Wireless Personal Communications (2024) 138:149–170
https://doi.org/10.1007/s11277-024-11476-7

Experimental Evaluation of Spectrum Handoff Management with Machine Learning Algorithms Using Software Defined Radio

Patan Babjan[1] · **V. Rajendran**[1]

Accepted: 15 July 2024 / Published online: 12 August 2024

Abstract
Although the design of spectrum switching has been studied, little is known about how random user movement affects the handoff. This issue can occur when a user moves to a new location. In this paper, the authors present a framework that verifies the necessity of spectrum handoff to improve the performance of the system by employing machine learning (ML) techniques. Some of these include the Logistic Regression, KNN Algorithm, SVM Algorithm, Naïve Bayes Classifier, Decision Tree Classification and Random Forest Algorithm. The system is implemented on a real-time dataset where all the users are separated in power domain using the concept of non-orthogonal multiple access (NOMA) technique. The dataset values are prepared using a software-defined radio experimental setup, which is used to analyse the performance of various ML techniques in terms of confusion matrix, specificity, precision, F1_score, sensitivity and accuracy. The performance of proposed system is compared with the literature and shown a significant improvement that proves the evidence of our findings.

Keywords Spectrum handoff · Spectrum sensing · Cognitive radio · Machine learning algorithms · Software defined radio · NOMA

1 Introduction

Due to the increasing importance of the radio waves, the use of dynamic techniques has been identified as a way to improve the efficiency of the spectrum management process [1]. This paper aims to introduce the various techniques that can be used to improve the efficiency of the spectrum. When the licensed spectrum isn't being used by the primary users, cognitive radio refers to this situation. The unused spectrum is then allocated to the second-

✉ Patan Babjan
patanbabjan90@gmail.com

[1] Electronics & Communication Engineering, VELS Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, India

ary users or unlicensed users. If the users in these areas resume their activities, the new users will have to vacate the spectrum.

The process of transferring a spectrum from one user to another involves a spectrum handover. This occurs when the latter has started to transmit and the licensed user has to access the channel. The secondary user then moves to an idle channel until the unlicensed user has finished his transmission. The spectrum handoff process can help design an efficient network architecture [2]. It involves carrying out various steps, such as the evaluation and maintenance phases. The evaluation phase of the process is when a device, which is a cognitive type, studies the environment and its specifications in order to determine if it should use the spectrum. After the device has decided that it needs to use the spectrum, it stops the transmission of the data and sends the frequency to a different channel.

Figure 1 shows the concept of spectrum handoff, which is related to the presence or absence of a primary channel and two secondary users [3, 4]. There are four possible scenarios involving the presence of a primary user in a channel. The primary users must immediately stop all communication and leave the spectrum. The secondary users must then look for the appropriate spectrum before they can resume their operations. The PU must also leave the channel completely. The SUs can then switch to the new channel to continue their communication. The allocation of the spectrum to the secondary users based on their priority is determined once the primary user moves to a different channel.

The spectrum handoff is composed of two types: reactive handoff [5] and proactive handoff [6]. The concept of reactive handoff is to plan and implement the operation according to the network's requirements, which can be affected by the prolonged handoff latency and interference. On the other hand, proactive handoff is carried out according to historical data usage. The authors of this paper discuss the advantages of reactive and proactive handoff over spectrum handoff. In one study [7], Guipponi et al. proposed a fuzzy-based method to handle spectrum handoff. In another [8], Wang and colleagues looked into the link maintainability of networks when the SU vacates a place. In another study [9], the authors analyzed the effects of spectrum handoff on the maintainability of a network after the SU moves out of a position. They discovered that the performance of the handoff had not been thoroughly studied. In order to minimize the disruption caused by the handoff, the authors [10] suggested that a voluntary spectrum handoff scheme be introduced.

The authors of [11] looked into the various factors that influence the performance of a spectrum handoff. They then developed a set of metrics that can help analyze the operation's progress. Some of these include the number of handoffs, the switching delay, the non-completion probability, and the link maintainability [12]. The spectrum handoff is mainly used for mobile platforms, such as smart phones [13, 14]. It can be performed efficiently by implementing a hybrid spectrum handoff strategy, which automatically identifies the channel that needs to be handled and provides a quick response. However, it can also cause poor handoff due to the delays in the traffic [15, 16]. The process of PUTPOSH is to determine if a handoff is required based on the arrival of the PU and the service time [17]. The queue model used in this process is called the M/G/1 queue. It shows the status of each task as it progresses. The slowest user will be notified when the idle channel is spotted [13].

Unfortunately, current multiple access techniques are not capable of handling the massive amount of traffic that will inevitably occur in the future. One of the most promising ways to improve the efficiency of future communications is by implementing non-orthogonal multiple access (NOMA) [18]. This technique can be used to provide a vastly increased spec-

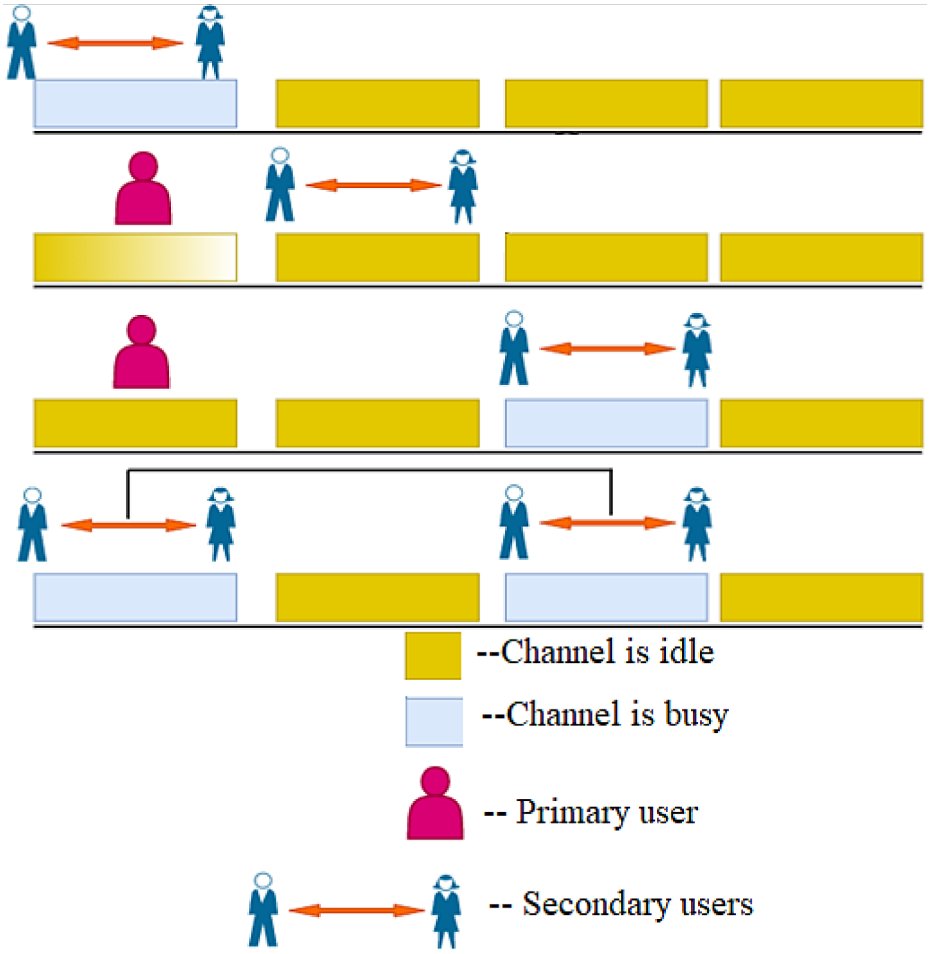

Fig. 1 The phases in Spectrum Handoff - The communication started between two SUs on free band 1; PU appeared, the SUs are searching for a new free band; SUs shifted to free band 3; PU vacated the band 1, so the SUs started communication between band 1 and band 3

tral efficiency. In NOMA, a multi-user signal is multiplexed using a superposition coding technique in the transmitter. The users are then sent using different power levels depending on their channel conditions. This method ensures that the users with the most problematic channel conditions are given the highest power allotment. Having the correct channel state information in the transmitter is very important in order to improve the efficiency of the system.

