Naive Bayes Classifier-Based Fire Detection Using Smartphone Sensors

by

Mohammad Mahdi Mahdavi Amjad

Supervisors:

Professor Ole-Christoffer Granmo

Dr. Jaziar Radianti

Terje Gjøsæter

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree Master of Science in Information and Communication Technology

University of Agder

Faculty of Engineering and Science

Department of Information and Communication Technology (ICT)

Grimstad, Norway

June 2014

Dedicated to my beloved wife, Maryam.

UNIVERSITY OF AGDER

Abstract

Faculty of Engineering and Science
Department of Information and Communication Technology (ICT)

Master's Thesis

Naive Bayes Classifier-Based Fire Detection Using Smartphone Sensors

by Mohammad Mahdi Mahdavi Amjad

For many years, smoke detectors have been used as the most crucial fire detection sensors. Although smoke detectors do their job very well, they are not perfect and may cause false or late alarms. This is because they only rely on one of the fire signs which is smoke. Fire has many other signs as well such as heat and light. It also affects its environmental parameters such as temperature and humidity. But typically, buildings are not equipped with sensors capable of sensing these changes. Recently, a few smartphone manufacturers have added temperature, humidity, and barometer sensors to their products which can be used for more reliable fire detection. In this thesis, a framework composed of one or more smartphones and a back-end server is proposed which can detect and visualize indoor fire. For this purpose, the smartphones continuously collect, preprocess, and analyze data from their sensors to detect if fire exists in their surroundings. The back-end server facilitates the analysis processes in smartphones and provides crisis management institutions such as police, fire department, and ambulance with real-time monitoring user interface so that they can easily grasp useful information about the fire's location and scale. The proposed fire detection framework is a learning system which needs to be trained by real data. Therefore, a wide range of experiments is precisely designed and performed to make sure that the system can immediately and accurately detect fire in diverse environmental conditions.

Keywords: Naive Bayes Classifier, Smartphone, Sensor, Fire Detection.

Acknowledgements

This thesis was submitted in partial fulfillment of the requirements for the degree Master of Science in Information and Communication Technology (ICT) at the University of Agder where the workload is set to a total of 30 ECTS credits. The work has been done in the period of January to June 2014.

I would like to express my sincere gratitude to my supervisor, Prof. Ole-Christoffer Granmo, who generously shared his novel ideas with me.

I am grateful to my co-supervisor, Dr. Jaziar Radianti for her valuable discussions and kind support.

I thank my co-supervisor, Mr. Terje Gjøsæter for his accurate and precise comments.

I thank Mr. Åsmund Næss and his colleagues at Safemar AS for supporting my project and letting me to perform exciting fire experiments in their training area.

Last but not least, I thank Mr. Otto Jensen and his colleagues at Grimstad brannstasjon (Fire Station) for performing a very difficult fire operation exclusively for this project and sharing precious information with me about real fire behavior.

Mohammad Mahdi Mahdavi Amjad

Grimstad, Norway, June 2014

Contents

List of Figures

List of Tables

Abbreviations

AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
FDBS	**F**ire **D**etection **B**ack-end **S**erver
HTML	**H**yper **T**ext **M**arkup **L**anguage
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
IIS	**I**nternet **I**nformation **S**ervices
IP	**I**nternet **P**rotocol
ITU	**I**nternationl **T**elecommunication **U**nion
JSON	**J**ava **S**cript **O**bject **N**otation
MAP	**M**aximum **A** **P**osteriori
PDF	**P**robability **D**ensity **F**unction
RDBMS	**R**elational **D**ata **B**ase **M**anagement **S**ystem
SMS	**S**hort **M**essage **S**ervice
SOA	**S**ervice **O**riented **A**rchitecture
SQL	**S**tructured **Q**uery **L**anguage
XML	e**X**tensible **M**arkup **L**anguage

Chapter 1

Introduction

According to the International Telecommunication Union (ITU), there was about six billion mobile-cellular subscriptions in 2011 [1]. Smartphones are also increasingly getting popular and only in 2013 about one billion smartphones were sold. Smartphones are not just used for making and receiving voice calls or sending text messages. They are more like small computers with huge processing power and memories. They are capable of connecting to various types of networks such as wireless, High-Speed Downlink Packet Access (HSDPA), and Long-Term Evolution (LTE) and are also equipped with high quality sensors such as front and back cameras, Global Positioning System (GPS), proximity, light, accelerometer, gyroscope, etc. Newer smartphones have even more advanced temperature, barometer, and humidity sensors.

There are many ongoing projects concentrating on the data captured from smartphones. They typically use these data for a wide range of analyses in different areas such as national security, health, and marketing [2][3]. Additionally, smartphones data can be used for monitoring the surroundings of the device and consequently of the person carrying it. For instance, it is possible to collect the temperature data perceived by a smartphone to detect the temperature of the environment in which the smartphone resides.

In this project, I propose a framework for using smartphones sensor data to detect fire and also real-time monitoring the smartphones environment in a back-end server. My approach focuses on indoor fire detection and the outdoor fire detection is left as future work. Nevertheless, the principles of indoor and outdoor fire detection are more or less the same, except that perhaps the outdoor sensors must meet extra requirements such as water and dust resistance. In fact, the proposed method in this thesis can be extended

for detecting much more phenomena, and from this perspective it is a general purpose approach.

But why do we need smartphones for fire detection while buildings have fire (mostly smoke) detection systems? The point is, a considerable percentage of conventional smoke detectors are dead; either malfunctioning or out of battery. They also trigger many false alarms and annoyingly beep when they are running on low battery. Moreover, the majority of smoke detectors are only capable of triggering an alarm, while smartphones can make calls, send messages, enable crisis management institutions to monitor the environment, or even propose some suggestions to their owners about the nearest exit or other necessary information in case of emergency.

The framework proposed in this thesis is designed in relation to the SmartRescue project where the developers work on a system which uses smartphone sensors data for real-time threat assessment. Additionally, in the SmartRescue project the system should be able to generate evacuation plans in an immediate manner.

1.1 Problem Statement

This thesis mainly discusses the following problems:

Sensor Selection: First of all, we need to decide on the sensors which can be used in a fire detection process. This depends on two main factors: sensors available in smartphones, and environmental parameters that change in case of having indoor fire. Obviously, one of the most important signs of fire is smoke, but at the moment, smartphones do not have smoke sensors. Therefore, we need to find a way to compensate for the absence of smoke senors in our system.

Reasoning: At the first glance, it may seem that by using the temperature sensor of a smartphone, it is quite easy to detect fire and for this purpose, it is enough to observe the temperature perceived by the sensor and to trigger an alarm if it exceeds a certain threshold. According to my preliminary experiments, there are many scenarios in which the smartphone sensors report unrealistic values. For instance, direct sunlight, heaters, and fireplaces can strongly affect the temperature perceived by smartphone. Other sensors such as light and humidity have similar sources of error. On the other hand, reasoning based on thresholds causes two issue: high thresholds result in late alarm triggering and low thresholds cause many false alarms.

Real-Time Monitoring: The majority of existing frameworks which use smartphone sensors, cannot perform reasoning and monitoring simultaneously. Thus, some of them lack remote monitoring and others can only analyze sensor data in their back-end servers. I believe a good fire detection framework must be able to work in real-time and provide the remote monitoring interface while providing its users with the reasoning results.

Firefighting: According to our interviews with firefighters, when they arrive to a burning place they do not have any idea about the fire phase. This can be deadly to them since the fire behavior depends on the phase it is in. Currently, fire fighters use their experiences to guess the fire phase and infer based on the smoke color, the burning sound, etc. Some fire fighting groups have Infrared (IR) cameras to detect the temperature of the burning place. Nevertheless, entering a burning building is always quite dangerous for them mostly because they do not know what is going on inside. Existing fire detection systems can only show them the room in which there is some smoke and no more.

Considering the above mentioned issues, lack of central visualization in existing fire detection systems and their inability in inference on existence of fire based on diverse sensor measurements are the most important problems addressed in this thesis.

1.2 Problem Solution

The main components of FireDetection, the framework proposed in this thesis, and their capabilities are as follows:

Smartphones: My experiments show that indoor fire quickly affects the room temperature, humidity, and air pressure. It also has some impact on light since normally, fire causes smoke and it may reduce the room light. Light sensor already exists in the majority of smartphones, even budget ones. Temperature, humidity, and barometer sensors are also available in limited number of high-end smartphones. Therefore, we have sufficient hardware resources available now which can be efficiently used for fire detection. For this purpose, an Android application is developed that collects values from temperature, humidity, light, and pressure sensors and using a modified implementation of Naive Bayes Classifier, analyzes them and calculates the risk of triggering an alarm. Afterwards, according to the analysis results the application decides to trigger the alarm. This application also continuously sends the sensors data and analysis results along with the location coordinates to a back-end server over the Internet. The reasoning process is independently done inside the smartphones, i.e. smartphones does not need to be connected to any other machine while reasoning.

FireDetection Back-End Server (FDBS): Sensors data are sent to this server to be logged (although each smartphone has its own log) and visualized. This feature enables remote monitoring which lets crisis management institutions such as police, fire department, hospitals, and ambulance to monitor fire location and scale in real-time. FDBS, also performs some heavy calculations and produces the required results for Naive Bayes Classification. The smartphones can access these results by invoking a web service in FDBS; once they do, there is no more need to be connected to the Internet for performing reasoning, although the remote monitoring feature still is dependent on the Internet connectivity. For remote monitoring, a web application is designed that plots the fire location, its seriousness, and all values captured from smartphones on a map. This map can be very helpful to firefighters.

Experiments: The Naive Bayes Classifier implemented in this framework is a supervised learning system which needs to be trained in order to be able to detect fire. This forces us to study the behavior of fire in terms of its impacts on environmental parameters. Hence, diverse experiments are designed and performed to train the reasoner module of the framework and to test the accuracy of its reasoning process. Approximately 70 percent of the experiment results are used for training the reasoner and the rest is used for performance evaluation.

1.3 Key Assumptions

In reality, environmental conditions can affect the perception of smartphone sensors. Detecting if the smartphone resides in normal condition or not, needs more consideration and is out of the scope of this thesis. Hence, during the reasoning process, it is assumed that the smartphone is placed indoor, its sensors are functioning correctly, and the generated data are valid. Moreover, the proposed framework is trained in southern Norway and its functionality in other climate conditions is not guaranteed.

1.4 Importance of Topic

Smartphones are getting more popular, and nowadays it is almost impossible to find a family in which no one owns one of them. According to Gartner, only in 2013 about one billion smartphones have been sold in the world so that for the first time in history of mobile phones, the number of smartphones sold was more than the number of feature phones sold [4]. This is an opportunity for us to benefit from the smartphones' features

in order to collect accurate data about our environment. Utilizing smartphone sensor data, we can monitor our environment and manage catastrophes in a quicker and more efficient way. This is the main aim of the framework proposed in this thesis. It can play a significant role in future catastrophes detection and crisis management. Likewise, it can be a good basis for systems generating evacuation plans.

