Detection of underperformance causes in 802.11 wireless networks through machine learning methods.

By Nikolaos Sakellariou

Dissertation for the degree of Bachelor

Supervisor: Korakis Athanasios
Assistant professor
Argyriou Antonios
Assistant professor

Volos, 2018

Πανεπιστήμιο Θεσσαλίας
Τμήμα ηλεκτρολόγων μηχανικών και μηχανικών υπολογιστών

Ανίχνευση αιτιών μειωμένης απόδοσης σε ασύρματα δίκτυα 802.11 με μεθόδους μηχανικής μάθησης

Νικόλαος Σακελλαρίου

Διπλωματική εργασία

Επιβλέποντες καθηγητές : Κοράκης Αθανάσιος
Επίκουρος καθηγητής ΠΘ
Αργυρίου Αντώνιος
Επίκουρος καθηγητής ΠΘ

Βόλος, 2018

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία αποτελεί την ολοκλήρωση των προπτυχιακών σπουδών μου στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας. Αποτελεί προϊόν εκτενούς αναζήτησης και έρευνας και πραγματοποιήθηκε με την υποστήριξη και καθοδήγηση συγκεκριμένων ανθρώπων, τους οποίους θα ήθελα να ευχαριστήσω θερμά.

Θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου, Αθανάσιο Κοράκη και Αντώνιο Αργυρίου για την εμπιστοσύνη που μου έδειξαν, αναθέτοντάς μου τη συγκεκριμένη εργασία.

Θα ήθελα να ευχαριστήσω, ακόμη τον Dr. Στράτο Κερανίδη για την καθοδήγηση και την ουσιαστική συμβολή του στην περάτωση της εργασίας.
Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον συνάδελφο Ηλία Συρίγο, καθώς διαδραμάτισε σημαντικό ρόλο στην εξέλιξη της μεταπτυχιακής διατριβής προσφέροντας απλόχερα τη βοήθεια του οποτεδήποτε και αν χρειάστηκε.

Νικόλαος Σακελλαρίου

Βόλος, 2018

ΠΕΡΙΛΗΨΗ

Συγκεκριμένες παθογένειες των WLAN ασύρματων δικτύων σε ρεαλιστικές περιστάσεις όπως στα οικιακά δίκτυα είναι δύσκολο να ανιχνευθούν εξαιτίας της πολυπλοκότητας του ασύρματου μέσου. Σε προηγούμενες μελέτες έχουν δοκιμαστεί διάφορες τεχνικές για την αναγνώριση και αντιμετώπιση των παθογενειών όπως η χρήση πολύ εξεζητημένου εξοπλισμού, αλλαγές στους οδηγούς των ασύρματων καρτών των συσκευών ή ακόμα και τεχνικές σε επίπεδο χρήστη. Στα πλαίσια αυτής της διατριβής θα δούμε μια τεχνική μηχανικής μάθησης με επίβλεψη η οποία χρησιμοποιεί στατιστικά των συσκευών που διαθέτουν οι οδηγοί καρτών σε επίπεδο χρήστη ώστε να προβλέψει και να αναγνωρίσει με ακρίβεια τις παθογένειες αυτές. Θα χρησιμοποιηθούν τέσσερεις αλγόριθμοι μηχανικής μάθησης με επίβλεψη και οι επιδόσεις τους θα συγκριθούν μεταξύ τους για την ταξινόμηση πέντε γνωστών παθογενειών του 802.11 πρωτοκόλλου(802.11 Contention,non-802.11 Contention, Low - SNR, Hidden Terminal, Capture Effect).

ABSTRACT

Specific WLAN pathologies experienced in realistic scenarios are hard to detect, due to the complexity of the wireless medium. Prior work has employed sophisticated equipment, driver modifications, or even application-layer techniques, towards diagnosing such pathologies. The main approach described in this thesis uses machine learning tools and techniques in order to correctly predict and classify five classes of problems. These problems are labeled as 802.11 Contention, non-802.11 Contention, Low - SNR, Hidden Terminal and Capture Effect. To begin with, a scenario was established in the Nitos testbed for each of these four pathologies. About 400 experiments were run for each of them to gather a satisfying training set. Then an svm (Support Vector Machines) model is trained resulting in an over 99% prediction accuracy. The exceptional results of the svm algorithm will be presented alongside the results (accuracy, precision, recall, training time) of other machine learning algorithms on the classification of these pathologies.

Thesis Subject

This thesis and the software developed in the context of, targets to contribute in the section of wireless networks. This software will be used in order to detect and classify pathologies of IEEE 802.11.

Table of Contents

1 Introduction

Nowadays, wireless local area networks – WLANS are an indefensible part of our everyday life. Their number is growing rapidly and with the many kind of devices that coexist simultaneously (mobile phones, tablets, wireless keyboards and mouses), the performance of these networks inevitably drops.

The most common WLAN pathogenies that support 802.11(Wi-Fi) technology are related with the bad channel quality between the Access Point and Station. However, there are also many cases that even a good quality communication channel can have very low performance because of malfunctions that exist in 802.11 wireless protocol's function like medium contention. Moreover, some other known pathogenies are related to the Hidden Terminal and Capture effect problems that are present mainly in dense topologies and are very common in urban areas where many wireless devices co-exist in just a few square meters area.

Detection and resolution of problems in 802.11 networks is an especially difficult procedure because of the complex and dynamic nature of the wireless medium. It demands the collection of specific information from the lowest layers of 802.11 protocol's operation that is difficult to be interpreted even from experts.

In corporate networks, the administrators of the network that have a very good understanding of the 802.11 protocol's functionality and they also possess specific tools, are responsible for giving solution in these problems. On the other hand, home owners are challenged to give solution to a much more complex problem because of the anarchic structure of WLANS, lack of specialized knowledge but also luck of specialized equipment.

Because of the above reasons and also because of the growing spreading of WLAN technologies, it is very important the development of

automatic tools that are able to give accurate solutions on the performance problems that exist in complex WLAN topologies.

In the effort to solve the performance problems of 802.11 protocol, different approaches have been tried that vary from in depth research of specific pathogenies though the use of specialized equipment, to solutions based on modification of wireless card's drivers for the collection of appropriate statistics.

The solution that is presented in this thesis is a mechanism that is able to detect pathogenies which cause reduced performance in the WLAN networks functionality by using the available information at user level from conventional Access Point devices.

This information is collected at the Physical Layer and is related to statistics such as packet rate, number of attempts, PDR. By owning these statistics, systems performance is strongly related in the effective interpretation of the way these pathogenies affect the natural operation of 802.11 protocol.

The main functionality of the prementioned mechanism is an algorithm that makes use of feature extraction and machine learning techniques in order to

- extract meaningful information from the collected data
- be trained with them
- and eventually to accurately predict when such a pathogeny exists.

More specifically the steps that took place in the context of this thesis are:

1. Modeling and extended experimentation of five well known pathologies in a controlled environment at nitos testbed for the collection of data for each pathogeny. Scripts for the automatic running and logging of the experiments were made (autoscript.py, auto_script2.py).

2. These data consist the training and testing set for our machine learning algorithms. Some scripts for the automatic parsing and exporting of the data in csv format were created (try_csv_exp1.py, try_csv_exp2.py, try_csv_exp3.py, try_csv_exp4.py, try_csv_exp5.py).

3. Five well known machine learning algorithms were applied on our data in order for the optimum one to be chosen.
   Four algorithms for **supervised learning**

   - Decision Trees
   - Random Forest
   - Naive Bayes
   - Support Vector Machines

   And one in **unsupervised learning** for understanding if an accurate classification could take place without providing our model with labels (our five classes of problems)
   - K-Means Clustering

4. Conclusions were made based on the 5 algorithms' performances regarding accuracy, precision, recall and computation time.

2. Background information

In this chapter some basic concepts are explained regarding the 802.11 protocol and Machine Learning.