The users are then sent using different power levels depending on their channel conditions. The stronger the channel, the faster the user can retrieve its signal. On the other hand, the weaker one, which is usually recognized as interference, performs a series of interference cancellation techniques to regain its original signal. In most conventional multiple access systems, the goal is to maintain the symmetry between the channels by using guard periods. However, this method can be very inefficient due to the interference caused by the

guard period. In NOMA, the signals are sent using different power levels and the guard period is completely removed. Despite the advantages of NOMA, it is still very challenging to maintain the level of user fairness due to the complexity of the receiver and the need for a perfect channel state information. In order to solve these issues, we have used machine learning techniques to improve the efficiency of the system.

2 Machine Learning Algorithms

Figure 2 shows a basic machine learning work flow, that starts with the training data and then the labels [19, 20]. These are used by the algorithm to distinguish between the different types of data. The machine learning algorithm block contains various labels and features that are required for training. After completing the training phase, the block generates a predictive modeling model. This model is then used to predict the future state of the data. The data collected during the prediction phase is then analyzed and transformed into features that are used in the model block's final output.

Classification and regression algorithms are two types of supervised machine learning techniques that are commonly used in the prediction phase [21]. In the former, we can predict the continuous values, while in the latter, we need to predict the categorical values. The classification algorithm is a method that identifies the new observations that are coming from the training data. A program can classify new observations by learning from the given data set and then grouping them into various classes or groups. For instance, if there are no cats or dogs in the dataset, then there are no classes or labels for "yes" or "no" Spam.

2.1 Logic Regression

Although the terms linear and logistic regression are similar, they are not used in the same way [22]. For instance, in linear regression, the goal is to find the optimal solution to a given problem. In logistic regression, the goal is to find the classification challenges that are related to the given function. Since p is an unbounded function, we need to first compute

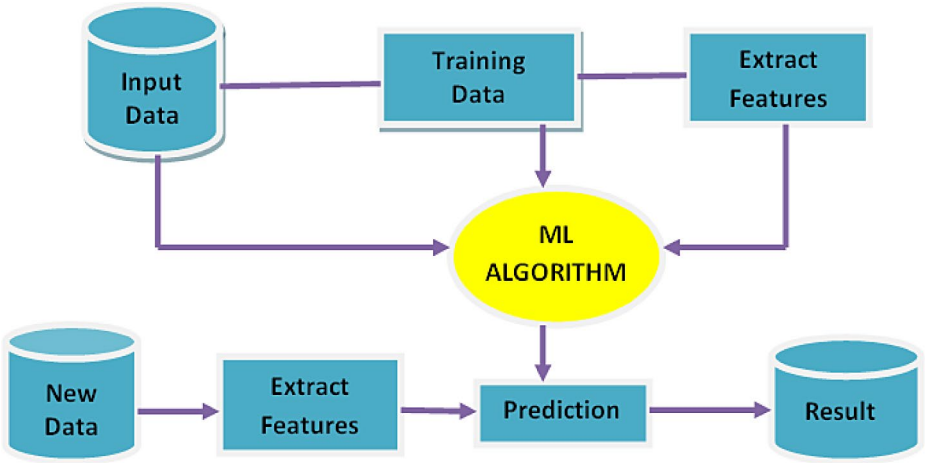

Fig. 2 Machine Learning workflow with training and prediction process

its logit transformation in order to make it a linear one. This is done by taking into account the log $p(x)/(1- p(x)$.

$$log\frac{p\,(x)}{1-p\,(x)} = \beta_{\,0} + \beta\,.x \tag{1}$$

After solving for $p(x)$:

$$p\,(x) = \frac{e^{\beta_{\,0}+\beta}}{e^{\beta_{\,0}+\beta}\,+1} \tag{2}$$

To convert a linear classification into a statistical model, we can select a threshold, such as 0.5. The likelihood of the predicted class is computed by taking into account the training data points x and y.

2.2 KNN Algorithm

The concept of finding the nearest neighbor is a process that involves identifying the points within a given data set that are closest to the source. The algorithm uses a combination of tests and majority votes to find the most appropriate cases. Before implementing KNN, the first step is to transform the data into its vector values [23]. This process takes into consideration the distance between the points and the test data, and it predicts if these are similar. The classification of points is based on the probability that they share the same points. The distance function can be utilized to determine the Minkowski, Hamming, or the Euclidean distance [24]. The distance between two points is computed using the formula known as the "Euclidean Distance".

$$\mathrm{D}((x1, y1)(x2y2)) = \sqrt{(\mathrm{x}_2 - \mathrm{x}_1)^2 + (\mathrm{y}_2 - \mathrm{y}_1)^2} \tag{3}$$

For each value of K, the algorithm finds the nearest neighbors of the data point. It then passes the class to the data point with the highest number of points out of all the other classes of the same K neighbors. After calculating the distance, the algorithm returns the class with the highest probability.

$$\mathrm{P}(Y = \mathrm{i}\,|\mathrm{X} = \mathrm{x}) = \frac{1}{k}\sum\nolimits_{\mathrm{j}\in\mathrm{A}}\mathrm{I}(\mathrm{y}^{(\mathrm{j})} = \mathrm{i}) \tag{4}$$

2.3 SVM Algorithm

One of the most widely used machine learning algorithms is SVM [25], which is used for both regression and classification. In this thread, we will talk about the classification task. It is typically used for medium and small data sets. The main objective of this algorithm is to find the optimal hyperplane that can efficiently separate the data points in two components. We have a set of training examples that are linearly separable. Each example has its own

labels that denote either $y{=}{+}1$ *or* $y{=}0$. We then create a form that describes the training data,

$$\{x_i, y_j\};\ where\, i = 1, \ldots\ldots, N, yi \in \{0, 1\}, x \in R^D \tag{5}$$

As the data points can be separated linearly, we assume D=2 to keep the explanation straightforward.

The objective of the SVM is to orient this hyperplane as far away from the nearest member of both classes as possible. Support vector examples are closest to the ideal hyperplane. From Fig. 3, we can see that two hyperplanes, H_1 and H_2, respectively, travel via support vectors of the +1 and 0 classes. So mx+c=0: H_1; mx+c=1: H_2.

Additionally, the distances between the H1 hyperplane and the origin are (0-*c*)/|*m*| and (1-*c*)/|*m*|, respectively. Margin can so be provided as.

$$\left.\begin{array}{c} \mathrm{M} = (1 - c)/\,|\mathrm{m}| - (0 - c)/\,|\mathrm{m}| \\ \mathrm{M} = 2/\,|\mathrm{m}| \end{array}\right\} \tag{6}$$

where *M* is the margin twice. Margin can thus be expressed as 1/|*m*|. The SVM objective is reduced to the fact of maximising the term 1/|*m*| because the ideal hyperplane maximizesthe margin.

2.4 Naïve Bayes Classifier Algorithm

The Nave Bayes classifier is a perfect tool for developing fast machine learning models [26]. It can predict an object's probability based on its condition. This is referred to as a probabilistic classification, and it is also known as Bayes' theorem, which is a type of rule that states that a hypothesis has a probability of being true. The Bayes' theorem can be written as.

$$P(\,A/B) = \frac{\mathrm{P}\,(\mathrm{B}/\mathrm{A})\ \mathrm{P}\,(\mathrm{A})}{\mathrm{P}\,(\mathrm{B})} \tag{7}$$

The concept of the posterior probability P(A|B), is that a given hypothesis will most likely come true. It is computed by taking into account the likelihood that the correct hypothesis

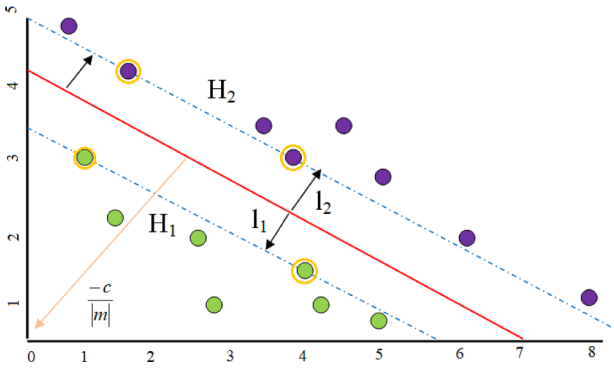

Fig. 3 SVM classification with hyperplanes

will be presented. On the other hand, the likelihood probability is based on the available evidence. The priority probability P(A), is the likelihood of a theory being presented before the evidence P(B) is examined.

2.5 Decision Tree Classification

The goal of the decision tree algorithm is to predict the class of a given dataset [27]. It starts by comparing the values of the record's value and the root attribute's value. The process continues until the tree's leaf node. When implementing a decision tree, there is one major issue that needs to be resolved: which attribute should be selected for the root node and sub-nodes. There are two methods that are commonly used to select the appropriate attribute: the Gini Index and the Information Gain.