1.5 Thesis Outline

Chapter 1 focuses on the general properties of the project such as the problem statement, problem solution at a glance, key assumptions, and the importance of the topic. In Chapter 2, a number of research projects related to various areas of this thesis are discussed and their strengths and weaknesses are studied. The theoretical backgrounds and the core concepts that my solution is based on, can be found in Chapter 3. Chapter 4 concentrates on the details of the experiments performed in order to understand indoor fire behavior and also the solution proposed in this thesis. In Chapter 5, different components of the prototype framework and its implementation details are explained precisely. The result gained by the FireDetection framework are described in Chapter 6. Finally in Chapter 7, the highlights of the results gained are summarized and a few possible improvements for the proposed framework are brought up.

1.6 Chapter Summary

The aim of this chapter was to define the research question and briefly bring up the axioms of the proposed solution. The chapter also showed how such a framework can be important considering the remarkable growth of smartphones popularity in the world.

Chapter 2

Related Work

As mentioned in the Introduction chapter, humidity and temperature sensors are rather new in smartphones but these two parameters along with the smoke are the most crucial parameters in fire detection. Of course smartphones are still not equipped with smoke detection sensors, hence, at the moment we have to use other sensors for fire detection. To the best of my knowledge, there is no smartphones-based fire detection systems in the market or as a published research project. Nevertheless, utilizing smartphone sensor data has been extensively scrutinized in a wide range of activity recognition, event detection, and context awareness frameworks. For instance, there are a huge number of scientific papers discussing methods for detecting the physical condition of the smartphone and consequently its carrier's body.

Typically, these framework more rely on motion and position sensors data to detect the orientation and the movement pattern of the smartphone. The inference or reasoning process is then done through a variety of classification and pattern recognition techniques. Although the focus of these researches is not hundred percent the same as mine, the principles of the two approaches are more or less similar. So, I think these sort of research are very relevant to what I do in this project. In the following sections, I bring up some related prior researches in two areas: utilizing smartphone sensors data and the techniques used for analyzing smartphones data.

2.1 Utilizing Smartphone Sensor Data

Pascu et al. [5] propose a framework which concentrates on capturing the articulated motions of hand gestures. To do this, they strap three smartphones on the person's

arm, forearm, and hand which send a combination of magnetometer, accelerometer, and gyroscope sensor values to a back-end server to be processes and visualized. One of the most remarkable features of this framework is its ability to work in real-time meaning that immediately after the person moves his/her arm, the information about the movement is sent to the back-end server and is shown by an application installed on it. Working in real-time is missing in many similar frameworks [6][7]; they typically store the smartphone sensor data in the device memory and then transfer it to the analyzer machine later on. Consequently, real-time monitoring will not be possible.

In this framework, the sensor data is sent to the back-end server in the form of Java Server Object Notation (JSON) [8] messages but before sending, data packages need to be prepared. Hence, data preparation is a very important phase in this framework. Data preparation is composed of noise reduction, and sensors data composition. The main shortage in this framework is reasoning. Although it is possible to observe the hand motion in real-time, the system cannot determine the type of the gesture. In fact, the whole system is focused on data acquisition phase and has nothing to do with the analytic part.

Shin et al. [9], propose an Android application which can change the layout of the smartphone's homepage icons and even highlight the ones that most likely the user wants to press in the next use. As features, they collect data from a wide range of sensors such as GPS, accelerometer, and illumination. Some Android system parameters such as time, battery level, wireless network and Bluetooth status, and call/SMS events are also used as other features. In this application, the inference process is performed by using Naive Bayes Classifier.

CoenoFire is another smartphone-based system introduced by Feese et al. [10] in which an Android application collects a wide range of information about a firefighting mission. Firefighters must place the smartphone in their suits when going to missions so that it can collect the required information. These information contain values reported by the sensors such as GPS, accelerometer, barometer, and microphone. GPS fixes are used to indicate the incident location and conditions such as being on the ground or building floors can be derived from barometer sensor data. Microphone also records the raw audio messages that firefighters send to other teammates and using the accelerometer sensor data, the system detects the firefighters body movement. CoenFire system has a back-end server containing two web services: one for handling HTTP POST messages containing smartphone data and another web-based user interface (UI) for visualization. The data visualization in CoenoFire is real-time meaning that it is possible to monitor the firefighters activities in the web-based UI. However, this system can only be used for

monitoring in back-end server and no inference takes place in it. This research does not reveal the details of the method of using barometer sensor for detecting the floor in which firefighters are. According to my experiments performed with a SAMSUNG Galaxy S4 and a SAMSUNG Galaxy Note II, the difference between air pressure values sensed by barometer sensor in the first and the seconds floors of a building is not meaningful enough to be relied on for such an inference. Additionally, the level of air pressure strongly depends on climate condition and can even change up to 25 millibars in one day

As another example of using smartphone sensors, Yi et al. [6] propose an Android application for detecting the physical activity of the person carrying the smartphone from a collection of predefined activities such as walking, going up/down stairs, running, jumping, etc. To do this, they use a combination of kinematic sensors namely accelerometer, gyroscope, and magnetic sensors. The point is that the subject person must fix the smartphone on his chest which is a very crucial limitation. The sensor data are stored in the device memory card and then analyzed inside the smartphone. Therefore there is no possibility for remote activity monitoring.

Keally et al. [11], propose a framework composed of an Android application and a set of sensors to detect the daily activity of a subject. In spite of previously mentioned solutions in which typically smartphone sensors are the only sources of data, in this system, a combination of smartphone sensors and some other physical sensors is fed to the classifier. The Android application collects sensors data and performs classification. For the classification, they exploit their Android implementation of the AdaBoost [12] classifier. The subject needs to strap four sensors plus the smartphone to different parts of his/her body, more specifically, head, left and right wrists, and left and right ankles. Physical sensors capture the temperature, light, acceleration, and voice (via microphone) and the smartphone sensors provide the classifier with the GPS and acceleration values. Again in this paper, there is no possibility of real-time monitoring.

Hoseini-Tabatabaei et al. [13] have done a comprehensive survey on new techniques of opportunistic mobile-centric context recognition systems. The main goals of this survey are to classify current methods, to give an overview and some analytical details of them, and to reveal weaknesses of the current systems. It also discusses about some of the most challenging parts of every context recognition system such as data acquisition, preprocessing, feature selection or feature extraction, labeling, and classification algorithms.

2.2 Classification and Pattern Recognition Techniques in Smartphones

Typically, classification and pattern recognition techniques implemented in smartphones, imposes extra consideration on developers. Although top smartphones are equipped with strong processors, massive memories, and robust operating systems, the battery drain is still an unsolved problem when it comes to heavy calculations and long operations. Most of the classification techniques and technologies are memory and process demanding. Therefore, implementing a good classification application in smartphones is quite challenging. Nevertheless, there are many strong frameworks proposed in this area which function very well. For instance, Derick et al. [14] propose a system referred to as MIRAOAD that can classify the driving style into two categorize of aggressive and typical. MIROAD is in fact a smartphones (iPhone) application running an implementation of K-Nearest Neighbors (k-NN). This framework detects right and left turns, aggressive right and left turns, aggressive breaking, excessive speed, etc. To do this, the smartphones is fixed on the dashboard of the car and continuously monitors the car maneuvers and detects aggressive ones. If the system senses a potentially dangerous maneuver, it starts to record video and optionally sends notification messages to an external system through the Internet connectivity of mobile network. The authors use data from accelerometer, magnetometer, and gyroscope sensors for classification. The proposed application dramatically drains battery and the smartphones are required to be plugged into the power outlet during its operation process. This system does not provide real-time monitoring of the car motion.

In contrast, Kose et al. [15] propose an Android application which can detect the physical activity of the person carrying the device in real-time. For this purpose, they only exploit the accelerometer sensor data and classify person's activity into four classes namely, running, walking, standing, and sitting. The main focus of the authors of this is on real-time classification which is entirely performed inside the smartphone. They also compare clustered k-NN and Naive Bayes classifiers and conclude that the former perform far better than the latter. However, the comparison is quite shallow in this paper and its criteria is not clear at all.

Another example of pattern recognition using smartphone sensor data, is what Bujari et al. did [16]. In this paper, authors aim to detect if the person carrying the smartphone is passing a road. They again exploit data captured from accelerometer sensor to extract a pattern for road crossing. Their solution is strictly dependent on the behavior of the pedestrian when crossing the road; the pedestrian must walk at normal speed

to the light, then stop for a moment, pass the street a bit faster, and finally return to his/her normal speed. They combine values of three different vectors (xyz) captured from the accelerometer sensor into a single feature called magnitude to make sure that the smartphones placement (hand, pocket, etc.) does not have any effect on the pattern recognition process. In their paper, patter recognition is limited to a very simple algorithm which works based on magnitude numeric value and the time. Authors of this paper do not utilize any classification technique and only rely on thresholds to infer the road cross. Activity recognition system proposed by Guiry et al. [17] performs in a way same except that the data collected from smartphone is analyzed off-line using various classifiers such as Support Vector Machine SVN [18], Naive Bayes, and C4.5 [19] to classify the activity.

2.3 Chapter Summary

In this chapter, some of the most relevant research in the area of classification and pattern recognition were discussed. It is now clear that majority of the systems that collect smartphone sensors data, concentrate on positioning sensors and more specifically, accelerometer to classify the physical activity of the person who carries the smartphones into classes such as running, walking, or sitting. Some of the papers use well-known classification techniques and others, propose their own algorithms. But the interesting fact is that, none of the discussed systems, provides it users simultaneously with real-time reasoning and remote monitoring. Moreover, due to implementation difficulties and rather complicated experiments required, several aspects of fire detection using smartphones are still untouched by the research community.

Chapter 3

Theoretical Background

As mentioned in Section 1.2, FireDetection framework is a learning system which exploits Naive Bayes Classification for inference on the existence of fire. The framework proposed in this thesis is discussed in depth in Chapters 4 and 5, however, understanding it demands some knowledge about a few machine learning and data mining concepts which are described in the following sections.

3.1 Naive Bayes Classifier

Naive Bayes Classifier is a probabilistic classifier based on Bayes' theorem [20][21]. Bayes' theorem describes the relation between conditional probabilities of a hypothesis and observations as given in Eq. 3.1. Assume that h represents the hypothesis and O represents the observation made.

$$P(h|o) = \frac{P(o|h)P(h)}{P(o)} \tag{3.1}$$

where:

- P(h) = prior probability of hypothesis
- P(o) = prior probability of observations o
- $P(h|o)$ = probability of hypothesis given o (posterior probability)
- $P(o|h)$ = probability of o given hypothesis (likelihood)

Typically, the most probable hypothesis or the maximum a posteriori hypothesis is required to be identified. The maximum a posteriori (h_{MAP}) is given by Eq. 3.2.

