A Project Report On

Spectrum Sensing in Cognitive Radio using Machine Learning

Submitted in partial fulfilment for the
degree of Bachelor of Technology in
Electronics and Telecommunications Engineering

Submitted by
Arjun Satheesh
Gautam Govindaraju
Prajakta Gurav
Priyanshu Bhagat

Under the guidance of
Dr.(Mrs.) Ranjan Bala Jain

Vivekanand Education Society's Institute of Technology
Hashu Adwani Memorial Complex,
Collector's Colony, Chembur, Mumbai, Maharashtra 400074

2022-2023

BE PROJECT APPROVAL

Project entitled **Spectrum Sensing in Cognitive Radio using Machine Learning** by **Arjun Satheesh, Gautam Govindaraju, Prajakta Gurav** and **Priyanshu Bhagat** is approved for the degree of Bachelor of Engineering.

Examiners:

1. _____

2. _____

Supervisor:
Dr.(Mrs.) Ranjan Bala Jain

Date:

Place:

CERTIFICATE

This is to certify that Arjun Satheesh, Gautam Govindaraju, Prajakta Gurav and Priyanshu Bhagat has completed the project report on the topic "Spectrum Sensing in Cognitive Radio using Machine Learning" satisfactorily in partial fulfilment for the Bachelor's Degree in Electronics and Telecommunication Engineering under the guidance of Dr.(Mrs.) Ranjan Bala Jain during the year 2022-2023 as prescribed by Mumbai University.

Guide

Dr.(Mrs.) Ranjan Bala Jain

Head Of Department

Dr. Chandansingh Rawat

Principal
J.M.Nair

Examiner 1

Examiner 2

DECLARATION

We declare that this written submission represents our ideas in our words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Arjun Satheesh

Prajakta Gurav

Gautam Govindaraju

Priyanshu Bhagat

Acknowledgement

This project could not have been possible without the participation and assistance of many people whose names may not be enumerated.Their contributions are sincerely appreciated and gratefully acknowledged. However, we would like to express our deep appreciation and indebtedness particularly to our guide, Dr.(Mrs.) Ranjan Bala Jain, whose knowledge and practical experience in the field helped us in better understanding of the subject. We are grateful for her constant support and guidance in the making of this project. We convey our sincere and heartfelt thanks to our H.O.D., Dr. (Mr.) C.D.Rawat and our principal, Dr. (Mrs.) Jayalekshmi M. Nair who have ensured that we carry foreword our work smoothly and help make this a wonderful learning experience. We would also like to thank our department and all the staff members who have supported us throughout our engineering and constantly provided ideas and solutions to the problems faced through the project phase and our engineering. Last but not the least, our thanks to our parents and friends for their endless support.

Arjun Satheesh
(D19A-01)

Gautam Govindaraju
(D19A-13)

Prajakta Gurav
(D19A-14)

Priyanshu Bhagat
(D19A-57)

Abstract

Cognitive radio is a form of wireless communication in which a transceiver can intelligently detect which communication channels are in use and which are not. The frequency bands that have been allocated to a different licensed user are insufficiently utilized. CR is treated as a favorable solution to address the frequency shortage issue. Spectrum sensing is the process of periodically monitoring frequency bands. The aim is to find out whether the primary user is present or absent by sensing the signal of the channel. It also helps in minimizing interference from other users, increases spectrum efficiency and improves the Quality of Service for users. Spectrum sensing can be implemented using Machine learning for optimal usage of channels and bandwidth as it does not require prior knowledge. In order to use available spectrum efficiently, CR is used. In this work, spectrum sensing is performed using three supervised machine learning algorithms namely Decision Tree, Support Vector Machine and K- Nearest Neighbour. The performance is measured in terms of accuracy, Probability of detection and Probability of false alarm and based on these parameters the most suitable algorithm is determined.

Keywords: *Machine learning, Spectrum sensing, K-Nearest Neighbour, Support Vector Machine, Probability of false alarm, Probability of detection, Primary user, Secondary user*

Contents

List of Tables

List of Figures

Nomenclature

ML	Machine Learning
SS	Spectrum Sensing
CR	Cognitive Radio
PU	Primary User
SU	Secondary User
DT	Decision Tree
SVM	Support Vector Machine
K-NN	K-Nearest Neighbour

Chapter 1

Introduction

The rapid increase in technology and the emergence of wireless communication devices such as GPS, Wi-Fi, mobile, Cordless telephones etc, has dramatically increased the demand for radio resources[7]. The utilisation of the frequency spectrum is not uniform. The usage of the frequency spectrum depends on some factors like, time and geographical location. Some of the frequency at that time might be partially occupied but at the same time some other frequency spectrum might be overcrowded. The frequencies that have been unused by the primary users are termed as white spaces or spectrum holes [5, 7]. Spectrum hole is a region in the assigned frequency band to a licensed user, also known as Primary User (PU) which is not being utilised by the PU at a specific time or geographical location. So, to save these frequency bands from getting wasted as RF spectrum is a scarce resource, at some specific time or geographical location, the idea of Cognitive Radio (CR) was first proposed by Dr. Joseph Mitola III in 1988 who is also the founder of CR and Software Defined Radio (SDR)[4]. Dr. Mitola first proposed the concept in a seminar at KTH Royal Institute of Technology and later it was published along with Gerald Q. Maguire Jr. in an article in 1999.

1.1 Functions of Cognitive Radio

CR is defined as an intelligent and aware radio system that can sense its operational environment and dynamically adjusts its radio operating parameters to modify the

system. The functions of CR include spectrum sensing, spectrum management, spectrum sharing and spectrum mobility. Spectrum Sensing(SS) is basically to detect unused portion of spectrum or spectrum holes. Spectrum management captures the best spectrum to meet user communication requirements. Spectrum sharing provides fair spectrum scheduling method among CR user i.e. it distributes spectrum among the scheduled users. Spectrum mobility performs transition of secondary user to a better spectrum which means that the unlicensed user will vacate the channel when licensed user is detected.

1.2 Need of Spectrum Sensing

The aim of spectrum sensing is to determine spectrum availability and the presence of the licensed users. Spectrum sensing detects the spectrum holes accurately. Secondary Users (SU) are those users who are unlicensed users but want to access the spectrum when they are in need. Through the help of CR, SU can be made available to use the spectrum for a specific period of time when the PU is absent this can increase the spectrum utilization without getting it wasted.

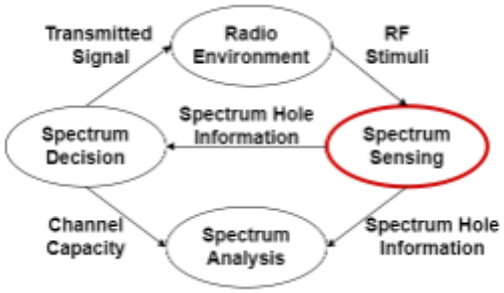

Figure 1.1: Cognitive Radio Cycle

The Figure(1.1) shows the flow of the Cognitive radio cycle. The cognitive radio continuously monitors its Rf environment. Whenever it detects any vacant channel in the spectrum sensing stage it moves to spectrum analysis stage where analysis of the obtained information is being done. Here the obtained information is converted into knowledge based on different classification methods. Next in the spectrum decision stage, based on the knowledge acquired, decisions are made for optimizing the usage of spectrum.

1.3 Spectrum Sensing Techniques

Spectrum sensing in cognitive radio is broadly divided into two categories i.e. spectrum sensing using machine learning models and non-machine learning techniques. Figure(1.2) shows all the spectrum sensing methods.

