



SILVANUS

**D5.3 - Demonstration of SILVANUS decision support system
for response coordination**

1st version



This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 101037247



Project Acronym SILVANUS

Grant Agreement number 101037247 (H2020-LC-GD-2020-3)

Project Full Title Integrated Technological and Information Platform for Wildfire Management

Funding Scheme IA – Innovation action

DELIVERABLE INFORMATION

Deliverable Number:	D5.3
Deliverable Name:	Demonstration of SILVANUS decision support system for response coordination 1 st version
Dissemination level:	PU
Type of Document:	DEM
Contractual date of delivery:	30/09/2023 (M20)
Date of submission:	
Deliverable Leader:	INTRA
Status:	Final
Version number:	Final – R2
WPLLeader/ TaskLeader:	DELL/INTRA
Keywords	Decision Support Systems, Data Fusion, Geographical Information Systems, Situational Awareness, Response Coordination
Abstract	This deliverable focuses on the demonstration of the tools developed for the SILVANUS decision support system for response coordination. The DSS tools are mainly the results of Task T5.4 and their implementation is based on integration of the SILVANUS firmware developed, the SILVANUS Knowledge Base and the real-time monitoring dashboard. This deliverable contains the description of the first version of the data toolkit while the final version will be reported in D5.5 planned to be submitted on M36.
Deliverable Leader:	INTRA
Lead Author(s)	INTRA, AMIKOM, UTH, DELL
Reviewers	ITTI, EDP, IST

Disclaimer

All information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose.

The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors’ view.

Document History			
Version	Date	Contributor(s)	Description
V0.1	12.06.2023	INTRA	ToC
V0.2	08.08.2023	AMIKOM	Sections 2, 4
V0.3	18.08.2023	UTH	Sections 5, 6
V1	23.08.2023	INTRA	Section 3 and integration of contributions
V1.1	15.09.2023	CTL, AMIKOM, UTH	Updated Section 2, 3, 5, 6 contributions
V2	18.09.2023	INTRA	Integration of updated contributions
Final	29.09.2023	INTRA	Integration of review comments
R1	02.08.2024	INTRA, KEMEA, VTG	Integration of revisions requested after the 2 nd project review (Table of contents replaced to include all subchapters, font sizes all set equal, terminology fixed to refer to wildfires only)

TABLE OF CONTENTS

TABLE OF CONTENTS	5
TABLE OF FIGURES.....	8
LIST OF ACRONYMS	11
EXECUTIVE SUMMARY.....	13
1. Introduction.....	14
2. Information fusion through the SILVANUS Knowledge Base	16
2.1. Concept of operation	16
2.2. Software implementation and results.....	16
2.2.1. Semantic Data Integration.....	16
2.2.2. Application Scenario.....	20
2.2.3. Rules design and implementation.....	22
2.3. Component integration	24
2.4. Plans for future extensions.....	24
3. Data fusion using Bayesian models and Monte-Carlo simulations	26
3.1. Concept of operation	26
3.1.1. Data Acquisition and Processing Method	26
3.1.1.1. Sources and acquisition method	26
3.1.1.2. Processing data into map of 14 variables.....	27
3.1.2. Concept and Mathematical Modelling: Fuzzy Logic, Montecarlo and Bayes Method	29
3.1.2.1. Fuzzy Logic	29
3.1.2.2. Basic Properties of Fuzzy Sets	29
3.1.2.3. Basic Operations on Fuzzy Sets.....	31
3.1.2.4. Monte Carlo Method	32
3.1.2.5. Bayesian Method	32
3.1.2.1. Case example.....	33
3.2. Software implementation and results.....	34
3.2.1. Data entry.....	34
3.2.2. Initialization and application execution	42
3.3. Component integration	50
3.4. Data Fusion Future Plan	52
4. Resource allocation of response teams	54
4.1. Concept of operation	54
4.1.1. Data Structures.....	54
4.2. Software implementation and results.....	55
4.2.1. Population based geographical distribution of impact risk estimation	55
4.2.2. Multi-objective optimization based Resource Allocation (MORA).....	57
4.2.2.1. Model Assumptions and Data Sources	57
4.2.2.2. Model formulation	58

4.2.2.3.	Model validation and demonstration.....	61
4.3.	Component integration	64
4.4.	Plans for future extensions.....	64
5.	Stakeholder notification decision on fire incidents using multilingual textual framework	66
5.1.	Concept of operation	66
5.2.	Software implementation and results.....	67
5.2.1.	Datasets	67
5.2.2.	Training Model	74
5.2.3.	Implementation Model	87
5.3.	Component integration	90
5.4.	Plans for future extensions.....	90
6.	Health Impact Component	91
6.1.	Concept of operation	91
6.2.	System implementation and results.....	92
6.2.1.	Air Quality Monitoring System	92
6.2.1.1.	Air Pollutants	92
6.2.1.2.	Sensors – Equiped by people.....	94
6.2.1.2.1.	Gas detection sensors.....	94
6.2.1.2.2.	Particles detection sensors	95
6.2.2.	Raspberry-Pi	96
6.2.3.	Sensors configuration.....	97
6.2.4.	Raspberry Pi implementation.....	98
6.2.5.	Complete air quality monitoring system.....	98
6.2.5.1.	Sensors connected to the Raspberry Pi.....	98
6.2.5.2.	Additional sensors and materials	98
6.2.5.3.	RESTful API.....	99
6.2.5.3.1.	RESTful API overview	99
6.2.5.3.2.	Deployed VM	100
6.2.5.3.3.	RESTful API endpoints.....	100
6.2.5.4.	Data visualization	106
6.3.	Sensors – Attached to vehicles or ground.....	107
6.3.1.	Smart Spot	107
6.3.2.	2. Smart Spot configuration.....	108
6.3.3.	Smart Spot technical dashboard	108
6.3.4.	Smart Spot MQTT Broker	109
6.4.	Component integration	112
6.5.	Plans for future extensions.....	112
7.	Evacuation route planning	113
7.1.	Concept of operation	113
7.2.	SW implementation and results.....	113

7.2.1.	Smoke dispersion	113
7.2.1.1.	Gaussian-plume models - Theoretical Background	114
7.2.1.2.	Implementation	117
7.2.2.	Evacuation route planning.....	119
7.2.2.1.	Risk assessment - Module triggering.....	119
7.2.2.2.	Module overview.....	120
7.2.2.2.1.	Module inputs.....	120
7.2.2.2.2.	Module operation	124
7.2.2.2.3.	Module outputs	130
7.2.2.3.	Simulation testing.....	130
7.2.2.3.1.	Experimental evaluation	130
7.2.2.3.2.	Output format.....	132
7.3.	Component integration	134
7.4.	Plans for future extensions.....	134
8.	Conclusions.....	135
9.	References.....	136

TABLE OF FIGURES

FIGURE 1: KBIF WITHIN T5.2	17
FIGURE 2: CASPAR ARCHITECTURE	18
FIGURE 3: ONTOTEXT'S GRAPHDB ENVIRONMENT FOR VISUALIZING RDF TRIPLES FOR CROATIAN PILOT DATA.....	20
FIGURE 4: MAPPING OF WILDFIRE INCIDENT LOCATION WITH CORRESPONDING INFORMATION TAB.....	21
FIGURE 5: GEOGRAPHICAL PLACEMENT OF HEALTH MONITORING DEVICES WITH ASSOCIATED INFORMATION TABS.....	21
FIGURE 6: PROXIMITY OF HEALTH MONITORING DEVICE TO FIRE INCIDENT LOCATION.....	22
FIGURE 7: RETRIEVE THE WILDFIRE INCIDENTS AND THE CORRESPONDING HEALTH MONITORING.....	23
FIGURE 8: WILDFIRE INCIDENTS AND THE CORRESPONDING HEALTH MONITORING	24
FIGURE 9: HIGH-LEVEL PROCESS OF DATA FUSION	28
FIGURE 10: SUPPORT, CORE AND HEIGHT OF A FUZZY SET.	30
FIGURE 11: ILLUSTRATION OF A-CUT CONCEPT	30
FIGURE 12: CONVEX AND NONCONVEX FUZZY SET.....	31
FIGURE 13: INTERSECTION OPERATION OF FUZZY SETS	31
FIGURE 14: UNION OPERATION OF FUZZY SETS.....	31
FIGURE 15: COMPLEMENT OF A FUZZY SET	32
FIGURE 16: MONTE CARLO FOR INTEGRAL AREA.....	32
FIGURE 17: SCHEME FOR CASE EXAMPLE OF HOW FUZZY LOGIC AND BAYESIAN METHOD WORK.	34
FIGURE 18: BUFFER TOOLS IN ARCGIS PRO.....	36
FIGURE 19: DISTANCE TO SETTLEMENT	36
FIGURE 20: DISTANCE TO ROAD	37
FIGURE 21: ELEVATION DATA.....	37
FIGURE 22: RASTER CALCULATOR TOOLS.....	38
FIGURE 23: FUEL LOAD.....	38
FIGURE 24: NDVI DATA	39
FIGURE 25: ASPECT DATA	40
FIGURE 26: ASPECT TOOLS IN ARCGIS.....	41
FIGURE 27: SLOPE DATA	41
FIGURE 28: SLOPE TOOLS IN ARCGIS PRO.....	42
FIGURE 29: RASTER TO POLYGON TOOLS.....	43
FIGURE 30: GENERATE SHAPEFILE GRID.....	43
FIGURE 31: OVERLAY ALL PARAMETERS TO ONE SHAPEFILE GRID	44
FIGURE 32: SPATIAL JOIN TOOLS.....	44
FIGURE 33: EXAMPLE CONVERT SHAPEFILE TO GEOJSON.....	45
FIGURE 34: DISTANCE TO SETTLEMENT	45
FIGURE 35: DISTANCE TO ROAD.....	45
FIGURE 36: POPULATION DENSITY.....	46
FIGURE 37: GDP	46
FIGURE 38: HISTORICAL FIRE	46
FIGURE 39: PRECIPITATION	46
FIGURE 40: TEMPERATURE.....	46
FIGURE 41: ELEVATION.....	46
FIGURE 42: FUEL LOAD.....	47
FIGURE 43: ASPECT	47
FIGURE 44: SLOPE	47
FIGURE 45: NDVI	47
FIGURE 46: LAND USAGE	47
FIGURE 47: VEGETATION TYPE.....	47
FIGURE 48: USER INTERFACE FOR PARAMETER INITIALIZATION AND APPLICATION EXECUTION.....	48
FIGURE 49: FIRE PROBABILITY	49
FIGURE 50: PRIORITY RESOURCE	49

FIGURE 51: DETAILED DATA OF A VARIABLE	50
FIGURE 52: OVER TIME DATA	50
FIGURE 53: EXTENDED PROCESS FOR DATA FUSION	52
FIGURE 54 - RESOURCE ALLOCATION DIAGRAM	54
FIGURE 55 - RESOURCE ALLOCATION INPUT RASTER	55
FIGURE 56 - PoC VALIDATION SCENARIO	56
FIGURE 57 - PoC EXAMPLE INPUTS	56
FIGURE 58 - PoC EXAMPLE OUTPUT (PER CELL CRITICALITY INDEX)	57
FIGURE 59 – MORA VALIDATION SCENARIO (A) AREA UNDER EXAMINATION (B) FIRE SPREAD MODEL (C) POPULATION DISTRIBUTION.....	62
FIGURE 60 – CELL PARAMETERS FOR THE AREA UNDER EVALUATION: (A) CELL INDEXES (B) FIRE ARRIVAL TIME PER CELL (IN MULTIPLES OF THE BASIC PERIOD) (C) POPULATION DENSITIES	62
FIGURE 61 - RESOURCE TYPES EXAMPLE	62
FIGURE 62 - RESOURCE ALLOCATION OPTIMAL SOLUTION.....	63
FIGURE 63 – INPUT AND OUTPUT OF RESOURCE ALLOCATION DSS.....	64
FIGURE 64: FRAMEWORK OF SOCIAL MEDIA SENSING	67
FIGURE 65: DATA DISTRIBUTION EACH CLASS	68
FIGURE 66: WORD LENGTH DISTRIBUTION.....	69
FIGURE 67: NER ENGLISH DATASET PREVIEW.....	70
FIGURE 68: DATA DISTRIBUTION EACH CLASS	71
FIGURE 69: WORD LENGTH DISTRIBUTION.....	72
FIGURE 70: DATA DISTRIBUTION EACH CLASS	73
FIGURE 71: WORD LENGTH DISTRIBUTION.....	74
FIGURE 72: LANGUAGE DATASETS DISTRIBUTION	74
FIGURE 73: MEMORY CELL OF LSTM	77
FIGURE 74: MEMORY CELL OF GRU	78
FIGURE 75: BIDIRECTIONAL RNNs STRUCTURE.....	78
FIGURE 76: BIDIRECTIONAL LSTM CLASSIFICATION REPORT	80
FIGURE 77: KERAS MODEL SUMMARY OF SPANISH FIRE PREDICTION	81
FIGURE 78: LOSS AND ACCURACY OF SPANISH FIRE PREDICTION MODEL TRAINING.....	81
FIGURE 79: SPANISH FIRE PREDICTION CLASSIFICATION REPORT	81
FIGURE 80: KERAS MODEL SUMMARY OF ITALIAN FIRE PREDICTION	82
FIGURE 81: LOSS AND ACCURACY OF ITALIAN FIRE PREDICTION MODEL TRAINING.....	83
FIGURE 82: ITALIAN FIRE PREDICTION CLASSIFICATION REPORT.....	83
FIGURE 83: BiLSTM AND CRF FOR LOCATION EXTRACTION.....	84
FIGURE 84: ACCURACY AND LOSS TRAINING.....	85
FIGURE 85: CLASSIFICATION REPORT OF NER USING BiLSTM AND CRF	85
FIGURE 86: ENGLISH LOCATION DETECTION ARCHITECTURE.....	86
FIGURE 87: CLASSIFICATION REPORT FOR ENGLISH NER MODEL	86
FIGURE 88: API CLASSIFICATION FOR FIRE	87
FIGURE 89: ENTITY RECOGNITION API.....	88
FIGURE 90: DIAGRAM CONCEPT OF SOCIAL MEDIA SENSING	89
FIGURE 91: HEALTH IMPACT COMPONENT – USE CASE	92
FIGURE 92: TABLE OF EUROPEAN AIR QUALITY INDEX (BASED ON POLLUTANT CONCENTRATIONS IN $\mu\text{G}/\text{M}^3$).....	93
FIGURE 93: GAS SENSORS SPECIFICATIONS.....	95
FIGURE 94: GAS SENSOR COMPONENTS	95
FIGURE 95: GAS DETECTION SENSOR.....	95
FIGURE 96: (A) BACK VIEW OF THE LASER SENSOR, (B) FRONT VIEW OF THE ADAPTOR, (C) BACK VIEW OF THE ADAPTOR, (D) OVERALL COMPONENTS	96
FIGURE 97: GPIO PINS	97
FIGURE 98: AIR QUALITY OBSERVATION SYSTEM.....	98
FIGURE 99: INTERIOR SYSTEM AND COMPONENTS.....	99
FIGURE 100: VM CHARACTERISTICS AND INTEGRATED COMPONENTS	100

