Títol: Data classification system for the citizens science platform
Mosquito Alert

Autor: Antonio Rodriguez Garcia

Data de defensa: 22/04/2016

Director: Frederic Bartumeus Ferré (CEAB-CSIC)

Ponent: Ricard Gavaldà Mestre (CS-FIB-UPC)

Titulació: Màster en Enginyeria Informàtica

Centre: FACULTAT D‘INFORMÀTICA DE BARCELONA (FIB)

Universitat: UNIVERSITAT POLITÈCNICA DE CATALUNYA
(UPC) – BarcelonaTech
Data classification system for the citizens science platform *Mosquito Alert*

Antonio Rodriguez Garcia

April 2016
Abstract

The *Mosquito Alert* project is an expert-validated citizen science platform for tracking and controlling disease-carrying mosquitoes. The project has been running for the past two years under the name AtrapaelTigre and focused on Spain. They are now trying to scale up rapidly to offer a global tool that will aid in the fight against Zika, Chikungunya, Dengue, and Yellow Fever. In the past two years they have built up a community of stakeholders in Spain, allowing them to bring together citizen scientists, academic/professional scientists (modelers and medical entomologists), and public health administrations to help minimize the mosquito-borne disease risk in this country. They have a solid multidisciplinary team in place and they are now beginning to work with interested groups and stakeholders in Latin America, the United States, and China.

They have had almost 20,000 people download their mobile phone applications and register as participants, and they expect this number to increase rapidly in light of the global attention to Zika in particular. With the mentioned applications citizens can help scientists detect adult mosquitoes and their breeding grounds. Users of the app can send a report of an spotted adult mosquito or breeding site. When sending the report to the project servers they answer a questionnaire, can attach a photography and specify the geolocalization of that report. These reports are validated by entomologists and are included in a database.

The main goal of this Master Thesis is to design an expert system able to, using machine learning techniques, predict if a new report sent to the servers is valid or not. This will allow the entomologists to focus in other more valuable tasks, such as verifying new reports from areas where the specimens have not yet been established, or expanding and improving a training set of pictures from time to time.

To do so various tasks have been performed, exploratory data analysis, data preparation and transformation, building and evaluating a number of classifiers, and designing a classification system that uses the classifiers and can be integrated in the existing information systems.


# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Context</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Methodology</td>
<td>15</td>
</tr>
<tr>
<td>1.3.1 Exploratory data analysis</td>
<td>15</td>
</tr>
<tr>
<td>1.3.2 General data cleaning and pre-processing</td>
<td>15</td>
</tr>
<tr>
<td>1.3.3 Classifier training, evaluation and selection</td>
<td>15</td>
</tr>
<tr>
<td>1.3.4 Real-time classification system design</td>
<td>16</td>
</tr>
<tr>
<td>1.4 Project development</td>
<td>17</td>
</tr>
<tr>
<td>1.4.1 Exploratory data analysis</td>
<td>17</td>
</tr>
<tr>
<td>1.4.2 Initial data samples</td>
<td>17</td>
</tr>
<tr>
<td>1.4.3 Data structure</td>
<td>17</td>
</tr>
<tr>
<td>1.4.4 Analysis</td>
<td>19</td>
</tr>
<tr>
<td>1.4.5 Final data samples</td>
<td>26</td>
</tr>
<tr>
<td>1.4.6 Data cleaning and pre-processing</td>
<td>29</td>
</tr>
<tr>
<td>1.4.7 Cleaning and consolidating data</td>
<td>29</td>
</tr>
<tr>
<td>1.4.8 Instance generation</td>
<td>29</td>
</tr>
<tr>
<td>1.4.9 Classifier training, evaluation and selection</td>
<td>32</td>
</tr>
<tr>
<td>1.4.10 Studied classifiers</td>
<td>32</td>
</tr>
<tr>
<td>1.4.11 Classifier training and evaluation</td>
<td>35</td>
</tr>
<tr>
<td>1.4.12 Experiments and results</td>
<td>37</td>
</tr>
<tr>
<td>1.4.13 Selected classifiers</td>
<td>45</td>
</tr>
<tr>
<td>1.4.14 Real-time classification system design</td>
<td>46</td>
</tr>
<tr>
<td>1.4.15 Instance generation system</td>
<td>46</td>
</tr>
<tr>
<td>1.4.16 Classification system</td>
<td>48</td>
</tr>
<tr>
<td>1.5 Discussion</td>
<td>49</td>
</tr>
<tr>
<td>1.5.1 Selected classifier</td>
<td>49</td>
</tr>
<tr>
<td>1.5.2 Improving performance</td>
<td>49</td>
</tr>
<tr>
<td>1.5.3 Important variables</td>
<td>50</td>
</tr>
<tr>
<td>1.5.4 System design</td>
<td>50</td>
</tr>
</tbody>
</table>
5.2.1 noSQL in-memory database .......................... 50
5.3 Report validation priority .............................. 51

6 Conclusions ............................................. 53
6.1 Goal review ............................................ 53
   6.1.1 General .......................................... 53
   6.1.2 Specific .......................................... 53
   6.1.3 Academic ......................................... 54
6.2 Personal evaluation .................................... 54

7 Future work ............................................. 55
7.1 Scalability .............................................. 55
   7.1.1 Code modifications .............................. 55
   7.1.2 GIS enabled database ............................ 56
   7.1.3 Considerations on computational resources ...... 56
7.2 Additional improvements .............................. 56
   7.2.1 Classifier tuning ................................. 56
   7.2.2 Priority system ................................. 56
   7.2.3 Another classifier ............................... 57

A Naive Bayes Classifier for Discrete Predictors ........ 61
List of Figures

4.1 Registered users per day, sample between June and December 2014 20
4.2 Registered users per month, sample between June and December 2014 21
4.3 Registered users by week day, sample between June and December 2014 22
4.4 Reports per day, sample between June and December 2014 24
4.5 Reports per month, sample between June and December 2014 24
4.6 Total reports per day of the week, sample between June and December 2014 25
4.7 Geolocated reports over various amplifications of Europe’s area. 27
4.8 Total reports per user, sample between June and December 2014 28
4.9 Designed system for real time instance generation. 32
4.10 Example of k-NN classification. 34
4.11 Rweka decision tree (M=3), 2015 dataset (grouped instances) 42
4.12 Designed real-time classification system. 47

5.1 Class distribution for report answers variables. 52
List of Tables

4.1 Reliability class proportion of 2014 instances. . . . . . . . . . . . 36
4.2 Reliability class proportion of 2015 instances. . . . . . . . . . . . 36
4.3 Confusion matrix . . . . . . . . . . . . . . . . . . . . . . . . . . 36
4.4 Confusion matrix, Naive Bayes, 2015 dataset (all instances) . . . 38
4.5 Confusion matrix, k-nearest neighbors (k=7, numfolds=10), 2015
dataset (all instances) . . . . . . . . . . . . . . . . . . . . . . . . 39
4.6 Confusion matrix, Rweka decision tree (M=14, numfolds=10),
2015 dataset (all instances) . . . . . . . . . . . . . . . . . . . . . 39
4.7 Confusion matrix, tree library, 2015 dataset (all instances) . . . 40
4.8 Confusion matrix, randomForest, 2015 dataset (all instances) . . 40
4.9 Class proportion of 2015 grouped instances. . . . . . . . . . . . . 41
4.10 Confusion matrix, Naive Bayes, 2015 dataset (grouped instances) 41
4.11 Confusion matrix, Naive Bayes, 2015 dataset (grouped instances) 41
4.12 Confusion matrix, Rweka decision tree (M=1, numfolds=10),
2015 dataset (grouped instances) . . . . . . . . . . . . . . . . . . 42
4.13 Confusion matrix, Rweka decision tree (M=3, numfolds=10),
2015 dataset (grouped instances) . . . . . . . . . . . . . . . . . . 42
4.14 Confusion matrix, tree library, 2015 dataset (grouped instances) 43
4.15 Confusion matrix, randomForest, 2015 dataset (grouped instances) 43
4.16 Class proportion of 2015 grouped instances testing set. . . . . . 43
4.17 Class proportion of 2015 grouped instances training set with repli-
cated negative instances. . . . . . . . . . . . . . . . . . . . . . . 43
4.18 Confusion matrix, Naive Bayes, 2015 dataset (grouped instances,
replicated negatives x10) . . . . . . . . . . . . . . . . . . . . . . . 44
4.19 Evaluation metrics, Naive Bayes, 2015 dataset (grouped instances,
replicated negatives x10) . . . . . . . . . . . . . . . . . . . . . . . 44
4.20 Confusion matrix, Random Forest, 2015 dataset (grouped in-
stances, replicated negatives x10) . . . . . . . . . . . . . . . . . . 44
4.21 Evaluation metrics, Random Forest, 2015 dataset (grouped in-
stances, replicated negatives x10) . . . . . . . . . . . . . . . . . . 44
4.22 Confusion matrix, Rweka decision tree (M=10, numfolds=10),
2015 dataset (grouped instances, replicated negatives x10) . . . . 45
4.23 Evaluation metrics, Rweka decision tree (M=10, numfolds=10),
2015 dataset (grouped instances, replicated negatives x10) . . . . 45
5.1 Confusion matrix, Naive Bayes, 2015 data-set (grouped instances,
replicated negatives x10, laplace smoothing), best case. . . . . . . 49
5.2 Evaluation metrics, Naive Bayes, 2015 data-set (grouped instances, replicated negatives x10, laplace smoothing), best case. . . . . . 50
Chapter 1

Introduction

1.1 Motivation

During the masters degree I enrolled some courses that increased my interest in data science.

The main subject which especially motivated me was Computing and Intelligent Systems (CSI-MEI). In this course, I learned various ways to understand the data beyond simple inputs to programs, but as more complex concepts. I also appreciated the utility of statistics when solving problems that seek higher knowledge of the one obtained using informatics.

The other course that motivated me to approach the more scientific, in fact biological, aspects of informatics was Techniques and Tools for Bioinformatics (TEB-MEI), in which algorithmic concepts were used to perform analysis of DNA patterns.