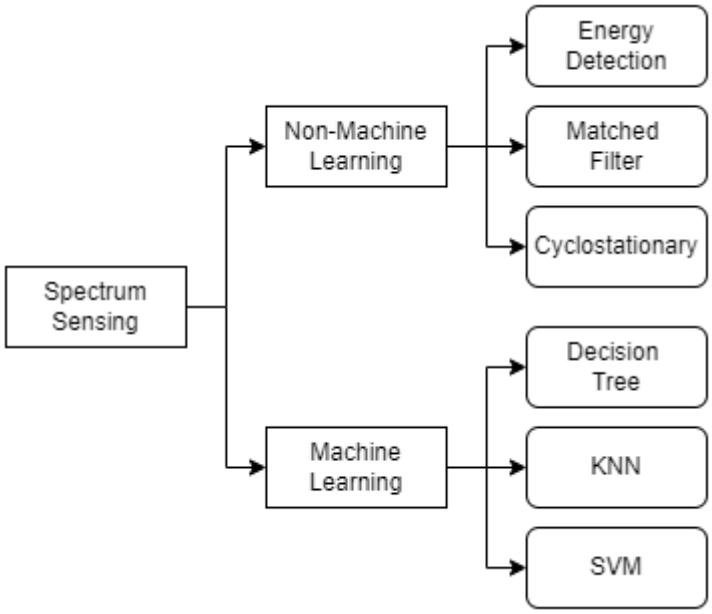

Figure 1.2: Spectrum Sensing Methods

1.3.1 Non-Machine Learning Techniques

There are mainly three methods of spectrum sensing methods for non-machine learning techinques. They are Energy detection, Matched filter and Cyclostationary[6, 13, 9].

Energy Detection

Energy detection is a popular spectrum sensing technique used in cognitive radio networks to detect the presence of a signal in a given frequency band [3]. The basic idea behind energy detection is to measure the energy of the received signal over a period of time and compare it to a predetermined threshold to determine the presence or absence of a signal.

Figure 1.3 shows the flow of Energy Detection,

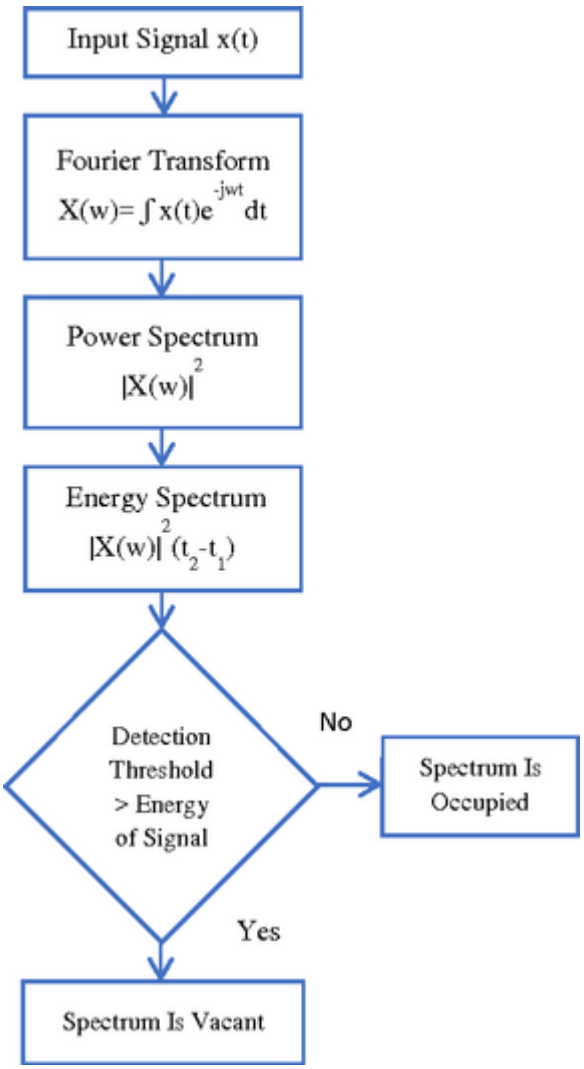

Figure 1.3: Energy Detection

The energy detection algorithm for spectrum sensing can be implemented using the following steps,

1. Select the frequency band: The first step is to select the frequency band of interest, which is usually determined by the application requirements.

2. Set the observation time: The observation time is the time interval over

which the energy of the received signal is measured. It is determined by the duration of the signal of interest and the minimum detectable signal power.

3. Set the detection threshold: The detection threshold is a predetermined value that determines the minimum energy level required to detect the presence of a signal. It is usually determined based on the noise level in the frequency band of interest and the desired false alarm probability.

4. Receive the signal: The next step is to receive the signal in the frequency band of interest using an antenna and a radio receiver.

5. Compute the energy: The energy of the received signal is then computed over the observation time by squaring the amplitude of the signal and integrating it over time.

6. Compare with the threshold: The computed energy is then compared with the detection threshold. If the energy exceeds the threshold, it is concluded that a signal is present in the frequency band of interest. Otherwise, it is concluded that the frequency band is idle.

The energy detection technique for spectrum sensing involves measuring the energy or power of the received signal and comparing it to a predetermined threshold to determine the presence or absence of a signal of interest in the spectrum[6].Energy detection is a simple and easy-to-implement spectrum sensing technique that does not require prior knowledge of the signal of interest. However, it has some limitations, such as susceptibility to noise and interference and sensitivity to the detection threshold. Moreover, it may not work well in the presence of frequency-selective fading or other types of interference. Therefore, other spectrum sensing techniques, such as matched filter sensing, may be used in conjunction with energy detection to improve the overall performance of spectrum sensing in cognitive radio networks.

Matched Filter

Matched filter is a commonly used signal processing technique for spectrum sensing, which involves detecting the presence of a signal in a given frequency band.The matched filter technique for spectrum sensing involves correlating an incoming signal with a known reference signal to detect the presence of the reference signal in the received signal[13]. Spectrum sensing is a critical function of cognitive radio networks, which aim to improve the utilization of the wireless spectrum by enabling unlicensed users to access unused or underutilized frequency bands.

The basic idea behind matched filter spectrum sensing is to correlate the received signal with the known signal, which is expected to be present in the frequency band of interest. The correlation operation maximizes the signal-to-noise ratio (SNR) of the received signal and enhances the detection of the known signal in the presence of noise and interference.

Figure 1.4 shows the flow of Matched Filter,

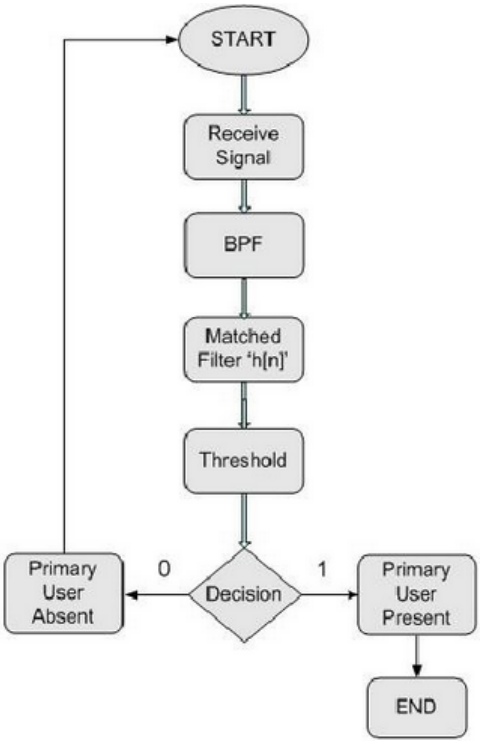

Figure 1.4: Matched Filter

The matched filter for spectrum sensing can be implemented using the following steps:

1. Design the filter: The first step is to design the matched filter by selecting a known signal that is expected to be present in the frequency band of interest. The matched filter is designed to have an impulse response that is the time-reversed and conjugated version of the known signal.The output of the matched filter is given by the convolution of the received signal x(t) with the reference signal h(t), which is expressed as,

$$y(t) = x(t) * h(t)$$

   where,

   * denotes convolution,

   h*(t) denotes the complex conjugate of the reference signal h(t).

   The output signal y(t) is then compared to a predetermined threshold to determine if a signal is present or not. If the output signal exceeds the threshold, the detector declares the presence of a signal in the frequency band of interest.

2. Receive the signal: The next step is to receive the signal in the frequency band of interest using an antenna and a radio receiver.

3. Apply the filter: The received signal is then passed through the matched filter, which correlates the received signal with the impulse response of the filter. The output of the matched filter is a time-varying function that represents the degree of similarity between the received signal and the known signal.

4. Detect the signal: The final step is to detect the presence of the known signal in the received signal by measuring the peak of the output of the matched filter. The peak corresponds to the time delay between the received signal and the known signal, and can be used to estimate the location and strength of the signal in the frequency band of interest.