FIGURE 101: /INSERT ENDPOINT SENT JSON FILE	102
FIGURE 102: /GET-LATEST-DATA ENDPOINT RETRIEVED JSON.....	103
FIGURE 103: /AQI ENDPOINT RETRIEVED JSON FILE	103
FIGURE 104: DATA.JSON FILE.....	105
FIGURE 105: META-DATA.JSON FILE.....	106
FIGURE 106: VISUALIZATION WEBPAGE	106
FIGURE 107: SMART SPOT - IOT DEVICE FOR ENVIRONMENTAL PARAMETER SENSING	108
FIGURE 108: VISUALIZATION DASHBOARD (O3)	109
FIGURE 109: VISUALIZATION DASHBOARD (PM2.5)	109
FIGURE 110: BROKER DEPLOYMENT AND CONFIGURATION	110
FIGURE 111: MQTT EXPLORER CLIENT	111
FIGURE 112: JSON FILE	111
FIGURE 113: PYTHON'S PAHO PACKAGE	112
FIGURE 114: SMOKE DISPERSION MODELING PROCESS (GOOD PRACTICE GUIDE, 2004)	114
FIGURE 115: GAUSSIAN AIR POLLUTANT DISPERSION PLUME AND COORDINATE SYSTEM (FROM EN.WIKIPEDIA.ORG)	115
FIGURE 116: PLUME RISE (FROM HTTP://WWW-PERSONAL.UMICH.EDU/~WEBERG/EFF_STACK_HEIGHT.HTM)	115
FIGURE 117: CURVES OF Σ_y AND Σ_z FOR STABILITY CLASSES AS A FUNCTION OF DISTANCE FROM THE SOURCE (VARMA, M. S. A. K. 2014).	117
FIGURE 118: DIFFERING WIND FLUCTUATIONS	118
FIGURE 119: MULTIPLE WILDFIRE SOURCES.....	118
FIGURE 120: VERTICAL STABILITY EFFECTS.....	119
FIGURE 121: EVACUATION ROUTE PLANNING MODULE TRIGGERING	120
FIGURE 122: EVACUATION ROUTE PLANNING INTERACTIONS.....	120
FIGURE 123: GEOJSON OUTPUT EXAMPLE WITH CORRESPONDING MAP	122
FIGURE 124: POST REQUEST TO /EVACUATIONS/SMOKE-FIRE ENDPOINT (2 SOURCES).....	123
FIGURE 125: GEOJSON RESPONSE FROM SIMPLE DIRECTIONS ENDPOINT CALL	126
FIGURE 126: GEOJSON RESPONSE FROM CUSTOMIZED PARAMETERS.....	128
FIGURE 127: CURL CALL TO /EVACUATIONS/CITIES ENDPOINT	129
FIGURE 128: CURL CALL TO /EVACUATIONS/ROUTES ENDPOINT.....	129
FIGURE 129: CITIES AND AVAILABLE ROUTES IN NORTHERN EUBOEA	130
FIGURE 130: FIRE SPREAD AND SMOKE DISPERSION PROGRESSION (EXP.1).....	131
FIGURE 131: ROUTES CHARACTERIZATION (EXP.1)	131
FIGURE 132: EXP.2 SETTING	132
FIGURE 133: ROUTES CHARACTERIZATION(EXP.2).....	132
FIGURE 134: ROUTES COMPARISON.....	133
FIGURE 135: GEOJSON SAMPLE OF A ROUTE	134

LIST OF ACRONYMS

Terminology/Acronym	Description
AQI	Air Quality Index
COP	Constrained Optimization Problem
CPU	Central Processing Unit
CRF	Conditional Random Forest
DEM	Digital Elevation Model
DSS	Decision Support System
EVI	Enhanced Vegetation Index
GDP	Gross Domestic Product
GIS	Geographical Information System
GPS	Global Positioning System
GPU	General Processing Unit
GRU	Gated Recurrent Units
HRF	Human-Related Factors
I2C	Inter-Integrated Circuit
IoT	Internet of Things
KB	Knowledge Base
KBIF	Knowledge Base Information Fusion
LSTM	Long Short-Term Memory
MORA	Multi-Objective based Resource Allocation
NDVI	Normalized Difference Vegetation Index
NER	Named Entity Recognition
NIR	Near Infra-Red
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
OWL	Web Ontology Language
PEF	Physical-Environmental Factors
PoC	Proof of Concept
RDF	Resource Description Framework
RNN	Recurrent Neural Network
ROS	Random Oversampling

Terminology/Acronym	Description
SAL	Storage Abstraction Layer
SAVI	Soil Adjusted Vegetation Index
SMOTE	Synthetic Minority Oversampling Technique
SoC	System on Chip
SPI	Serial to Parallel Interface
UART	Universal Asynchronous Receiver/Transmitter
UI	User Interface
USGS	United States Geological Survey

EXECUTIVE SUMMARY

This is the third deliverable of WP5 focusing on the demonstration of the tools developed for the SILVANUS decision support system for response coordination. While the tools described in this deliverable are mainly the results of activities performed in the context of Task T5.4 - *Data toolkit for decision support system*, the tool implementation is based on integration of the SILVANUS firmware developed in the context of Task T5.1 *Big-data Analytics framework for situational awareness on fire danger index*, the SILVANUS Knowledge Based developed in the context of Task T5.2 *Semantic framework for information fusion* and the real-time monitoring dashboard developed in the context of Task T5.3 *Real-time monitoring of wildfire behaviour in temporal space for response coordination*. This deliverable contains the description of the first version of the data toolkit while the final version will be reported in D5.5 planned to be submitted on M36. Six main tools have been developed in the first phase of the project implementing:

- information fusion through the SILVANUS Knowledge Base,
- data fusion using Bayesian models and Monte-Carlo simulations to model the probability and intensity of wildfires,
- optimized resource allocation of response teams,
- stakeholder notification on wildfire incidents using a multilingual textual framework
- health impact assessment, and
- evacuation route planning in case of wildfire events

The above tools are expected to assist commanders by improving situational awareness and producing information related to projections about the evolution of wildfires and potential optimal decisions to react in order to reduce the potential impact mainly focusing on the potential extension of humans to risks of life and property losses. While the type of the deliverable is Demonstrator, this report includes both details about the software implementation of the tools as well as the methodological approach and scientific background related to the algorithm development and the tools design.

1. Introduction

The objective of task T5.4 of WP5 is to develop analytic algorithms, which can exploit the data made available by SILVANUS sensors deployed across the forest as well as data made available through geographic information systems (GIS) and other external data sources to develop algorithms and methodologies that effectively combine disparate knowledge resources into a single framework leveraging upon the information fusion algorithms reported in the literature including (i) data association, (ii) state estimation, and (iii) decision fusion. In addition, the task also focuses on the resource allocation of response teams in the field depending on the evolution of the incident and the current status of the available response teams performing an optimization process for the coverage of the area under consideration and the use of the available resources upon the data collected by the SILVANUS monitoring components. Finally, decision making deals with the prediction of the affected areas and the modelling of the surroundings in order to be able to deliver evacuation paths for the affected people, which will have functional synergies with some of the outcomes in T5.3. The map of the area will be the basis for detecting the appropriate routes for evacuation if necessary. The algorithms developed in T5.4 take into consideration population and underlying geospatial data in order to avoid proposing evacuation paths that will be overcrowded and minimization of population exposure to health risks.

The remainder of this document is structured as follows:

Section 2 describes the main tool of SILVANUS to process the data received from disparate sensors and produce and manage the knowledge developed within SILVANUS to be shared and exploited for response coordination to control the spread of wildfires. Thus, this tool is an enabler to incorporate human expertise into automated frameworks. Specifically, this section describes how the SILVANUS Knowledge Base Information Fusion (KBIF) can support the promotion of situational awareness and the methodological analysis of results delivered by various project components.

Section 3 describes data fusion using Bayesian models and Monte-Carlo simulations to model the probability and intensity of wildfires. Specifically, it describes models of interaction of multiple factors that contribute to the creation of conditions that increase the probability of ignition and intensity of wildfires and an application that can produce visual results as outcome of these models to improve situational awareness of decision makers.

Section 4 describes algorithms to optimize resource allocation of response teams based on the projection of fire-spread that is part of the SILVANUS wildfire management platform. The geospatial distribution of population and other GIS-related land properties as well as the distribution of firefighting units are taken into account and a multi-objective optimization problem can be formulated and solved in order to lead to cost minimization in terms of the minimum total risk of fire impact given the available resources to plan for the mitigation of this impact.

Section 5 describes a tool to assist in stakeholder notification on wildfire incidents using a multilingual textual framework. This tool can assist in early and effective notification of citizens exposed to wildfire events.

Section 6 describes a methodology to model health impact assessment. This tool is important to model the geospatial distribution of dangerous areas during wildfire events and take proper actions to mitigate risk of exposing citizens to potential health threats.

Section 7 describes a complementary tool to the one described in section 6 in order to plan evacuation routes in case of wildfire. Following the same objective i.e. to mitigate risk of exposing citizens to potential health threats, routes on a map can be appropriately planned

to avoid areas under direct fire threat of exposure to health threats based on the temporal conditions during wildfires.

Finally, Section 8 presents the main conclusion of this deliverable.

2. Information fusion through the SILVANUS Knowledge Base

2.1. Concept of operation

A key element of the SILVANUS project is the Knowledge Base Information Fusion (KBIF) from CTL, which leverages rule-based systems to incorporate human expertise into automated frameworks. The SILVANUS KBIF functions with the overarching goals of promoting situational awareness and methodically analyzing results from various project components. Extraction of subtle insights and cultivation of a common lexicon for depicting actions, scenarios, and warnings are the main objectives driving the development of the KBIF. The KBIF serves a crucial role in enabling entities including government agencies, fire and emergency services, environmental agencies, and health departments to make prompt and informed choices by unifying tasks.

By integrating project components that are supported by a common vocabulary derived from the Silvanus ontology (D3.1) and include significant data, this integration provides a uniform repository of knowledge for those components. This section offers an introductory description of the Knowledge Base (KB), integrated as a crucial element of the Semantic framework for information fusion within WP5. Its primary purpose is to establish a singular knowledge repository that unifies project components, affording each component a shared vocabulary. The overarching aim is to construct a comprehensive and scalable semantic framework for information fusion, designed to produce a Resource Description Framework¹ (RDF) -based cohesive semantic KB.

The aforementioned KB serves as a foundation for decision support and drawing insights from an array of diverse/heterogeneous sources. It is important to highlight that while this deliverable provides a detailed view of this decision-making aspect, the comprehensive overview of the entire KB concept, spanning from start to finish, will be presented in the upcoming D5.4 "Semantic Information Fusion Framework" (anticipated by M36).

2.2. Software implementation and results

2.2.1. Semantic Data Integration

Semantic Data Integration (also referred to as "semantic data fusion" or "ontology population") involves the process of infusing the initially vacant semantic model with instance data sourced from sensors (i.e., raw data) and other SILVANUS analytical components (i.e., higher-level data). This section provides an overview of the semantic data integration process embraced within SILVANUS. A comprehensive exploration of this endeavor will be offered in forthcoming project deliverables, specifically in D5.4, as previously mentioned.

The workflow, encompassing the KBIF as an integral component of the T5.2 Semantic framework for information fusion, is visually presented in Figure 1.

¹ <https://www.w3.org/RDF/>

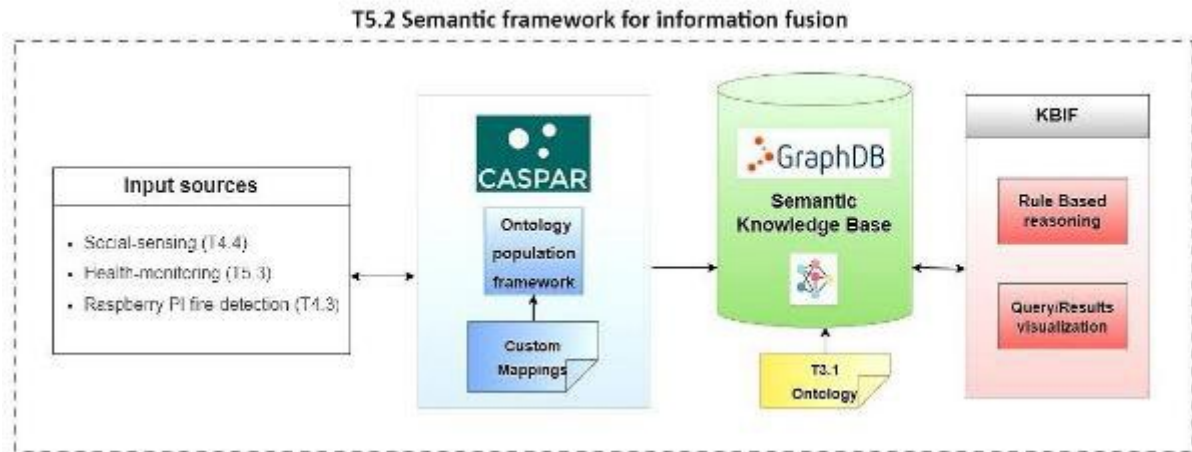


Figure 1: KBIF within T5.2

The SILVANUS ontology and its core concepts were comprehensively outlined in Deliverable 3.1. This semantic framework was developed within the stable Protégé editor environment and encoded using OWL². During the entire development period, significant efforts were devoted to incorporating a wide spectrum of forest and wildfire-related knowledge and seamlessly integrating it into a rule-based structure. Through the use of SPARQL³, a dynamic query language designed to glean insights from data stored in the RDF format, these principles are carefully stated. The power of SPARQL rests in its capacity to specify complicated patterns that line up with designated data associations woven into the underlying structure of RDF graphs.

