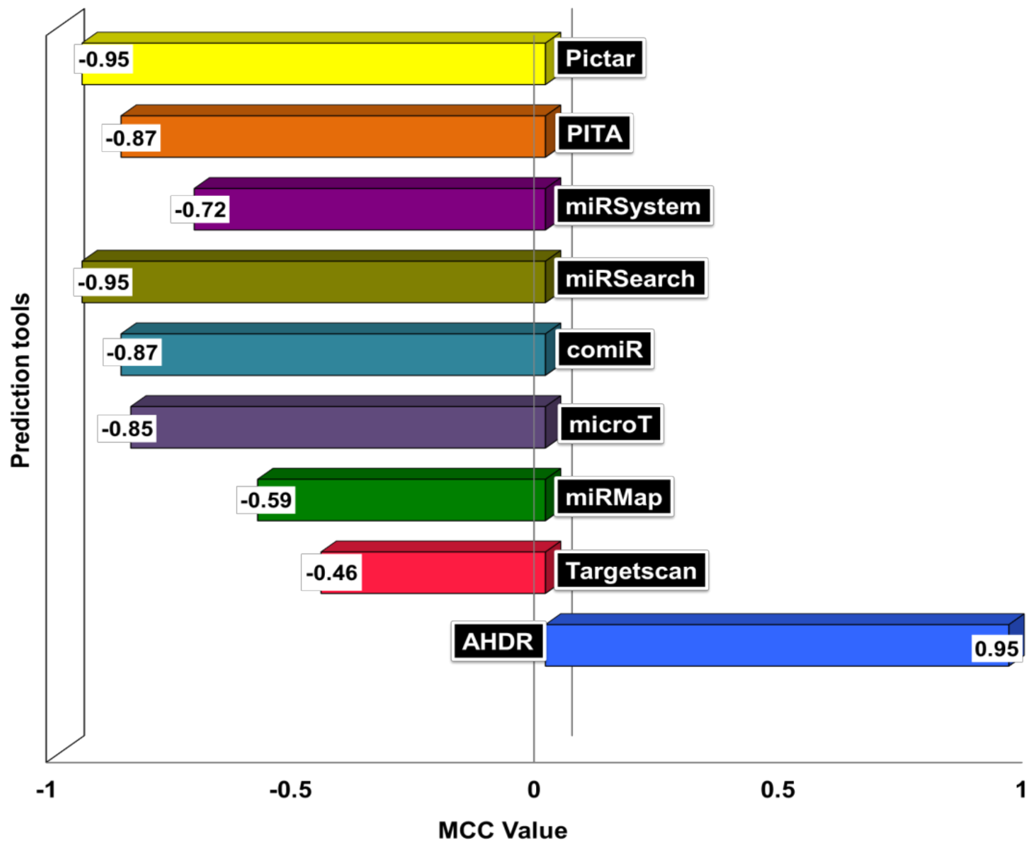

Figure 15. Performance comparison with existing prediction algorithms based on the MCC value. MCC is a correlation coefficient used to measure the quality of the two-class label dataset. Therefore, MCC value of +1 indicates a perfect prediction, whereas 0 no better than random prediction and −1 represents the total disagreement of the prediction.

3.6 Interaction identification with validated targets

Furthermore, the predicting power of the AHDR was evaluated using an experimentally validated miRNA target interaction in comparison with other known algorithms. For this purpose, a total of 26315 validated miRNA target interactions were retrieved from strong experimental evidences such as Luciferase reporter assay, a reporter assay, and microarray [38] as summarized in Appendix 8.7. These data were used for evaluation of the AHDR along with eight other commonly used miRNA target approaches, including *TargetScan* [13], *miRMap* [51], *microT* [52], *comiR* [53], *miRSearch* [54], *miRSystem* [55], *PITA* [18], and *Pictar* [56] (Figure 16).

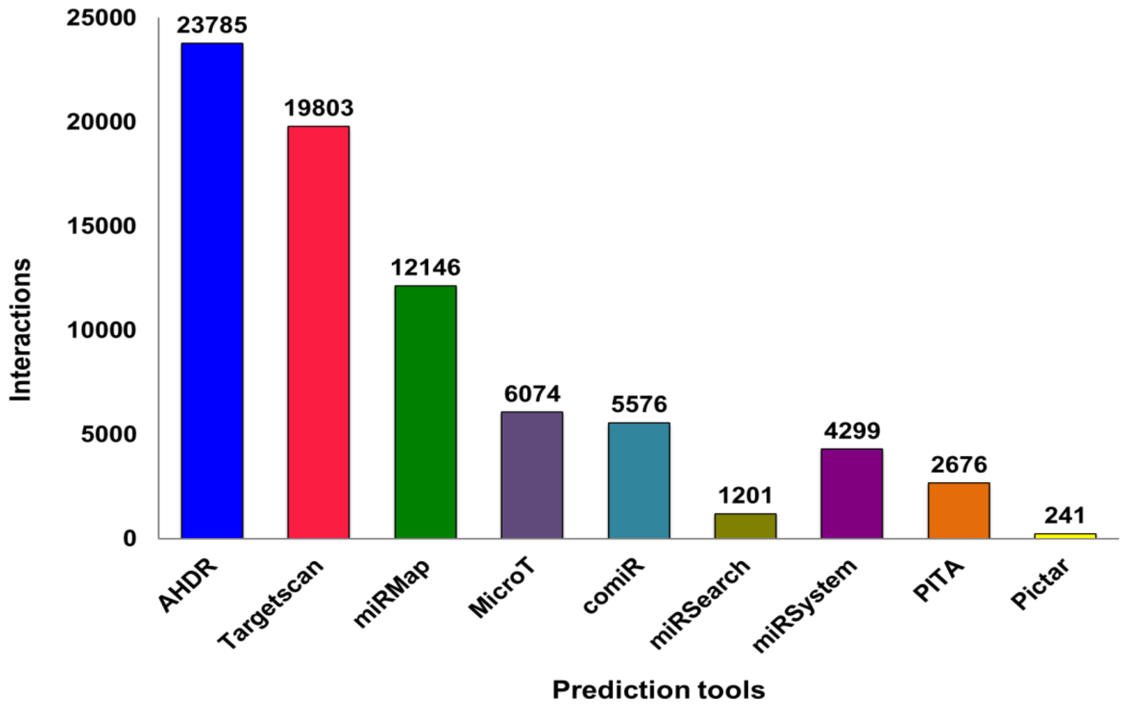

Figure 16. Number of overlapping interactions on validated targets. To evaluate the performance of tools based on overlapping, 26315 verified interactions were retrieved from *miRTarBase* [38]. AHDR shows the large number of overlapping interactions followed by *TargetScan* [13], *miRMap* [51] whereas *Pictar* [56] shows the least number of overlapping interactions.

AHDR has identified 23785 miRNA target interactions, which comprise over 90% of all predictors. Other two top predictors were *TargetScan* [13] and *miRMap* [51], which have identified 19803 and 12146 interactions, respectively (Figure 16).

3.7 Performance of miRNA-target prediction of AHDR on top 20 cancer genes

To evaluate the performance of the ADHR for miRNA target prediction flanking critical genes, a list of top 20 critical genes associated with cancer was collected from a recent study focusing on comprehensive characterization of cancer driver genes derived from 9,423 tumor exomes [57]. A comparative analysis of the ADHR and other target prediction tools using this dataset was performed. Out of 20 oncogenes, only 15 genes were found in the current datasets, which have interactions with miRNA (Figure 17).

Gene	TargetScan	miRSystem	miRmap	miRSearch	PITA	MicroT	PicTar	CoMir	AHDR	AHDR (miRNAs)
CTNNB1	4	1	0	1	0	0	0	1	4	*-4474-3p, *-3688-5p, *-532-3p, *-4765
DICER1	5	4	6	1	2	2	0	1	6	***-519b-3p**, **let-7b-5p**, ***-519a-3p**, *-586, ***-4282,** *-106a-3p
EGFR	3	3	3	2	2	2	0	3	4	*-133a-3p, ***-146a-5p**, *-574-3p, ***-30a-5p**
HRAS	0	1	0	0	0	0	1	0	2	**let-7b-5p**, ***-603**
KRAS	1	3	0	1	1	1	1	2	3	*-487b-3p, **let-7b-5p**, *-96-5p
NRAS	3	1	3	0	0	0	0	1	3	*-2276-3p, *-32-5p, **let-7b-5p**
PIK3CA	0	0	1	0	0	0	0	1	1	***-375**
PIK3R1	4	2	3	0	2	2	0	2	4	*-448, *-8485, ***-1277-5p**, ***-603**
PTPN11	2	2	2	2	1	1	0	3	4	*-6793-3p, ***-489-3p**, *-23a-3p, *-4659a-3p
RHOA	4	0	1	1	0	0	0	0	6	*-6877-5p, ***-6499-3p**, ***-483-3p**, *-6768-3p, ***-375**, ***-6778-5p**
SF3B1	2	0	0	0	0	0	0	1	2	***-2276-3p**, *-4326
SMAD4	2	5	5	0	1	1	0	0	6	***-519b-3p**, ***-483-3p**, ***-146a-5p**, *-4799-5p, *-449a, ***-519a-3p**
TP53	12	2	8	1	3	3	0	3	17	*-6780b-5p, ***-489-3p**, *-2110, let-7a-3p, *-4793-3p, *-6797-5p, ***-375**, ***-1277-5p**, ***-6778-5p**, *-1301-3p, *-1273f, *-8071, *-6751-5p, ***-30a-5p**, *-1207-5p, *-6880-
VHL	6	1	3	0	1	1	0	0	6	***-6499-3p**, ***-2276-3p**, *-1248, *-496, *-6787-3p, ***-4282**

Figure 17. miRNA-target interactions in top cancer genes computed by several miRNA-target predictors reveals that AHDR is outperforming other tools. Some of miRNAs are interacting with more than one oncogene, are marked in different colors. CTNNB1 - Catenin beta 1; DICER1 - Dicer 1, Ribonuclease III; EGFR - Epidermal growth factor receptor; HRAS - HRas Proto-Oncogene; KRAS - KRAS Proto-Oncogene; NRAS - N-ras oncogene; PIK3CA - Phosphatidylinositol 3-kinase; PIK3R1 - Phosphoinositide-3-Kinase Regulatory Subunit 1; PTPN11 - Protein Tyrosine Phosphatase Non-Receptor Type 11; RHOA - Ras Homolog Family Member A; SF3B1 - Splicing Factor 3b Subunit 1; SMAD4 - SMAD Family Member 4; TP53- tumor protein p53; VHL - Von Hippel-Lindau Tumor Suppressor. * =miR

Overall, the ADHR is outperforming all other tools in miRNA-target prediction (Figure 16). For example, the ADHR has picked up 17 miRNA interactions for tumor protein p53 (TP53) gene whereas other existing tools have shown lower interactions like twelve by *TargetScan* [13], eight by *miRmap* [51] and three each by *PITA* [18], *MicroT* [52], and *comiR* [53] (Figure 17). Moreover, these known tools did not pick even a single interaction as marked in the red shade in Figure 17. Some miRNAs are interacting with more than one oncogene (marked by colors in Figure 17) like let-7b-5p and miR-375.

4. DISCUSSION

Since the discovery of miRNAs, their roles in several physiological conditions and human diseases have been attributed towards novel therapeutic targets [1]. At the same time, the race for miRNA target prediction has begun, which is a challenging task as each miRNA has multiple targets and vice versa [2]. Therefore, the correct identification of miRNA-target interactions remains a challenge. The way to address this problem is usually experimental validation of these miRNA target interactions, but the cost and timeline are limiting factors [2]. Hence, computational applications are developed for miRNA target prediction.

Generally, computational approaches for prediction of miRNA targets are grouped into two categories: *ab initio* and ML-based methods [58]. The first category, *ab initio* methods are based on the base pairings within the seed region for miRNA targeting, which has been established from experimental methods. Tools such as *TargetScan* [13], *PITA* [18] and *PicTar* [58], use computational algorithms to scan positions with the miRNA sequence and scoring functions to filter target sites. These tools prioritize the removal of false positive interactions from the predicted target interactions; however, these algorithms have the drawback of omitting miRNAs from the results. Hence, this may be leading to higher numbers of false negative interactions [58]. In this sense, *TargetScan* [13] is a sequence-based tool, which leads to low sensitivity (Figure 14). Early *ab initio* methods were built on features like the complementarity of the miRNA and/or the free energy of the miRNA/mRNA duplex and these are statistically derived from limited experimental evidences [58]. Currently, due to the increase in data reported from different assays, especially those involving next-generation sequencing (NGS) and CLIPseq datasets, more comprehensive and valuable features have been discovered and implemented in several *ab initio* methods.

The second category is based on ML methods that improve the accuracy of the prediction algorithm in the tools, such as *miRMark* [59], *MBSTAR* [60], *TargetMiner* [16] and *TargetSpy* [46]. ML approach is different from *ab initio* methods by using interaction descriptors as input features to machine learning models. Early ML algorithms, such as LR, RF, SVM, and DT have been frequently used (*NBmiRTar* [61], *miTarget* [62], *Genmir ++* [63], and *HuMiTar* [64]) in miRNA target predictions. *NBmiRTar* uses the *miRanda* output to train naïve Bayes (BNB) classifier [61]. The generated post-processed dataset of *NBmiRTar* contains artificial target sites without any experimental validation. *miTarget* uses SVM with the kernel of the rbf, and the positional, structural, and thermodynamic interaction characteristics for prediction [62]. Likewise, *TargetMiner* also uses the SVM with the RBF kernel, and the dataset contains validated positive interactions and unarticulated negative interactions [16]. Although diverse approaches have been proposed, the problem of high false positive interactions, as well as low accuracy, still exists. The two main reasons for this issue are: (a) artificially generated features are used as the input for the learning classifier model, where a basic knowledge on miRNA target interaction, as well as suitable feature selection algorithm, are lacking, which directly affect the accuracy of the predictor, and (b) the amount of positive data is vastly greater than the amount of negative data because most of the published miRNA target interactions involve positive data, and this imbalanced dataset can lead to underfitting and overfitting problems.

Recently, a new technique called DL as a subgroup of the ML has become a futuristic approach with several applications in bioinformatics and genomics [26, 65, 66], including miRNA target prediction [67]. The current work is utilizing a DL-based state-of-the-art algorithm, which is an optimized DFFN named as AHDR. The AHDR is capable to overcome issues of the imbalanced dataset, high false positive and negative interactions, and the suitable feature selection and representation. DFFN architecture is a powerful classifier, due to its ability to cope with complex data and its potential for modeling data of high non-linearity. Additionally, DFFN is vulnerable to adversarial perturbations in the dataset.