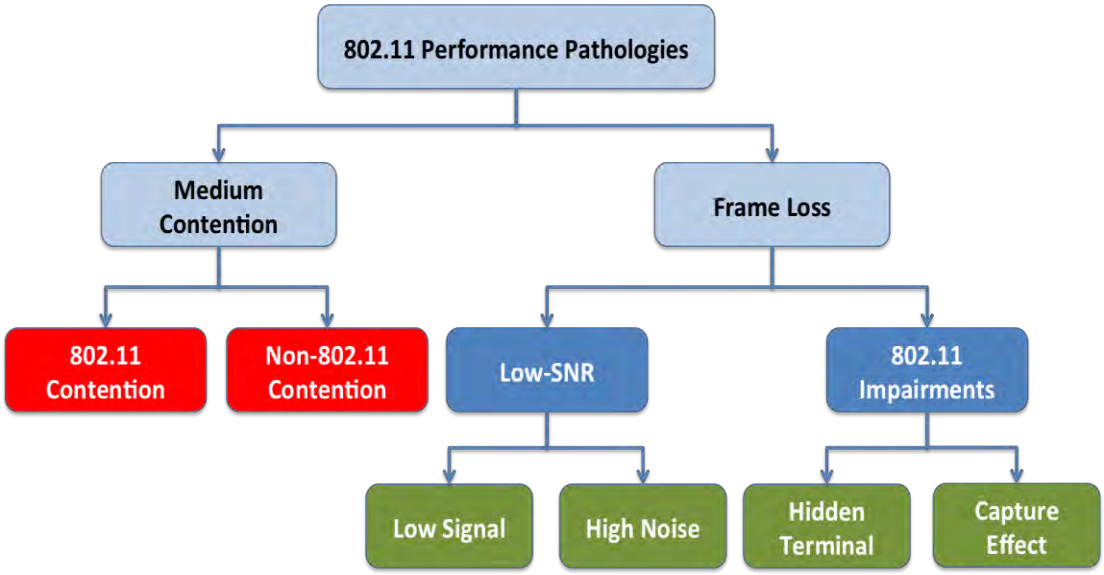

Figure 1: Pathogeny classification

2.1 802.11 protocol

Performance of an 802.11 wireless network depends equally on the **availability for channel accesses on the wireless medium(PDR)** as also on **the percentage of the successful transmissions** (number of attempts). The mechanism that detects and classifies the pathogenies is based on the above assumption. In the beginning for simplicity, our pathogenies are classified in two classes. The first is Medium Contention where the device that is about to make a transmission finds the medium occupied and postpones the transmission. The second class is Frame Loss which contains pathogenies where the wireless medium is detected as free from the device that wants to start a transmission.

The transmission begins but eventually the packet fails to be received successfully. This classification can be seen in figure 1.

2.2 Machine Learning Algorithms

2.2.1 Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel Functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosin Kernel Functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation

2.2.2 Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

2.2.3 Random Forest

In random forests each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

2.2.4 Naïve Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector $x_1$through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive independence assumption that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori Map (MAP) estimation to estimate P(y) and $P(x_i \mid y_i)$; the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

On the flip side, although naive Bayes is known as a decent classifier, it is known to be a bad estimator, so its probability outputs are not to be taken too seriously.

2.2.5 K-Means Clustering

The k-Means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids"; note that they are not, in general, points from X, although they live in the same space. The K-means algorithm aims to choose centroids that minimise the *inertia*, or within-cluster sum of squared criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_j - \mu_i||^2)$$

Inertia, or the within-cluster sum of squares criterion, can be recognized as a measure of how internally coherent clusters are. It suffers from various drawbacks:

- Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes.
- Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called "curse of dimensionality"). Running a dimensionality reduction algorithm such as PCA prior to k-means clustering can alleviate this problem and speed up the computations.

    - K-means is often referred to as Lloyd's algorithm. In basic terms, the algorithm has three steps. The first step chooses the initial centroids, with the most basic method being to choose k samples from the dataset X. After initialization, K-means consists of

looping between the two other steps. The first step assigns each sample to its nearest centroid. The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid. The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly.

-
- K-means is equivalent to the expectation-maximization algorithm with a small, all-equal, diagonal covariance matrix.
- The algorithm can also be understood through the concept of Voronoi diagrams. First the Voronoi diagram of the points is calculated using the current centroids. Each segment in the Voronoi diagram becomes a separate cluster. Secondly, the centroids are updated to the mean of each segment. The algorithm then repeats this until a stopping criterion is fulfilled. Usually, the algorithm stops when the relative decrease in the objective function between iterations is less than the given tolerance value. This is not the case in this implementation: iteration stops when centroids move less than the tolerance.
- Given enough time, K-means will always converge, however this may be to a local minimum. This is highly dependent on the initialization of the centroids. As a result, the computation is often done several times, with different initializations of the centroids. One method to help address this issue is the k-means++ initialization scheme. This initializes the centroids to be (generally) distant from each other, leading to provably better results than random initialization, as shown in the reference.
- A parameter can be given to allow K-means to be run in parallel, called `n_jobs`. Giving this parameter a positive value uses that many processors (default: 1). A value of -1 uses all available processors, with -2 using one less, and so on. Parallelization generally speeds up computation at the cost of memory (in this case, multiple copies of centroids need to be stored, one for each job).

2.2.6 Definitions

Precision – Recall

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p).

$$P = \frac{T_p}{T_p+F_p}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n).

$$R = \frac{T_p}{T_p+F_n}$$

These quantities are also related to the (F_1) score(**F-Score**), which is defined as the harmonic mean of precision and recall.

$$F1 = 2\frac{P \times R}{P+R}$$

3. METHODOLOGY & IMPLEMENTATION

The methodology used for the implementation of this thesis can be broken down in two parts.

- Model, set, run the experiments on controlled environments and collect the relative data
- Creation of models for classification of the pathogenies in supervised learning.

The main purpose of the implementation is for the created machine learning models to make good use of the collected data that the drivers of the wireless cards can offer at a user level in order to make accurate predictions regarding the detection of the pathologies. In other words, when this software is installed in a conventional device in a home network, this device will be able to understand when one of these five pathologies takes place and in future works even to resolute them in order to increase the network's performance.

IEEE 802.11 Related Pathologies *(as described in Syrigos I., Keranidis S., Korakis T., Dovrolis C. (2015) Enabling Wireless LAN Troubleshooting)*

Here are described the four general pathologies that we detect. They are further broken down into a total of six classes in our experiments. The two metrics that are used for the logging of the statistics for each experiment are:

PDR: The probability that a packet is received correctly

Average Attempts: Packet trains per second.

The *channel modulation* is also logged and is used as a third variable in our model.

Medium Contention

Contention-based pathologies frequently occur in dense WLAN deployments, where multiple 802.11 devices concurrently attempt to

access the medium. However, as the unlicensed spectrum is also exploited by other wireless protocols (e.g. Bluetooth, Zigbee) and a large range of RF devices (e.g. cordless phones, security cameras), the medium is further congested due to non-802.11 emissions. The resulting decrease in available channel access opportunities is directly dependent both on the channel airtime captured by 802.11 transmissions, as well as the transmission Duty Cycle (DC) of non-802.11 RF devices.

The crucial impact of medium contention is clearly highlighted in cases of contending stations that utilize diverse PHY rate configurations, which leads to performance anomaly. The high bitrate stations observe a higher throughput degradation in comparison with the lower rate nodes. This degradation is a result of the low number of channel access attempts due to the high channel airtime utilization by the low bitrate stations. Consequently, we expect the **Average Attempts** metric to decrease across increasing PHY rate configurations of the concerned station. However, regarding the **PDR** metric, higher bitrates should result in higher number of collisions, due to simultaneously expiring back-off timers, and thus to a decrease in FDR, but not in that extent of considering it as a significant trend.

Non-802.11 Medium Contention

In case of non-802.11 contention, devices with fixed transmission DC, such as microwave ovens, can be interpreted as low bitrate stations which do not comply with the 802.11 standard and hence do not perform a "Back off" procedure. As a result, we expect a decrease in **Average Attempts** metric across increasing PHY rates, as it happens in 802.11 contention. Another consequence of the absence of backoff mechanism in non-802.11 devices is that collisions can occur in the middle of a frame transmission and so higher PHY rates will result in lower probability of collisions. Taking that into consideration, we expect an increasing trend in the PDR metric.

Low SNR

Is experienced as a consequence of the low Received Signal strength, resulting from channel fading and shadowing or due to high- Power non-802.11 emissions that result in Noise level increase. Considering that complex modulation schemes require higher link SNR to ensure reliable communication, in comparison with basic schemes, we expect the PDR performance to significantly decrease across increasing PHY rates, under low-SNR conditions. Furthermore, the decrease in PDR would also lead in a decrease in the **Average Attempts** metric.

802.11 Impairments (Hidden Terminal or Capture Effect)

In addition, significant frame delivery inefficiencies may also be attributed to 802.11 impairments, phenomena appearing in cases that concurrent channel access and subsequent frame collisions cannot be avoided through the 802.11 Channel Sensing mechanism. More specifically, the "Hidden-Terminal" anomaly occurs in cases that the receiver node lies within the transmission range of two active 802.11 nodes that are mutually hidden and cannot sense each other resulting in frame collisions. In cases that no remarkable difference is observed in the received signal strength of colliding frames at the intermediate node, the "Hidden-Terminal" phenomenon appears symmetrically for both flows. However, the most frequently observed case is the "Capture-effect" phenomenon, in which case a considerable difference in RSSI values is observed, resulting in a higher probability of successful decoding for the high-power frames. As a result, the link "capturing" the medium experiences lower collision probability accessing the medium more frequently and resulting in higher performance penalty for the affected links.
Longer duration transmissions experiencing higher probability of collision, so we expect to see an PDR increase across increasing PHY rate values of the affected link. However, hidden nodes suggest longer distances from the AP and consequently an underlying low-SNR pathology, so we also expect an PDR decrease in high PHY bitrates. In overall, we should identify a highly varying PDR metric accross PHY

rates and additionally more notable variations under "Capture-effect" scenarios where the impact is more severe. As regards the **Average Attempts** metric, although the underlying low-SNR conditions should impose a decreasing trend, the impact of PDR variation, which as mentioned before is higher under "Capture-effect" scenarios, would enforce **Average Attempts** to not display a clear trend.