CASPAR⁴ (Structured Data Semantic Exploitation Framework), CTL's flexible domain-agnostic semantic data integration framework, handles the ingestion of inputs into the SILVANUS Semantic KB. CASPAR has been expanded within SILVANUS to accommodate the project's specific input needs, such as ingesting data from social media sensing data sources (T4.4) into the KB, among other data sources. The overall architecture of CASPAR is outlined in Figure 2. The tool uses a series of connected methods as shown to ingest data into a semantic model:

- Automated Acquisition of Structured Data
- Mapping of Input Data Fields
- Semantic Knowledge Integration
- Enrichment from Linked Open Data Sources
- Rule-Based Semantic Reasoning

² <https://www.w3.org/OWL/>

³ <https://www.w3.org/TR/rdf-sparql-query/>

⁴ <https://caspar.catalink.eu/>

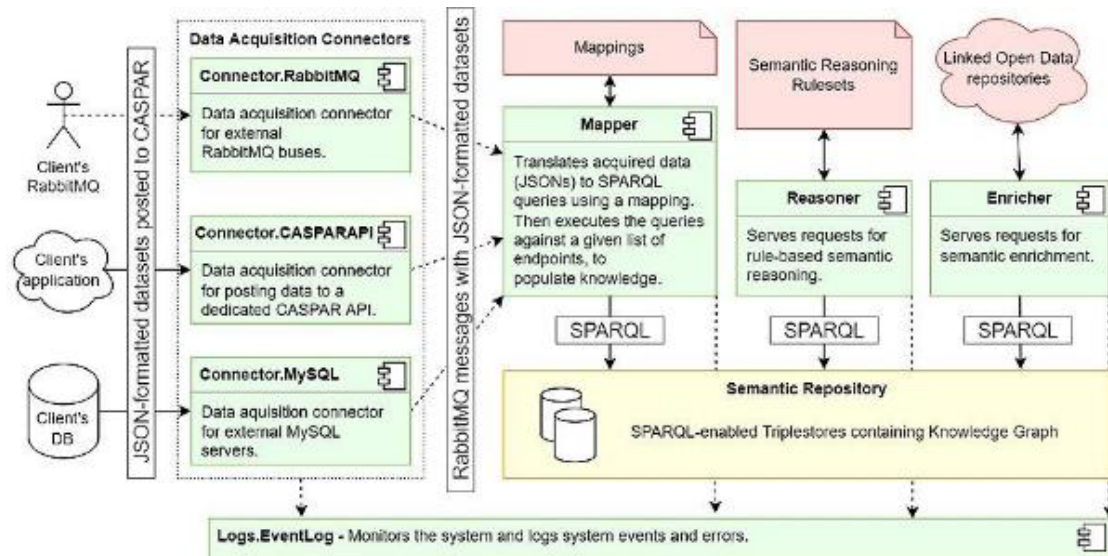


Figure 2: CASPAR architecture

In the context of SILVANUS, a standardized JSON structure has been collaboratively developed and agreed upon with each contributing partner. This structure acts as a container to encapsulate the distinctive data inputs provided by each partner. These inputs encompass a wide array of observations spanning domains such as social media insights, wildfire/smoke detection from IoT edge devices, and air quality assessments. This structured data is seamlessly funneled into CASPAR for automated integration into the KB, wherein they manifest as discrete individuals or objects. Below is an illustrative example of this generic JSON blueprint, specifically pertaining to CTL's IoT edge device (T4.3):

```
{
  "uuid": "01879875-7508-7b37-abc5-6f8df91c8ccc",
  "sensor_type": [
    "humidity",
    "temperature",
    "camera"
  ],
  "range": {
    "value": 5,
    "unit": "meters"
  },
  "timestamp": "2023-03-13T07:41:39.463138Z",
  "location": [
    {
      "placename": "test-pilot",
      "geometry": {
        "type": "Point",
        "coordinates": [
          {
            "lat": 35.1765597,
            "lon": 33.3829851
          }
        ]
      }
    }
  ]
},
  "sensory_data": {
    "temperature": {
      "value": 13.4,
      "unit": "celsius"
    },
    "humidity": {
      "value": 55.5,

```

```

        "unit": "percentage"
    },
    "gas_sensor": {
        "smoke": 43.8,
        "liquid_petroleum_gas_lpg": 10.24,
        "methane": 23.12,
        "hydrogen": 23.8,
        "unit": "parts_per_million(ppm)"
    }
},
"visual_data": {
    "fire_detection": {
        "contains_fire": false,
        "fire_score": 0.0229,
        "fire_probability": "none"
    },
    "smoke_detection": {
        "contains_smoke": true,
        "smoke_score": 0.9546,
        "smoke_probability": "high"
    },
    "image":
"data:image/png;base64,iVBORw0KGgoAAAKhQQLkGxTT1wAAAAASUVORK5CYII="
}
}

```

In this example JSON, the wildfire probability is denoted as "none" (fire_probability: "none"), indicating no fire was detected, and the smoke presence is indicated as "true" (contains_smoke: true), with the smoke probability marked as "high" (smoke_probability: "high"), meaning there is a high probability the IoT device has detected a smoke event. For a more detailed explanation of each field, you can refer to D4.1.

At the current stage, CTL reached an important milestone by developing the mechanism to convert JSON data into RDF triples. As a result of this achievement, KB can effectively be populated with data from three components: CTL's IoT edge device (Task 4.3), CERTH's social media sensing (T4.4), and UTH's health monitoring (T5.3).

Additionally, to the (dummy) data utilized for testing, we have successfully incorporated, in the KB (actual) data from the Croatian pilot site (collected from the IoT edge device). Figure 3, depicts the RDF triples for the case of the collected data, using Ontotext's GraphDB⁵ environment.

⁵ <https://www.ontotext.com/products/graphdb/>

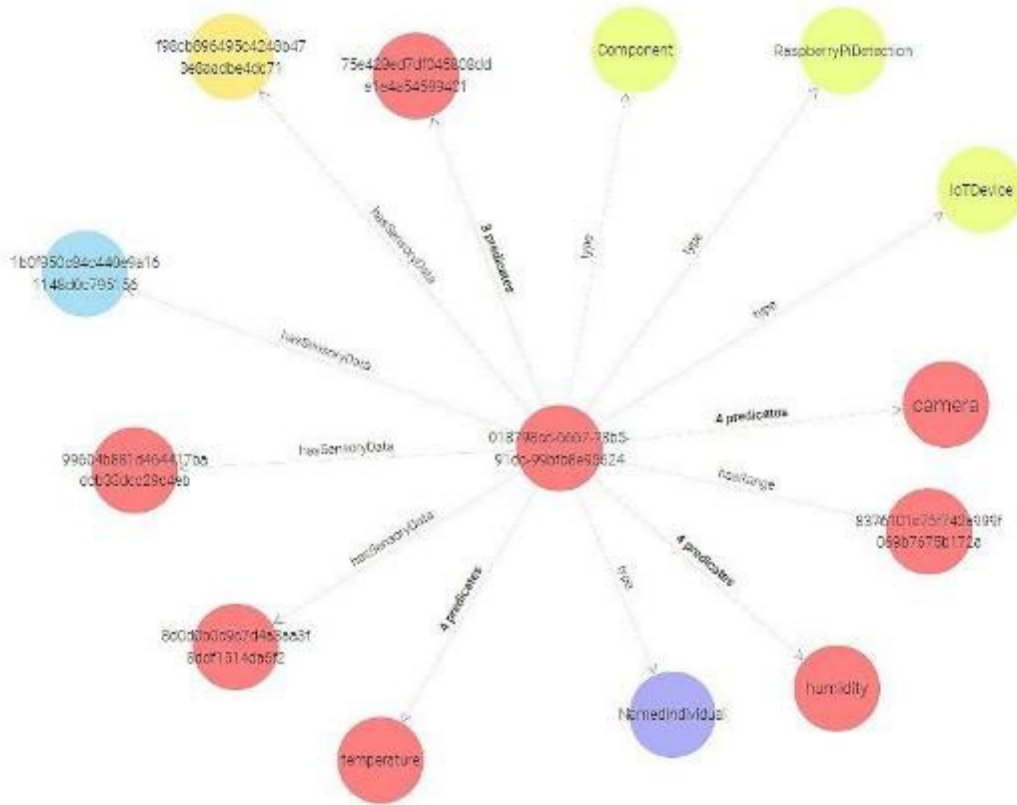


Figure 3: Ontotext's GraphDB environment for visualizing RDF triples for Croatian pilot data.

2.2.2. Application Scenario

This section presents a simple but real-world application scenario that demonstrates how effectively a User Interface (UI)—in this case, a map—integrated with the KB could possibly be used. This demonstration aims to underscore the practicality of this integration. The scenario revolves around a stakeholder concerned about wildfire incidents and fluctuations in air quality within proximity. By means of the UI, the agency can monitor ongoing wildfire occurrences (Figure 4). Subsequently, the KB engages in querying and retrieving specific metrics related to these events, encompassing essential measurements.

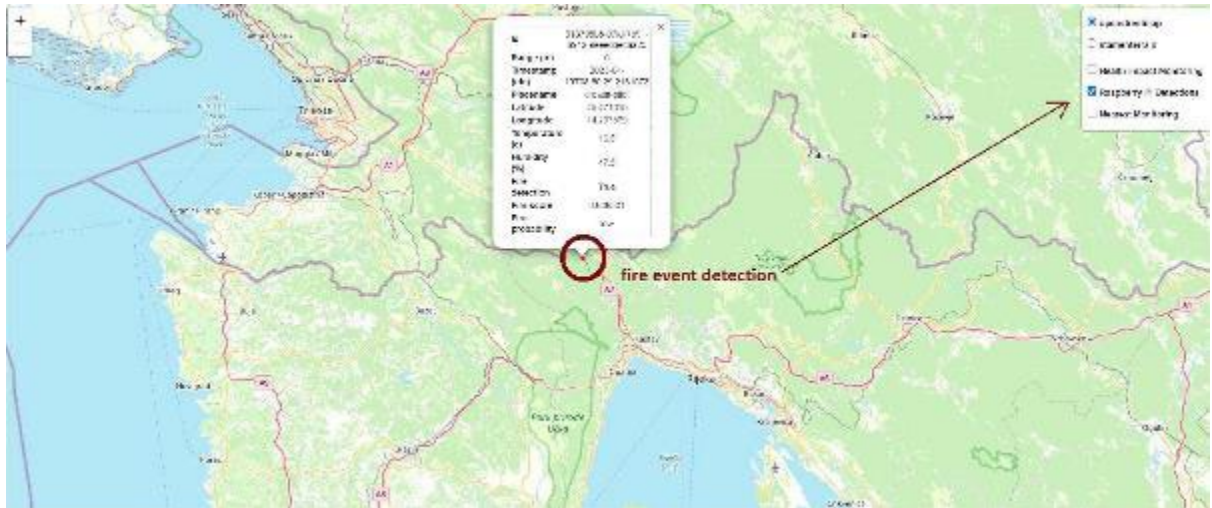


Figure 4: Mapping of wildfire Incident location with corresponding information tab

Through the user interface (UI), the agency gains the capability to effortlessly oversee ongoing wildfire incidents (data received from IoT edge devices). Additionally, the UI allows for the convenient identification of health monitoring devices (i.e., carried by first responders) within proximity. By accessing the dedicated tab, the agency can access data concerning air quality measurements for the area of interest (refer to Figure 5) and take any necessary precautions/actions, for example extracting first responders from an area because of toxic fumes.

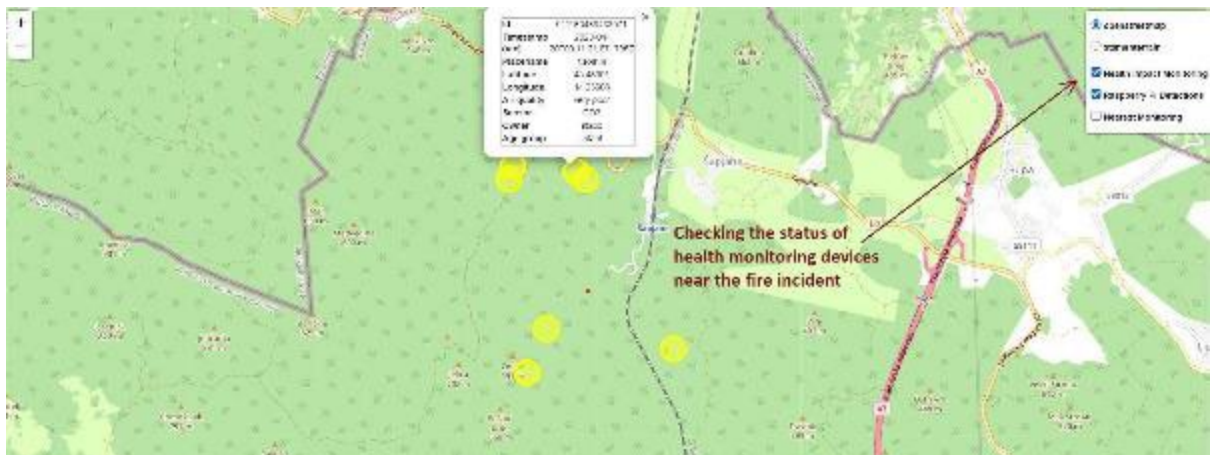


Figure 5: Geographical placement of Health Monitoring devices with associated information Tabs

Moreover, the UI enables the agency to verify the closest health monitoring device in relation to a wildfire incident (Figure 6), which in turn can help them to identify which responder will arrive first at the area of interest or who is in more imminent danger.



Figure 6: Proximity of Health Monitoring device to Fire Incident location

As demonstrated in this simple example, the fusion of information from various components –accompanied with the rule designing– can assist agencies make more strategic decisions during critical situations. It is worth mentioning that the aforementioned scenario focused on Phase B (Detection and Response) but given the variety of data types available in the project and the consortium’s expertise (needed for the rule design), the KB can be used throughout all phases.

Lastly, we would like to clarify that the UI presented above was created to give a possible visualization of the KBIF output. Still, as the development of the component is in progress and yet to be integrated in the SILVANUS Dashboard the final product (UI) may vary.