Lastly, the fact I was doing internship at inLab-FIB collaborating in traffic simulation research projects has lead me to, at least, want to explore the way informatics can be put at the service of science and research.

For these reasons this thesis proposal caught my eye, since its scope and the required techniques and methodology are in line with my actual concerns.

The structure of this document is the following:

In this chapter the motivation and context of the Master Thesis have been presented. Chapter 2 presents the goals to be achieved during the project realization. Chapter 3 describes briefly the phases of the project and the methodologies used. Chapter 4 is the main chapter of this document and provides the details of the tasks performed in every phase and the results obtained. A discussion of these results is presented in Chapter 5 followed by the conclusions in Chapter 6. Finally Chapter 7 presents future lines of work.

1.2 Context

This work is focused on data from the citizens science platform Mosquito Alert developed at at CREALF (Center for Research and Ecological Applications) and the Center for Advanced Studies of Blanes (CEAB-CSIC\textsuperscript{1}) under the direction

\textsuperscript{1}Spanish Research Council
of the ICREA Research Professor Frederic Bartumeus. *Mosquito Alert* is an expert-validated citizen science platform for tracking and controlling disease-carrying mosquitoes. The platform has been running for the past two years under the name AtrapaelTigre and focused on Spain. They are now trying to scale up rapidly to offer a global tool that will aid in the fight against Zika, Chikungunya, Dengue, and Yellow Fever. In the past two years they have built up a community of stakeholders in Spain, allowing them to bring together citizen scientists, academic/professional scientists (modelers and medical entomologists), and public health administrations to help minimize the mosquito-borne disease risk in this country. They have a solid multidisciplinary team in place and we are now beginning to work with interested groups and stakeholders in Latin America, the United States, and China.

They have had almost 20,000 downloads of the associated app, and several thousands of continuous participants in average. They expect this number to increase rapidly in light of the global attention to Zika in particular. With the mentioned applications citizens can help scientists detect adult mosquitoes and their breeding grounds. This methodology is called citizen science that refers to involving the general public in scientific research, which can include sharing knowledge, intellectual abilities, resources, or tools[2].

More precisely, the users of the app can send a report of an spotted adult mosquito or breeding sites (i.e. small water containers like flower pots in terraces and gardens, and in the public space fountains, sewers or water drainers). When sending the report to the platform servers they answer a questionnaire, can attach a photography and specify the geolocalization of that report. These reports are validated by entomologists and are included in a database. This database is the main data source for this Master Thesis, one of its main objectives is to analyze it using various techniques that allow the extraction of further knowledge which at its time will allow to develop the platform or making better informed decisions.

Specially, the main goal of this Master Thesis is to design an expert system able to, using machine learning techniques, predict if a new report sent to the servers is valid or not. This will allow the entomologists to focus in other more valuable tasks, such as verifying new reports from areas where the specimens have not yet been established.
Chapter 2

Goals

2.1 General

1. To develop a data processing system to extract knowledge of the data generated by the users of the described application and the reports they send.

2. To analyze various machine learning algorithms (classifiers) and select the most useful to identify valid reports.

3. To design a system architecture that integrates them into the main project system.

2.2 Specific

1. Perform a proper data analysis.

2. Train and evaluate classifiers.


2.3 Academic

1. Acquire knowledge about R programming language.

2. Learn new data-analysis techniques and consolidate the ones already known.

3. Put into practice the knowledge acquired during the master degree.
Chapter 3

Methodology

The work done in this Master Thesis can be divided in four main activities. This short chapter describes every one of them and the tasks done. In other words describes how the problem has been solved in a bottom-top way from analyzing the raw data to designing the classification system.

3.1 Exploratory data analysis

The first part of the project consists in an exploratory analysis of the provided data that allows the identification of relevant variables or trends for its further study. The used methodology goes from simple observation to application of basic statistical methods. It also allows to design the proper data structures or cleaning procedures developed in the next part of the project.

3.2 Data cleaning and pre-processing

As the data is not always presented in an structured and formatted way that allows its immediate use, there is a need for selecting the valid data and preparing it for its use. In this section all aspects related with these topics are covered. First the cleaning and consolidation of the data and finally, one of the most important outputs of this research work is described, the real-time instance generation system.

3.3 Classifier training, evaluation and selection

When analyzing the problem to solve from a more general point of view it is easy to conclude that its a classification problem. For each report the goal is to determine if it is valid or not, or in other words, if it is a correct report (corresponds to a true occurrence of the invasive species) or an incorrect one. This, in fact, is a little bit more complex since the previously classified reports are grouped into 4 classes, but this aspect will be covered in the following sections.

There is an specific kind of machine learning algorithms, called classifiers, that address this type of problems. These classifiers take as input a set of
classified elements and its particularity is that the output is another “algorithm” that is able to predict the class of an element that was not in the first set. This first set is called training set and consists in a subset of the Mosquito Alert reports described before. These sample reports have been classified by entomologists and the Mosquito Alert project directors. So the goal is to “train” a classifier with this data in order to classify future reports accurately.

In this section, various classifiers will be presented along with the reasons for its selection for study. After that, training and evaluation methodologies are described, first the various subsets of instances used for training the classifiers and next the validation methodology. Finally the most relevant results are presented and, according to them, the classifier that best suits the project needs is selected.

3.4 Real-time classification system design

Finally, the main goal of this project is to design a classification system that can be integrated in the platform Mosquito Alert. In this section the design of this system is proposed and described along with all its particularities.
Chapter 4

Project development

4.1 Exploratory data analysis

As commented before this section explains how the analysis of the project data structure has been performed. At the beginning of the project some data samples regarding 2014 were disclosed to begin its analysis. After that, at the beginning of 2016, a new dataset was disclosed containing 2015 data which have been used during the next stages of the Master Thesis. This section presents the data analysis performed with 2014 data and adds comments for the final dataset including both, 2014 and 2015, datasets.

4.1.1 Initial data samples

The first set of data analyzed consists in a sample of two tables of the Mosquito Alert database. These data describes the registered users between June and December 2014 and the reports sent by them.

4.1.2 Data structure

The two samples are presented in CSV\(^1\) files that have been renamed to users2014.csv and reports2014.csv. In order to analyze these data in a fast but effective way I used some of the R programming language embedded functions. To start I loaded the data to R objects and extracted various metrics of the samples.

```
''users2014.csv'' sample
```

First, the users sample contains 6890 observations of 10 variables which are explained in text files obtained along the samples. The names and definitions of these variables are the following:

- **userID** Univocal identifier for each user after deleting the users registered prior the release of the app for Android\(^2\) (June, 13 2014) as for iOS\(^2\) (June, 24 2014).
- **userRegistTimeOriginal** Original time for users sign in.

---

\(^1\)Comma Separated Values
\(^2\)Mobile device Operating System
userRegistDatetime  Date and time for users sign in.

userRegistDate  Date for users sign in.

userRegistMonthNum  Number corresponding to the month of users sign in.

userRegistMonthString  Name, in spanish, of the users sign in month.

userRegistWeekdayString  Name, in spanish, of the users sign in day of the week.

userRegistWeekdayNum  Number corresponding to the users sign in day of the week considering Sunday as 0.

userSyst  Operating system of the users mobile device.

userDaysSystRelease  For each user, time between app release for user’s platform and users sign in.

‘‘reports2014.csv’’ sample

The reports sample contains 2979 observations of 20 variables. Some of the data are marked as not validated and others are directly not well formed thus, invalid. The variables are described as the following:

reportVersionID  Unique identifier for a report version. As indicated in the files provided, only the last version of a report is included in the data set.

reportVersionNum  Version number of a report. The code is based in integer numbers increasing along more editions are performed over a report. So the first version is represented with the number 0 and it will be increasing this value by 1 for every edition. In case the report is deleted, even if it has been edited, the value holds −1[3].

userID  Identifier for the user that has sent the report. Corresponds to the users sample identifiers.

reportID  Unique report identifier.

reportType  Type of report (either adult mosquito or breeding site).

reportNote  The comments, made by the user, attached to a report.

os  Operating system of the mobile device form which the report has been sent.

hide  Field indicating if a report must be hidden.

reportCreationDatetime  Original creation time for a report.

reportCreationDate  Date when a report was submitted.

reportVersionDatetime  Original time of creation of a report version.

reportVersionDate  Date when a report version was created.
4.1. EXPLORATORY DATA ANALYSIS

**reportCreationMonthNum** Number representing the month a report was submitted.

**reportCreationMonthString** Name, in Spanish, of the month a report was created.

**reportCreationWeekdayString** Name, in Spanish, of the week day a report was created.

**reportCreationWeekdayNum** Number corresponding to a report creation day of the week considering Sunday the first as 0.

**reportLong** Geographical longitude for an issued report. May contain the value −9999 to indicate the current report is not geolocalized.

**reportLat** Geographical latitude for an issued report. May contain the value −9999 to indicate the current report is not geolocalized.

**missionNum** Numerical identifier for a mission to which the report belongs. The value −9999 states the report doesn’t belong to any mission.

**missionName** Text identifier for the mission to which the report belongs. Can be NULL.

**class** The class assigned to the report by the experts.

### 4.1.3 Analysis

Once described the sample data structure, relevant characteristics of the variables conforming it can be analyzed as well as some relations between them. Descriptive statistical analysis allows to have a better understanding of the date topology and to identify elements that require to be treated to perform further analysis.

```
"users2014.csv" sample
```

**userID** The first variable of this sample is **userID** representing the user identifiers. Each identifier is conformed by 5 sets of hexadecimal values separated by the character “-”. More precisely the sizes of the sets are 8, 4, 4, 4 i 12 respectively. As a detail the identifiers of iOS users are placed in capitals while the Android ones are in lowercase.

Another mandatory verification is to test the absence of any of them, but as expected there aren’t missing values. This test is important because this variable is one of the main relationships to the reports set.

One last verification is to ensure the uniqueness of the identifiers. To prove that, the total number of registers has been compared to the one after repeated deletion and they are the same. Indeed there are 6890 registers with unique values in the **userID** field.

**userRegistTimeOriginal** The next variable is **userRegistTimeOriginal** describing the original sign in time for each user. It’s an **ISO 8601**\(^3\) date format.