3.1 DATA COLLECTION

This is the most basic and important part of the thesis as also the most time consuming. This part's aim is the collection of a great amount of data in controlled experimental environments for each one of the pathogenies.

More specifically conditions will be set for:

- 802.11 Medium Contention
- Non-802.11 Medium Contention
- Low SNR(because of low power or because of high noise)
- 802.11 impairments (Hidden Terminal or Capture Effect)

For the data collection, some automated scripts were made in python and bash that logged the values of our data per time periods(<1s).

The three metrics/variables that were logged and are used for the machine learning models are the:

- Percentage of the packets that were transmitted successfully
- Percentage of time that the channel was occupied
- Packet Rate(modulation)

This stage took place on the experimentation testbed NITOS of University of Thessaly. Nitos structure contains three different environments of experimentation (external, internal, office)

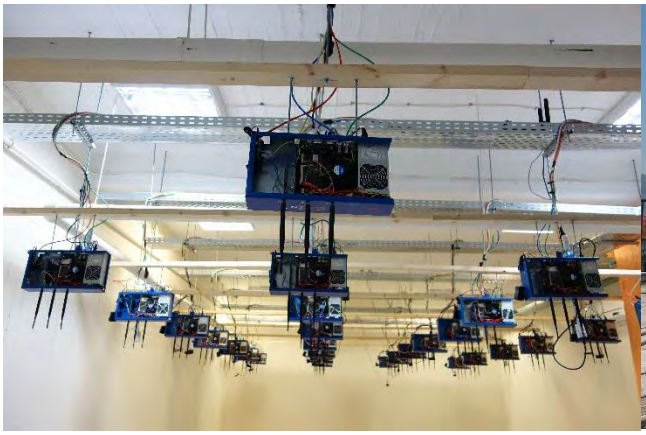

figure 2: Indoor NITOS testbed

figure 3: Outdoor NITOS testbed

which count a total number more of 100 wireless nodes with high specifications. Each node contains two wireless cards that support all versions of 802.11 protocol (ac, n, a/, b) and the two frequency bands (2.4 GHz, 5 GHz).

The data collection was done with the method of active measurement tests where the specific method of **Varying Bitrate Probing** was applied. In this method traffic is generated in the WLAN network with multiple packet trains by using a tool that creates packets(iperf). Every packet train is transmitted in each of the different 802.11 PHY rates. Low rates have a higher probability of successful transmission but also with longer duration which as a result makes the packet train more vulnerable to collisions.

For each pathogeny a thorough experimentation took place in order to be found the topologies that can simulate the pathogeny.

802.11 Medium Contention

Different contention scenarios took place with:

- the existence of varying number of stations in the network (i.e 1,2,4)

- varying number of inserted traffic (i.e 1Mbps, 2Mbps, …, 20 Mbps)
- varying PHY rate of stations (MCS0, MCS1,…MCS9)

Non-802.11 Medium Contention

Different contention scenarios took places with non-802.11 devices such as microwave oven in a distance that could be detectable from the station.

Low SNR

Different Low SNR scenarios took place with:

- Placement of station in varying distances from the access point
- Use of attenuators
- Coexistence of various types of devices(Zigbee, wireless cameras) in a distance that they cannot be detected from the station but they are able to increase its noise floor.

Hidden Terminal

Placement of access point between the station where the collection of statistics takes place and the hidden station in symmetrical distances from both.

Different Hidden Terminal scenarios took place with:

- varying number of inserted traffic (i.e 1Mbps, 2Mbps, …, 20 Mbps)
- varying PHY rate of stations (MCS0, MCS1,…MCS9)

Capture Effect

Placement of access point between the station where the collection of statistics takes place and the hidden station closer to hidden station.

Different Capture effect scenarios took place with:

- varying number of inserted traffic (i.e 1Mbps, 2Mbps, …, 20 Mbps)
- varying PHY rate of stations (MCS0, MCS1,…MCS9)

3.2 IMPLEMENTATION OF MACHINE LEARNING MODELS

In this part, the data that were collected in first part alongside their label (in which pathogeny they correspond) will be fed to the machine learning models.

For the models' implementation was used the popular python libraries scikit-learn, matplotlib and numpy.

The data were modeled in two ways. At first a 3-dimensional model was tested with best algorithm's performance(accuracy,precision,recall) close to 95%. In the second model the data had 16 dimensions. They were of the form [PDR_mcs6, #attemts_mcs6, PDR_mcs9, #attemts_mcs9,… PDR_mcs54, #attemts_mcs54]. That way the best algorithm's performance was over 99%.

The methodology that I followed:

- **Feature Selection** where for the first model as features were selected the ***PDR, Average Attempts and Channel Modulation***. For the second model as features for each sample were selected the: ***PDR_mcs6, #attemts_mcs6, PDR_mcs9, #attemts_mcs9, PDR_mcs12, #attemts_mcs12, PDR_mcs18,***

#attemts_mcs18, PDR_mcs24, #attemts_mcs24, PDR_mcs36, #attemts_mcs48, PDR_mcs54, #attemts_mcs54

- **Label Assignment**: At each instance a label was assigned with a value from 1 to 5 corresponding to the five pathologies for classification.
- **Cross Validation**: The data are shuffled and split to a training and testing set. Usually 80% of the data set is used for the training set and 20% for the test set. In our models were used 80%-20%(training-test) in the first data model and 85%-15% in the second one.
- **Data Fitting**: The training set is fed to the algorithms.
- **Model Testing:** The models are tested by using the algorithms' predict function with parameter the test set. Then we compare the predicted results with the initial test set's labels. (Train set and test set are usually in the form of X, Y where X is a list o features and Y a list of the corresponding labels. i.e. X_train= [[95,92,9], [60,87,6], …], Y_train=[1,3,..])

First Data Model

The Data that were to be fed to the algorithms were 3 dimensional. Each sample consisted of [PDR, #ATTEMPTS, MODULATION].

Samples were collected for a total number of six Classes. These are:

- Low-SNR → one AP and one station in great distances in various topologies.
- 802.11 Contention 4 nodes → two AP and two stations competing in the same channel
- Capture-effect (802.11 impairments) → one AP and two stations in triangular topology. Stations kept symmetrical distances from the AP

- Hidden Terminal (802.11 impairments) → one AP and two stations in triangular topology. Stations kept symmetrical distances from the AP
- Microwave Oven (non-802.11 contention) → one AP, one station and a node where microwave oven simulation software was running.

For each class were collected from 500-1500 samples depending on the difficulty of running the experiment.

Each sample has the form [PDR, #ATTEMPTS, MODULATION].

To begin with, the software reads the data from multiple csv files. Manipulates them until they are in the form of [PDR, #ATTEMPTS, MODULATION, LABEL] where label is one of the six classes (i.e Hidden Terminal). After the cross validation, our data are shuffled and split into a training set and a test set. The training set is used to train the classifiers and the test set is used afterwards for checking the performance of the classifiers.

Second Data Model

The same samples as in the first model from the same scenarios and topologies were collected and were transformed in a vector with 16 values. Each sample has the form [PDR_mcs6, #attemts_mcs6, PDR_mcs9, #attempts_mcs9, …, PDR_mcs54, #attemts_mcs54].

That means that each instance is characterized by eight tuples of (PDR, attempts), one for each modulation. This is a more specific model in comparison with the first one so we expect better results.