Matched filter spectrum sensing is a powerful technique that can improve the accuracy and reliability of spectrum sensing in cognitive radio networks. However, it requires prior knowledge of the known signal, which may not be available in all cases. Moreover, it may not work well in the presence of frequency-selective fading or other types of interference. Therefore, other spectrum sensing techniques, such as energy detection, may be used in conjunction with matched filter sensing to improve the overall performance of spectrum sensing in cognitive radio networks.

Cyclostationary

Cyclostationary spectrum sensing is a signal processing technique used in cognitive radio networks to detect the presence of a signal in a given frequency band. The technique is based on the fact that many wireless signals have cyclostationary properties, meaning that their statistical properties vary periodically with time.

Figure 1.5 shows the flow of Cyclostationary feature detection,

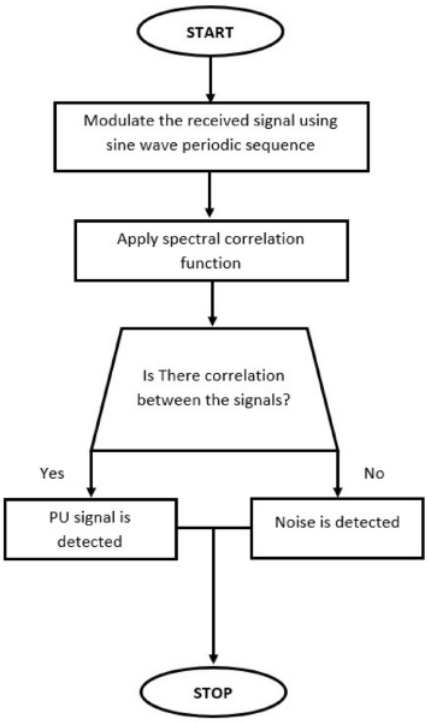

Figure 1.5: Cyclostationary feature detection

The basic idea behind cyclostationary spectrum sensing is to analyze the cyclostationary properties of the received signal and use them to distinguish between the signal of interest and other sources of interference. cyclostationary feature detection technique for spectrum sensing involves analysing the cyclic properties of the received signal to identify unique patterns or features that indicate the presence of a signal of interest in the spectrum[9]. The cyclostationary spectrum sensing algorithm can be implemented using the following steps,

1. Compute the cyclic autocorrelation function: The first step is to compute the cyclic autocorrelation function (CAF) of the received signal. The CAF is a statistical measure that quantifies the cyclostationary properties of the signal. It is defined as,

$$R_x x(\tau, f) = E[x(t)x^*(t-\tau)e^{-j2\pi ft}]$$

where,

x(t) is the received signal,

τ is the cyclic time lag,

f is the cyclic frequency,

E[.] denotes the expected value.

2. Compute the cyclic periodogram: The CAF is then used to compute the cyclic periodogram, which is a spectral analysis tool that provides information about the cyclostationary properties of the signal. The cyclic periodogram is defined as,

$$S_{xx}(f) = \int_{-\infty}^{\infty} R_{xx}(\tau, f)e^{-j2\pi f\tau}d\tau$$

where,

$R_{xx}(\tau, f)$ is the CAF,

$S_{xx}(f)$ is the cyclic periodogram.

3. Estimate the cyclic frequency: The cyclic frequency of the signal is estimated by identifying the peaks in the cyclic periodogram. The peak corresponding

to the cyclic frequency of the signal of interest is used to detect its presence in the frequency band of interest.

4. Set the detection threshold: The detection threshold is set based on the noise level in the frequency band of interest and the desired false alarm probability. If the peak in the cyclic periodogram exceeds the detection threshold, it is concluded that a signal is present in the frequency band of interest. Otherwise, it is concluded that the frequency band is idle.

Cyclostationary spectrum sensing has the advantage of being able to distinguish between the signal of interest and other sources of interference based on their cyclostationary properties. However, it requires prior knowledge of the cyclostationary properties of the signal and is computationally more complex than other spectrum sensing techniques, such as energy detection.

1.3.2 Machine Learning Techniques

Machine Learning is a branch of the broader field of artificial intelligence that makes use of statistical models to develop predictions. It is often described as a form of predictive modelling or predictive analytics and traditionally, has been defined as the ability of a computer to learn without explicitly being programmed to do so.

In basic technical terms, machine learning uses algorithms that take empirical or historical data, analyze it, and generate outputs based on that analysis. In some approaches, the algorithms work with so-called "training data" first and then they learn, predict, and find ways to improve their performance over time. There are three main approaches to machine learning: supervised, unsupervised, and reinforcement learning. There are also hybrid approaches including semi-supervised learning. In supervised learning, the computer is trained on a set of data inputs and outputs, to learn a general rule that maps the given inputs to the given outputs.

Two main types of supervised learning are,

1. Classification: which entails the prediction of a class label.

2. Regression: which entails the prediction of a numerical value.

In unsupervised learning, the learning algorithm is not given this type of guidance; instead, it works to discover the pattern or structure in the input on its own. Two main types of unsupervised learning are,

1. Clustering: which involves discovering groups within the dataset that share similar characteristics.

2. Density estimation: which involves evaluating the statistical distribution of the data set.

Unsupervised learning methods also include visualization with the data and projection, which reduces the dimensions of the data, a form of simplification. In reinforcement learning, the computer and algorithms will confront a problem in a dynamic environment and as it works to perform a given goal, it will receive feedback (rewards), which will reinforce its learning and goal-seeking effort.

Some of the machine learning algorithms that are under classification include Decision Trees, Support Vector Machines (SVMs), and K-Nearest neighbours (KNN) which are explained below.

Decision Tree

It is one of most widely used supervised ML algorithms which can be used for classification as well as regression. It is a structured classifier where at each node, based on certain conditions, splits are being done for the purpose of classification. It has two types of nodes, decision node and leaf node. Decision nodes are used to make decisions based on the specified conditions while leaf nodes are the last nodes of the decision tree that shows the output. DTs are easy to understand as they make decisions similar to human psychology. Figure 1.6 illustrates the working of Decision Tree model.

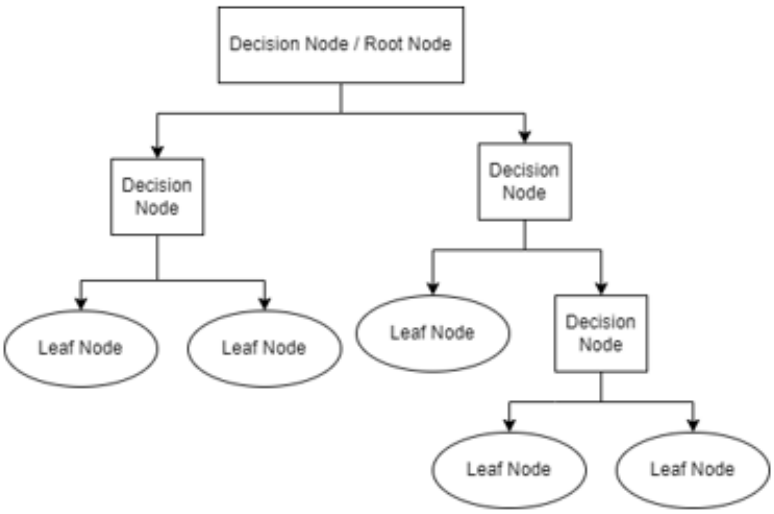

Figure 1.6: Decision Tree Flowchart

The main elements of DT are,

- Root Node: It is the start node of DT. Here, the entire dataset is split into two or more sets.
- Splitting: The process of dividing the nodes into sub-nodes based on conditions is known as splitting.
- Branch: It is a tree formed after splitting.
- Pruning: The process of removing unwanted branches from the tree is known as pruning.
- Parent and Child Node: Root node is parent node and all other nodes are child nodes.
- Leaf Node: They are the last nodes of DT that shows the final output.