2.5.1 Information Gain

The information gain metric is used to measure the changes in the entropy after a feature segmentation has been carried out. It takes into account the amount of information that a feature provides us about a given class. A decision tree algorithm tries to maximize the information gain (in Eq. (8)) by implementing a method that splits the nodes and attributes into different categories.

$$\text{Gain}\,(\text{H,A}) = \text{Entropy}\,(H) - \sum_{\text{Values}(A)} \frac{|H_v|}{|H|}.\text{Entropy}\,(H_v) \tag{8}$$

Where H_v is a subset of set H and has the same value as attribute v, and where the range of attribute A is (A).

Entropy always has a value between 0 and 1. When it equals 0, it is superior to when it equals 1, and when it equals 1, it is inferior. This is how the entropy is calculated:

$$\text{Entropy}\ \ (H) = \sum_{x=1}^{n} P_x \log_2^x \tag{9}$$

P_x is the ratio between the n-th attribute value and the sample size of the subset.

2.5.2 GINI Index

The GINI index is a representation of the purity of a class when it has split into a specific attribute. The better the split, the higher the sets' purity. If there are multiple class labels in a dataset D, the index is calculated as follows.

$$Gini\,(L) = 1 - \sum_{i=1}^{k} P_i^2 \tag{10}$$

If the data are divided into two subsets, D_1 and D_2, of sizes S_1 and S_2, then the relative frequency p_i of the i^{th} class can be calculated. The Gini is then described as.

$$\text{Gini}_{\text{A}}(D) = \frac{S_1}{S}\text{Gini}(D_1) + \frac{S_2}{S}\text{Gini}(D_2) \tag{11}$$

Impurity reduction is computed as.

$$\Delta\,\text{Gini}(A) = \text{Gini}(D) + \text{Gini}_{\text{A}}(D) \tag{12}$$

2.6 Random Forest Algorithm

The Random Forest technique can perform both regression and classification tasks. It can be used with various decision trees, such as Aggregation and the Bootstrap framework [28]. Instead of relying on individual trees, the goal of the Random Forest technique is to combine several decision trees into a final output. This is done through the use of multiple learning models. One of these is called the Random Forest Base Model. In this part, we perform feature sampling and row sampling from the collected data.

3 Proposed System Model

The proposed system's workflow is shown in Fig. 4. The operation of each block is described in this section. For our system, it is assumed that there are two base stations, namely, BTS1 and BTS2, which have coverage areas of CVG1 and CVG2 as shown in Fig. 5, respectively. Also, there are various users who are maintaining varying distances from the two base stations. In order to maximize the efficiency of NOMA's spectrum usage, we have incorporated it into an existing system that separates the users according to their power domain. This method is performed by taking into account the users' distance from the base station and power requirement as independent variables and requirement of handoff ('0' or '1') as dependent variable. The resulting dataset is then generated using software defined radio.

In Fig. 5, the moving SU is crossing the boundary of the CVG1. Due to the presence of a PU, the current being used by the SU is being sent to another channel. Through a cooperative spectrum sensing system, the two SUs maintain the same frequency. The user's red line crosses the system's frequency when it senses the change. This model considers the two conditions of the spectrum. This model can be easily extended to other spectrum channels. In order to solve this issue, we have developed a novel method that uses machine learning techniques to manage the handoff between the two SUs. ML is a widely used computational intelligence tool that can be used in various fields. The ML model can predict the handoff and idle point of the two SUs based on the data collected by their neighbours. It can also find a new spectrum band whenever the environment changes. To improve its performance, a cooperative sensing mechanism has been implemented.

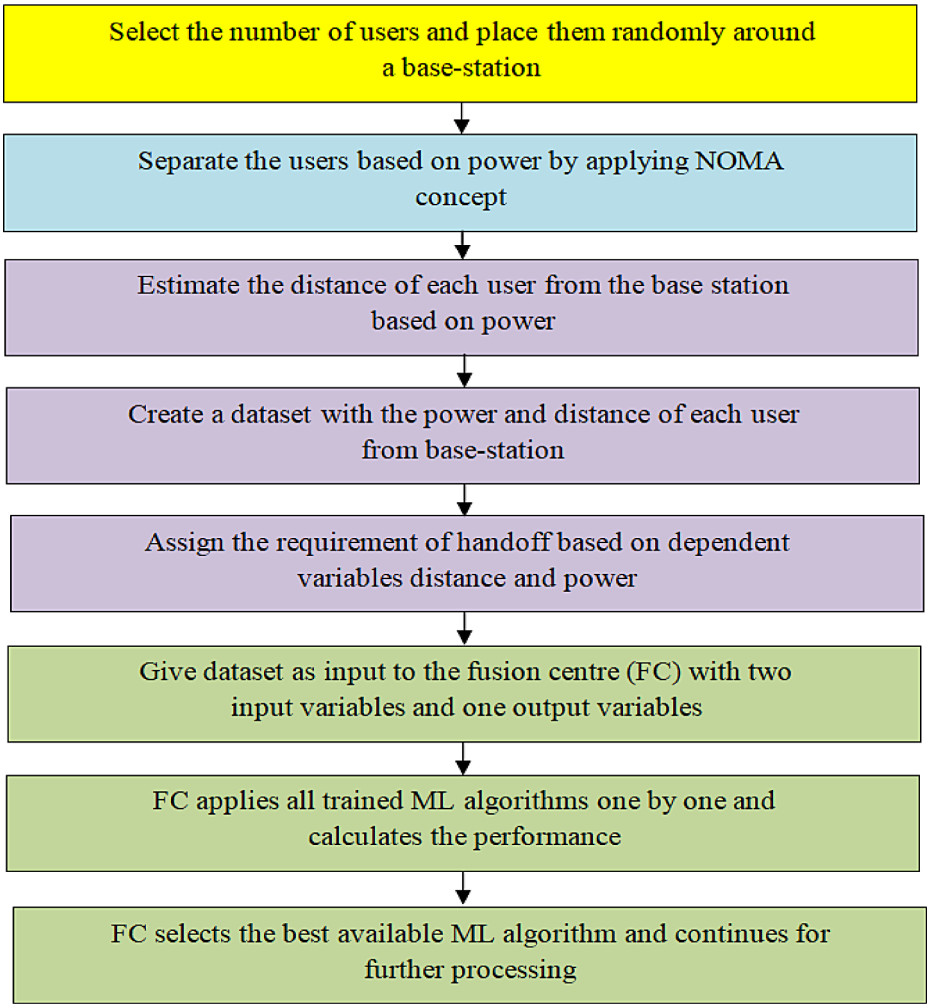

Fig. 4 Workflow of the proposed system

4 Experimental Setup-Software Defined Radio (SDR)

The data set is created using the help of software defined radios [29], which is used to improve the interoperability of commercial radio systems. The experimental setup shown in Fig. 6 is composed of two universal software radio peripheral (USRPs) and a couple of computers which are installed with GNU Radio software. The goal of this study was to analyze the performance of different ML algorithms using the USRP N210 hardware and GNU Radio software. These innovations help to reduce the cost of developing and deploying commercial radio systems. One of the most popular platforms that supports software defined radios is the USRP from Ettus Research. This is used in education and wireless networks.