AHDR identifies potential descriptors of each interaction by directly analyzing the whole mature miRNA transcript, rather than focusing only on the seed region, for example, 95 potential descriptors of miRNA-target interactions have been proposed. Next phase, several stages of pre-processing were applied for the generation of precise two-class labels dataset. This dataset was used during the training of AHDR. Performance measures have shown that AHDR consistently outperformed existing methods (Figures 14 and 15).

In modern days, datasets are becoming very enriched in terms of information when dealing with genomic applications like miRNA target interaction. These high dimensional datasets harbor hundreds of features. Hyper-dimensionality of datasets can cause errors in models, due to increased training time with several features. Hence, the model can face issues of overfitting. To avoid this, a careful selection of relevant features is essential for the performance of the classifier algorithm. In this study, the LASSO algorithm was applied to identify relevant descriptors from generated 95 descriptors of the miRNA-target interactions (Figure 10). *DeepMirTar* [68] has used 750 features generated from miRMark [61] which generate a complex dataset for miRNA target interactions, by narrowing down the features to the most relevant ones, as performed in the current work. *DeepMirTar* [68] relied on the random forest (RF) method for selecting features, however, LASSO has many advantages over other methods as (a) it provides a very good prediction accuracy, due to shrinking and removing the coefficients of each feature, (b) it helps to increase the model interpretability by eliminating irrelevant variables (as in Figure 11) that are not associated with the putative miRNA target interactions, thus reducing the bias [49].

An important phase in the application of ML methods is the evaluation of the generalization performance of the trained model. It must be verified that the learning capacity of the learning classifier can perform equally well on the novel data. Therefore, the 10-fold cross-validation is applied, in which all data points of each interaction is tested for at least once. This approach reduces overfitting and underfitting problems. Given that there is a trade-off between ACC, TNR, and MCC after the 10-fold cross-validation, the training approach resulted in a slight loss in the accuracy and MCC values, whereas TNR level slightly increased, which shows that AHDR has a high ability to classify interactions (Figure 13).

According to Saito and Rehmsmeier, the AUC measures the accuracy of the predictor [69]. Therefore, high AUC evaluates the algorithm over all possible interactions (Figure 12). Whereas the maximum F-scores obtained across predictors, served as a measure quantifying the best achievable predictive performance. In terms of this theory, AHDR delivered substantial performance boosts (Figure 14) over other existing predictors based on both *ab initio* and ML approaches. Interestingly, some resources such as *TargetScan* [13] (*ab initio* approach) and *miRMap* [53] (ML-based approach) ranked among the top performing tools, as measured by maximum AUC score and sensitivity, but ranked among the worst according to F-score and ACC. This is because these resources provide accurate predictions, but only for a small subset of genes and miRNAs. *TargetScan* [13] is the user-friendly sequence-based database, but predictions are similar for all members of a miRNA family that can lead to high false positive interactions. Whereas in more detail in the context of performance measures, the Matthews correlation coefficient (MCC) is more informative than other performance matrices measures (such as F1 score and accuracy) in evaluating the performance of algorithm for putative predictions because it takes into account the balance ratios of the four prediction categories such as true positives (the number of validated targets predicted), true negatives (the number of genes that were neither predicted nor validated), false positives (the number of predicted targets that were not validated), and false negatives (the number of validated targets not predicted). Therefore, the value of MCC ranges from -1 to 1, which represents low and high-quality predictions, respectively. The values of MCC were close to zero, which indicated predictions are like random predictions [51, 52]. This study revealed that AHDR is found to achieve much higher MCC values of 0.95 as compared to other predictors (Figure 15). Beside *TargetScan* [13] and *miRMap* [53], the rest of the other target prediction methods obtain the MCC ranges from −0.72 to -0.95. AHDR seems to be the best target prediction algorithm so far compared to the others in terms of statistical performance measurement.

In addition to this work, there are only two studies known so far, which are using the DL approach for miRNA target prediction as *MiRTDL* [70] and *deepTarget* [71]. *MiRTDL* has used CNN architecture to analyze miRNA target features, however, the selected descriptors of each interaction were generated with insufficient information for miRNA target mechanism [70]. *MiRTDL* has used only 20 features for the training purpose, which is inherently subjective and uncertain. Thus, the algorithm faces similar problems as *ab initio* based and ML approaches. In contrast to *MiRTDL* [70], *deepTarget* [71] is based on a unidirectional two-layered RNN architecture [71]. Although this architecture was only little bit effective, however, adopting even more sophisticated approaches may further boost the capability of *deepTarget* [71] for scanning of subtle interactions that often remained undetected. Therefore, these approaches lack putative and functional prediction interactions.

One of the other aims of this thesis was to predict the possible miRNAs targets in an oncogene. Therefore, a list of top 20 oncogenes was deduced in which ADHR performed better than other known miRNA-target predictors (Figure 17) as exemplified using top oncogenes extracted from a recent study [57]. Several known tools have failed to pick even a single interaction (red shade in Figure 17), which hints their limitations. During this analysis, some miRNAs are interacting with more than one oncogene (marked by color fonts in Figure 17) generated by AHDR.

The extensive presence of *let-7b-5p* suggests that it is a well-known tumor-suppressor belongs to *let* family in breast cancer [72], and also it is known that *let-7b-5p* is a negative regulator of insulin-like growth factor receptor 1 (IGF1R) in multiple myeloma [73]. Mutation in DICER1 leads to the loss of tumor-suppressor miRNAs, including *let-7b-5p* [74] and additionally, *let-7b-5p* is a negative regulator of other oncogenes such as HRAS and KRAS [75]. The *miR-146a-5p* inhibits cell proliferation and metastasis and it induces apoptosis through the EGFR generated signaling pathway in the lung cancer [76]. Similarly, *miR-146a-5p* is also a negative regulator of the SMAD4 gene by reducing fibrosis in the skeletal muscle after injury, hence it might act as a potential therapeutic [77]. *miR-603* downregulates the expression of HRAS protein in prostate tumor cells [78]. Overexpression of *miR-30a-5p* inhibits cell proliferation in lung carcinogenesis via EGFR signaling pathways [79].

The miRNA, *miR-30a-5p* is a negative regulator of the TP53 gene which causes resistance to anti-cancer drugs [80]. *miR-375* is a positive regulator of the PIK3CA gene during anti-proliferative and apoptosis effects [81]. Similarly, this *miR-375* is regulator for other oncogenes like RHOA [82] and TP53 [83]. *miR-489-3p* inhibits cell proliferation in hypopharyngeal squamous cell carcinoma (HSCC) by targeting PTPN11 gene [84] and also circulating *miR-489-3p* have shown significant expression variations in breast cancer by targeting TP53 [72, 85]. The miRNA *miR-483-3p* is the regulator of oxaliplatin resistance in human colorectal cancer cells [86]. *miR-483-3p* serves a crucial suppressor of TP53 signaling in liver cancer [87]. Additionally, *miR-4282* targets Myc gene and it serves as inhibitor of the proliferation in breast cancer [88]. Taken together, in this case study the findings reflect that the AHDR approach can identify the oncogenic interactions of miRNAs.

In this study, AHDR algorithm was developed, which is based on the advanced deep learning technology for human miRNA target prediction. AHDR outperformed other predictors in the test dataset. Furthermore, the AHDR showed top interactions on the experimentally verified dataset. In addition, the AHDR also detected many putative miRNAs and their interactions with oncogenes. Interestingly, the AHDR predicts potential interactions instead of a high number of false positive interactions. This detailed explanation contributes to the importance of deep learning technology in genomics.

5. SUMMARY

miRNAs are small, non-coding RNA evolutionary conserved molecules, which are largely known to regulate gene expression by annealing to target genes. Predicting possible binding sites associated with it allows us to understand their regulations. Over the past decades, significant efforts have been made to improve the knowledge of miRNAs. One primary concern was miRNA-target interaction, despite more than a dozen bioinformatic approaches that have been developed using *ab initio* methods as well as ML methods. However, these approaches face a notorious challenge in terms of tiny alignment regions (7-8 nucleotides in the seed region) between miRNAs and targets. This often leads to higher numbers of false positive interactions. In order to resolve this issue, a DFFN-based DL algorithm was developed which is called as advanced hierarchical deep-rooted learning (AHDR). In the data mining step, a balanced training dataset was constructed using high-quality positive datasets, which were derived from the best-curated mRNA target databases such as *TarBase*. Whereas, the negative dataset was generated using random mutations of 95% across human genome-wide transcripts. Several parameters were applied to both datasets to overcome the issue of overfitting and underfitting. In the training step, different DL algorithms were applied to the generated dataset to identify potential models for the target prediction. The results revealed the performance of each prediction algorithm. Furthermore, the performance prediction of AHDR was analyzed and compared with other high impact published prediction algorithms on a strong experimentally verified dataset.

The purpose of this study was to demonstrate the flexibility and impact of the DL framework to miRNA-target site prediction. However, it also facilitates holistic physical and biological models integrating heterogeneous data from different sources that foster the understanding of disease development, progression, and treatment possibilities.

6. REFERENCES

1. Bartel, DP: MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell,* 116**:** 281-297, 2004.
2. Trionfini, P & Benigni, A: MicroRNAs as master regulators of glomerular function in health and disease. *J Am Soc Nephrol,* 28**:** 1686-1696, 2017.
3. Paul, P, Chakraborty, A, Sarkar, D, Langthasa, M, Rahman, M, Bari, M, Singha, RS, Malakar, AK & Chakraborty, S: Interplay between miRNAs and human diseases. *J Cell Physiol,* 233**:** 2007-2018, 2018.
4. Chen, C, Ponnusamy, M, Liu, C, Gao, J, Wang, K & Li, P: MicroRNA as a therapeutic target in cardiac remodeling. *Biomed Res Int,* 2017**:** 1-25, 2017.
5. Putteeraj, M, Fairuz, YM & Teoh, SL: MicroRNA dysregulation in Alzheimer's disease. *CNS Neurol Disord Drug Targets,* 16**:** 1000-1009, 2017.
6. Karaca, E, Aykut, A, Erturk, B, Durmaz, B, Guler, A, Buke, B, Yeniel, AO, Ergenoglu, AM, Ozkinay, F, Ozeren, M, Kazandi, M, Akercan, F, Sagol, S, Gunduz, C & Cogulu, O: MicroRNA expression profile in the prenatal amniotic fluid samples of pregnant women with Down syndrome. *Balkan Med J,* 35**:** 163-166, 2018.
7. Lai, NS, Koo, M, Yu, CL & Lu, MC: Immunopathogenesis of systemic lupus erythematosus and rheumatoid arthritis: the role of aberrant expression of non-coding RNAs in T cells. *Clin Exp Immunol,* 187**:** 327-336, 2017.
8. Liu, Q, Wu, DH, Han, L, Deng, JW, Zhou, L, He, R, Lu, CJ & Mi, QS: Roles of microRNAs in psoriasis: immunological functions and potential biomarkers. *Exp Dermatol,* 26**:** 359-367, 2017.
9. Lu, Q, Sun, Y, Duan, Y, Li, B, Xia, J, Yu, S & Zhang, G: Comprehensive microRNA profiling reveals potential augmentation of the IL1 pathway in rheumatic heart valve disease. *BMC Cardiovasc Disord,* 18**:** 53, 2018.
10. Dweep, H, Sticht, C & Gretz, N: In-silico algorithms for the screening of possible microRNA binding sites and their interactions. *Curr Genomics,* 14**:** 127-136, 2013.
11. Peterson, SM, Thompson, JA, Ufkin, ML, Sathyanarayana, P, Liaw, L & Congdon, CB: Common features of microRNA target prediction tools. *Front Genet,* 5**:** 23, 2014.
12. Betel, D, Wilson, M, Gabow, A, Marks, DS & Sander, C: The microRNA.org resource: targets and expression. *Nucleic Acids Res,* 36**:** D149-D153, 2008.
13. Agarwal, V, Bell, GW, Nam, JW & Bartel, DP: Predicting effective microRNA target sites in mammalian mRNAs. *Elife,* 4: e05005: 1-38, 2015.
14. Liu, W & Wang, X: Prediction of functional microRNA targets by integrative modeling of microRNA binding and target expression data. *Genome Biol,* 20**:** 18, 2019.
15. Miranda, KC, Huynh, T, Tay, Y, Ang, YS, Tam, WL, Thomson, AM, Lim, B & Rigoutsos, I: A pattern-based method for the identification of microRNA binding sites and their corresponding heteroduplexes. *Cell,* 126**:** 1203-1217, 2006.
16. Bandyopadhyay, S & Mitra, R: TargetMiner: microRNA target prediction with systematic identification of tissue-specific negative examples. *Bioinformatics,* 25**:** 2625-2631, 2009.
17. Liu, H, Yue, D, Chen, Y, Gao, SJ & Huang, Y: Improving performance of mammalian microRNA target prediction. *BMC Bioinformatics,* 11**:** 476, 2010.
18. Kertesz, M, Iovino, N, Unnerstall, U, Gaul, U & Segal, E: The role of site accessibility in microRNA target recognition. *Nat Genet,* 39**:** 1278-1284, 2007.
19. Kruger, J & Rehmsmeier, M: RNAhybrid: microRNA target prediction easy, fast and flexible. *Nucleic Acids Res,* 34**:** W451-W454, 2006.
20. Hamet, P & Tremblay, J: Artificial intelligence in medicine. *Metabolism,* 69**:** S36-S40, 2017.