For selecting the best attribute in a decision node, different Attribute Selection Measures (ASM) are available. They are,

- Information Gain(G): It shows how much information an attribute gives. It is measured as the change in entropy after splitting the node. The Information Gain is given as,

  $$\text{G} = \text{S} - [(W_{avg}) * S_f]$$

  W_{avg} is Weighted sum of entropy of each child node, S_f is Entropy of each child node

  Entropy(S) is used to measure the impurity of a given attribute. It is given as,

  $$\text{S} = - P_{yes} * log_2(P_{yes}) - P_{no} * log_2(P_{no})$$

  where,

  S : Total number of samples,

  P_{yes} : probability of yes at a particular decision node

  P_{no} : probability of no at a particular decision node

- Gini Index: It measures the impurity of an attribute while creating a split in a node. An attribute with low Gini index should be preferred above high one. If the Gini index is 0.5, it means the elements are equally splitted among the nodes. Gini index is calculated as,

  $$\text{Gini} = 1 - \sum_{i=0}^{n} n_i = (p_i)^2$$

  Thus, it can be said that this algorithm is very simple to understand and can be useful for solving decision related problems. Moreover, less data cleaning is required compared to other algorithms. However, lots of layers make it complex. Furthermore, for more class labels, there is an increase in computational complexity of DT.

K-Nearest Neighbours

KNN is based on supervised ML techniques and it is one of the simplest ML algorithms. The algorithm assumes the similarity between the new case/new data

and the available cases and classifies the new case in the class most similar to the available class. KNN algorithm stores all available data and ranks new data points based on similarity. This means that when new data arrives, it can be easily classified into well group categories using KNN. With the help of KNN, we can easily identify the category or class of a particular dataset. Fig 1.7 illustrates KNN algorithm with cluster formation,

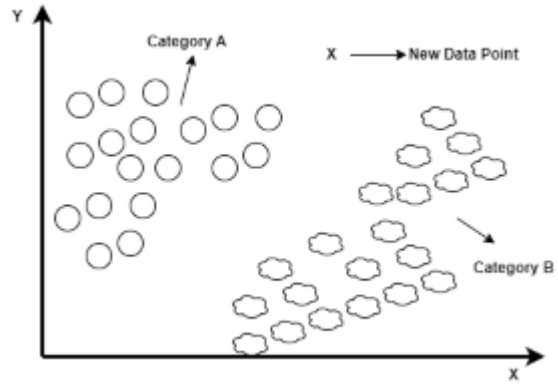

Figure 1.7: KNN Sample showing clusters

Suppose we have a new data point and we need to put it in the required category. Firstly, we will choose the number of neighbours (K). There is no particular way to determine the best value for "K", so we need to try some values to find out the best of them. One of the ways to set the value of K can be found using.

$$\mathrm{K} = \sqrt{n}$$

Where,
"n" is the total number of samples

A very low value for K such as K=1 or K=2, can be noisy and can lead to the overfitting problem. Large values for K are good, but too large value of K can lead to underfitting problem. It could make the model inefficient. Next, the calculation of the Euclidean distance between the new data and available data points are found. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

$$\text{Euclidean distance (d)} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

where,

d is the Euclidean distance,

$P_{yes} = (X_1, Y_1)$ is the coordinate of the first point

$P_{no} = (X_2, Y_2)$ is the coordinate of the second point

By calculating the Euclidean distance, we will get the nearest neighbours. New data will be considered in the category with the highest nearest neighbours.

Support Vector Machine

SVM is one of the most well-liked ML algorithms which is mainly used to solve classification and regression problems. However, it is primarily employed in ML Classification issues. SVM generates a decision boundary which divides n-dimensional space into classes allowing it to quickly classify new data points in the future. It selects the extreme vectors and points that assist in the creation of the hyperplane. Support vectors, which are used to represent these extreme cases, are the basis for the SVM algorithm.

There are two types of SVM:

- Linear SVM: Linear SVM is used for data that can be divided into two classes using a single straight line. This type of data is called linearly separable data, and the classifier employed is known as a Linear SVM classifier.
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data. If a dataset cannot be classified using a straight line, it is considered non-linear data, and the classifier employed is referred to as a Non-linear SVM classifier.

The main elements of SVM are:

- Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

- Support vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. These vectors support the hyperplane, hence called a support vector.

- Branch: Margin: The SVM algorithm determines which line from each class is closest to the other. Support vectors are the names for these points. Margin is the distance between the hyperplane and the vectors.

Maximizing this margin is the aim of SVM. The ideal hyperplane is the one with the largest margin.
Fig 1.8 shows the formation of hyperplane in SVM.

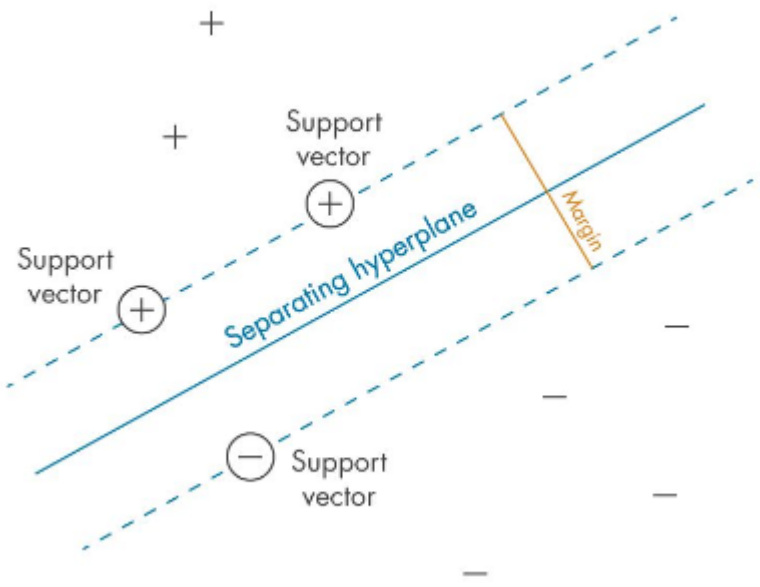

Figure 1.8: SVM sample showing clusters

Chapter 2

Review of Literature Survey

Spectrum sensing is one of the most important functions of cognitive radio. One of the primary functions of CR is spectrum sensing. Sensing of the spectrum holes is the objective of spectrum sensing so as to obtain the band state i.e., whether the spectrum is free or occupied. One of the primary functions of CR is spectrum sensing. Sensing of the spectrum holes is the objective of spectrum sensing to obtain the band state i.e., whether the spectrum is free or occupied.

The concept of cognitive radio architecture in centralized and ad-hoc networks is explored in detail in [4]. The synergy between software-defined radio and cognitive principles is fully exploited by assuming that the cognitive radio architecture is based on a software-defined radio endowed with a perceptive system capable of sensing the radio environment, in particular the use of the radio spectrum. The so-called cognition cycle is an architecture subsystem including an inference hierarchy, the temporal organization and the flow of inferences and control states. The cycle continually observes the environment, orients itself, creates plans, decides, and then acts. The cognition functions are implemented via cognition elements consisting of data structures, processes and flows. These include data structures and related processing elements that may be modelled as topological maps over abstract domains. As software-defined radio is the fundamental platform for cognitive radio, the former is reviewed in this paper, including overviews of software radio architecture (SRA) and software communications architecture (SCA), the transition and evolution towards cognitive radio architecture, and other relevant

technical aspects. Energy detection method for spectrum sensing is used in [6] to sense the presence of primary users. The energy detection algorithm is programmed in MATLAB to determine the availability of the used spectrum of the licensed band. Three different modulation techniques BPSK, QPSK and QAM under Additive White Gaussian Noise (AWGN) are specified for increasing signal quality and detection. P_d increases with an increase in P_{fa}. It is also observed that among these modulation techniques, QAM is the best method for spectrum sensing. Energy Detection performs poorly under a low SNR ratio.

In [13], matched filter-based spectrum sensing in a new cognitive radio scenario where the primary user randomly transmits with one of the multiple power levels is used. Besides, the traditional sensing target where the secondary user (SU) should decide the presence of PU, an additional new target added here could be also recognizing the power levels of PU, which achieves more "cognition" for CR. However, the only drawback of matched filter requires prior knowledge about the PU signal. [9] focuses on cyclostationary feature detection-based spectrum sensing for accurate, fast and efficient primary signal detection. Cyclostationary feature detection (CFD) based spectrum sensing exploits the second-order periodicity of the modulated signal. Spectrum sensing or presence of the primary user (PU) detection is done in the LabVIEW environment in this work. Cyclostationary-based detection gives excellent information on signal characteristics even at very low SNR due to its robustness to the noise and also realizes the modulation scheme used for transmitting the signal. Cyclostationary feature detection is more complex and has reliability issues.