The algorithms that were used for the models are:

Random Forest Classifier

```
clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X_train, y_train)
rf_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test,rf_pred)
print "Random Forest accuracy is: ",accuracy
```

Decision trees classifier

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy = accuracy_score(y_test,pred)
print "Decision trees accuracy is: ",accuracy
```

Naive Bayes

```
clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy = accuracy_score(y_test,pred)
print "Naive Bayes accuracy is: ",accuracy
```

SVM Classifier

```
clf = svm.SVC(kernel= 'rbf')#'linear')
clf.fit(X_train, y_train)
svm_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test,svm_pred)
print "SVC linear kernel accuracy is: ",accuracy
```

3.3 Experimental Results

In this section, the results of the models will be presented.

First Data Model

In the figures that follow we can see the collected instances labeled by color. **PDR** is represented by the x-axis and **Number of Attempts** is represented by y-axis. There is also a third dimension for each instance which represents the modulation but is not visible in these graphs.

First the training and testing sets in figures and short descriptions will be presented followed by graphical representation in the form of scatter plots of the algorithms predictions given the test set. Lastly the tables with the concentrated results will be shown.

Initial training set

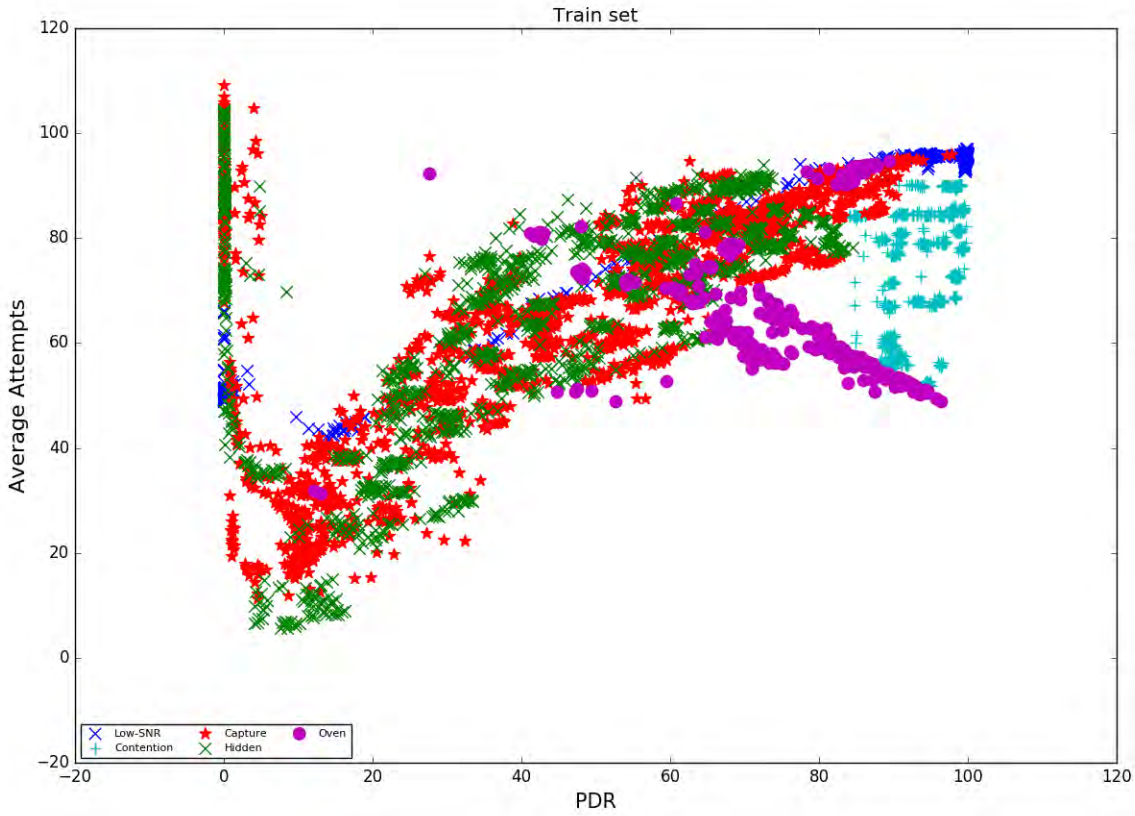

figure 4. training set representation

Figure 4 shows a graphical representation of the training set used in our first data model which constitutes of ***5348 instances.***

In the two observable dimensions of figure 4 we can see that Low SNR, MW Oven and Contention class data are more concentrated in specific areas of the 2-dimensional space and are more uniform, whereas Hidden Terminal and Capture Effect are sparser and less homogeneous.

A look at the Test Set

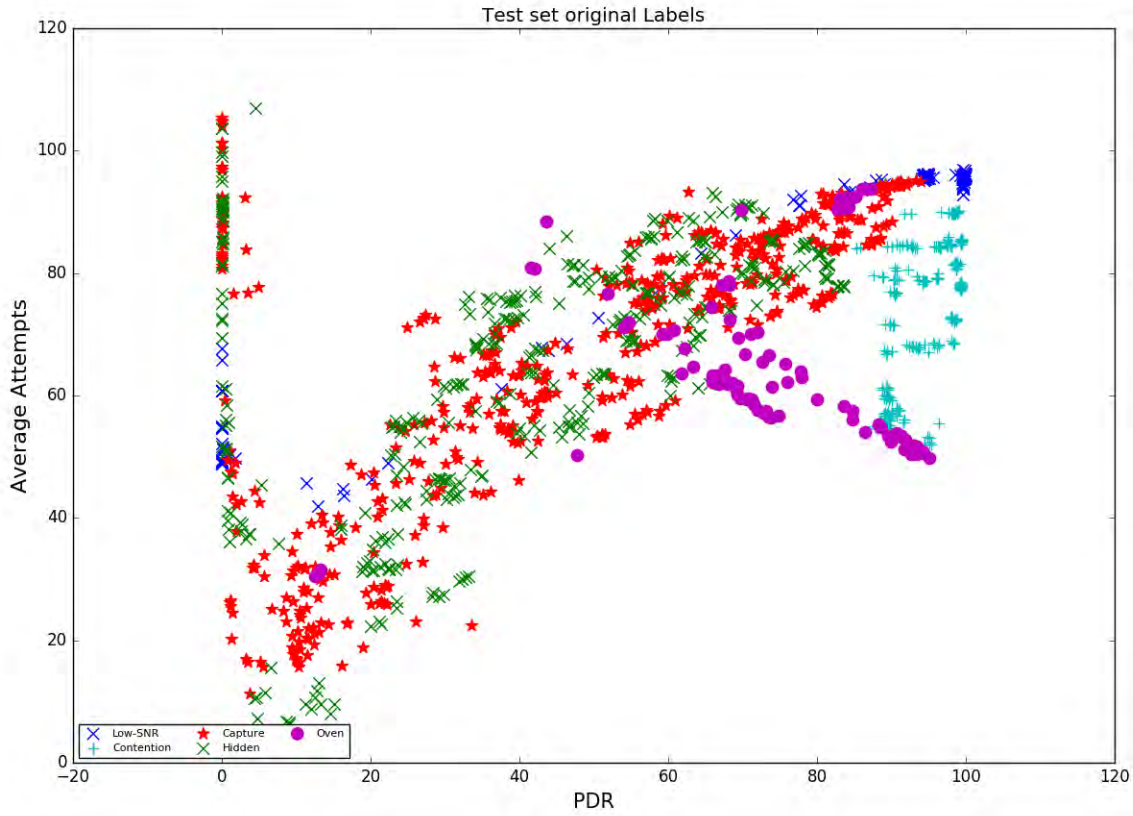

figure 5. Test set

In figure 5 is shown the test set that is used for testing the performance of the algorithms in terms of accuracy, precision and recall. It is used to evaluate the results of the algorithms' classification performance (prediction set). It constitutes of 1338 samples which is 20% of the total data set.

Classification of Random Forests

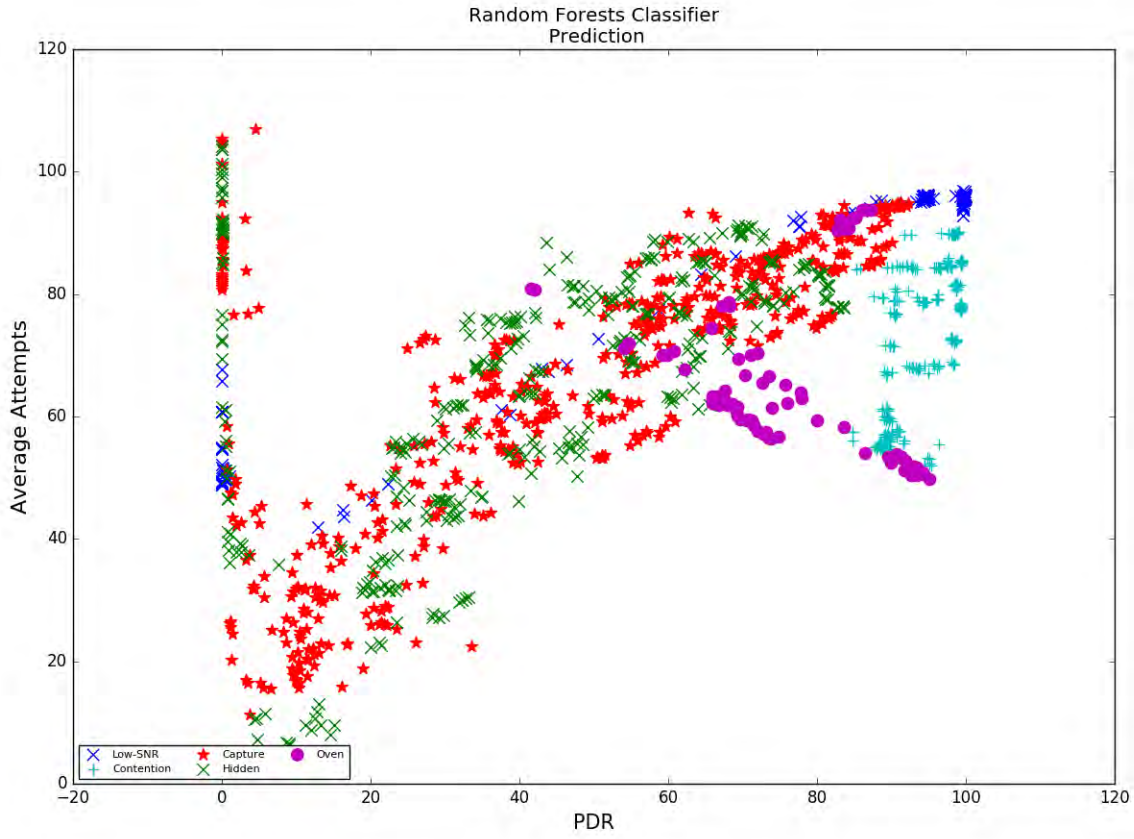

Figure 6. Random Forests Prediction

In figure 6 the prediction results of the Random Forest classifier are shown. After the classifier was fed with the training set its performance was tested on the testing set (figure 5). It resulted in an accuracy of 95% and precision of 0.9499.

