FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Hugo Bronze Canelas de Brito Prata

# INTELLIGENT DATA CONSOLIDATION METHODS FOR CROWD-SOURCED CONTRIBUTIONS

## A contribution to FireLoc

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Prof. Alberto Cardoso and Prof. Jacinto Estima, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

January 2023

# Abstract

It is of the most importance to tackle a Natural Disaster as early as possible, so as to mitigate its effects to the fullest. While preparation and prevention are important steps in dealing with Natural Disasters, the speed at which an organisation can respond to fluid events can also greatly influence the outcome of such disasters. Thanks to recent developments in automation, Artificial Intelligence (AI), and the widespread access to mobile technology, it is our belief that it is possible to leverage this new technological reality to contribute to the fight against Natural Disasters.

The present work aims to develop a smart system capable of refining individual submissions of occurrences into clusters made up of more complex members, allowing for features such as their geo-location, evaluation, or the tracking of their evolution over time. To reach this goal different techniques related to Clustering and Data Fusion were researched. Subsequently, some were chosen by taking into account their performance and specifications in regards to the data that is expected in the context of this project.

This project is a modular contribution to the FireLoc project, a system designed for the identification, positioning and monitoring of forest fires with the aid of crowd-sourced data. This project also attempts to provide a new general approach to dealing with Natural Disasters, by providing a smart system that manages inflows of information-rich, geo-located events.

# Keywords

FireLoc, Disaster Detection and Monitoring, Geo-location, Data Fusion, Clustering, Smart Systems

# Contents

# Acronyms

**AI** Artificial Intelligence.

**BN** Bayesian Networks.

**CNN** Convolutional Neural Networks.

**DBSCAN** Density-based Spatial Clustering of Applications with Noise.

**FCT** Foundation for Science and Technology.

**GPS** Global Positioning System.

**GUI** Graphical User Interface.

**H-DBSCAN** Hierarchical Density-based Spatial Clustering of Applications with Noise.

**INES-C** Institute for Systems Engineering and Computers at Coimbra.

**ML** Machine Learning.

**MRF** Markov Random Fields.

**NN** Neural Networks.

**OPTICS** Ordering Points to Identify the Clustering Structure.

**PCA** Principal Component Analysis.

**UC** University of Coimbra.

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years we have been able to witness unprecedented developments both in the field of *Intelligent Systems* as well as *the Internet of Things*. Smart Algorithms and Artificial Intelligence (AI) have become part of our daily lives, and not only are they now a pillar of industry (Siemens, 2021), but also of society itself. From automating farming to dealing with train logistics, as well as suggesting new products to consumers through social media, *Smart Algorithms* are ever more present in our lives.

And with each passing year, technology as a whole evolves even further (Reock, 2020), becoming ever more intertwined both within itself as well as with its users. The average smartphone owner can now boast both an high-end 4k camera and a micro-computer that not only rivals the most advanced machines of yesteryear but that also has access to any place on Earth, thanks to technologies such as Global Positioning System (GPS) and the advancement of Social Media. All of this within the same package. Our smartphones are truly a powerful tool which, barring a lack of connection to the internet, can have unhinged access to virtually any other piece of technology, information or person, regardless of time or place.

But even in a relatively high-tech world, natural disasters still occur on a daily basis, and contemporary methodologies for preventing and combating these disasters are being overwhelmed. With the worsening of climate change in recent years (Mohanty, 2021), both the number of instances and severity of natural disasters have increased. Floods, fires, droughts. These events grow more and more common, and the developed world isn't an exception. Even considering the additional preventive measure taken in recent years, such as the increase in investment in organisations that counter these disasters, such as firefighters and foresters, the increasing trend of disasters has yet to show signs of subsiding.

It only makes sense to take advantage of the new technological reality, along with the availability of vast amounts of data, to aid and complement any system or organisation which holds the intent of mitigating natural disasters.

## 1.1 The FireLoc Project

The FireLoc Project is one such system that aims to help in the fight against natural disasters, by utilizing now-widespread mobile technologies. With universal access to smartphone cameras and the internet, Fireloc aims to draw on crowd-sourced data to locate, pin-point and monitor forest fires, with the goal of assisting authorities in the early identification and geo-location of ignitions, so that these may be tackled with as little delay as possible.


Figure 1.1: The Fireloc logo as of the making of this document.

FireLoc [1] is a joint effort of several professors of University of Coimbra (UC) and members of Institute for Systems Engineering and Computers at Coimbra (INES-C) (Alberto Cardoso et al., 2021), funded by the Foundation for Science and Technology (FCT), an organization within the Ministry of Science, Technology and Higher Education of Portugal. The FireLoc system uses data collected by citizens using a dedicated mobile app that enables the automatic triangulation of observed fires from the few known observation points. On top of the geo-located coordinates, there is also an option to submit geographic coordinates, text, and/or imagery through the app. By analysing the crowd-sourced data, the expectation is that the software would then be able to set forth a more detailed description of these occurrences to the respective authorities.

FireLoc is yet to be completed, but is currently built on three main components:

- The **data collection component** (i.e., the FireLoc app) which was developed with mobile devices in mind, while also being complemented by other modules that allow the collection of data through additional sources;
- A **data integration and processing component**, which handles duties such as geolocating observed events with the available data, assess upload on an user basis, and estimate the risk of events;
- The **data visualization component**, which includes a multi-platform Graphical User Interface (GUI) meant to be used by the authorities and other end-users, as well as an administrative interface.

In hindsight, the components that this research project focuses on are, for the most part, the second component, and to a smaller degree, the third component. This will be elaborated upon in the following subsections.

[1] More detailed information at `https://fireloc.org`

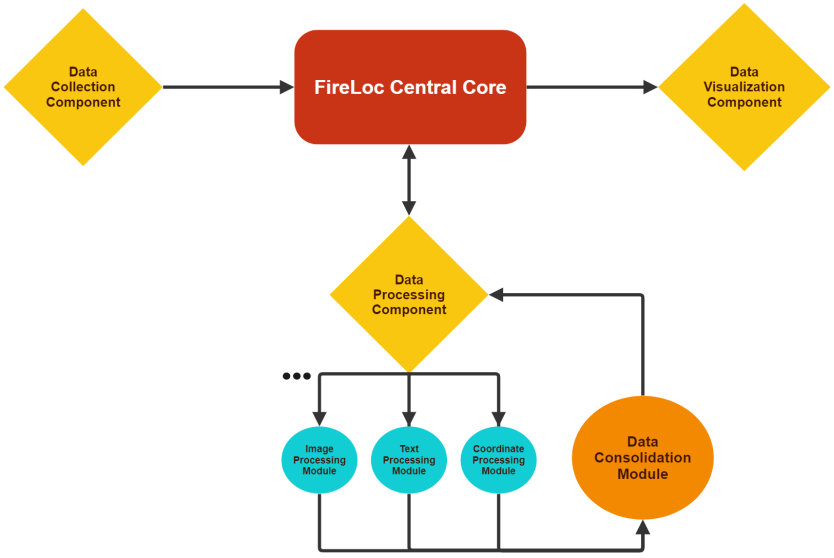


Figure 1.2: The Fireloc System, showing part of the data processing modules.

The new contribution module, named "data consolidation module" in the previous diagram, is meant to work both in a stand-alone mode, as well as in synchrony with other Fireloc modules within the data integration and processing component. As of the writing of this document, the workings of the Fireloc system and the new module that was created within this research project can be summarised as the following:

The **new contribution module** effectively receives submissions in real time which, at the time of input, would've already been analysed by the previous FireLoc components. These Fireloc components handle the coordinates and geo-location, as well as search and confirm visual information regarding both wildfires (i.e. fire and smoke) and landmarks (e.g. gas stations). Finally, they also parse through text within the submissions in search of relevant information. The modules in question have access to other sources of information besides crowd-sourced submissions. These include open-source satellite imagery, meteorological data, and data from the OpenStreetMap[2] project, all of which complement the datasets with valuable information.

The Fireloc data processing component utilizes several forms of AI throughout the modules that are part of it. The analysis of wildfires is done through **Deep Learning image recognition algorithms** to find signs of smoke and flames. On the other hand, the analysis of text is done through **Natural Language Processing**, which attempts to understand whether the submission holds any sort of relevant textual information on the situation, terrain, or landmarks close by. This process allows for the creation of simple and accurate "occurrences" which hold all the relevant information, while rejecting data that is deemed to be irrelevant by the algorithms. Some examples of irrelevant data would be submissions of pictures with the absence of any signs of fire or smoke, or malicious/spam submissions. Once the crowd-sourced data is treated, it is forwarded to our module.

[2]More detailed information at `https://www.openstreetmap.org`

## 1.2 Goals of this Project

The goal of this research project is to elaborate and prototype on a new stand-alone module that manages the processing, validation, and aggregation of data that results from the analysis of crowd-sourced submissions done by the FireLoc system. The inputs given to the new module by Fireloc will be used to produce concise and aggregated information about complex events, in a way that would otherwise be unachievable with just a single source of information. These complex events are to be displayed within a basic map GUI.

In summary, the Fireloc System will handle the early processing and validation of volunteer contributions, while the new module will follow up and process the contributions into a standard event data-structure made up of one or more of these contributions.

In its current state, Fireloc produces several occurrences which may or may not be related, and which may be complete or missing pieces of information. These Fireloc contributions need to be analysed so as to understand whether to associate them to existing events, create a new event, or to discard them. Should a contribution be associated to an existing event, or should it lead to the creation of a new event, both cases result in the attempt to identify the following pieces of information: geospatial location, the events' chronology (if there were any prior related events), and any element that is deemed to bring forth **new** relevant information about an event (should said event already exist).

By employing forms of AI and Machine Learning (ML) to achieve this projects' goals, the overall process would be completed in an efficient manner while also limiting both the cost of maintenance, as well as the need for human supervision and/or intervention. Finally, by allowing this information to be showcased to the authorities through a visual interface, the entirety of the decision-making process should become more streamlined. This would result in an optimal reduction in delays when it comes to mitigating natural disasters that can't always be predicted or prepared for. There would also be a reduction in the inherent delays and errors in answering fluid conditions away from the front-lines, such as from a logistical hub.

The main goals established in the context of this research project were the following:

- Research on possible methodologies to be used in the creation of a stand-alone module;
- Development of a fully functioning prototype module capable of handling and displaying real-time, heterogeneous data;
- Apply the developed module to realistic field-testing.

The first goal was elaborated throughout the research done for the State of the Art and background of the methodologies to be employed during development.

Several works of similar nature were studied so as to understand optimal ways to tackle the problems at hand.

The second goal was met during our development iterations, where we created the module, which was made up of three main components: A data aggregation component which utilized clustering techniques (H-DBSCAN), a data fusion component (a mix of Medium and High level fusion), and a visualization component which displayed the fused incidents on an interactive map (utilizing the Folium tool). For this goal, data was also generated, both randomly and based on real-world events, while meeting the numerous requirements set for the module[3]. These development iterations were divided between the testing and selection phase, or the "initial" phase, where methods and architecture were chosen, and the main development phase which created the module itself.

The final goal was met through several phases of testing. Both synthetic and real-world data was used to test the module. Synthetic data was randomly generated through various scripts, following rules created by using real-world data as a basis. The real-world dataset, on the other hand, was created using the October 2017 fires in Portugal as a reference.

## 1.3 Contributions of this Project

As previously stated, this project aimed to produce a new module, which can either be integrated into the FireLoc system, or used in a stand-alone mode. On top of this, the following documentation was also produced:

- State of the Art in regards to the context of this work and its methodologies;
- Architecture of the module and it's role within Fireloc;
- Prototyping results;
- Suggestions on improvements and future work.

By publicising the previous documentation, we also hope to contribute to the overall evolution of the State of the Art of the various fields present in this research project. This also means to play a part in and expand the efforts to mitigate the effects of Natural Disasters world-wide.

## 1.4 Document Structure

The remainder of this proposal is organized as follows: Chapter 2, Background and State of the Art, enumerates key concepts that are relevant to the work at hand, as well as the State of Art, which goes over contemporary works relevant

[3]See table 3.1 for an in-dept enumeration of the data used by the module.

to this project along with the technologies and methodologies employed by their authors.

In Chapter 3, Development Process, the chosen approach and methodologies are outlined. This chapter is divided into three phases of development. The initial phase, which puts into practice the research outlined in the state of the art and the several methodologies to be used. The second phase settles on the methodologies and showcases an early architecture of the module, and also functions as a proof-of-concept. The final phase completes the development of a fully working module that is ready for field-testing.

Chapter 4, Experiments and Results, enumerates the several test and prototyping runs, along with their results and any improvements or changes that result from them.

Finally, Chapter 5 - Conclusion, reviews the previous chapters while also presenting a summary of what was achieved throughout this research project. Some suggestions for future works and improvements are also proposed within this chapter.

# Chapter 2

# Background and State of the Art

This chapter is reserved to further elaborate on the background, concepts, and methodologies that are relevant to the context of the creation of a new **module** for the FireLoc System, that can also act as a generic, stand-alone approach to solving the issue of real-time autonomous aggregation, management and monitoring of data on events such as **wild fires** or other natural disasters. The use of crowd-sourced data to attempt to solve problems involving event detection has recently started to become a common approach, at the time of writing this research document. Examples of works that guided this research projects' approach include (Afyouni et al., 2022), which attempted to detect unusual events (both generic and specific) by utilizing social media. At an early stage, an attempt was done in dividing the researched concepts into coherent groups that represent a specific role or methodology within our new module. This was done so as to better understand the requirements imposed on the project, and to also function as well-defined milestones within our research for relevant methodologies and other similar works.

For each of these groups, a synopsis is presented within their respective subsections. These address the different methods available for each role, showcase examples of real-world applications along with a resume of their inner workings, as well as their individual relevance to the goals of this research project. The module was divided into three main groups of methodologies:

- Methods for Data Correlation [1];
- Methods for the Intelligent Component;
- Methods for Data Visualisation.

In the following sections, we will elaborate on the methods that appear to be the most widely used when attempting to face similar technical challenges to those of this research project.

[1] Images and Text are processed beforehand by a different FireLoc module, which includes a Natural Language Processor, among other features

## 2.1 Methods for Data Correlation

According to the contribution and goals set in chapter 1, the module needs to be able to merge and correlate different types of data from different sources, ranging from data which only includes simple coordinates to data which may include images and/or text. This process must be done in a way that extracts all the meaningful data that is to be used by the remaining components of the module. One possible methodology to meet these requirements is a process called ***Data Fusion***.

***Data fusion*** is the process of integrating diverse information from multiple sources (such as sensors and cameras) to produce comprehensive and unified data about a more complex entity, in a way that proves more desirable than using an individual source of data (i.e. more reliable or efficient) (Chatzichristos et al., 2022).

***Data Fusion*** is a widely used technique across several fields of research when it comes to correlating data in a way that accurately describes the real world and its inner relations. This is due to the need to integrate data from different sources, as using multiple sources of data is a proven way of reducing uncertainties, imperfections, outliers or any other obstructions to meaningful data, as shown in works such as (Abdulhafiz and Khamis, 2013). Data Fusion is therefore commonly used in projects that handle heterogeneous inputs from multiple different sources.

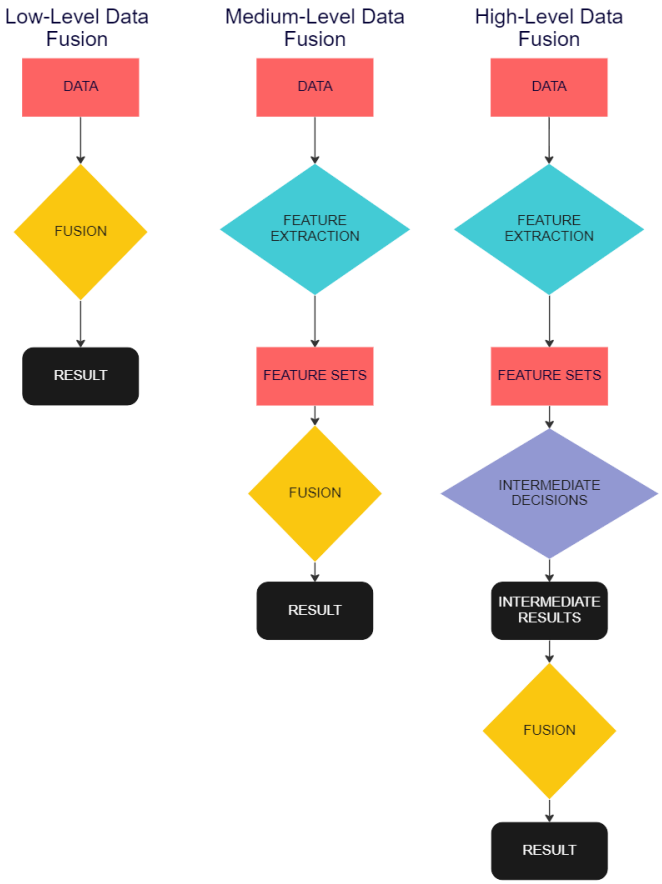


Figure 2.1: The three basic levels of Data Fusion.