In [3], spectrum sensing is investigated based on energy detection. Scenarios are studied where the noise distribution is Gaussian and scenarios considering the noise of uncertainty. The behaviour of the probability of detection as a function of the probability of a false alarm for a parameterized number of samples and signal-to-noise ratio (SNR) is presented. To evaluate the performance of spectrum sensing methods, simulations are made to the SNR in the range of -25dB to 5 dB. It is observed that noise uncertainty degrades the system performance when the SNR ratio is low. Threshold selection on spectrum sensing performance is difficult to obtain. Paper[5] describes that cognitive radio is a technology which finds the existence of a licensed user in the radio spectrum using dynamic spec-

trum access techniques. A comparison of Energy Detection and Matched Filter based spectrum sensing techniques are done. MATLAB implementation is done by calculating the threshold value by assuming a fixed value of the probability of a false alarm. Then, on comparing the test statistics with the threshold value, the detection is made. The performance obtained from matched filter sensing technique is far better compared to energy detection at low SNR conditions. In terms of complexity, the matched filter is a more complex technique to build than energy detection. Optimization methods for selecting a threshold are required.

The dataset of 1600 samples is generated using an ARDUINO UNO card and a 433 MHz Wireless transmitter (ASK (Amplitude-Shift Keying) and FSK (Frequency-Shift Keying) modulation type) [7]. The reception interface is constructed using an RTL-SDR dongle connected to MATLAB software. This dataset is used for training the machine learning techniques. Evaluation of machine learning-based spectrum sensing is done on the basis of the probability of detection, miss-detection, false alarm and accuracy. Medium Gaussian classification of SVM and coarse classification of KNN perform well in comparison to the other classifiers. The decision tree has good probability of detection but higher value of probability of false alarm which is not desired. The dataset used contains only 1600 signals with just 2 modulation types which are ASK and FSK.

[12] describes spectrum sensing techniques in terms of speed and complexity. Hence, this paper develops an algorithm for the spectrum prediction process in CR adapted by gradient boosting algorithm to predict the status of the channel as busy, idle and middle states. The proposed scheme can predict the channel status effectively with a high prediction accuracy of 99%. Gradient boosting performs better when compared to other algorithms. Allocating the idle channel to the SUs and performing spectrum handoff to SU whenever the PUs want to transmit packets is a challenging task in this model. Also, reducing the bias results in overfitting which can be overcome with ensemble methods.

In [2], a single observation input given to the classifier consists of three statistical features.Both linear and non-linear kernels are used to compare the results. The SVM-based classifier trained with polynomial kernel function surpasses the classifiers trained with RBF kernel function and linear kernel function.

A novel cooperative spectrum sensing (CSS) algorithm for cognitive radio (CR)

networks is used in [11] which is based on machine learning techniques which are used for pattern classification. Unsupervised and Supervised and weighted KNN learning-based classification techniques are implemented for CSS. Supervised SVM classifier with the linear kernel performs well in terms of probability of detection. In terms of updating the training energy vectors in process, the KNN performs extremely well. The proposed CSS approaches can be further improved to gradually train the classifiers by using the energy vectors obtained one by one so that they can adapt to the varying environment without training all over again.

In [1], a spectrum sensing method based on Machine learning models for cognitive radio networks is proposed. A dataset of size 5000 was generated and used to train, validate and test several ML techniques, including random forest, Support Vector Machine with different kernels, decision tree, Naïve Bayes, and K-nearest neighbours. These models are tested on the basis of the Probability of false alarm, Probability of detection and miss-detection. The simulation results show that the random forest outperforms all other machine learning methods.

Lastly in [8], a reliable spectrum sensing scheme is proposed using KNN (Machine learning algorithm). Here each CR produces a sensing report under varying conditions and based on the global decision either it transmits the signal or stays silent. A global decision is made at the fusion centre by combining all the local decisions of CR through majority voting. The proposed scheme has better detection performance and better spectral hole exploitation capability than the conventional OR rule. Even at -10 dB SNR this model successfully detects 80 per cent of the time. A soft decision combination has been used which gives a lot of overhead.

Chapter 3

System Model

3.1 Problem statement

Spectrum sensing is a key function of cognitive radio to prevent harmful interference with licensed users (PUs) and identify the available spectrum for improving the spectrum's utilization [9]. However, detection performance in practice is often compromised with Multi-path Fading, shadowing and receiver uncertainty issues. To mitigate the impact of these issues, cooperative spectrum sensing is an effective method to improve detection performance by exploiting spatial diversity. Moreover, due to the spectrum being mostly allocated and divided by Service providers to the Primary users, providing more spectrum to expand existing services or offer new ones has become more challenging. Recent studies have shown that the issue is not the lack of spectrum, but rather spectrum access This means that the capacity of the spectrum is not being exploited to its full extent [7].

3.2 Working

Cognitive radio is mainly used to intelligently sense the RF spectrum to get a brief idea of the number of channels being used by PUs (Primary users) and the number of channels which are vacant and not being used. All the CR Secondary users perform spectrum sensing using Cognitive radio to check the status of the channels. If a particular channel of a licensed user is not in use and can be used

for communication the access is given to a secondary user. This decision is made with a lot of pre-determined factors into consideration and traditional techniques like measuring the energy of a signal using an energy detector and various other methods are used to detect the presence/absence of a user. However, the accuracy is not satisfactory and these techniques do not perform well in real-time conditions where a lot of external factors like environment, random signals, and channel noise are taken into consideration. So, Machine learning algorithms are adopted to adaptively sense the environment and learn from the randomness for much better accuracy. This makes the overall system tolerant to any type of condition. We are comparing three machine learning algorithms that are Decision Tree (DT), Support Vector Machine (SVM) and K – Nearest Neighbours (KNN) algorithms with various pre-defined factors and training these algorithms to calculate the accuracy of each algorithm. The dataset of the first 20 Samples was implemented first using the Decision tree algorithm which is a classification ML-based algorithm. The accuracy of the decision tree-based algorithm implemented was very less than expected (50%). Afterwards, a random dataset of 1000 samples was generated. Here, 800 samples were used as training datasets and 200 were used as testing datasets. In the decision tree, the split of a particular node is based on the information gain and entropy of a particular attribute. The attribute with the maximum gain is chosen for the split of a node. Thus, the decision of presence or absence will be taken accordingly. In the KNN algorithm, different clusters are created based on the number of neighbours (value of K) specified. The given set of data is then arranged into these clusters taking into consideration which data is closer to which cluster. As a result, the prediction of presence or absence will be taken. In the SVM algorithm, the dataset will be split into two regions with a support vector as a boundary condition for the prediction. Finally, the model will be able to decide whether the primary user is present or absent in a particular spectrum.

3.3 Implementation

The proposed solution for spectrum sensing has been implemented using three machine learning algorithms, namely decision tree, k nearest neighbour and support vector machine algorithms.

3.3.1 Data Acquisition

The most important step for our project is acquiring dataset. The dataset used in this project is obtained from IEEE Dataset website [10]. Fig 3.1 illustrates the dataset obtained from [10] which has the following attributes,

	I/Q signal	Modulation_type	SNR	Amplitude	Primary_User_Presence
0	-5.3783-10.269i	13	-20	11.592173	Yes
1	0.12619-0.071555i	23	0	0.145066	Yes
2	0.07027-1.4843i	50	-10	1.485962	Yes

Figure 3.1: Dataset

1. Signal I/Q Information:
   I/Q information refers to the in-phase (I) and quadrature (Q) components of a signal, which are used to represent the amplitude and phase information of a modulated signal.

   In a modulated signal, the I component represents the amplitude of the signal at the carrier frequency, while the Q component represents the amplitude of the signal at a frequency that is shifted by 90 degrees (i.e., in quadrature) from the carrier frequency. Together, these two components contain all the information needed to fully represent the modulated signal.

   The I/Q information is commonly used in various communication systems, such as digital radio and software-defined radio (SDR), where the I/Q data is often processed digitally to extract the signal's amplitude and phase information. The I/Q data can also be used to implement various modulation schemes, such as amplitude modulation (AM), frequency modulation (FM), and phase modulation (PM), among others.