\(^3\)https://en.wikipedia.org/wiki/ISO_8601
CHAPTER 4. PROJECT DEVELOPMENT

timestamp. This has to be considered while working with these values. A prior analysis with R displays the following:

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;2014−06−13,01:35:08&quot;</td>
<td>&quot;2014−06−24,08:40:15&quot;</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>&quot;2014−07−01,16:42:11&quot;</td>
<td>&quot;2014−07−24,04:35:42&quot;</td>
<td></td>
</tr>
<tr>
<td>3rd Qu.</td>
<td></td>
<td>Max.</td>
</tr>
<tr>
<td>&quot;2014−08−02,04:03:50&quot;</td>
<td>&quot;2014−12−31,13:51:47&quot;</td>
<td></td>
</tr>
</tbody>
</table>

The more relevant aspects are the minimum and maximum values which allow to verify that, indeed, the sample users signed in to the app after the first release day (2014-06-13) and there are no 2015 records.

**userRegistDatetime** Contains the same information than the previous field but in another precomputed format in order to increase efficiency.

**userRegistDate** This variable is similar to the previous one and stores only the sign in date without the time. This field allows to generate figure 4.1 to observe the register distribution along the studied period.

![Figure 4.1: Registered users per day, sample between June and December 2014](image)

Observing the figure is possible to identify certain dates when a lot of users have registered.

**userRegistMonthNum** As commented, corresponds to the sign in month number for each user. A prior analysis shows the following:
4.1. EXPLORATORY DATA ANALYSIS

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.000</td>
<td>6.000</td>
<td>7.000</td>
<td>7.125</td>
<td>8.000</td>
<td>12.000</td>
</tr>
</tbody>
</table>

The limits are June (6) and December (12). It can be noted the most part of the registers where originated at the beginning of the sample period. To clarify this, the number of registers per month is displayed next along with the plot 4.2 of the same data:

<table>
<thead>
<tr>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>3399</td>
<td>1720</td>
<td>579</td>
<td>346</td>
<td>454</td>
<td>330</td>
<td>62</td>
</tr>
</tbody>
</table>

Figure 4.2: Registered users per month, sample between June and December 2014

**userRegistMonthString** Stores the same information that **userRegistMonthNum** but formatted as a lowercase string containing the month name in Spanish.

**userRegistWeekdayString** and **userRegistWeekdayNum** Both state for the register day of the week of each user in string and numeric format respectively, as described before.

Using this data is possible to observe the total number of user registers for every day of the week and plot the figure 4.3. The 2014 week day with more registers was Monday (1).

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>926</td>
<td>2415</td>
<td>1000</td>
<td>720</td>
<td>506</td>
<td>504</td>
<td>819</td>
</tr>
</tbody>
</table>
userSyst Is possible to observe the proportion of users using each of the operating systems. There are two possible options being Android the most used by 5434 (78.9%) of the 6890 registered users while the other 1456 (21.1%) use iOS.

userDaysSystRelease The mean time for users to download the application is 38.27 days. There are no major differences between Android and iOS users.

'reports2014.csv' sample

reportVersionID Has a similar format than user identifiers (five groups of hexadecimal characters separated by minus signs (-) of sizes 8, 4, 4, 4 and 12 respectively). There are no missing values.

reportVersionNum Is an integer describing the total number of times a report has been modified or if it has been deleted. Allows to obtain the number of reports modified one or more times.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2527</td>
<td>438</td>
<td>11</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Therefore of a total number of 2979 reports there are 452 (15.17%) of them that have been modified at least once. Most of them only one time.
4.1. EXPLORATORY DATA ANALYSIS

**userID** Corresponds to the user identifiers previously described. There are 1388 unique values so, only the 20,1% of the total 6890 registered users has sent reports.

**reportID** A set of 4 alphanumeric characters. There are no missing values.

**reportType** Is interesting to observe the different report types and know their proportion.

<table>
<thead>
<tr>
<th></th>
<th>adult</th>
<th>mission</th>
<th>site</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportion</td>
<td>0.58006042</td>
<td>0.34776771</td>
<td>0.07217187</td>
</tr>
</tbody>
</table>

There are three report types:

- **adult** Is the default report. Corresponds to an adult specimen.
- **mission** These are reports submitted by an explicit request of the project directors.
- **site** Is a report corresponding to a breeding site.

**reportNote** Contains the comments users have attached to the reports. Of 2979 reports there are 641 (21,5%) that have information in this field.

**os** Operating system from which a report has been sent. There are no missing values. The following are the proportion of each operating system over all reports.

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>iPhone OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportion</td>
<td>0.8620342</td>
<td>0.1379658</td>
</tr>
</tbody>
</table>

Considering the raw data, **Android** users (representing 78.9%) send slightly more reports (86.2%) than **iOS** ones (13.8% of the reports sent by 21.1% of the users).

A priori i a “ull” els usuaris d’**Android** (recordem que representen un 78.9%) envien lleugerament més informes (86,2%) que els de **iOS** (13,8% dels informes enviats pel 21,1% dels usuaris). The mean of reports sent per **iOS** user is 0.28 while Android ones send a mean of 0.47 reports per user. Thus, Android users sent reports average is higher.

**reportCreationDatetime** All fields related to time stamps can be analyzed by observing the plot in the figure 4.4. Note that, for some reason, there is an increase in the number of sent reports in November.

**reportCreationMonthNum** Number of the month of a report creation. Figure 4.5 shows the number of sent reports per month. August was the month when more reports were submitted.

**reportCreationWeekdayNum** Number corresponding to a report creation day of the week considering Sunday the first as 0.

Figure 4.6 allows to conclude that the day of the week is not very relevant to the number of submitted reports.

**reportLong** and **reportLat** There are 1043(35%) of a total of 2979 registers that are not geolocated. The analysis of the two variables shows the following for the latitude:
Figure 4.4: Reports per day, sample between June and December 2014

Figure 4.5: Reports per month, sample between June and December 2014
4.1. EXPLORATORY DATA ANALYSIS

Figure 4.6: Total reports per day of the week, sample between June and December 2014

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>−32.54</td>
<td>39.88</td>
<td>41.39</td>
<td>40.38</td>
<td>41.63</td>
<td>62.95</td>
</tr>
</tbody>
</table>

And for the longitude:

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>−123.1</td>
<td>0.04337</td>
<td>2.050</td>
<td>0.05710</td>
<td>2.539</td>
<td>126.9</td>
</tr>
</tbody>
</table>

The values of the geolocated reports are between correct and reasonable margins and looking at the medians is possible to identify that the data collection area is over Catalonia approximately. To make a first analysis easier the plots in Figure 4.7 show the reports over various maps.

**missionNum, missionName** There are 943 reports with -9999/NUL values in these fields. Therefore a 34.8% of the reports are missions distributed in the following way:

<table>
<thead>
<tr>
<th>#Coche</th>
<th>Murcia</th>
<th>Photo</th>
<th>Quiz</th>
<th>Share</th>
<th>Tigafotos</th>
</tr>
</thead>
<tbody>
<tr>
<td>240</td>
<td>16</td>
<td>347</td>
<td>143</td>
<td>201</td>
<td>89</td>
</tr>
</tbody>
</table>

**class** This variable represents the classification of the report. It can take 5 possible values:

- **-2** The report is definitely not a valid specimen.
- **-1** The report doesn’t seem to be a valid specimen. But is not sure.
There isn’t enough information to classify the report.

1. The report seems to be a valid specimen. But is not sure.
2. The report is definitely a valid specimen.

This variable is very important in this study and will be covered in depth in Section 4.3.2 where its particularities will be more contextualized.

Finally the most naive data crossing is to relate reports and users. Figure 4.8 allows to observe the number of reports sent by each user. The maximum number of reports sent by one user is 38 and is possible to observe that a reasonable low number of users sent the major part of the reports.

4.1.4 Final data samples

The initial sample was useful to analyze the structure of the data and understand the topology of the problem. Most of the scripts and data structures used in the project had been developed using this sample as source of data. Lately they’ve been slightly modified to meet the requirements of the final samples. These final samples, which were disclosed after the end of 2015, contain all the information of the initial ones and are very similar in structure to the ones first provided. There are in total 16967 users and 10618 reports along with some new fields such as the answers of the users to taxonomic questions about the reported specimen. There is a difference in the methodology used to classify the first data samples and the final ones and this has to be considered in the future steps of the work.
4.1. EXPLORATORY DATA ANALYSIS

(a) Geolocated reports over Catalonia area.

(b) Geolocated reports over Spain area.

(c) Geolocated reports over Europe area.

Figure 4.7: Geolocated reports over various amplifications of Europe’s area.
Figure 4.8: Total reports per user, sample between June and December 2014
4.2 Data cleaning and pre-processing

This section describes the procedures that have been applied to the raw data to obtain the final input features for the classifiers training.

4.2.1 Cleaning and consolidating data

As explained before, the original dataset includes two separate csv files, one for the users data and one for the reports. These files are mere dumps of the database tables in plain text and contain information that is not useful for this study, they also contain records with missing data. Moreover, the format of some fields is not the best most suitable to work with.

Along with the data two R scripts were to pre-process a bit Mosquito Alert data. These scripts have been revised and adapted to work with the structure of the dataset and perform the following actions.

“generate_clean_users2015.R”
- Format date and time.
- Delete not valid registers such as users registered before the release.

“generate_clean_reports2015.R”
- Format date and time.
- Create month and weekdays for creation date.
- Format and consolidate geolocation coordinates.
- Delete not valid registers such as reports submitted before the release, the ones submitted for testing purposes and the ones marked as deleted.
- Select only the last version of each report.

Both scripts generate the files all_users.csv and all_reports.csv respectively that can be used to feed the instance generation script presented in the next section.

4.2.2 Instance generation

Classification algorithms take as input a set of already classified elements. These elements contain a set of properties, or variables, that define each of them. The values hold by these variables are the ones used by the classifier to determine if the element belongs to a class or to another. In this project the elements are the reports, and they have a set of variables produced during its creation. This variables have been described and analyzed in Section 4.1.