There are several methodologies within the concept of Data Fusion (see fig. 2.1) that, in works such as (Kashinath et al., 2021b), (Schmitt and Zhu, 2016) and (Khaleghi et al., 2013), are commonly grouped in three levels: **low (data) level**, also called data association, **medium (feature) level**, also called state estimation, and **high (decision) level**, also called decision fusion. More recently, new types of data fusion are starting to emerge, such as Kernel Data Fusion (Smolinska et al., 2019). In this work, we will only elaborate on the three most basic and well-known types of Data Fusion.

Some examples of projects successfully applying Data Fusion can be seen through the several works done within the field of traffic monitoring, which elucidate on the use of the different levels of Data Fusion used to handle data from multiple sensors, (Kashinath et al., 2021a). (Zhang et al., 2016) delineates an approach on using Data Fusion to achieve real-time urban traffic state estimation using Global Positioning System (GPS) and loop detectors.

Within the context of traffic sensing, a Data Fusion approach would seek to reconstruct data from these sensors with the goal of finding correlations that result in better data, and therefore better decisions and results. The use of fluid GPS coordinates to monitor vehicles is something akin to the aim of our own research project, and is thus a major point of interest.

### 2.1.1 Low Level Data Fusion

Low Level Data Fusion is considered to be the simplest fusion method to achieve a combination of inputs (Smolinska et al., 2019). In this case, the data is rearranged into a new data matrix, where the different variables are placed one after the other (even if redundant). The resulting matrix will be the sum of the previously separated data.

**The goal of low level data fusion** is to improve raw data quality at early stages of data processing. It combines several sources of raw data and seeks to produce new raw data, (Floudas et al., 2007). This should in theory result in the newly fused data becoming more informative, synthetic and easier to handle than the original data. At this level, increases in data quality are achieved using techniques such as filters that perform data cleaning and de-noising, as well as estimation of missing values, or removal of outliers. Computationally-wise, this level of Data Fusion is considered to be the lightest.

An example of low-level Data Fusion can be observed through the application of Kalman Filters, which were used in the works of (Wang et al., 2014) to fuse heterogeneous traffic data, along with Gaussian mixture models. Other techniques employ the use of estimation to compensate for missing information and to reduce sampling, with (Saffari et al., 2022) showing an application of Data Fusion in large-scale urban networks, or the use of weighted averages to improve the precision of sensor measurements by correcting these same measurements with Data Fusion, as seen in (Yang et al., 2019) where it was applied to object speed measuring.

### 2.1.2 Medium Level Data Fusion

Medium Level Data Fusion is based on feature extraction which maintains relevant variables while eliminating undesirable variables from the datasets. This can be done with a multitude of algorithms that have been developed for this purpose (Smolinska et al., 2019), (Floudas et al., 2007). The use of these algorithms often requires thorough research of the features to develop an efficient solution, however.

**The goal of medium level data fusion** is to merge and filter data so as to extract meaningful features from different sources. Unlike low-level data fusion, there's and additional step prior to the final fusion step that is common to all Data Fusion techniques. It is within this step that fusion of features extracted from the original data occurs. In other words, data reduction is applied through one of several available feature reduction methods. One way of achieving this data reduction is through Principal Component Analysis (PCA). This step is where the main difficulty of medium-level fusion lies, because of the necessity to understand which features are relevant and need to be kept, and which are not and are therefore undesirable, as well as the innumerable ways of achieving this with varying levels of performance.

(Huang and Narayanan, 2016) applied Medium Level Data Fusion within the field of healthcare, by researching on the detection of falls of senior-citizens. These falls were monitored through several different sensors in different axis, such as wearable sensors and cameras, and the data gathered by these sensors was then correlated with feature-level data fusion and support vector classification. This resulted in a higher detection rate and lower false alarm rate. Other applications of this kind of data fusion can be explored in the works by (Zhu et al., 2017), where GPS data was fused with data from users phones to estimate the time until a bus reached a designated bus stop, and (Gao et al., 2019) where video was used as the main target of the fusion process. Interestingly, on the work by (Zhu et al., 2017), it was proven that fusion does not guarantee a desirable result, and that correlation structure needs to be taken into account when elaborating a fusion algorithm.

### 2.1.3 High Level Data Fusion

High Level Data Fusion works on a decision level, and makes use of selection, inference and state estimation. This type of fusion also falls under the notion of distributed detection systems, which uses multiple sensors and estimation to identify objects, (Floudas et al., 2007). The first step of high level fusion is to fit supervised models to each data matrix. These models are regression models which provide continuous responses for the inputted data, deciding on the data class membership using high-level inference. These decisions are later combined into a complex final model. This can be surmised into a process of selecting one out of a multitude of hypothesis, while taking into account both the decisions of a given number of sensors as well as the effect which noise and interference have on these same sensors (Smolinska et al., 2019).

(Soua et al., 2016b) showcased an application of high-level data fusion by utilizing it to estimate the final destination of ongoing traffic in their works. Another work of interest within the topic of traffic monitoring is (Soua et al., 2016a), where data fusion was used along with fuzzy logic, the latter used to simulate human reasoning rather than binary logic in state estimation, with the goal of making the most efficient transitions of traffic light state within a large urban junction by utilizing state estimating.

Due to the inherent traits of High Level Data Fusion, it is widely used in several other fields outside of traffic monitoring. Classification problems such as industrial quality control and maintenance are one interesting application of decision-level data fusion, with the works of (Wei et al., 2021) as a particularly interesting example in the field of aircraft manufacturing. In the context of this research project, the most interesting work found during our research into the available methodologies for this chapter was (Texier et al., 2019), which used decision-level fusion for disease outbreak detection and surveillance. This work proved that data fusion based approaches were at least equivalent to all contemporary standard algorithms, and oftentimes even yielded a great efficiency gain.

## 2.2 Methods for the Intelligent Component

The next component is the **Intelligent System**. As stated in chapter 1, this component is meant to be an intelligent component that utilizes forms of Artificial Intelligence (AI) and Machine Learning (ML) to autonomously model data into coherent structures that can then be visualized and understood by humans, and further processed if needed.

As a **brief introduction to Intelligent Systems**, AI is a tool which enables a machine to simulate human behaviors. ML on the other hand is a subset of AI, which allows a machine to automatically learn from past data without programming explicitly for a single goal. This usually means taking data and looking for underlying trends within it through the use of a wide ranging choice of algorithms. AI are designed to make decisions, often using real-time data. Using sensors, digital data, or remote inputs, they are capable of combining huge amounts of information from a variety of different sources, and act on the insights derived from this data, in some circumstances even without any form of human insight or supervision. This allows the users of AI to access automation capabilities that are next to unlimited.

**In the context of this contribution**, it is seen as of great importance to be capable of dealing with an huge influx of data in real-time in a most efficient manner. The Intelligent System Component is required to handle a dataset made up of heterogeneous data from the previous components, the bulk of which are events made up of simple geo-located coordinates. It would then proceed to analyse and mold these data so that they can be used by the following components. More precisely, the Intelligent System Component needs to meet the following requirements within acceptable time limits for real world use:

- Receive a dataset of events from the other components of the Fireloc System;
- Organise events within the dataset by their similarity to each other;
- Prioritise certain events depending on their content and who submitted them;
- Leverage redundancy and handle duplicate events and submissions;
- Allow for a Sequence/Progression of events, or in other words, deal with the aging of events.

There are several contemporary AI techniques that are capable of fulfilling these requirements, and not all of these requirements call for the use of ML to achieve the most efficient solutions, since ML can easily result in additional computational loads. The following sub-chapters elaborate on these requirements based on the methodologies we found to be most commonly employed to solve similar issues. The expected uses that ML is meant to have in this research project are, firstly, in the field of Clustering data for visualization of similarities between events and density in geo-location, and secondly, the use of deep-learning in the assignment of priority to events, as well as the management of redundancy and duplicate events. Finally, we also expect to use forms of Data Fusion and Data aggregation to handle incoming inputs from FireLoc. Due to the necessary emphasis on automation, following trends and leveraging data, it was decided to narrow down the relevant techniques to be researched and evaluated to the ML techniques that are elaborated in the following sub-chapters. These were found to be decisively relevant candidates for handling event geo-locating and monitoring problems such as the one described in this document, having already been used to solve similar problems such as those described in (Song et al., 2010), (Haldi Widianto et al., 2020) , or (Afyouni et al., 2022).

### 2.2.1 Novel Approaches to Clustering

This work defines a clustering approach as novel in the case it employs the use of algorithms which weren't created, or aren't commonly used strictly for the purpose of clustering data. Examples of these are Markov Random Fields (MRF) and Bayesian Networks (BN), which see widespread usage across several research fields with roles unrelated to clustering.

#### Markov Random Fields

MRF, also known as Markov Networks, are a form of representing dependencies. They describe a system by local interaction and denote features of a system by using terms representing their spatial or contextual dependencies. Markov networks are also undirected and can be cyclic, which allows this model to represent infinite loops in its dependencies. A benefit of MRF is that these types of networks are designed based on both statistical and structural information that

standard clustering methods tend to neglect (Wang et al., 2013). In the case of our research project, this would allow for more specific grouping when using MRF for clustering.

An example of this benefit would be the following scenario: if, although point A is closer to B, and as such standard clustering methods such as K-means group them together, there could be other more relevant ways of grouping these points such as A with C, when taking into account topology and elevation. We refer to this as the Elevation Problem, which is exemplified in figure 2.2, with the expected K-means grouping in red, and a custom MRF in green.

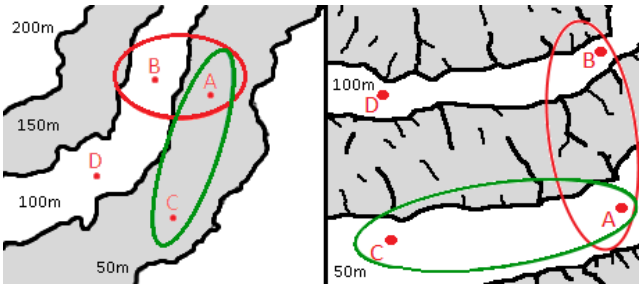


Figure 2.2: Examples of the elevation problem, from two perspectives.

Within this context, it would make more sense to group point D with point B and point C with point A, this being due to the nature and accessibility of the terrain. MRF are also widely used in the field of terrain mapping due to this characteristic, as seen in works such as (Tse et al., 2015).

MRF also face certain setbacks, however. They are computationally demanding, and while there are many algorithms that are capable of optimising these networks, one of which is *Iterated Conditional Modes*, but this then adds another layer of complexity to the solution. MRF are often used in image processing and computer vision. Using these networks for clustering is a novel approach which is well documented in (Song et al., 2010). Other interesting works that use this type of technique for clustering purposes are often found in the field of medicine, thanks again to the higher accuracy of this technique. Examples of these works are (Suliga et al., 2008) where the goal of MRF is to find clusters of pixels that represent cancerous masses. Another more relatable research project which used MRF for clustering purposes is (Li et al., 2021), however. In this specific work, MRF were used to cluster social events for organisers in the context of event-based social networks, with the goal of event management. This specific work proves us that MRF could effectively be used in the context of our research project. Overall, it can be concluded that MRF are more robust and detailed than standard Clustering methods, at the cost of computational performance, proven by the results shown in the previous works. This means that MRF are the better option when accuracy takes precedence over any other metric when evaluating an algorithm.

#### Bayesian Networks

BN are probabilistic models that share similarities with MRF. One difference, however, is that Bayesian networks are directed and acyclic, so they can induce dependencies between events, making these models ideal for predicting the contributing factor of an event, or in other words, patterns. Unlike MRF however, they can't handle infinite loops within them (cycles), and so if the data was generated from a model where several variables correlate to each other then BN wont be able to model this relationship (Ben-Gal, 2008).

A relevant example would be to predict relationships between diseases and their symptoms, one of many examples being in cancer research. Such a case was extensively described in (Zhao et al., 2021), where BN were used with the goal of revealing molecular structures of tumours so that these could then be further researched. In the previous research article, the performance of BN proved to be better compared to the other algorithms it was up against, mainly due to the inherent pattern-finding traits of this particular type of network. BN, not unlike MRF, are also considered computationally demanding. All branches must be calculated in order to calculate the probability of any one branch. Not only that, there is no commonly accepted way for creating BN from data, all while they are considerably difficult to create, requiring "*a priori*" knowledge for achieving the most efficient solutions. In works such as (Pham and Ruz, 2009), it was demonstrated that, at the cost of extra computation power, BN prevailed over standard clustering techniques such as K-means, which were shown to be 10% less accurate.

Another interesting work that uses this type of technique for clustering purposes is (Marek et al., 2014). In this specific work, spatial analysis was applied to medical datasets with the goal of mapping disease events through clustering. According to this work, Baysean algorithms are already widely used to smooth data so that become easier to spot. This work created clusters both in the spatial and space-time planes, with the clusters depicting the risk of health anomalies in a density map.

### 2.2.2 Standard Clustering Techniques

There are many clustering techniques described in contemporary literature, such as Probabilistic, Partitional, Spectral or Grid based Clustering, amongst others. The two main Clustering techniques considered for this contribution are Centroid-based and Density-based Clustering, which are the most well-documented and commonly used techniques.

#### Centroid-based clustering

Centroid-based clustering organizes the data into non-hierarchical clusters.
Centroid-based algorithms are simple, efficient and scale well to large datasets. Examples of this technique include **K-means**, which is the most widely-used

centroid-based clustering algorithm. K-means aims to partition data into clusters in a way where each individual observation belongs to the cluster with the nearest mean to a cluster centroid. The initial k centroids are randomized, and as the cluster grows the center is recalculated (LEDU, 2018), (GeeksforGeeks, 2023). An example of the employment of K-means over a dataset of points can be seen in figure 2.3. This technique is often used in Market and Image Segmentation, but is far from limited to these fields.

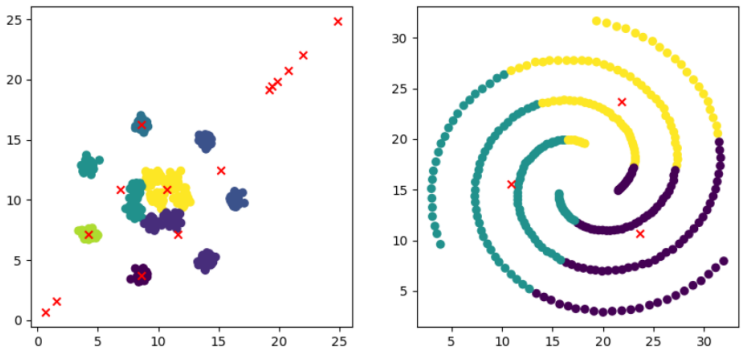

Figure 2.3: Example of Centroid Clustering (K-means) applied to two datasets.

Some of the issues with K-means include sensitivity to initial conditions and outliers, requiring manual setting of an optimal K-value and distance metric, and the inherent randomness to K-means, which can result in less efficient results on the short-term. These variables influence the shape of the clusters. K-means also has difficulty handling clusters of varying density and/or arbitrary shapes.

When it comes to event detection, use-cases of K-means can be seen in works such as (Oladimeji et al., 2015), where K-means was applied with the goal of event detection within systems such as fire alarms. This work achieved the goal of event detection by performing data aggregation on the nodes created by K-means, which clustered the data around two possible labels (the two possible outputs of the system) and then followed with pattern recognition utilizing Convolutional Neural Networks (CNN). This work concluded that utilizing this combination of methodologies significantly improve fire detection performance when compared with what were defined as standard approaches: Feed-Forward Neural Networks (NN) or Naive Bayes Classifiers.