2.2.3. Rules design and implementation

The development of the SILVANUS KBIF has placed a strong emphasis on the rigorous application of ontology engineering approaches in conjunction with the efficient use of SPARQL. These sophisticated methods have been integrated into Ontotext's GraphDB environment, serving as foundational pillars for the system's functionality. Figure 7, displays a sample SPARQL query, emphasizing the actual implementation of these strategies.

```

PREFIX : <https://silvanus-project.eu/wp-content/uploads/2022/09/SILVANUS-Ontology.zip#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?HealthMonitoringDatetime ?AirQuality ?IoTDatetime ?FireProbability ?FireDetection ?Placename
WHERE {
  ?health rdf:type :HealthImpactMonitoring;
    :hasGeoLocation ?geoLocation1;
    :hasTimestamp ?HealthMonitoringDatetime;
    :hasAirQualityIndex/rdfs:label ?AirQuality.
  ?geoLocation1 :hasPlacename ?Placename.
  ?rasp rdf:type :RaspberryPiDetection;
    :hasGeoLocation ?geoLocation2;
    :hasTimestamp ?IoTDatetime;
    :hasSensoryData ?sensoryData.
  ?geoLocation2 :hasPlacename ?placename2.
  ?sensoryData :hasFireDetection ?FireDetection;
    :hasFireProbabilityIndex/rdfs:label ?FireProbability.
  FILTER (CONTAINS(UCASE(STR(?Placename)), "CROATIA"))
  FILTER (CONTAINS(UCASE(STR(?placename2)), "CROATIA"))
  FILTER (?HealthMonitoringDatetime > "2022-01-01T00:00:00Z"^^xsd:dateTime && ?HealthMonitoringDatetime
< "2024-01-01T00:00:00Z"^^xsd:dateTime)
  FILTER (?IoTDatetime > "2022-01-01T00:00:00Z"^^xsd:dateTime && ?IoTDatetime < "2024-01-
01T00:00:00Z"^^xsd:dateTime)
  FILTER (?AirQuality = "very poor")
  FILTER (?FireDetection = true)}

```

Figure 7: Retrieve the wildfire incidents and the corresponding health monitoring

For gaining insights portrayed in the maps above, this query is an invaluable tool. These conclusions are based on the data we previously discussed, which was collected by CTL's IoT edge device (T4.3) in the context of the Croatian pilot – that is comprehensively elaborated in deliverable D4.1. Additionally, by integrating data, this SPARQL query demonstrates the potential of synergy. It expertly combines CTL's insights with UTH's data from their IoT device within T5.3, focusing on air quality. It is important to note that UTH's data, utilized for fusion purposes, is not sourced from the Croatian pilot but is instead dummy data specifically generated for integration. The end result, highlighted by the strategic alignment of SPARQL queries that harmonize both data sources, is a holistic perspective that enhances decision-making capacity (Figure 8).

HealthMonitoringDatetime	AirQuality	IoTDatetime	FireProbability	FireDetection	Placename
Apr 20, 2023, 3:11:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 20, 2023, 2:41:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 21, 2023, 1:03:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 20, 2023, 1:51:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 20, 2023, 4:00:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 19, 2023, 4:00:21 PM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella
Apr 21, 2023, 5:43:21 AM	very poor	Apr 19, 2023, 11:50:28 AM	low	true	Croella

Figure 8: Wildfire incidents and the corresponding health monitoring

2.3. Component integration

In the realm of application integration, the Storage Abstraction Layer (SAL) can incorporate our system into its operations. This integration is facilitated by utilizing RabbitMQ⁶ to consume data pushed by other components to the SAL. Following this, the system undertakes a transformation process before populating the semantic KB with the necessary data for the population procedure, as mentioned in the previous sections.

In terms of deployment, our approach involves the utilization of a Virtual Machine running Ontotext's GraphDB⁷. While comprehensive details regarding these aspects will be presented in the forthcoming D5.4, additional in-depth documentation about the deployment process can be accessed on the Silvanus GitHub repository: <https://github.com/silvanus-prj/semantic-knowledge-base>.

2.4. Plans for future extensions

Anticipating future developments, our project outlines several key directions for progression. One significant focus involves expanding collaborative partnerships to incorporate additional metadata into the KB (e.g. CEMA Mobile App from UISAV). These discussions are already underway, and forthcoming updates will be detailed in D5.4.

Additionally, our roadmap includes the improvement and expansion of our ruleset as well as the introduction of additional SPARQL queries. The objective of these enhancements is to bolster the Knowledge Base's (KB) analytical capabilities, providing potential stakeholders with a richer and more diverse pool of fused data at their disposal.

Under the guidance of ITTI, the next step involves incorporating these cutting-edge insights directly into the SILVANUS Dashboard. As a visual depiction of the project's development, this interface will enable stakeholders to interact with the enhanced functionality, improved rules,

⁶ <https://www.rabbitmq.com/>

⁷ <https://www.ontotext.com/products/graphdb/>

and complicated query responses. We expect to unlock greater decision-making capability and enable a more thorough project experience through these focused efforts.

3. Data fusion using Bayesian models and Monte-Carlo simulations

3.1. Concept of operation

Wildfires can be disastrous with significant impacts and high cost. Additionally, forests extend across large volume of areas which are often considered remote. These conditions need complex decisions related to resources management prior as well as during wildfire events that are the target of this Data Fusion application. The objective for developing this application is providing support to decision makers and stakeholders in allocating the resources. Two main measurements that characterize the location regarding wildfire risk management are: Fire Risk Probability and Priority Level Resources Allocation. Fire Risk Probability depend on various variables that characterizing the locations, while the Priority Level Resources Allocation takes into account the area of the forest to order the urgency level. This application will be helpful for preventive action or to minimize the cost if a disaster happens.

3.1.1. Data Acquisition and Processing Method

3.1.1.1. Sources and acquisition method

Wildfire risk is influenced by a combination of anthropogenic (human-related) and physical factors (Kolanek et al., 2021; Ma et al., 2020; Marlon et al., 2008). The interaction of these factors contributes to the creation of conditions that increase the probability and intensity of wildfires. Anthropogenic factors considered in this study include distance to settlement, distance to road, population density, historical wildfire-spot density, and Gross Domestic Product (GDP). The physical factors analyzed in this study consist of land use classification as well as general land/forest structure related data including Normalized Difference Vegetation Index (NDVI), fuel load, elevation, slope, aspect, annual precipitation, and temperature. A detailed description of the data acquisition sources is described in Table 1.

Table 1 - Data acquisition source

No	Variable	Data Sources	Processing
1.	Distance to Settlement (m)	Landsat 8	Buffer analysis with certain distance to settlement
2	Distance to Road (m)	OpenStreetMap	Buffer analysis with certain distance to road
3	Elevation (m)	ASTER GDEM imagery	Classification of elevation based on certain interval
4	Fuel load (ton/km ²)	Landsat 8 imagery	Fuel load estimation using vegetation indices
5	Historical wildfire (events/0.75km ²)	Local government	Recorded data from local disaster management authority
6	Land Usage	Landsat 8 imagery	Supervised classification from imagery
7	NDVI	Landsat 8 imagery	Calculation using infrared and near-infrared band
8	Population Density (people/km ²)	World Population Dataset (www.worldpop.org)	Classification of population density based on certain interval
9	GDP (21million idr/km ²)	World Bank Database (data.worldbank.org)	Classification of GDP based on certain interval
10	Vegetation Type	Landsat 8 imagery	Supervised classification from imagery
11	Aspect (°)	ASTER GDEM imagery	Aspect calculation from elevation data
12	Slope (°)	ASTER GDEM imagery	Aspect calculation from elevation data
13	Temperature (°C)	ERA5 Reanalysis	Average of annual temperature
14	Precipitation (mm/y)	ERA5 Reanalysis	Total annual rainfall

3.1.1.2. Processing data into map of 14 variables

Figure 9 shows the data source acquisition scheme and processing into GeoJSON format that was used in the present work. Satellite and Light Detection and Ranging sources are utilized by Landsat and digital elevation (DEM) data and were completed by field observations data. The 14 variables utilized in this study were sourced from various open data repositories. To determine the distance to roads, OpenStreetMap data was employed. The buffer method was applied to calculate the DR (distance to the road) and distance to settlements (DS) within a specific area. Buffer analysis investigates the spatial relationship between a central object and other objects within a designated distance (Zhou et al., 2018).

The primary data source for this study is Landsat 8 imagery, which serves as the foundation for generating a range of derived data, such as distance to human settlements, Normalized Difference Vegetation Index (NDVI), fuel load, land usage, and vegetation type. NDVI is the ratio of the difference between the near-infrared band (NIR) and the red band (R) and the sum of these two bands (Yengoh et al., 2016) (Eq. 3.1). Fuel load estimation is calculated based on vegetation indices including EVI (Enhanced Vegetation Index), SAVI (Soil Adjusted Vegetation Index) and NDVI (Bao et al., 2022) (Eq. 3.2).

$$NDVI = (NIR-RED)/(NIR+RED) \quad (3.1)$$

Where, NIR stands for Near-Infrared Light and RED stands for Visible Red Light.

$$FUEL\ LOAD = 50.78EVI + 237.64NDVI + 109.38SAVI - 102.83 \quad (3.2)$$

Where, EVI stands for Enhanced Vegetation Index, SAVI stands for Soil Adjusted Vegetation Index and NDVI stands for Normalized Difference Vegetation Index. Settlements, land usage and vegetation type were classified using maximum likelihood method. Maximum Likelihood method has been extensively used as the remote sensing classification method (Yonezawa, 2007). This approach assumes that the digital number (DN) values of the image within each user-defined class adhere to a multivariate normal probability distribution. To calculate the distance to a settlement, the classification of areas into settlement and non-settlement categories is performed. This involves calculating the distance from a specific location to the settlement area. The classification of land usage and vegetation type were described in Table 1 and Table 2.

Table 2 - Classification of land usage

<i>Level</i>	<i>Description</i>
1	Paddy field, Dry land, Water area, Unused land, Urban-rural fringe, Industrial and mining land, Residential land
2	Meadow, Grassland, Alpine vegetation
3	Broad-leaved forest, Shrub

4	Coniferous forest, Theropencedrymion
---	--------------------------------------

Table 3 - Classification of vegetation types

<i>Level</i>	<i>Description</i>
1	Desert, Swamp, Cultivated plants
2	Meadow, Grassland, Alpine vegetation
3	Broad-leaved forest, Shrub
4	Coniferous forest, Theropencedrymion

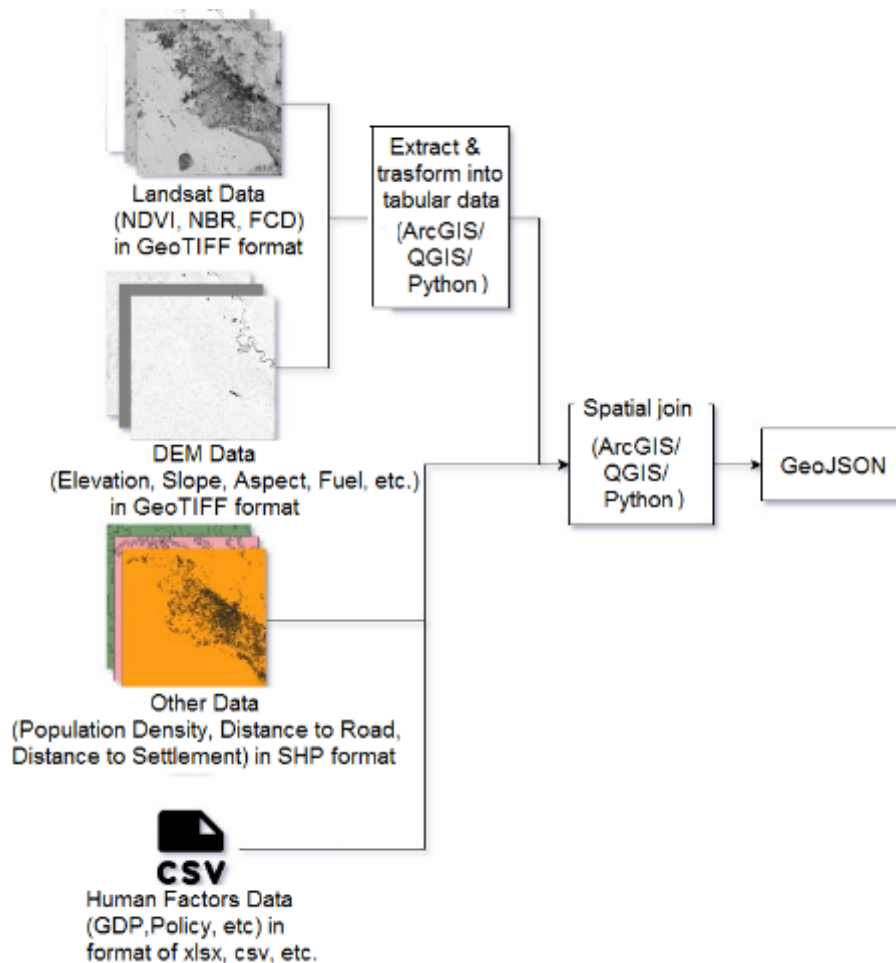


Figure 9: High-level process of data fusion

The 14 variables were processed into spatial data with a consistent format and grid size using ArcGIS software before conducting Machine Learning analysis. Data processing involved transforming the variables into GeoTIFF, shapefile, and tabular formats. Additionally, all the variables were further processed and converted into the shapefile format, maintaining the same grid size. ArcGIS plays a crucial role in processing and converting various spatial data. It offers a wide range of tools and functionalities that enable users to manipulate, analyse, and visualize spatial data effectively. It allows users to import and convert different types of spatial data, such as shapefiles, raster datasets, GPS data, and tabular data.

The variables that affect wildfire probability have been comprised by two groups: human related factors and physical-environment factors. When the data from GIS is ready, the next process is the data fusion that converting into decision support system. The data fusion needs some methodology that propose in the present work. Fuzzy logic was used to aggregate both human related variables and physical-environment variables. Montecarlo method was used to calculate the area of characteristics map. Bayes method was used to calculate the resources allocation priority to help the stakeholders to make the resources distributions decision.

3.1.2. Concept and Mathematical Modelling: Fuzzy Logic, Montecarlo and Bayes Method

3.1.2.1. Fuzzy Logic

In the field of scientific and technical computations, various equations which describe realistic problems like natural phenomena or engineering problems can lead to solving a system of linear equations. The exact numerical data might be unrealistic, but uncertain data could be considered as more aspects of a real-world problem. The fuzzy data is being used as a natural way to describe uncertain data. Fuzzy logic was proposed by Zadeh (1965), and, afterward, many papers and books were published in fuzzy system theory.

3.1.2.2. Basic Properties of Fuzzy Sets

Suppose B is a fuzzy set of elements, denoted generically by x , defined on the universe of discourse X , and represented by a membership function $\mu_B(x)$, then:

(a) Support of a fuzzy set:

The support of a fuzzy set B , $S(B)$, is the crisp set of all elements $x \in X$ such that $\mu_B(x) > 0$, and is written formally as in Eq. 3.3.

$$S(B) = \{ x \in X \mid \mu_B(x) > 0 \} \quad (3.3)$$

(b) Core of a fuzzy set:

The core of a fuzzy set B , $C(B)$, is the crisp set of all elements $x \in X$ such that $\mu_B(x) = 1$, and is written formally as in Eq. 3.4.

$$C(B) = \{ x \in X \mid \mu_B(x) = 1 \} \quad (3.4)$$

(c) Height of a fuzzy set:

The height of a fuzzy set B , $H(B)$, is the largest membership degree corresponding to any element in the set, and is written formally as in Eq. (3.5).

$$H(B) = \sup_{x \in X} \mu_B(x) \quad (3.5)$$

(d) Normal and subnormal fuzzy set:

A fuzzy set B is called normal when $H(B) = 1$ and called subnormal when $H(B) < 1$.