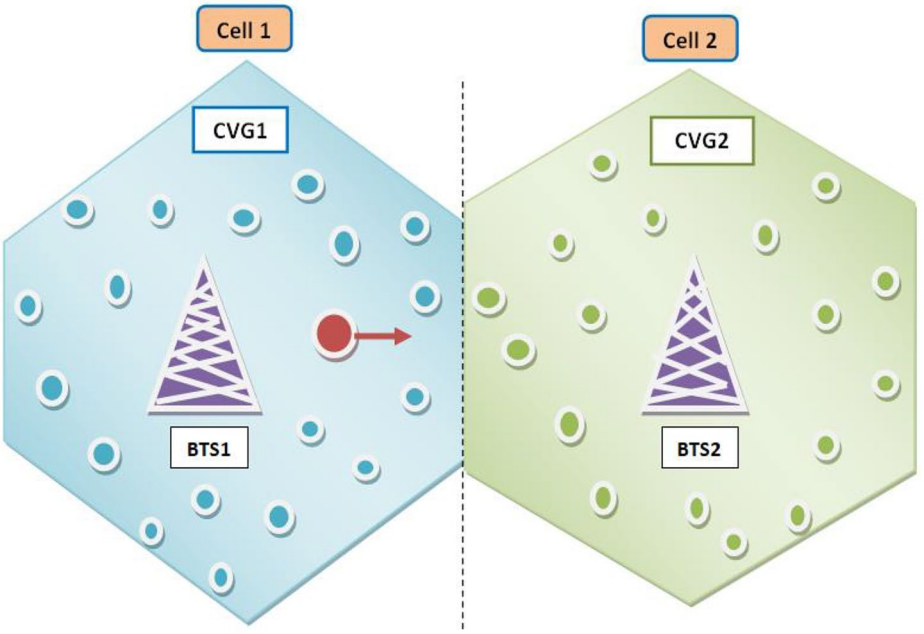

Fig. 5 Spectrum handoff with two cells

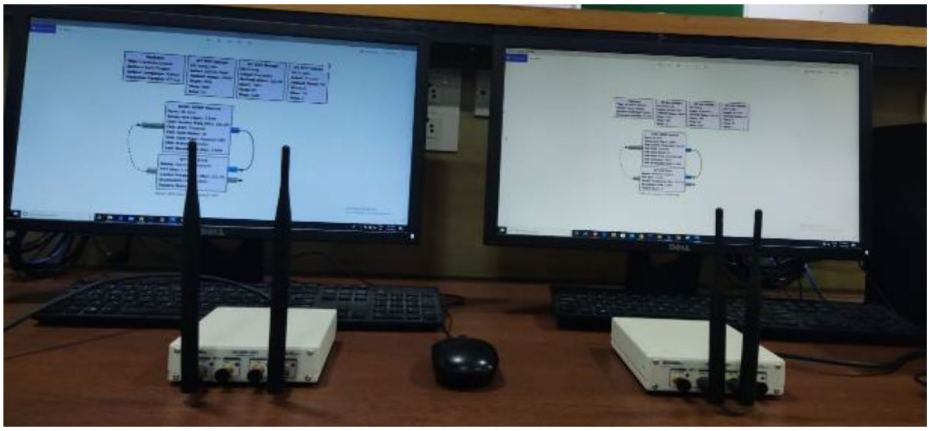

Fig. 6 Experimental setup

The Ettus N210 and N200 series of SDR kits are designed for various applications, such as those that can operate in the DC to 6 GHz range. The daughter-board's RF capabilities are determined by the installed platform. The other components used during the implementation were the mixed-signal daughter-boards from the XCVR2450 and SBX. The USRP features a wide variety of features, such as an integrated digital-to-digital converter. This can be used for various signal processing functions, such as up-sampling and down-sampling. It can also communicate with a host computer. The host computer can access the USRP through the

driver provided by Ettus. The software used to operate it is known as GNU Radio, and it has its own oscillator and timing. Through the help of the host computer and the software, the various parameters of the USRP are calculated and noted.

5 Results and Discussion

In this paper, a dataset is created to decide the necessity of spectrum handoff. Various machine learning classification algorithm are employed to provide the optimal boundary based on the trained data. Figures 7, 8, 9 and 10, show the trained set result and test set result for 100 and 500 number of users when various ML algorithms such as Logistic Regression, KNN Algorithm, SVM Algorithm, Naïve Bayes Classifier, Decision Tree Classification and Random Forest Algorithm. These plots are made by using two independent variables i.e., Distance from the base station on the x-axis and Power of each user on the y-axis. The graph shows two regions: the blue and the yellow. The former is represented by the observations in the blue region, while the latter is represented by the observations in the yellow region. Data points in the graph correspond to the users of the dataset, and the two regions represent the prediction and blue observations.

The blue point observations are for which requirement of spectrum handoff (dependent variable) is probably 0, i.e., users who are under the coverage of base-station 1. The yellow point observations are for which requirement of spectrum handoff is probably 1 means user is not under the coverage of base-station 1. Therefore, it is observed that blue point observations don't require spectrum handoff, when this user crosses the boundary line then it requires spectrum handoff. It is a good model and prediction. However, there are some data points that are in different regions which can be ignored. To minimize this error, we will use the confusion matrix to analyze the data. The classification shown in Fig. 7(a), (b) and 9(a), (b) is a linear model which is used for logistic regression. In the future, we will learn about non-linear classification techniques. In the first example, the boundary shown in the Fig. 7(c), (d) and 9(c), (d) is irregular because it is a K-NN algorithm that finds the nearest neighbor. It has also classified the users according to their categories. For instance, the blue region is for those who don't require the handoff, while the yellow region is for those who do. Although the model is showing good results, there are still some yellow and blue points in the different regions. This is not a big issue since doing this model prevents overfitting. The output of the model shown in Fig. 7(e), (f) and 9(e), (f) is similar to the one shown in the previous example. In the output, the hyperplane has been used to classify the users according to their categories. It has also divided the two classes into the blue and yellow regions.

The Nave Bayes classifier (see Fig. 8(a), (b) and 10(a), (b)) shows that it has a fine boundary and segregated the data points. It is a Gaussian curve, and we have used it in our code. However, there are some errors in the predictions that we have made in Confusion matrix. Despite these, it is still a good classifier. The decision tree classification output shown in Fig. 8(c), (d) and 10(c), (d) is different from the other models. It has both horizontal and vertical lines that are splitting the data according to the Distance and Power variable. This is because the tree is trying to capture all the data. Figure 8(e), (f) and 10(e), (f) (Random Forest Algorithm output) is very much similar to the Decision tree classifier. So, in the Random Forest classifier, we have taken 10 trees that have predicted Yes or NO for the handoff. The classifier took the majority of the predictions and provided the result. We can

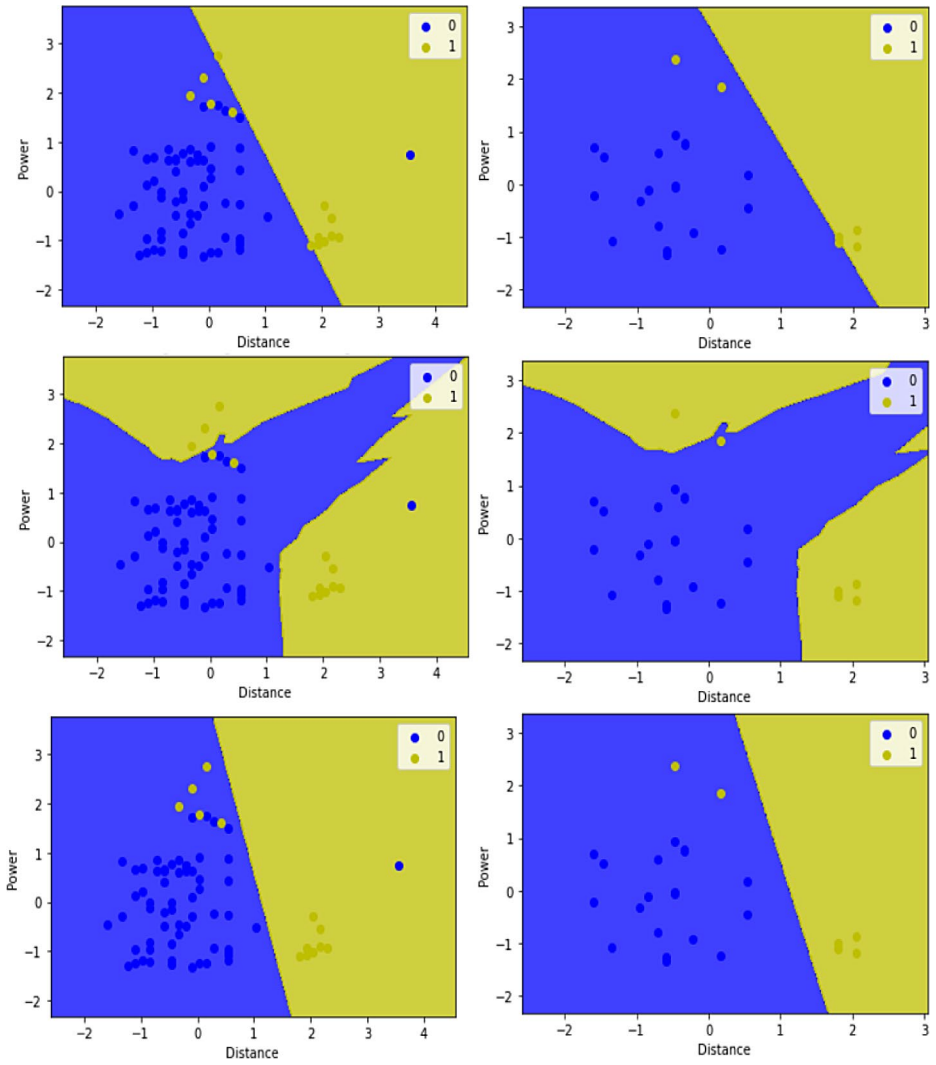

Fig. 7 Visualization (100 number of users) of (**a**) training set result for Logistic Regression (**b**) test set result for Logistic Regression (**c**) training set result for KNN Algorithm (**d**) test set result for KNN Algorithm (**e**) training set result for SVM Algorithm (**f**) test set result for SVM Algorithm

check that there is a minimum number of incorrect predictions without the overfitting issue. We will get different results by changing the number of trees in the classifier.