A first simple approach would be to take the raw reports and train the classifiers with them. Although this exercise has been done, the results are not quite good, as expected. This is because the raw variables don’t contain much more knowledge than the represented by their own values, furthermore there are no trivially identifiable patterns. So, since raw report variables are not enough to train good classifiers, the need of more complex and meaningful variables
arises. By adding these new variables to the classifiers input their performance improves. Therefore instead of training the classifiers with the raw reports it is necessary to create an extended structure containing the raw variables plus more elaborated and significant ones. This structure has been called instance.

**Added variables**

The second-order, or more processed, variables added to an instance during its generation are:

- **reportNote** Is a boolean indicating if the report has a comment or not.
- **reportTimeOfDay** Is a discretization of the report timestamp. It’s obtained by dividing the hour the report has been generated by 4. Represents one of the four 6-hour intervals of a day starting from 0 for the time interval 00:00-05:59 and ending in 3 (18:00-23:59).
- **newUser** Is another boolean indicating if the report is the first one sent by the users who submitted it.
- **userNumReports** Is the total number of reports the user has submitted since the current one. The current selected report is not considered.
- **userAccuracy** Is the number of reports sent by the user and classified as valid over the total number of reports of the same user.
- **userTimeForFirstReport** Is the amount of time, in seconds, elapsed between the user sign in and the submission of the first report.
- **userTimeSinceLastReport** Is the amount of time, in seconds, between the last report sent by the user and the current one.
- **userMeanTimeBetweenReports** Is the mean time between every two consecutive reports sent by the user.
- **userNumActionAreas** In Mosquito Alert platform there is a grid that divides the geographical space of the study in cells. These cells are squares about 4 kilometers per side and they are used in this project as reference system. The variable **userNumActionAreas** is the number of these cells from which the user has sent at least one report independently of its classification.
- **userMobilityIndex** This variable tries to express the user mobility between cells and its activity in each of them. A user that always sends reports from a specific cell but has sent only one report from a different cell is definitively different from one who sends the same amount of reports from two different cells. In an effort to express this user movement activity the variable has been computed as the standard deviation of the number of reports sent from every different cell the user has been active on.
- **reports1kmLast** There are four variables that represent the number of reports sent in a circular area of 1 kilometer around the current report in an increasing time window in the past. The used variables use time windows of one hour, one day, one week and one month. They are named: reports1kmLastHour, reports1kmLastDay, reports1kmLastWeek and reports1kmLastMonth.
validReports1kmLast* There are four more variable of the same form as the previous ones described but containing the proportion of valid reports in the same conditions. In other words, the number of valid reports over the total reports in a 1 kilometer area around the current report during a certain time in the past. They are named: validReports1kmLastHour, validReports1kmLastDay, validReports1kmLastWeek and validReports1kmLastMonth.

Also some of the raw variables are dispensable and have not been included to the generated instances. The preserved ones are:

os The operating system of the device from which the report has been generated.

reportMonth The previously described reportCreationMonthNum renamed.

reportQ1Answ, reportQ2Answ, reportQ3Answ These three variables contain the responses of the user to the three taxonomic questions of the questionnaire. The three questions are the following:

- Is small, black and has white stripes?
- Has a white stripe in both head and thorax?
- Has white stripes in both abdomen and legs?

For each question the user can select one of three options (No/I don’t know/Yes) represented by the integer numbers -1, 0 and 1 respectively.

class This is the reliability class entomologists assigned to the report and has one of the possible values described before.

So, finally an instance contains 23 variables and an additional one defining the reliability class.

**Instance context**

The described new variables provide more knowledge to an instance making it more meaningful. But most of them contain implicit contextual information, such as the one provided by reports that have been submitted previously. For this reason, when creating instances, it is necessary to maintain contextual information.

This contextual information has been classified in two types: user context and environmental context. The user context contains, for every user, the necessary information to compute the described variables, for example, various time periods, the list of cells visited and other metrics. It is, in some way, an anonymous profile for each user. On the other hand the environment context represents the report distribution over the geographical space considering also the time and the reliability classification.

Using this contextual information is possible to generate the instances as described. Importantly, this task has to be performed in real time, or at least in chronological order, for each report submitted to the system. The reason is, of course, the context is time dependent and to properly generate a report instance it has to be generated after the previous report sent chronologically and before
the next one. To achieve this goal all the necessary data structures and scripts have been implemented in Python. The figure 4.9 represents this system.

The workflow of the system starts when a report arrives. The latter is provided as input to the script create_instance.py which initializes the necessary data structures for the instance creation and obtains the raw variables directly from the report dataset. After this it uses the functions provided by classes users.py and environment.py to obtain the other context dependent variables and complete the instance. Finally again uses the functions of the other two classes to provide the new information extracted by the current report in order to maintain the context updated.

**Generated instances**

Using the described procedure, a set of 1787 instances, representing all the reports sent during 2015 for which an instance has been possible to create, has been generated. There is also a set of 307 instances for the 2014 reports. They have been generated in separated sets because, as commented before, the methodologies used for their classification are slightly different.

**4.3 Classifier training, evaluation and selection**

**4.3.1 Studied classifiers**

For this work, four classifier algorithms have been selected for its evaluation. Next a brief description of each one is presented. Some parts of the classifier descriptions are extracted directly from the corresponding Wikipedia\(^4\) articles in order to make as clear as possible the explanation. All these parts are indicated by the corresponding citation. Anyway, more information about these classifiers can be found in the literature\(^5\)[6][7].

\(^4\)https://en.wikipedia.org
Naive Bayes
This classifier applies Bayes’ theorem\(^5\) to compute the conditional a-posterior probabilities of a categorical class variable given independent predictor variables [8].

In other words, Bayes’ theorem allows to calculate the probability of an event, in this case the reliability class of the report, based on conditions that may be related with that event[9]. After some derivations from Bayes theorem that we omit here, the probability of each class value \(c\) given an instance, or a set of attribute, value pairs is approximated by Naive Bayes as:

\[
Pr(class = c | attr_1 = value_1 \ldots attr_k = value_k) = \prod_{i=1}^{k} Pr(attr_i = value_i | class = c)
\]

All probabilities (Pr) are estimated from the data.

The reasons for selecting this classifier are (i) the fact it performs well in some real world situations, (ii) it’s highly scalable, (iii) requires a number of parameters linear in the number of variables, and (iv) its training time is also linear on the number of elements in the training set [9]. Also in the case of this study it is interesting the case of interpretation of the resulting model which is formed by the class distribution of the dependent variable and the conditional probability tables, unlike in other classifiers, for example the next one, which model is formed by the whole dataset.

Finally the implementation used in this project is the one contained in R package 'e1071'[8].

k-nearest neighbours
K-nearest neighbours is a classification algorithm that considers the class of the previously classified elements near a sample to determine its class.

The logic of this algorithm is pretty well described by this Wikipedia example [10] regarding figure 4.10:

The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If \(k = 3\) (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If \(k = 5\) (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

In other words, there is a need for a distance concept and a \(K\) parameter. Given a sample \(x\) to predict, select the \(k\) closest objects to \(x\) and return the major class. If there is no specific knowledge, normally the euclidean distance is used for \(x\).

One of the main reasons for selecting this classifiers is that only has one parameter (\(k\)). On the disadvantages, the fact that it can not perform good if the distance function is not well selected, the slowness of the prediction, and

---

\(^5\)https://en.wikipedia.org/wiki/Bayes’_theorem

\(^6\)Image By Antti Ajanki AnAj (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html), CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/) or CC BY-SA 2.5-2.0-1.0 (http://creativecommons.org/licenses/by-sa/2.5-2.0-1.0)], via Wikimedia Commons
CHAPTER 4. PROJECT DEVELOPMENT

Figure 4.10: Example of k-NN classification.

the difficulty to interpret the model because it is the whole dataset since all data is eligible to be considered when classifying a new sample. In any case, its simplicity justifies its selection.

The used implementation is the one provided by the RWeka Package\(^7\).

**Decision trees**

Classification trees are predictive models that use a decision tree to classify an element.

A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute, each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules \([13]\).

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split \([14]\).

In this work, two implementations for constructing decision trees have been used: RWeka\(^7\) implementing the C4.5\(^8\) algorithm and tree\(^9\) that implements CART\(^10\).

**Random forests**

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the

---

\(^7\)https://cran.r-project.org/web/packages/RWeka/index.html
\(^8\)https://en.wikipedia.org/wiki/C4.5_algorithm
\(^9\)https://cran.r-project.org/web/packages/tree/index.html
\(^10\)https://en.wikipedia.org/wiki/Decision_tree_learning
same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them[15].

In other words, this algorithm logic consists in a construction of multiple Decision Trees and output the mode of the classes predicted by them. Its main purpose is to avoid the over-fitting typical of Decision Trees.

The implementation used in this project is the one provided by the R library randomForest which implements Breiman’s random forest algorithm (based on Breiman and Cutler’s original Fortran code) for classification and regression[16].

4.3.2 Classifier training and evaluation

Training and testing sets

The construction of a classifier is called training, it is done using a set of already classified instances that are used to create the classification rules as described, for each classifier in Section 4.3.1. After a classifier has been trained, it has to be tested to validate its performance. The validation details are described next but, to validate a classifier, a set of classified instances is needed. One naive approach is to train and test a classifier with all the validated instances available, since, as more instances are used to train a classifier better it generally performs. This is an error because the classifier can present over-fitting. This phenomenon is produced when a classifier specializes too much in predicting elements of the training set, performing very well classifying those elements but very bad classifying unseen ones. Is for this reason that usually the original instance set is divided in two subsets, one for training and one for testing. In this project all classifiers have been trained using a randomly selected set of the 70% of all the classified instances available, leaving the remaining 30% for testing purposes. These percentages are used traditionally in a large majority of the machine learning literature, although there is really no formal justification.