2. Signal to Noise ratio(SNR) Ratio:
   SNR (Signal-to-Noise Ratio) is a commonly used metric in spectrum sensing that describes the ratio of the signal power to the noise power in a particular frequency band. In the context of spectrum sensing, the SNR can be used to determine the reliability of the signal detection. A high SNR indicates a strong signal presence relative to the noise, which makes it easier to detect the signal. Conversely, a low SNR indicates a weak signal presence relative to the noise, which can make it more difficult to accurately detect the signal. In order to improve the SNR for spectrum sensing, various techniques can be employed such as signal filtering, averaging, and adaptive thresholding. These techniques can help reduce the noise in the signal and improve the sensitivity of the spectrum sensing system.

3. Modulation Types:
   Modulation is the process of modifying a carrier signal in order to encode information onto it. In other words, it involves altering certain properties of the carrier signal, such as its amplitude, frequency, or phase, to represent the information being transmitted.

   There are several types of modulation, including:

   (a) Analog Modulation:
   Analog modulation is a type of modulation that involves the modulation of an analog carrier signal to transmit analog information, such as voice or music. Examples of analog modulation techniques include amplitude modulation (AM), frequency modulation (FM), and phase modulation (PM).

   (b) Frequency Shift Keying (FSK):
   FSK is a digital modulation technique that involves modulating the frequency of a carrier signal to represent digital information. It is commonly used in applications such as wireless communication, digital data transmission, and satellite communication.

   (c) Phase Shift Keying (PSK):
   PSK is a digital modulation technique that involves modulating the

phase of a carrier signal to represent digital information. It is commonly used in applications such as wireless communication, digital data transmission, and satellite communication.

(d) Pulse Amplitude Modulation (PAM): PAM is a digital modulation technique that involves modulating the amplitude of a series of pulses to represent digital information. It is commonly used in applications such as digital data transmission and signal processing

(e) Quadrature Amplitude Modulation (QAM):
QAM is a digital modulation technique that involves modulating the amplitude and phase of a carrier signal to represent digital information. It is commonly used in applications such as wireless communication, digital data transmission, and digital television broadcasting. QAM is often used in conjunction with other modulation techniques such as PSK or FSK

The choice of modulation type depends on various factors such as the bandwidth available, the noise in the transmission medium, the distance between the transmitter and receiver, and the type of information being transmitted.

3.3.2 Data Processing

The dataset obtained from IEEE Dataport[10] had three attributes which are:

1. Signal I/Q information.
2. Signal to Noise(SNR) Ratio.
3. Modulation Types.

In the obtained dataset, the information about the amplitude and presence or absence of the primary user was not mentioned. From the I/Q information, we obtained the amplitude of the signal as follows,

$$Amplitude = \sqrt{I^2 + Q^2}$$

For obtaining the decision if primary user is present or absent different threshold conditions are applied on amplitude and SNR on signals of different modulation types which is shown in Table 3.1.

Table 3.1: Threshold conditions

Modulation Type	SNR(dB)	Amplitude(V)
BPSK	3.8	-
QPSK	6.6	-
8PSK	9.4	-
16PSK	12.1	-
32PSK	14.9	-
64PSK	17.7	-
4QAM	6.0	0.89
8QAM	8.4	0.71
16QAM	11.1	0.56
32QAM	13.9	0.44
64QAM	16.7	0.35
128QAM	19.5	0.28
256QAM	22.3	0.22
2FSK	2.9	-
4FSK	5.3	-
8FSK	8.1	-
16FSK	10.8	-
4PAM	3.3	0.97
8PAM	6.0	0.89
16PAM	8.8	0.71
AM-DSB	-0.4	1.41
AM-DSB-SC	3.3	0.97
AM-USB	-2.2	1.78
AM-LSB	-2.2	1.78
FM	14.9	-
PM	12.7	-

3.3.3 Software used

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-

friendliness, has made it one of the most-used programming languages today. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non- programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

Some of them are as follows,

- Data analysis and machine learning
- Web development
- Automation or scripting Software testing and prototyping
- Everyday tasks
- It has a simple syntax that mimics natural language, so it's easier to read and understand. This makes it quicker to build projects, and faster to improve on them.
- It's versatile. Python can be used for many different tasks, from web development to machine learning.
- It's beginner friendly, making it popular for entry-level coders.
- It's open source, which means it's free to use and distribute, even for commercial purposes.
- Python's archive of modules and libraries—bundles of code that third-party users have created to expand Python's capabilities—is vast and growing.
- Python has a large and active community that contributes to Python's pool of modules and libraries and acts as a helpful resource for other programmers. The vast support community means that if coders run into a stumbling block, finding a solution is relatively easy; somebody is bound to have encountered the same problem before.

Google Colab is an online notebook-like coding environment that is well-suited for machine learning and data analysis

- It comes equipped with many Machine Learning libraries and offers GPU usage. It is mainly used by data scientists and ML engineers. Google Colab is free
- Easy to get started
- Allows access to GPUs/TPUs
- Easy to share code with others
- Easy graphical visualizations in Notebooks

JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

The Jupyter Notebook has several menus that you can use to interact with your Notebook. The menu runs along the top of the Notebook just like menus do in other applications. Here is a list of the current menus:

- File
- Edit
- View
- Insert
- Cell
- kernel
- Widgets
- Help

3.3.4 Training the Models

There are many ML algorithms which could be addressed for SS. In order to predict the presence of user in a channel, we have employed three different ML methods to identify vacant spectrum. They are DT, KNN and SVM. Total number of samples used to train the model is 300000. The ratio of training to testing samples is 2:1, i.e. 2000000 samples of dataset are used for training the model and 1000000 samples are used for testing the models.

1. **Training the DT model**: In order to train DT model, the classifier DT was used. In this, the root node was chosen as SNR because the SNR attribute had the least Gini index value and the decision node is chosen which has a low value of Gini index. This parameter determines the maximum depth of the decision tree, which controls the complexity of the model. A higher max(depth) can lead to a more complex tree that may be prone to overfitting, while a lower max(depth) can result in a simpler tree that may underfit the data."max(depth)" parameter was set to 15 in this model.

2. **Training the KNN model**: In KNN model finding out the appropriate value of K is an integral part. This is because if we choose a lower value of K, then the model will be overfitted while on the other hand if its value is high then the model may be underfitted. Thus, we have to choose an optimal value of K. In our model, we have used 447 as the value of K after calculating using Eqn(i) with the number of training samples. At this value we get appropriate results. 'metric' parameter specifies the distance metric used to compute distances between samples. In this model, metric is set as 'minkowski' which is a generalisation of Manhattan and euclidean distance. 'weights' parameter specifies the weight function used in prediction. It is set to 'uniform', where all neighbours have equal weights.

3. **Training the SVM model**: In this model, we used three kernels, namely linear, Radial Basis Function (RBF) and polynomial kernels. Among these, RBF and polynomial kernels had low efficiency of detection. As a result, the linear kernel was finalized to be used as the kernel in SVM. To get more ac-

curate results, certain parameters of SVM are varied. C is the regularization parameter and it is set to 1 in this model as a smaller value of C is preferred for large datasets, as it can help prevent overfitting.Another parameter which is used to reduce the training time for large datasets is 'shrinking' which is set to "True" by default.

3.3.5 Performance measures

The performance of all the three models are measured on the basis of the following parameters:

1. **Accuracy**: Accuracy in spectrum sensing refers to the ability of a spectrum sensing system to correctly identify the presence or absence of a primary user signal in a specific frequency band. The accuracy of spectrum sensing is crucial for the efficient utilization of the radio spectrum, which is a limited resource.

2. **Probability of detection**: The probability of detection (P_d) in spectrum sensing is a measure of the ability of a spectrum sensing system to correctly identify the presence of a primary user signal in a specific frequency band. It is defined as the probability that the sensing system correctly detects the presence of the primary user signal when it is actually present.

   P_d is an important performance metric for spectrum sensing because it determines the ability of the sensing system to accurately detect the primary user signal, which is essential for the efficient utilization of the radio spectrum. A high P_d means that the sensing system is able to accurately detect the primary user signal, while a low P_d means that the system is more likely to miss the primary user signal

3. **Probability of false alarm**: The probability of false alarm (P_{fa}) in spectrum sensing is a measure of the probability that a spectrum sensing system incorrectly detects the presence of a primary user signal when it is actually absent in a specific frequency band. In other words, P_{fa} measures the probability that the sensing system indicates the presence of a primary user signal when no signal is present.