### Density-based clustering

Density-based clustering connects areas of high density into clusters. Examples of this technique include **Density-based Spatial Clustering of Applications with Noise (DBSCAN)** and **Ordering Points to Identify the Clustering Structure (OPTICS)**. These are mainly used to find relevant associations and structures within data and focus on density approaches to said data. These are very simple to implement, only requiring two initial inputs, the minimum size of a cluster, and the maximum distance between its members. Density-based clustering allows for arbitrary-shaped distributions as long as dense areas are present in the dataset (Dey, 2023). By also having a notion of noise, density-based clustering

is, by design, more robust to outliers since it does not assign them to clusters. Unfortunately, these algorithms still have difficulty with data clusters of varying densities, much like K-means. The previous dataset of points utilized for the K-means example can now be seen in figure 2.4 being clustered with DBSCAN.

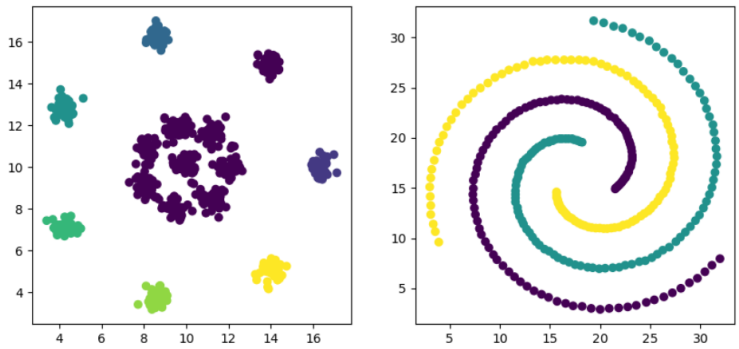


Figure 2.4: Example of Density Clustering (**DBSCAN**) applied to two datasets.

The main theory behind the DBSCAN algorithm is quite simple: a point belongs to a cluster if said point is close to other points from that same cluster. This proves to be a challenge when the dataset is sparsely populated.

DBSCAN has several variants. One of these, **Hierarchical Density-based Spatial Clustering of Applications with Noise (H-DBSCAN)**, which turns the standard DBSCAN algorithm into an hierarchical clustering algorithm. A big difference between these two variations is that H-DBSCAN assigns would-be noise points to the cluster that provides the highest overall stability and/or density, resulting in a lower amount of noise points. H-DBSCAN is more robust thanks to this, as it is now able to better handle areas of varying density, and thus allowing for a more flexible exploration of the dataset. The initial variables are also not static for the run-time of the algorithm, unlike the standard DBSCAN. The initial values adapt to varying data density and shape as it gets processed (Leland McInnes, 2016). In essence, H-DBSCAN builds a tree-shaped structure which explores clusters within multiple levels of granularity.

**OPTICS** is yet another density-based clustering algorithm (AlindGupta, 2023). While it's similar to DBSCAN, OPTICS creates an ordering of the data points based on their density connectivity, the **reachability plot**. This plot utilizes both a core and reachability distances. Reachability is a measure of how easily one point of data can be connected to another one, within the density threshold. Areas with low reachability distance indicate dense cluster regions, while areas with high reachability distances represent sparse cluster regions, or even noise. This reveals the structure of the data, and so allows for the identification of clusters at different levels of density. In other words, unlike DBSCAN, there is no predefined density threshold. In its place, density is determined implicitly based on the reachability distances. OPTICS is less sensitive to the variables set at the start of the implementation, and provides a more flexible exploration of clusters compared to the standard DBSCAN.

When it comes to using DBSCAN, we can see that it has in fact been used in works similar to this contribution. Some include (Haldi Widianto et al., 2020),

which elaborates on the use of DBSCAN with the goal of identifying disasters through Twitter. DBSCAN has also been used in both (Karanja, 2016) (Anwar et al., 2019), where DBSCAN was employed to identify areas with wildfire risk, utilizing historical data of wildfire hot-spots as occurrence points, and then calculating risk based on cluster density. This makes this technique particularly relevant in the context of this documents research topic. Another interesting work in the context of using clustering to observe trends is (Cerezo-Costas et al., 2018), which used geo-located social media posts to help detect and understand unexpected behaviors in urban areas in real time, some examples being abnormal patterns and contrasting location densities which could signify a multitude of activities. This was achieved using several social media platforms such as Twitter and Instagram, and was put to test within large urban areas such as New York. The methodologies used along with DBSCAN included Natural Language Processing to automate the understanding of the social media posts posts, and thread-based data aggregation techniques. Another interesting work in the context of geo-located events is (Huang et al., 2018), where DBSCAN was once again used with the goal of detecting events and their textual content which could then be used for a multitude of research purposes, such as marketing or geo-social studies. This work focused on Twitter posts as a source of data, and utilized DBSCAN to cluster tweets according to their spatial and temporal characteristics. Afterwards, these tweets were subjected to analysis by a text processor. The results proved to be promising, as using data collected from four college cities over the span of two years resulted in the identification of several events that occurred within that time-frame.

In conclusion, when it came to using density-based clustering, we could see that these previous works show several common traits with the goals of our own research project, and thus we found them to be very enlightening in the context of handling event detection in both the spatial and temporal spaces. Being standard clustering techniques means that these suffer from some inherent problems when it comes to accuracy. The earlier elevation problem that MRF solved still stands when using DBSCAN for example, so caution is needed when dealing with three-dimensional data, or two-dimensional data that uses elevation data, such as with topographic maps. Depending on the accuracy needed, either in geo-location coordinates or local terrain characteristics, density clustering may not be a viable methodology for certain works.

## 2.3 Methods for Data Visualisation

The final stage of our modules' data processing involves displaying the final form of the data into a visual interface. This interface is not the final Fireloc interface. The function of this interface is to allow for the visualization of the current state of the data, mainly for testing purposes. It can, however, come to influence or guide the final form of the Fireloc interface.

As for the requirements set for this part of the module, they mainly involve the display of the processed events onto a real-world map, with a high degree of

accuracy. It must allow for the visualization of the events' inner data, such as its keywords, the users who contributed to said event, or the submissions that the event is made up of. There is also a need for the display of the events' evolution over time.

There are many ways of visualizing data within the programming language that was used in this research project, that is, Python. The libraries which were researched for this include the **"Matplotlib"** lib and the **"Folium"** lib, which will be the focus of this sub-chapter.

### Matplotlib and Folium for Data Visualization

Matplotlib is a popular and widely used data visualization library in Python. It provides a comprehensive set of tools for creating static, animated, and interactive visualizations. Pyplot is the main tool to be used from this library in the context of our work. Pyplot is a module, which was made popular due to its data visualization capabilities and ease of use. It provides a simple interface that is capable of creating a number of different plots, such as line, bar, or scatter plots. It also allows for more specialized plots, such as histograms. Most of the works researched in the context of this project utilized this library in one form or another as a means of displaying data. Examples of this library can be directly seen in the examples given for the application of standard clustering algorithms to datasets.

Folium is yet another Python library used for creating interactive and customizable maps. It's built using the leaflet[2] library, and allows the easy visualization of spatial data within web-based maps. This makes it an ideal choice for visualizing the geospatial data that our events are based on. Folium provides a simple and intuitive interface for creating maps, requiring only the initial location and a zoom level for the simplest map. More complex maps can be then created by adding more features to additional layers, such as markers, lines, or heat-maps, amongst others. Folium is also very easy to integrate with other python components and libraries.

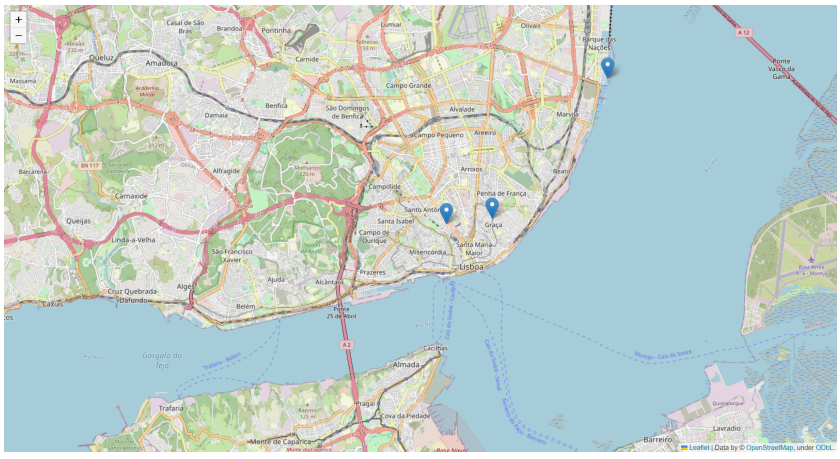


Figure 2.5: Example of **Folium** with three events visible in blue.

[2]More information on leaflet at https://leafletjs.com

Effectively, Folium meets all the requirements of this project when it comes to data visualization. It simplifies the process of visualizing geospatial data, and provides a wide range of customization options. All of this makes Folium a viable choice for handling the role of data visualization within our project. Both Folium and Matplotlib examples utilizing real-world data could be examined in (Becca_R, 2019). Examples include geolocation, traffic congestion in the USA, the level of bike infrastructure, and finally, a way to visually correlate all of these features within a single map.

## 2.4 Review on the State of the Art

This section will briefly go over and summarise the chapter on Background, Concepts & State of the Art. The first section of this chapter, Data Correlation, goes over the Data Correlation portion of the work. This section focuses on Data Fusion, and some of the techniques used to achieve it. By the end of this section, and after the research of several works of similar nature, it could be expected that the type of Data Fusion to be used would be a mixture of Medium and High Level Data Fusion. This would be further elaborated upon in the following chapter, Methodology.

The second section, the Intelligent Component, went over several methodologies to process and group data of similar nature. More importantly, this chapter goes over clustering techniques, which are the first step to data processing within the module being created in this research project. This section went over several types of clustering, which were implemented and tested in the following chapters.

The third section, Data Visualisation, went over possible ways of displaying the now processed information. Both Matplotlib and Folium ended up being used in this research project. Matplotlib proved to be essential during the testing and prototyping phase of this project due to its various tools. Folium on the other hand, became the front-end of the module created during this research project as a contribution to Fireloc.

# Chapter 3

# Development Process

In this chapter, we elaborate on the work performed to prototype on the methodologies considered within chapter 2, the State of the Art, and underline the initial work done to enable the development of the module that this project aims to create. This chapter will also go over the several iterations of development on the module, explaining the methodologies, data, tests, metrics, and any other considerations that arose during said development.

As previously stated, this project aimed to create a module that can be used both as a stand-alone program and synchronously with other modules within the Fireloc system. Its primary objective is to receive pre-processed data about occurrences, which may or may not be complete, and use it to create more complex entities which are, theoretically, more accurate. This is to be achieved through Data Fusion. To aid in the process of Data Fusion, several methodologies needed to be considered, and subsequently selected based on how and whether they met the necessary requirements.

The research and development process undertaken during this project will be elaborated upon in the following sub-chapters, which divide the overall work into concise, goal-defined iterations. A process of testing and prototyping occurred during and after each iteration. This process had the goal of confirming the theory developed during the state of the art, and confirming the results of each iteration, along with the several tests done throughout them. This also served the purposed of enabling us to take conclusions on the several methodologies, their performance and results. These results served both as a guideline on the current state of the project at the time, as well as a basis to plan the next development iteration. The results can be found on the following chapter, **4 - Experiments and Results**.

## 3.1 Early Work and First Development Iteration

The first iteration involved the viability testing of the several methodologies at hand, as well as the creation of a dataset that could be used to simulate the expected inputs that this module would receive from the Fireloc system. The main

goal of this iteration was to confirm the State of the Art, as well as getting the various methodologies to work with our datasets. The selection of the methodologies to be used in the final version of the module also happened in this iteration.

### 3.1.1 Dataset Generation

The first step in this iteration was the creation of two datasets. A complete dataset, **Dataset A**, with all the data types that are expected under the set requisites in a real-case scenario, and a simpler dataset, **Dataset B**, which was made up of two numerals representing coordinates, two Boolean's representing the presence of fire and/or smoke, and a string of text representing a "keyword". Dataset B is meant to be used only as an input in debugging and proof-of-concept tests. On the other hand, Dataset A is meant to represent real-life submissions of new occurrences, so it needs to account for the possibility of missing information within some submissions. While the "keywords", "district", and "parish" variables may or may not be missing from submissions due to being acquired from optional user-generated text, the coordinates (latitude and longitude) and time variables are mandatory and automatically acquired at the time of submitting an occurrence. Variables generated internally by Fireloc are also never missing. This is the case for ID variables (user and submission), and variables related to the quality of interaction of users with the Fireloc app (user rating). Variables signaling the presence of fire and/or smoke resulting from image processing are always present as well, albeit holding either a positive or negative value. Both datasets have 150 unique members.

For Dataset A, its data was generated by limiting the generation of coordinates to the longitude and latitude limits of certain districts of mainland Portugal, such as Coimbra, Lisbon, or Porto. The submission and user IDs were generated incrementally, and for each unique user ID, an unique user rating was also generated. Time was also generated incrementally, based on the date and time of the machine being used. Boolean's were assigned values randomly, and a dictionary of strings was used to randomly choose keywords related to the topic of wildfires. The assignment of districts and parishes happened after the generation of coordinates due to these being limited by district. Parishes were random, but assigned based on the district they belonged to at the time. An effort was made to have a generous amount of missing data, and a small amount of erroneous data was also added, such as wrong districts for a given area. In summary, Dataset A will have the following structure:

```
sub. id|date-time|user id|user rating|fire verified|smoke verified|latitude|longitude|district|parish|keywords
```

Figure 3.1: Expected structure of the modules' inputs.

For Dataset B, due to the lack of focus on data of high accuracy, the generation of data involved a simple semi-random generation of points of data within certain boundaries, with a subsequent addition of "noise". These points were then complemented with the additional information for Dataset B. That is, the two Boolean's, which were also randomized, and the string of text, which once more utilized a dictionary of strings to randomly choose keywords from.

Finally, a random generator of coordinates was also created so that each algorithm could each be stress- tested using a number of highly-different input datasets. Since this function was only to be used with the clustering algorithms, only coordinates were considered for the testing.

### 3.1.2 Clustering Methodologies

Clustering was deemed the most efficient way of separating data based on the most important metric in regards to similarity between submissions, that being latitude and longitude. The latitude and longitude variables are always present in a submission, and they hold the highest weight in deciding whether a submission is similar to others or not.

There are a number of clustering algorithms available, both as pre-made python libraries and as standardized approaches to using tools such as Neural Networks for the sake of clustering. The following clustering methodologies were considered:

- Baysean Neural Networks;
- Markov Random Fields;
- K-means Clustering;
- DBSCAN/iDBSCAN Clustering;
- OPTICS/iOPTICS Clustering;
- H-DBSCAN Clustering.

Utilizing clustering techniques shows promise, as they're seen as an efficient way of grouping points of data utilizing distance metrics. This was proven during the research done in chapter 2, where most works of similar nature to this research project utilized a form of data clustering. Furthermore, these clustering algorithms also have the added benefit of already being implemented within several python libraries, such as ***SciPy*** and ***pyclustering***, which highly-optimized implementations written in low-level languages.

The first milestone within the clustering algorithm testing and prototyping phase was to have an implementation of each of the previous algorithms in a state that made them capable of running Dataset B. The only metric used to group submissions at this point was the distance between points of data. Since most clustering algorithms are already implemented within the aforementioned python libraries, a simple initialization of the clustering parameters was all that was needed to start utilizing these algorithms. Having them in a working state, the next step was to tinker with the several variables that affect the outcome of the clustering algorithms.

Firstly, the novel techniques which were considered in the state of the art. Unfortunately, these techniques could not be fully implemented in a timely manner.

This is due to the constraints of creating a custom implementation of a clustering algorithm within the set time-frame, as well as the necessity of having pre-clustered training data for the methodologies utilizing Neural Networks. On the other hand, when taking into account MRF implementations, we noted that many of the implementations of previous works, that were the basis for our own MRF implementation, utilized standard clustering techniques at one point of their implementation on their adjacency matrices. This is due to the fact that standard clustering algorithms are more computationally efficient and easier to implement. By combining MRF-based methods with standard clustering, excessive computational weight is bypassed to a degree. We therefore decided to follow these works and implement MRF with the aid of K-means Clustering.