In the Tables 1 and 2, the predicted output and real test output are given for 100 and 500 number of users respectively. We can clearly see that there are some values in the prediction vector, which are different from the real vector values. These are called prediction errors and are highlighted in the tables for better understanding. If we want to know the number of correct and incorrect predictions, we need to use the confusion matrix. The concept of the confusion matrix is a table that shows the rows that represent the actual classes that the model should have been able to achieve as shown in Figs. 11 and 12. The columns in the matrix

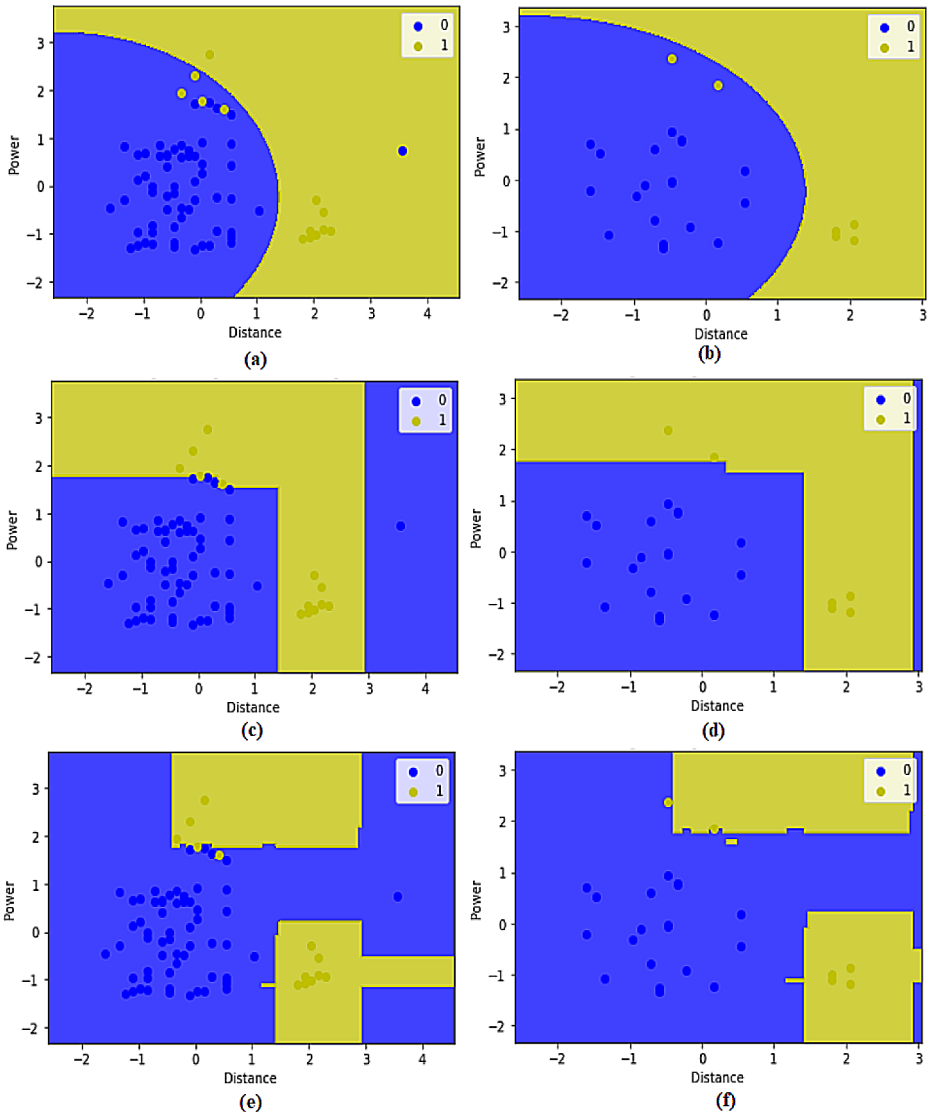

Fig. 8 Visualization (100 number of users) of (**a**) training set result for Naïve Bayes Classifier (**b**) test set result for Naïve Bayes Classifier (**c**) training set result for Decision Tree Classification (**d**) test set result for Decision Tree Classification (**e**) training set result for Random Forest Algorithm (**f**) test set result for Random Forest Algorithm

represent the predictions that the algorithm has made. However, it is also easy to see which ones are wrong. True or False means that the model was correct, while the other means that there was an error or a wrong prediction. With the creation of the Confusion Matrix, we can now measure the quality of the model. In the Fig. 11(a), we can see the confusion matrix, which has $0+3=3$ incorrect predictions and $19+3=22$ correct predictions. The number of

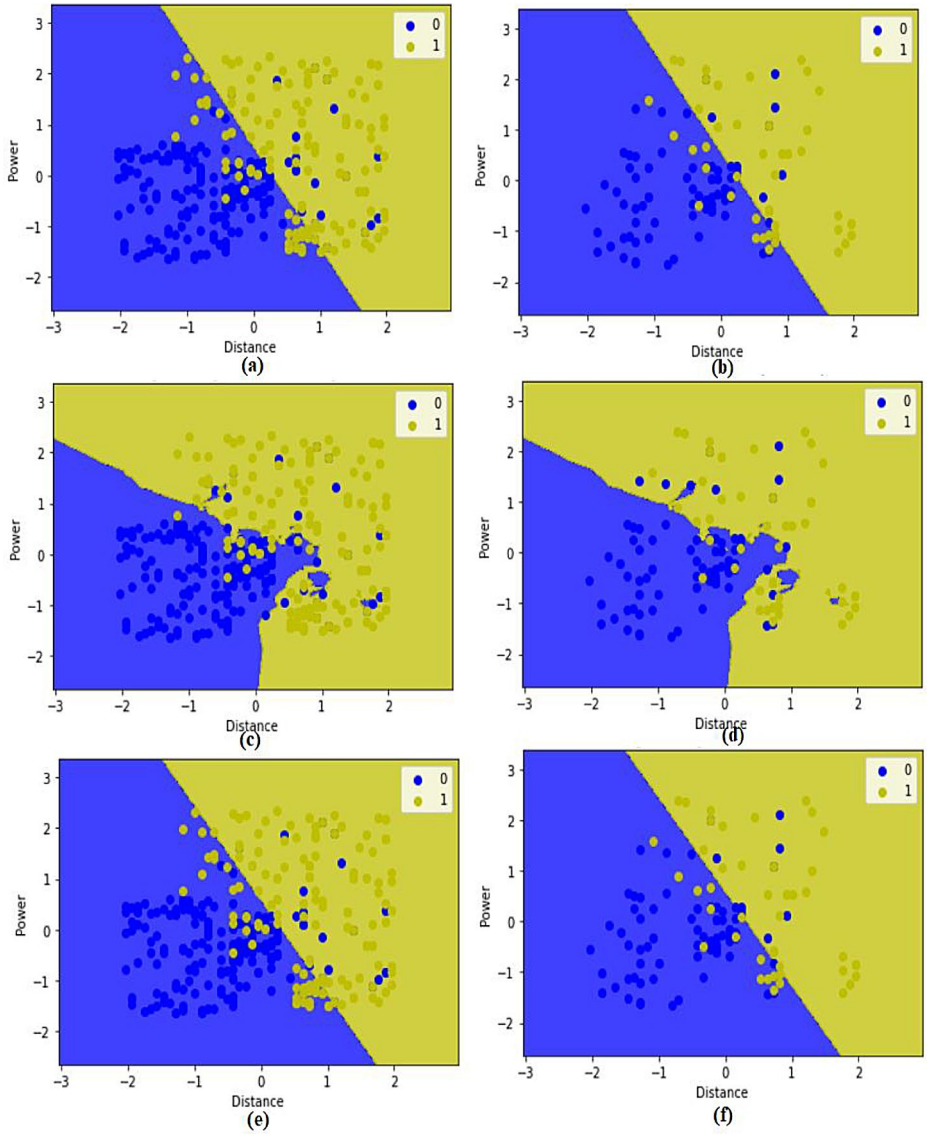

Fig. 9 Visualization (500 number of users) of (**a**) training set result for Logistic Regression (**b**) test set result for Logistic Regression (**c**) training set result for KNN Algorithm (**d**) test set result for KNN Algorithm (**e**) training set result for SVM Algorithm (**f**) test set result for SVM Algorithm

correct and incorrect predictions are generated and shown in Figs. 11 and 12 for 100 and 500 number of users.