The properties of fuzzy sets (a) to (d) are illustrated graphically in Figure 10.

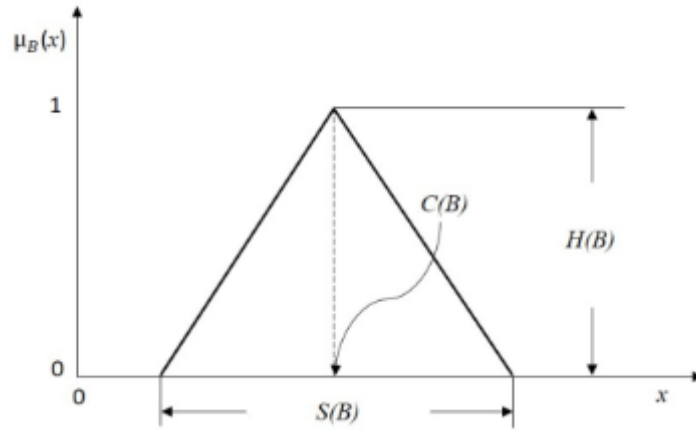


Figure 10: Support, core and height of a fuzzy set.

(e) α -cut of a fuzzy set:

Given any number $\alpha \in [0,1]$, the concept α -cut, denoted by B_α , of the membership function $\mu_B(x)$ is the crisp set of all elements, x , such that $\mu_B(x) \geq \alpha$, and is written formally as in Eq. 3.6. The α -cut concept, which also called α -level, can be illustrated graphically as shown in Figure 11.

$$B_\alpha = \{ x \in X \mid \mu_B(x) \geq \alpha \} \quad (3.6)$$

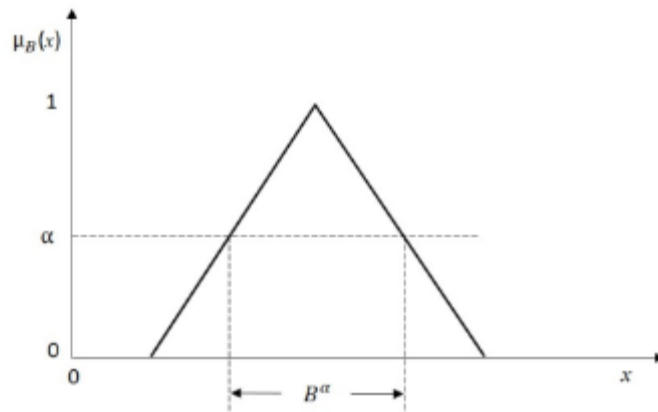


Figure 11: Illustration of a-cut concept

(f) Convex fuzzy set:

A fuzzy set B is called convex if all α -cut sets are convex, which means that joining any points of the α -cut lies completely within the set. Another definition of a convexity of a fuzzy set can be expressed as in Eq. 3.7, where $\mu(x)$ is a piecewise continuous function. A graphical illustration of a convex and nonconvex fuzzy set is shown in Figure 12.

$$\mu_B(x_2) \geq \min (\mu_B(x_1), \mu_B(x_3)), \quad \forall x_1 \leq x_2 \leq x_3 \quad (3.7)$$

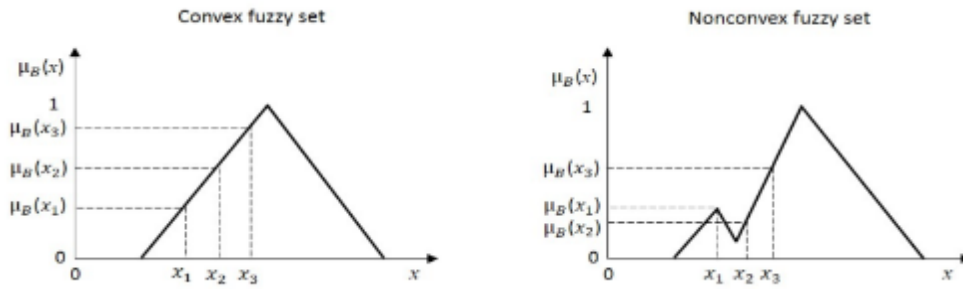


Figure 12: Convex and nonconvex fuzzy set

3.1.2.3. Basic Operations on Fuzzy Sets

Given two fuzzy sets, A and B defined on a universe of discourse X , Zadeh (1965) suggested that their *intersection*, *union* and *complement* operations can be generalised from the notion of standard crisp sets operations, *intersection*, *union* and *complement*, and written mathematically as in equations (3.8), (3.9) and (3.10), respectively. Graphical illustrations of these operations are shown in Figure 13, Figure 14 and Figure 15, respectively.

$$\mu_{A \cap B}(x) = \mu_A(x) \cap \mu_B(x), \quad \forall x \in X \quad (3.8)$$

$$\mu_{A \cup B}(x) = \mu_A(x) \cup \mu_B(x), \quad \forall x \in X \quad (3.9)$$

$$\mu_{A^c}(x) = 1 - \mu_A(x), \quad \forall x \in X \quad (3.10)$$

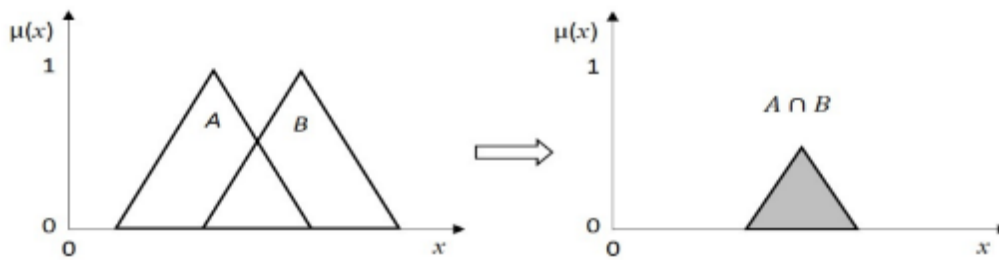


Figure 13: Intersection operation of fuzzy sets

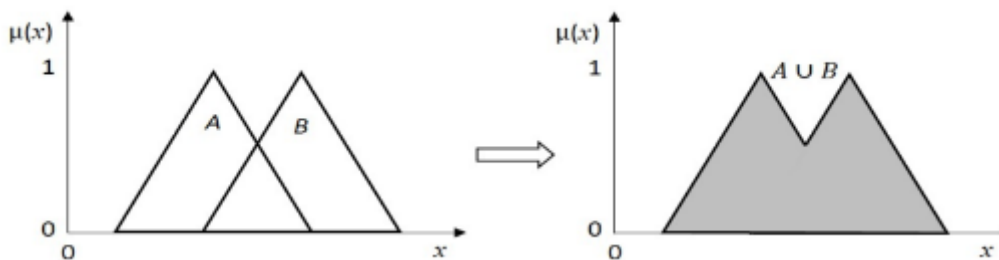


Figure 14: Union operation of fuzzy sets

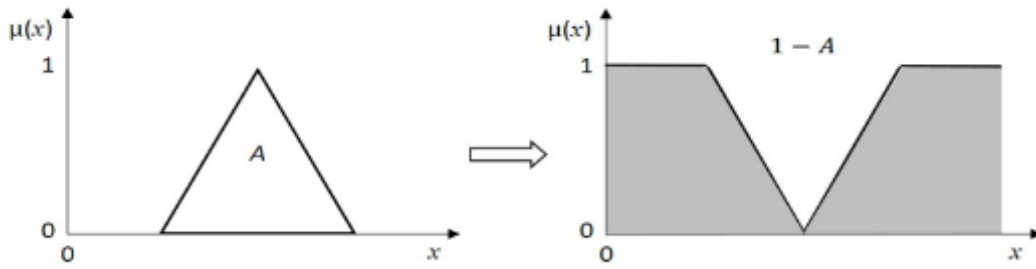


Figure 15: Complement of a fuzzy set

3.1.2.4. Monte Carlo Method

Wildfire risk management needs information about the characteristics and areas on the map. The map is always available in complicated geometry and difficult to calculate the area. The Monte Carlo technique gives an easy but powerful and accurate method to handle that problem. The area finite area integral may effectively use the Monte Carlo method [Landau and Binder, 2009][Reiter,2008]. As shown in Figure 16, to calculate the area A, under $f(x)$ with the left border a and right border b is expressed as:

$$A = \int_a^b f(x) dx \quad (3.11)$$

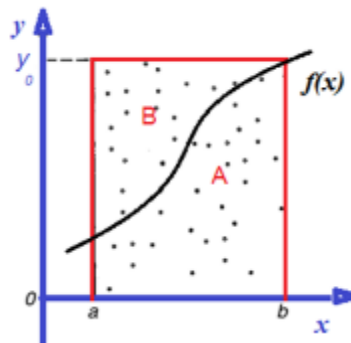


Figure 16: Monte Carlo for integral area

A lot of random N dots were hit at the red square and will equal the number of dots that hit areas A and B ($N = A + B$). The proportion area A of the total red squared area is expressed as:

$$\frac{A}{N} = \frac{\int_a^b f(x) dx}{y(b-a)} \quad (3.12)$$

So, to estimate the area A expressed as:

$$\hat{A} = \frac{A}{A+B} y(b-a) \quad (3.13)$$

where A is the number of dots that hit area A and B is the number of dots that hit area B .

3.1.2.5. Bayesian Method

The *sample space* Y is the set of all possible datasets, from which a single dataset y will result. The *parameter space* Θ is the set of possible parameter values, from which we hope to identify

the value that best represents the true population characteristics. The idealized form of Bayesian learning begins with a numerical formulation of joint beliefs about y and ϑ , expressed in terms of probability distributions over Y and Θ . The posterior distribution is obtained from the prior distribution and sampling model via *Bayes' rule* (Hoff, P.D., 2009):

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int_{\Theta} p(y|\bar{\theta})p(\bar{\theta}) d\bar{\theta}} \quad (3.14)$$

Suppose $\{H_1, \dots, H_K\}$ is a partition of H , $\Pr(H) = 1$, and E is some specific event. The axioms of probability imply the following:

Rule of total probability:
$$\sum_{k=1}^K \Pr(H_k) = 1 \quad (3.15)$$

Rule of marginal probability:
$$\Pr(E) = \sum_{k=1}^K \Pr(E \cap H_k) = \sum_{k=1}^K \Pr(E|H_k) \Pr(H_k) \quad (3.16)$$

Bayes' rule:
$$\Pr(H_j|E) = \frac{\Pr(E|H_j) \Pr(H_j)}{\Pr(E)} = \frac{\Pr(E|H_j) \Pr(H_j)}{\sum_{k=1}^K \Pr(E|H_k) \Pr(H_k)} \quad (3.17)$$

3.1.2.1. Case example

The important characteristics of the wildfire variables are the weights of every variable. Here, we will explain how to calculate the weights for variables. The reference that we use here were using weights data in China (Weijie, C., et. al., 2021) as shown in the Table 4. The different location must be having the different characteristics and so the difference weights.

Table 4: Weights used for the area under examination

No.	Wildfire-Related Variables	Weight
1	DS	0.1265
2	Vegetation Type	0.1227
3	DR	0.1182
4	Annual precipitation	0.1043
5	Fire—spot density	0.0997
6	Land-Usage Type	0.0922
7	Elevation	0.0873
8	NDVI	0.0789
9	Aspect	0.0554
10	Fuel load	0.0376
11	Population density	0.0297
12	Annual temperature	0.0245
13	Slope	0.0134
14	GDP	0.0096
		Total Weight = 1.0000

After every weight was found, then the next calculations are grouping all 14 variables into two groups: human-related factors (HRF) and physical-environmental factors (PEF).

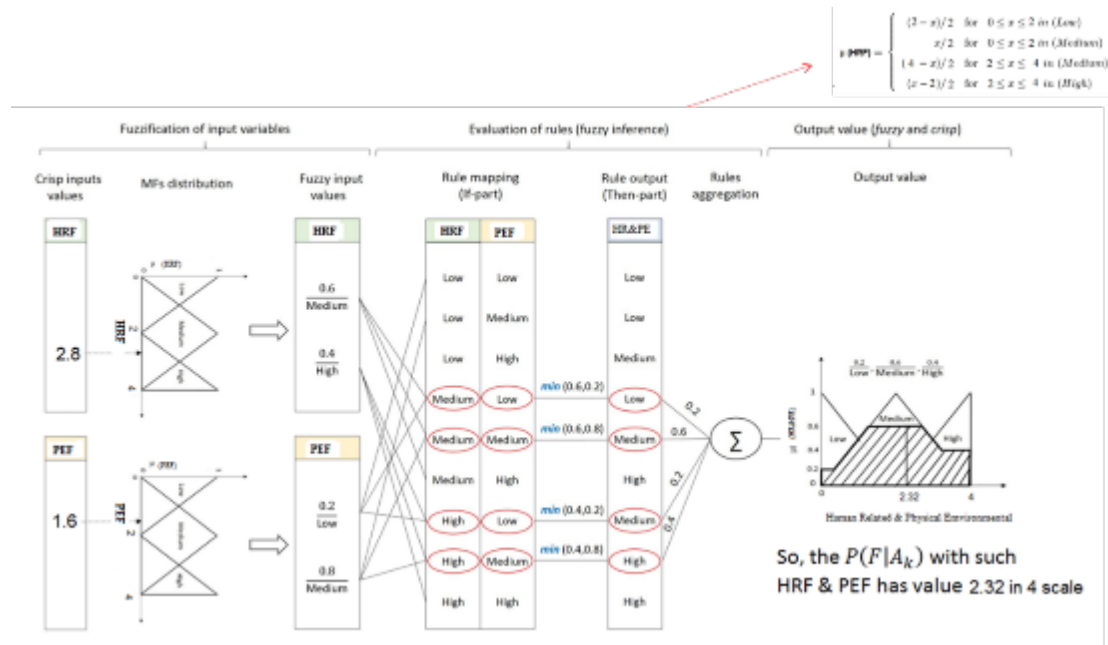


Figure 17: Scheme for case example of how Fuzzy Logic and Bayesian Method work.

The first parts on the left in Figure 17 are HRF (human-related factors) or anthropomorphic variables and PEF (physical-environmental factors). Five (GDP, Population Density, Distance to Settlement, Distance to Road, and Historical Fire) out of 14 variables are HRF in Table 1. While the rest (9 variables) are PEF. All variables are discretized into 4 levels as shown in Figure 17. The values of 2.8 for HRF and 1.8 for PEF are the averaged value for the 4 scale. Those crisp values are fed into Fuzzy Logic due to the advantages of handling the vague quantities of human factors that must be mixed with physical-environmental factors.