21. Libbrecht, MW & Noble, WS: Machine learning applications in genetics and genomics. *Nat Rev Genet,* 16**:** 321-332, 2015.
22. Xie, P, Gao, M, Wang, C, Zhang, J, Noel, P, Yang, C, Von Hoff, D, Han, H, Zhang, MQ, & Lin, W: SuperCT: a supervised-learning framework for enhanced characterization of single-cell transcriptomic profiles. *Nucleic Acids Res*, 47: e48, 2019.
23. Chang, H, Han, J, Zhong, C, Snijders, AM & Mao, JH: Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications. *IEEE Trans Pattern Anal Mach Intell,* 40**:** 1182-1194, 2018.
24. Krittanawong, C, Zhang, H, Wang, Z, Aydar, M & Kitai, T: Artificial intelligence in precision cardiovascular medicine. *J Am Coll Cardiol,* 69**:** 2657-2664, 2017.
25. Demirci, F, Akan, P, Kume, T, Sisman, AR, Erbayraktar, Z & Sevinc, S: Artificial neural network approach in laboratory test reporting: learning algorithms. *Am J Clin Pathol,* 146**:** 227-237, 2016.
26. Eraslan, G, Avsec, Z, Gagneur, J & Theis, FJ: Deep learning: new computational modelling techniques for genomics. *Nat Rev Genet*, 20(7): 389-403, 2019.
27. Cao, C, Liu, F, Tan, H, Song, D, Shu, W, Li, W, Zhou, Y, Bo, X & Xie, Z: Deep learning and its applications in biomedicine. *Genomics Proteomics Bioinformatics,* 16**:** 17-32, 2018.
28. Zeng, M, Li, M, Fei, Z, Wu, F, Li, Y, Pan, Y & Wang, J: A deep learning framework for identifying essential proteins by integrating multiple types of biological information. *IEEE/ACM Trans Comput Biol Bioinform*, 1:1, 2019.
29. Zeng, H, Edwards, MD, Liu, G & Gifford, DK: Convolutional neural network architectures for predicting DNA-protein binding. *Bioinformatics,* 32**:** i121-i127, 2016.
30. Pan, X, Rijnbeek, P, Yan, J & Shen, HB: Prediction of RNA-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks. *BMC Genomics,* 19**:** 511:1-11, 2018.
31. Chen, HH, Chiu, YC, Zhang, T, Zhang, S, Huang, Y & Chen, Y: GSAE: an autoencoder with embedded gene-set nodes for genomics functional characterization. *BMC Syst Biol,* 12(S8):1-13, 2018.
32. Wolterink, JM, Leiner, T, Viergever, MA & Isgum, I: Generative adversarial networks for noise reduction in low-dose CT. *IEEE Trans Med Imaging,* 36**:** 2536-2545, 2017.
33. Nibali, A, He, Z & Wollersheim, D: Pulmonary nodule classification with deep residual networks. *Int J Comput Assist Radiol Surg,* 12**:** 1799-1808, 2017.
34. Li, Y, Huang, C, Ding, L, Li, Z, Pan, Y & Gao, X: Deep learning in bioinformatics: introduction, application, and perspective in the big data era. *Methods*, pii: S1046-S2023, 2019.
35. Rampasek, L & Goldenberg, A: TensorFlow: biology's gateway to deep learning? *Cell Syst,* 2**:** 12-14, 2016.
36. Kozomara, A & Griffiths-Jones, S: miRBase: annotating high confidence microRNAs using deep sequencing data. *Nucleic Acids Res,* 42**:** D68-D73, 2014.
37. Aken, BL, Achuthan, P, Akanni, W, Amode, MR, Bernsdorff, F, Bhai, J, Billis, K, Carvalho-Silva, D, Cummins, C, Clapham, P, Gil, L, Giron, CG, Gordon, L, Hourlier, T, Hunt, SE, Janacek, SH, Juettemann, T, Keenan, S, Laird, MR, Lavidas, I, Maurel, T, McLaren, W, Moore, B, Murphy, DN, Nag, R, Newman, V, Nuhn, M, Ong, CK, Parker, A, Patricio, M, Riat, HS, Sheppard, D, Sparrow, H, Taylor, K, Thormann, A, Vullo, A, Walts, B, Wilder, SP, Zadissa, A, Kostadima, M, Martin, FJ, Muffato, M, Perry, E, Ruffier, M, Staines, DM, Trevanion, SJ, Cunningham, F, Yates, A, Zerbino, DR & Flicek, P: Ensembl 2017. *Nucleic Acids Res,* 45**:** D635-D642, 2017.
38. Chou, CH, Chang, NW, Shrestha, S, Hsu, SD, Lin, YL, Lee, WH, Yang, CD, Hong, HC, Wei, TY, Tu, SJ, Tsai, TR, Ho, SY, Jian, TY, Wu, HY, Chen, PR, Lin, NC, Huang, HT, Yang, TL, Pai, CY, Tai, CS, Chen, WL, Huang, CY, Liu, CC, Weng, SL, Liao, KW, Hsu, WL &

Huang, HD: miRTarBase 2016: updates to the experimentally validated miRNA-target interactions database. *Nucleic Acids Res,* 44**:** D239-D247, 2016.
39. Ding, J, Li, X & Hu, H: TarPmiR: a new approach for microRNA target site prediction. *Bioinformatics,* 32**:** 2768-2775, 2016.
40. Karagkouni, D, Paraskevopoulou, MD, Chatzopoulos, S, Vlachos, IS, Tastsoglou, S, Kanellos, I, Papadimitriou, D, Kavakiotis, I, Maniou, S, Skoufos, G, Vergoulis, T, Dalamagas, T & Hatzigeorgiou, AG: DIANA-TarBase v8: a decade-long collection of experimentally supported miRNA-gene interactions. *Nucleic Acids Res,* 46(D1): D239-D245, 2018.
41. Tabas-Madrid, D, Muniategui, A, Sanchez-Caballero, I, Martinez-Herrera, DJ, Sorzano, CO, Rubio, A & Pascual-Montano, A: Improving miRNA-mRNA interaction predictions. *BMC Genomics,* 15(S10):1-12, 2014.
42. Gruber, AR, Lorenz, R, Bernhart, SH, Neubock, R & Hofacker, IL: The Vienna RNA websuite. *Nucleic Acids Res,* 36**:** W70-W74, 2008.
43. Sturm, M, Hackenberg, M, Langenberger, D & Frishman, D: TargetSpy: a supervised machine learning approach for microRNA target prediction. *BMC Bioinformatics,* 11**:** 292,1-17, 2010.
44. Cahsai, A, Anagnostopoulos, C & Triantafillou, P: Scalable data quality for big data: The Pythia framework for handling missing values. *Big Data,* 3**:** 159-172, 2015.
45. Ghoshal, A, Shankar, R, Bagchi, S, Grama, A & Chaterji, S: MicroRNA target prediction using thermodynamic and sequence curves. *BMC Genomics,* 16**:** 999: 1-22, 2015.
46. Artzi, S, Kiezun, A & Shomron, N: miRNAminer: a tool for homologous microRNA gene search. *BMC Bioinformatics,* 9**:** 39: 1-7, 2008.
47. Kamkar, I, Gupta, SK, Phung, D & Venkatesh, S: Stable feature selection for clinical prediction: exploiting ICD tree structure using Tree-Lasso. *J Biomed Inform,* 53**:** 277-290, 2015.
48. Shortreed, SM & Ertefaie, A: Outcome-adaptive lasso: variable selection for causal inference. *Biometrics,* 73**:** 1111-1122, 2017.
49. Ruuska, S, Hamalainen, W, Kajava, S, Mughal, M, Matilainen, P & Mononen, J: Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behav Processes,* 148**:** 56-62, 2018.
50. Bastanlar, Y & Ozuysal, M: Introduction to machine learning. *Methods Mol Biol,* 1107**:** 105-128, 2014.
51. Vejnar, CE, Blum, M & Zdobnov, EM: miRmap web: comprehensive microRNA target prediction online. *Nucleic Acids Res,* 41**:** W165-W168, 2013.
52. Paraskevopoulou, MD, Georgakilas, G, Kostoulas, N, Vlachos, IS, Vergoulis, T, Reczko, M, Filippidis, C, Dalamagas, T & Hatzigeorgiou, AG: DIANA-microT web server v5.0: service integration into miRNA functional analysis workflows. *Nucleic Acids Res,* 41**:** W169-W173, 2013.
53. Coronnello, C & Benos, PV: ComiR: combinatorial microRNA target prediction tool. *Nucleic Acids Res,* 41**:** W159-W164, 2013.
54. Lewis, BP, Burge, CB & Bartel, DP: Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. *Cell,* 120**:** 15-20, 2005.
55. Lu, TP, Lee, CY, Tsai, MH, Chiu, YC, Hsiao, CK, Lai, LC & Chuang, EY: miRSystem: an integrated system for characterizing enriched functions and pathways of microRNA targets. *PLoS One,* 7**:** e42390, 2012.
56. Blin, K, Dieterich, C, Wurmus, R, Rajewsky, N, Landthaler, M & Akalin, A: DoRiNA 2.0--upgrading the doRiNA database of RNA interactions in post-transcriptional regulation. *Nucleic Acids Res,* 43**:** D160-D167, 2015.
57. Bailey, MH, Tokheim, C, Porta-Pardo, E, Sengupta, S, Bertrand, D, Weerasinghe, A, Colaprico, A, Wendl, MC, Kim, J, Reardon, B, Kwok-Shing Ng, P, Jeong, KJ, Cao, S, Wang,

Z, Gao, J, Gao, Q, Wang, F, Liu, EM, Mularoni, L, Rubio-Perez, C, Nagarajan, N, Cortes-Ciriano, I, Zhou, DC, Liang, WW, Hess, JM, Yellapantula, VD, Tamborero, D, Gonzalez-Perez, A, Suphavilai, C, Ko, JY, Khurana, E, Park, PJ, Van Allen, EM, Liang, H, Group, MCW, Cancer Genome Atlas Research, N, Lawrence, MS, Godzik, A, Lopez-Bigas, N, Stuart, J, Wheeler, D, Getz, G, Chen, K, Lazar, AJ, Mills, GB, Karchin, R & Ding, L: Comprehensive characterization of cancer driver genes and mutations. *Cell,* 174**:** 1034-1035, 2018.
58. Reczko, M, Maragkakis, M, Alexiou, P, Papadopoulos, GL & Hatzigeorgiou, AG: Accurate microRNA target prediction using detailed binding site accessibility and machine learning on proteomics data. *Front Genet,* 2**:** 103: 1-13, 2011.
59. Menor, M, Ching, T, Zhu, X, Garmire, D & Garmire, LX: mirMark: a site-level and UTR-level classifier for miRNA target prediction. *Genome Biol,* 15**:** 500: 1-15, 2014.
60. Bandyopadhyay, S, Ghosh, D, Mitra, R & Zhao, Z: MBSTAR: multiple instance learning for predicting specific functional binding sites in microRNA targets. *Sci Rep,* 5**:** 8004: 1-12, 2015.
61. Yousef, M, Jung, S, Kossenkov, AV, Showe, LC & Showe, MK: Naïve bayes for microRNA target predictions--machine learning for microRNA targets. *Bioinformatics,* 23**:** 2987-2992, 2007.
62. Kim, SK, Nam, JW, Rhee, JK, Lee, WJ & Zhang, BT: miTarget: microRNA target gene prediction using a support vector machine. *BMC Bioinformatics,* 7**:** 411: 1-12, 2006.
63. Huang, JC, Morris, QD & Frey, BJ: Bayesian inference of MicroRNA targets from sequence and expression data. *J Comput Biol,* 14**:** 550-563, 2007.
64. Ruan, J, Chen, H, Kurgan, L, Chen, K, Kang, C & Pu, P: HuMiTar: a sequence-based method for prediction of human microRNA targets. *Algorithms Mol Biol,* 3**:** 16: 1-12, 2008.
65. Liu, Q & Hu, P: Association analysis of deep genomic features extracted by denoising autoencoders in breast cancer. *Cancers,* 11: 494: 1-13, 2019.
66. Chen, KM, Cofer, EM, Zhou, J & Troyanskaya, OG: Selene: a PyTorch-based deep learning library for sequence data. *Nat Methods,* 16**:** 315-318, 2019.
67. Pla, A, Zhong, X & Rayner, S: miRAW: A deep learning-based approach to predict microRNA targets by analyzing whole microRNA transcripts. *PLoS Comput Biol,* 14**:** e1006185, 2018.
68. Wen, M, Cong, P, Zhang, Z, Lu, H & Li, T: DeepMirTar: a deep-learning approach for predicting human miRNA targets. *Bioinformatics,* 34**:** 3781-3787, 2018.
69. Saito, T & Rehmsmeier, M: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One,* 10**:** e0118432, 2015.
70. Shuang, C, Maozu, G, Chunyu, W, Xiaoyan, L, Yang, L & Xuejian, W: MiRTDL: A deep learning approach for miRNA target prediction. *IEEE/ACM Trans Comput Biol Bioinform,* 13**:** 1161-1169, 2016.
71. Paraskevopoulou, MD, Karagkouni, D, Vlachos, IS, Tastsoglou, S & Hatzigeorgiou, AG: microCLIP super learning framework uncovers functional transcriptome-wide miRNA interactions. *Nat Commun,* 9**:** 3601: 1-16, 2018.
72. Qattan, A, Intabli, H, Alkhayal, W, Eltabache, C, Tweigieri, T & Amer, SB: Robust expression of tumor suppressor miRNA's let-7 and miR-195 detected in plasma of Saudi female breast cancer patients. *BMC Cancer,* 17**:** 799: 1-10, 2017.
73. Xu, H, Liu, C, Zhang, Y, Guo, X, Liu, Z, Luo, Z, Chang, Y, Liu, S, Sun, Z & Wang, X: Let-7b-5p regulates proliferation and apoptosis in multiple myeloma by targeting IGF1R. *Acta Biochim Biophys Sin (Shanghai),* 46**:** 965-972, 2014.
74. Rakheja, D, Chen, KS, Liu, Y, Shukla, AA, Schmid, V, Chang, TC, Khokhar, S, Wickiser, JE, Karandikar, NJ, Malter, JS, Mendell, JT & Amatruda, JF: Somatic mutations in DROSHA and DICER1 impair microRNA biogenesis through distinct mechanisms in Wilms tumours. *Nat Commun,* 2**:** 4802: 1-10, 2014.