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(o|h) \\ &= \arg\max_{h \in H} \frac{P(o|h)P(h)}{P(o)} \\ &= \arg\max_{h \in h} P(o|h)P(h) \end{aligned} \tag{3.2}$$

Now, let $H = h_j \in \{h_1, h_2, \ldots, h_m\}$ be the hypotheses, assuming that hypotheses are *mutually exclusive* and *exhaustive* and $\langle O_1 = o_1, O_1 = o_2, \ldots, O_n = o_n \rangle$ be the various observations made. Then the most probable hypothesis is given by Eq. 3.3.

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h_j|o_1, o_2, \ldots, o_n) \\ &= \arg\max_{h \in H} \frac{P(o_1, o_2, \ldots, o_n|h_j)P(h_j)}{Po_1, o_2, \ldots, o_n} \\ &= \arg\max_{h \in H} P(o_1, o_2, \ldots, o_n|h_j)P(h_j) \end{aligned} \tag{3.3}$$

Naive Bayes Classifier assumes that the conditional probability of observations given hypothesis equals to the production of conditional probabilities of each observation given the hypothesis according to Eq. 3.4.

$$P(o_1, o_2, \ldots, o_n|h_j) = \prod_i P(o_i|h_j) \tag{3.4}$$

By substitution of $P(o_1, o_2, \ldots, o_n|h_j)$ by $\prod_i P(o_i|h_j)$ in Eq. 3.3, Naive Bayes Classifier is given by Eq. 3.5.

$$h_{NB} = \arg\max_{h_j \in H} P(h_j) \prod_i P(o_i|h_j) \tag{3.5}$$

Naive Bayes Classifier is a supervised learning algorithm which means it needs to be trained before being able to do classification. Therefore, it must have a training set. The training set contains a number of observations and the classes in which they are classified. For example, the training set shown in Table 3.1 contains the values of four parameters (T, L, H, P) and a class (Fire) in which various sequence of parameter values are classified.

TABLE 3.1: A training set containing labeled data rows.

T	L	H	P	Fire
A	B	A	C	NO
A	A	B	A	NO
B	B	A	A	NO
A	A	C	C	NO
B	A	B	C	YES
B	C	C	A	YES
B	A	B	B	YES

The aim of a Naive Bayes Classifier is to classify an unseen sequence of parameter values into one of the classes in training set. Assume that the values to be classifiers is B, A, A, C. The classifier must classify this sequence of values into one of the Fire classes: YES or NO. According to Eq. 3.5, the hypothesis with the bigger likelihood must be selected. The classifier needs to refer to the training set to calculate probabilities of each class based on the probability distribution in the training set. For calculating the probability of the class NO, the classifier must count the number of data rows in which T equals to B when the data row is classified as NO. There is 1 data row with this criteria while there are 3 data rows in which the T equals to B and the data row is classified as YES. Hence, the conditional probability of T equals to B given NO equals to 1/4. The classifier calculates all the conditional probabilities

As shown in Table 3.2 raw values of smartphone sensors are numeric (i.e. continuous) whereas, the input data of the Naive Bayes Classifier should be nominal. Thus, a method is required for converting numeric data to nominal data as discussed in 3.2.1.

TABLE 3.2: An example of sensor values captured from smartphone sensors.

Temperature (°C)	Light (lux)	Humidity (percent)	Pressure (mbar)
23	350	33	989.5
22.5	400	32	1001.5
23	410	33	1000.4
23	510	35	993.9
24	71	33	998.4
24	55	32	1002.3

3.2 Preprocessing

Typically, when working with sensor data, it is required to preprocess the raw data. The preprocessing steps include any modification on raw data which improves the classification process especially, detecting and removing outliers (captured data that are not consistent with the rest of observations), feature extraction, and discretization [22][23].

3.2.1 Discretization

Data sets containing numeric data similar to the example given in Table 3.2 are called continuous. Modified Naive Bayes Classifiers can handle continuous input data but typically, the data needs to be discretized in order to be classified by this classifier [24]. The process of converting continuous data into nominal data is referred to as discretization. To discretize a continuous data we need to define labeled ranges in which the data can be categorized into. For instance, we can define a labeled range for temperature values as 0 to 4 labeled A and another one as 5 to 9 labeled B.

Afterwards, we feed the classifier with a table of these labels (nominal values) instead of real numeric values. It must be considered that the way we define these ranges affects the accuracy of the reasoning process: bigger ranges result in less accuracy and less calculation whereas smaller ranges improve the accuracy while imposing more calculation to the system. On the other hand, the ranges must be defined according to the distributions of parameter values. My proposed method of discretization is explained in detail in Subsection 4.2.2.

3.3 Chapter Summary

In this chapter, some of the most important concepts and theoretical background on which the FireDetection framework is based were explained.

Chapter 4

Solution

The goal of this thesis is to find a method for detecting indoor fire utilizing available sensors in smartphones. As a solution, I introduce FireDetection which is a framework composed of one or more smartphones and a back-end server called FDBS. This framework, as a single coherent system, detects fire and visualizes useful information about it such as its location and the reports admitting it. For this purpose, smartphones collect data from their temperature, humidity, light, and pressure sensors and after analysis decide on the existence of fire in their surroundings. Inference in smartphones is based on Naive Bayes Classification since it is efficient and does not need complicated and heavy calculations.

On the other hand, the FDBS maintains the classifier's training set and performs some classification-associated calculations. This improves the battery consumption in smartphones since they do not have to perform classification calculations. Instead, they can access these calculation results via a web service provided by the FDBS. Once a smartphone accesses these results it can start its classification process, hence, the smartphones can function independently. In case the smartphones can connect to the Internet, they send information to the FDBS that are used for real-time monitoring and visualization. The following sections explain how this framework works assuming that the training set is formed and loaded on the smartphones. Of course before the system starts, there are some prerequisites which are briefly described at the end of this chapter in Section 4.6.

4.1 Data Acquisition

First of all, the framework needs to capture data from its surroundings. This is possible by using the smartphone's built-in sensors, but which ones? Today's smartphones have several high quality sensors such as accelerometer, gyroscope, magnetometer, light, etc. The aim of FireDetection framework is to detect fire, hence, it must receive data from those sensors which are affected by an indoor fire. Of course it would be nice if smartphones have smoke sensor because in most cases fire makes smoke and in many fire detection systems, smoke is the most (or one of the most) important signs of fire. However, it is still possible to compensate for the lack of smoke sensor in smartphones.

According to my experiments, an indoor fire dramatically impacts the room temperature, humidity, light, and pressure. Fire has different phases [25] but generally during all of them, temperature is significantly higher than normal (more than 200 °C) and higher temperature results in lower relative humidity. In smoky phases the room light also decreases and fire flame consumes oxygen and affects the air pressure. As you see the four selected parameters are decisive enough for inference on existence of fire. These experiments and their analysis results are described in 4.4.

Now we need a mechanism for acquiring the aforementioned parameter values from smartphones sensors. For this purpose, I developed an Android application (see 5.1) which captures momentary values from temperature, humidity, light, and pressure sensors. Whenever a sensor reports a new value, the application creates a new data row containing all sensors values. Moreover, in each data row the location coordinates, a time-stamp, and eight more values are added. These extra eight values, are four means and four variances, each of which is calculated from the most recent five values reported by the sensor.

The data collected from smartphone sensors are stored on the memory card of the device, and another copy of them is also sent to the FDBS to be logged and visualized.

4.2 Preprocessing

As said in Chapter 3 in almost all data mining problems, raw data require preprocessing. The preprocessing in FireDetection framework includes two main tasks which are feature selection/extraction and discretization. Nevertheless, there may be other trivial data modifications needed, but I prefer to perform them manually due to time limitation.

4.2.1 Selecting and Extracting Features

One of the most critical decisions to be made when performing classification is to select appropriate features. Some features can be simply selected among raw data samples and others need to be calculated or in other words, extracted from existing data samples. Considering the smartphone sensors available at the moment, and the most important impacts of fire on a room's spatial parameters, I use the sensors referred to in Section 4.1 to detect fire plus the GPS coordinates to determine its location. But the measurements of these four sensors can be easily affected by unwanted environmental factors. For instance, temperature sensed by a smartphone exposed to direct sunlight does not represent the room temperature. A big part of these type of issues is solved by precisely training the reasoner. But a Naive Bayes reasoner shows better performance when it fed with better features.

During the reasoning process, we may encounter conditions in which one of the sensors perceives a very high or low value and hence, the whole reasoning process is affected. Assume that the temperature sensor normally reports temperature values around 20 °C. It suddenly senses a very high temperature as 40 °C and reports it. In case of having a training set only containing the values of the four sensors, most likely, such a temperature represents fire, regardless of other parameter values. But what if the smartphone is placed near a fireplace or a heater?

To overcome these sort of issues, I define eight more features; four mean values and four variance values. Mean value helps the reasoner to understand if a huge change reported by sensor is permanent. Variance value also represents the rate of changes and based on it the reasoner can determine that the room parameters are stable or not. The mean and the variance values are calculated from the most recent five values reported by each sensor using Eg. 4.1 and Eq. 4.2 respectively.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} T_i \tag{4.1}$$

$$Variance = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{4.2}$$

4.2.2 Converting Numeric Data Set to Nominal

Until now, we know how a data row containing numeric sensors data, extra features (means and variances), GPS coordinates, and time-stamps is formed by smartphones. However, as mentioned in Subsection 3.2.1, numeric data sets should to be converted to nominal data sets in order to improve the performance of the Naive Bayes Classifier. To perform the discretization process, we need to define labeled intervals.

Now, the question is that how the size and the number of intervals should be? This is crucial since it can affect the accuracy of the classification. On the other hand, same sized intervals are not efficient since there are areas in each parameter distribution that does not offer any useful information. For example, as illustrated in Figure 4.1(a) temperature values smaller than 0 °C are not important to be divided into many intervals since typically in normal indoor places, such a temperature is not sensed at all.

To form efficient intervals, the mean value of each parameter is required since the decisive intervals are always near mean values. According to my experiments, the mean value of the room temperature is 22 °C and so, intervals closer to 22 are defined smaller than the others. Afterwards, the size of each interval must be decided. The interval sizes vary in accordance to the mean and the distribution of each parameter values.

Table 4.1 lists the proposed intervals for each parameter and their corresponding nominal values (e.g. A, B, C, etc.) where T, L, H, P, TM, LM, HM, and PM stand for temperature, light, humidity, pressure, temperature mean, light mean, humidity mean, and pressure mean respectively. In this table, the temperature unit is centigrade degree (°C), the light unit is lux, the humidity unit is percent (%), and the pressure unit is millibar.

TABLE 4.1: Proposed intervals for each parameter (feature) and their corresponding nominal values.