As commented before in Section 4.2.2, two sets of 1787 and 307 classified instances have been generated for the years 2015 and 2014 respectively. In Section 4.1.3, the “class” variable, representing the reliability for a given report to be a tiger mosquito, has been noted to be one of the most important in this study, this is because is the one used to create instances: If there is no value for this variable, a training instance can not be generated since the classification for that report is unknown. One particularity of the two generated instance sets (2014 and 2015) is the proportion of the different reliability classes. In Tables 4.1 and 4.2 the number of instances for each class can be observed in the instances generated from the reports of 2014 and 2015 respectively. Note the class “0” has been removed, this is because although the reports has been revised and classified its meaning is the same as an unclassified report and, along with the major proportion in both sets, would have lead to an over-fitting of the models to a class that is useless.

Another important particularity of the proportions observed is the fact that negative classified instances are very much lower in number than positive ones. This may produce biases in the model under construction leading to positive

---

CHAPTER 4. PROJECT DEVELOPMENT

<table>
<thead>
<tr>
<th>Class</th>
<th>-2</th>
<th>-1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>14</td>
<td>12</td>
<td>105</td>
<td>176</td>
</tr>
</tbody>
</table>

Table 4.1: Reliability class proportion of 2014 instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>-2</th>
<th>-1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>91</td>
<td>29</td>
<td>856</td>
<td>811</td>
</tr>
</tbody>
</table>

Table 4.2: Reliability class proportion of 2015 instances.

biased classifiers. This has been considered when designing additional training sets.

In addition to training the classifiers with the described sets, some variations on these sets have been worked out seeking the correction of biases or limitations in the resulting classifiers. The final part of this section describes these variations, and the most relevant results are described in the next section.

To start, since 2014 and 2015 report have been classified in an slightly different way, the classifiers have been trained using the two sets and combining both.

Some experiments changing set variables have been done in order to identify the most relevant ones in the classification. Another experiment using only instances related to first reports of the users have been done trying to know if there is a difference between reports sent for the first time and the others. In both cases there are no significant results.

Finally, in order to correct the unbalanced reliability class proportion the negative classified instances latter have been replicated and, in addition, the classification problem has been simplified considering only two categorical variables: positive including reliability classes 1 and 2, and negative including -1 and -2. This methodology is also used in the main project, *Mosquito Alert*, when computing some models.

Validation

In order compare classifiers and to state a classifier is better than another the need of a way to compare classifiers performance arises. After classifying the testing set using a classifier a confusion matrix is obtained which provides detailed information on the classifier performance.

<table>
<thead>
<tr>
<th>Total population</th>
<th>True class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td>valid</td>
<td>not valid</td>
</tr>
<tr>
<td></td>
<td>True positive (Tp)</td>
<td>False positive (Fn)</td>
</tr>
<tr>
<td></td>
<td>False negative (Fp)</td>
<td>True negative (Tn)</td>
</tr>
</tbody>
</table>

Table 4.3: Confusion matrix

This matrix as can be seen in Table 4.3 represents the intersection between the classifier results and the classification set classes. A classified element can be one of four types:

**True Positive (Tp)** A positive element classified as positive.
4.3. CLASSIFIER TRAINING, EVALUATION AND SELECTION

True Negative (Tn) A negative element classified as negative.

False Positive (Fp) A negative element classified as positive.

False Negative (Fn) A positive element classified as negative.

From the confusion matrix one can derive various useful metrics. The first one is Accuracy which represents the proportion of correct classified elements. It can be calculated using the Equation 4.2.

\[
\text{Accuracy} = \frac{Tp + Tn}{Tp + Fp + Tn + Fn}
\]  

(4.2)

Next, Precision represents, for example considering the valid class, the proportion of valid predicted elements that in fact are valid. The equation, for valid class, is 4.3.

\[
\text{Precision} = \frac{Tp}{Tp + Fp}
\]  

(4.3)

Finally, the Recall represents the proportion of valid elements selected. Its Equation for the for valid class is 4.4.

\[
\text{Recall} = \frac{Tp}{Tp + Fn}
\]  

(4.4)

At this point these metrics can be used to compare classifiers, but using them separately may lead to misinterpretations, for example, since as commented before one class is significantly smaller than another, a classifier returning always the larger class will have a very good accuracy, another classifier classifying correctly only a 10% of elements will have a very good precision and so on...

For this reason it is necessary to have a metric which takes in account various of the aspects commented. This metric is called F-measure and uses Precision and Recall in a proportion that can be adjusted. In this project the same proportion is used, this is the harmonic mean of recall and precision and is an special case called F1measure, the Equation being:

\[
F - \text{measure}(F1) = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]  

(4.5)

In addition, although this example has been presented for only two classes (positive and negative), the metrics need to be calculated for each of the classes the classifier can provide.

A final comment is that since randomness takes part in the training process, for example when selecting the training instances, the same classifier results may differ from one to another depending on which instances have been selected for its training. Because of this, each classifier has been trained 10 times selecting different instances, and comparisons are done based on averages.

4.3.3 Experiments and results

Next, the most relevant experiments of this study are described. The results of each experiment and their interpretation have guided the design of the next experiment leading to the finally selected classifier.
CHAPTER 4. PROJECT DEVELOPMENT

Considerations

As commented in Section 4.3.2 some experiments have been made using the two training sets (2014 and 2015) separately, the results have been similar in the basic experiments with both so the ones presented here are performed only with the 2015 training set which has more instances. The reason, as said before, is because of the differences in the classification process. In the same way, the exercise of training the classifiers with two subsets of instances, one containing only the instances representing the first report for each user and the other containing the remaining, has produced equally similar results so the whole training set is used.

Classifier training and evaluation using all instances

In these cases, the training set corresponds with the generated instances for year 2015 the proportions are the same as in Table 4.2.

Naive Bayes

The confusion matrix generated after training the selected implementation of Naive Bayes with all the 2015 dataset is the one represented in Table 4.4.

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>True class</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>-2  0  0  3  2</td>
</tr>
<tr>
<td>-1</td>
<td>89 29 756 564</td>
</tr>
<tr>
<td>1</td>
<td>1 0 26 30</td>
</tr>
<tr>
<td>2</td>
<td>1 0 71 215</td>
</tr>
</tbody>
</table>

Table 4.4: Confusion matrix, Naive Bayes, 2015 dataset (all instances)

To have an overall impression of the classifier we used the whole dataset with the 4 reliability classes. As can be seen in the confusion matrix the classifier does not perform very well (check the values off the diagonal). In fact, it has a bias to the “-1” class: all negative classified instances are classified as positive one.

k-nearest neighbours

For this classifier we used the RWeka implementation. One of the particularities of this library is that it implements a k-fold cross validation method. This method consists on dividing the provided dataset in k subsets and for each one, use the other k − 1 to train the classifier and use the current subset to test it. This way all instances will be used at least once to test the classifier. Finally the results provided are the average of the k folds. This k variable is parametrized and for this experiment is fixed at 10. Thus having this validation method, the classifier has been trained also with all the dataset.

Another parameter that can be selected for this classifier training is another k that states for the number of neighbors used in the classification. After some experiments this has been fixed to 7 due to computational time restrictions.

Considering all that, after training the classifier under these conditions the resulting confusion matrix is the one in Table 4.5.
4.3. CLASSIFIER TRAINING, EVALUATION AND SELECTION

This classifier does not also perform well, the major part of negative classified instances are predicted as positive and the positive ones are not accurately predicted.

Decision trees
As commented in Section 4.3.1 two implementations have been used for this kind of classifier. First, we present the one implemented in the RWeka library. As in the k-nearest neighbours case, the k-fold cross validation method is also used to present the final results.

This implementation also allows some control options and one of them ($M$) is used in this study to tune the classifier training procedure. This option defines the minimum number of instances for each leaf of the classification tree. So if $M = 1$ the tree will grow until its maximum depth because every node will be split until each leaf has only one instance classified by the classification rule represented by the path from this leaf to the tree root. Assigning 1 to this parameter, as explained, is not a good decision because the classifier will grow too much, having a lot of splits and on the other hand over-fitting can be produced. Is for this reason that experiments have been made for $M$ values of $1, 3, 5, 10, 14$ and $20$.

In Table 4.6 we present the confusion matrix of the experiment with $M = 14$. This value for $M$ produces a total of 20 splits which is a reasonable amount in the case of an hypothetical code based implementation.

The results are similar to k-nearest neighbours ones in table 4.5 and the conclusions are the same.

Another implementation of this classifier is the one provided by the library tree. This one does not provide the validation method of RWeka and as in Naive Bayes the whole 2015 dataset has been used as the initial performance exploration. It also has been used without parameters and produces the confusion matrix in Table 4.7.
CHAPTER 4. PROJECT DEVELOPMENT

Table 4.7: Confusion matrix, tree library, 2015 dataset (all instances)

<table>
<thead>
<tr>
<th>True class</th>
<th>-2</th>
<th>-1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>28</td>
<td>8</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>62</td>
<td>21</td>
<td>767</td>
<td>521</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>74</td>
<td>288</td>
</tr>
</tbody>
</table>

This one is also not quite good. The classifier is biased to the “1” class (which has the smaller error) but its not useful to predict any of the other reliability classes.

Random Forest

Table 4.8 shows the confusion matrix obtained, after training the selected random forest classifier implementation under the same conditions. The validation context is the same as for Naive Bayes and tree and 500 trees have been created. This number of trees has been selected after some experiments looking to find balance between performance and execution time, since the number of training instances will be increasing during time.

<table>
<thead>
<tr>
<th>True class</th>
<th>-2</th>
<th>-1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>21</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>57</td>
<td>16</td>
<td>543</td>
<td>333</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>6</td>
<td>306</td>
<td>475</td>
</tr>
</tbody>
</table>

Table 4.8: Confusion matrix, randomForest, 2015 dataset (all instances)

The results of this last experiment with all 2015 instances is similar to the ones obtained with the other classifiers. They all have error rates near 40% and are unable to properly classify any class. The following experiments will make use of various techniques trying to increase the performance of the classifiers.

Classifier training and evaluation using “completely classified” instances

Given the inability of the studied classifiers to accurately predict an instance class from a set of four possible classes, the next step is to simplify the problem by defining only two possible classes. As commented in Section 4.3.2 this technique has been used in some processes in the main project so its replicated in this one. To achieve that, the instances have been grouped by the sign of their classification producing two classes:

**positive** All instances with class “1” or “2” have been grouped and assigned to this class.

**negative** All instances with class “-1” or “-2” have been grouped and assigned to this class.
The proportion of the classes after grouping the classes is the one in table 4.9. As can be seen, the number of negative instances is still much lower than positive ones, in fact is simply the sum of the previous negative classified instances so this aspect has not changed.