P_{fa} is an important performance metric in spectrum sensing because it determines the ability of the sensing system to avoid false detection of primary user signals, which can result in unnecessary interference with other wireless systems operating in the same frequency band.

Chapter 4

Results and Analysis

In this chapter,the results are analysed for different ML models on the basis of performance measures. Confusion matrix is used to find the performance measures and it is explained in Section 4.0.1.

4.0.1 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm.

The confusion matrix consists of four basic characteristics that are shown in Fig 4.1 that are used to define the measurement metrics of the classifier,

1. True Positive:Predicted positive and it's true.
2. True Negative: Predicted negative and it's true.
3. False Positive: Predicted positive and it's false.
4. False Negative: Predicted negative and it's false

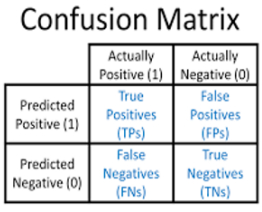

Figure 4.1: Confusion matrix

4.1 Decision Tree

Figure 4.2 shows the training accuracy of the DT model. The training accuracy of the model is found to be 98.56%

Feeding the train data into the model

```
In [6]: model_DT = tree.DecisionTreeClassifier(criterion='gini',max_depth=15, min_samples_split=10)
        model_DT.fit(inputs_train,target_train)

Out[6]: DecisionTreeClassifier(max_depth=15, min_samples_split=10)

In [7]: model_DT.score(inputs_train,target_train)

Out[7]: 0.98561
```

Figure 4.2: DT training output

Fig 4.3 shows the testing accuracy and number of misclassified samples. The testing accuracy of the model is found to be 98.49%, 1501 samples are misclassified or predicted incorrectly.

Finding the testing accuracy of the model

```
In [12]: model_DT.score(inputs_test,target_test)

Out[12]: 0.98499
```

Finding out number of misclassified samples

```
In [13]: target_pred = model_DT.predict(inputs_test)

         misclassified = np.where(target_test!=target_pred)
         len(misclassified[0])

Out[13]: 1501
```

Figure 4.3: DT testing output

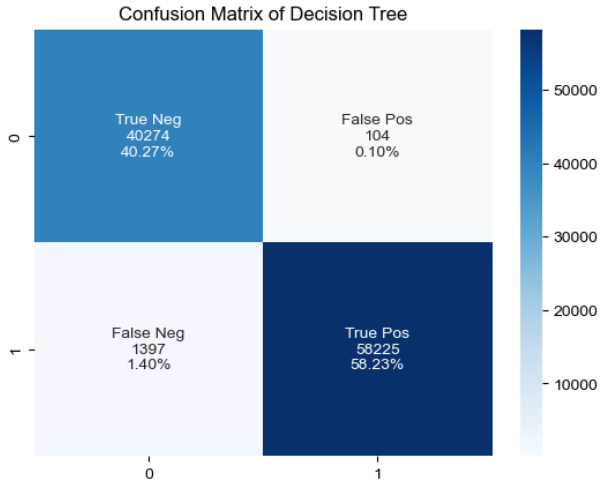

Figure 4.4: Confusion matrix of DT

The Confusion matrix for DT model is given in Fig 4.4 and the Probability of detection (P_d) and Probability of false alarm (P_{fa}) can be obtained from the confusion matrix and is given in Fig 4.5,

Obatining the Probability of Detection

```
In [16]: PD = TP/(FN+TP)
         print("The probablity of detection of the model is "+ str(PD*100) + '%')

         The probablity of detection of the model is 97.65690516923283%
```

Obtaining the Probablilty of False Alarm

```
In [17]: PF = FP/(FP+TN)
         print("The probablity of false alarm of the model is "+ str(PF*100)+'%')

         The probablity of false alarm of the model is 0.25756600128783%
```

Figure 4.5: The Probability of detection (P_d) and Probability of false alarm (P_{fa}) for DT

4.2 Support Vector Machine

Fig 4.6 shows the training accuracy of SVM model. The training accuracy of the model is found to be 74.25%.

Feeding the train data into the model

```
In [7]: svmclassifier = SVC(kernel='linear',C=1, shrinking=True )   #For linearly seperable data
        svmclassifier.fit(inputs_train, target_train)

Out[7]: SVC(C=1, kernel='linear')

In [7]: svmclassifier.score(inputs_train,target_train)

Out[7]: 0.74259
```

Figure 4.6: SVM training output

Fig 4.7 shows the testing accuracy and number of misclassified samples. The testing accuracy of the model is found to be 74.22%, 25772 samples are misclassified.

Finding the testing accuracy of the model

```
In [12]: svmclassifier.score(inputs_test,target_test)

Out[12]: 0.74228
```

Finding out number of misclassified samples

```
In [13]: target_pred = svmclassifier.predict(inputs_test)

         misclassified = np.where(target_test!=target_pred)
         len(misclassified[0])

Out[13]: 25772
```

Figure 4.7: SVM testing output

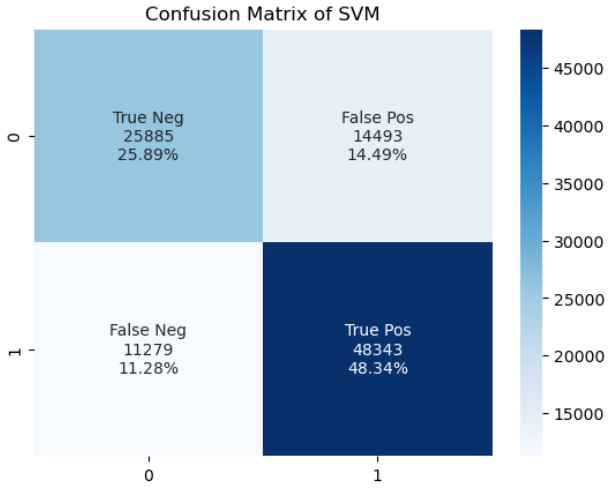

Figure 4.8: Confusion matrix for SVM

The Confusion matrix for SVM model is given in Fig 4.8 and the Probability of detection (P_d) and Probability of false alarm (P_{fa}) can be obtained from the confusion matrix and is given in Fig 4.9,

Obatining the Probability of Detection

```
In [16]: PD = TP/(FN+TP)
         print("The probablity of detection of the model is "+ str(PD*100) + '%')

         The probablity of detection of the model is 81.08248633054913%
```

Obtaining the Probablilty of False Alarm

```
In [17]: PF = FP/(FP+TN)
         print("The probablity of false alarm of the model is "+ str(PF*100) + '%')

         The probablity of false alarm of the model is 35.89330823715885%
```

Figure 4.9: The Probability of detection (P_d) and Probability of false alarm (P_{fa}) for SVM

4.3 K-Nearest Neighbour

Fig 4.10 shows the training,testing accuracy and number of misclassified samples. The training accuracy of the model is found to be 95.25% The testing accuracy of the model is found to be 95.08%, 4915 samples are misclassified.

Feeding the train data into the model and finding its training accuracy

```
In [10]: classifier = KNeighborsClassifier(n_neighbors=447,weights='uniform', p=1, metric='minkowski')
         classifier.fit(inputs_train,target_train)

Out[10]: KNeighborsClassifier(n_neighbors=447, p=1)

In [11]: classifier.score(inputs_train,target_train)*100

Out[11]: 95.25200000000001
```

Finding the testing accuracy of the model

```
In [12]: classifier.score(inputs_test,target_test)*100

Out[12]: 95.085
```

Finding out number of misclassified samples

```
In [13]: target_pred = classifier.predict(inputs_test)

         misclassified = np.where(target_test!=target_pred)
         len(misclassified[0])

Out[13]: 4915
```

Figure 4.10: KNN training and testing output

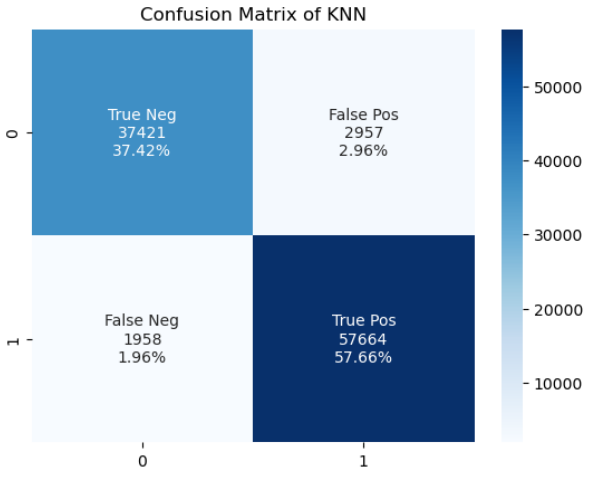

Figure 4.11: Confusion matrix for KNN

The Confusion matrix for KNN model is given in Fig 4.11 and the Probability of detection (P_d) and Probability of false alarm (P_{fa}) can be obtained from the confusion matrix and is given in Fig 4.12,