The authors of this study analyzed the performance of the ML algorithms on various parameters such as Accuracy, Precision, Sensitivity, Specificity, F1_score, Confusion Matrix by varying the number of users and presented in Tables 3 and 4. The accuracy of a model is measured by how often it is correct. The precision measure is used to evaluate the

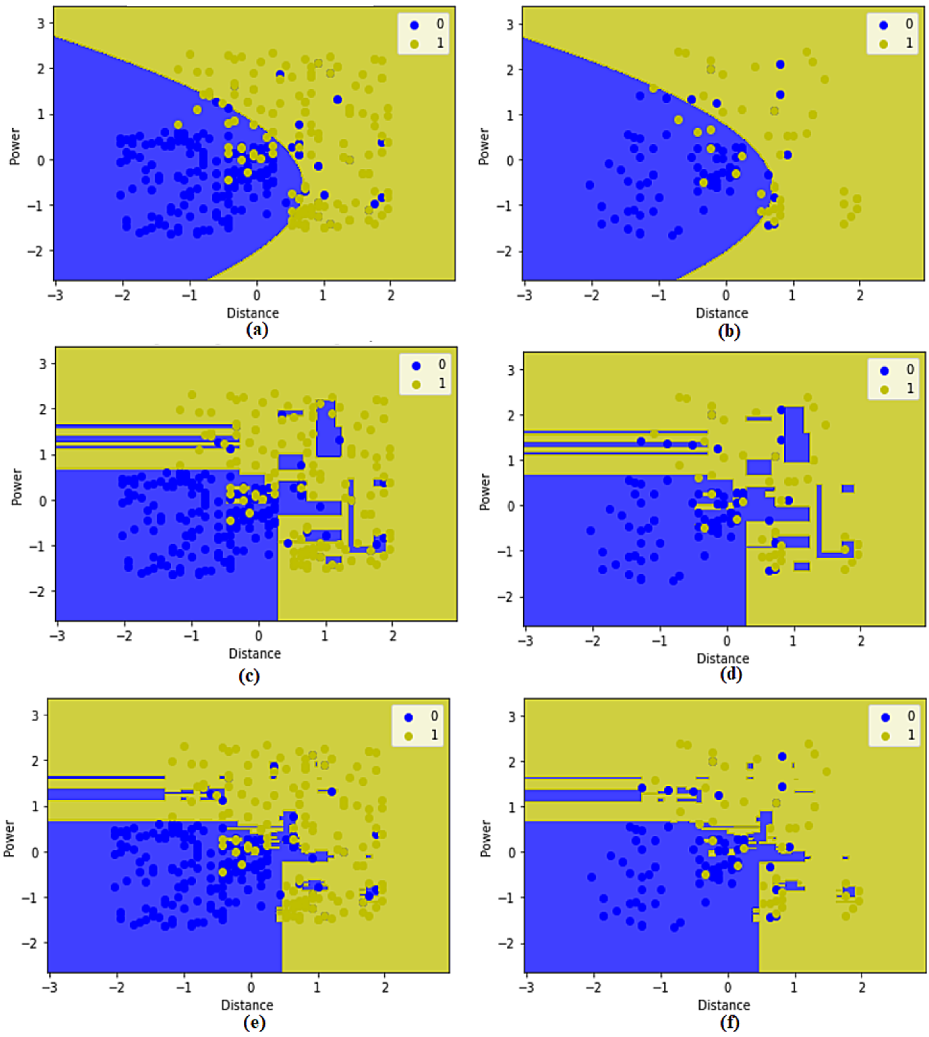

Fig. 10 Visualization (500 number of users) of (**a**) training set result for Naïve Bayes Classifier (**b**) test set result for Naïve Bayes Classifier (**c**) training set result for Decision Tree Classification (**d**) test set result for Decision Tree Classification (**e**) training set result for Random Forest Algorithm (**f**) test set result for Random Forest Algorithm

amount of positive percentage. On the other hand, sensitivity is a measure of how good a model is at predicting false negatives. This is because, if a model is correct about predicting a positive outcome, then it should also consider the true positives. The sensitivity measure is useful in assessing the accuracy of a model when it comes to predicting a positive outcome. Specificity, on the other hand, is a measure of how well a model can predict a negative outcome. It takes into account both the false and positive cases. The F-score is a harmonic measure that takes into account both the sensitivity and precision of a model. However, it does not take into account the True Negative values. It has been observed from the Tables 3 and 4 that the number of test users increases with the number of values, which leads to more

Table 1 Predicted output and real test output of various ML algorithms for 100 number of users (bold items show the highest values compared to remaining)

Logistic Regression	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 **0 0** 1 0 0 0 **0** 0 0 0]
KNN Algorithm	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 **0** 0 0 0]
SVM Algorithm	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 **0** 1 1 0 0 0 **0** 0 0 0]
Naïve Bayes Classifier	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 **0** 1 1 0 0 0 **0** 0 0 0]
Decision Tree Classification	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
Random Forest Algorithm	Test set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0]
	Predict set	[1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 **0** 1 1 0 0 0 1 0 0 0]

Table 2 Predicted output and real test output of various ML algorithms for 500 number of users

Logistic Regression	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[0 1 **1 0 0 0** 0 **0 0** 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 **0 1 0** 0 0 0 1 0 0 1 0 1 **0** 0 **0** 1 0 0 **1** 0 **0** 1 **1** 0 1 0 0 0 1 0 0 **0** 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 **0** 1 0 0 **1** 0 **1 0** 0 0 0 **0 1 1** 1 0 1 0 1 0 0 0 **1** 0 0 1 0 **1** 0 1 **1** 1 0 0 0 **1 1** 0 **0** 1 0 0 0 0 1 0]
KNN Algorithm	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[**1** 1 **1** 0 1 1 0 1 1 0 1 0 0 **1** 1 0 1 0 0 1 1 0 0 0 0 1 1 1 **1** 1 0 0 0 1 0 0 1 0 1 1 0 **0** 1 0 0 **1** 0 1 1 0 **1** 1 0 0 0 1 0 0 1 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 0 **1 0 0** 0 0 0 0 0 **0** 0 0 0 1 **1** 1 0 0 1 0 1 0 **1** 0 **1** 0 0 1 0 **1** 0 **0** 0 1 0 0 0 **1** 0 0 1 1 0 0 0 0 1 0]
SVM Algorithm	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[0 1 **1 0 0 0** 0 **0 0** 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 **0 1 0** 0 0 0 1 0 0 1 0 1 **0** 0 **0** 1 0 0 0 0 **0** 1 **1** 0 1 0 0 0 1 0 0 **0** 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 **0** 1 0 0 0 0 1 **0** 0 0 0 **0 1** 1 1 0 1 0 1 0 0 0 **1** 0 0 1 0 **1** 0 1 **1** 1 0 0 0 **1 1** 0 **0** 1 0 0 0 0 1 0]
Naïve Bayes Classifier	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[0 1 **1** 0 1 1 0 1 **0** 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 **0 1** 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0 **0 1** 0 **1** 1 0 0 0 1 0 0 **0** 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 0 **1 0** 1 0 0 0 0 0 **0** 0 0 0 **0 1** 1 **1** 0 1 0 1 0 **1** 0 **1** 0 0 1 0 0 0 1 0 1 0 0 0 **1** 0 0 **0** 1 0 0 0 0 1 0]
Decision Tree Classification	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[0 1 **1** 0 1 1 **1** 1 1 0 1 0 0 1 **1** 1 0 1 0 0 1 1 0 0 0 0 **0** 1 **0 1** 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 **0 1 0** 1 0 **1** **1** 0 1 0 0 0 0 0 1 1 1 0 1 0 1 0 **1** 0 **1** 0 0 1 0 **1 0** 0 1 0 0 0 **1** 0 0 1 1 0 0 0 0 1 0]
Random Forest Algorithm	Test set	[0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0]
	Predict set	[**1 1 1** 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0 1 1 1 **1** 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 **1** 0 1 0 0 0 1 0 0 1 0 1 0 **0** 1 0 0 0 1 0 0 0 1 0 0 1 1 0 **1 1** 1 1 0 **1** 0 0 **1 0** 0 0 0 **0 1 1** 1 0 1 0 1 **0** 1 0 0 1 0 **1 0 0 1** 1 0 0 0 **1 1** 0 1 1 0 0 0 0 1 0]