When it came to centroid-based clustering, it was decided to again implement K-means, due to its commonality, as well as ease of implementation. From this technique, we expected that the results would be very randomised. This would be mainly due to the initially random centroids, which then evolve over time as the clusters grow. Other variables included the minimum number of members of a centroid, which also plays an important role in this technique.

When it came to density-based clustering, several algorithms were chosen for testing: OPTICS, DBSCAN and H-DBSCAN. In both OPTICS and DBSCAN, the maximum distance between points played the most important role in deciding on whether to group certain points or not. Both still need an additional variable to function. That is, the minimum amount of members to form a cluster, which for the most part was defined as 3. We found density-based clustering to show the most promise due to how it inherently handled data grouping.

One additional detail that had to be taken into account was the presence of noise, and how to deal with it. It makes no sense to ignore these points of data within our context. It is very much possible that a single occurrence will be submitted in a remote area, and ignoring such a possibility in the context of natural disasters would be both erroneous and dangerous. To solve this, a function was created which iterates over the points of data labeled as "noise" by the clustering algorithms, and proceeds to decide on what to do with a "noise" point using one of the two following possibilities:

- Should the "noise" point be within a set threshold distance from any given cluster, the point becomes a member of the nearest cluster;
- Otherwise, the point is saved and labeled as a possible starting point to a cluster, and is prepared for a special-case Data Fusion procedure.

This solved the issue of noise in the clustering results, and proved to show satisfactory results with both Dataset A and B, as well as with random inputs. Having solved the issue of noise, there was no longer a risk of "ignoring" possible remote occurrences of natural disasters.

Another requirement set for the module was to be able to handle real-time submissions. This proved to be an interesting optimization problem, due to the fact

that it is not known for certain when these submissions arrive, and in what quantity. The standard clustering algorithms don't have any mechanism to cluster a single new point of data without iterating through all the existing points. This is non-optimal, as iterating through hundreds or thousands of points every time a single point needs to be processed is an extremely computationally expensive operation. In an attempt to understand how to solve this issue, stream-clustering and incremental-clustering were researched, along with the **iDBSCAN** and **iOPTICS** algorithms, in specific. Unfortunately, these algorithms are still in their prototyping phases, and as such are yet to be made readily available. Non-the-less, there exists an algorithm that allows for efficient single-point clustering operations, that is, **H-DBSCAN**.

Having a working instance of H-DBSCAN was the next step in our clustering prototyping. While this algorithm isn't available in the previous python libraries, it is readily available in its own library of the same name. H-DBSCAN, unlike its non-hierarchical counterpart, has 2 methods available for clustering data. The first method is a standard, static input of data that applies clustering to a dataset. The second method enables us to cluster new data after the initial clustering is applied. Following the testing with Dataset A, an additional function was created with the goal of calling this second method. This function is called within a "while true" infinite loop, which simulates the real-time factor of data inputting.

### 3.1.3 Data Fusion Procedure

Having clustered the points of data based on their distance to each-other, the process of applying data fusion has been made much more efficient. Taking into account the context of natural disasters, it only makes sense to consider fusing occurrences within a certain radius of the first one. Clustering has solved this issue, therefore the next step is to process the data of all the clustered points. In this development iteration, the fusion process only involves simple processing of the variables within each point of data. As a result, it produces an "**Event**", that's made up of, and holds the information of at least one or more occurrences.

This is how, during this iteration, Data Fusion dealt with the various data within the submitted occurrences:

- ID variables are all saved within an array, should they be unique, and a new unique ID is created for said event;
- The newest date becomes the event date, while all other dates are saved within an array, to be used as an event history;
- User ratings are used to calculate an average rating for the event;
- The Fire and Smoke checks are updated accordingly, taking into account all of the occurrences within the cluster. Positives can never become negatives;
- Event Coordinates are updated through the average of all the coordinates within the cluster, thus producing a centroid;

- Text variables, such as districts, parishes, and keywords, are all turned into 2D arrays, where the first half holds a string of the data, and the second half holds a weight value, which is calculated through the rate at which said string is referenced within the cluster.

In the end, this results in an event whose structure is a dictionary made up of keys, which are the type of variable stored, and values, which are the results of the data fusion as described above, for said variable (key).

### 3.1.4 Plotting and Mapping of Data

Plotting the information stored within the clustered and fused data proved to be a simple affair by using two different python libraries. The first library to be used was "**matplotlib**", which allowed us to use "**Pyplot**"'s scatter functions to display the before and after of the several tests done through this iteration.

The second step involved mapping the more realistic data from Dataset A. For this, it was decided to utilize **Folium**. This python library processes whatever data it receives through python and maps the results in realistic maps through "Leaflet". This allowed us to accurately map events in specific locations of interest, such as natural reserves or forested areas. After processing the data, Folium allows the drawing of event icons that act as pop-up buttons onto a map, thus allowing for a simple GUI that enables the user to see all the information regarding any mapped event. Folium is used through a number of web-browsers, and can either be instanced for as long as the program runs or saved within an HTML file that can then be opened or updated. During this iteration, only the most basic information was displayed.

## 3.2 Second Development Iteration

The second iteration involved the actual development of the contribution module, along with deeper testing and prototyping of the chosen methodologies for each component of the module. These were: H-DBSCAN for the intelligent component and a focus on Feature-level Data Fusion. In addition to this, we proceeded to the creation of a dataset that simulated the 2017 October fires which happened in Portugal throughout the fall. This Dataset would be used both for testing purposes, as well as to simulate real field-testing. Dataset A, which was also improved, was also complemented with an additional dataset of the same structure and generation, Dataset C. All of these developments will be elaborated in the following sections.

### 3.2.1 October Dataset Generation and Dataset C

As previously stated, two new datasets were created for the purpose of the second development iteration. These were Dataset C, which is an extension of Dataset

A, created through the same process, and that is simply meant to be read in parts so as to simulate real-time batches of information being handed to our new module. The second dataset was a modified version of a dataset on the 2017 October fires in Portugal, originally created by another Fireloc Team by using the tool. We chose this time-period because these fires have become infamous due to their gravity at the time. This dataset uses real events and real coordinates, so we hope to utilize it as a form of "field-testing".

Both of the new datasets include the exact same variables as Dataset A, but in the case of the October fires dataset, some variables were missing, and therefore had to be generated. This was because the original October dataset was initially created by another Fireloc contributor focused on text analysis, and so was missing information unique to other Fireloc modules, such as ours. These variables were:

- - User ID - which was randomly generated;
- - User Rating - which was randomly assigned to an user ID. Each ID has an unique rating;
- - Smoke Confirmed - randomly generated;
- - Keywords - Only the keyword "incendio" was present in the dataset, so keywords from our keyword dictionary were randomly added to 20% of the dataset, while "incendio" was removed from 80% of the dataset.

Furthermore, some variables from the original October dataset had to be processed in specific ways. Those were:

- - Date Time - Which had to be processed to our format: day/month/year hour:minute;
- - District and Parish variables - which had to be found and taken from the text variable included in the original dataset, which also included other pieces of text - a separate script was created to handle this.

This resulted in the creation of a new October fires dataset of our own, which is the final dataset to be used in our testing and prototyping efforts.

### 3.2.2 Utilizing H-DBSCAN Clustering

Having updated the data, the next step was to fully implement the clustering phase of data correlation. The previous iteration set H-DBSCAN as the best candidate for our clustering algorithm, so we proceed to implement the following functions based on it:

- The "Fit Predict" Function - Which when the module is started, clusters all the available data, and creates a clusterer instance trained in the dataset in the process. This is a standard clustering algorithm implementation;

- The "Approximate Predict" Function, which uses the instance to try and cluster points to already existing clusters;

Approximate Predict is especially useful due to the fact that it's a significantly computationally light process. It can't, however, create new clusters [1]. It can only assign data to existing clusters, or label it as noise. Our clustering process takes advantage of this. When new data is received, an attempt is made to assign it to existing clusters through Approximate Predict. Should this not succeed, the data is added to a queue until a re-clustering threshold is met and calls the Fit Predict function again. This threshold can be set as a time value, such as every hour, or as a queue length value, which re-clusters once the queue reaches said threshold. On a real-world application, both should be used so as to handle periods of extreme data submission, of both low and high input density.

One further modification was done to our clustering implementation: To solve the issue of noise within the vicinity of clusters, a threshold was created which represents a second clustering distance. This distance is used to calculate whether a noise point is relatively close to an existing cluster, or if it's totally isolated in an area of low density. Should a noise point be within this threshold of an existing cluster, this point will be added to said cluster. On the other hand, if there are no clusters in the vicinity, this point will be added to a "noise cluster" which will be later used in the data fusion procedure. This helps to solve the issue of density variations in the cluster borders without ignoring submissions or creating several 1-member clusters.

### 3.2.3 Improving the Data Fusion Procedure

The next step was to refine the data fusion procedure, taking into account the results of the first iteration. Two primary functions were created for this goal, along with several other auxiliary functions. We will first elaborate on the two main fusion functions:

- The "Data Fusion" Function, which applies our standard data fusion procedure to clustered data;
- The "Noise Handler" Function, which needs to handle the processing and fusion of any leftover data labelled as noise.

The Data Fusion function receives the previously clustered data, processes it through the use of several auxiliary functions, and creates a structure to store the fused data.

Variables that are used for identification or history purposes are directly handled within this function. These include the processing of user IDs, event IDs, user ratings, date-time variables, and the Boolean's that confirm fire and smoke. While

[1] See https://hdbscan.readthedocs.io/en/latest/prediction_tutorial.html for an in-dept example

date-time and ratings are used by other auxiliary functions, these are also saved so as to keep track of all the data related to each submission, and to simplify finding specific submissions. Date-time is further used to assign a latest date to the event, which is the newest submissions' date, along with an event age, with the oldest submission. ID variables are saved so as to know which users contributed to an event, and which submissions make up said event. An event rating is also calculated through an average of its user ratings. The rest of the submission variables are handled by the auxiliary functions, and then saved saved within the event. The auxiliary functions are, and have the following purposes:

- The "Time-Decay" function, which calculates the age of a submission and assigns a weight based on it, giving more importance to newer submissions. A submission loses half its value after 24hours and then quickly trends to zero. Decay as it is, is limited to 20%, but can be changed to other values or used as a threshold to reject older submissions. This time decay is used along with user rating to calculate a "***Submission Weight***";
- The "Weighted Haversine Centroid" function, which calculates the centroid of an event through all its submissions. This function uses the Submission Weight factor to give more importance to newer submissions from users of a higher rating;
- The "Location Processor" function, which handles the location data (districts and parishes) present in the submissions. It keeps count of the existing data on location names, and calculates a percentage of submissions pointing to that location. If a fire advances its front towards a new district, these weights will shift over time as new submissions are received, modifying the final percentage calculation. It uses the Submission Weight factor to prioritize newer submissions;
- The "Keyword Processor" function, handles keyword text in a similar way to location text, but with the exception that several keywords may be present in a submission, and that each keyword has an unique weight assigned to it. This assigns more weight to keywords that signify a higher hazard (i.e. "smoke" vs "chemical plant" or "gas station"). Should a keyword become more relevant at one point, the keyword weight naturally increases, and therefore so will the keyword percentage. Again, Submission Weight affects the final values of these variables;
- The "Event Hazard Level" function, which calculates a simple hazard level based on the keywords hazard weight and the Boolean variables present in the event, assigning a higher hazard level the more data describing hazards is present. At the time of this document, three levels were created: Low, Average, and High.

### Handling Submissions Labeled as Noise

The Noise Handler function on the other hand, needs to take care of the fact that outliers always exist, and due to dealing with natural disasters, no submission should be ignored and possibly cause vital information to not be displayed. In essence, the Noise Handler is always run after the Fit Predict function. The goal being that, once this function receives any existing leftover noise data, it processes it into single-member clusters labeled as isolated events. The fusion procedure is then applied to these events as normal, but with slight modifications due to the fact that variables involving averages and other calculations that require multiple submissions need to be approached differently.

All of the variables that are used for history/logging purposes simply hold the data of the single submission that makes up the event, while the auxiliary functions are applied normally, with the exception of the centroid calculation which is redundant for a single coordinate. Their results reflect the fact that a single submission can only result in a less accurate, one-sided event. They are labeled as such, and are identifiable through their negative IDs, which start at -1, and are visually different (explained in the next Folium Improvements sub-section).

### Event Variable Summary

Through the process of Data Fusion, an event creates, or comes to hold the following variables:

| Variable | Description |
|---|---|
| Event ID | Unique ID which allows users to identify the event/incident |
| Event Hazard Level | A calculation on how much the event poses as an hazard, utilizes its keywords and booleans |
| Date - Latest Update | A variable which displays when a certain event was last updated with a new submission |
| Date - Age | A calculation of the age of an event from creation until current time |
| Date - History | A log of dates at which an event was updated |
| User IDs | A log of user IDs which contributed to a certain event |
| Submission IDs | log of submission IDs which make up a certain event |
| Rating Average | An event trust rating calculated through contributing user ratings |
| Smoke Confirmation | A boolean which confirms the presence of smoke |

| Variable | Description |
|---|---|
| Fire Confirmation | A boolean which confirms the presence of fire |
| Latitude | North–south coordinate position, in degrees |
| Longitude | West–East coordinate position, in degrees |
| Districts | High-level administrative location of where an event is perceived to be placed at by users |
| Parishes | Medium-level administrative location of where an event is perceived to be placed at by users |
| Keywords | Worded descriptions, by users, of the overall area and landmarks near the event |

Table 3.1: Summary of which processed data is held within an event

### 3.2.4 Improving Data Visualization through Folium

Having updated the event structure to hold the new data, it now needed to be displayed within the Folium instance in an easy to read format. In this subsection, the modifications which were made to the Folium implementation will be elaborated upon.

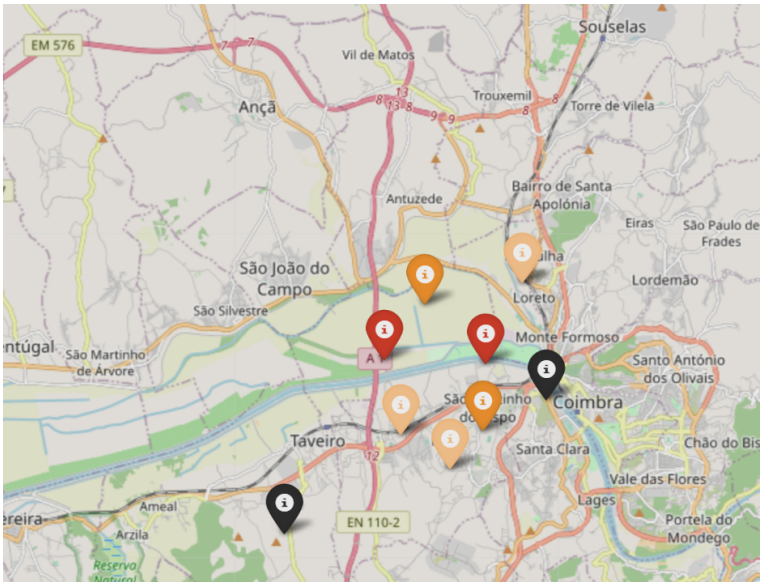


Figure 3.2: Example of fused events.

The first goal was to make events both more visually appealing and differentiable. For this, it was decided to add a heat-based colour scheme to the icons through an events' hazard level: beige (low hazard), orange (average hazard), red (high hazard), as well as black for isolated events (noise events).

This hazard level is calculated through a simple algorithm, which utilizes the

keywords present within the events' data, as well as the booleans which signal the present of smoke and/or fire. In the case of no keyword being yet present in the events' data, or at most having smoke confirmed, an event is labeled as low-priority. Should fire be confirmed, or keywords which signify lower-level hazards be present (which ones, are defined by the module instance owner, some default values are present), then the event is labeled as a medium-priority incident. Should highly hazardous keywords be added, such as explosive materiel, chemicals or bio-hazards, then the event becomes high-priority. Which keywords are deemed highly hazardous is defined by the module instance owner, some default values are present.

This simple visual aid made finding specific events within a group easier, and doubled as a warning to both the inaccuracy of isolated events, and to the existence of priority (red) events.

Following this, improvements to the display of information were implemented. The text box was widened, and information was spaced and formatted so as to allow better readability.