In Fuzzy logic, the parameters involve fuzzification and defuzzification where crisp input will be processed also into crisp output via center-of-gravity calculations. The calculated value 2.32 as shown in Figure 17 is $P(F|A_k)$ or the Probability of Fire level 2.32 in 4 scale if the area A_k must be fired. Here we need the area of A_k that is calculated using Monte Carlo method. After all area has the value of $P(F|A_k)$, then the priorities of $P(A_k|F)$ are calculated using Bayes theorem. The meaning of $P(A_k|F)$ is the level priorities if all domain under consideration must be fired.

3.2. Software implementation and results

3.2.1. Data entry

The complete results for all 14 variables are presented in Figure 34 up to Figure 37. All measurements were scaled into four levels. The standard of factor discretization is presented in Table 3 (Chen et al., 2021). Table 5 shows the first, second, third and fourth bracket represent the value of 1, 2, 3 and four level, respectively. Five of those 14 variables are the human related factors that influence the fire risk characteristics of the forest or wildland. Those five are Distance to Settlement, Distance to Road, Population Density, GDP (gross

domestic product) and Historical Fire. The rest of 9 variables are physical environmental factors that constitute from meteorological factors (Temperature, Aspect, and Precipitation) and physiographic factors (Elevation, Fuel Load, Land Usage, NDVI, Vegetation Type and Slope). Detailed data acquisition and methods was described below.

Table 5: Variables influence

No	Variable	Proportional
1.	Distance to Settlement (m)	(0, 356.5), (356.5, 635.7), (635.7, 1041.8), (1041.8, ∞)
2	Distance to Road (m)	(0, 832.6), (832.6, 2043.7), (2043.7, 3860.3), (3860.3, ∞)
3	Elevation (m)	(0, 145), (145, 295), (295, 575), (575, ∞)
4	Fuel load (ton/km ²)	(0, 1), (1, 1.3), (1.3, 23.3), (23.3, ∞)
5	Historical fire (events/0.75km ²)	(1), (2), (3), (≥ 4)
6	Land Usage	refer to Table 1
7	NDVI	(0, 0.8), (0.8, 0.86), (0.86, 0.90), (0.90, 1)
8	Population Density (people/km ²)	(0, 91.9), (91.9, 144.7), (144.7, 303.1), (303.1, ∞)
9	GDP (21million idr/km ²)	(0, 194), (194, 400), (400, 638), (638, ∞)
10	Vegetation Type	refer to Table 2
11	Aspect (°)	North (0°, 45°) / (315°, 360°), East (45°, 135°), South (135°, 225°), West (225°, 315°)
12	Slope (°)	(0, 3.3), (3.3, 10), (10, 18.3), (18.3, 90]
13	Temperature (°C)	(<19.9), (19.9 – 21.5), (21.5 – 22.8), (>22.8)
14	Precipitation (mm/y)	(<1730.7), (1730 – 1950.6), (1950.6 – 2095.6), (>2095)

a. Distance to settlement

Prior to computing the distance to the settlement, it is needed to conduct land cover classification to identify the settlement area as shown in Figure 18 and Figure 19 . This classification process utilizes Landsat satellite imagery and Sentinel satellite imagery as data sources. The chosen method for land cover classification is the maximum likelihood method. Subsequently, a buffer analysis is performed to determine specific areas located at a given distance from the settlement area. The buffer distance is determined according to the criteria outlined in Table 5. The Euclidian buffer method was implemented in this study. This method measures distances on a two-dimensional Cartesian plane, calculating the distances between two points on a flat surface. The Euclidean buffer is suitable for analysing distances around features in a projected coordinate system within a relatively small area, such as one UTM zone.

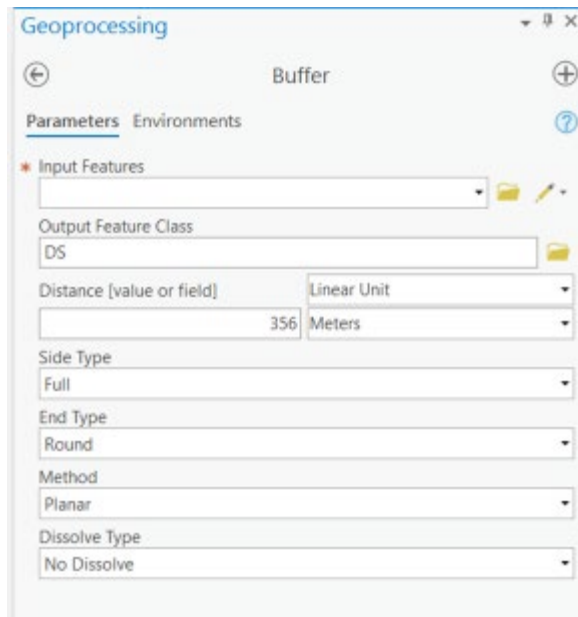


Figure 18: Buffer tools in ArcGIS Pro

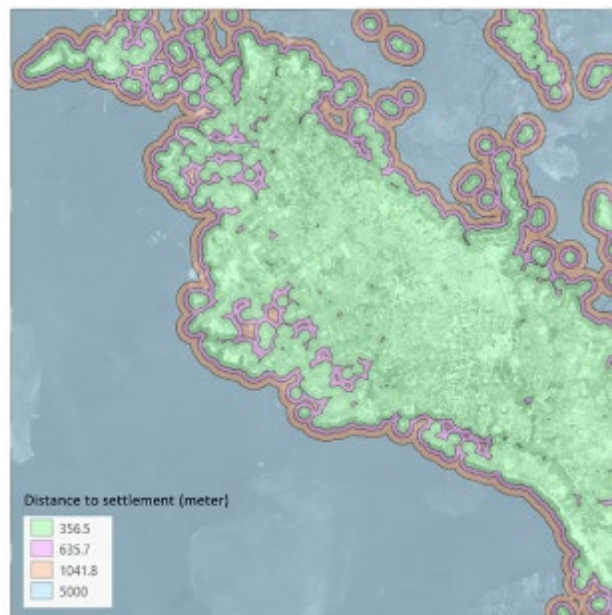


Figure 19: Distance to settlement

b. Distance to road

The primary data required for this analysis consists of road data sourced from OpenStreetMap (Figure 20). To ascertain the distance to the road location area, the buffer analysis method is applied within the ArcGIS software. The buffer distance is tailored based on the criteria provided in Table 5.

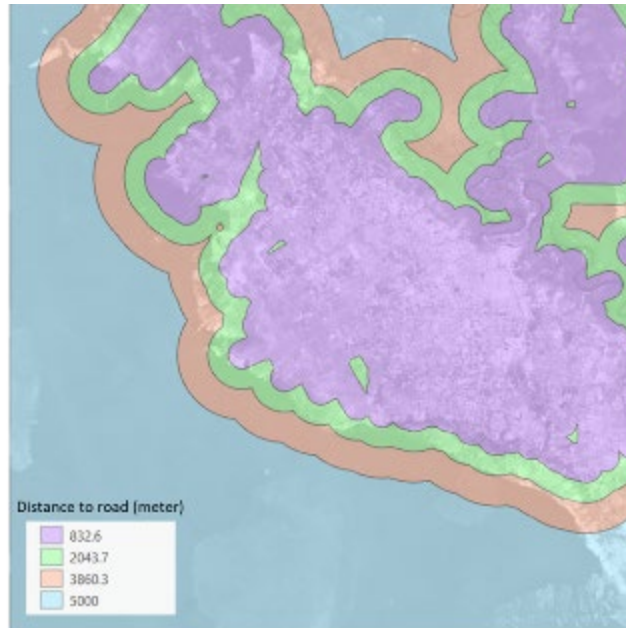


Figure 20: Distance to road

c. Elevation

The essential data for elevation classification is the Digital Elevation Model (DEM). One widely used global DEM source is ASTER GDEM, which was found to be used extensively in various studies. Access to ASTER GDEM is available at Earth data website. It's important to note that ASTER GDEM data has a resolution of 30 meters, making it suitable for large-scale topographic analysis, though it may not be ideal for high-resolution local studies. Elevation classification was performed using the ArcGIS software (Figure 21), employing the reclassify tools in accordance with the criteria outlined in Table 5.

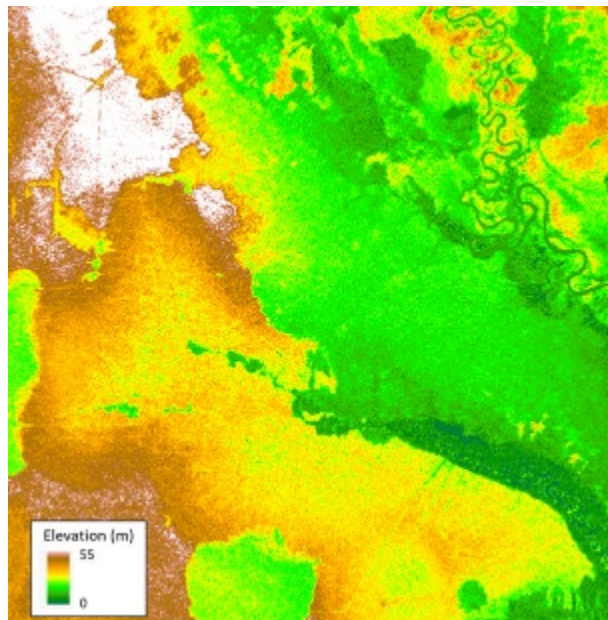


Figure 21: Elevation data

d. Fuel load

Landsat satellite imagery is required to calculate the fuel load. The Landsat imagery data can be obtained from the USGS website by downloading all available bands (Figure 22). The calculation of fuel load (Figure 23) can be performed using either ArcGIS or Python. Fuel load estimation is calculated based on vegetation indices including EVI (Enhanced Vegetation Index), SAVI (Soil Adjusted Vegetation Index) and NDVI (Bao et al., 2022).

$$NDVI = (NIR-RED)/(NIR+RED)$$

Where, NIR stands for Near-Infrared Light and RED stands for Visible Red Light.

$$FUEL\ LOAD = 50.78EVI + 237.64NDVI + 109.38SAVI - 102.83$$

Where, EVI stands for Enhanced Vegetation Index, SAVI stands for Soil Adjusted Vegetation Index and NDVI stands for Normalized Difference Vegetation Index.

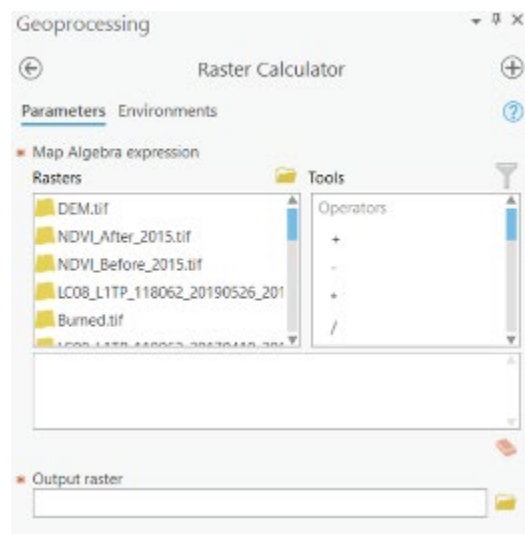


Figure 22: Raster calculator tools

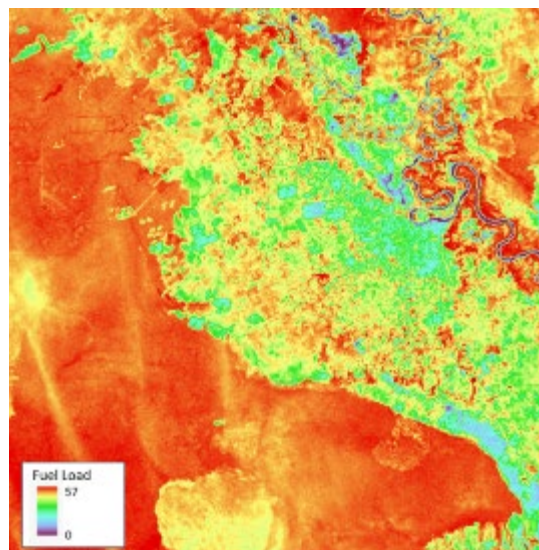


Figure 23: Fuel load

e. Historical fire

To calculate historical wildfire events within a specific unit area, it is necessary to acquire data on forest fire events. This data can be obtained from the local government, such as Indonesia's

Regional Disaster Management Agency, for example. The density of wildfire events was computed using ArcGIS software. The study area was divided into a grid, with each grid cell measuring 0.75 km²

f. Land Use

Land use identification using multispectral imagery has been extensively studied and applied in various research and practical applications. In this study, we utilized Landsat to perform identification and classification of land usage using maximum likelihood method. Land usage was classified as follows: 1) Paddy field, Dry land, Water area, Unused land, Urban-rural fringe, Industrial and mining land, Residential land; 2) Meadow, Grassland, Alpine vegetation; 3) Broad-leaved forest, Shrub; and 4) Coniferous forest, Theropencedrymion. Additionally, each land usage is assigned a score based on the criteria outlined in Table 5.

g. NDVI

NDVI has been widely used as an indicator of several factors such as canopy density, biomass, plant health, and vegetation productivity (Ghaffarian et al., 2020; Turubanova et al., 2015; Verbesselt et al., 2016). Furthermore, NDVI is also effective to assess vegetation damage, stress, recovery (Rezaei et al., 2021; Smith et al., 2014). NDVI time series monitoring using remote sensing images can be used to determine vegetation growth and recovery regarding to this ecological resilience program. NDVI is the ratio of the difference between the near-infrared band (NIR) and the red band (R) and the sum of these two bands (Rouse et al., 1974; Yengoh et al., 2015).

$$NDVI = (NIR-RED)/(NIR+RED)$$

Where NIR stands for Near-Infrared Light and RED stands for Visible Red Light. NDVI was processed using ArcGIS and Python (Figure 24).