	A	B	C	D	E	F	G	H	I
T	[1-]	[0-7]	[8-12]	[13-15]	[16-19]	[20-24]	[25-35]	[30-39]	[40+]
L	[0]	[1-149]	[150-799]	[800-10000]	[+10000]				
H	[0-9]	[10-19]	[20-24]	[25-29]	[30-34]	[35-39]	[40-50]	[50+]	
P	[-979]	[980-999]	[1000-1024]	[1025+]					
TM	[1-]	[0-7]	[8-12]	[13-15]	[16-19]	[20-24]	[25-29]	[30-39]	[40+]
LM	[0]	[1-150]	[151-500]	[501-10000]	[+10000]				
HM	[0-9]	[10-19]	[20-24]	[25-29]	[30-34]	[35-39]	[40-50]	[50+]	
PM	[-979]	[980-999]	[1000-1024]	[1025+]					

Not surprisingly, variance values of temperature, humidity, and pressure are so close to zero in normal conditions while light variance values can get very big because smartphones can be exposed to direct sunlight. Therefore for the first three parameters two intervals A (close to zero with a very tiny range for each one) and B (otherwise) and for the light variances twenty intervals with the same width from 0 to 200,000 lux are defined.

4.3 Deciding on Existence of Fire

Once the smartphones sensor data are discretized, the Naive Bayes Classifier (reasoner) can start its inference process to determine if a fire exists in the smartphone's surroundings. To do this, the reasoner receives a data row containing nominal values of temperature, light, humidity, pressure, 4 means, and 4 variances. At this stage, the reasoner must refer to the training set and fetch required conditional probabilities. According to Eq. 4.3, Naive Bayes Classifier needs the conditional probabilities of each observation (parameter values) given hypotheses (Fire, Normal) along with the probabilities of hypotheses to calculate the maximum likelihood:

$$\begin{aligned} P(Fire|O) &= P(Fire)\prod_{i=1}^{n} P(O=o_i|Fire) \\ P(Normal|O) &= P(Normal)\prod_{i=1}^{n} P(O=o_i|Normal) \end{aligned} \tag{4.3}$$

where O represents the set of nominal value of each parameter. After this calculation, the reasoner has the conditional probabilities of both Fire and Normal classes. The original Naive Bayes Classifier can only decide on the class in which the input sequence of values must be classified. In our case, the result of such a classification would be Fire or Normal which only relies on the bigger likelihood to one of the classes (Fire/Normal). This can be useful in activity recognition frameworks in which the classification result does not have any risk. But fire detection, naturally, is a risky process and even a correct classification has some risk.

On the other hand, we want to visualize the fire in the FDBS and by this classification in case of having fire, the visualizer can only show a point on the map in which the probability of having fire is bigger than the probability of having normal condition and nothing more. The ideal visualization would be a map that not only illustrates the location of a fire, but also depicts the seriousness of the condition. This can be also

used for alarm triggering. Therefore, I modify the original classifier so that firstly, it can produce two numbers representing the risk of triggering alarm and staying silent and secondly, this numbers can be easily visualized to show the momentary probability of having fire.

The proposed reasoner uses a method for calculating the risks of the two actions (triggering alarm and staying silent). Each action imposes a risk to the system and therefore, is assigned a numeric value. The risk values of each action in cases of fire and normal are listed in Table 4.2.

TABLE 4.2: Constant values used for calculating the risks of triggering the alarm or staying silent.

	Alarm	Silence
Fire	1.0	10.0
Normal	5.0	0.0

As an example, assume that the reasoner have calculated both of the conditional probabilities of fire and normal classes given a sequence of observations as x and y respectively. According to the risk values, the risk of triggering alarm is calculated as:
$R(Alarm) = 1.0 \times x + 5.0 \times y$
and the risk of staying silent in case of having fire is calculated as:
$R(Silence) = 10.0 \times x + 0 \times y = 10.0 \times x$

The action with smaller risk will be the final result of the reasoning process. Once the Risk(Silence) becomes bigger than Risk(Alarm) the reasoner triggers the alarm. This also means that the visualizer module of FDBS must render a circle on the map using fire location coordinates. The diameter of this circle shows the seriousness of a fire case (ζ) which is given by Eq. 4.4.

$$\zeta = \frac{P(Fire|O)}{P(Normal|O)} \tag{4.4}$$

The value of ζ can be very big or very small therefore, it cannot be directly associated with the circles diameter. Hence, the mapping mechanism is defined which maps the seriousness values to circle diameters as shown in Table 4.3. The logic behind this mapping is simple. For example, if the ratio of $P(Fire|O)$ to $P(Normal|O)$ is 0.6, it means that the conditional probability of fire given a specific sequence of parameter

values is 40 percent smaller than the probability of normal given the same sequence of parameter values and so on.

TABLE 4.3: The mapping table from seriousness level or numeric ζ values to circle diameter.

Seriousness Level	ζ Range	Circle Diameter
1	(0.6-1)	1000
2	(1-1.6)	2000
3	(1.6-2)	3000
4	(2+)	4000

An example of visualized smartphones data is depicted in Figure 5.4 in the next chapter.

4.4 Experiments

Typically, in supervised learning systems, the training set is extremely important and the FireDetection framework is not an exception. To make a good training set, we have to design and perform many experiments. We also need to evaluate the accuracy of the classification or reasoning process. For this thesis, I designed and performed experiments in two phases: analysis and training which are explained in the following subsections.

4.4.1 Analysis Phase

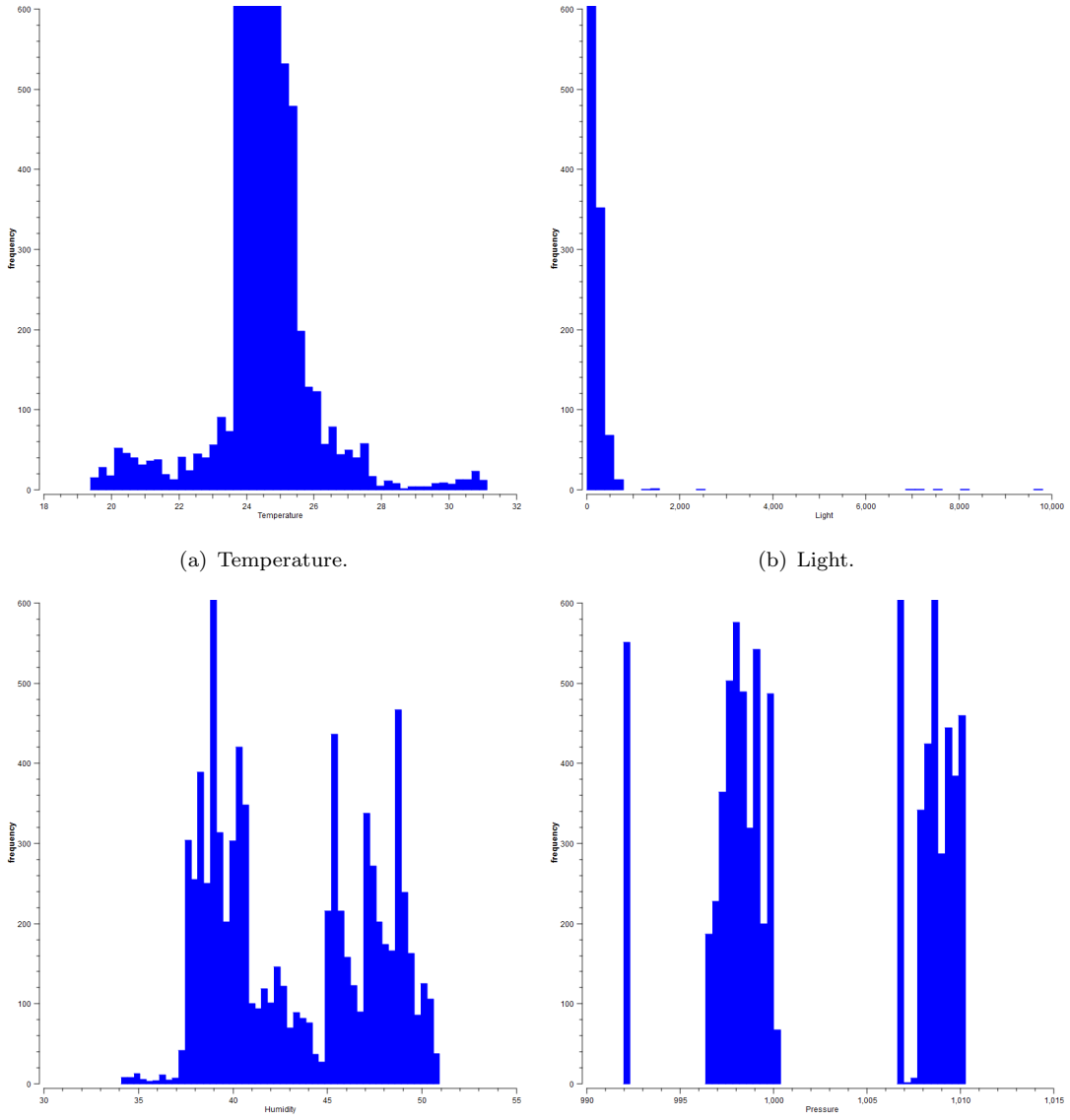
The analysis phase of experiments can be done in normal condition (i.e. absence of fire) since in this phase, the goal is to extract mean values of each parameter. Mean values are used to form the intervals required for discretization process (see 4.2.2). By normal condition, I mean possible combinations of the room's parameter values given that there is no open fire in the room. To do this, I modified the Android application so that it only stored raw numeric sensor values plus means and variances as discussed in 4.2.1. Then the smartphone running the application was exposed to diverse normal conditions as listed in Table 4.4.

TABLE 4.4: Experiments performed in analysis phase and the parameters that are subjects to investigate.

Place and Condition	Parameter(s) to Investigate
table, near closed window	light
table, near open window	temperature, light, humidity
table, near heater	temperature, humidity
desk, under the room light	light
desk, under the reading lamp	temperature, light
university's corridors	temperature, light, humidity, pressure
supermarket	temperature, light, humidity, pressure

The variance values are used as features to show sudden changes in room parameter values. But as mentioned earlier, the variance values in normal condition are so frail because changes in room parameters take place gradually when there is no fire. Typically, room's parameter values do not change significantly, but there are factors which can affect them. For instance, toggling the windows status (open to close or close to open) or starting a cooler or heater may make some changes in the room's temperature and humidity but these are not as significant as changes made by fire. There are cases in which the smartphone senses extra ordinary values such as when it is exposed to direct sunlight. In such a case, although the smartphone perceives massive amount of luminance (e.g. 50000 lux), this condition is still steady and consequently, the light variance will be decreasing after a while.

The data captured in analysis phase were numeric and so, they could not be used in training set. It was required that the whole experiment was done again while the smartphone discretized sensors values. The results of the experiments of this phase performed according to the scenarios listed in Table 4.4, are illustrated in Figure 4.1.



(a) Temperature.

(b) Light.

(c) Humidity.

(d) Pressure.

FIGURE 4.1: Distributions of the four major parameters involved in fire detection process.

4.4.2 Training Phase

In spite of the analysis phase, the training phase should contain data from a real fire. Although it was possible to use applications such as Fire Dynamics Simulator (FSD) [26] for forming the training set, we decided to perform real experiments. Therefore, we asked Safemar AS [27] to let us participate in their firefighting drills. Safemar AS trains participants in a wide range of courses related to security and safety such as firefighting, first aid, sea rescuing operations, etc.