<table>
<thead>
<tr>
<th>Class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>120</td>
<td>1667</td>
</tr>
</tbody>
</table>

Table 4.9: Class proportion of 2015 grouped instances.

Using this modified dataset the experiments are repeated for each of the classifiers.

**Naive Bayes**

The results for this classifier trained with all instances are represented in Table 4.10.

<table>
<thead>
<tr>
<th>True class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative</td>
<td>110</td>
<td>1100</td>
</tr>
<tr>
<td>positive</td>
<td>10</td>
<td>567</td>
</tr>
</tbody>
</table>

Table 4.10: Confusion matrix, Naive Bayes, 2015 dataset (grouped instances)

In this case the results are a little bit better, at least the classifiers recall for negatives is quite good although its precision is bad. In the following, to check for the robustness of the result, the classifier is trained with 70% of the instances leaving the other 30% for testing. The resulting confusion matrix is shown in the Table 4.11.

<table>
<thead>
<tr>
<th>True class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative</td>
<td>28</td>
<td>329</td>
</tr>
<tr>
<td>positive</td>
<td>5</td>
<td>175</td>
</tr>
</tbody>
</table>

Table 4.11: Confusion matrix, Naive Bayes, 2015 dataset (grouped instances)

As expected, the results are slightly worse, considering the classification proportions, but it maintains the same behavior, this is good indeed because at least proves the robustness of the model.

**Decision trees**

Various values for the $M$ parameter (*Rweka* implementation) are presented because the overall results are slightly better in this scenario (completely classified (grouped) instances). The goal is to find the trade off between accuracy and number of splits. Although it has been trained with $M$ values of 1, 3, 5, 10, 14 and 20 the results for $M = 1$ and $M = 3$ are presented in Tables 4.12 and 4.13 respectively.

The resulting model for training with $M = 1$ contains 17 splits and the resulting one for training with $M = 3$ contains 13 and maintains approximately
the same performance. The splits for this last model can be observed in Figure 4.11. For each split, the selected variable is showed and the decision values are over the edges. In the leaves (at the bottom of the decision tree) the number of instances classified by the rule represented in the path are indicated by \( n \), and the chart indicates the proportion of class instances in this leaf.

Despite, in a first sight, all experiments obtained quite good accuracies, the recall for all of them in the negative class is poor. In addition the good accuracy results are caused because of the major proportion of correctly predicted positive instances. This phenomenon can be observed also in Figure 4.11 in the leaf marked as ‘’Node 27’’ where 1648 instances have been classified and the major part are positive ones. This eclipses the bad performance in negative class.

In the other implementation, \texttt{tree}, the results (see Table 4.14), are similar or even worse in terms of precision and recall for negative classes.
4.3. CLASSIFIER TRAINING, EVALUATION AND SELECTION

Table 4.14: Confusion matrix, tree library, 2015 dataset (grouped instances)

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>True class</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>34</td>
</tr>
<tr>
<td>positive</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>1658</td>
</tr>
</tbody>
</table>

Table 4.15: Confusion matrix, randomForest, 2015 dataset (grouped instances)

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>True class</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>27</td>
</tr>
<tr>
<td>positive</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>1660</td>
</tr>
</tbody>
</table>

Random Forest

Finally, the Random Forest implementation despite a good error rate, as the previous experiments in this conditions (completely classified instances), presents an large errors when classifying negative instances. These results are presented in table 4.15.

Classifier training and evaluation using “completely classified” instances and extending negatives

Up to now, a common phenomenon has been observed in all classifiers, this is produced due the major proportion of positively classified instances over the negative ones and leads to a good overall performance but bad results when classifying negative instances. To address this problem, there is the option to replicate the negative instances in order to make their proportion equal to the positive ones. This may lead to a performance increase when classifying negative instances. This replication is only performed in the training set, leaving the testing one with the original proportions. So, starting from the same group of instances used in the previous section which class proportion can be observed in Table 4.9, two subsets have been selected. The one used for testing purposes has the class proportion presented in table 4.16. In the one used for training the negatively classified instances have been replicated 10 times in order to approximately achieve an equally proportioned set. This proportion is shown in Table 4.17.

Table 4.16: Class proportion of 2015 grouped instances testing set.

<table>
<thead>
<tr>
<th>Class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>34</td>
<td>503</td>
</tr>
</tbody>
</table>

Table 4.17: Class proportion of 2015 grouped instances training set with replicated negative instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>860</td>
<td>1164</td>
</tr>
</tbody>
</table>
Using this two sets the following classifiers have been trained and tested, this time the evaluation metrics used are the ones presented in Section 4.3.2.

**Naive Bayes**
Table 4.18 shows the confusion matrix for the Naive Bayes classifier under these training conditions. The evaluation metrics calculated for each class in the confusion matrix is shown in Table 4.19.

<table>
<thead>
<tr>
<th>True class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>31</td>
</tr>
<tr>
<td>positive</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.18: Confusion matrix, Naive Bayes, 2015 dataset (grouped instances, replicated negatives x10)

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.380</td>
</tr>
<tr>
<td>Precision</td>
<td>0.983</td>
</tr>
<tr>
<td>Recall</td>
<td>0.344</td>
</tr>
<tr>
<td>F-measure (F1)</td>
<td>0.51</td>
</tr>
</tbody>
</table>

Table 4.19: Evaluation metrics, Naive Bayes, 2015 dataset (grouped instances, replicated negatives x10)

Although its accuracy is not good (0.38), it has a very good precision when classifying positive instances and also a high recall for the negative ones (both larger than 0.9). These two properties are very important and make this classifier useful in the context of the Mosquito Alert platform. These two particularities on this topic will be discussed next in Chapter 5.

**Random Forests**
Table 4.20 is the resulting confusion matrix for the Random Forests classifier trained under the same conditions. The metrics used for this classifier evaluation are the ones on table 4.21.

<table>
<thead>
<tr>
<th>True class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>20</td>
</tr>
<tr>
<td>positive</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4.20: Confusion matrix, Random Forest, 2015 dataset (grouped instances, replicated negatives x10)

Observing the tables, this classifier is the most balanced of the three presented in this section. It has quite good precision and recall for positive instances even though for negative ones is ambiguous in most aspects. Anyway its performance for positives makes it a candidate to be considered in the classification system.
### 4.3. CLASSIFIER TRAINING, EVALUATION AND SELECTION

#### Table 4.21: Evaluation metrics, Random Forest, 2015 dataset (grouped instances, replicated negatives x10)

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.877</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.97</td>
<td>0.277</td>
</tr>
<tr>
<td>Recall</td>
<td>0.897</td>
<td>0.588</td>
</tr>
<tr>
<td>F-measure (F1)</td>
<td>0.932</td>
<td>0.377</td>
</tr>
</tbody>
</table>

Table 4.22: Confusion matrix, Rweka decision tree (M=10, numfolds=10), 2015 dataset (grouped instances, replicated negatives x10)

<table>
<thead>
<tr>
<th>True class</th>
<th>Predicted class</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>negative</td>
<td>21</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>positive</td>
<td>13</td>
<td>415</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.23: Evaluation metrics, Rweka decision tree (M=10, numfolds=10), 2015 dataset (grouped instances, replicated negatives x10)

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.812</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.97</td>
<td>0.193</td>
</tr>
<tr>
<td>Recall</td>
<td>0.825</td>
<td>0.618</td>
</tr>
<tr>
<td>F-measure (F1)</td>
<td>0.892</td>
<td>0.294</td>
</tr>
</tbody>
</table>

Despite of keeping this classifier as a possible candidate for correct report classification, and the fact that some of its metrics are quite good, the overall performance and results are not relevant enough to allow its integration in the classification system in a successful way.

#### 4.3.4 Selected classifiers

Our results suggest that there are two classifiers that may suit the project needs. The first one is *Naive Bayes* as it can be used to filter a large number of positives with high certainty (large precision for positives, although with low recall). In other words, it selects many reports as positive that are positive with high probability, and that therefore do not need to be inspected by experts. On the other hand *Random Forests* is the most balanced classifier and may be considered to be included in the system if for some further application a classifier with high accuracy is desired.
4.4 Real-time classification system design

Since this project is part of the bigger initiative called Mosquito Alert the design of the proposed real-time classification system has to be considered part of an already existing architecture.

Although it is possible to integrate almost any system into another designing and implementing the proper interfaces, the smartest way to go is to design the system to be integrated considering the particularities and restrictions already present in the existing system. Most of the decisions taken during the design of this system were supported by this idea regarding aspects as the programming languages used or even the selected database engine.

Considering this, in order to integrate the selected classifier in the current Mosquito Alert system, the architecture depicted in Figure 4.12 is proposed. During this section the most important considerations and design details are commented.

The basic workflow starts when a new report arrives, then an instance is generated as explained in Section 4.2.2. After that, the instance is evaluated by the classifier system and the report is stored along with its classification in the reports database. For the reasons explained in the previous Section (4.3.4) the report may have not been classified as valid, in this case, the entomologists participating in the project can obtain this unclassified report and classify it manually. After that, this manually classified report would become part of the training set of the classifier that has to be trained periodically in order to increase its performance due to the new information provided by the new manually classified reports.

As observed, this architecture contains two subsystems. One of them is the instance generation system presented in Section 4.2.2 and the other is the subsystem which integrates the classifier.