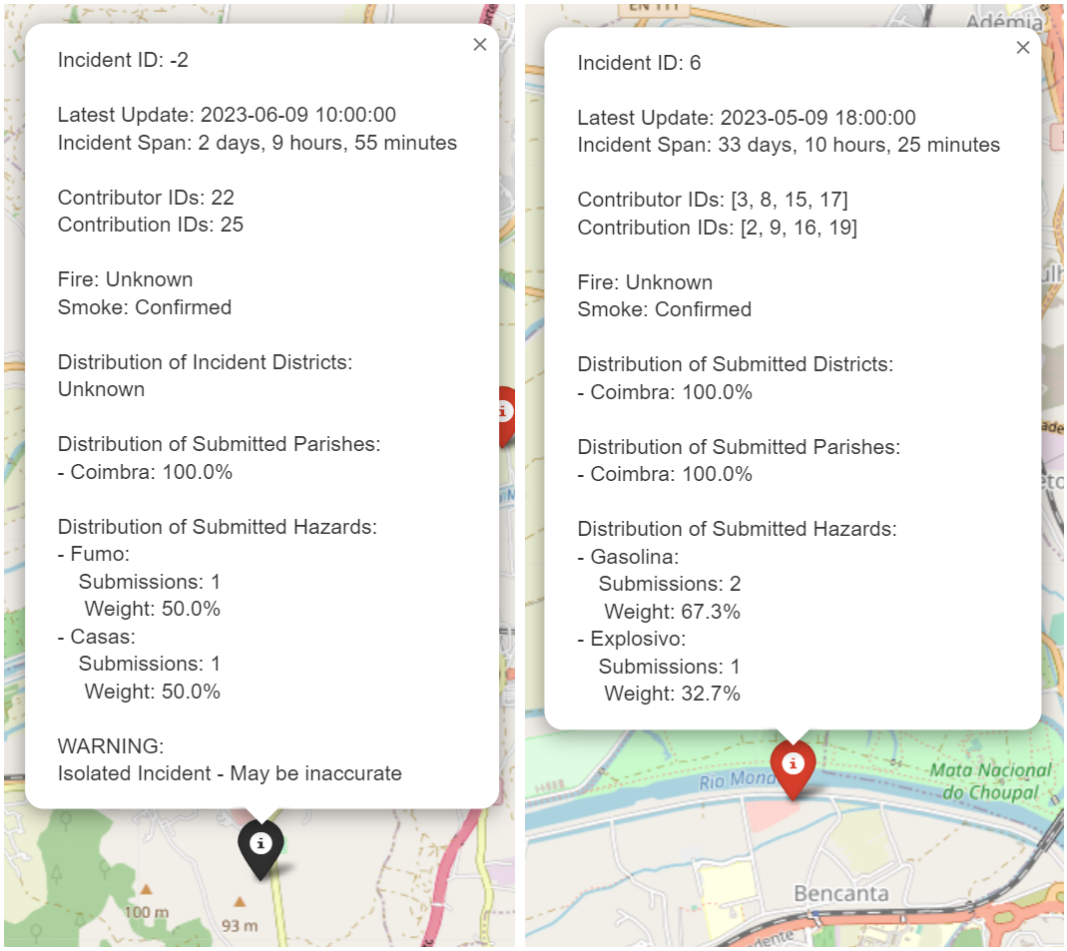


Figure 3.3: Example of the improved text pop-up with the relevant event data.

Other quality-of-life improvements were implemented, such as fixing the zoom level of icons to be able to see them with the map zoomed out, and an automatic addition of a text-based warning for isolated events.

Most of these modifications to our Folium implementation were done in parallel to the implementation of the auxiliary functions which were elaborated on in the previous sub-section, such as the Event Hazard Level function resulting in an implementation of colour-coding.

Three different Folium functions were created in total, with two being used mainly for debug purposes, displaying all the submissions and colour-coded clusters pre-data fusion, and the last one being the main implementation to be used by our module. A save function was also implemented so as to allow the user to choose between updating the same map or creating a map slideshow.

# Chapter 4

# Experiments and Results

This chapter presents the different tests performed throughout this research project, along with their respective results and the conclusions taken from them. The first half of this chapter, Lab-Testing Results, explains the several tests that were carried out during the development iterations of our module. This includes all tests done with artificial data. This is data that is randomly generated or otherwise created within a lab environment. On the other hand, the second half is reserved for tests pertaining to the stage of this project which was past the final development iteration. More precisely, stress-tests under "real-world" conditions, with "real-world" data that was generated by various human sources. In this section, we include the data that was collected from the 2017 October fires, which took place in Portugal.

## 4.1 Synthetic Data Results

This section of the chapter presents the tests carried out during the development iterations of the module. Each subsection will expose the tests carried out through a given iteration, examine their results and finalize with a conclusion on these same results and how they influenced the next iteration.

### 4.1.1 Initial Algorithm Testing & Selection

This subsection goes over the tests done after the initial development iteration. These tests were done with the goal of understanding the various algorithms at our disposal, as well as to choose which one performed the best within the context of our research project.

Clustering Testing was based on two factors: Clustering speed and visual clustering quality. Speed tests utilized 100 datasets and calculated the average time to cluster them successfully. Quality tests focused on specific datasets and were based on a visual analysis of the results of the clustering process, where one compared all the end results of the clustering process and selected the result with

the most desirable outcome: Least amount of noise, most consistent/continuous cluster shapes. For the speed datasets, the input was randomly generated, while for the quality datasets, both randomly generated and Dataset B were used. During early testing, several algorithm initialization values were used so as to try and achieve the best results for specific datasets during quality testing. Such was the case with using smaller cluster sizes and distances (when applicable) for dataset B, due to the smaller number of points, and also because of calculations based on degrees.

### Clustering Speed Testing

For the performance tests, each algorithm was run 100 times, and each of these 100 iterations used the same randomly generated data for each algorithm. The amount of clusters generated was 20, while each cluster was made up of 40 points. A variable "noise_factor" controlled the dispersion of the clusters. Testing was done from very dense cluster shapes up until no clusters became discernible (0.5, 1, 5, 10, 20, 30, 40). Higher values will lead to more dispersed points, while lower values will result in more compact clusters. In other words, higher values result in more extreme density variations in the dataset.

On the initialization of the clustering algorithms, the following settings were used as a starting point:

- K-means & Markov - n_clusters=20, n_init='auto'
- DBSCAN - eps=15, min_samples=10
- OPTICS - eps=15, min_samples=10
- HDBSCAN - min_cluster_size=10, min_samples=10

These values focused on pushing the algorithms to higher computation times rather than get the best results possible.

The results of the various performance tests can be analyzed in the following table:

| Clustering Algorithm | Factor 0.5 | Factor 1 | Factor 5 | Factor 10 | Factor 20 | Factor 30 | Factor 40 |
|---|---|---|---|---|---|---|---|
| K-means | 0.0086 | 0.0092 | 0.0096 | 0.0094 | 0.0106 | 0.0568 | 0.0809 |
| DBSCAN | 0.0062 | 0.0062 | 0.0058 | 0.0075 | 0.0071 | 0.0060 | 0.0090 |
| OPTICS | 0.4099 | 0.4559 | 0.4402 | 0.4451 | 0.4462 | 0.4338 | 0.5082 |
| H-DBSCAN | 0.0126 | 0.0180 | 0.0172 | 0.0177 | 0.0177 | 0.0130 | 0.0190 |
| Markov | 0.2274 | 0.2325 | 0.2743 | 0.2907 | 0.2523 | 0.2104 | 0.3141 |

Table 4.1: Algorithm speed results - varying levels of dispersion

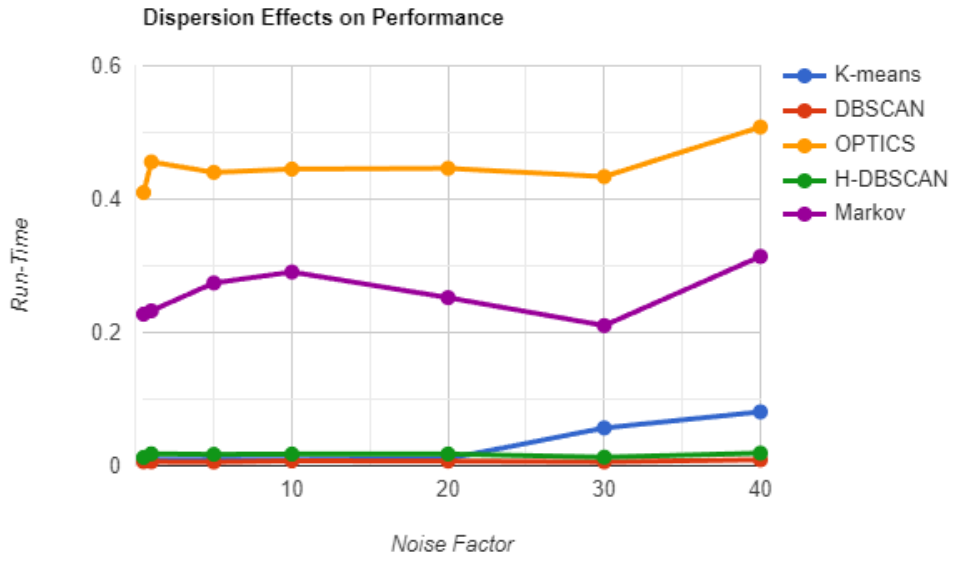


Figure 4.1: Line-graph - Effects of dispersion/noise factor on performance.

The influence of dataset size in the performance was also confirmed, as can see in the following table. There is no expected dataset size, in the sense that a dataset will be as large as the number of user submissions. To test this, datasets of size 800, 1600, 2500, 5000, 10000, and 25000 were used, with the final value of 25000 being an extreme case. We base this on the fact that the 2017 October fires dataset has around 8000 entries dispersed throughout several weeks.

| Clustering Algorithm | Size 800 | Size 1600 | Size 2500 | Size 5000 | Size 10000 | Size 25000 |
|---|---|---|---|---|---|---|
| K-means | 0.0568 | 0.0881 | 0.0703 | 0.0648 | 0.0718 | 0.0758 |
| DBSCAN | 0.0060 | 0.0150 | 0.0249 | 0.0618 | 0.1536 | 0.4578 |
| OPTICS | 0.4338 | 0.8906 | 1.6002 | 2.9602 | 7.2011 | 28.0004 |
| H-DBSCAN | 0.0130 | 0.0354 | 0.0509 | 0.1087 | 0.2324 | 1.2569 |
| Markov | 0.2104 | 1.1674 | 2.8284 | 9.2303 | 60.4741 | 291.0427 |

Table 4.2: Algorithm speed results - varying dataset sizes

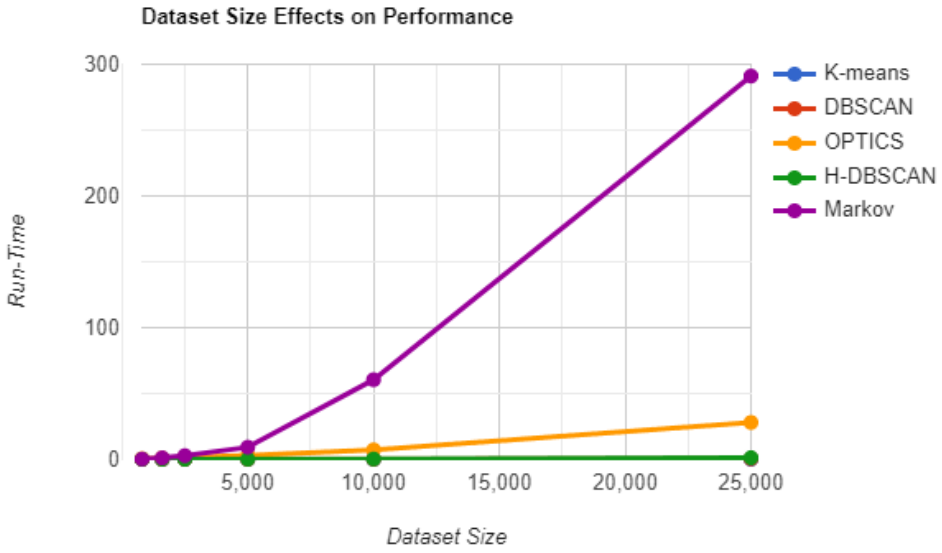


Figure 4.2: Line-graph - Effects of dataset size on performance.

### Clustering Quality Analysis

When it comes to "quality", the clustering settings highly affect the results of the tests and how the algorithms behave towards the datasets. These values were adapted throughout the testing so as to get the best general results possible for specific datasets and densities. Randomly generated datasets were used along with Dataset B so as to analyse the clustering results with and without a bias (where one knows which clusters to expect). It is important to note, however, that algorithms which require less modifications are preferable, due to the real-time data input factor.

Analysing the "quality" of a cluster is highly dependant on the context of the problem and dataset. In the case of our research project, on a standard scenario, we expect that submissions will form consistent clusters based on locations and their surrounding areas. In this context, a desirable output will involve labeling as few noise points as possible in low density areas while trying to create clear, coherent borders out of the areas of higher densities, which we assume are urban areas or areas with high traffic. The amount of clusters generated per dataset was 10, while each cluster was made up of 40 points. For each noise factor, 5 datasets were generated and tested. The noise factors used were 10, 20, 30 and 40.

On the initialization of the clustering algorithms, the following settings were used as a starting point:

- K-means & Markov - n_clusters=10, n_init='auto'
- DBSCAN - eps=10, min_samples=5
- OPTICS - eps=10, min_samples=5
- HDBSCAN - min_cluster_size=5, min_samples=5

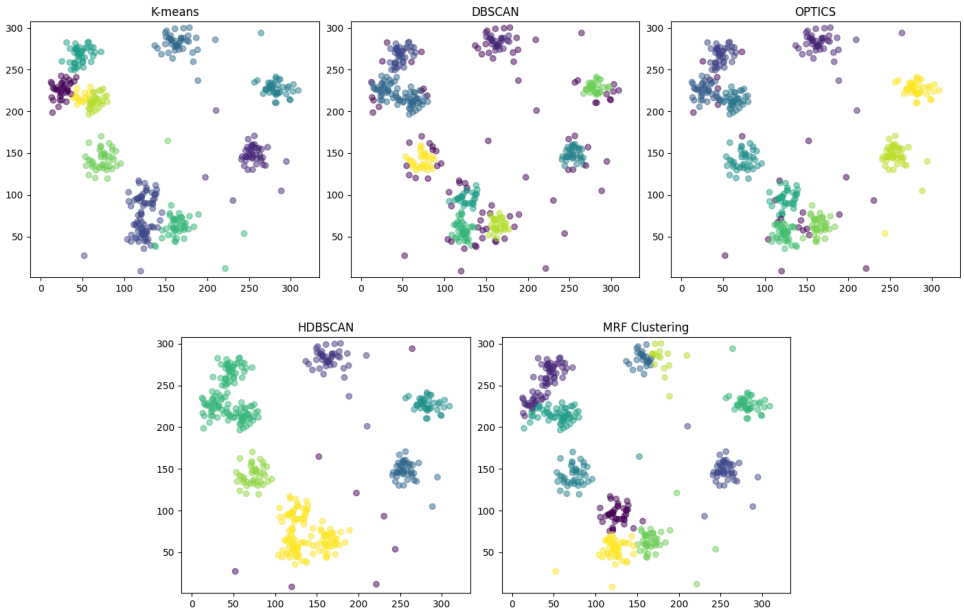


Figure 4.3: Clustering test, noise factor of 10.

On the tests done with a density factor of 10, or in other words, with dense clusters, we can see that every algorithm produces a low-noise solution, with well-shaped clusters. It was necessary to drastically raise OPTICS's eps and min_samples (both to 15), to get acceptable results. The same occurred with DBSCAN, having to raise the eps to 15 as well.