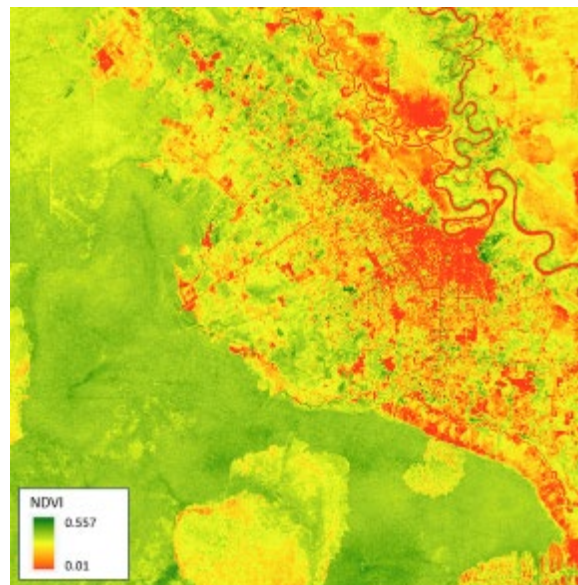


Figure 24: NDVI data

h. Population density

Population density data was acquired from the World Population Database with a resolution of 1 km x 1 km. Moreover, the population density is categorized based on the criteria specified in Table 3, utilizing the Reclassify tool within ArcGIS.

i. GDP

The GDP data represents economic factors relevant to the inhabitants of the forest fire-prone region. This data was obtained from the World Bank database, but the resolution is at the country level. For more detailed economic data, additional information from local government sources can be incorporated. The GDP data, initially in tabular format, was transformed into spatial data by integrating it with administrative boundary spatial data. The resulting converted data is then represented in the form of a shapefile.

j. Vegetation type

Landsat imagery was used as the primary data source for classifying vegetation types in this study, although other multispectral satellite images could be utilized as well. The vegetation types are grouped into four categories: 1) Desert, Swamp, Cultivated plants; 2) Meadows, Grassland, Alpine vegetation; 3) Broad-leaved forest, Shrub; and 4) Coniferous forest, Theropence drymion. Furthermore, each vegetation type is assigned a score based on the criteria provided in Table 5.

k. Aspect

The aspect of a slope refers to the direction in which the slope faces. It is typically measured in degrees from 0° to 360°, with 0° representing north and 180° representing south. The aspect of a slope plays a significant role in various processes and phenomena, including hydrological processes, erosion, vegetation distribution, and solar radiation exposure. We utilized ASTER GDEM data to calculate aspect using Spatial Analyst tools in ArcGIS (Figure 25, Figure 26). Furthermore, the aspect calculation result data is classified according to Table 5.

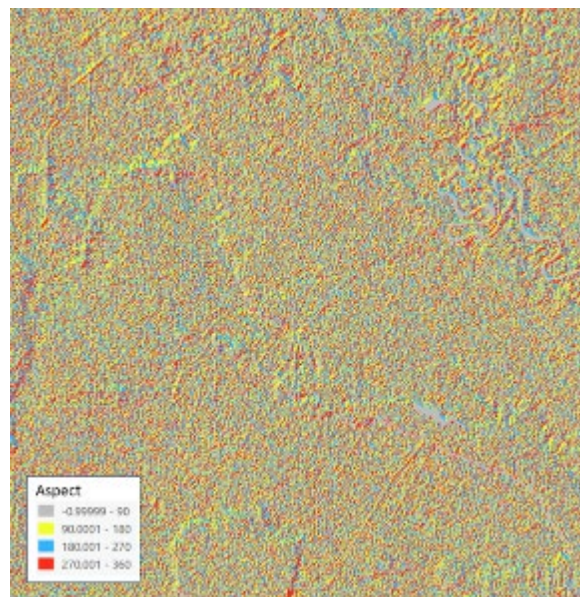


Figure 25: Aspect data

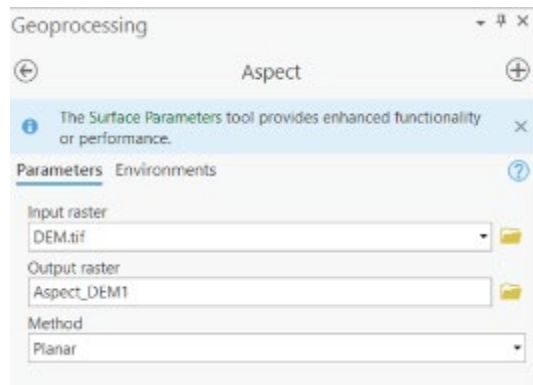


Figure 26: Aspect tools in ArcGIS

I. Slope

Slope, in this context, refers to the measurement of elevation change between adjacent cells in a digital elevation model (DEM). ASTER GDEM data was processed using Calculate Slope in ArcGIS (Figure 27, Figure 28). The Calculate Slope tool creates a surface that represents slope by utilizing elevation data.

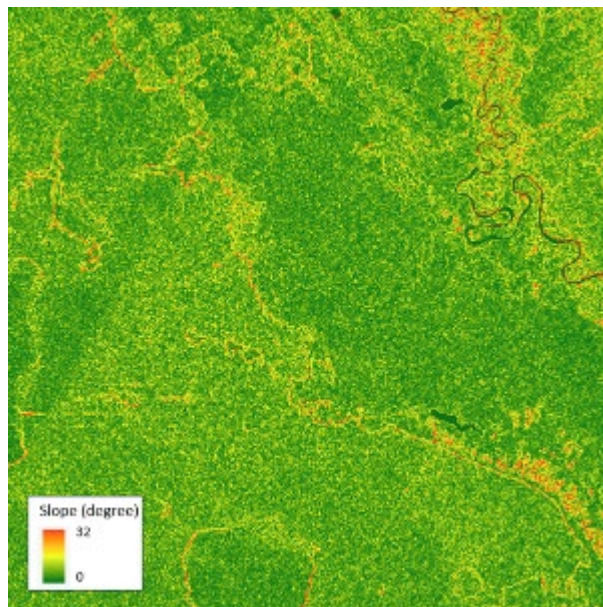


Figure 27: Slope data

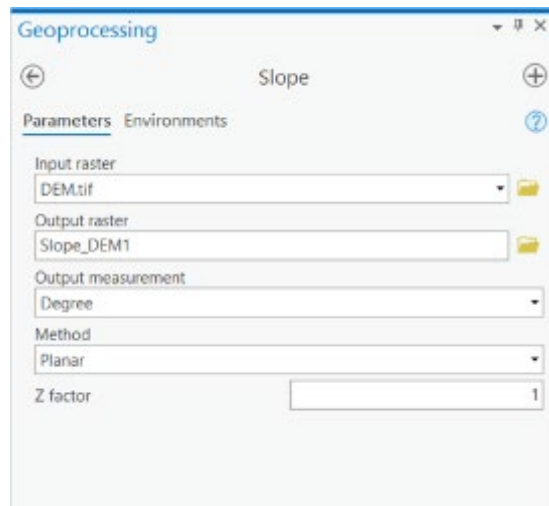


Figure 28: Slope tools in ArcGIS Pro

m. Temperature

Temperature data was obtained from ERA5-reanalysis. The fifth generation of the ECMWF's reanalysis of the previous eight decades' worth of global climate and weather is known as ERA5. Data is accessible starting in 1940. For a vast range of atmospheric, oceanic, and land-surface parameters, ERA5 gives hourly estimates. Due to its relatively coarse spatial resolution, a single ERA5 data grid can effectively cover the entire pilot project location. The variable chosen for this study is the 2m temperature, focusing on its annual average value.

n. Precipitation

The precipitation data utilized in this study comprises the annual total precipitation obtained from ERA5-reanalysis data. Additionally, the total precipitation was categorized based on the criteria outlined in Table 5.

3.2.2. Initialization and application execution

The 14 variables data consists of various formats. To conduct Machine Learning analysis, the data needs to be in GeoJSON format. Hence, it is necessary to convert and merge the data into GeoJSON format by following these steps:

1. Convert all data files to shapefiles format (Figure 29).

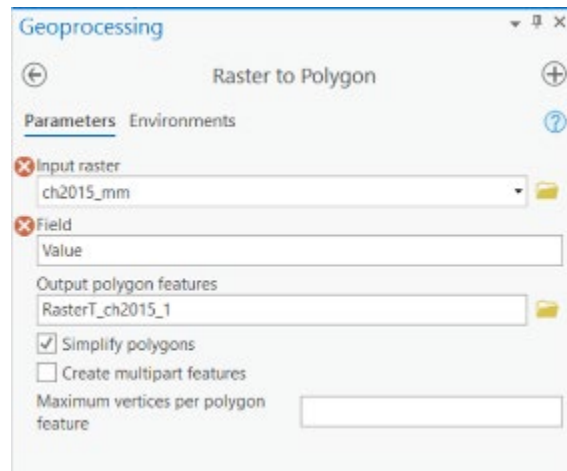


Figure 29: Raster to polygon tools

2. Generate a polygon shapefile grid with dimensions of 0.75km x 0.75 km (Figure 30).

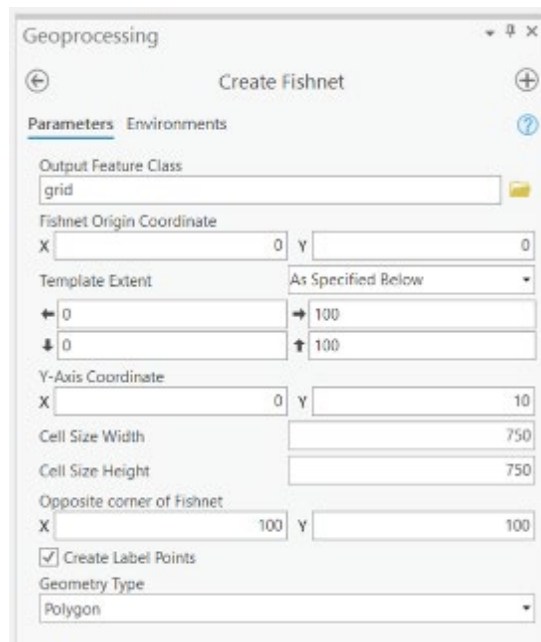


Figure 30: Generate shapefile grid

3. Overlay all the shapefiles onto the shapefile grid to create a new shapefile grid (Figure 31).

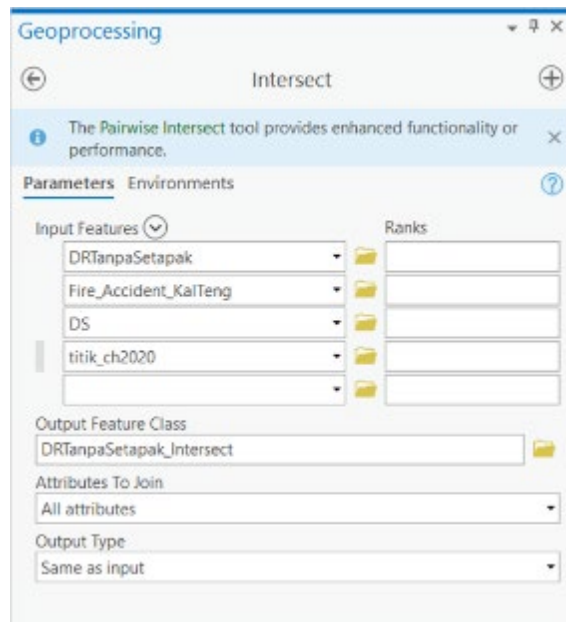


Figure 31: Overlay all parameters to one shapefile grid

4. Conduct a spatial join for all the new grid shapefiles, combining them into a single shapefile with complete attribute data (Figure 32).

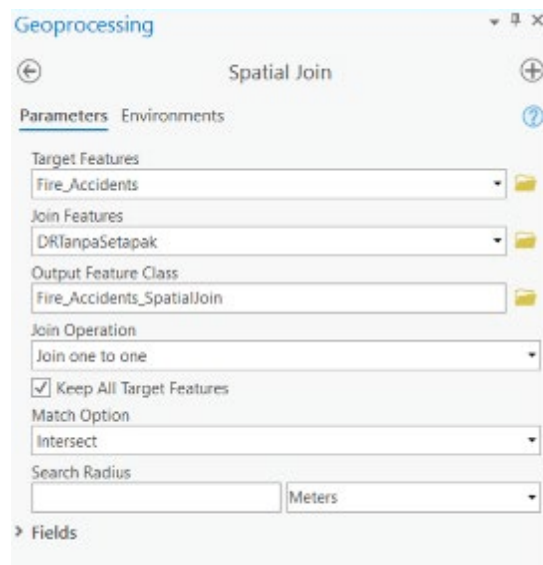


Figure 32: Spatial join tools

5. Convert the grid shapefile to GeoJSON format (Figure 33).

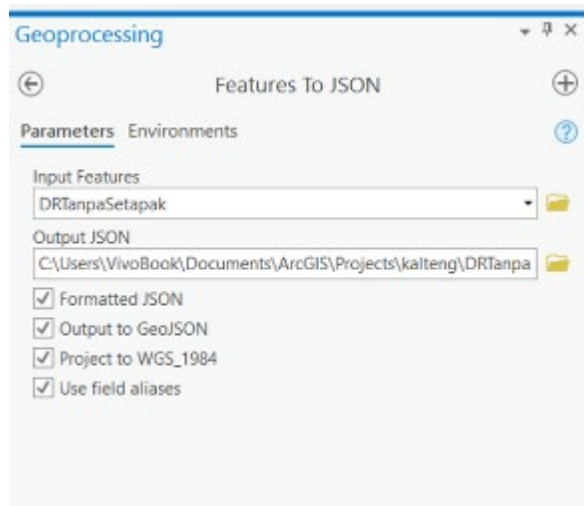


Figure 33: Example convert shapefile to GeoJSON

According to survey, around 90% of wildfires has direct or indirect relation with human factors (Depicker et al., 2020)

Distance to Settlement (Table 5; no. 1) and Distance to Road (Table 5; no. 2) show the main variables related to human factors for fire probability due to those variables represent the main human activities factors. Figure 34 shows that the dominant colour is red (3.1 – 4) means the area was dominated remotes area or far from settlements. Figure 35 shows the colour of the map is dominated in green colour means there are so many roads that available on the area for human movement.

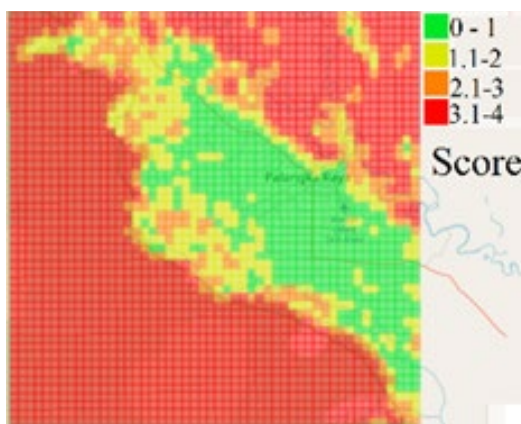


Figure 34: Distance to Settlement

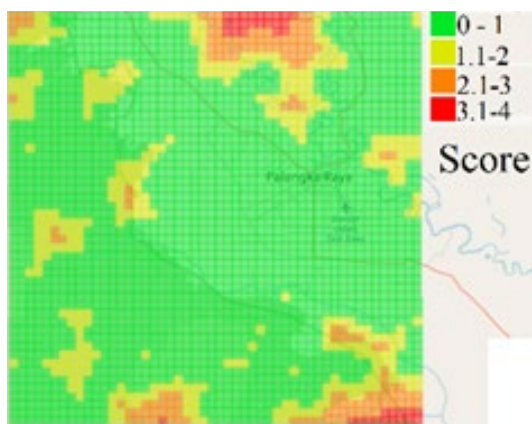


Figure 35: Distance to Road

Population density represents the population assemblage or the number of people in every unit of area. In a densely populated area it is reasonable to expect high positive correlation with human activities and high fire probabilities in consequence. Figure 36 shows the Population Density Map that is represented by the colored scale. GDP represents the economic status of the region. For this present case as shown in Figure 37, the GDP per capita is nearly similar, so the level of GDP in area also similar to the area. This means that denser areas will consequently have a higher level GDP. But this phenomenon may be different for the others area when the area has non uniform GDP per capita.