Obatining the Probability of Detection

```
In [16]: PD = TP/(FN+TP)
         print("The probablity of detection of the model is "+ str(PD*100) + '%')

         The probablity of detection of the model is 96.71597732380664%
```

Obtaining the Probablilty of False Alarm

```
In [17]: PF = FP/(FP+TN)
         print("The probablity of false alarm of the model is "+ str(PF*100) + '%')

         The probablity of false alarm of the model is 7.323294863539551%
```

Figure 4.12: The Probability of detection (P_d) and Probability of false alarm (P_{fa}) for KNN

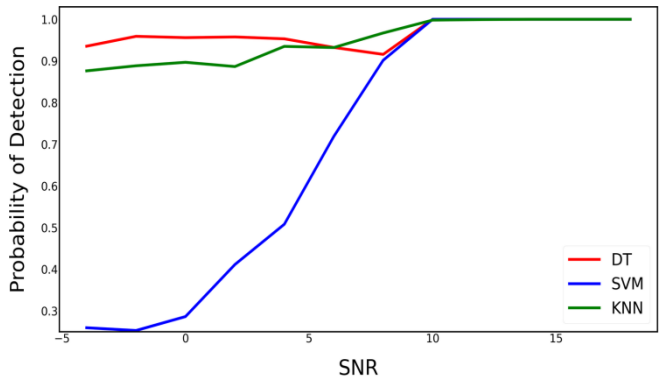

Figure 4.13: P_d Vs SNR

Fig. 4.13 shows the probability of detection Vs SNR for various models. It is observed that in case of SVM for low values of SNR, the probability of detection is very low and detection is difficult with this algorithm. However, DT and KNN perform better for lower values of SNRs. Also, it can be observed that for SNR greater than 10 dB, all algorithms perform similarly so any algorithm can be chosen for higher values of SNR.

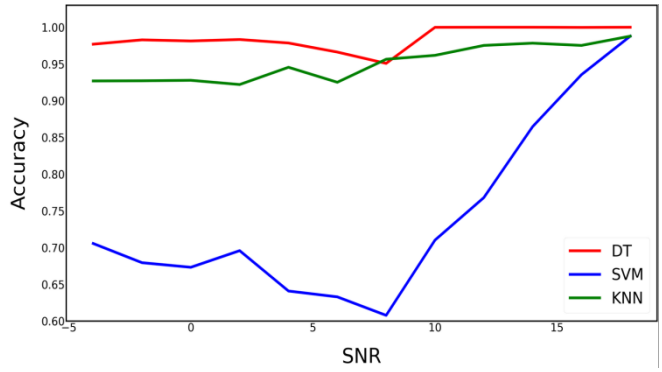

Figure 4.14: Accuracy Vs SNR

Fig. 4.14 shows the Accuracy Vs SNR for various models. It is observed that for SVM the accuracy decreases at first when SNR is increased as it is not able to distinguish between signal and noise at low values of SNR and SVM is a complex model and it undergoes overfitting which includes the underlying features which has information about signal and noise and also fits random noise present in the data. The accuracy increases after a certain value of SNR as the signal becomes prominent and it is easier for the model to predict presence of user. Even for KNN, accuracy decreases at certain SNR values due to overfitting problem. Overfitting occurs for lower values of K in KNN. Thus, for higher SNR values, all models give high accuracy but for lower SNR values DT is the best choice.

Table 4.1: Performance measures for various ML models

Models	Accuracy	P_d	P_{fa}
Decision Tree	98.49	0.97	0.0025
Support Vector Machine	74.22	0.81	0.35
K-Nearest Neighbour	95.08	0.96	0.07

Table 4.1 illustrates the performance measures that are accuracy, P_d and P_{fa} for different ML models that are DT, SVM and KNN. Out of the three models, DT is found to perform better based on the evaluation parameters as this algorithm is very simple to understand and it has good computational efficiency.

Chapter 5

Conclusion

The conventional methods for SS like, energy detection, matched filter and cyclostationary are time consuming and require prior knowledge of the surrounding environment. Moreover, SS using matched filter is complex as it requires threshold optimization. Hence, there is a need of simple and accurate method. Machine Learning model is a viable solution that provides simplicity, high accuracy and does not require any prior knowledge. One of the main criteria for using ML algorithms in spectrum sensing is the ability to utilize them in the absence of knowledge of the radio parameters such as Probability of Detection (Pd), Probability of False alarm (Pfa), SNR values etc. In this paper, larger data size chosen gives better training accuracy as compared to [5]. Decision tree performs better due to higher computational efficiency. Probability of Detection is high for higher values of SNR. At low SNR also our model shows higher probability of detection compared to[12]. Probability of False alarm obtained for all the three models are almost negligible and are much better than in paper[12]. ML techniques are preferred over non-ML techniques for spectrum sensing in cognitive radio due to their improved accuracy, adaptability, robustness, flexibility, and scalability. ML algorithms can learn complex patterns from dynamic spectrum data, handle diverse types of data, and scale well to large datasets, making them well-suited for spectrum sensing tasks in cognitive radio networks.

References

[1] Youness Arjoune and Naima Kaabouch. On spectrum sensing, a machine learning method for cognitive radio systems. In *2019 IEEE International Conference on Electro Information Technology (EIT)*, pages 333–338. IEEE, 2019.

[2] Sana Ullah Jan, In Soo Koo, et al. Performance analysis of support vector machine-based classifier for spectrum sensing in cognitive radio networks. In *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 385–3854. IEEE, 2018.

[3] Euler Edson Apaza Medina and Silvio Ernesto Barbin. Performance of spectrum sensing based on energy detection for cognitive radios. In *2018 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, pages 948–951. IEEE, 2018.

[4] Joseph Mitola and Gerald Q Maguire. Cognitive radio: making software radios more personal. *IEEE personal communications*, 6(4):13–18, 1999.

[5] A Parvathy and Gayathri Narayanan. Comparative study of energy detection and matched filter based spectrum sensing techniques. In *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 147–153. IEEE, 2020.

[6] Ashish Ranjan, Balwinder Singh, et al. Design and analysis of spectrum sensing in cognitive radio based on energy detection. In *2016 International Conference on Signal and Information Processing (IConSIP)*, pages 1–5. IEEE, 2016.

[7] Mohammed Saber, Abdessamad El Rharras, Rachid Saadane, Aroussi Hatim Kharraz, and Abdellah Chehri. An optimized spectrum sensing implementation based on svm, knn and tree algorithms. In *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 383–389. IEEE, 2019.

[8] Hurmat Ali Shah and Insoo Koo. Reliable machine learning based spectrum sensing in cognitive radio networks. *Wireless Communications and Mobile Computing*, 2018, 2018.

[9] Khadeeja Sherbin and V Sindhu. Cyclostationary feature detection for spectrum sensing in cognitive radio network. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1250–1254. IEEE, 2019.

[10] K Tekbıyık, C Keçeci, AR Ekti, A Görçin, and GK Kurt. Hisarmod: A new challenging modulated signals dataset. *IEEE Dataport*, 2019.

[11] Karaputugala Madushan Thilina, Kae Won Choi, Nazmus Saquib, and Ekram Hossain. Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE Journal on selected areas in communications*, 31(11):2209–2221, 2013.

[12] N Usha, K Viswavardhana Reddy, and NN Nagendra. Dynamic spectrum sensing in cognitive radio networks using ml model. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 975–979. IEEE, 2020.

[13] Xinzhi Zhang, Rong Chai, and Feifei Gao. Matched filter based spectrum sensing and power level detection for cognitive radio network. In *2014 IEEE global conference on signal and information processing (GlobalSIP)*, pages 1267–1270. IEEE, 2014.