75. Wagner, S, Ngezahayo, A, Murua Escobar, H & Nolte, I: Role of miRNA let-7 and its major targets in prostate cancer. *Biomed Res Int,* 2014**:** 1-14, 2014.
76. Huang, WT, Cen, WL, He, RQ, Xie, Y, Zhang, Y, Li, P, Gan, TQ, Chen, G & Hu, XH: Effect of miR146a5p on tumor growth in NSCLC using chick chorioallantoic membrane assay and bioinformatics investigation. *Mol Med Rep,* 16**:** 8781-8792, 2017.
77. Sun, Y, Li, Y, Wang, H, Li, H, Liu, S, Chen, J & Ying, H: miR-146a-5p acts as a negative regulator of TGF-beta signaling in skeletal muscle after acute contusion *Acta Biochimica et Biophysica Sinica,* 49**:** 628-634, 2017.
78. Moustafa, AA, Ziada, M, Elshaikh, A, Datta, A, Kim, H, Moroz, K, Srivastav, S, Thomas, R, Silberstein, JL, Moparty, K, Salem, FE, El-Habit, OH & Abdel-Mageed, AB: Identification of microRNA signature and potential pathway targets in prostate cancer. *Exp Biol Med (Maywood),* 242**:** 536-546, 2017.
79. Zhu, J, Zeng, Y, Li, W, Qin, H, Lei, Z, Shen, D, Gu, D, Huang, JA & Liu, Z: CD73/NT5E is a target of miR-30a-5p and plays an important role in the pathogenesis of non-small cell lung cancer. *Mol Cancer,* 16**:** 34: 1-15, 2017.
80. Park, D, Kim, H, Kim, Y & Jeoung, D: miR-30a regulates the expression of CAGE and p53 and regulates the response to anti-cancer drugs. *Mol Cells,* 39**:** 299-309, 2016.
81. Zhou, N, Qu, Y, Xu, C & Tang, Y: Upregulation of microRNA-375 increases the cisplatin-sensitivity of human gastric cancer cells by regulating ERBB2. *Exp Ther Med,* 11**:** 625-630, 2016.
82. Abdelmohsen, K, Hutchison, ER, Lee, EK, Kuwano, Y, Kim, MM, Masuda, K, Srikantan, S, Subaran, SS, Marasa, BS, Mattson, MP & Gorospe, M: miR-375 inhibits differentiation of neurites by lowering HuD levels. *Mol Cell Biol,* 30**:** 4197-4210, 2010.
83. Liu, Y, Xing, R, Zhang, X, Dong, W, Zhang, J, Yan, Z, Li, W, Cui, J & Lu, Y: miR-375 targets the p53 gene to regulate cellular response to ionizing radiation and etoposide in gastric cancer cells. *DNA Repair (Amst),* 12**:** 741-750, 2013.
84. Kikkawa, N, Hanazawa, T, Fujimura, L, Nohata, N, Suzuki, H, Chazono, H, Sakurai, D, Horiguchi, S, Okamoto, Y & Seki, N: miR-489 is a tumour-suppressive miRNA target PTPN11 in hypopharyngeal squamous cell carcinoma (HSCC). *Br J Cancer,* 103**:** 877-884, 2010.
85. Kuppa, SS, Jia, W, Liu, S, Nguyen, H, Smyth, SS, Mills, GB, Dobbin, KK, Hardman, WJ & Murph, MM: Autotaxin exacerbates tumor progression by enhancing MEK1 and overriding the function of miR-489-3p. *Cancer Lett,* 432**:** 84-92, 2018.
86. Liang, H, Xu, Y, Zhang, Q, Yang, Y, Mou, Y, Gao, Y, Chen, R, Chen, C & Dai, P: MiR-483-3p regulates oxaliplatin resistance by targeting FAM171B in human colorectal cancer cells. *Artif Cells Nanomed Biotechnol,* 47**:** 725-736, 2019.
87. Pepe, F, Pagotto, S, Soliman, S, Rossi, C, Lanuti, P, Braconi, C, Mariani-Costantini, R, Visone, R & Veronese, A: Regulation of miR-483-3p by the O-linked N-acetylglucosamine transferase links chemosensitivity to glucose metabolism in liver cancer cells. *Oncogenesis,* 6**:** e328, 2017.
88. Zhao, J & Jiang, GQ: MiR-4282 inhibits proliferation, invasion and metastasis of human breast cancer by targeting Myc. *Eur Rev Med Pharmacol Sci,* 22**:** 8763-8771, 2018.

7. CURRICULUM VITAE AND PUBLICATIONS

PERSONAL INFORMATION

Name:	Alisha Parveen
Nationality:	Indian
Date of Birth:	24.11.1989
Place of Birth:	Allahabad, India
Civil Status:	Unmarried

UNIVERSITIES

1st September 2015 - 31st August 2018	Ph.D. student (Bioinformatics), Medical Research Center, University of Heidelberg, Germany. **Title:** Advanced hierarchical learning approach for microRNA and target prediction
1st May 2015 – 30th September 2015	Research assistant (Bioinformatics), Technical University of Dresden. **Title:** Computational analysis using RNAither of siRNA.
1st July 2011 – 30th July 2013	Master of Science (Bioinformatics), Jamia Millia Islamia, New Delhi, India. **Title:** Meta-analysis of genome wide associated genes (GWAS) related to cardiovascular diseases.
31st August 2008 – 29th April 2011	Bachelor of science (biological science), University of Delhi, New Delhi, India.

SCHOOLING

2008	10+2 (12th), Central Board of Secondary Education, New Delhi, India.
2006	High school (10th), Central Board of Secondary Education, New Delhi, India.

RESEARCH EXPERIENCE

1ST November 2011 – 30th August 2012	Research trainee (bioinformatics), Defence research and development organization (DRDO), New Delhi, India. **Title:** Computational network model predicts the drug effects on SHP-1 mediated intracellular signaling through c-kit molecule

PUBLICATIONS

1: **Parveen A**, Gretz N, and Dweep H: Obtaining miRNA-target interaction information from miRWalk2.0. *Curr. Protoc. Bioinform., 55:12.15.1-12.15.27, 2016.*

2: Sticht C, Torre C, **Parveen A**, and Gretz N: miRWalk: An online resource for prediction of microRNA binding sites. *PLoS ONE., 13 (10): e0206239, 2018.*

PRESENTATIONS AND TALKS

1: **Parveen A** and Gangenahalli G U: Computational network model predicts the drug effects on SHP-1 mediated intracellular signaling through c-Kit, International interdisciplinary science conference (I-ISC) on Protein Folding and Diseases, Vol 3, Issue 2, pp. 26, 2012.

2: **Parveen A** and Hassan I: Meta-analysis of genome-wide association studies on gene related to the cardiovascular disorders, National Conference on Recent Trends in Protein Structural Biology (NCRTPSB) 2013 on Structural Biology, Vol. 4, No. 2, pp. 27, 2013. ISSN- 0975-8151.

8. APPENDIX

Appendix 8.1. Negative dataset generation. This python code randomly mutates the binding region of the human mRNA, which hinders the binding of miRNA. This generates negative human mRNA dataset. The input file is in the fasta format.

```
   from random import random, choice
   import sys
   from itertools import groupby
      def fasta_iter(fasta_name):
"""given a fasta file. yield tuples of header, sequence""
       fh = open(fasta_name)

       # ditch the boolean (x[0]) and just keep the header or sequence since
       faiter = (x[1] for x in groupby(fh, lambda line: line[0] == ">"))
       for header in faiter:
              header = header.next()[1:].strip()

# join all sequence lines to one.
           seq = "".join(s.strip() for s in faiter.next())
           yield header, seq
   def main(fasta_name, mutation_freq):
       for header, seq in fasta_iter(fasta_name):
          seq = list(seq)
           for i, s in enumerate(seq):
               val = random()
               if val < mutation_freq:
    # choose a random nucleotide that's different.
   seq[i] = choice([x for x in "ACTG" if x != s.upper()])
           print ">%s\n%s" % (header, "".join(seq))
   if __name__ == "__main__":
       main(sys.argv[1], float(sys.argv[2]))
```