Random Forest confusion matrixes

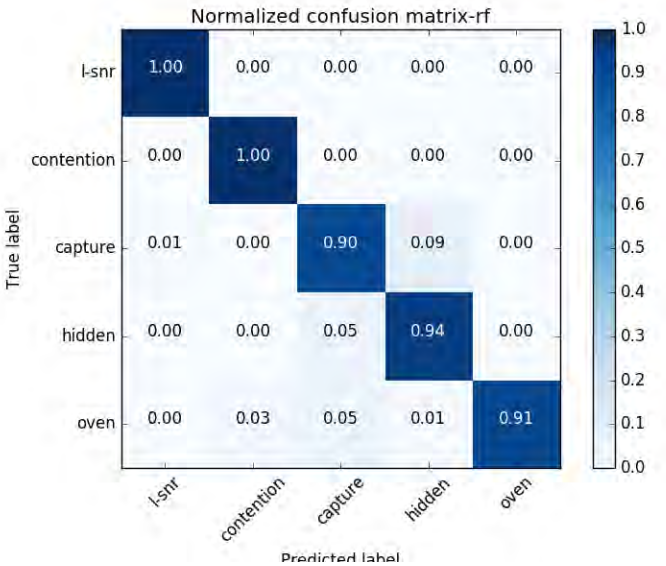

Figure 7. normalized RF confusion matrix

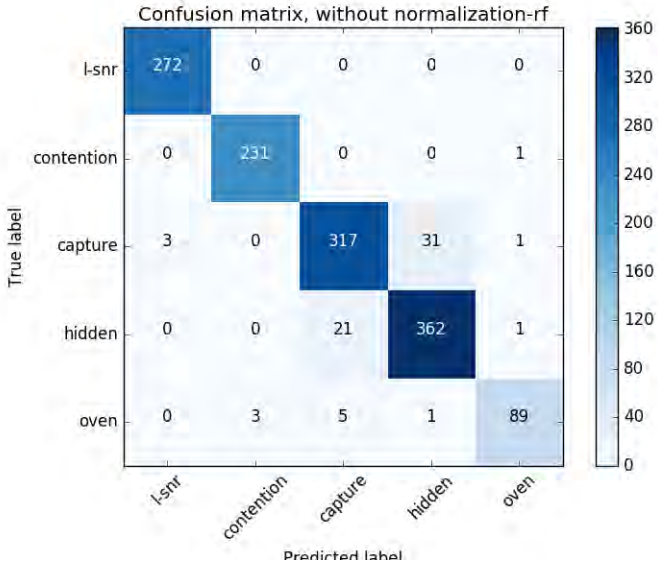

Figure 8. without normalization RF confusion matrix

In figures 7 and 8 the confusion between classes and also the reason of the predicted accuracy can be observed. We expected to see a lower precision and accuracy in the 2 middle classes (capture, hidden) because of the sparsity of their samples and the close correlation between them.

Classification of Decision Tree

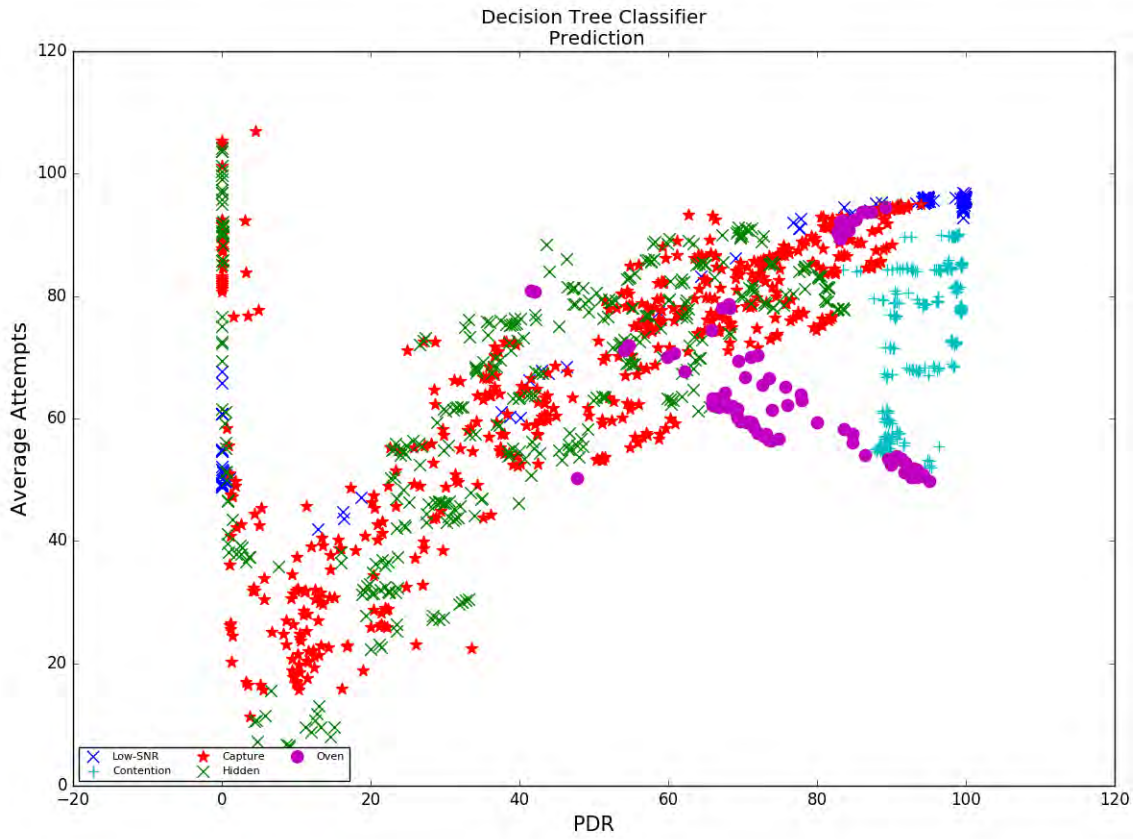

Figure 9. Decision Tree prediction

In figure 9 the prediction results of the Decision Tree classifier on the test set (figure 5) are shown. It resulted in an accuracy of 93% and precision of 0.9325. It is a simple yet powerful algorithm.

Naïve Bayes classification

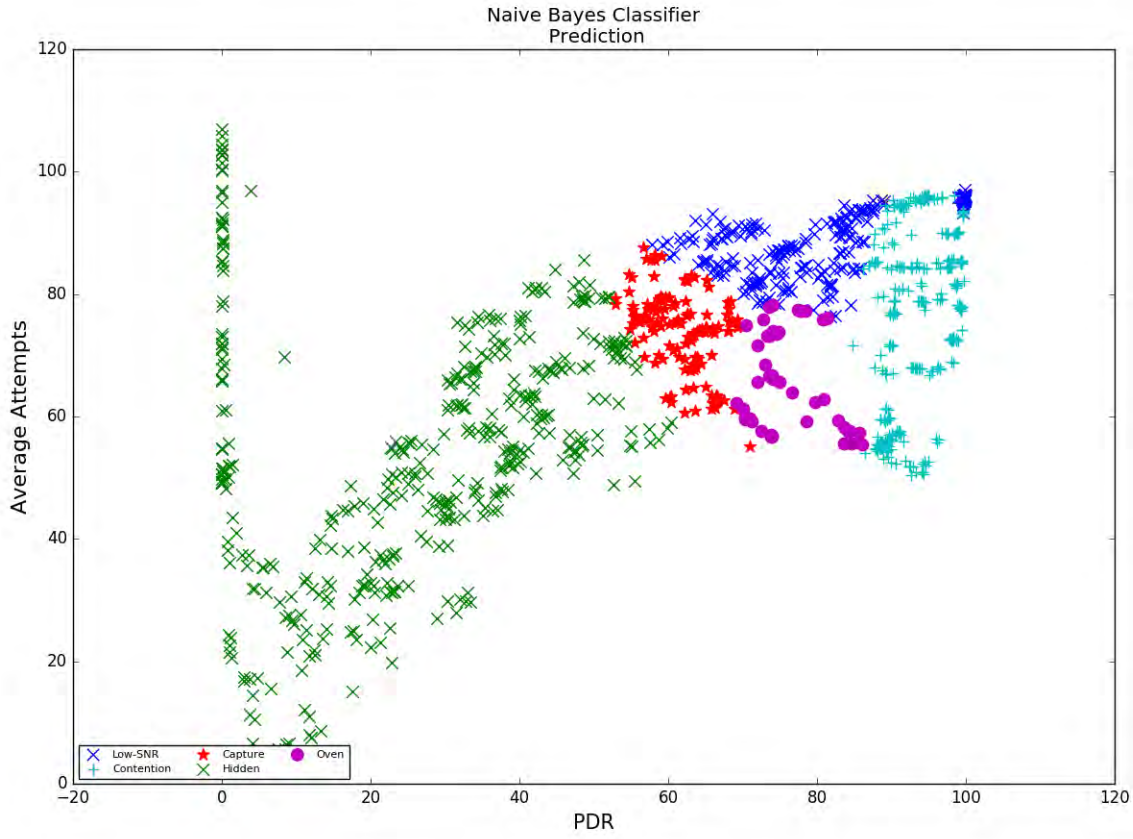

Figure 10. Naïve Bayes prediction

In figure 10 are shown the bad classification results from Naïve Bayes. These were expected because of the non-independent data.

Support Vector Machines classifier

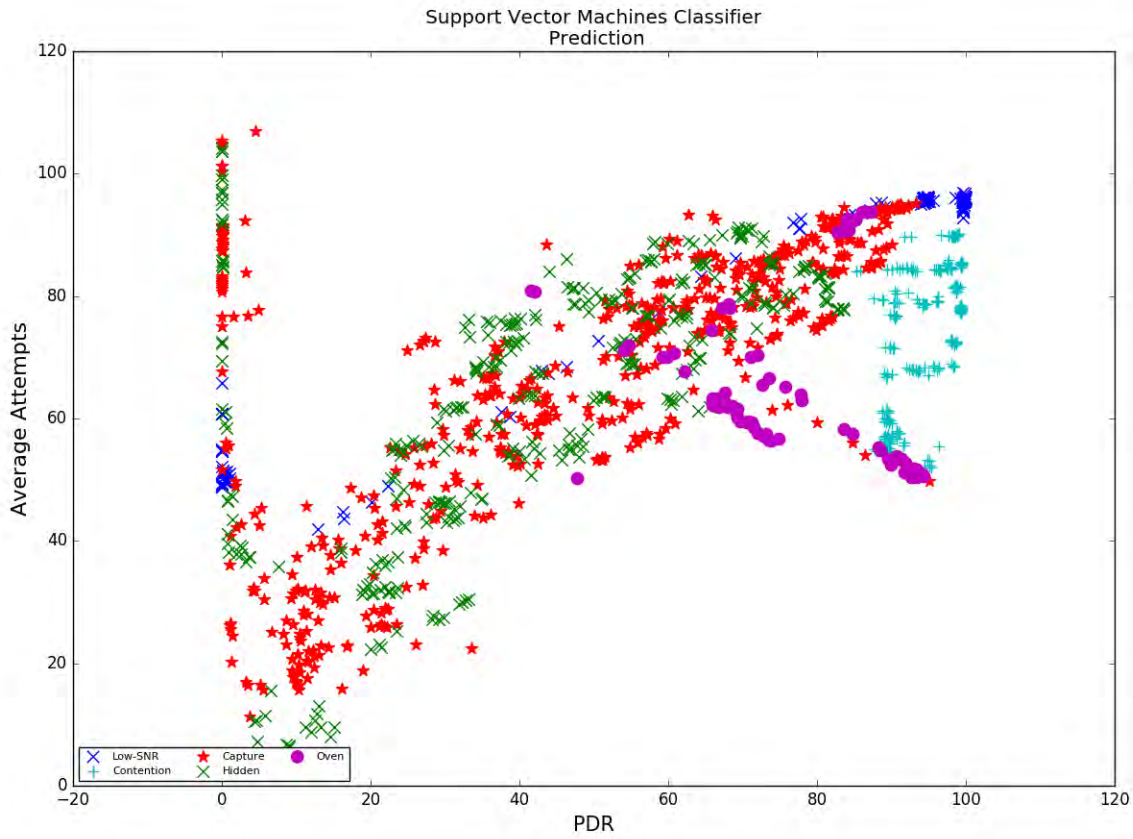

Figure 11. SVM prediction

In figure 9 the prediction results of the Support Vector Machines classifier on the test set are shown. It resulted in an accuracy of 94% and precision of 0.9469. SVM is a classifier that performs well on most classification problems if configured correctly. In our model the rbf kernel is used with the default gamma and c parameter values (gamma: default='auto' meaning 1/n features will be used, c: (default=1)). Its main drawback is its cost in training time($O(n^2)$ which becomes $O(n^3)$ as the number of features increments where n is the number of samples).

Support Vector Machines Confusion matrixes

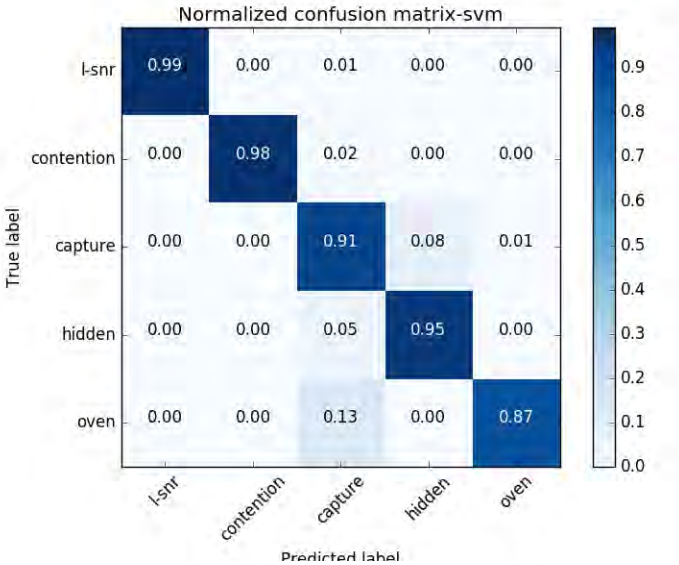

Figure 12. SVM normalized confusion matrix

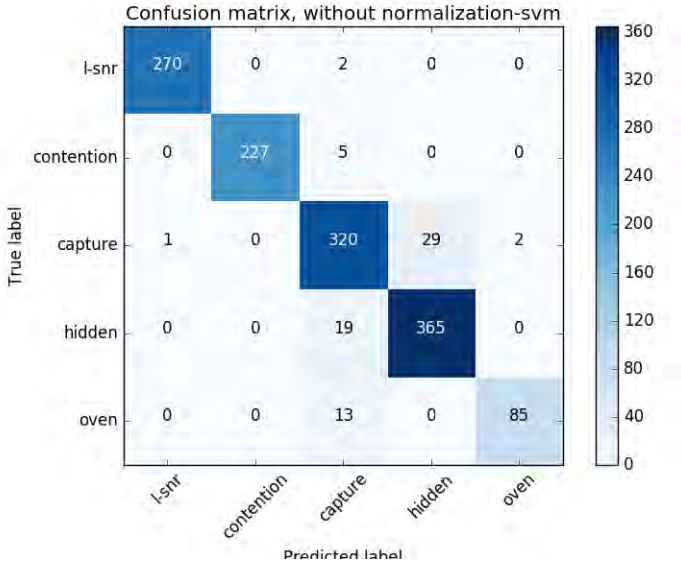

Figure 13. SVM without normalization confusion matrix

SVM's precision and accuracy is pretty close to Random Forest's but it takes six times as long to train. This is due to SVM's time complexity of $O(n^2)$ in comparison with RF's $O(v * n \log(n))$ where v is the number of variables and n is the number of instances. Also, from the

matrix, one can make the hypotheses that capture effect and hidden terminal are closely related. Which is true as hidden terminal and capture effect being two very similar pathologies that occur for similar reasons and are derived from the same class of pathologies as shown in figure 1 and stated in "Syrigos I., Keranidis S., Korakis T., Dovrolis C. (2015) Enabling Wireless LAN Troubleshooting. In: Mirkovic J., Liu Y. (eds) Passive and Active Measurement".