4.4.1 Instance generation system

The representation of this system in Figure 4.12 encapsulates a system very similar to the presented in Figure 4.9 in Section 4.2.2. The only difference between both systems is the data persistence support which in the first one is presented as in-memory space of the Python scripts running the environment and in this one as a database. This is an important aspect since the code conforming this subsystem will have to be modified to be fully integrated in the main system. By now, all the solutions have been designed and developed in an isolated way to allow its full functionality in standalone mode. As can be observed in the mentioned Figure 4.9 the script containing the main logic of the instance creation process is `create_instance.py`. The other two scripts `users.py` and `environment.py` act as interface to the data and add various operations regarding the data they handle. Having this in mind the main modifications that have to be done in the scripts are the following:

- `create_instance.py` The input of this script has to be modified to obtain the new reports from a database and not from a CSV file.
- `users.py` The data structures defined in this file have to be redefined to be directly stored in a database, or at least loaded and stored for a faster operation. The provided functions have to remain the same.
4.4. REAL-TIME CLASSIFICATION SYSTEM DESIGN

Figure 4.12: Designed real-time classification system.

In the same way as the modifications considered in the `users.py` script, the data structures have to be redefined and the provided functions must remain the same. In this case the logic in the functions may be improved if the database is able to operate with geospatial data as proposed in Section 7.1.2 of Chapter 7.
4.4.2 Classification system

The other important system presented in Figure 4.12 is the classification system. This subsystem has to be able to classify instances and to retrain the classifier. The design will depend on the decisions taken by the Mosquito Alert team members but here we introduce the main components, which are basically R scripts.

Training script. This script implements the experiment that produces the selected classifier and stores the classifier object in a file.

Classifier. The classifier is the result of the training script and is stored in an R workspace file.

Classification script. This script must load the classifier object and one instance, evaluate it, and return the classification result.

Since R can be invoked from command line a Python script is proposed to be triggered when a new instance is created to retrieve it and to call the classification script, after the classification is performed it must store the result in the report database.

Another important point of this subsystem is the classifier retrain. To achieve this functionality a trigger must be designed to be triggered when a given amount of new manually classified instances are created. This can be directly a trigger in the database executing a procedure that somehow causes the execution of the training script. There are various ways to achieve that execution, for example, a procedure directly written in Python that calls the R script, or a mechanism to add a cronjob to run the same script.
Chapter 5

Discussion

In this Chapter, the main results of the Master Thesis are commented along with related ideas giving a broader contextualization, and willing to introduce results in a slightly different way.

5.1 Selected classifier

For the reasons presented in Section 4.3.3 the selected classifier that best suits current Mosquito Alert needs is the Naive Bayes classifier trained using all 2015 instances, generated following the procedure described in Section 4.2.2 grouped by the sign of their class and with the negative instances replicated 10 times. The reason is that it can detect approximately 1 third of the valid reports with a precision near 98%. This is also endorsed by a third particularity that is its high negative recall which strengthens the fact there will be very few false positives.

5.1.1 Improving performance

In order to improve the positive recall there is a possibility to tune the classifier prediction with Laplace smoothing\(^1\). By doing this an increase of the prediction performance can be achieved as shown in tables 5.1 and 5.2 about the classification results and evaluation metrics for the tenth iteration of the most successful experiment tuning this parameter.

<table>
<thead>
<tr>
<th>Predicted class</th>
<th>True class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>negative</td>
</tr>
<tr>
<td>negative</td>
<td>26</td>
</tr>
<tr>
<td>positive</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 5.1: Confusion matrix, Naive Bayes, 2015 data-set (grouped instances, replicated negatives x10, laplace smoothing), best case.

These results are for the best performance observed without affecting the negative recall, so despite is possible to achieve more positive recall its important to preserve the properties that make this classifiers eligible for the classification system. This results are purely experimental and it will be necessary

\(^1\)https://en.wikipedia.org/wiki/Additive_smoothing
CHAPTER 5. DISCUSSION

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.495</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.976</td>
<td>0.112</td>
</tr>
<tr>
<td>Recall</td>
<td>0.468</td>
<td>0.843</td>
</tr>
<tr>
<td>F-measure (F1)</td>
<td>0.624</td>
<td>0.195</td>
</tr>
</tbody>
</table>

Table 5.2: Evaluation metrics, Naive Bayes, 2015 data-set (grouped instances, replicated negatives x10, laplace smoothing), best case.

Further research to assure this parameter can be fixed at a given value or by a deterministic system in order to maximize the classifier positive recall in each one of its future training.

5.1.2 Important variables

The detailed model of the selected classifier can be found in Appendix A. Is the direct R output of the classifier object. The data displayed for A-priori is the class distribution for the dependent variable and the Conditional probabilities are a list of tables, one for each predictor variable. For each categorical variable a table giving, for each attribute level, the conditional probabilities given the target class. For each numeric variable, a table giving, for each target class, mean and standard deviation of the (sub-)variable[8].

As all the variables in one instance are numerical so a good way to see what of them are the more important in the prediction is to plot the distributions. Figure 5.1 shows, as an example, the class distribution for three of the variables in the selected classifier which are the ones representing the answers to the questions. As can be observed, there is a notable differentiation between classes. As higher this differentiation, more decisive the variable is in the classification process.

5.2 System design

Since there is an already a broader existing system in Mosquito alert, and only some information of the main components of this bigger system have been provided, the proposed design is quite unspecific as commented in Section 4.4. Anyhow after this project finalization the code will be published to GitHub\(^2\) to allow the main project participants to access it and make the appropriate modifications in order to build the proposed system with the modifications they consider as part of its own system.

5.2.1 NoSQL in-memory database

In Section 4.4.1 the modifications to be done to the users.py script are presented but one of the things that may be interesting to consider, especially for this script, is the use of an in-memory NoSQL database to maintain the previously mentioned context. By doing this the system performance will be slightly better and there are solutions that allow this kind of design such as Redis\(^3\).

\(^2\)https://github.com/
\(^3\)http://redis.io/
which according to its site is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.

5.3 Report validation priority

Classification probabilities

Until now, the output of the classifier has been presented as the possible values of the categorical variable \texttt{class}, but there is another format for the output of the classifiers consisting in the probability for each of the classes. In fact, the predicted class is decided selecting the largest of this probabilities. However, the actual probability values can be more useful. More precisely these values can be used to order a set of unclassified reports from most probable to less probable to be valid and, after that, verify how much of them begging from the top are indeed valid. This exercise would allow modifying the prediction threshold for each class and reduce the number of reports to classify manually (see Section 7.2.1).

Priority system

Based on all the above, the idea of a classification priority system arises. Basically it may consist in inverse ordering the remaining unclassified reports by their probability of being positive and provide them in order to the entomologists to focus on the most difficult to classify. Other option would be to decide proportions of difficulty depending on the current needs. The classification can also be enriched with other classification priorities among the classification difficulty such as the location interest of the report. This parameter will have to represent the interest of a report according to its location being more interesting the reports sent form locations where the studied mosquito specimens have not yet been established or spotted.

This system otherwise can play a bad role in the classifier training because it will lead to the current training context where the class proportion is heavily unbalanced to positive. So, to attach this problem, the same system can be used occasionally to expressly classify negative reports to use them in the future training of the classifier.
(a) Class distribution for \texttt{reportQ1Answ} variable which models the answer to the question: \textit{Is small, black and has white stripes?}.

(b) Class distribution for \texttt{reportQ2Answ} variable which models the answer to the question: \textit{Has a white stripe in both head and thorax?}.

(c) Class distribution for \texttt{reportQ3Answ} variable which models the answer to the question: \textit{Has white stripes in both abdomen and legs}.

Figure 5.1: Class distribution for report answers variables.
Chapter 6

Conclusions

In this chapter the accomplishment of the proposed goals is reviewed and, after that, a more personal evaluation of the project is presented.

6.1 Goal review

The proposed goals for the project have been achieved as presented next. For each category, the goals are presented again along with a brief explanation about its achievement.

6.1.1 General

1. To develop a data processing system to extract knowledge of the data generated by the users of the described application and the reports they send. ✓

2. To analyze various machine learning algorithms (classifiers) and select the most useful to identify valid reports. ✓

3. To design a system architecture that integrates them into the main project system. ✓

The first one of the goals in this section has been achieved developing the instance generation system which, indeed, extracts knowledge from the users sent reports in order to create the instances. The second one has been achieved by all the work done as described in Section 4.3 of Chapter 4. Finally, the third one achievement is proved by the Section 4.4 of the same chapter.

6.1.2 Specific

1. Perform a proper data analysis. ✓

2. Train and evaluate classifiers. ✓


Since these objectives are derivatives of the General Objectives, its achievement is proved by the same reasons as in the previous section.
6.1.3 Academic

1. Acquire knowledge about R programming language. ✓

2. Learn new data-analysis techniques and consolidate the ones already known. ✓

3. Put into practice the knowledge acquired during the master degree. ✓

This ones are more subjective to evaluate, but before I started the project my only contact with R was during my internship at inLab when participating in the Sensetrack project\(^1\). Some R scripts where developed in this project but I only integrated them in the system, so I had never coded anything in R or used its console beyond the essential. At the beginning of the project I took two online courses via Coursera\(^2\). The courses where The Data Scientist’s Toolbox and R Programming both from Johns Hopkins University. With the knowledge acquired during this courses I could start analyzing data and scripting all the functions I used during the project. Of course I learned much more during the development solving various questions and aspects appearing during the process.

The second one has also been achieved, for example I obtained a deeper knowledge of some of the classifiers and the replication technique was new for me. Finally, although not related directly, I can assure during the master degree my abilities have increased in various aspects and I’m sure I would not have done the project in the same way when I started the degree.

6.2 Personal evaluation

I think this Master Thesis has been a very interesting experience form various points of view. Academically, I approached a new programming language and I have explored a little bit more in dept a field of informatics I only approached in one subject. On the other hand I have not only learned about informatics but also some basic concepts of biology. More personally, the fact that the developed solution will be integrated and used in a real world application makes me proud of having had the opportunity to collaborate in an initiative I think is very useful in many levels form scientific to social.

Of course, it has not been easy to develop this project. Specially in the classifier part I have learned that machine learning not only consists in “throwing” data to feed some algorithm and wait for the results but a considerable data cleaning, preparation and interpretation is necessary to obtain non trivial results.

Finally, from the Mosquito Alert team perspective, the Master Thesis provides a mechanism to reduce the volume of work to do by entomologists and, as has been said, leaves them more time to focus in activities adding much value to the whole project. I also expect some of the recommendations made in the next chapter 7 will be considered when taking decisions in terms of scalability.