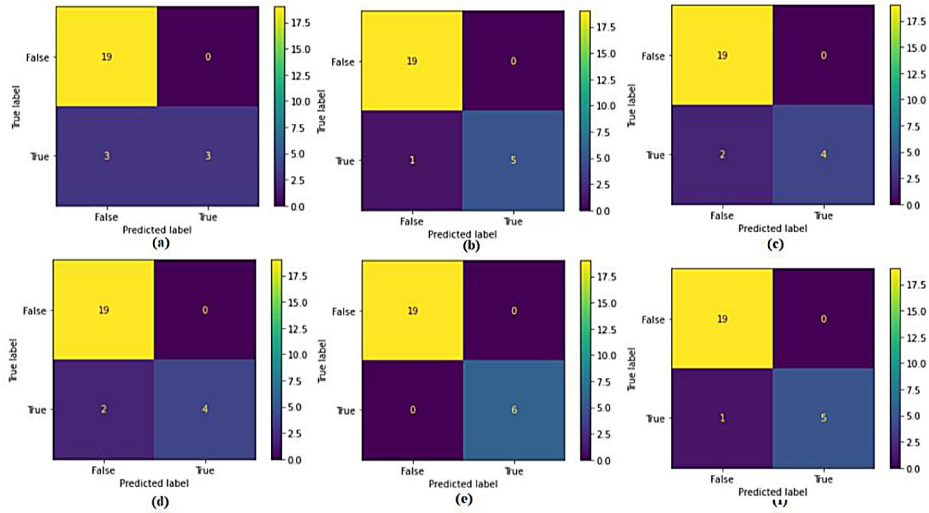

Fig. 11 Confusion matrix of various ML algorithms for 100 number of users (**a**) Logistic Regression (**b**) KNN Algorithm (**c**) SVM Algorithm (**d**) Naïve Bayes Classifier (**e**) Decision Tree Classification (**f**) Random Forest Algorithm

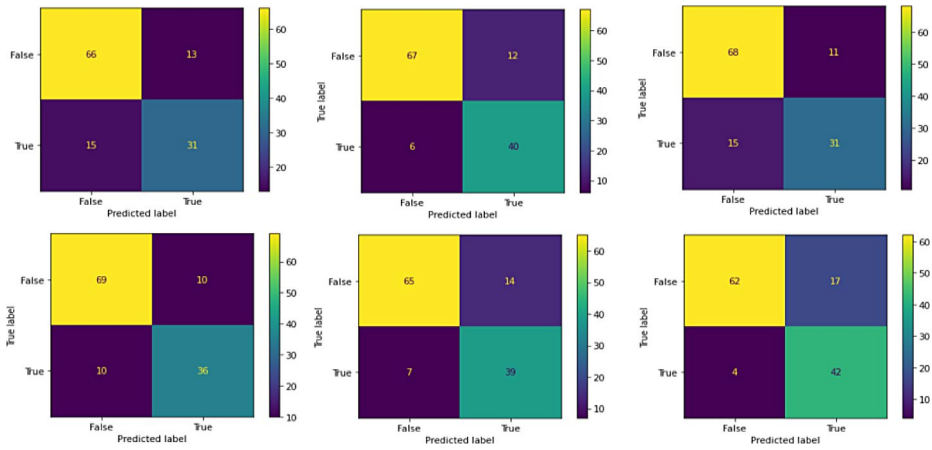

Fig. 12 Confusion matrix of various ML algorithms for 500 number of users (**a**) Logistic Regression (**b**) KNN Algorithm (**c**) SVM Algorithm (**d**) Naïve Bayes Classifier (**e**) Decision Tree Classification (**f**) Random Forest Algorithm

errors. It is also believed that the system will become more efficient by having fewer test users and more trained ones.

5.1 Comparison with the Literature

The Table 5 shows the various performance improvements that the proposed system has made compared to the literature [30–32]. In terms of precision, sensitivity, specificity, and

Table 3 Performance analysis of various ML algorithms for 100 users (bold items show the highest values compared to remaining)

Algorithm	Accuracy	Precision	Sensitivity	Specificity	F1_score
Logistic Regression	0.88	1.0	0.5	1.0	0.666
KNN Algorithm	0.96	1.0	0.833	1.0	0.909
SVM Algorithm	0.92	1.0	0.666	1.0	0.8
Naïve Bayes Classifier	0.92	1.0	0.666	1.0	0.8
Decision Tree Classification	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
Random Forest Algorithm	0.96	1.0	0.833	1.0	0.909

Table 4 Performance analysis of various ML algorithms for 500 users (bold items show the highest values compared to remaining)

Algorithm	Accuracy	Precision	Sensitivity	Specificity	F1_score
Logistic Regression	0.776	0.704	0.673	0.835	0.688
KNN Algorithm	**0.856**	0.769	0.869	0.848	**0.816**
SVM Algorithm	0.792	0.738	0.673	0.860	0.704
Naïve Bayes Classifier	0.84	**0.782**	0.782	**0.873**	0.782
Decision Tree Classification	0.832	0.735	0.847	0.822	0.787
Random Forest Algorithm	0.832	0.711	**0.913**	0.784	0.799

F1_score, the improvements are evaluated according to the ML algorithm. In a study, Purab Nandi et al., [30] analyzed the performance of several popular ML algorithms to predict the right values for users. Compared to Ref. [30], the proposed system performed significantly better in terms of sensitivity, specificity, and accuracy. In addition, it exhibited a remarkable increase in enhancement values of 10.1%, 25%, and 5.49%, respectively. In a study conducted on a cognitive radio network, Geetanjali et al., [31] presented a novel attack that can be used by an intruder to access the spectrum. The proposed system's performance is validated by means of a weighted error probability of 80%. The proposed system's improvement of 15% is compared with the previous model's accuracy of 92%. In order to ensure that the radio waves are distributed across the spectrum, Wajhal Gaurav et al., [32] proposed a machine learning-based prediction method. The performance of this technique is compared with our proposed system and it is observed a good improvement in terms of accuracy, precision and F1_score with a maximum increment of 7.79%, 4.8% and 5.6%, respectively. In some cases, the proposed system performed poorly when compared to the literature. But, in the overall sense, the system has significantly improved, which shows the proof of our findings.

6 Conclusions

The authors of this paper presented a framework that aims to improve the performance of a system by implementing various machine learning techniques for spectrum handoff. These include the Logistic Regression, KNN Algorithm, SVM Algorithm, Naïve Bayes Classifier, Decision Tree Classification and Random Forest Algorithm. The system is implemented on a live dataset that contains all the users in the power domain using the non orthogonal multiple access (NOMA) technique. The data collected from this system is then analyzed using

Table 5 Performance comparison of proposed system with the existing literature

Ref.	Type of Algorithm	Parameter	Value (in %)	Proposed method value (in %)	Performance improvement (in %)
[30]	SVM	Accuracy	94	93	-1.07 ↓
	KNN		89	98	10.1 ↑
	LR		92	90	-2.17 ↓
	NB		88	93	5.68 ↑
	RF		91	97	6.59 ↑
	SVM	Specificity	86	100	16.27 ↑
	KNN		85	100	17.64 ↑
	LR		85	100	17.64 ↑
	NB		80	100	25 ↑
	RF		86	100	16.27 ↑
	SVM	Sensitivity	94	87	-7.44 ↓
	KNN		89	90	1.12 ↑
	LR		92.5	89	-3.78 ↓
	NB		92	93	1.08 ↑
	RF		91	96	5.49 ↑
[31]	SVM	Accuracy	80	92	15 ↑
[32]	DT	Accuracy	77	83	7.79 ↑
	RF		78	83	6.41 ↑
	LR		84	77	-8.33 ↓
	SVM		86	79	-8.13 ↓
	DT	Precision	81	84	3.70 ↑
	RF		83	87	4.81 ↑
	LR		88	85	-3.40 ↓
	SVM		92	95	3.26 ↑
	DT	F1_score	89	91	2.24 ↑
	RF		83	86	3.61 ↑
	LR		87	88	1.14 ↑
	SVM		89	94	5.61 ↑

a software-defined radio experimental setup. The performance of different ML techniques is compared with the accuracy, sensitivity, specificity, and F1_score and confusion matrix of the users. The number of test users has been observed to increase with the number of values. The increasing number of test users can lead to errors and lead to the system becoming more inefficient. This is because the number of trained users and fewer test users will help the system become more efficient. It is also observed that proposed system has shown some poor performance compared to literature in some case, however, in overall our system has shown a significant improvement which shows evidence of our findings.