(a) The ship.

(b) The container and the wood fire insde it.

(c) A fire fighter brings out the smartphones.

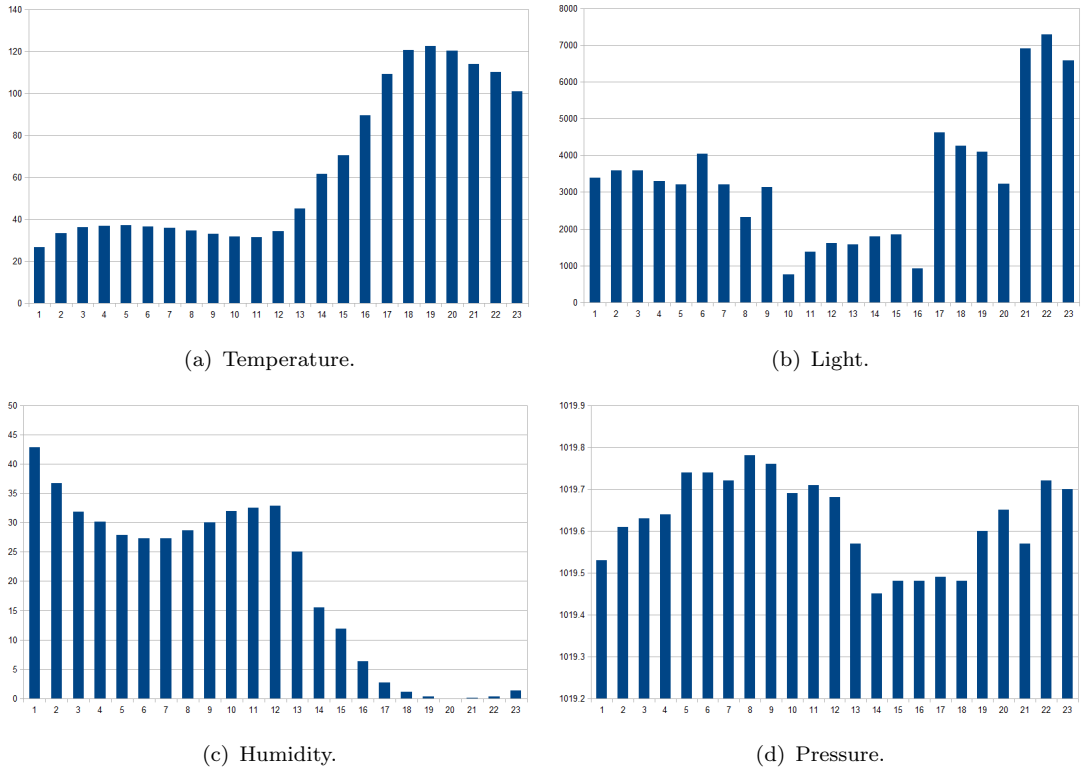
(d) A fire fighter controls the condition inside the container.

FIGURE 4.2: Various fire experiments in Safemar training yard.

At Safemar AS, we collected data from two various indoor fires made inside an old ship and a container. The ship and the container can be seen in Figure 4.2. The ship room was bigger than the container. The former was approximately 150 square meters and the latter was 20 square meters. Both rooms were completely dark and during the drills the doors were sealed. Inside the ship room they made fire in big lamp oil containers, hence, the fire made a huge amount of smoke and soot. But the fuel in the container was a dump of wooden stuff and so, it was more realistic and more similar to room furniture. During the fire drills, the smartphones were held by the firefighter in the height of a table (approximately one meter from ground).

The ceiling of the ship room was about three times higher than the normal rooms' (2.8-3.0 meters). Considering that the high temperature air, gases, and smoke go up to the ceiling, in the ship room the rise of the temperature could not be sensed by smartphones. Therefore, I decided to use the results of the container experiment for forming the training set and ignore the data captured in the ship room. A portion of

the data captured from the container experiment is illustrated in Figure 4.3 where the left side vertical axes represent the parameter values in aforementioned units and the horizontal axes hold the record number or time (since the records are captured in equal time intervals). Note that in this phase the results are not shown as distribution graphs, instead, they are plotted as variation diagrams to show the dependency between the four parameter values.



(a) Temperature.

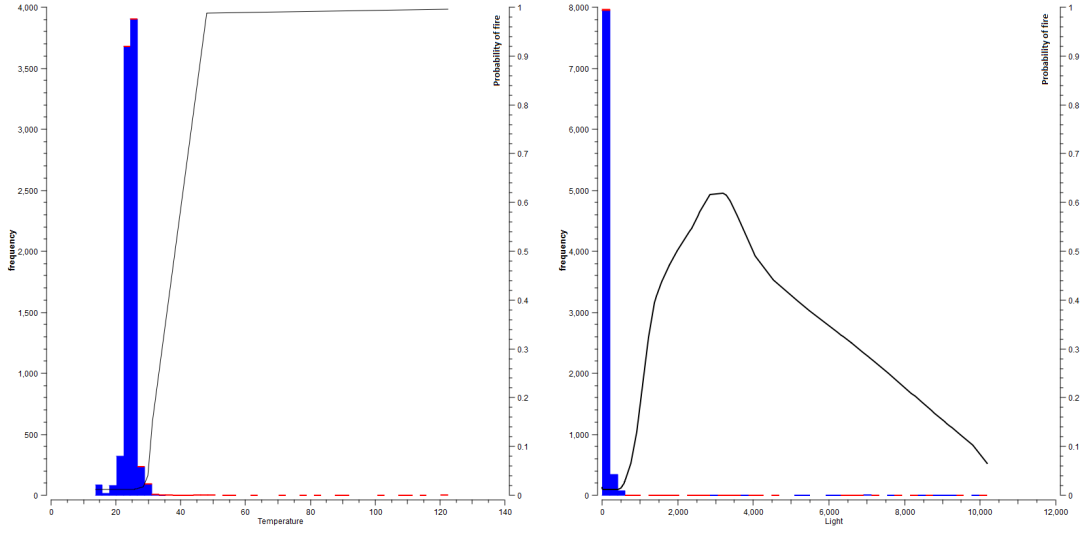
(b) Light.

(c) Humidity.

(d) Pressure.

FIGURE 4.3: Variation diagrams of the four parameters captured in Safemar experiments.

In order to form the training set, it is required to manually set the value of the fire column in all data rows, i.e. classify the data rows into to the classes of fire and normal. To do this, the data collected from the Safemar experiment and also the data collected in normal condition during one week, were merged together. Figure 4.4 depicts the distributions of the temperature, light, and humidity values in the FireDetection training set.



(a) Temperature.

(b) Light.

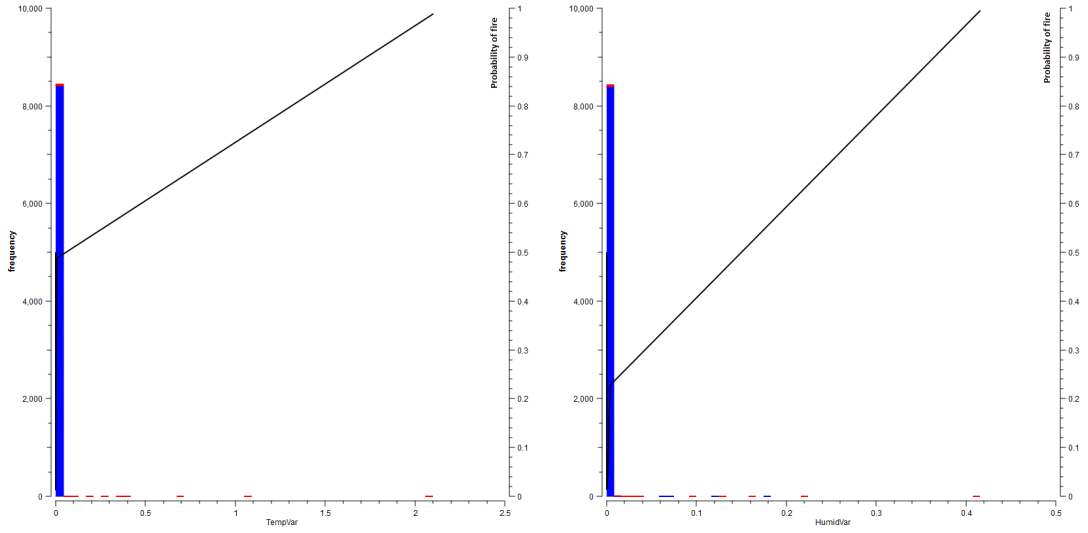
(c) Humidity.

FIGURE 4.4: Distributions of temperature, light, and humidity, in the training set and their relation to probability of fire.

In above figures, the two classes of fire and normal are illustrated with different colors (red for fire and blue for normal) and the values of the probability of fire are shown by black lines. In the Safemar experiment, fires did not have any meaningful effect on the room's air pressure since the container room had a good ventilation system and air could easily enter it. The ventilation system compensated for the lack of oxygen. That is why the distribution of pressure values has not been included in Figure 4.4. However, according to interview with firefighters, in many fire cases especially in smaller rooms,

the level of pressure drastically changes during fire. Therefore, the pressure and its mean and variance values are used in the reasoning process.

The distributions of the temperature and humidity variance values were also impressive and clearly show their association with the probability of fire where bigger variance values in both parameters is mostly seen when fire exists in smartphone's surroundings.



(a) Temperature variance. (b) Humidity variance.

FIGURE 4.5: Distributions of temperature variance and humidity variance in the training set, and their relation to probability of fire.

The distributions of these two parameters with the curves of fire probability are shown in Figure 4.5.

4.5 Experiments Analysis

The results gained in Safemar experiments are quite interesting in terms of explaining indoor fire behavior. Comparing the four graphs in Figure 4.3 the change in all four parameter values is evident near record number 12. At the beginning, when there is no fire all parameters especially temperature and humidity, show steady curves near their corresponding mean values. As expected, at the time-step 13 when the room temperature goes up to 53 °C 4.3(a) the level of relative humidity falls down to almost 25 percent 4.3(c). Fire consumes oxygen and if the room does not have an appropriate ventilation system the vacuum condition happens which mean decrease in pressure. But if the room have an air conditioning or ventilation system (as in our experiments) then the fluctuations in air pressure happens 4.3(d). The impact of fire on the room light is more

complicated and cannot be judged simply. According to this experiment, interviews with firefighters, and also other fire guides such as [25], each fire has different phases such as pre-ignition, flaming, glowing, and smoldering. In each of these phases the room light can be increased or decreased due to smoke covering lights or flame or ignition. At least we can be sure that an indoor fire changes the value of the room's light (increase or decrease) and this can be seen in Figure 4.3(b).

After performing these experiments, it is quite safe to state that smartphones with their current set of sensors can detect fire. In other words, changes in room parameters in case of fire are big enough to be sensed by smartphone sensors.