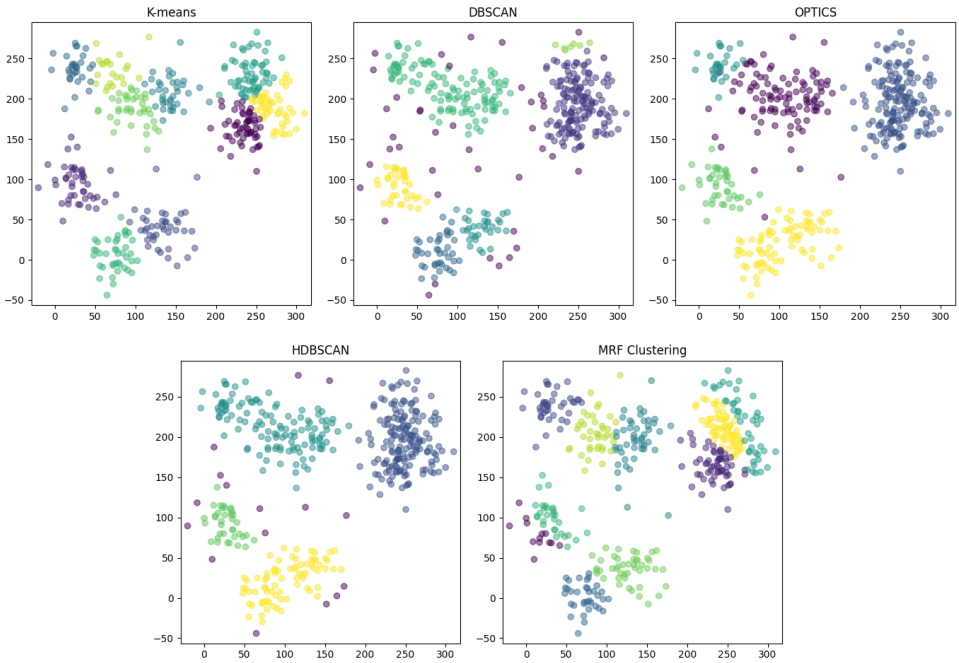


Figure 4.4: Clustering test, noise factor of 20.

On the tests done with a density factor of 20, clusters are spread out to a medium degree, and with a higher variation in density. OPTICS isn't able to produce a

low-noise solution without again raising both eps and min_samples (both to 25). K-means, Markov and H-DBSCAN produce the most well-shaped clusters.

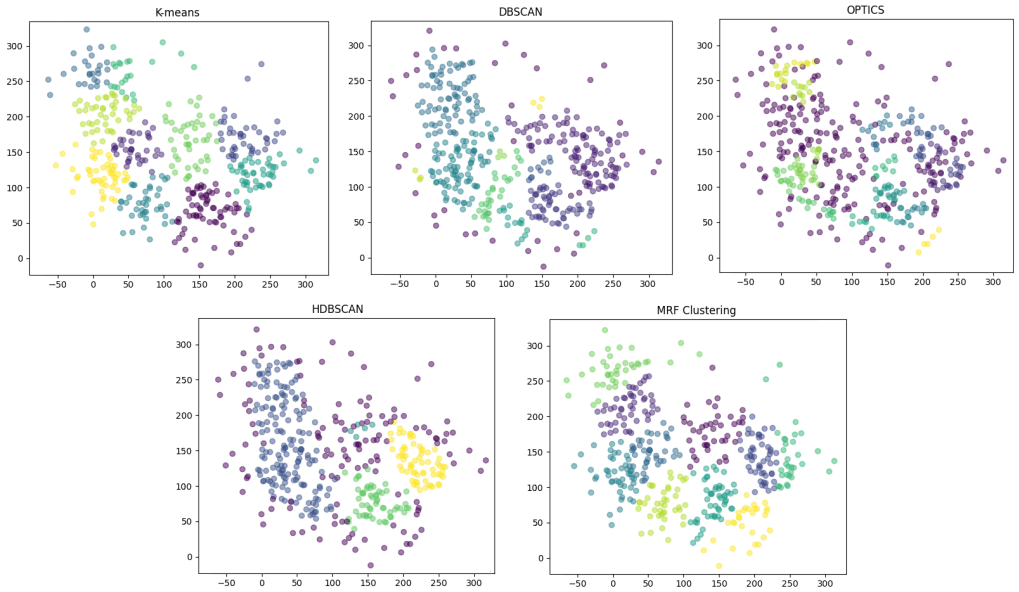


Figure 4.5: Clustering test, noise factor of 30.

On the tests done with a density factor of 30, clusters are even more spread out and have an even higher variation in density. OPTICS remains behind other algorithms. H-DBSCAN and DBSCAN produce similar results, but while DBSCAN produces less noise, H-DBSCAN produces a lower number of clusters, which also are of higher quality. K-means and Markov are able to form random clusters with ease. All algorithm values remained the same.

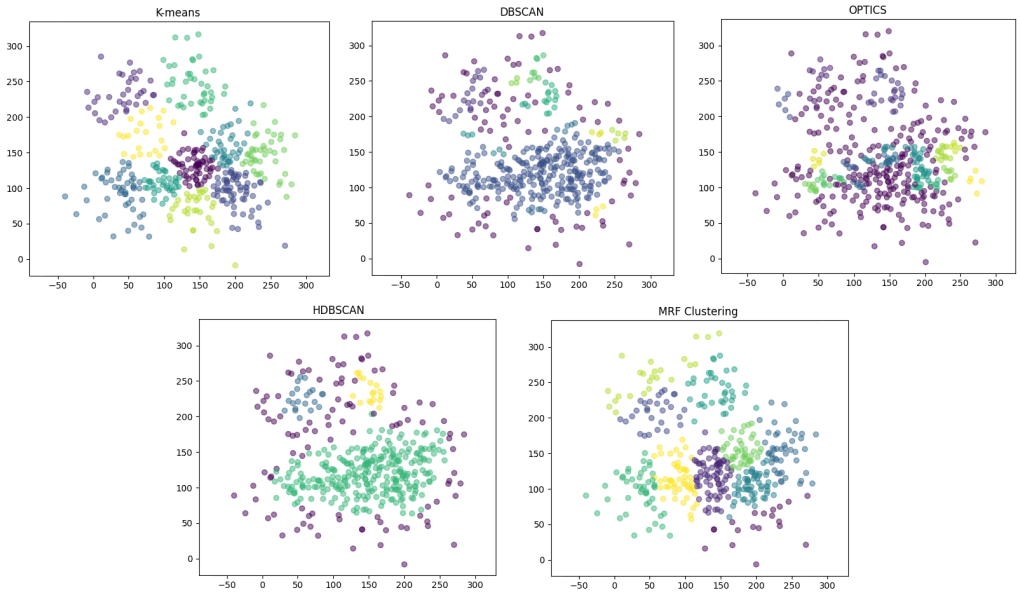


Figure 4.6: Clustering test, noise factor of 40.

A density factor of 40 was the maximum level of dispersion used for testing. After this value, clusters become rare in density-based algorithms, often forming single

super-clusters. K-means and Markov are again able to form random clusters with ease. OPTICS remains unable to create quality clusters, instead finding points of high density within large quantities of noise. H-DBSCAN and DBSCAN produce similar results. Both tend to form super-clusters, but H-DBSCAN appears to form a lower number of "useless" clusters around the main cluster. While several settings were tested, the results did not deviate far from the settings used at the start of these tests.

These results enable us to analyze how the several algorithms behave in regards to different cluster shapes and densities, and then to evaluate the results based on how much noise was created. As an additional test, we visually assign noise points to clusters, and try to understand whether the final clusters form logical gatherings of incident submissions.

Following this, Dataset B was used as the next input. Dataset B is made up of randomly generated points which, for accuracy purposes, were based on real coordinates. Two levels of inputs were considered, where the first level focused on high density points, or in other words, as if looking at the entire "country" on a map. The expectation was that it'd produce few clusters, mainly around the "cities".

For the following settings, this test produced the following results:

- K-means & Markov - n_clusters=5, n_init='auto'
- DBSCAN - eps=1, min_samples=3
- OPTICS - eps=2, min_samples=5
- HDBSCAN - min_cluster_size=5, min_samples=5

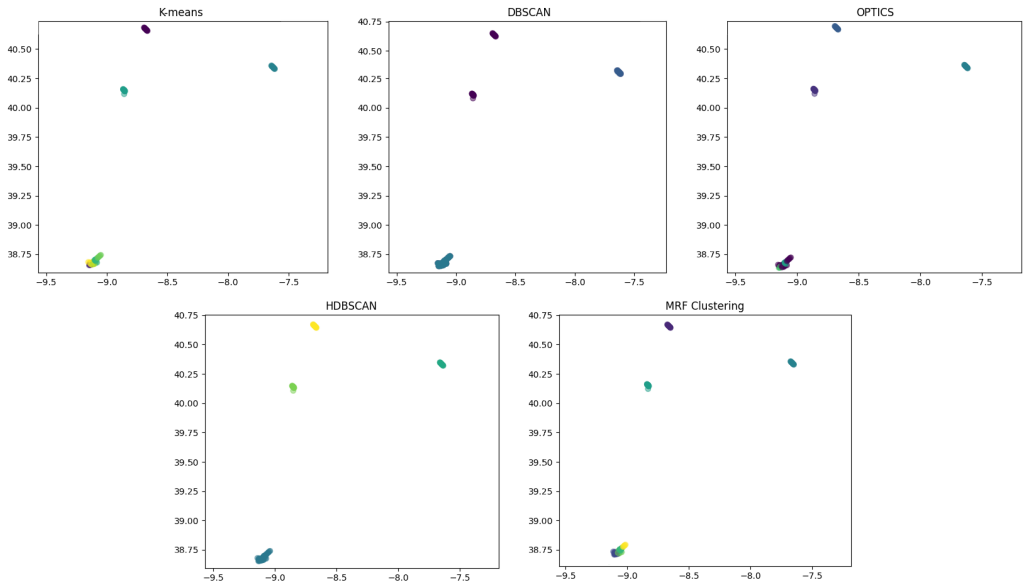


Figure 4.7: Clustering test for dataset B - northern half of the country.

In this test, DBSCAN and H-DBSCAN produced similar results. OPTICS produced decent results, but there are "useless" clusters which can be found in the

lower left corner of the graph. K-means and Markov both produced good, yet random results, as to be expected.

The second level focused on lower density points, as if looking at a single region or city. This would mean a higher dispersion of points and some occurrences of noise. The goal was to see how the algorithms fared against more sensible data, with much smaller variations in distances between points.

For the following settings, this test produced the following results:

- K-means & Markov - n_clusters=5, n_init='auto'
- DBSCAN - eps=4, min_samples=3
- OPTICS - eps=4, min_samples=5
- HDBSCAN - min_cluster_size=5, min_samples=5

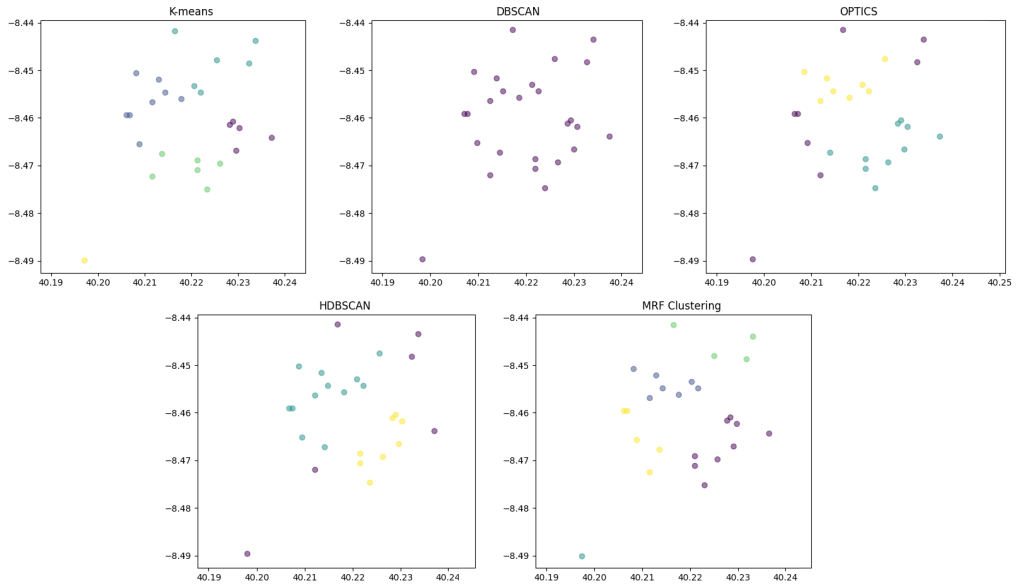


Figure 4.8: Clustering test for dataset B - single region (Coimbra).

In this test, DBSCAN was not able to form more than a single cluster. All "eos" and "min_sample" values between [1-10] were used and mixed. Curiously, OPTICS produced decent results. We assume this is due to the low amount of points of data present in the city dataset. H-DBSCAN produced results that were near equal to OPTICS, but managed them with lower noise levels. K-means and Markov both produced good, yet random results, as to be expected.

The tests using real-world data required the settings for both OPTICS and DBSCAN to be revised and their values lowered, while H-DBSCAN kept the settings from the earlier tests.

### Data Fusion Testing

Having tested the clustering implementations, we moved to the testing of the early data fusion implementation.

The clusters which were created from the previous tests utilizing dataset B had to be fused into a single entity, their information variables saved for logging, and their incident variables had to be processed and fused. The first step was to apply fusion based on coordinates. This meant that the cluster became a single point whose centroid was the average of all points of the cluster. In the case of OPTICS and H-DBSCAN, the noise (-1) cluster was also fused.

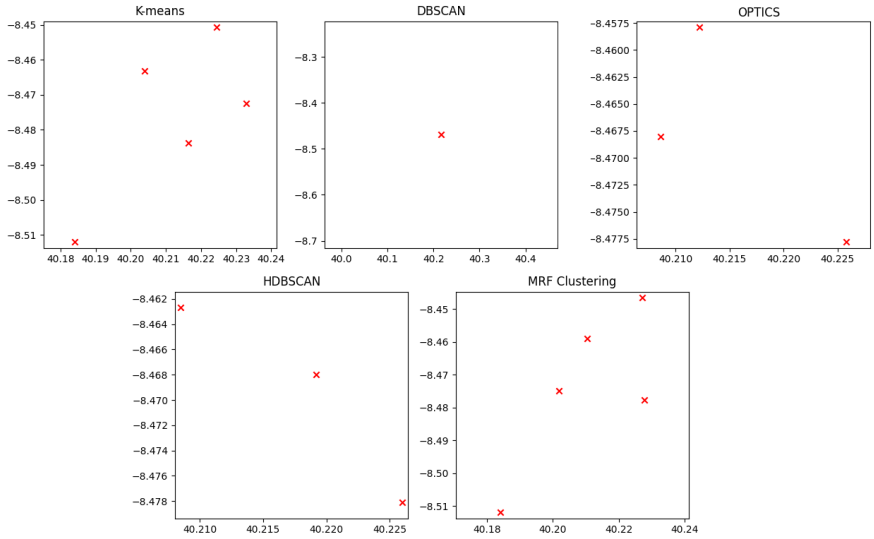


Figure 4.9: Data Fusion test for dataset B - single region (Coimbra).

These results show us how clustering could be used as a process of aggregating similar submissions for data fusion by calculating the centre of an events' expected location. Comparing the overall area of the cluster with the fused centroid, we can see that clustering allows for a consistent and accurate guess of what submissions belong to, or make up an event simply by utilizing the available coordinate data. For the fusion procedure itself, a standard average calculation was used to calculate an approximation of where the events' centre would be. When it came to the remaining variables, these were simply saved within lists and other placeholders, or light calculations were applied to benefit the most recent data.

This was not optimal for the accuracy of the events' data. The next logical step would be to prototype on how to include these variables in the data fusion procedure. Before moving to this, however, an implementation of Folium had to be created to display the existing data and to analyse the accuracy of the coordinates of the event. A cluster was hand-made around the Beaches in "Figeira da Foz", with the goal of originating a fused event in an exact spot. Five different submissions were added to an input file. These originated the following incident:

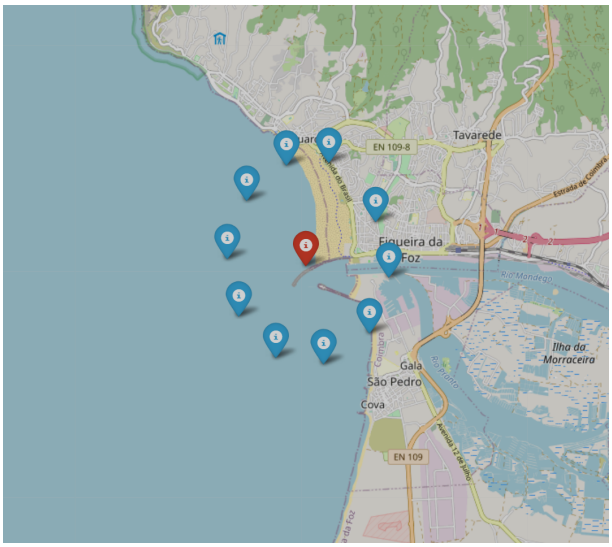


Figure 4.10: Accuracy test using H-DBSCAN, Data Fusion, and Folium.