Acknowledgements Not applicable.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Mr. Patan Babjan and Dr V. Rajendran. The first draft of the manuscript was written by Mr. Patan Babjan and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability No datasets are used in this paper.

Declarations

Ethics Approval and Consent to Participate The authors have followed all Ethics and the authors are responsible for any unethical things happens.

Competing Interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Arjoune, Y., & Kaabouch, N. (2019). A comprehensive survey on spectrum sensing in cognitive radio networks: Recent advances, new challenges, and future research directions. *Sensors19*(1), 126.
2. Aggarwal, Manav, T., Velmurugan, & Nandakumar, S. (2021). Comparative analysis of algorithms governing Spectrum Handoff in Cognitive Radio Networks. *Wireless Personal Communications*, *121*(3), 1423–1435.
3. Yawada, P. S., & Dong, M. T. (2019). Intelligent process of spectrum handoff/mobility in cognitive radio networks. *Journal of Electrical and Computer Engineering*.
4. Srivastava, V. (2023). Innovative Spectrum Handoff Process Using a Machine Learning-Based Metaheuristic Algorithm. *Sensors 23*(4), 20–11.
5. Gouda, A. E., et al. (2018). Reactive spectrum handoff combined with random target channel selection in cognitive radio networks with prioritized secondary users. *Alexandria Engineering Journal*, *57*(4), 3219–3225.
6. Charan, G., & Alrabeiah, M. (2021). Vision-aided 6G wireless communications: Blockage prediction and proactive handoff. *IEEE Transactions on Vehicular Technology*, *70*, 10193–10208.
7. Prasad, R., Krishna, & Jaya, T. Optimal network selection in cognitive radio network using simple additive weighting method with multiple parameters. (2019). *International Conference on Smart Systems and Inventive* Technology *(ICSSIT)*. IEEE
8. Hoque, S., Arif, W., & Sen, D. (2020). Assessment of spectrum handoff performance in cognitive radio cellular networks. *IEEE Wireless Communications Letters*, *9*(9), 1403–1407.
9. Nandakumar, S., et al. (2019). Efficient spectrum management techniques for cognitive radio networks for proximity service. *Ieee Access: Practical Innovations, Open Solutions*, *7*, 43795–43805.
10. Nandakumar, S., Sai Bharadwaj, G. V. S., & Srivastava, D. (2019). Efficient spectrum handoff using hybrid priority queuing model in cognitive radio networks. *Wireless Personal Communications*, *108*, 203–212.
11. Devi, M., Kalpana, & Umamaheswari, K. (2021). Optimization techniques for spectrum handoff in cognitive radio networks using cluster based cooperative spectrum sensing. *Wireless Networks*, *27*, 2173–2192.
12. Mishra, M. P., & Vidyarthi, D. P. (2019). Spectrum Handoff in Cognitive Radio Cellular Network: A Review. *8th International Conference System Modeling and Advancement in Research Trends (SMART)*. IEEE, 2019.
13. Shekhar, S., Hoque, S., & Arif, W. (2020). Analysis of spectrum handoff delay using finite queuing model in cognitive radio networks. *International Journal of Communication Networks and Distributed Systems*, *25*(3), 249–264.
14. Haldorai, A., & Kandaswamy, U. (2019). *Intelligent spectrum handovers in cognitive radio networks*. EAI/Springer Innovations in Communication and Computing. Springer.
15. Preetha, K. S., & Kalaivani, S. (2020). Analysis of spectrum handoff schemes for cognitive radio networks considering secondary user mobility. *International Journal of Grid and Utility Computing*, *11*(4), 443–456.
16. Yawada, P. S. and Dong, M. T. (2019). Intelligent process of spectrum handoff/mobility in cognitive radio networks. *Journal of Electrical and Computer Engineering*, 1–12.
17. Arshid K., Hussain, I., Bashir, M. K., Naseem, S., Ditta, A., Mian, N. et al. (2020). Primary user traffic pattern based opportunistic spectrum handoff in cognitive radio networks. *Applied Science*, *10*(5), 3–19.

18. Reddy, B. S. (2021). Experimental validation of non-orthogonal multiple access (NOMA) technique using software defined radio. *Wireless Personal Communications*, *116*(4), 3599–3612.
19. Ahmed, R., et al. (2022). Hybrid machine-learning-based spectrum sensing and allocation with adaptive congestion-aware modeling in CR-assisted IoV networks. *IEEE Internet of Things Journal*, *9*, 25100–25116.
20. Ahmad, H. B. (2019). Ensemble classifier based spectrum sensing in cognitive radio networks. *Wireless Communications and Mobile Computing* (2019).
21. Sen, P., Chandra (2020).,supervised classification algorithms in machine learning: A survey and review. *Emerging technology in modelling and graphics* (pp. 99–111). Springer.
22. Wood, S. N. (2017). *Generalized additive models: An introduction with R*. CRC.
23. Wang, B., et al. (2020). A novel weighted KNN algorithm based on RSS similarity and position distance for Wi-Fi fingerprint positioning. *Ieee Access: Practical Innovations, Open Solutions*, *8*, 30591–30602.
24. Du, W. S. (2018). Minkowski-type distance measures for generalized orthopair fuzzy sets. *International Journal of Intelligent Systems*, *33*(4), 802–817.
25. Tharwat, A. (2019). Parameter investigation of support vector machine classifier with kernel functions. *Knowledge and Information Systems*, *61*(3), 1269–1302.
26. Chen, S., et al. (2020). A novel selective naïve Bayes algorithm. *Knowledge-Based Systems*, *192*, 105361.
27. Fletcher, S., & Md Zahidul, I. (2019). Decision tree classification with differential privacy: A survey. *ACM Computing Surveys (CSUR)*, *52*(4), 1–33.
28. Schonlau, M., & Rosie Yuyan Zou. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, *20*(1), 3–29.
29. Reddy, B. S. (2018). Experimental validation of timing, frequency and phase correction of received signals using Software defined Radio Testbed. *Wireless Personal Communications*, *101*(4), 2085–2103.
30. Nandi, P.K.R., Anupama, Agarwal, H., Patel, K., Bang, Vedant, Bharat, Manan, Guru, & Madhen (2023). Analysis of ML Algorithm for geriatric fall detection due to the effects of various user characteristics. https://doi.org/10.20944/preprints202305.0917.v1.
31. Rathee, G., et al. (2020). A secure spectrum handoff mechanism in cognitive radio networks. *IEEE Transactions on Cognitive Communications and Networking*, *6*(3), 959–969.
32. Wajhal, G. (2021). Proactive handoff of secondary user in cognitive radio network using machine learning techniques. *Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020*. Springer Singapore.

Patan Babjan is research scholar in Electronics & Communication Engineering from VELS Institute of Science, Technology & Advanced Studies (VISTAS) in Chennai. He completed his M.Tech in E.C.E (VLSI & Embedded Systems) from AZAD College of Engineering and Technology, Moinabad (Hyderabad) & completed his B.Tech in E.C.E from Sri Sai Institute of Technology and Science, Rayachoty (Kadapa). His research interests include Cognitive Radio Networks, Software Defined Radio, Wireless Communication Networking and Digital Electronics & Microprocessors.

V. Rajendran received his M.Tech in Physical Engineering from Indian Institute of Science, Bangalore, India and received his PhD degree on Electrical and Electronics Engineering from Chiba University, Japan in 1981 and 1993, respectively. He is currently, a professor and director of the department of Electronics and Communication Engineering in VELS Institute of Science, Technology & Advanced Studies (VISTAS), Pallavaram, Chennai, India. He was awarded MONBUSHO Fellowship, Japanese Govt. Fellowship (1988–1989) through the Ministry of Human Resource and Development, Govt. of India. He was elected twice as Vice Chairman– Asia of Execution Board of Data Buoy Co-operation Panel (DBCP) of Inter-Governmental Oceanographic Commission (IOC)/World Meteorological Organization (WMO) of UNSCO, in October 2008 and September 2009, respectively. He was a Life fellow of Ultrasonic Society of India, India (USI) in January 2001. He was a Life fellow of Institution of Electronics and Telecommunication Engineering (IETE), India, in January 2012. His area of interest includes cognitive radio and software-defined radio communication, antennas and propagation and wireless communication, under water acoustic signal processing and under water wireless networks. He has published 84 papers in web of science and Scopus-indexed journal.