4.6 Prerequisites

FireDetection framework is a learning system and hence, before it can start to infer on fire existence it must be trained. As mentioned in Section 3.1 for training the classifier, a data set must be formed containing all possible classes (fire and normal in our FireDetection framework). Forming the training set mandates several prerequisite tasks.

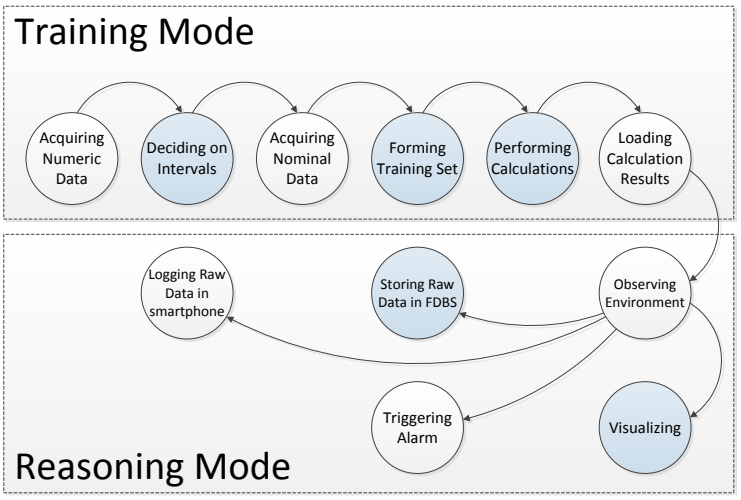


FIGURE 4.6: The main process of the FireDetection framework.

Prerequisite tasks are mostly about data acquisition and training the reasoner so that it can detect fire accurately. This includes all the tasks that must be finished before the system can start detecting fire. As shown in Figure 4.6 these tasks are performed when the framework is in its training mode. Once the prerequisites are fulfilled, the framework changes to reasoning mode and the smartphones are ready to observe their environment and detect fire and the FDBS can visualize the fire information.

The first prerequisite task is to collect numeric data from smartphone senors and based on it to decide on the size of the discretization intervals. When the intervals are decided, the discretization process can start and hence, the smartphone can create the training set. At this stage, the smartphones receive their sensor data, discretize them, and send them to the FDBS to be stored in the database. Afterwards, The FDBS calculates the results required for classification and makes them available to smartphones through a web service. I think going deeper into the details of prerequisite tasks is not necessary since they are done once and in fact, are not a part of the FireDetection functionalities.

4.7 Chapter Summary

In this chapter, I focused on the details of the proposed solution. I mentioned that the proposed framework is called FireDetection and is composed of the smartphones and the FDBS. The smartphones detect fire based on Naive Bayes Classification in an independent way and the FDBS provide users with remote monitoring and data visualization graphical user interface. The experiments performed during this project and the results gained were also brought up and discussed in this chapter. These results strongly indicated that the idea behind this project is valid.

Chapter 5

FireDetection Implementation

FireDetection framework is composed of two main entities namely the smartphones and the FDBS each of which contains several components as demonstrated in Figure 5.1. The framework is designed so that it can independently detect fire and trigger the alarm which can be an on-screen notification, an SMS or phone call, or a message to the FDBS.

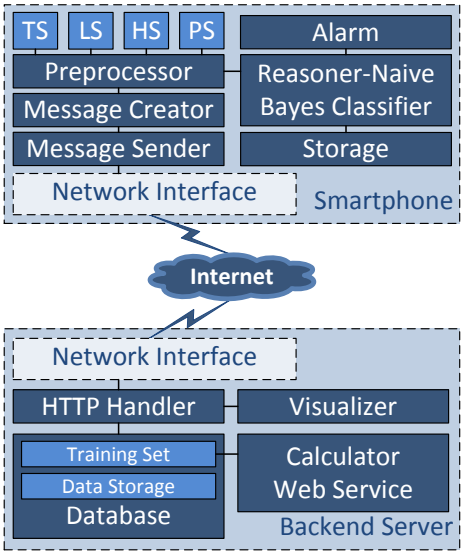


FIGURE 5.1: The general components of the FireDetection framework.

Currently, the FDBS can serve more than one smartphone, although as a proof of concept it only interacts with one. Nevertheless, in case of having more than one smartphone, the FDBS should have some additional components such as big data handler, aggregation module, and a more complex analyzer which can consider all smartphones measurements

together. Following sections address the roles of FireDetection components and reveal the framework's implementation details.

5.1 Smartphone

The smartphones used in our system must have temperature, humidity, and barometer sensors in addition to other usual sensors. Temperature and humidity are the most important parameters in fire detection since indoor fire increases the temperature of the room and decreases the level of relative humidity. Additionally, the selected smartphones should be high performer in terms of processing power and memory. Currently, SAMSUNG Galaxy S4 is one of the best choices for this purpose because it has required sensors, a quad-core 1.6 gigahertz processor and 2 gigabytes of random access memory (RAM). This smartphone is an Android device so the prototype is designed for Android smartphones.

Although the smartphones can detect fire independently, the raw data and the seriousness of the report in each time interval are stored in FDBS's database to be used for visualization and further analyses. The smartphone sensor data are sent through the Internet, so, the smartphone must be connected to the Internet to be able to send them, otherwise, the data is only stored in its SD card. The smartphones start sending messages to the FDBS as soon as they are connected to the Internet.

Basically, the smartphones in FireDetection framework have two working modes: training mode and reasoning mode. The training mode is used to form the training set. Although in this mode the sensor values and other parameters are generated and sent to the FDBS, but the whole process is not fully automatic in this mode and some modifications are needed on the training set. Nevertheless, the process in reasoning mode is completely automatic and independent from external resources.

5.1.1 Sensors

The technical information of the sensor of the SAMSUNG Galaxy S4 which are used in FireDetection framework are listed by the Table 5.1. Note that mA in this table is an abbreviation of milliampere.

TABLE 5.1: SAMSUNG Galaxy S4 technical sensors information.

Sensor	Power	Maximum Range	Unit
Ambient Temperature	0.3 mA	165.0	°C
Relative Humidity	0.3 mA	100	percent
Light	0.75 mA	60000.0	lux [28]
Barometer	1.0 mA	1013.25	millibar

5.1.1.1 Ambient Temperature

The temperature sensor plays the most important role in fire detection because indoor fires impact the ambient temperature and heat dramatically. There is no straight forward way of detecting heat using smartphone sensors but using the ambient temperature sensor the temperature of the surrounding space of the smartphone can be sensed in centigrade degrees. As shown in the Table 5.1 the maximum value that the smartphone temperature sensor can sense is 165 °C. This is sufficient for fire detection since the mean value of the room temperature is much less (see experiments in Chapter 4).

5.1.1.2 Relative Humidity

Fire also consumes the water vapor in the air and consequently affects the level of humidity in its environment. The humidity sensor gives the relative humidity which is useful when detecting fire. Relative humidity takes values from 0 to 100 and ϕ is defined by the Eq. 5.1 where the e_w and e_w^* represent the pressure of the water vapor existing in the air and the biggest possible pressure at the same temperature respectively.

$$\phi = \frac{e_w}{e_w^*} * 100 \tag{5.1}$$

5.1.1.3 Light

Light is another parameter that can potentially be impacted by fire. It is difficult to determine the exact effect of indoor fires on the room light since depending on environmental parameters, fire may show different behaviors in terms of flame, smoke, etc. For instance, different materials burn in diverse speed and make various types and amounts of smoke. Also the level of oxygen in the room directly affects the flame where more oxygen typically results in bigger flames. In such various conditions, we cannot exactly

state that the fire increases the light because of its flame or decreases it because of its soot or smoke. Therefore, the only thing that can be safely stated is that fire in a room affects the level of light, so, it is used as a parameter in FireDetection framework.

There are special situation in which the smartphone is exposed to the direct sun light. In such a case, the system must be able to distinguish between high temperatures caused by direct sun light from the ones made by a real fire. The light sensor is one possible option that helps the system to detect cases with high temperatures caused by direct sunlight. Typically, the sunlight is much stronger than any other light sources that can be found in indoor spaces (see experiments in Chapter 4).

5.1.1.4 Barometer

The barometer sensor measures the air pressure in millibar. Generally in physics, pressure is defined as the proportion of force to a surface as shown in Eq. 5.2 where P is pressure, F is force and A is area which the force applies [29].

$$P = \frac{F}{A} \tag{5.2}$$

Note that in Table 5.1 the air pressure unit is millibar although there are other units for air pressure (and in general pressure) such as Pascal and Torr [28]. According to this definition, air pressure is the weight of air pillar above a surface. The air pressure has a complicated relation with the air temperature. But in indoor places, this relation is easier to explain where higher temperatures result in lower air pressure. High temperatures in a room causes air molecules (oxygen, nitrogen, etc.) to move more quickly and consequently the density of the room's air decreases. However, there is one caveat that is the room's ventilation system. If the room has an appropriate ventilation system then the pressure does not notably change.