---
\(^1\)http://inlab.fib.upc.edu/ca/sensetrack
\(^2\)https://www.coursera.org
Chapter 7

Future work

At the beginning of this project, the main project Mosquito Alert was called atrapaeltigre.com and was focused to identify only the tiger mosquito (Aedes albopictus) in the Iberian Peninsula. During the last months the project has been expanding and currently its scope covers two types of mosquito adding the yellow fever mosquito (Ae.aegypti) and new territories in South America and east Asia. For this reason one of the main future challenges is the scalability of the system in order to support the growing flow of reports and an increase on its complexity.

In this chapter some ideas and indications are proposed to follow up with the work done on this new context. First, the questions that are directly affecting the system scalability, and second, other improvements or ideas that can help the whole Mosquito Alert platform be more efficient.

7.1 Scalability

As long as the Mosquito Alert project is expanding, the system needs to escalate accordingly to meet the future requirements caused by a more complex database able to store reports of various mosquito species and an increasing report flow coming from various geographical locations around the world generated by an also increasing user base. In this scenario various proposals are presented in order to make the system able to provide the same functionalities like until now with the same quality of service.

This section’s recommendations have been divided in three areas regarding modifications to be performed to the code generated in this Thesis, modifications in the database management system and changes in the computational resources.

7.1.1 Code modifications

The instance generation system uses all the environment reports to calculate instance variables such as the valid reports in 1km surrounding the current report during last month. As long as the reports are geolocated and there is a grid division of the geographical space used in the project, this operations can be highly optimized by using only the reports in the adjacent cells of the
current one. To achieve this improvement there is the need to modify the data structures storing reports in the environment module to store them according to the cell they belong which is already computed in the current version of the code. Another way to achieve that in a much more scalable way is to incorporate geographical modules to the current project database as described in the next section.

### 7.1.2 GIS enabled database

When working with geospatial data, it is a good effort to make the database administration system to be able to store and even operate with this kind of information in a more efficient way. The Mosquito Alert platform uses PostgreSQL as database administration system and there is an extension called PostGIS which adds support for geographic objects allowing location queries to be run in SQL. I personally have used this extension in other projects and I’m sure it could be very useful to perform some operations. For example the previously described operation of finding the reports in an area surrounding a point can be done directly in the database in a very efficient way. Installing this extension can provide an important improvement to the system performance and adapting the code to allow the use of its full functionality could be an additional improvement.

### 7.1.3 Considerations on computational resources

After applying these recommendations, there will be no need to increase drastically the computational resources dedicated to the project. All the proposed system operations can be performed in real time for a considerable flow of reports.

### 7.2 Additional improvements

There are also some aspects of the designed system that can be improved. They are presented in these final sections.

#### 7.2.1 Classifier tuning

As explained in Section 5.1.1 it is possible to improve the classifier performance by tuning the prediction parameters, more precisely the Laplace smoothing related ones. The optimal values have to be discovered through empirical exercises that, for time-restriction reasons, have been performed in a very superficial level during this project. For this reason, further research should entail study how to best-tune these in order to obtain the optimal performance of the classifiers.

#### 7.2.2 Priority system

The possibility to obtain the classifier results as probabilities for each class as commented in Section 5.3 may lead to a more fine tuning for the classification

---

1 [http://www.postgresql.org/](http://www.postgresql.org/)
2 [http://www.postgis.net/](http://www.postgis.net/)
threshold and to a classification priority system to concentrate the entomologists classification effort on the more interesting reports. The aspects for considering a report interesting are discussed also in the same section but the system has not been implemented yet and can be a next step to develop in order to optimize the entomologists task.

7.2.3 Another classifier

In Section 4.3.4 Random Forests is also selected apart of Naive Bayes. In this study the possible applications of this classifier in the Mosquito Alert system have not been discussed but it will be a good exercise to try to identify how this classifier can suit an existing system need that hasn’t already been identified nor solved by the Naive Bayes approximation suggested in this work.
Bibliography


[3] Estudi de les tendències de participació i els perfils d’usuaris dins del projecte AtrapaelTigre.com sobre el monitoratge del mosquit tigre (Aedes albopictus), Biology degree final project, Eva Lara Martínez, Anna Menció Domingo, Frederic Bartumeus Ferré, Centre d’Estudis Avançats de Blanes (CEAB - CSIC), Available at: http://dugi-doc.udg.edu/bitstream/handle/10256/11261/TFG.pdf?sequence=1


[16] Package ‘randomForest’ - Breiman and Cutler’s Random Forests for Classification and Regression, Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener, Available at: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
Appendix A

Naive Bayes Classifier for Discrete Predictors

The following is the selected *Naive Bayes* classifier as given directly by R:

```r
naiveBayes.default(x = training_set[, 1:nvars - 1], y = training_set[, nvars])
```

A-priori probabilities:
```
training_set[, nvars]
egatiu  positiu
0.4249012  0.5750988
```

Conditional probabilities:
```
<table>
<thead>
<tr>
<th></th>
<th>reportNote</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>negatiu</td>
<td>positiu</td>
</tr>
<tr>
<td>training_set[, nvars][,1]</td>
<td></td>
<td>0.3953488</td>
<td>0.4892100</td>
</tr>
<tr>
<td>os</td>
<td></td>
<td>0.4115120</td>
<td>0.4923191</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th></th>
<th>reportMonth</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>negatiu</td>
<td>positiu</td>
</tr>
<tr>
<td>training_set[, nvars][,1]</td>
<td></td>
<td>8.081395</td>
<td>1.679952</td>
</tr>
<tr>
<td>reportTimeOfDay</td>
<td></td>
<td>8.415808</td>
<td>1.048583</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>reportTimeOfDay</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>negatiu</td>
<td>positiu</td>
</tr>
<tr>
<td>training_set[, nvars][,1]</td>
<td></td>
<td>1.988372</td>
<td>0.8563151</td>
</tr>
<tr>
<td>positiu</td>
<td></td>
<td>2.017182</td>
<td>0.7576814</td>
</tr>
</tbody>
</table>
```
### APPENDIX A. NAIVE BAYES CLASSIFIER FOR DISCRETE PREDICTORS

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Predictors</th>
<th>Negative Class</th>
<th>Positive Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportQ1Answ</td>
<td><code>[,1]</code></td>
<td>0.5348837</td>
<td>0.9810997</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.7107633</td>
<td>0.1647954</td>
</tr>
<tr>
<td>reportQ2Answ</td>
<td><code>[,1]</code></td>
<td>0.1511628</td>
<td>0.7448454</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.6566121</td>
<td>0.4882269</td>
</tr>
<tr>
<td>reportQ3Answ</td>
<td><code>[,1]</code></td>
<td>0.4534884</td>
<td>0.9587629</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.6765140</td>
<td>0.2271749</td>
</tr>
<tr>
<td>newUser</td>
<td><code>[,1]</code></td>
<td>0.8837209</td>
<td>0.6134021</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.3207458</td>
<td>0.4871795</td>
</tr>
<tr>
<td>userNumReports</td>
<td><code>[,1]</code></td>
<td>0.3953488</td>
<td>2.2121993</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>1.9377930</td>
<td>6.948687</td>
</tr>
<tr>
<td>userAccuracy</td>
<td><code>[,1]</code></td>
<td>0.0201550</td>
<td>0.2014294</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.1150075</td>
<td>0.3690078</td>
</tr>
<tr>
<td>userTimeForFirstReport</td>
<td><code>[,1]</code></td>
<td>600247.3</td>
<td>1319376.7</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>3818986</td>
<td>5447502</td>
</tr>
<tr>
<td>userTimeSinceLastReport</td>
<td><code>[,1]</code></td>
<td>10524.58</td>
<td>123275.69</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>88240.57</td>
<td>580408.60</td>
</tr>
<tr>
<td>userMeanTimeBetweenReports</td>
<td><code>[,1]</code></td>
<td>21527.18</td>
<td>128208.51</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>131877.9</td>
<td>508685.7</td>
</tr>
<tr>
<td>userNumActionAreas</td>
<td><code>[,1]</code></td>
<td>0.1511628</td>
<td>0.6030928</td>
</tr>
<tr>
<td></td>
<td><code>[,2]</code></td>
<td>0.4453208</td>
<td>1.0942784</td>
</tr>
<tr>
<td>Variable</td>
<td>Training Set</td>
<td>Negatiu</td>
<td>Positiu</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>userMobilityIndex</td>
<td></td>
<td>0.00247093</td>
<td>0.0162135</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.04151621</td>
<td>0.2732878</td>
</tr>
<tr>
<td>reports1kmLastHour</td>
<td></td>
<td>0.03488372</td>
<td>0.1835920</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.14776632</td>
<td>0.6251976</td>
</tr>
<tr>
<td>reports1kmLastDay</td>
<td></td>
<td>0.04651163</td>
<td>0.2107129</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.25343643</td>
<td>0.7442738</td>
</tr>
<tr>
<td>reports1kmLastWeek</td>
<td></td>
<td>0.2325581</td>
<td>0.8722126</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.6426117</td>
<td>1.3313233</td>
</tr>
<tr>
<td>reports1kmLastMonth</td>
<td></td>
<td>0.5697674</td>
<td>1.668308</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>1.5687285</td>
<td>3.026041</td>
</tr>
<tr>
<td>validReports1kmLastHour</td>
<td></td>
<td>0.00000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.05781685</td>
<td>0.2300383</td>
</tr>
<tr>
<td>validReports1kmLastDay</td>
<td></td>
<td>0.01162791</td>
<td>0.1072664</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.09654823</td>
<td>0.2899247</td>
</tr>
<tr>
<td>validReports1kmLastWeek</td>
<td></td>
<td>0.04554264</td>
<td>0.1849922</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.17222938</td>
<td>0.3584786</td>
</tr>
<tr>
<td>validReports1kmLastMonth</td>
<td></td>
<td>0.04779439</td>
<td>0.1756374</td>
</tr>
<tr>
<td>training_set[, nvars]</td>
<td></td>
<td>0.23494232</td>
<td>0.3737211</td>
</tr>
</tbody>
</table>