As can be seen, we were able to accurately place and cluster several blue submissions in a circle around the targeted area using H-DBSCAN. Once Data Fusion was applied, a red event was created in the centre of the targeted area.

### 4.1.2 Discussion on the Results of the First Iteration

The goal of the initial tests were to prove the viability of the techniques being considered for our project, as well as to then proceed to choose the best techniques.

When it came to Clustering, we could see that both OPTICS and Markov lagged behind in terms of performance. Since the first dataset tests were reasonably small in size, all algorithms still managed to cluster the data within a second of run-time, but it is evident that the larger the datasets become the higher run-times will be. This is because these algorithms have a linear time complexity at best (K-means), and exponential at worst (OPTICS). Markov, on the other hand, includes heavy calculations to create a matrix of values used to model the relationships or dependencies between data points. This raises the run-time exponentially. The middle term is represented by H-DBSCAN, which is an hierarchical algorithm with quadratic time complexity. While it was not the fastest algorithm, it fared well in all performance use cases.

On the visual results of the clustering, we could see the following for the randomly generated datasets:

- K-means and Markov managed to cluster most data, but the resulting clusters suffered from randomness of the initial centroids. This would cause clusters to form in areas where it would be most optimal for only one to exist, for example. This makes them inadequate for the context of our project;
- DBSCAN and H-DBSCAN produced similar results on higher density datasets, but H-DBSCAN handled lower density datasets better. On dataset B, the

low density and resulting noise made it so that DBSCAN was not viable;

- The OPTICS implementation could not produce favourable results without extensive revision of its initial settings for each new dataset, which makes it inadequate for handling volatile streams of data;

On the visual results for dataset B, we came to the conclusion that the randomness of K-means and Markov made them undesirable. This is because of the need to know the amount of clusters beforehand, on top of the randomness of creating said clusters, and the resulting creation of too many random fused events. We also found that both Markov and OPTICS were too heavy for the needs of our project, since these did not return an accuracy level that justified the additional increase in computational loads.

This left us with DBSCAN and H-DBSCAN as the best result-yielding algorithms. Taking into account all the results, H-DBSCAN proved to create the best results in all situations, and also yielded good performance results, with no revisions being necessary. It produced consistent clusters in various levels of density, and labeled points as noise in relatively lower levels compared to the other algorithms. It also had no need to know how many clusters existed beforehand and produced the most accurate fused events.

H-DBSCAN was therefore chosen as the algorithm to be used for data clustering.

### 4.1.3 Module Development and Prototyping Tests

This subsection goes over the tests done during and after the final development iteration. These tests were done with the goal of successfully applying the chosen algorithms, improving the accuracy of the various calculations necessary for the data fusion and clustering process, as well as to test any newly implemented features.

#### Clustering Improvements

After the implementation of H-DBSCAN, the main concern became the existence of noise, as well as the testing of its secondary clustering function, "approximate predict". Through the various tests, further adjustments were also made to the initial variables of H-DBSCAN so as to produce the best clusters with the least amount of noise.

For "approximate predict", a simple test was done upon implementation: four points were manually inputted after reading dataset A, two points close to existing clusters, and two points far from existing clusters. For this test, the results were also straightforward: "approximate predict" behaved as expected, and inputs near existing clusters were clustered successfully, while the two points far from any clusters were labeled as noise. No further tests were deemed necessary for this feature since it is unable to create new clusters, only add to existing ones.

For dealing with leftover noise, a function which acted as a second layer of clustering was created. This function iterated over any points labeled as noise, and calculated whether any of them were within a threshold of an existing cluster. Should this function deem that one point is indeed within this threshold, then said point is added to the cluster. This threshold was the target of the tests, as an acceptable value had to be defined.

The following image shows the results of both tests, with black icons signaling noise. The noise points which are close to clusters are re-clustered, while those that are too far remain labeled as noise. The addition of a new point through "approximate predict" can also be seen in the top-right corner, near "Loreto", where the beige cluster gains a new member. These results were achieved with a threshold of 1.4.

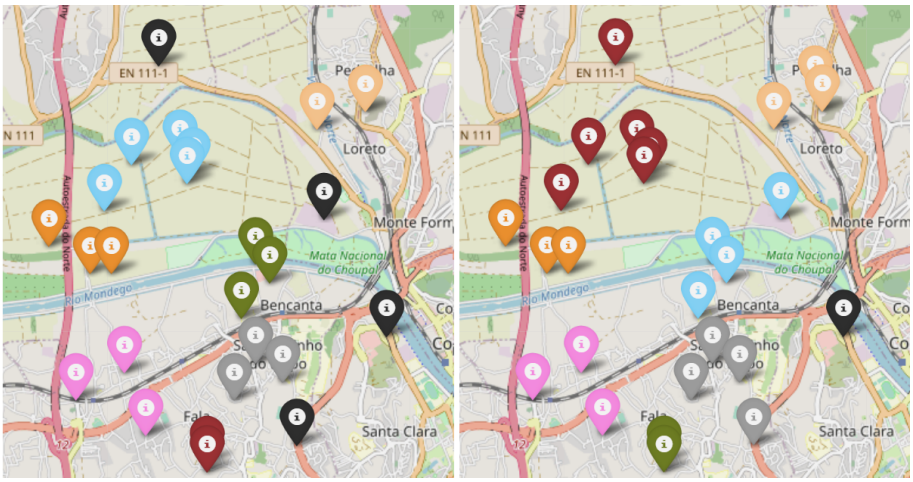


Figure 4.11: Noise re-clustering test using data from Coimbra city.

The value which showed the best results was 1.5. Values under 1.4 were too low to re-cluster noise points which we felt belonged to certain clusters, while values above 1.6 started to cluster noise points which were outright outliars which should be considered isolated points. H-DBSCAN was set to min_samples=1 and min_cluster_size=2, or in other words, clusters started at two points, with one as the core.

### Data Fusion - Haversine Calculations

While the fused centroid proved to be reasonably accurate when it came to the calculation of its coordinates, this project always had the goal of calculating the most accurate coordinates from the available data.

The current calculation assumed the points belonged to a flat space plane. This is obviously inaccurate since in reality the coordinates belong to a globe shaped entity. This means that the curvature of the Earth needs to be taken into account in the centroid calculations. An Haversine average was therefore implemented for both distance and centroid calculation, utilizing the following formula:

$$\operatorname{hav}(\theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)\ , \qquad \theta = \frac{d}{r}$$

Figure 4.12: Haversine formula, (Wikipedia)

Where r is Earths radius and d is the distance between two points. Phi is a points' latitude value and lambda is the points' longitude value. Having implemented this, a single cluster was created and the centroid calculation was applied.

| | Standard | Haversine | Difference |
|---|---|---|---|
| Latitude | 40.2300 | 40.2298 | 0.005% |
| Longitude | -8.4428 | -8.4427 | 0.006% |

Table 4.3: Comparison of centroid calculation algorithms.

These results show us that, for calculations at a "city" level, Haversine is only about 0.01% more accurate. While this value appears to be low, one needs to take into account that it also scales when processing larger areas, while the increase in computational weight is negligible. Since it is overall desirable to have as much accuracy as possible within our context, since the module is meant to be of general use, this update on distance calculations was added to the final version of the module.

**Data Fusion - Rating Weighting**

An addition to the data fusion procedure was the use of user rating as a factor on deciding the value of submissions. A function was implemented with the goal of giving more importance to submissions coming from users of higher rating. For this, a simple calculation was applied on an users' rating to convert it from the [1-20] format to a [0-1] format, which can then be multiplied with the various values present within the event. ID and boolean variables were not influenced by this factor, while coordinates and textual variables were. For coordinates, a weighted average was applied, while for textual variables, a decay variable was applied. This makes the centroid tend to the values of more "trusted" submissions.

Adding 10 points in a line within "Coimbra", where user ratings increase to the left and decreases to the right, we get the following centroids:

| | Standard | Weighted |
|---|---|---|
| Latitude | 40.2223 | 40.2280 |
| Longitude | -8.4250 | -8.4292 |

Table 4.4: Comparison of standard and weighted Haversine calculations.

This shows a significant difference when plotting the centroids within Folium. While the standard (green) centroid is dead-centre when looking at the coordinates (blue), the weighted (red) centroid tends towards the side with the coordinates with highly rated users.

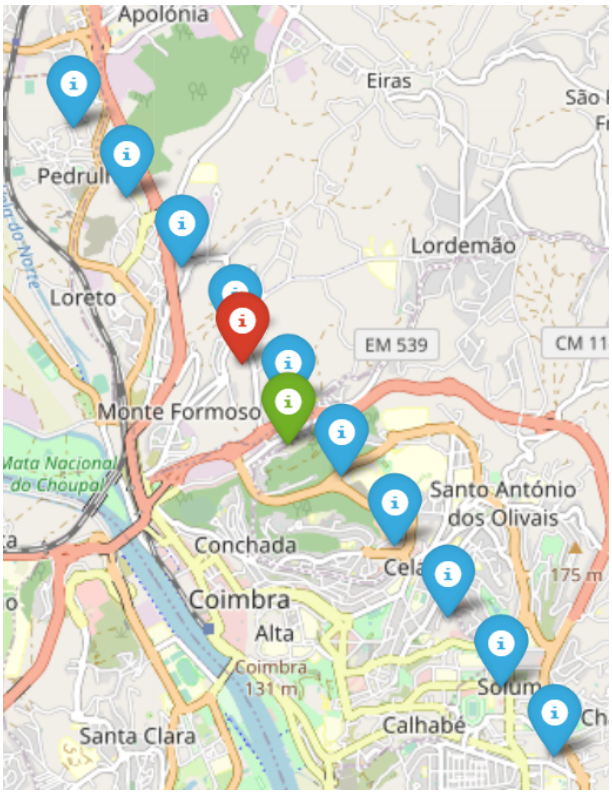


Figure 4.13: Results of the weighted and standard Haversine calculations.

The same theory was applied to the text variables. Since these are displayed through the basis of calculating a distribution percentage of which keywords are mentioned within an event, keywords mentioned by higher rated users will maintain their value, while keywords which are mentioned by lower rated users will have their value reduced.

Creating two events, one with an user with a rating of 20 (1 out of 1, or 100%), and one with an user with a rating of 10 (0.5 out of 1, or 50%), we get the following results:

```
Hazard Distribution:
Fumo: {'counter': 1, 'weight': 1}
Fogo: {'counter': 1, 'weight': 3}

Hazard Distribution:
Fumo: {'counter': 1, 'weight': 0.75}
Fogo: {'counter': 1, 'weight': 2.25}
```

Figure 4.14: Results of the weighted and standard text distribution calculations.

These results show a drop of 25% in the weights of the text values, for a difference of 10 points in rating. This value seemed appropriate for our context, but can be adjusted by the modules' user.

**Data Fusion - Time Decay**

An important requisite for our module was the factoring of age and time in a submissions value. Newer submissions should have a higher value than older submissions. For this, an approach was implemented based on the previous user rating factoring. A decay factor variable was created, along with a maximum decay cap and a custom date variable, to allow the user to choose whether to delete old submissions or let them trend to zero, as well as to choose whether to use the machines' date-time or a custom date (to, for example, allow the simulation of older events). For testing purposes, the cap was set to 20% of the submissions original value, and the custom date was put aside, since we're using data generated based on the machines' date-time.

A decay function was implemented in the same fashion as the rating system. Each time a new fusion procedure is started, the date-time values are refreshed. Any new submissions are used to update the most recent dates, and the old date-times are updated according to the machines' date-time. Upon doing this, a decay weight [0-1] is calculated utilizing these values. This decay will be multiplied with the various variables within the event.

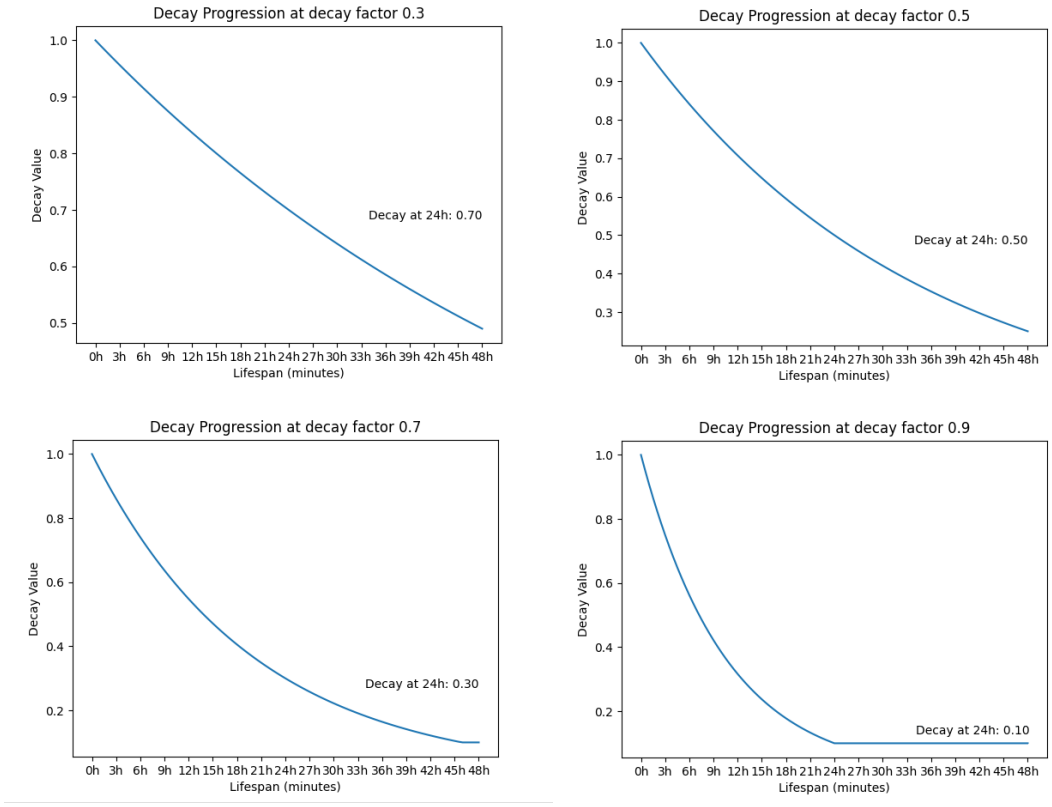


Figure 4.15: Decay progression using decay factors of [0.3,0.5,0.7,0.9].

The following results were achieved, when using a decay factor of 0.5. Three events were created, depicting the three stages of a submissions' lifespan: Collection, maturity, and EoL.

Incident Span: 0 days, 0 hours, 38 minutes

Hazard Distribution:
Fumo: {'counter': 1, 'weight': 0.92013}
Gasolina: {'counter': 1, 'weight': 4.6006}

Incident Span: 1 days, 0 hours, 2 minutes

Hazard Distribution:
Fumo: {'counter': 1, 'weight': 0.4692663}
Gasolina: {'counter': 1, 'weight': 2.3463060000000002}

Incident Span: 27 days, 10 hours, 33 minutes

Hazard Distribution:
Fumo: {'counter': 1, 'weight': 0.18}
Gasolina: {'counter': 1, 'weight': 0.9}

Figure 4.16: Results of the decay calculations on an event.

Creating an event with a single text variable, we can see that at the point of creation, it's variable has near 100% of its weight. At 24h, the weight is almost halved. While afterwards, the weight is capped at 20% of its original value.

We can also see the placement update within Folium, when an event made up of only old submissions is updated with a new submission.


Figure 4.17: Decay progression and influence of a new submission in an event.