Appendix 8.2. Perl script for the generation of structural features: In this Perl code, structural features generated include symmetric and asymmetric loops, symmetric and asymmetric bulges in both in seed and outseed region in the miRNA. The input file is the duplex structure of miRNA target interaction [61].

```
#!/usr/bin/perl
use Term::ANSIColor;

if (@ARGV==0) {
  print STDERR "usage: features_mirna_positive.pl  <output of miranada>\n";
  exit;
}
open (ST,">$ARGV[0].statistics");
open (STB,">$ARGV[0].stb");
open (INFO, $ARGV[0]);   #The output of find_transcripts
#open (MATURE, $ARGV[1]); #The mature microRNA file
#open (FOUT ,$ARGV[2]  );
print ST "FHR: d5  bpmirna seed-nblgus seed-nloops seed-nasymloops","\n" ;
print ST "FHR: bp5' 5'-nblgus 5'-nloops 5'-nasymloops","\n";
print ST "FHR:seed part(8 features): #bulges with length 1,2,..,7 and >7\n"
;
print ST "FHR:seed part:(8 features) #symtric loops with length 1,2,..,7
and >7\n" ;
print ST "FHR:seed part: (8 features)#asymetric loops with length 1,2,..,7
and >7\n" ;
print ST "FHR:out-seed part: (8 features)#bulgesc loops with length
1,2,..,7 and >7\n" ;
print ST "FHR:out-seed part: (8 features)#symetric loops with length
1,2,..,7 and >7\n" ;
print ST "FHR:out-seed part: (8 features)#asymetric loops with length
1,2,..,7 and >7" ;


#@nblgsmirna      =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;
#@nblgsoutmirna   =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

#@nloopsmirna     =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
#@nsymloopsmirna  =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
#@nasymloopsmirna =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

#@nloopsoutmirna     =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
#@nsymloopsoutmirna  =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
#@nasymloopsoutmirna =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

$flagformirna = 0 ;
#Read The mature microRNA to an array
#@mature = <MATURE>;
@location=();
$counter = 0 ;
$sum5d=0;
$sum3d=0;
$summirna =0 ;
$summirnab=0;
$line ="blablabla";
$indexcounter=0;
#while ( $line !~/HDR/ )
  #   {$line=<INFO>;} #Reach the first HDR

while ( !eof(INFO)  )
{
  while ( $line !~/>/ & !eof(INFO)){
      $line=<INFO>;} #Reach the first ">"
      print $line;
       if ($line =~/>/){
            if ( $line =~ />(.*)/ ) { $hdrname =$1;}
       }
      $title = $line;
      $gene_info = <INFO>;
      @ref=split(/ /,$gene_info);
      #$hdrname = join "",$hdrname,"\t",$ref[10].$ref[11].$ref[12];
      $hdrname = join "",$hdrname,"\t",$ref[10].$ref[11].$ref[12];
      $empty=<INFO>;
      $stem1 = <INFO>;
      $dashes= <INFO>;
      $stem2 = <INFO>;
      if ( $stem1     =~/(Ref:)\s+(5'\s*[NATCGUatcgu-]*[NATCGUatcgu-
]\s*3')/ )
                  {   $tmp_stm1 = $2;
                      if ($tmp_stm1 =~/5'\s*([NATCGUatcgu-]*[NATCGUatcgu-
])\s*3'/ )
                        {$stm1    = $1; $stem_l=length($stm1) }
                   }
      if ( $stem2     =~/(Query:)\s+(3'\s*[NATCGUatcgu-]*[NATCGUatcgu-
]\s*5')/ )
                  {    $tmp_stm2 = $2;
                      if ($tmp_stm2 =~/3'\s*([NATCGUatcgu-]*[NATCGUatcgu-
])\s*5'/ )
                                {$stm2    = $1;}
                    }
       if ($stem1 =~m/(Ref:\s+)/g)
           { $first_t=pos($stem1);}
       $new_dashes = substr($dashes,$first_t);
      if ( $new_dashes    =~/\s+(\|+[\|\s]*\|+)\s+/              )
                  { $dash = $1;}
     $new_dashes=~s/\n//g;
    print "\n$tmp_stm1\n$new_dashes\n$tmp_stm2\n";
    my $count = 0 ;
    my $i = 1 ;
    my $loc=$stem_l;
    while ($count < 8)
       { $s = substr($stm1,$stem_l-$i,1);
        if ($s =~/[ATCGUatcgu]/ )
          { ++$count;}
        ++$i;
        --$loc;
     }
 my $seed = substr($stm1,$loc) ;
    $seed=~s/-//g;
    $seed=~tr/[Tt]/[Uu]/;
    $seed=~tr/[ATCGUN]/[atcgun]/;
      ++$indexcounter;
     #$hdrname =~tr/\n//;
      print ST "\n>HDR:$hdrname";
      print ST "\n## $seed";
      print ST "\n## Index:$indexcounter\n";
      $new_dashes=~s/\./ /g;
      $new_dashes=~s/\:/ /g;
      #extractFeatures($stem1,$stem2,$dash);
      extractFeatures($tmp_stm1,$tmp_stm2,$new_dashes);
     $hdrname = "blabla";
     $line = <INFO>;
  }#End of while ( !eof(INFO) );

close INFO;
#     print ST "\n>HDR:";

print "Total sum of stem-loop microRNA is ",$summirna,"\n";
print "Total sum of microRNA's is ",$summirnab,"\n";

print "Total sum of stem-loop 5' is ",$sum5d,"\n";
print "Total sum of stem-loop 3' is ",$sum3d,"\n";

print STB "nt \n nblgs-mirna, nblgs-outmirna, \n nloops-mirna, nloops-
outmirna,\n nsym-loops-mirna, nsym-loops-outmirna, \n nasym-loops-mirna,
nasym-loops-outmirna\n\n";

print STB "Total sum of microRNA's is ",$summirnab,"\n";
print STB  "Total sum of stem-loop microRNA is ",$summirna,"\n";
print STB "Total sum of stem-loops 5' is ",$sum5d,"\n";
print STB "Total sum of stem-loops 3' is ",$sum3d,"\n";
print STB "Total sum of stem-loops out-mirna(5'+3') is ",
$sum5d+$sum3d,"\n\n";

for ($k=1; $k<=15; $k++)
  {    $result = sprintf("%4d %4d %4d %4d %4d %4d %4d %4d %4d
\n",$k,$nblgsmirna[$k],$nblgsoutmirna[$k],$nloopsmirna[$k],$nloopsoutmirna[
$k],$nsymloopsmirna[$k],$nsymloopsoutmirna[$k],$nasymloopsmirna[$k],$nasyml
oopsoutmirna[$k] );
       print STB $result;
  }
close STB;
print STDOUT "finishd.";

############################################################################
################################################
sub extractFeatures{
                  my ($stem1)= shift (@_);
                  my ($stem2)= shift (@_);
                  my ($dashes) = shift (@_); #We refer to dashes directly
my $stem_l = 0;
      if ( $stem1     =~/(5'\s*)([NATCGUatcgu-]*[NATCGUatcgu-])(\s*3')/ )
                 { $stm1     = $2; $stem_l = length($stm1); }
      if ( $stem2     =~/(3'\s*)([NATCGUatcgu-]*[NATCGUatcgu-])(\s*5')/ )
                 { $stm2     = $2; $stem_l = $1;$prime5=$3;}
      #if ( $dashes     =~/\s+(\|+[\|\s]*\|+)\s+/              )
                 #{ #$dash = $1;
                  my $t = $dashes;
                  $dash = substr($t,3,$stem_l);
                  #} #
my $count = 0 ;
my $i = 1 ;
my $loc=$stem_l;
while ($count < 8)
 { $s = substr($stm1,$stem_l-$i,1);
   if ($s =~/[ATCGUatcgu]/ )
     { ++$count;}
   ++$i;
   --$loc;
 }

 $location[0] = $loc ;
 $location[1] = $stem_l-1;
```

```
    #Initioliza the blugs,loops array (changed from 15 to 100)
     for ($inb=1;$inb<=100;++$inb)
        { $nblgsmirna[$inb]=0;
          $nsymloopsmirna[$inb]=0;
          $nasymloopsmirna[$inb]=0;
          $nblgsoutmirna[$inb]=0;
          $nsymloopsoutmirna[$inb]=0;
          $nasymloopsoutmirna[$inb]=0;
         }
      #The mature microRNA found at the direction 5' to 3'
      if ($location[0] != -1 )
         {
          if ($hdrflag ==1 ){ print STDOUT  $hdrname;}

           print STDOUT  $title  ;
           $prevmirna = substr ($stem1,0,$location[0]+3);
           $cmirna = substr ($stem1,$location[0]+3,$location[1]-
$location[0]+1);
           $aftermirna = substr($stem1,3+$location[1]+1 );
           print STDOUT $prevmirna; print STDOUT
color("blue"),$cmirna,color("reset"); print STDOUT $aftermirna;
           print STDOUT $dashes;
           print STDOUT $stem2;

           #print distance from 5' direction
            $st5arm = substr($stm1,0,$location[0]);  $st3arm =
substr($stm1,$location[1]+1);
            $d1 = ($st5arm=~tr/ACGTUNacgtun//);  $d2 = ($st3arm
=~tr/ACGTUNacgtun//);
            $result = sprintf ("%4d ",$d1);
            print ST $result;
            #Deal with seed microRNA
            $n = $location[1] - $location[0] +1;
            $pstem1 = substr($stm1,$location[0],$n);
            $pdash = substr($dash,$location[0],$n);
            $pstem2 = substr($stm2,$location[0],$n);
            $flagformirna = 1 ;
            findBulges($pstem1,$pdash,$pstem2);
            $pstem1 =~ s/\-//g; $pstem2 =~s/\-//g;
            $summirna = $summirna + length($pstem1)+ length($pstem2);
            $summirnab = $summirnab + length($pstem1);
            print "\n", length($pstem1)+length($pstem2),"
",length($pstem1)," ";

            #Deal with 5' to seed microRNA
            $n = $location[1] - $location[0] +1;
            $pstem1 = substr($stm1,0,$location[0]);
            $pdash = substr($dash,0,$location[0]);
            $pstem2 = substr($stm2,0,$location[0]);
            $flagformirna=0;
            findBulges($pstem1,$pdash,$pstem2);
            $pstem1 =~ s/\-//g; $pstem2 =~s/\-//g;
            $sum5d = $sum5d + length($pstem1)+ length($pstem2);
            print length($pstem1)+length($pstem2)," ";
         }
      #The mature microRNA found at the direction 3' to 5'
#print to the features file the results of bulges loops length

 printOutBulgsLoopsLength(@nblgsmirna);
 printOutBulgsLoopsLength(@nsymloopsmirna);
 printOutBulgsLoopsLength(@nasymloopsmirna);
 printOutBulgsLoopsLength(@nblgsoutmirna);
 printOutBulgsLoopsLength(@nsymloopsoutmirna);
 printOutBulgsLoopsLength(@nasymloopsoutmirna);

}#End of function
############################################################################
#########################################################
sub printOutBulgsLoopsLength{
             (@arr) = @_;
my $outlier = 0 ;
my $result;
for ($k=8;$k<=$#arr;++$k)
   { $outlier = $outlier+ $arr[$k] ;}

for ($k=1 ; $k<=7; $k++)
  {
    $result = sprintf("%4d ",$arr[$k]);
    print ST $result;
  }
#print the outlier bigger than 7
$result = sprintf ("%4d ",$outlier);
print ST $result;

}
############################################################################
##########
# sub getMature() :
############################################################################
#########
sub getMature{
       my ($hdrname) = @_ ;
   my $flag = 1 ;
   $i = 0 ;
   $hdrname = lc ($hdrname);
   $arrlength = @mature;
  while ( $flag == 1 && $i < $arrlength)
    {  $line = lc($mature[$i] );
      if ( $line=~/$hdrname/ )
        { $st = $mature[$i+1] ;
          chomp ($st);
         $flag = 0 ;
        }# End of if ( )
      ++$i ;
    }#End of while
return $st ;

}#End of function
############################################################################
#####################

sub positionSubstring{
       my ($stem) = shift(@_);
       my ($mirna)= shift(@_);
  @letpos=();
  @lc=();

 push (@lc,-1);

 for ($i = 0 ; $i <length ($stem); $i++)
    { $let = substr($stem,$i,1);
     if (  $let ne '-'   )
```

```
        { push (@letpos,$i);}
    }

 $stem =~s/-//g;
 $location = index ($stem,$mirna);
  my $mm=$mirna;
  my $z=0;
 while ($location == -1 & $z <4 )
   {  $mm =  substr($mirna,0,length($mirna)-$z);
     $location = index ($stem,$mm);
     $z=$z+1;
  }
  $mirna = $mm;
 if ($location != -1 )
  { $start = $letpos[$location];
    $end   = $letpos[$location + length($mirna) - 1 ];
    shift (@lc);
    push (@lc,$start);
    push (@lc,$end);
    #my $result = sprintf("s:%4d e:%4d",$start,$end);
    #print ST $result;
      }

return @lc;
}#End of sub routine
############################################################################
#
# sub findBulges($stem1,$dashes,$stem2)
############################################################################
#
sub findBulges{
     my ($stem1) = shift (@_);
     my ($dash)  = shift (@_);
     my ($stem2)= shift (@_);
 my $nbulges = 0;
 my $nsloops = 0;
 my $nloopsnonsym = 0;
 my $sumb= 0 ;
 my $suml = 0 ;
 my $bp;
 #print ST "\n",$stem1,"\n",$dash,"\n",$stem2;
 while ($dash =~ m/\s+/g) {
    #print "\n", pos ($dash) - length($&) ," ", pos ($dash) ;
    $f = pos ($dash) - length($&) ;
    $l = pos ($dash) ;
    $s1 = substr($stem1,$f,$l-$f); #copy the corsponding part from stem1
    $s2 = substr($stem2,$f,$l-$f); #copy the corsponding part from stem2
   # print ST "\n",$s1, " ",$s2,"\n";
   # if (  ($s1 =~/\-+[ATCGUNatcgun]+/ ) || ($s2 =~/\-+[ATCGUNatcgun]+/) )
#Non symetric loops
 if (  ($s1 =~/\-+[ATCGUNatcgun]+/ ) || ($s1=~/[ATCGUNatcgun]+\-+/ ) ||
($s2 =~/[ATCGUNatcgun]+\-+/) ||($s2=~/\-+[ATCGUNatcgun]+/)  ) #Non symetric
loops
            { ++$nsloops; ++$nloopsnonsym ; $suml=$suml+$l-$f;
              $width = $l-$f;
              if ($flagformirna == 1 )
                   { ++$nasymloopsmirna[$width];
                     ++$nloopsmirna[$width]       } #number of loops with
length width
             else  {++$nasymloopsoutmirna[$width];
                    ++$nloopsoutmirna[$width]    }
              }#Non symetric loops
    else   { if ($s1 =~m/\-+/  || $s2 =~m/\-+/ ) #Pure bulges
               { ++$nbulges ; $sumb= $sumb+$l-$f ;
                 $width = $l-$f;
                 if ($flagformirna == 1)
                    { ++$nblgsmirna[$width];}
                 else { ++$nblgsoutmirna[$width];}
               } #pure bulges
              else  { ++$nsloops; $suml = $suml+$l-$f;
                       $width = $l-$f;
                       if ($flagformirna == 1)
                           { ++$nsymloopsmirna[$width];
                             ++$nloopsmirna[$width]  }
                       else { ++$nsymloopsoutmirna[$width];
                              ++$nloopsoutmirna[$width] }
                    } #pure loops ;
           }#End of else
    }#End of while
    $avgbulges = 0 ;
    $avgloops = 0 ;
    if ( $nbulges != 0 ) { $avgbulges = $sumb/$nbulges;
    if ($nsloops !=0 ) {$avgloops = $suml/$nsloops;}
    $stlen = ($stem1 =~tr/NACGTUnacgtu//) + ($stem2=~tr/NACGTUnacgtu//);
    $bp    = ($dash=~tr/\|//);
    $ll = $stlen;
    if ($stlen == 0 ) {$stlen = 1;}
    #print ST   $nbulges,  $sumb/$nbulges,  $nsloops ,  $suml/$nsloops ,
$nloopsnonsym;
    #$result = sprintf("%4d %4d %10.4f %4d %10.4f
%4d",$ll,$nbulges,$avgbulges,$nsloops,$avgloops,$nloopsnonsym);
    $result = sprintf("%4d %4d  %4d
%4d",$bp,$nbulges,$nsloops,$nloopsnonsym);
    print ST $result;
}#End of function
```

Appendix 8.3.A java program for feature generations from the alignment of seed sequences. This java code generates the miRNA pairing information is summarized over the seed region of the miRNA. The input dataset is the duplex structure of miRNA target interactions.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package new_datasetcsv;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import jxl.write.WriteException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
//Added
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.util.CellReference;
import org.apache.poi.xssf.streaming.SXSSFWorkbook;
//added
import java.io.FileWriter;
import com.csvreader.CsvWriter;
//import java.io.IOExceptio;


/**
 *
 * @author alisha
 */
public class new_datasetcsv {

    public static void main(String[] args) throws IOException,
WriteException {
        String Path_in = "/home/alisha/Downloads/CSV Code+Libraries/IN/";//
give the input foleder path
        String Path_out = "/home/alisha/Downloads/CSV
Code+Libraries/out/";// give the output folder path
        File folder = new File(Path_in);
        File[] listOfFiles = folder.listFiles();
        FileWriter writer = null;

        for (int f = 0; f < listOfFiles.length; f++) {
            System.out.println("Start " + listOfFiles[f].getName());
            //if (listOfFiles[f].getName().equalsIgnoreCase("positive0")) {
           // Workbook workbook = new SXSSFWorkbook();
            //Sheet sheet = workbook.createSheet("Datatypes in Java");
            //int rowNum = 0;
              CsvWriter csvOutput = new CsvWriter(new FileWriter(Path_out +
listOfFiles[f].getName().replace(".fasta", ".csv"),true),',');
```

```
            // TODO code application logic here
            BufferedReader br = null;
            FileReader fr = null;
            br = new BufferedReader(new FileReader(Path_in +
listOfFiles[f].getName()));
            //int i = 0;
            String sCurrentLine = null;
            String miRNA = null;
            String Mrna = null;
            String Rvalue = null;
            String R1 = null;
            String R2 = null;
            String R  = null;
             int match = 0;
             int mismatch  =0;
             int N_AU  =0;
             int N_GC = 0;
            ArrayList<String> features = new ArrayList<String>();
            int n_feat = 0;
            int flag1 = 0;
            int flag2 = 0;
            while ((sCurrentLine = br.readLine()) != null) {
                if (sCurrentLine.contains(">hsa-miR-")) {
                    String[] temp = sCurrentLine.split("\\s+");
                    miRNA = temp[0].replaceAll(">", "").trim();

                    Mrna = temp[1].trim();
                    flag1 = 1;
                    //  Rvalue.add(temp[2].replaceAll("R:", "").trim());

                }else if(sCurrentLine.contains("Forward:")){

                 //String[] temp1 = sCurrentLine.split("\\s+");
                 //R1 = temp1[7].replaceAll("R:", "").trim();
                 //R2=  " "+temp1[8]+" "+temp1[9];
                 //R = R1+R2;
                   R = sCurrentLine;
                   R = R.replaceAll("   Forward:      Score: ","" );
                   R = R.trim();
                   String temp1[] = R.split("\\s+");
                   temp1[7] = null;
                   temp1[8] = null;
                   temp1[9] = null;
                   temp1[10] = null;
                   temp1[11] = null;
                   temp1[12] = null;
                   R = temp1[4].replaceAll("R:", "")+" "+temp1[5]+"
"+temp1[6];

                } else if (sCurrentLine.contains("Ref:      5' ")) {
                    String ref = sCurrentLine;
                    sCurrentLine = br.readLine();
                    sCurrentLine = br.readLine();
                    String query = sCurrentLine;
                    ref = ref.replaceAll("Ref:      5'", "");
                    ref = ref.replaceAll("3'", "");
                    ref = ref.trim();
                    ref = ref.replaceAll("[a-z]", "");
                    ref = ref.trim();

                    query = query.replaceAll("Query:    3'", "");
                    query = query.replaceAll("5'", "");
                    query = query.trim();
                    query = query.replaceAll("[a-z]", "");
                    query = query.trim();
                    System.out.println("Ref " + ref + "  " + query);
                    System.out.println(R);
                    int start = 0;

                    if (ref.length() == query.length()) {
                        start = ref.length();

                    }
                    //features.add(new ArrayList<String>());
                    /*
                    if (ref.length() == query.length()) {
                        start = ref.length() - 1;
                        end = 0;
                    } else if (ref.length() > query.length()) {
                        int mis = ref.length() - query.length();
                        for (int i = 0; i < mis; i++) {
                            query = "*" + query;
                        }
                        start = ref.length() - 1;
                        end = mis;
                    }
                    */
                    for (int i = 0; i < start; i++) {
                        if (!ref.isEmpty() && !query.isEmpty()) {
                            //System.out.println("Length "+ref.length());
                            //int temp_num =
Integer.parseInt(temp[i].trim());
                            String feat_temp = "";
                            feat_temp = "" + ref.charAt(i) +
query.charAt(i);
                            //feat_temp=feat_temp+;
                            features.add(feat_temp);
                        }
                    }
                    flag2 = 1;

                }

                if (flag1 == 1 && flag2 == 1) {
                   // int colNum = 0;
                   /* Row row = sheet.createRow(rowNum++);
                    int colNum = 0;
                    Cell cell = row.createCell(colNum++);
                    cell.setCellValue((String) miRNA);
                    cell = row.createCell(colNum++);
                    cell.setCellValue((String) Mrna);*/
                    csvOutput.write(miRNA);
                    csvOutput.write(Mrna);


                    //setting up extra condition to check array boundary
indices//
                    for (int j = 0; j < features.size(); j++) {
                        String feat_temp = features.get(j);
                        double feat_num = -2;
                        if (feat_temp.contentEquals("AT") ||
feat_temp.contentEquals("GC")) {
```

```
                            match++;
                            if(feat_temp.contentEquals("GC")){
                             N_GC++;
                            }
                            feat_num = 1;
                        } else if (feat_temp.contentEquals("AU") ||
feat_temp.contentEquals("UG")) {
                            feat_num = 2;
                            if (feat_temp.contentEquals("AU")){
                                N_AU++;
                            }
                        } else if (feat_temp.contentEquals("GT") ||
feat_temp.contentEquals("TC") || feat_temp.contentEquals("AG") ||
feat_temp.contentEquals("AC")
                                || feat_temp.contentEquals("TU")) {
                            feat_num = 0;
                            mismatch++;
                        } else if (feat_temp.contentEquals("AA") ||
feat_temp.contentEquals("TT") || feat_temp.contentEquals("GG") ||
feat_temp.contentEquals("CC")) {
                            feat_num = 0;
                        } else if (feat_temp.contentEquals("A-") ||
feat_temp.contentEquals("T-") || feat_temp.contentEquals("G-") ||
feat_temp.contentEquals("C-")) {
                            feat_num = -1;
                        }