5.1.2 Message Creator

This module is responsible for creating XML messages that are supposed to be sent to the FDBS. Another version of the sensor data is stored in a text file on the device memory card in the form of tab-separated values. This text file is only used as a log file. An example of XML file generated by this module and received by the FDBS is illustrated in Figure 5.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<parameters>
  <temperature>18.024141311645508</temperature>
  <light>87.0</light>
  <humidity>31.517568588256836</humidity>
  <pressure>1016.0999755859375</pressure>
  <latitude>58.33534591</latitude>
  <longitude>8.57539572</longitude>
  <temperatureMean>15.773664315541586</temperatureMean>
  <lightMean>55.0</lightMean>
  <humidityMean>33.77412827809652</humidityMean>
  <pressureMean>1016.096669514974</pressureMean>
  <temperatureVariance>0.3704314972424072</temperatureVariance>
  <lightVariance>0.0</lightVariance>
  <humidityVariance>0.4241601758622466</humidityVariance>
  <pressureVariance>2.226564619301547E-5</pressureVariance>
  <seriousness>1000</seriousness>
  <currentDateTime>Apr 1, 2014 7:48:21 PM</currentDateTime>
</parameters>
```

FIGURE 5.2: An example of XML messages sent from smartphone to FDBS.

In Android programming, the application cannot request the sensor values. Instead the system calls a method called `OnSensorChanged()` each time a sensor has a new value to report. Therefore, technically there is no control on the time intervals that this method is invoked [30]. The message creator updates the XML message parameters each time one of the sensor values changes and leaves others without any changes. This assures us that the XML messages always contain the most recent sensor values.

5.1.3 Message Sender

The XML message created by MessageCreator module is wrapped up in an HTTP POST request and is sent to the FDBS by the message sender module. In Android applications, blocking processes cannot be implemented in the main threads and since all network related components are considered as blocking processes, the message sender module is put in a separate thread. The message sender, sends the XML message every five second, if the smartphone is connected to the Internet. Otherwise, the XML messages are overwritten and the sensors data are only stored in the device memory card.

5.1.4 Storage

As illustrated in Figure 5.1 the main training set required for the reasoning process is maintained and manipulated by the FDBS. Therefore, there is no need to perform heavy calculations inside the smartphone. However, the calculation results must be sent

to the smartphone and kept somewhere inside it. According to Android official documentation, there are currently five options for storing data in an Android application as shared preferences, SQLite database [31], internal storage, external storage, and network connection [32]. The FireDetection framework is a prototype so its storage performance is not the most important issue, but I briefly mention that since the information I want to store is not particularly large, database is not necessary. On the other hand having text files forces the application to parse them regularly which is not a good idea as well.

I utilize the shared preferences which is a storage space managed by the Android operating system (OS) and can be used to store rather small amount of data in the form of key-value. Typically, the shared preference space is used to keep the application configurations but I use it to store the results of the calculations required by the reasoner. The information stored in shared preference are the values of the conditional probabilities of various observations given fire and normal based on data in training set. Table 5.2 lists some examples of the data items stores in the shared preferences.

TABLE 5.2: Examples of conditional probabilities data stored in the shared preferences.

Key (string)	Value (float)	Description
TAgF	0.0035	cond. prob. of temperature equals to A given fire
TBgN	0.002	cond. prob. of temperature equals to B given normal
HCgF	0.00081	cond. prob. of humidity equals to C given fire
LMCgN	0.0066	cond. prob. of light mean equals to C given normal
PVFgF	0.009	cond. prob. of pressure variance equals to F given fire

The most important advantage of the shared preference is that reading from and writing to it are quite easy in terms of programming and there is no need to implement a text parser in the Android application. Moreover, it is private and other applications cannot access its values.

5.1.5 Preprocessor

Preprocessor module of the smartphones is responsible for calculating mean and variance values and also performing discretization on numeric sensors data in respect to the intervals given in Table 4.1. The momentary means and variances are calculated by `CalculateMean()` and `CalcualteVariance()` methods respectively. The discretization process is also performed continuously by the `Discretize()` method. These three methods are called when a new sensor value is reported and hence, the

momentary nominal values of sensors' numeric values are immediately accessible to reasoner.

5.1.6 Reasoner

The reasoner module of the smartphone is where the classification takes place. The inputs to this module are the sensor data and the other features previously generated by the preprocessor. The input of every single reasoning round contains the nominal values of 12 features as mentioned in Subsection 4.2.2. For instance, assume that the sequence of feature values is like A, A, B, ... Knowing the order of the parameters in this sequence, the reasoner refers to the probability values kept in the storage and fetches them according to the input. It needs to find the conditional probability of temperature equals to A given fire and normal, conditional probability of humidity equals to A given fire and normal and so on. To do this, the reasoner invokes a method called `LoadTrainingSet(String key)` and provides it with the "key" associated to the desired "value". Considering the rule of making key names explained in 5.2, the value of conditional probability of temperature equals to A given fire is stored in association with the "TAgF" key. Therefore, invoking the `LoadTrainingSet(String key)` method with input string of "TAgF" will return "0.0035" which is our desired probability. Once all conditional probability values are fetched from shared preference storage, the reasoner calculates the conditional probabilities of fire and normal given the sequence of values. Then it calculates the risks of triggering alarm or staying silent in accordance to the risk calculation explained in 4.3, and decides if the alarm should be triggered. The momentary value of seriousness of the case is sent to the FDBS to be used by visualizer for plotting maps and other monitoring purposes.

It is possible that the training set does not have the value of the conditional probability associated with a specific key. For instance, the temperature observed by the sensor is converted to the value X but calling the `LoadTrainingSet(X)` returns 0. Therefore, considering Eq. 4.3, the value of $P(Fire|T = x, L = y, \dots)$ will be equal to zero regardless of other conditional probabilities. This issue is called zero-frequency and must be treated somehow since in reality we know that even if we do not have such a record in our training set it does not mean that the probability of its occurrence is zero. There a few solution to overcome this issue such as using Laplace Estimator [33] but the easiest solution is to simply remove the effect of such a case by putting "1" instead of "0". This, results in extraordinarily high probability. Another option is to put a very small value instead of zero but this may also affect the correctness of the reasoning

since it means that the reasoner is relying on something other than the training set. In FireDetection framework this problem is solved by manually checking the training set and making sure that all possible keys have one value.

5.1.7 Alarm

There can be many types of notification generated in case of having fire such as a beep (or ring), a voice call or an SMS sent to fire station or any other involved authorities, etc. In this project the smartphone plays an alarm sound when detecting fire and shows a fire icon on the application screen as depicted in Figure 5.3. Moreover, the visualization module of the FDBS continuously observes the smartphone environment and visualize the status of the probable fire.

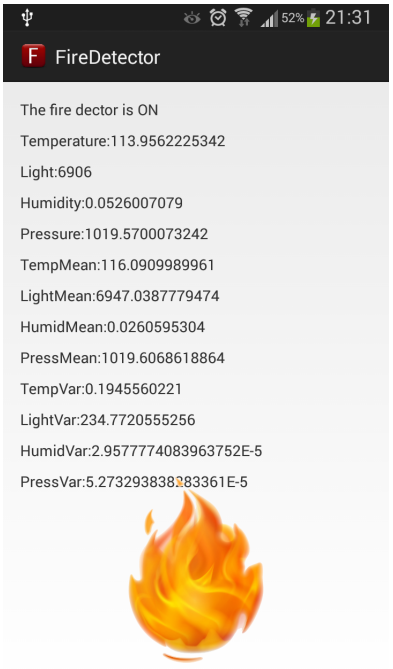


FIGURE 5.3: The fire icon on the smartphone screen when the fire is detected.

5.2 FDBS

Battery duration is one of the main challenges in every project putting the burden of heavy calculations on smartphones. So, I try to strip down the smartphone calculation tasks as much as possible and perform them in FDBS. The two main tasks performed in FDBS are maintaining the training set and calculating conditional probabilities based

on it. As illustrated in Figure 5.1, the FDBS has a visualizer component as well that provides us with remote monitoring.

5.2.1 HTTP Handler

In FireDetection framework, smartphones send their sensors data in the form of a Hyper Text Transfer Protocol (HTTP) POST request containing an XML message as its body to the FDBS to be stored and visualized. Therefore, the FDBS must be able to handle HTTP POST requests. By handling I mean receiving the request, parsing XML message in its body, extracting parameters and their values, and passing them to database and visualizer modules. To do this, the HTTP handler must be deployed in a web server. All modules in FDBS are developed using Microsoft technologies and the HTTP handler is not an exception. Microsoft's Internet Information Services (IIS) is used for deploying the HTTP handler. The IIS must be configured so that it can redirect incoming requests to different handlers and web applications that are deployed in it. For this purpose, the IIS checks the extension of the incoming request and based on it, chooses the right handler or web application to hand over the request. I defined a new extension for requests sent from smartphones as .smrtrsq. Hence, smartphones must send HTTP POST request to the following Uniform Resource Locator (URL):