Blue icons represent the original points, with +24 hours of lifespan, while green represents the original centroid of said points. Upon receiving a new submission with less than one minute of lifespan, coloured in beige on the map, we can see a new event centroid in red. This new centroid trends towards the new submission, which has the most weight out of all available submissions. Should, in 24 hours, a new event be added on top of the beige submission, the red centroid will trend further north. This shows us that the decay algorithm behaves as expected. For a faster weight decay, or in other words, to give even higher priority to newer submissions, one simply needs to use an higher decay value.

### 4.1.4 Discussion on the Results of Module Prototyping

.....

.....

.....

## 4.2 Real-World Data Results

.....

.....

.....

# Chapter 5

# Conclusion

It is of the utmost importance that critical systems such as those used for disaster response are not neglected or allowed to become obsolete. One way to avoid this is to complement these systems with new, modular technology. Fireloc seeks to complement these systems with modern tech, allowing the use of mobile phone systems and their own app to receive crowd-sourced submissions of people near the location of disasters. This would bring a breathe of fresh air to the true and tested, yet aging response systems.

Handling submissions of heterogeneous data to create something that can be used by the authorities requires a processing step which is not yet implemented within FireLoc, however. This is the goal of this thesis: To create a module that manages the processing, validation, and aggregation of data that results from the analysis of crowd-sourced submissions done by the FireLoc system.

Through the research done during the writing of the State of the Art, we became convinced that the best approaches to reach the goals we were given for this research project were techniques involving Data Fusion and Clustering. These techniques have many variations, and thus we focused ourselves on those that appeared most common on problems of similar nature. We found these to be Density Clustering (DBSCAN, OPTICS) and Centroid Clustering (K-means). When it came to the process of Data Fusion, a mixed approach was theorized, utilizing both concepts of medium and high level Data Fusion.

Through the first implementation of these techniques, it became clear that it was possible to obtain acceptable results through all of them. Knowing this, further tests were done to find out which ones created the most reliable output, based on the circumstances of the theme of the project, that is, wildfires. This meant higher standards in the level of accuracy and consistency of the results. From these results, it was decided to use H-DBSCAN for the first step of data grouping. This was due to the fact that H-DBSCAN produced the least noise per number of points, with the best tolerance for varying density, or in other words, because it allowed the most optimal exploration of the dataset.

The final iteration had the H-DBSCAN implementation for the FireLoc input completed, as well as a fully working Data Fusion implementation, capable of

processing all the data expected by FireLoc. In conjunction with the Folium implementation, these three main components formed the new contribution module, which was promptly put through assessment and testing.

After considering the results of these tests, both through the October Fires dataset, as well as the datasets that were pseudo-randomly generated based on real-life data, we found that, while further adjustments of the various variables of the clustering and data fusion algorithms could result in better outputs, the module reliably produced output of high quality and accuracy.

The main difficulty encountered throughout this project was the lack of a fully complete, accurate and modern dataset which could be used for testing and prototyping purposes. The October fires dataset was hindered by the fact that it was 6 years old at this point, which caused trouble when testing age-based variables and how they reacted to our data fusion approach. Non-the-less, the module still worked as intended and produced the expected results, albeit with the issue of lower variable weights due to the age of the input (+5 years).

## 5.1 Contributions

The word done throughout the span of this research project resulted in the following contributions:

- The state-of-the-art and background research documentation describing the focus of our research and the various techniques considered for this project;
- A functional and automated data aggregation module, capable of working both as an integration to FireLoc, as well as in a stand-alone mode;
- Test results of the application of this module when using the October Fires of 2017, which took place in Portugal, as well as the generated datasets;
- Suggestions on improvements to be done to this module or any future works basing themselves on it.

## 5.2 Future Work & Suggestions

The module which has been developed so far still has room for improvement. At the time of integrating the module with the FireLoc System, specific tweaks to how the module functions are called can be made to optimize efficiency, such as focusing more on clustering variables to existing clusters or re-clustering the entire current dataset. This is the same for the modules' default values, such as the clustering variable initialization, variable weights, and age decay limits. Truly optimal values will always need more real-world testing, which were not possible in a lab environment.

New Clustering algorithms are also being developed at the time of writing this document. These include stream-clustering algorithms, such as IOptics or Online DBSCAN. The fact these algorithms are built around the concept of clustering real-time streams of data makes them better options over the algorithms which were available at the time (in theory).

Additional algorithms or functions can be created to further help confirm incident placement within areas of interest. This includes, for example, an algorithm that compares the received coordinates and location-related text with a database of coordinates of all existing districts and parishes. This would then return a likelihood of belonging to certain areas, rather than basing the likelihood entirely on user-submitted data. Limiting clustering areas is also another option. By dividing the area of interest (Portugal, in our case) in smaller sectors, one can limit where clusters are formed to a degree, and thus avoid unwanted clusters in specific sectors. This would need to be further researched for viability.

Data visualisation also has room for improvement, by utilizing Folium in a more integrated way. ipWidgets could be used to upgrade the interface and add more features to the map. For example, the ability to go back in time through a date slider would certainly be useful for logging purposes. Implementing this would requite access to older map versions, which is already implemented in the current module version. Changing icons based on the keywords within an incident is also another option (i.e. changing a default icon to a barrel or explosion icon) These upgrades, however, may prove to be redundant due to the fact that FireLoc already has an interface module, so it remains to be seen how much of our Folium implementation will be used in the final integrated version of the module.

# References


Waleed Abdulhafiz and Alaa Khamis. Handling data uncertainty and inconsistency using multisensor data fusion. *Advances in Artificial Intelligence*, 2013, 01 2013. doi: 10.1155/2013/241260.

Imad Afyouni, Aamir Khan, and Zaher Al Aghbari. Deep-eware: spatio-temporal social event detection using a hybrid learning model. *Journal of Big Data*, 9, 06 2022. doi: 10.1186/s40537-022-00636-w.

Jacinto Estima Alberto Cardoso, Cidália Fonte, José Paulo de Almeida, and Joaquim Patriarca. The fireloc project: Identification, positioning and monitoring forest fires with crowdsourced data. In *The FireLoc Project: Identification, Positioning and Monitoring Forest Fires with Crowdsourced Data*, 2021.

AlindGupta. Ml: Optics clustering explanation, Jan 2023. URL `https://www.geeksforgeeks.org/ml-optics-clustering-explanation/`.

Muchamad Anwar, Wiwien Hadikurniawati, Edy Winarno, and Aji Supriyanto. Wildfire risk map based on dbscan clustering and cluster density evaluation. *Advance Sustainable Science, Engineering and Technology*, 1, 11 2019. doi: 10.26877/asset.v1i1.4876.

Becca_R. Data Visualization with Python Folium Maps — towardsdatascience.com. `https://towardsdatascience.com/data-visualization-with-python-folium-maps-a74231de9ef7`, 2019. [Accessed 21-May-2023].

Irad Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd, 2008. ISBN 9780470061572. doi: https://doi.org/10.1002/9780470061572.eqr089.

Héctor Cerezo-Costas, Ana Fernández-Vilas, Manuela Martín-Vicente, and Rebeca P. Díaz-Redondo. Discovering geo-dependent stories by combining density-based clustering and thread-based aggregation techniques. *Expert Systems with Applications*, 95:32–42, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.11.019.

Christos Chatzichristos, Simon Van Eyndhoven, Eleftherios Kofidis, and Sabine Van Huffel. Chapter 10 - coupled tensor decompositions for data fusion. In Yipeng Liu, editor, *Tensors for Data Processing*, pages 341–370. Academic Press, 2022. ISBN 978-0-12-824447-0. doi: https://doi.org/10.1016/B978-0-12-824447-0.00016-9. URL `https://www.sciencedirect.com/science/article/pii/B9780128244470000169`.

Debomit Dey. Dbscan clustering in ml: Density based clustering, Jan 2023. URL `https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/`. [Accessed 08-Jan-2023].

Nikos Floudas, Aris Polychronopoulos, Olivier Aycard, Julien Burlet, and Malte Ahrholdt. High level sensor data fusion approaches for object recognition in road environment. In *2007 IEEE Intelligent Vehicles Symposium*, pages 136–141, 2007. doi: 10.1109/IVS.2007.4290104.

Jun Gao, Yi Murphey, and Honghui Zhu. Personalized detection of lane changing behavior using multisensor data fusion. *Computing*, 101, 12 2019. doi: 10.1007/s00607-019-00712-9.

GeeksforGeeks. K means clustering - introduction, Jan 2023. URL `https://www.geeksforgeeks.org/k-means-clustering-introduction/`. [Accessed 08-Jan-2023].

Mochammad Haldi Widianto, Ivan Sudirman, and Muhammad Awaluddin. Application of density based clustering of disaster location in realtime social media. *TEM Journal*, pages 929–936, 08 2020. doi: 10.18421/TEM93-13.

Che-Wei Huang and Shrikanth Narayanan. Comparison of feature-level and kernel-level data fusion methods in multi-sensory fall detection. In *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2016. doi: 10.1109/MMSP.2016.7813383.

Yuqian Huang, Yue Li, and Jie Shan. Spatial-temporal event detection from geo-tagged tweets. *ISPRS International Journal of Geo-Information*, 7(4), 2018. ISSN 2220-9964. doi: 10.3390/ijgi7040150.

Stephen Karanja. Density-based cluster analysis of fire hot spots in kenya's wildlife protected areas. In *Density-based Cluster Analysis Of Fire Hot Spots In Kenya's Wildlife Protected Areas*, 2016.

Shafiza Ariffin Kashinath, Salama A. Mostafa, Aida Mustapha, Hairulnizam Mahdin, David Lim, Moamin A. Mahmoud, Mazin Abed Mohammed, Bander Ali Saleh Al-Rimy, Mohd Farhan Md Fudzee, and Tan Jhon Yang. Review of data fusion methods for real-time and multi-sensor traffic flow analysis. *IEEE Access*, 9:51258–51276, 2021a. doi: 10.1109/ACCESS.2021.3069770.

Shafiza Ariffin Kashinath, Salama A. Mostafa, Aida Mustapha, Hairulnizam Mahdin, David Lim, Moamin A. Mahmoud, Mazin Abed Mohammed, Bander Ali Saleh Al-Rimy, Mohd Farhan Md Fudzee, and Tan Jhon Yang. Review of data fusion methods for real-time and multi-sensor traffic flow analysis. *IEEE Access*, 9:51258–51276, 2021b. doi: 10.1109/ACCESS.2021.3069770.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14 (1):28–44, 2013. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2011.08.001.

Education Ecosystem LEDU. Understanding k-means clustering in machine learning, Sep 2018. URL https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. [Accessed 08-Jan-2023].

Steve Astels Leland McInnes, John Healy. How hdbscan works. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html, 2016. [Accessed 01-May-2023].

Xiao Li, Yashas Malur Saidutta, and Faramarz Fekri. Social event planning using hybrid pairwise markov random fields. *International Journal of Intelligent Systems*, 36, 07 2021. doi: 10.1002/int.22569.

Lukas Marek, Vít Pászto, Pavel Tucek, and Jiří Dvorský. Spatial clustering of disease events using bayesian methods. In *Spatial Clustering of Disease Events Using Bayesian Methods*, volume 1139, 04 2014.

Arup Mohanty. Impacts of climate change on human health and agriculture in recent years. In *2021 IEEE Region 10 Symposium (TENSYMP)*, 2021.

Muyiwa O. Oladimeji, Mikdam Turkey, Mohammad Ghavami, and Sandra Dudley. A new approach for event detection using k-means clustering and neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5, 2015. doi: 10.1109/IJCNN.2015.7280752.

D. Pham and Gonzalo Ruz. Unsupervised training of bayesian networks for data clustering. *Proceedings of The Royal Society A Mathematical Physical and Engineering Sciences*, 465:2927–2948, 09 2009. doi: 10.1098/rspa.2009.0065.

Eran KinsbrunerandJustin Reock. The evolution of smartphones - and web technology development, Jan 2020. URL https://www.perfecto.io/blog/evolution-of-smartphones-web.

Elham Saffari, Mehmet Yildirimoglu, and Mark Hickman. Data fusion for estimating macroscopic fundamental diagram in large-scale urban networks. *Transportation Research Part C: Emerging Technologies*, 137:103555, 2022. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2022.103555.

Michael Schmitt and Xiao Zhu. Data fusion and remote sensing – an ever-growing relationship. *IEEE Geoscience and Remote Sensing Magazine*, 4:6–23, 12 2016. doi: 10.1109/MGRS.2016.2561021.

Siemens. https://new.siemens.com/global/en/company/stories/industry/ai-in-industries.html. In *AI and Industry 4.0*, 2021. [Accessed 05-Jan-2023].

Agnieszka Smolinska, Jasper Engel, Ewa Szymanska, Lutgarde Buydens, and Lionel Blanchet. Chapter 3 - general framing of low-, mid-, and high-level data fusion with examples in the life sciences. In Marina Cocchi, editor, *Data Fusion Methodology and Applications*, volume 31 of *Data Handling in Science and Technology*, pages 51–79. Elsevier, 2019. doi: https://doi.org/10.1016/B978-0-444-63984-4.00003-X.

Ran Song, Yonghuai Liu, Ralph Martin, and Paul Rosin. Markov random field-based clustering for the integration of multi-view range images. In *Markov Random Field-Based Clustering for the Integration of Multi-view Range Images*, pages 644–653, 11 2010. ISBN 978-3-642-17288-5. doi: 10.1007/978-3-642-17289-2_62.

Ridha Soua, Arief Koesdwiady, and Fakhri Karray. Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3195–3202, 2016a. doi: 10.1109/IJCNN.2016.7727607.

Ridha Soua, Arief Koesdwiady, and Fakhri Karray. Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3195–3202, 2016b. doi: 10.1109/IJCNN.2016.7727607.

M. Suliga, R. Deklerck, and E. Nyssen. Markov random field-based clustering applied to the segmentation of masses in digital mammograms. *Computerized Medical Imaging and Graphics*, 32(6):502–512, 2008. ISSN 0895-6111. doi: https://doi.org/10.1016/j.compmedimag.2008.05.004.

Gaëtan Texier, Rodrigue S. Allodji, Loty Diop, Jean-Baptiste Meynard, Liliane Pellegrin, and Hervé Chaudet. Using decision fusion methods to improve outbreak detection in disease surveillance. *BMC Medical Informatics and Decision Making*, 19(1):38, Mar 2019. doi: 10.1186/s12911-019-0774-3.

Rina Tse, Nisar Ahmed, and Mark Campbell. Unified terrain mapping model with markov random fields. *Robotics, IEEE Transactions on*, 31:290–306, 04 2015. doi: 10.1109/TRO.2015.2400654.

Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov Random Field Modeling, Inference & Learning in Computer Vision & Image Understanding: A Survey. *Computer Vision and Image Understanding*, 117(11):1610–1627, 2013. doi: 10.1016/j.cviu.2013.07.004.

Chunhui Wang, Qianqian Zhu, Zhenyu Shan, Yingjie Xia, and Yuncai Liu. Fusing heterogeneous traffic data by kalman filters and gaussian mixture models. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 276–281, 2014. doi: 10.1109/ITSC.2014.6957704.

Yupeng Wei, Dazhong Wu, and Janis Terpenny. Decision-level data fusion in quality control and predictive maintenance. *IEEE Transactions on Automation Science and Engineering*, 18(1):184–194, 2021. doi: 10.1109/TASE.2020.2964998.

Wikipedia. Haversine formula - wikipedia. `https://en.wikipedia.org/wiki/Haversine_formula`. [Accessed 14-Apr-2023].

Junjia Yang, Shijun Wang, Xuezhu Na, and Yang Yang. A weighted data fusion method in distributed multi-sensors measurement and control system. In *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*, pages 654–658, 2019. doi: 10.1109/ICISCE48695.2019.00135.

Pan Zhang, Lanlan Rui, Xuesong Qiu, and Ruichang Shi. A new fusion structure model for real-time urban traffic state estimation by multisource traffic data fusion. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6, 2016. doi: 10.1109/APNOMS.2016.7737227.

Yize Zhao, Changgee Chang, Margaret Hannum, Jasme Lee, and Ronglai Shen. Bayesian network-driven clustering analysis with feature selection for high-dimensional multi-modal molecular data. *Scientific Reports*, 11, 03 2021. doi: 10.1038/s41598-021-84514-0.

Lin Zhu, Fangce Guo, John Polak, and Rajesh Krishnan. Multisensor fusion based on data from bus gps, mobile phone, and loop detectors in travel time estimation. *Computing*, 01 2017.