                    }
                      /* cell = row.createCell(colNum++);
                       cell.setCellValue((String) R);
                       cell = row.createCell(colNum++);
                       cell.setCellValue((int) match);
                       cell = row.createCell(colNum++);
                       cell.setCellValue((int) mismatch);
                       cell = row.createCell(colNum++);
                       cell.setCellValue((int) N_AU);
                       cell = row.createCell(colNum++);
                       cell.setCellValue((int) N_GC);*/
                      csvOutput.write(R);
                      csvOutput.write(String.valueOf(match));
                      csvOutput.write(String.valueOf(mismatch));
                      csvOutput.write(String.valueOf(N_AU));
                      csvOutput.write(String.valueOf(N_GC));
          //Start valaues (Again Loop)
          for (int j = 0; j < features.size() ; j++) {
                       String feat_temp = features.get(j);
                       double feat_num = -2;
                       if (feat_temp.contentEquals("AT") ||
feat_temp.contentEquals("GC")) {

                            feat_num = 1;
                        } else if (feat_temp.contentEquals("AU") ||
feat_temp.contentEquals("UG")) {
                            feat_num = 2;

                        } else if (feat_temp.contentEquals("GT") ||
feat_temp.contentEquals("TC") || feat_temp.contentEquals("AG") ||
feat_temp.contentEquals("AC")
                                || feat_temp.contentEquals("TU")) {
                            feat_num = 0;
                        } else if (feat_temp.contentEquals("AA") ||
feat_temp.contentEquals("TT") || feat_temp.contentEquals("GG") ||
feat_temp.contentEquals("CC")) {
                            feat_num = 0;
                        } else if (feat_temp.contentEquals("A-") ||
feat_temp.contentEquals("T-") || feat_temp.contentEquals("G-") ||
feat_temp.contentEquals("C-")) {
                            feat_num = -1;
                        }
                        /*cell = row.createCell(colNum++);
                        cell.setCellValue((Double) feat_num);*/
                       csvOutput.write(String.valueOf(feat_num));

                    }
     //End Values
                    csvOutput.endRecord();
                    features.clear();
                    flag1 = 0;
                    flag2 = 0;
                    match  =0;
                    mismatch = 0;
                    N_AU =0;
                    N_GC = 0;
                    n_feat++;
                    R1 =null;
                    R2= null;
                    R = null;
                }
                if (n_feat == 1000000) {
                    break;
                }
            }

            try {

                 csvOutput.close();
                FileOutputStream outputStream = new
FileOutputStream("/home/alisha/Downloads/CSV
Code+Libraries/out/empty.csv");
                //workbook.write(outputStream);*/
                // workbook.close();

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

            System.out.println("Done" + listOfFiles[f].getName());
        }
    }
}
```

Appendix 8.4. A java program for feature generations from the alignment of outseed sequences. This java code generates the miRNA pairing information is summarized over the 3' region and outseed region of the miRNA. The input dataset is the duplex structure of miRNA target interactions.

```
package new_datasetsmall;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import jxl.write.WriteException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
//Added
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.util.CellReference;
import org.apache.poi.xssf.streaming.SXSSFWorkbook;
//added
import java.io.FileWriter;
import com.csvreader.CsvWriter;
//import java.io.IOExceptio;
/*** @author alisha
 */
public class new_datasetsmall {

    public static void main(String[] args) throws IOException,
WriteException {
        String Path_in = "/home/alisha/Downloads/Final Updated Code+
Libraries/in_positive/";// give the input foleder path
        String Path_out = "/home/alisha/Downloads/Final Updated Code+
Libraries/pos_out_lowercase/";// give the output folder path
        File folder = new File(Path_in);
        File[] listOfFiles = folder.listFiles();
        FileWriter writer = null;

        for (int f = 0; f < listOfFiles.length; f++) {
            System.out.println("Start " + listOfFiles[f].getName());
            //if (listOfFiles[f].getName().equalsIgnoreCase("positive0")) {
           // Workbook workbook = new SXSSFWorkbook();
            //Sheet sheet = workbook.createSheet("Datatypes in Java");
            //int rowNum = 0;
              CsvWriter csvOutput = new CsvWriter(new FileWriter(Path_out +
listOfFiles[f].getName().replace(".fasta", ".csv"),true),',');
            // TODO code application logic here
            BufferedReader br = null;
            FileReader fr = null;
            br = new BufferedReader(new FileReader(Path_in +
listOfFiles[f].getName()));
            //int i = 0;
            String sCurrentLine = null;
            String miRNA = null;
            String Mrna = null;
            String Rvalue = null;
            String R = null;
 int match = 0;
 int mismatch  =0;
 int othermismatch = 0;
 int N_AU  =0;
 int N_GC = 0;
 int N_UG = 0;

ArrayList<String> features = new ArrayList<String>();
int n_feat = 0;
int flag1 = 0;
int flag2 = 0;
csvOutput.write("miRNA");
csvOutput.write("Mrna");
csvOutput.write("R");
csvOutput.write("Total-Matches");
csvOutput.write("Total-Mismatces");
csvOutput.write("Total-gc");
csvOutput.write("Total-au");
csvOutput.write("Total-ug");
csvOutput.write("Total-Othermismatces");
csvOutput.endRecord();

while ((sCurrentLine = br.readLine()) != null) {
    if (sCurrentLine.contains(">hsa-miR-")) {
        String[] temp = sCurrentLine.split("\\s+");
        miRNA = temp[0].replaceAll(">", "").trim();

        Mrna = temp[1].trim();
        flag1 = 1;
        //  Rvalue.add(temp[2].replaceAll("R:", "").trim());

    }else if(sCurrentLine.contains("Forward:")){
     R = sCurrentLine;
     //System.out.println(R);
     R = R.replaceAll("   Forward:  Score: ","");
     R = R.replaceAll("Q:", "");
     R = R.replaceAll("R:", "");
     R = R.replaceAll("Align Len ", "");
     //System.out.println(R);
     R = R.replaceAll("Energy:", "");
     //System.out.println(R);
     R = R.substring(21);
     R = R.replaceAll("to", "");
     String[] temp1 = R.split("\\s+");
     R = temp1[0].trim()+" "+"to"+" "+temp1[1].trim();

    } else if (sCurrentLine.contains("Ref:      5' ")) {
        String ref = sCurrentLine;
        sCurrentLine = br.readLine();
        sCurrentLine = br.readLine();
        String query = sCurrentLine;
        ref = ref.replaceAll("Ref:      5'", "");
        ref = ref.replaceAll("3'", "");
        ref = ref.trim();
        ref = ref.replaceAll("[A-Z]", "");
        ref = ref.replaceAll("[\\-]", "");
        ref = ref.trim();


        query = query.replaceAll("Query:    3'", "");
        query = query.replaceAll("5'", "");
```

```
                    query = query.trim();
                    query = query.replaceAll("[A-Z]", "");
                    query = query.replaceAll("-", "");
                    query = query.trim();
                    System.out.println("Ref " + ref + "  " + query);
                   // System.out.println(R);
                    int start = 0;

                    if (ref.length() == query.length()) {
                        start = ref.length();

                    }
                   /*  //features.add(new ArrayList<String>());
                    int start = -1;
                    int end = -1;
                    if (ref.length() == query.length()) {
                        start = ref.length() - 1;
                        end = 0;
                    } else if (ref.length() > query.length()) {
                        int mis = ref.length() - query.length();
                        for (int i = 0; i < mis; i++) {
                            query = "*" + query;
                        }
                        start = ref.length() - 1;
                        end = mis;
                    }*/
                    for (int i = 0; i < start; i++) {
                        if (!ref.isEmpty() && !query.isEmpty()) {
                            //System.out.println("Length "+ref.length());
                            //int temp_num =
Integer.parseInt(temp[i].trim());
                            String feat_temp = "";
                            feat_temp = "" + ref.charAt(i) +
query.charAt(i);

                            //feat_temp=feat_temp+;
                            features.add(feat_temp);
                            //System.out.println(feat_temp);
                        }
                    }
                    flag2 = 1
                }

                if (flag1 == 1 && flag2 == 1) {
                   // int colNum = 0;
                   /* Row row = sheet.createRow(rowNum++);
                    int colNum = 0;
                    Cell cell = row.createCell(colNum++);
                    cell.setCellValue((String) miRNA);
                    cell = row.createCell(colNum++);
                    cell.setCellValue((String) Mrna);*/
                    csvOutput.write(miRNA);
                    csvOutput.write(Mrna);


                    //setting up extra condition to check array boundary
indices//
                    for (int j = 0; j < features.size(); j++) {
                        String feat_temp = features.get(j);
                        //double feat_num = -2;
                        if (feat_temp.contentEquals("at") ||
feat_temp.contentEquals("gc")||feat_temp.contentEquals("ta") ||
feat_temp.contentEquals("cg")) {
                            match++;

if(feat_temp.contentEquals("gc")||feat_temp.contentEquals("cg")){
                             N_GC++;
                            }
                           // feat_num = 1;
                        } else if (feat_temp.contentEquals("au") ||
feat_temp.contentEquals("ug")) {
                            //feat_num = 2;

if(feat_temp.contentEquals("ug")||feat_temp.contentEquals("gu")){
                            N_UG++;
                            }

                            if
(feat_temp.contentEquals("au")||feat_temp.contentEquals("ua")){
                                N_AU++;
                            }
                        } else if (feat_temp.contentEquals("gt") ||
feat_temp.contentEquals("tc") || feat_temp.contentEquals("ag") ||
feat_temp.contentEquals("ac")
                                || feat_temp.contentEquals("tu") ||
feat_temp.contentEquals("cu")||feat_temp.contentEquals("tg") ||
feat_temp.contentEquals("ct") || feat_temp.contentEquals("ga") ||
feat_temp.contentEquals("ca")
                                || feat_temp.contentEquals("ut") ||
feat_temp.contentEquals("uc")) {
                            //feat_num = 0;
                            mismatch++;
                        } else if (feat_temp.contentEquals("aa") ||
feat_temp.contentEquals("tt") || feat_temp.contentEquals("gg") ||
feat_temp.contentEquals("cc")) {
                           // feat_num = 0;
                           othermismatch++;
                        } else if (feat_temp.contentEquals("A-") ||
feat_temp.contentEquals("T-") || feat_temp.contentEquals("G-") ||
feat_temp.contentEquals("C-")) {
                           // feat_num = -1;
                        }

                    }
                       /* cell = row.createCell(colNum++);
                        cell.setCellValue((String) R);
                        cell = row.createCell(colNum++);
                        cell.setCellValue((int) match);
                        cell = row.createCell(colNum++);
                        cell.setCellValue((int) mismatch);
                        cell = row.createCell(colNum++);
                        cell.setCellValue((int) N_AU);
                        cell = row.createCell(colNum++);
                        cell.setCellValue((int) N_GC);*/
                       csvOutput.write(R);
                       csvOutput.write(String.valueOf(match));
csvOutput.write(String.valueOf(mismatch));
csvOutput.write(String.valueOf(N_GC));

csvOutput.write(String.valueOf(N_AU));csvOutput.write(String.valueOf(N_UG))
;        csvOutput.write(String.valueOf(othermismatch));
```

```
     //Start valaues (Again Loop)

     //End Values
                    csvOutput.endRecord();
                    features.clear();
                    flag1 = 0;
                    flag2 = 0;
                    match  =0;
                    mismatch = 0;
                    othermismatch = 0;
                    N_UG = 0;
                    N_AU =0;
                    N_GC = 0;
                    n_feat++;
                }
                if (n_feat == 22704425) {
                    break;
                }
            }
            try {

                csvOutput.close();
                FileOutputStream outputStream = new
FileOutputStream("/home/alisha/Downloads/Final Updated Code+
Libraries/pos_out_upparcase/empty.csv");
                //workbook.write(outputStream);*/
                // workbook.close();

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            System.out.println("Done" + listOfFiles[f].getName());
        }
    }
}
```

Appendix 8.5. An example of training dataset. After feature generation from several code and algorithms, 95 features are generated of each miRNA target interaction. After several pre-processing stages (section 2.4) resultant dataset used for training of AHDR algorithm. The input dataset was saved as the excel file.

miRNA	mRNA	Seed_Interacti	LCP	m/e	BRL	AlignLen	Tot_mismatch	align5'	Score	Total_AU	PLC	NP	P3'	d5	numirna	Tot_Mat	Total_GC	Energy	Tot_gc
hsa-miR-7159-5p	ENST00000540285	0	9	-18.9	16	22	8	8	142	3	7	14	7	15	6	3	4	-7.7	1
hsa-miR-381-3p	ENST00000261606	1	11	-19.6	16	22	8	8	142	4	5	14	8	15	6	1	5	-8.4	1
hsa-miR-448	ENST00000545174	1	9	-19.4	18	22	8	8	142	4	1	14	8	15	3	3	4	-8.2	1
hsa-miR-103a-2-5p	ENST00000395753	1	13	-19.4	14	22	8	8	142	5	1	13	7	15	6	1	3	-8.2	1
hsa-miR-3682-5p	ENST00000298004	0	12	-20.7	13	22	8	8	142	5	1	12	7	15	4	3	3	-9.0	1
hsa-miR-541-5p	ENST00000368892	0	9	-6.5	18	22	8	8	142	3	5	15	4	15	6	3	2	-9.0	0
hsa-miR-1276	ENST00000337428	0	13	-9.5	40	22	8	8	142	4	27	17	7	15	6	4	2	-8.7	1
hsa-miR-3120-3p	ENST00000373827	1	7	-17.4	23	22	8	8	142	4	16	12	7	15	7	3	3	-8.9	1
hsa-miR-489-3p	ENST00000357325	1	9	-9.0	16	22	8	8	142	6	6	13	0	15	6	2	1	-8.5	0
hsa-miR-4659a-3p	ENST00000542285	1	9	-14.8	15	22	8	8	142	6	1	13	6	15	6	3	1	-8.4	1
hsa-miR-146a-5p	ENST00000371566	1	9	-19.4	18	22	3	2	142	4	1	14	8	9	7	1	4	-8.7	1
hsa-miR-8485	ENST00000265909	0	13	-19.4	14	22	3	2	142	5	1	13	7	9	7	1	3	-7.9	1
hsa-miR-5002-5p	ENST00000360013	0	12	-20.7	13	22	3	2	142	5	1	12	7	9	6	1	3	-8.8	1
hsa-miR-101-5p	ENST00000440884	1	9	-6.5	18	22	3	2	142	3	5	15	4	9	7	0	2	-8.8	1
hsa-miR-4477a	ENST00000319801	1	13	-9.5	40	22	3	2	142	4	27	17	7	9	7	1	2	-8.5	1
hsa-miR-375	ENST00000635253	1	7	-17.4	23	22	3	2	142	4	16	12	7	9	7	1	3	-8.7	1

Appendix 8.6. AHDR architecture. In training AHDR classifier, several deep learning packages such as Keras and sklearn is used to train AHDR classifier. Several machine learning algorithms have been applied before and after feature selection algorithms. The performance of classifier was presented by generating ACC, AUC, TPR, TNR, and MCC (as shown in Table 6)