```
http://IP-Address/FireDetector/request.smrtrsq
```

Once the extension is defined and the HTTP handler is deployed in IIS, all incoming requests having the above mentioned extension will be automatically redirected to FireDetection HTTP handler.

5.2.2 Database

The smartphone sensor data received and parsed by HTTP handler, are then delivered to the database which is implemented in Microsoft SQLServer 2008 R2. developed by Microsoft. The database consists of a number of tables and stored procedures available to the handler enabling it to manipulate the database. General functionality that the stored procedures provide is "insert into database" so that the HTTP handler can insert the sensor data into the database. The database consists of two tables: one for storing the training set and the other one for storing the raw data received from smartphones.

5.2.3 Calculator Web Service

Although in Figure 5.1 the calculator web service is depicted as a single module, its implementation is a bit different. The implementation details of this module is out of the scope of this project, however, it is composed of a stored procedure in the database and a web service in the FDBS. This module has two responsibilities: first, to calculate the conditional probabilities required by the reasoner (i.e. classifier) and second, to make them accessible to the reasoner as a web service. Consequently, the smartphone does not need to perform heavy calculations during its reasoning process and the web service helps the reasoner to keep its resources updated.

5.2.4 Visualizer

XML messages containing sensor data, GPS coordinates, and the seriousness value are sent to the FDBS in every five seconds. Therefore, the visualizer module of the FDBS can visualize the location of the smartphones and the momentary seriousness of fire reports in a real-time manner. To do this, the Google Maps Application Programming Interface (API) [34] is used for visualization. Google Maps API is a JavaScript library that can be integrated to web applications and web sites. The map is embedded in a C Sharp web application which interacts with the Microsoft SQLServer database and fetches the most recent records received by executing a stored procedure. The visualizer web application, executes this stored procedure and refreshes the map every five seconds which means any changes in database is visualized in the map almost immediately. To refresh the map, an Asynchronous JavaScript And XML (AJAX) library called Ajax Control Toolkit is used.

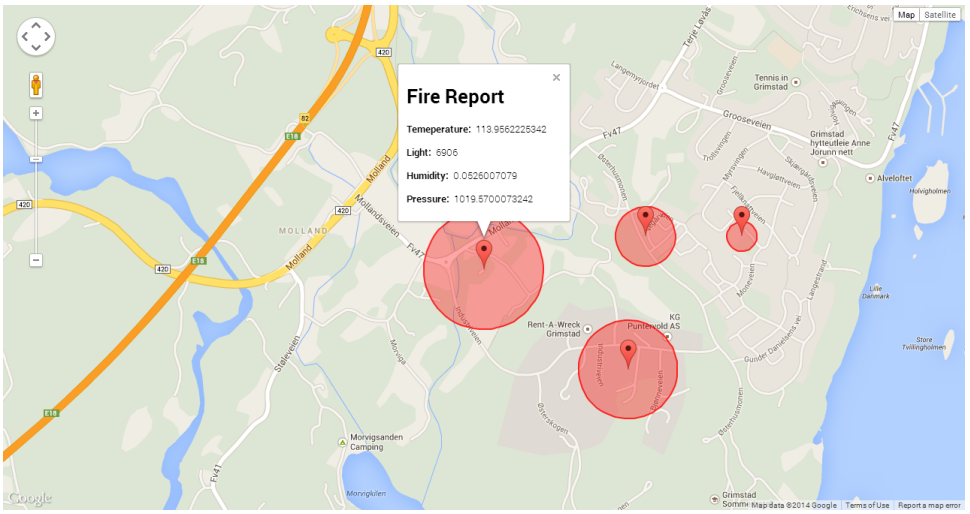


FIGURE 5.4: The visualizer web application showing the location and the seriousness of the fire reports. Circles diameter represent the seriousness of the report.

As illustrated in Figure 5.4 the GPS coordinates of the fire reports are used to draw four red circles on the map representing the places of the fire incidents. The diameters show the seriousness of the case calculated and sent by the smartphones based on the values of the ζ explained in Section 4.3. This map also provides users with some useful information about the temperature, humidity, light, and the air pressure sensed by the smartphones which have sent the fire report. These information are available by clicking on each circle.

5.3 Chapter Summary

In this chapter, implementation details of the components of the proposed framework were described. We saw how different components of the FireDetection framework interact each other to detect and visualize probable fires. Collecting data from smartphones' surroundings, preprocessing, analyzing, logging, and sending it to the FDBS machine, reasoning based on it and finally, visualizing it were among the FireDetection tasks highlighted in this chapter.

Chapter 6

Evaluation and Results

In this chapter, the results achieved by the Naive Bayes Classifier are offered. The results are expressed in a confusion matrix which shows the percentage of correctly classified and wrongly classified parameter sequences. A comparison between performance of the Naive Bayes Classifier with a few other classifiers is also made and its discussion can be found in this chapter.

6.1 FireDetection Performance

Reasoner module of the smartphones in FireDetection framework showed 76.3 percent accuracy in classification of normal cases and 89.7 percent in classification of fire cases. This results are given as a confusion matrix in Table 6.1.

TABLE 6.1: Confusion matrix representing the accuracy of the classification.

		Predicted Classes	
		Normal	Fire
Actual Classes	Normal	76.3	23.7
	Fire	10.3	89.7

To test the importance of the variance values, I removed all variance values from the training set and tested the reasoner again. As shown in Table 6.2 although the accuracy of the classification of normal cases improves, the classification of fire cases degrades to 70.7 from 89.7 percent. Considering that the accuracy of the system on detecting fire is more important to FireDetection framework, I decided to keep the variance values. We can compensate for the rather low accuracy of the classification of normal cases by exploiting the proposed risk calculation method explained in Section 4.3.

TABLE 6.2: Confusion matrix representing the accuracy of the classification while variance values are removed from the training set.

		Predicted Classes	
		Fire	Normal
Actual Classes	Fire	93.1	6.9
	Normal	29.31	70.7

6.1.1 Alternative Discretization Methods

The proposed discretization method is used in FireDetection framework, however, there are alternative discretization methods which can affect the classification accuracy. Data mining tools such as Weka [35] and Orange [36] offer different discretization methods. For example in the Orange framework, there are three discretization options: Entropy-MDL, equal-width, and equal-frequency. Studying the Entropy-MDL algorithm [37] is out of the scope of this thesis but the equal-length and equal-frequency method were considered when designing intervals. These two methods can be useful when the type of the data set and its characteristics are unknown to the classifying system, and from that perspective the classifier is a multi-purpose system which does not care about the data set semantics, whereas, the FireDetection framework is specialized to work with a constant training set and can only detect fire. Therefore, it does not need automatic discretization algorithm.

6.2 Alternative Classification Methods

There are many alternative classification methods that could be used in FireDetection framework. To see if they show better classification accuracy, I used the Orange data mining tool. The Orange provides many classifiers, but I chose three of them that are: Neural Network, k-NN, and Logistic Regression. As expected [38], all three show slightly better accuracy than the Naive Bayes Classifier but judging about their efficiency is difficult unless all classifiers are implemented on the smartphone.

6.3 Chapter Summary

In this chapter, the results achieved by the FireDetection framework were offered. We discussed the accuracy of the classification process and also the possible alternative methods and technologies that could have been engaged.

Chapter 7

Conclusion

In the present thesis, I introduced the FireDetection framework which detects indoor fire and visualizes its location and scale based on reports containing smartphone sensors data. The FireDetection is composed of the smartphones and the FDBS. The inference process on existence of fire is performed using Naive Bayes Classifier implemented inside the smartphones. The smartphones also send data to FDBS for visualization purposes. The implemented classifier, has been modified so that it can produce the ζ metric of fire which represents the seriousness of the case, i.e. the fire report. Under defined criteria, this metric along with GPS coordinates are plotted on a web based map. This framework is a learning system and has been trained in southern Norway. Therefore, its functionality is only valid in similar climates. The reason for this is that the reasoning process is based on parameters such as temperature, humidity, and air pressure, subject to change in different environmental conditions. As a conclusion, the highlights of the results obtained in this project are listed as follows:

Sensor Selection: It is now safe to state that currently available smartphones can feasibly infer on existence of fire in their surroundings. The classification (or inference) results offered in 6.1 and also the experiments presented in Sections 4.4 and 4.5 prove our hypothesis of fire detection using smartphone senors.

Reasoning: The reasoning process proposed in this thesis was also quite efficient both in term of results and battery consumption in smartphones. In total Naive Bayes Classifier gained 83.5 percent accuracy that considering the defined mechanism for triggering the alarm, is sufficiently reliable.

Experiments: The valuable opportunity of performing exclusive fire experiments in a real environment is definitely one of the strong points of this thesis. Highly accurate

results gained in the reasoning process approves the quality of performed experiments and motivates further research and development in this area.

Visualization: One of the most important goals of this project was to provide crisis management institutions with a real-time monitoring tool helping them to know more about a fire. The output of the visualizer web application is one step towards safer firefighting operations.

7.1 Future Work

Some of the possible improvements of the present framework are rooted in the limitations and assumptions we had to define at the beginning of the project. For instance, the framework has been designed and implemented to be used in Norway and using it in other countries with significant climate differences may lead to false results unless the training set is updated according to the new environmental parameters. Therefore, the first possible improvement of this framework can be the possibility of updating the training set directly from the smartphone user interface so that there will be no need to manipulate the code both in the smartphone and the FDBS.

The proposed framework can be extended to be used in outdoor locations as well. Monitoring forests to prevent probable fires from getting too large can be an interesting expansion of this system. But then, we need to replace the smartphone sensors with a huge number of sensors mounted (or dropped) on various places in the forest. Outdoor sensors must be a bit different in terms of thresholds, being water and dust resistant, etc. However, the principles of reasoning and fire detection seems to be the same.

Another potential functionality of the FireDetection framework is providing its users with survival hints. Currently, our smartphones can detect fire individually and the FDBS can visualize the fire according to the information received from them. Another reasoner residing in FDBS can analyze smartphones reports, find the best way to save more people, and send them messages containing useful information. This interactive system needs huge amount of prior information about the place of fire (emergency exits, stairs, etc.) and it demands lots of memory and processing power, but it can save lives.

During the data collection and experiment phases in this project, we noticed that the level of pressure and humidity are extremely dependent to the weather. These two parameter can have a wide range of values. For instance, we had pressures from 1001 to 1025 millibar in different days. It would be useful if the smartphone could access the

momentary mean values of the four parameters through an external web service. The main advantage of such a web service is that the smartphone can dynamically adapt its reasoning to the weather condition. This should be discussed in the future. Moreover, in this project we assumed that the data captured from smartphone senors are accurate and valid but in reality not all smartphones sense the same in the same condition. This happens because there are various sensor vendors and smartphone manufacturers. Although we did not have enough time to address this issue, a combination of the external web service and data quality techniques could reduce such impacts on the reasoning process.

Bibliography

[1] International Telecommunication Union. *The World in 2011: ICT Facts and Figures*. ITU, 2011.

[2] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela H Byers. Big data: The next frontier for innovation, competition, and productivity. 2011.

[3] C Dobre and F Xhafa. Intelligent services for big data science. *Future Generation Computer Systems*, 2013.

[4] Gartner. Annual smartphone sales surpassed sales of feature phones, March 2014. URL `http://www.gartner.com/newsroom/id/2665715`.

[5] Tudor Pascu, Martin White, and Zeeshan Patoli. Motion capture and activity tracking using smartphone-driven body sensor networks. In *Innovative Computing Technology (INTECH), 2013 Third International Conference on*, pages 456–462. IEEE, 2013.

[6] Yi He and Ye Li. Physical activity recognition utilizing the built-in kinematic sensors of a smartphone. *International Journal of Distributed Sensor Networks*, 2013.

[7] Lin Sun, Daqing Zhang, Bin Li, Bin Guo, and Shijian Li. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In *Ubiquitous intelligence and computing*, pages 548–562. Springer, 2010.

[8] Douglas Crockford. Json: The fat-free alternative to xml. In *Proc. of XML*, 2006.

[9] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 173–182. ACM, 2012.

[10] Sebastian Feese, Bert Arnrich, Gerhard Troster, Michael Burtscher, Bertolt Meyer, and Klaus Jonas. Coenofire: Monitoring performance indicators of firefighters in real-world missions using smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 83–92. ACM, 2013.

[11] Matthew Keally, Gang Zhou, Guoliang Xing, Jianxin Wu, and Andrew Pyles. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 246–259. ACM, 2011.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[13] Seyed Amir Hoseini-Tabatabaei, Alexander Gluhak, and Rahim Tafazolli. A survey on smartphone-based systems for opportunistic user context recognition. *ACM Computing Surveys (CSUR)*, 45(3):27, 2013.

[14] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1609–1615. IEEE, 2011.

[15] Mustafa Kose, Ozlem Durmaz Incel, and Cem Ersoy. Online human activity recognition on smart phones. In *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, pages 11–15, 2012.

[16] Armir Bujari, Bogdan Licar, and Claudio E Palazzi. Road crossing recognition through smartphone's accelerometer. In *Wireless Days (WD), 2011 IFIP*, pages 1–3. IEEE, 2011.

[17] John J Guiry, Pepijn van de Ven, and John Nelson. Classification techniques for smartphone based activity detection. In *Cybernetic Intelligent Systems (CIS), 2012 IEEE 11th International Conference on*, pages 154–158. IEEE, 2012.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[20] Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 2006.

[21] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[22] Mehmed Kantardzic. Data-mining concepts. *Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition*, pages 7–8, 2011.

[23] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[24] Tzu-Tsung Wong. A hybrid discretization method for naïve bayesian classifiers. *Pattern Recognition*, 45(6):2321–2325, 2012.

[25] National Wildfire Coordinating Group (US). *Fire Effects Guide*. National Wildlife Coordinating Group, 1994.

[26] Kevin McGrattan, Simo Hostikka, Randall McDermott, Jason Floyd, Craig Weinschenk, and Kristopher Overholt. Fire dynamics simulator, technical reference guide, volume 2: Verification. *National Institute of Standards and Technology, Gaithersburg, Maryland, USA, and VTT Technical Research Centre of Finland, Espoo, Finland*, 2, 2013.

[27] Safemar AS. Safemar as-home, May 2014. URL `http://www.safemar.no/`.

[28] H Wayne Beaty. Units, symbols, constants, definitions, and conversion factors. *Standard Handbook for Electrical Engineers*, 2006.

[29] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of physics extended*. John Wiley & Sons, 2007.

[30] Android. Sensors overview — android developers, April 2014. URL `http://developer.android.com/guide/topics/sensors/sensors_overview.html`.

[31] D Richard Hipp and D Kennedy. Sqlite, 2007.

[32] Android. Storage options — android developers, May 2014. URL `http://developer.android.com/guide/topics/data/data-storage.html`.

[33] Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive bayes. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann Publishers Inc., 2002.

[34] Google Inc. Google maps javascript api v3, April 2014. URL `https://developers.google.com/maps/documentation/javascript/tutorial`.

[35] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[36] Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. *Orange: From experimental machine learning to interactive data mining*. Springer, 2004.

[37] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *KDD*, pages 114–119, 1996.

[38] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.