Performance

Algorithm Name	Accuracy	Precision	Recall	F-score	Time elapsed (sec)
Random Forest	0.9499	0.9499	0.9499	0.9497	0.7345
Decision Tree	0.9327	0.9325	0.9327	0.9325	0.0739
Support Vector Machines	0.9469	0.9478	0.9469	0.947	4.472
Naïve Bayes	0.5523	0.5252	0.5523	0.5149	0.0159

The **number of samples** for training set and test set respectively are:

Number of instances	Low-SNR	802.11 Contention	Capture Effect	Hidden Terminal	MW Oven
Train set	**1087**	**928**	**1408**	**1535**	**390**
Test set	**272**	**232**	**352**	**384**	**98**

The total number of instances for the training set of this model is 5348.

The total number of instances for the test set of this model is 1338.

Second Data Model

The Data that were fed to the algorithms are 16-dimensional. The same number of samples as in the first model were collected and were transformed in a vector with 16 values. Each sample has the form [PDR_mcs6, #attemts_mcs6, PDR_mcs9, #attempts_mcs9, …, PDR_mcs54, #attemts_mcs54].

That means that each instance is characterized by eight tuples of (PDR, attempts), one for each modulation. This is a more specific model in comparison with the first one so we expect better results.

The **number of samples** per class are:

	Low-SNR	802.11 Contention	Capture Effect	H. Terminal	MW Oven
#Instances	169	145	240	240	061

The total number of instances that could be collected in the context of this thesis is **855**.

Although it is a small number, our model's performance was great with its best algorithm (Random Forest) exceeding 98% accuracy. Because of the peculiarity of our instances that are in the form of a 16-value vector, the data set and the classification predictions cannot be represented with graphs.

A table with the algorithms' performance follows and the confusion matrixes are presented in the next pages.

Performance

Algorithm Name	Accuracy	Precision	Recall	F-score	Time elapsed (sec)
Random Forest	0.9844	0.9848	0.9844	0.9845	0.16299
Decision Tree	0.9767	0.9769	0.9767	0.9767	0.04399
Support Vector Machines	0.9457	0.9455	0.9457	0.9455	0.9879

In the next 3 figures we can see the confusion matrixes for Random Forests, SVM and Decision Tree algorithms.

Decision Tree

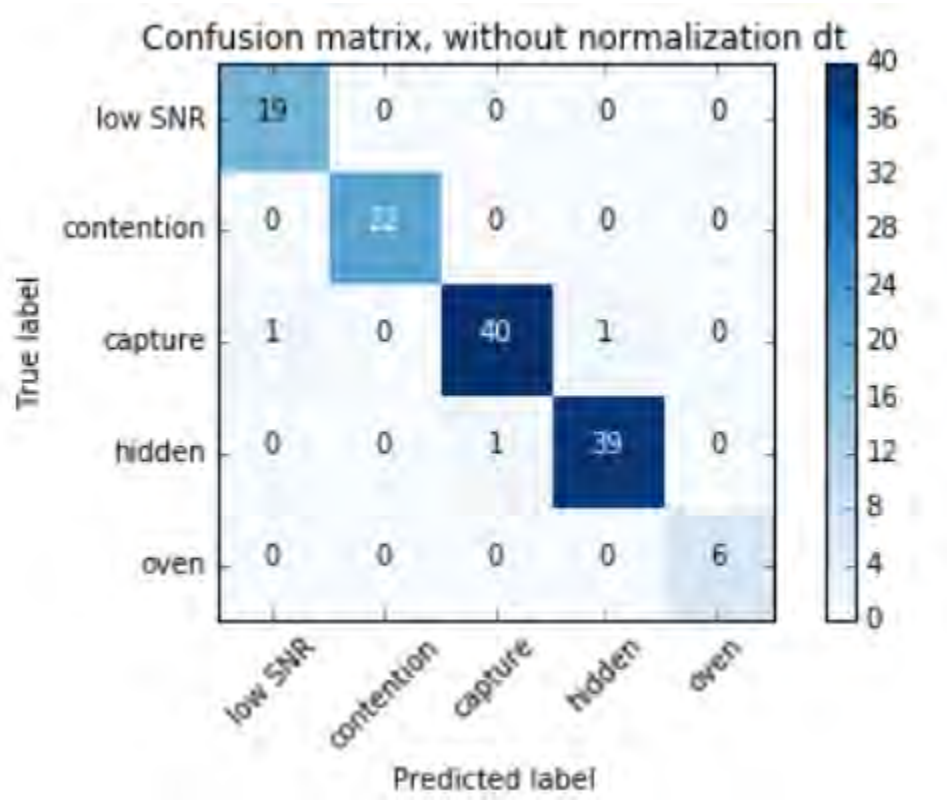

Figure 14. confusion matrix 2nd model DT

Although Decision Tree is a simple algorithm, it produced significant results with an accuracy of ~97% and a very small training time cost. Its speed is entirely down to its complexity of $O(n_{features}\ n_{samples}\ \log(n_{samples}))$ where number of features is 16 and the number of samples is 855. As is shown in figure 14 it has a pretty good confusion matrix by mislabeling only 3 instances.

Support Vector Machines