```
#importing Libraries used
import pandas as pd
import numpy as np
##usefull preprocessing and etc classes
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from keras import backend as K
from sklearn.metrics import precision_score
from sklearn.metrics import auc
from sklearn.metrics import recall_score, confusion_matrix
##ml model classes
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn import svm
##importing Feature selection Classes
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.svm import SVC
from sklearn.feature_selection import RFE
from sklearn.linear_model import RandomizedLogisticRegression
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVC
from sklearn.cross_validation import KFold

# # Defining Machine learning Models

# ### Logisitic Regression
def LogisticModel(xTrain, yTrain, xTest, yTest):
    logreg = LogisticRegression(C=1e5)

    logreg.fit(xTrain,yTrain)
    yPredict = logreg.predict(xTest)
    precisionScore=precision_score(yTest, yPredict, average="binary")
    tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
    recall=recall_score(yTest, yPredict)
    score =logreg.score(xTest, yTest)
    return  logreg,score, precisionScore,recall, tn, fp, fn, tp, yPredict


# ### KNN model
def KNNModel(xTrain, yTrain, xTest, yTest):
    neighModel = KNeighborsClassifier(n_neighbors=5)
    neighModel.fit(xTrain,yTrain)
    yPredict = neighModel.predict(xTest)
    precisionScore=precision_score(yTest, yPredict, average="binary")
    tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
    recall=recall_score(yTest, yPredict)
    score =neighModel.score(xTest, yTest)
    return  neighModel, score, precisionScore,recall, tn, fp, fn, tp, 
yPredict

# ### NN
def NNModelFunction(xTrain, yTrain, xTest, yTest, DenseNumber):
model2 = Sequential()
model2.add(Dense(16, activation='sigmoid', input_dim=DenseNumber))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(loss='binary_crossentropy',optimizer='rmsprop', 
metrics=['accuracy'])
model2.fit(xTrain,yTrain, epochs=60, batch_size=128)
scores= model2.evaluate(xTest, yTest)
yPredict = model2.predict_classes(xTest, batch_size=128)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
return model2,scores[1], precisionScore,recall, tn, fp, fn, tp, yPredict

 ### GradientBoostingClassifier
def boostingalgo(xTrain, yTrain, xTest, yTest):
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, 
max_depth=1, random_state=0).fit(xTrain, yTrain)
yPredict = clf.predict(xTest)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
score =clf.score(xTest, yTest)
return  clf, score, precisionScore,recall, tn, fp, fn, tp, yPredict

# ### Random Forest
def randomForestModel(xTrain, yTrain, xTest, yTest):
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(xTrain, yTrain)
yPredict = clf.predict(xTest)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
score =clf.score(xTest, yTest)
return  clf, score, precisionScore,recall, tn, fp, fn, tp, yPredict

####### Gradientboosting
def gradientboostingModel(xTrain, yTrain, xTest, yTest):
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, 
max_depth=1, random_state=0).fit(xTrain, yTrain)
yPredict = clf.predict(xTest)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
score =clf.score(xTest, yTest)
return  clf, score, precisionScore,recall, tn, fp, fn, tp, yPredict

# ### BernoulliNB
def bernoModel(xTrain, yTrain, xTest, yTest):
clf = BernoulliNB()
clf.fit(xTrain, yTrain)
```

```
yPredict = clf.predict(xTest)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
score =clf.score(xTest, yTest)
return  clf, score, precisionScore,recall, tn, fp, fn, tp, yPredict

# ### Decision Tree
def DecisionTreeModel(xTrain, yTrain, xTest, yTest):
treeClas= tree.DecisionTreeClassifier()
treeClas.fit(xTrain,yTrain)
yPredict = treeClas.predict(xTest)
precisionScore=precision_score(yTest, yPredict, average="binary")
tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
recall=recall_score(yTest, yPredict)
score =treeClas.score(xTest, yTest)
return  treeClas,score, precisionScore,recall, tn, fp, fn, tp,yPredict

# ### SVM
def SVMModel(xTrain, yTrain, xTest, yTest):
    svmTModel = svm.SVC()

    svmTModel.fit(xTrain,yTrain)

    yPredict = svmTModel.predict(xTest)
    precisionScore=precision_score(yTest, yPredict, average="binary")
    tn, fp, fn, tp = confusion_matrix(yTest, yPredict).ravel()
    recall=recall_score(yTest, yPredict)
    score =svmTModel.score(xTest, yTest)
    return  svmTModel, score, precisionScore,recall, tn, fp, fn, tp,
yPredict

# # Feature Selection
# ### VarianceThreshold
def VarianceThresholdMethod(Input, output):
    selectionModel =VarianceThreshold(threshold=0)
    selectionModel.fit(Input, output)
    indexes=selectionModel.get_support(indices=True)
    columns=list(Input.columns)
    selectedList =[]
    for i in indexes:
        selectedList.append(columns[i])
    return selectedList

# ### Chi `Square
def ChiSqure(Input, output):
    model = SelectKBest(chi2, k=10)
    temp =Input.copy()
    temp=temp.drop(["POS1","POS7",  "m/e"  , "POS9"
,"POS8","POS4","POS3","POS2","POS9","POS4","POS10"
               ,"POS6","Energy", "POS5","PyloP_Flanking","PhyloP_Stem"],
axis=1)
    model.fit(temp, output)
    indexes= model.get_support(indices=True)
    columns=list(temp.columns)
    selectedList =[]
    for i in indexes:
        selectedList.append(columns[i])
    return selectedList

# ### Recursive feature elimination
```

```
def RecursiveFeatureEmlination(Input, output):
    svc = SVC(kernel="linear", C=1)
    rfe = RFE(estimator=svc, n_features_to_select=1, step=1)
    rfe.fit(Input, output)
    indexes= rfe.get_support(indices=True)
    columns=list(Input.columns)
    selectedList =[]
    for i in indexes:
        selectedList.append(columns[i])
    return selectedList

# ### RandomizedLogisticRegression
def RLRFeatureSelection(Input, output):
    clf = RandomizedLogisticRegression()
    clf = clf.fit(Input, output)
    indexes=clf.get_support(indices=True)
    columns=list(Input.columns)
    selectedList =[]
    for i in indexes:
        selectedList.append(columns[i])
    return selectedList

# ### Tree-based feature selection
def treeBasedSelection(Input, output):
clf = ExtraTreesClassifier()
clf = clf.fit(Input, output)
    model = SelectFromModel(clf, prefit=True)
    indexes=model.get_support(indices=True)
    columns=list(Input.columns)
    selectedList =[]
    for i in indexes:
selectedList.append(columns[i])
    return selectedList

# ### LASSO
def lasso(Input, outout):
    lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(Input, output)
    model = SelectFromModel(lsvc, prefit=True)
    columns=list(Input.columns)
    indexes = list(model.get_support(indices=True))
    selectedList =[]
    for i in indexes:
        selectedList.append(columns[i])
    return selectedList

#### Now loading Dataset
df = pd.read_csv("prepocessed_numeric_fullDataSet.csv")
Input=(df.loc[:, df.columns != 'seed'])
output = (df['seed'])

#### DFFN without Using Feature selection
X_train, X_test, y_train, y_test = train_test_split(Input,output,
test_size=0.3, random_state=5)

DNN(np.array(X_train), np.array(y_train),  np.array(X_test),
            np.array(y_test) ,X_train.shape[1])

# #### APPLYING DEEP LEARNING ON LASSO SELECTED FEATURES
one=lesso(Input, output)
two=lesso(Input, output)
three=lesso(Input, output)
```

```
commonLassoFeatures=list(set(one)&set(two)&set(three))
print(len(commonLassoFeatures))
X_train, X_test, y_train, y_test =
train_test_split(Input[commonLassoFeatures],output, test_size=0.3,
random_state=5)
x_train.shape
DNN(np.array(X_train), np.array(y_train),  np.array(X_test),
            np.array(y_test) ,X_train.shape[1])
X_train.shape[1]

#### APPLYING ALL ML MODELS WITHOUT USING ANY FEATURE SELECTION METHOD
##### Separting Data into Testing and Training
X_train, X_test, y_train, y_test = train_test_split(Input,output,
test_size=0.3, random_state=5)

# #### After division applying ML model
LR_Model, LR_score, LR_precisionScore,LR_recall, LR_tn, LR_fp, LR_fn,
LR_tp, LR_yPredict=LogisticModel(X_train,y_train, X_test, y_test )
LR_score

# In[36]:
BA_Model, BA_score, BA_precisionScore,BA_recall, BA_tn, BA_fp, BA_fn,
BA_tp, BA_yPredict=boostingalgo(X_train, y_train, X_test, y_test)

# In[37]:
BN_Model, BN_score, BN_precisionScore,BN_recall, BN_tn, BN_fp, BN_fn,
BN_tp, BN_yPredict=bernoModel(X_train,y_train, X_test, y_test)
BN_score

# In[39]:
DT_Model, DT_score, DT_precisionScore,DT_recall, DT_tn, DT_fp, DT_fn,
DT_tp, DT_yPredict=DecisionTreeModel(X_train,y_train, X_test, y_test)

# In[40]:
SVM_Model, SVM_score, SVM_precisionScore,SVM_recall, SVM_tn, SVM_fp,
SVM_fn, SVM_tp, SVM_yPredict=SVMModel(X_train,y_train, X_test,y_test)

# In[41]:
NN_Model, NN_score, NN_precisionScore,NN_recall, NN_tn, NN_fp, NN_fn,
NN_tp, NN_yPredict=NNModelFunction(np.array(X_train),np.array(y_train),
np.array(X_test), np.array(y_test) ,X_train.shape[1])
NN_score

# In[43]:
DNN_Model, DNN_score, DNN_precisionScore, DNN_recall, DNN_tn, DNN_fp,
DNN_fn, DNN_tp, DNN_yPredict=DNN(np.array(X_train), np.array(y_train),
np.array(X_test), np.array(y_test),X_train.shape[1])
DNN_score

### After Feature selection apply ML models on it
#### Variance threshold applying
InputColumnsVari=VarianceThresholdMethod (Input, output)
len(InputColumnsVari)
VInput = Input[InputColumnsVari]
VX_train, VX_test, Vy_train, Vy_test = train_test_split(VInput,output,
test_size=0.3, random_state=5)

# In[47]:
KNNModel(VX_train,Vy_train, VX_test, Vy_test)

# In[48]:
```

```
bernoModel(VX_train,Vy_train, VX_test, Vy_test)

# In[49]:
SVMModel(VX_train,Vy_train, VX_test, Vy_test)

# In[50]:
NNModelFunction(np.array(VX_train),Vy_train,
                np.array(VX_test), Vy_test, 70)

# In[51]:
DNN(np.array(VX_train),Vy_train, np.array(VX_test), Vy_test,
len(InputColumns))

##### Tree-based feature selection

oneT=treeBasedSelection(Input, output)
twoT=treeBasedSelection(Input, output)
threeT=treeBasedSelection(Input, output)
commonTreeFeatures=list(set(oneT)&set(twoT)&set(threeT))
treeInput = Input[commonTreeFeatures]
len(commonTreeFeatures)
TX_train, TX_test, Ty_train, Ty_test = train_test_split(treeInput, output,
test_size=0.3, random_state=5)

# In[54]:
print(LogisticModel(TX_train,Ty_train, TX_test, Ty_test))

#### Recursive feature elimination
oneR=RecursiveFeatureEmlination(Input, output)
twoR=RecursiveFeatureEmlination(Input, output)
threeR=RecursiveFeatureEmlination(Input, output)
commonReFeatures=list(set(oneR)&set(twoR)&set(threeR))
len(commonReFeatures)

# In[ ]:
ReInput = Input[commonReFeatures]
RX_train, RX_test, Ry_train, Ry_test = train_test_split(ReInput,
                                                        output,
test_size=0.3, random_state=5)

# In[ ]:
print(LogisticModel(RX_train,Ry_train, RX_test, Ry_test))

# In[ ]:
bernoModel(RX_train,Ry_train, RX_test, Ry_test)

# In[ ]:
KNNModel(RX_train,Ry_train, RX_test, Ry_test)

# In[ ]:
SVMModel (RX_train, Ry_train, RX_test, Ry_test)

# In[ ]:
NNModelFunction (np.array (RX_train), Ry_train, np.array (RX_test),
Ry_test, len (commonReFeatures))

# In[ ]:
DNN (np.array (RX_train), Ry_train, np.array (RX_test), Ry_test, len
(commonReFeatures))

# ### Apply Lasso First and then take those Features and train ML Models
```

```
one=lesso(Input, output)
two=lesso(Input, output)
three=lesso(Input, output)

# In[21]:
commonLassoFeatures=list(set(one)&set(two)&set(three))
print(len(commonLassoFeatures))

# In[22]:
LassoInput = np.array(Input[commonLassoFeatures])
input_number=len(commonLassoFeatures)
input_number

# ## 5 Cross Validation
kf_total = KFold(len(LassoInput), n_folds=5, shuffle=True, random_state=4)
DNNR=[]
precisionR=[]
recallR=[]
TruePositive =[]
FalsePositive =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])

    model, score, precision, recall, TN, FP, FN, TP,
yPredict=DNN(np.array(TrainX),
np.array(Trainy),np.array(TestX),np.array(Testy),input_number)
    DNNR.append(score)
    precisionR.append(precision)
    recallR.append(recall)
    TruePositive.append(TP)
    FalsePositive.append(FP)
    print("!!!!!")

print ("Accuracy:")
print(DNNR)
print("Precision")
print(precisionR)
print("ReCall")
print(recallR)
print("TruePositive")
print(TruePositive )
print("False Posisive")
print(FalsePositive )

# ### 10 cross validation
kf_total = KFold(len(LassoInput), n_folds=10, shuffle=True, random_state=4)
# In[62]:
LR10=[]
precisionR10=[]
recallR10=[]
TruePositive10 =[]
FalsePositive10 =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])
    model,score, precision, recall, TN, FP, FN, TP,
yPredict=LogisticModel(TrainX, Trainy, TestX,Testy )
    LR10.append(score)
    precisionR10.append(precision)
    recallR10.append(recall)
    TruePositive10.append(TP)
    FalsePositive10.append(FP)
    print("!!!!!")

# In[63]:
print ("Accuracy:")
print(np.mean(LR10))
print("Precision")
print(np.mean(precisionR10))
print("ReCall")
print(np.mean(recallR10))
print("TruePositive")
print(np.mean(TruePositive10) )
print("False Posisive")
print(np.mean(FalsePositive10) )

# #### multivariate Bernoulli Naïve Bayes
NB10=[]
precisionR10=[]
recallR10=[]
TruePositive10 =[]
FalsePositive10 =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])
    model,score, precision, recall, TN, FP, FN, TP,
yPredict=bernoModel(TrainX, Trainy, TestX,Testy )
    NB10.append(score)
    precisionR10.append(precision)
    recallR10.append(recall)
    TruePositive10.append(TP)
    FalsePositive10.append(FP)
    print("!!!!!")

print ("Accuracy:")
print(np.mean(NB10))
print("Precision")
print(np.mean(precisionR10))
print("ReCall")
print(np.mean(recallR10))
print("TruePositive")
print(np.mean(TruePositive10) )
print("False Posisive")
print(np.mean(FalsePositive10) )

SVM10=[]
precisionR10=[]
recallR10=[]
TruePositive10 =[]
FalsePositive10 =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])
    model,score, precision, recall, TN, FP, FN, TP,
yPredict=SVMModel(TrainX, Trainy, TestX,Testy )
    SVM10.append(score)
    precisionR10.append(precision)
    recallR10.append(recall)
    TruePositive10.append(TP)
    FalsePositive10.append(FP)
    print("!!!!!")

print ("Accuracy:")
print(np.mean(SVM10))
print("Precision")
print(np.mean(precisionR10))
print("ReCall")
print(np.mean(recallR10))
print("TruePositive")
print(np.mean(TruePositive10) )
print("False Posisive")
print(np.mean(FalsePositive10) )

# #### NN
NNR10=[]
precisionR10=[]
recallR10=[]
TruePositive10 =[]
FalsePositive10 =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])

    model,score, precision, recall, TN, FP, FN, TP,
yPredict=NNModelFunction(np.array(TrainX), np.array(Trainy)
                    ,
                    np.array(TestX),np.array(Testy)
                    ,len(commonLassoFeatures))
    NNR10.append(score)
    precisionR10.append(precision)
    recallR10.append(recall)
    TruePositive10.append(TP)
    FalsePositive10.append(FP)
    print("!!!!!")

# In[67]:
print ("Accuracy:")
print(np.mean(NNR10))
print("Precision")
print(np.mean(precisionR10))
print("ReCall")
print(np.mean(recallR10))
print("TruePositive")
print(np.mean(TruePositive10) )
print("False Posisive")
print(np.mean(FalsePositive10) )

# #### DNN
DNNR10=[]
precisionR10=[]
recallR10=[]
```

```
TruePositive10 =[]
FalsePositive10 =[]
for train, test in kf_total:
    TrainX=pd.DataFrame(LassoInput[list(train),:])
    Trainy=pd.DataFrame(output[list(train)])
    TestX=pd.DataFrame(LassoInput[list(test),:])
    Testy=pd.DataFrame(output[list(test)])

    model,score, precision, recall, TN, FP, FN, TP,
yPredict=DNN(np.array(TrainX), np.array(Trainy)
                    ,
                    np.array(TestX),np.array(Testy)
                    ,len(commonLassoFeatures))
    DNNR10.append(score)
    precisionR10.append(precision)
    recallR10.append(recall)
    TruePositive10.append(TP)
    FalsePositive10.append(FP)
    print("!!!!!")

print ("Accuracy:")
print(np.mean(DNNR10))
print("Precision")
print(np.mean(precisionR10))
print("ReCall")
print(np.mean(recallR10))
print("TruePositive")
print(np.mean(TruePositive10) )
print("False Posisive")
print(np.mean(FalsePositive10) )

#### Finding embedded score for each Feature
#### LASSO Features Accuracy
LassoDict= dict()
for l in commonLassoFeatures:
X_train, X_test, y_train, y_test = train_test_split(
Input[l],output, test_size=0.3,
random_state=5)score=NNModelFunction(X_train,y_train, X_test, y_test,1)
LassoDict[l]=score

# In[71]:
LassoDict

# ##### Our FInal Model is DNN on LASSO Features
X_train, X_test, y_train, y_test =
train_test_split(Input[commonLassoFeatures],output, test_size=0.3,
random_state=5)

# In[73]:
model,score, precision, recall, TN, FP, FN, TP,
yPredict=DNN(np.array(X_train), np.array(y_train),
np.array(X_test),np.array(y_test),len(commonLassoFeatures))

# In[74]:
df_pre=pd.DataFrame(X_test)

# ###### Prediction Probablity

# In[75]:
y_predicted_prob = model.predict(np.array(X_test))
```

```
# In[76]:
np.set_printoptions(formatter={'float_kind':'{:f}'.format})
pd.options.display.float_format = '{:,.20f}'.format

# In[77]:
predicted_array=np.transpose(np.array(y_predicted_prob))[0]

# In[78]:
df_pre["Prediction Probablity"]=predicted_array

# #### Loading Number values of miRNA and MRNA
# In[79]:
df_miRNA = pd.read_csv("miRNA_info.csv")

# In[ ]:
df_miRNA.head()

# In[ ]:
df_mRNA = pd.read_csv("mRNA_info.csv")

# In[ ]:
cross_Prediction
=df_pre[["miRNA","mRNA","binding_region_length","Prediction Probablity"]]
# In[ ]:
cross_Prediction["miRNA"]=cross_Prediction["miRNA"].replace(list(df_miRNA["
number"]),list(df_miRNA["value"]))
# In[ ]:
cross_Prediction["mRNA"]=cross_Prediction["mRNA"].replace(list(df_mRNA["num
ber"]),list(df_mRNA["value"]))

cross_Prediction.to_csv("Predictions)
```

Appendix 8.7. Test dataset of miRNA target. For performance evaluation of AHDR, experimentally verified dataset used as a test dataset, which is retrieved from strong experimental evidence such as Luciferase reporter assay, green fluorescent protein reporter assay, Reverse transcriptase-PCR, and Pulsed SILAC.

miRNA	**Total experimentally verified targets**
hsa-miR-140-5p	85
hsa-miR-133a-3p	110
hsa-miR-665	540
hsa-miR-448	48
hsa-miR-515-5p	185
hsa-miR-6891-5p	111
hsa-miR-3940-5p	62
hsa-miR-431-3p	5
hsa-miR-6832-5p	237
hsa-miR-103a-2-5p	57
hsa-miR-6877-5p	35
hsa-miR-1298-3p	45
hsa-miR-3682-5p	50
hsa-miR-3155a	97
hsa-miR-5089-5p	334
hsa-miR-648	67
hsa-miR-367-5p	119
hsa-miR-6740-5p	42
hsa-miR-450a-2-3p	75
hsa-miR-6793-3p	144
hsa-miR-3622a-3p	100
hsa-miR-541-5p	80
hsa-miR-6515-3p	181
hsa-miR-1276	141
hsa-miR-6499-3p	519
hsa-miR-2276-3p	145
hsa-miR-3120-3p	114
hsa-miR-6780b-5p	178
hsa-miR-4769-5p	59
hsa-miR-1306-5p	183
hsa-miR-519b-3p	314
hsa-miR-489-3p	60
hsa-miR-23a-3p	63
hsa-miR-6796-5p	51
hsa-miR-675-3p	52
hsa-miR-2110	96
hsa-miR-1269a	30
hsa-miR-483-3p	157
hsa-miR-4659a-3p	221

miRNA	Total experimentally verified targets
hsa-miR-146a-5p	168
hsa-miR-1289	76
hsa-miR-8485	217
hsa-miR-563	22
hsa-miR-425-3p	43
hsa-miR-4474-3p	48
hsa-miR-5002-5p	65
hsa-miR-3672	213
hsa-let-7a-3p	111
hsa-miR-4488	66
hsa-miR-3691-5p	39
hsa-miR-3688-5p	56
hsa-miR-532-3p	208
hsa-miR-4747-5p	215
hsa-miR-6813-3p	65
hsa-miR-3653-3p	113
hsa-miR-3150b-3p	180
hsa-miR-6858-5p	100
hsa-miR-6768-3p	95
hsa-miR-4797-3p	60
hsa-miR-4684-5p	217
hsa-miR-4638-5p	222
hsa-miR-4300	57
hsa-miR-6868-3p	178
hsa-miR-101-5p	55
hsa-miR-4650-3p	97
hsa-miR-6771-3p	179
hsa-miR-6748-3p	63
hsa-miR-1248	153
hsa-miR-4463	63
hsa-miR-6794-5p	119
hsa-miR-4763-5p	68
hsa-miR-6132	79
hsa-miR-1306-3p	11
hsa-miR-579-5p	1
hsa-miR-3917	1
hsa-miR-4793-3p	293
hsa-miR-575	110
hsa-miR-496	110
hsa-miR-320c	1
hsa-miR-6818-3p	149
hsa-miR-3162-3p	43
hsa-miR-661	328

miRNA	**Total experimentally verified targets**
hsa-miR-6797-5p	193
hsa-miR-5705	10
hsa-miR-6767-3p	40
hsa-miR-4722-5p	1
hsa-miR-4477a	134
hsa-miR-375	462
hsa-miR-424-3p	29
hsa-miR-499b-3p	55
hsa-miR-1277-5p	618
hsa-miR-1238-5p	51
hsa-miR-4799-5p	89
hsa-miR-3074-5p	94
hsa-miR-95-5p	116
hsa-miR-7157-5p	148
hsa-miR-4436a	1
hsa-miR-4634	33
hsa-miR-664b-5p	15
hsa-miR-4525	114
hsa-miR-486-3p	133
hsa-miR-302c-5p	136
hsa-miR-449a	117
hsa-miR-5001-5p	67
hsa-miR-4765	43
hsa-miR-3619-5p	117
hsa-miR-7113-5p	103
hsa-miR-766-3p	469
hsa-miR-487b-3p	16
hsa-miR-330-3p	130
hsa-miR-4257	1
hsa-miR-4264	78
hsa-miR-489-5p	1
hsa-miR-92b-5p	47
hsa-miR-6829-3p	1
hsa-miR-6766-3p	36
hsa-miR-5584-5p	122
hsa-miR-4710	1
hsa-miR-3910	102
hsa-miR-885-5p	60
hsa-miR-302e	417
hsa-miR-4715-3p	101
hsa-miR-1282	18
hsa-miR-3124-5p	3
hsa-miR-4750-5p	8

miRNA	**Total experimentally verified targets**
hsa-miR-191-3p	16
hsa-miR-6841-3p	46
hsa-miR-1301-5p	31
hsa-miR-3928-3p	1
hsa-miR-6799-5p	493
hsa-miR-20a-3p	72
hsa-miR-6787-3p	212
hsa-miR-1915-3p	184
hsa-miR-603	497
hsa-miR-32-5p	489
hsa-miR-5706	55
hsa-miR-6778-5p	127
hsa-miR-1301-3p	190
hsa-miR-1273f	207
hsa-miR-4777-5p	45
hsa-miR-3938	62
hsa-miR-371b-3p	42
hsa-miR-370-3p	98
hsa-miR-574-3p	19
hsa-miR-4751	40
hsa-miR-8071	57
hsa-miR-23a-5p	126
hsa-miR-329-5p	87
hsa-miR-372-3p	429
hsa-miR-4479	1
hsa-miR-105-3p	3
hsa-miR-4326	68
hsa-let-7b-5p	1148
hsa-miR-6751-5p	76
hsa-miR-3192-3p	56
hsa-miR-6856-3p	59
hsa-miR-6807-5p	439
hsa-miR-6735-5p	84
hsa-miR-5680	168
hsa-miR-30a-5p	693
hsa-miR-1207-5p	185
hsa-miR-6844	1
hsa-miR-4438	288
hsa-miR-3189-3p	60
hsa-miR-6880-5p	186
hsa-miR-6805-3p	89
hsa-miR-302b-5p	105
hsa-miR-623	221

miRNA	**Total experimentally verified targets**
hsa-miR-875-3p	86
hsa-miR-6869-3p	11
hsa-miR-657	83
hsa-miR-4689	100
hsa-miR-136-3p	43
hsa-miR-519a-3p	316
hsa-miR-5003-3p	128
hsa-miR-576-5p	92
hsa-miR-371a-3p	20
hsa-miR-6762-3p	38
hsa-miR-586	84
hsa-miR-621	25
hsa-miR-6791-3p	272
hsa-miR-4664-5p	64
hsa-miR-4436b-3p	86
hsa-miR-3150a-5p	23
hsa-miR-3692-3p	119
hsa-miR-338-5p	94
hsa-miR-4693-3p	84
hsa-miR-543	97
hsa-miR-151b	25
hsa-miR-211-3p	83
hsa-miR-7150	102
hsa-miR-4445-3p	17
hsa-miR-96-5p	184
hsa-miR-5579-5p	37
hsa-miR-887-3p	10
hsa-miR-593-5p	63
hsa-miR-3689a-5p	45
hsa-miR-618	45
hsa-miR-6504-3p	327
hsa-miR-4282	279
hsa-miR-1228-3p	221
hsa-miR-106a-3p	152
hsa-miR-491-5p	106
hsa-miR-4685-3p	261
hsa-miR-548al	14
hsa-miR-6765-3p	109
hsa-miR-511-3p	237
hsa-miR-4286	103
hsa-miR-1471	3
hsa-miR-6767-5p	3
	26315

9. ACKNOWLEDGEMENTS

First, I would like to thank my advisor Prof. Dr. med. Norbert Gretz, head of Zentrum für Medizinische Forschung (ZMF), University of Heidelberg. His persistent support made my Ph.D. possible. I really appreciate his patience, listening to my ideas; and correcting the draft version of my papers and thesis.

I sincerely thank Ms. Edda Eustachi and Ms. Sina Schwalm for their support throughout my stay at the ZMF. I would also like to thank Dr. Carsten Sticht for support during the work and also for his corrections while while writing my thesis. I thank Dr. Harsh Dweep for his support at the beginning of my work in the lab.

I would like to thank Dr. Abhishek Kumar (DKFZ, Heidelberg) for his support during my thesis and critical corrections. Additionally, I thank Prof. Prashanth Suravajhala (Birla Institute of Scientific Research), Dr. Daraksha Parveen (Microsoft Research, Microsoft Inc.) and Mr. Sandeep Sarde (Wageningen University & Research, Wageningen) for their proof readings.

In the end, I would like to thank my family and friends for always being supportive, primarily my parents, my brother, and my sisters.