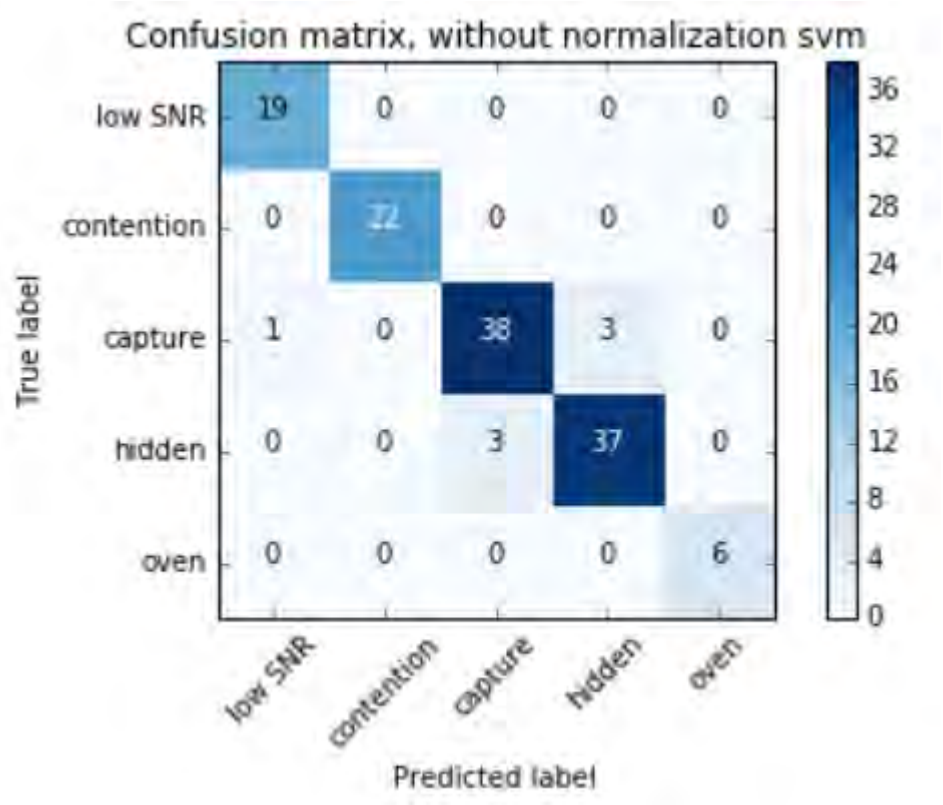

Figure 15. Confusion matrix 2nd model SVM

Support Vector Machines performance in this model is the worst of the three tested algorithms with an accuracy of 94% and a training cost 6 times greater than Random Forest's and 24 times greater than Decision Tree. This is down to the fact that as the number of features gets bigger SVM's time complexity goes from $O(n^2)$ to $O(n^3)$. In this second data model, we have a pretty small number of samples which could lead to a very small computation time but we also have a pretty big number of features that leads to slow SVM training speed. In figure 15 we can see that SVM tends to confuse capture effect with hidden terminal mislabeling 6 instances. It also made a false prediction of 1 instance of capture effect class with low SNR.

SVM on this model is configured with a linear kernel which means that it uses a linear function to create the decision boundaries in

contrast with the first model where rbf kernel was the optimum choice(by experimentation).

Random Forest

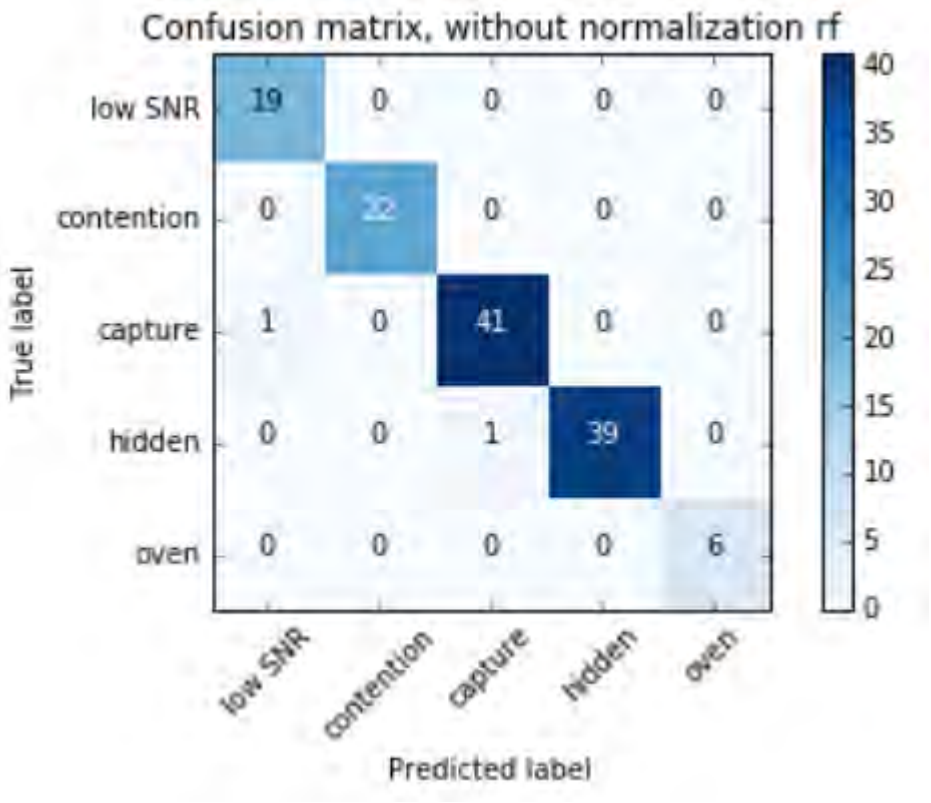

Figure 16. confusion matrix 2nd model RF

Random Forest dominates in this data model too. Its accuracy is close to 99% with a precision of 0.98 and a training time of 0.162 seconds. These results are exceptional if we consider the very small number of instances that were used as training set.

To summarize, in the confusion matrixes of this data model we can see less missclassification in comparison with the first model. Although training instances are fewer, they contain more information and that is the reason of the increased accuracy and precision.

As a bonus, an **unsupervised learning** technique was used. Its purpose was to possibly identify new classes and information in our data.

Figure 17 shows the **K-Means Clustering** algorithm's results for 6 classes. Keep in mind our data are 3-dimensional and we only plot the two dimensions (PDR, #Attempts). The hidden dimension is the channel modulation.

The clustering run on the whole dataset. Although it may have some hidden information, no meaningful observations were made.

Its computation time was 0.513999938965 seconds.

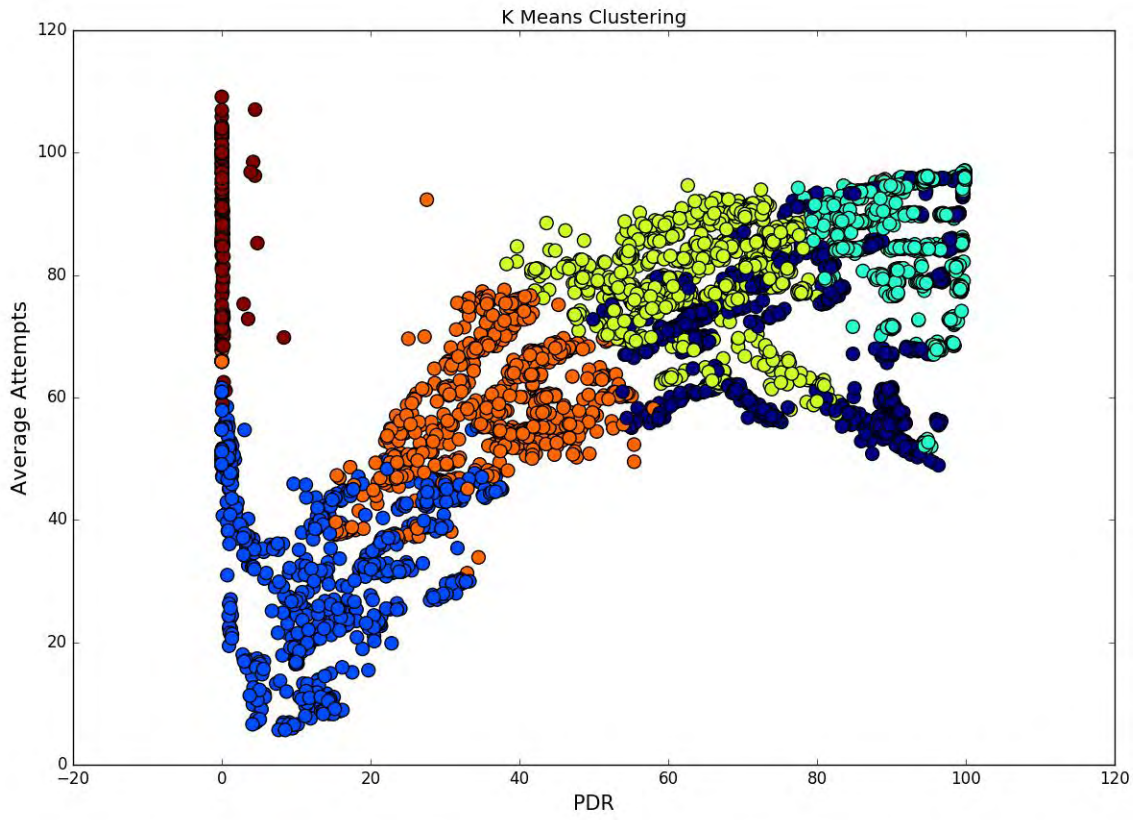

Figure 17. K Means Clustering on the whole dataset

4. SUMMARY

4.1 Conclusions

To summarize, we have seen that in the **First Model** is possible with a relatively good accuracy to automatically diagnose one of these pathologies given that the model-software has been trained on 3-dimensional labeled Data. The approach using the Random Forest algorithm gave the highest performance. The models that were pretty close to Random Forest's accuracy were SVM and Decision Tree. Random Forest and Decision Tree models are closely-related as the first is a more complex form of the second. Their cost in training time was relatively small and their performance in terms of accuracy, precision and recall was expected because of the structure of our data that could be modeled like a decision tree when looking them in 3 dimensions. The **Second Model** gave exceptional results but this was expected because of the longer vector (more features) that was used in the training set.

4.2 Future Enhancements

As future enhancement, more statistics can be collected from wireless card drivers in order to further increase model's performance. Moreover, a mobile application could be created that will be used for testing in populated urban arias. Then the collected statistics can be gathered in a central database in order to be feed into machine learning model (Deep Neural Networks?) for increasing prediction accuracy in devices that do not operate in controlled environments (NITOS testbed). Finally, this software can be deployed and tested in commercial devices in uncontrolled environments.

REREFERENCES

Syrigos I., Keranidis S., Korakis T., Dovrolis C. (2015) Enabling Wireless LAN Troubleshooting. In: Mirkovic J., Liu Y. (eds) Passive

and Active Measurement. PAM 2015. Lecture Notes in Computer Science, vol 8995. Springer, Cham

Feilu, L., Jian, L., Zhifeng, T., Korakis, T., Erkip, E., Panwar, S.: The hidden cost of hidden terminals. In: Proceedings of ICC (2010)

Lee, J., Kim, W., Lee, S., Jo, D., Ryu, J., Kwon, T., Choi, Y.: An experimental study on the capture effect in 802.11a networks. In: Proceedings of WinTECH (2007)

Rayanchu, S., Mishra, A., Agrawal, D., Saha, S., Banerjee, S.: Diagnosing wireless packet losses in 802.11: separating collision from weak signal. In: Proceedings of INFOCOM (2008)

Lakshminarayanan, K., Seshan, S., Steenkiste, P.: Understanding 802.11 performance in heterogeneous environments. In: Proceedings of HomeNets (2011)

Kanuparthy, P., Dovrolis, C., Papagiannaki, K., Seshan, S., Steenkiste, P.: Can user-level probing detect and diagnose common home-WLAN pathologies. SIGCOMM CCR **42**(1), 7–15 (2012)

Cheng, Y., Bellardo, J., Benkö, P., Snoeren, A., Voelker, G., Savage, S.: Jigsaw: solving the puzzle of enterprise 802.11 analysis. In: Proceedings of SIGCOMM (2006)

Jangeun, J., Peddabachagari, P., Sichitiu, M.: Theoretical maximum throughput of IEEE 802.11 and its applications. In: Proceedings of NCA (2003)

Giustiniano, D., Malone, D., Leith, D., Papagiannaki, K.: Measuring transmission opportunities in 802.11 links. IEEE/ACM Trans. Netw. **18**(5), 1516–1529 (2010)

http://scikit-learn.org/stable/