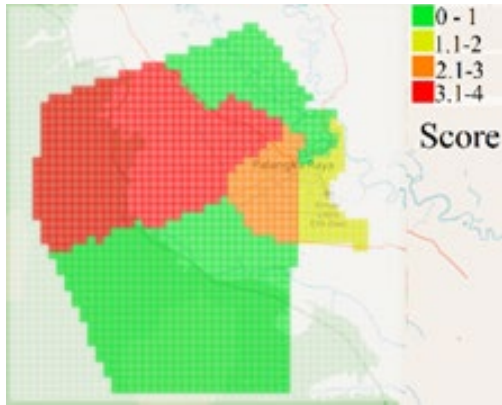


Figure 36: Population Density

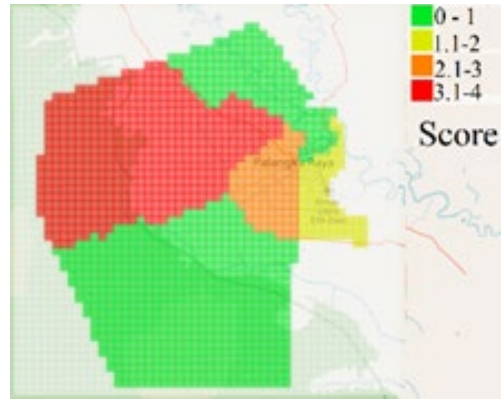


Figure 37: GDP

Historical Fire variables represent the distribution of past wildfire events on certain area. The data monitored by National Meteorology, Climatology and Geophysics and also recorded by Forestry and Environment Ministry. Figure 38 shows the Historical Fire map that has one center highest event. The Precipitation level in the research area is posed in the Figure 39. The precipitation figure shows only red color means in the research area has the same level of precipitation in the highest level.

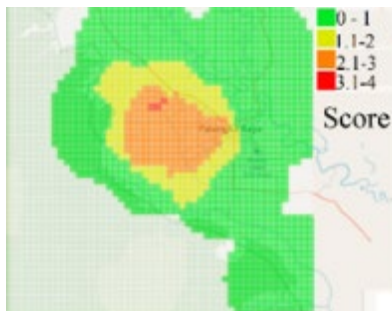


Figure 38: Historical Fire

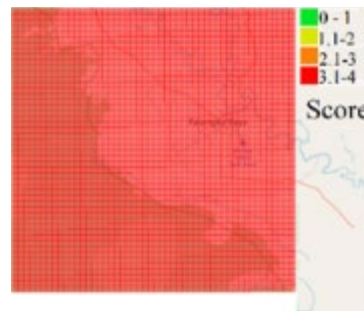


Figure 39: Precipitation

The same character parameter for the research area is the temperature. Figure 40 shows that for average annual temperature has same level with the highest score. That means that the area of interest has the same level with the highest score of the average annual precipitations.

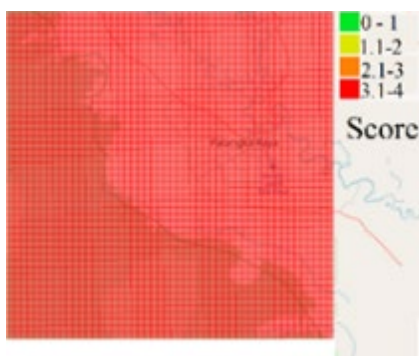


Figure 40: Temperature

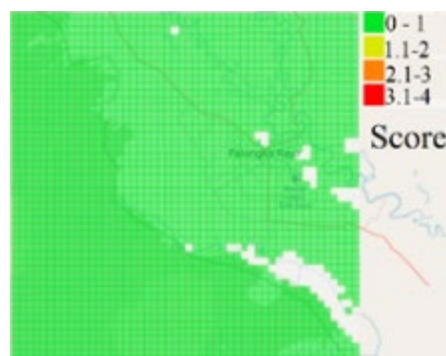


Figure 41: Elevation

The opposite result to temperature is observed in relation to the elevation. Figure 41 shows only one green color, which means that across the entire area of interest we observe uniformly the lowest level of elevation (flat fields). Otherwise, the same result with the elevation is observed related to the fuel load. Figure 42 shows one level fuel load that is in green color or the lowest scale. The little bit different character parameter is posed by aspect.

Figure 43 shows the colored map result of aspect with score 2 domination and little scattered level 1 and 3 across area. Similar unit in degree (°) with aspect but with detail situational profile is slope as shown in Figure 44. The slope or local gradient character of the research area has quite various level although only in the range of 1 -3 score that means there are no steep contours.

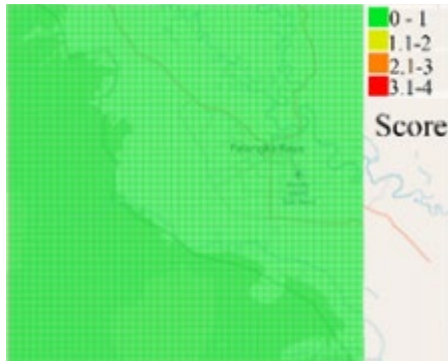


Figure 42: Fuel load

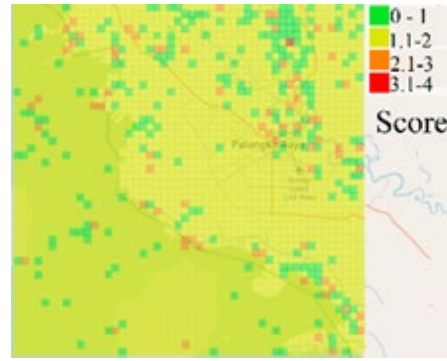


Figure 43: Aspect

The next parameter is related to the greenness level *i.e.* normalized difference vegetation index (NDVI). The coverage of vegetations in the surface of the land is represented by the NDVI level. Figure 45 shows the colored map that the area dominates by level 1. The detailed information that what available on the surface land plots on the next parameters, *i.e.* Land Usage and Vegetation Type as shown in Figure 46 and Figure 47, respectively.

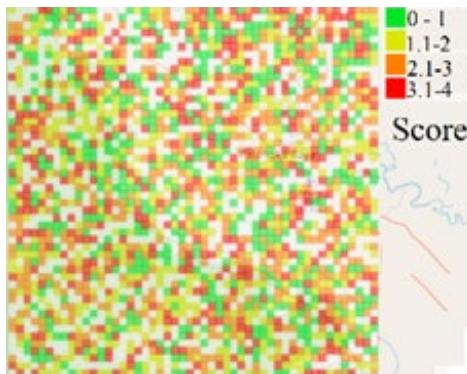


Figure 44: Slope

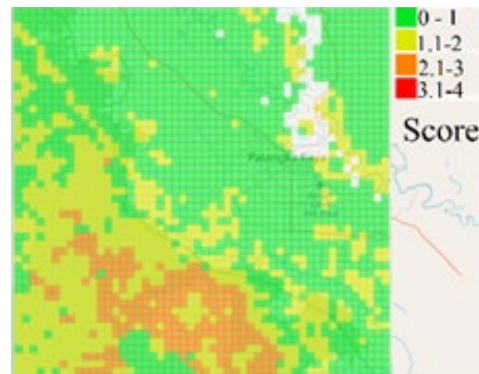


Figure 45: NDVI

As shown in Figure 46 and Figure 47, the results in colored map very similar due to the criteria of both Land Usage and Vegetation type also quite similar

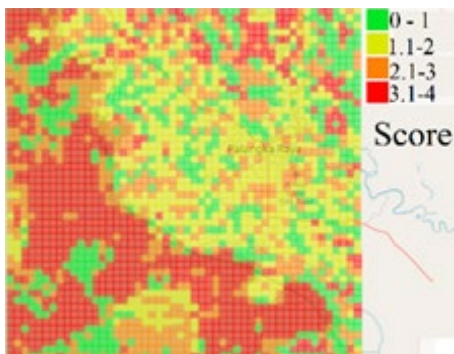


Figure 46: Land usage

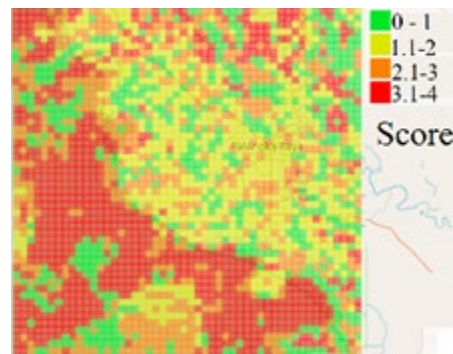


Figure 47: Vegetation Type

Before analysis, all the parameters are pre-processed and consolidated into a single JSON file. Once prepared, this file can be uploaded to the system, as illustrated in C, to conduct a priority resources area analysis.

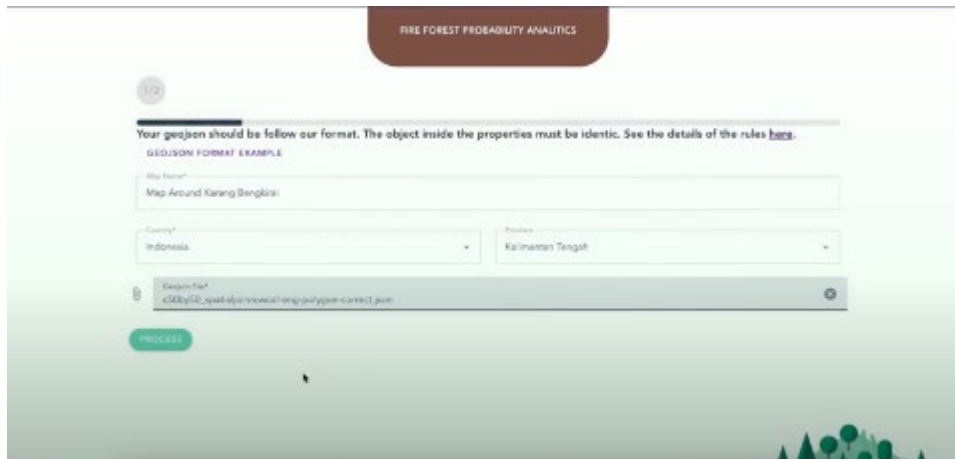


Figure 48: User interface for parameter initialization and application execution

As shown in Figure 48, the user is required to input the map name, country, and province to provide supplementary information for the JSON data. This supplementary data is utilized as support data for creating a data library.

The analysis process initiates by determining the wildfire probability of the specified area using fuzzy logic. This system leverages the Flask-Python Framework and executes several sequential steps, including fuzzification, inference, and defuzzification.

The fuzzification step involves converting crisp (precise) parameters into fuzzy sets. Fuzzy sets are used to represent linguistic variables, which are terms like "low," "medium," or "high" that describe the input variables in a fuzzy logic system.

Moving on to the second step, inference, the fuzzy logic system employs a set of rules expressed in the form of "IF-THEN" statements. These rules establish relationships between the fuzzified input variables and the output variable. The system combines the fuzzy rules and the degree of membership of the input variables to determine the degree of membership of the output variable's fuzzy set.

The last step of fuzzy logic is defuzzification. The fuzzy output obtained from the inference step needs to be converted back into a crisp value by centroid methods. The Centroid method is considered the center of mass or the "center of gravity (cog)" of the fuzzy set, and it serves as the final output of the calculation of wildfire probability, *i.e* shown in Figure 49.

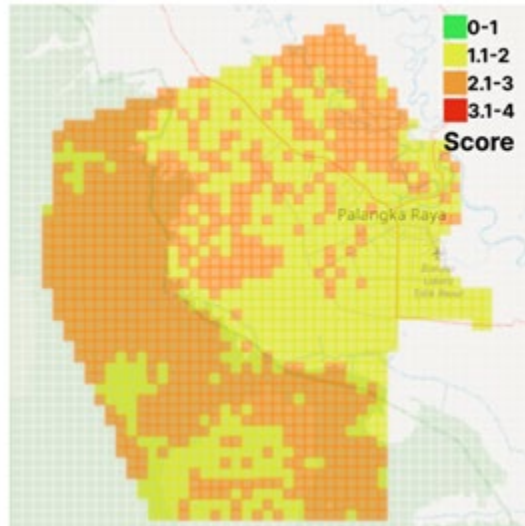


Figure 49: Fire Probability

To analyze the allocation priority of resources, we integrate the wildfire probability and the area-wide considerations using Bayesian methods. The area-wide calculation involves employing a ratio approach through Monte-Carlo methods, where random points are distributed across the entire grid area and counted accordingly, *i.e* priority resource shown in Figure 50.

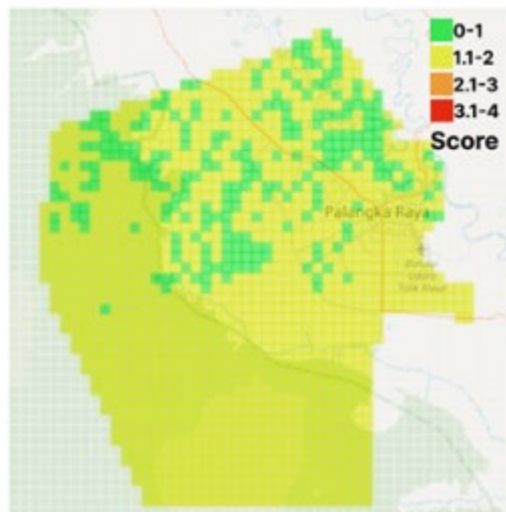


Figure 50: Priority Resource

Furthermore, the obtained results offer support for displaying detailed information about specific areas by simply clicking on the grid, as illustrated in Figure 51. Additionally, the system is equipped to handle time series data, as depicted in Figure 52. This enables users to explore and analyze data over a period of time, enhancing the system's capabilities for comprehensive data visualization and analysis.

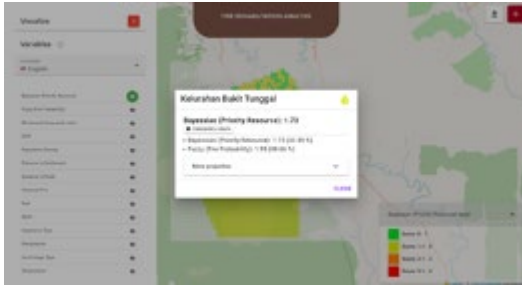


Figure 51: Detailed data of a variable

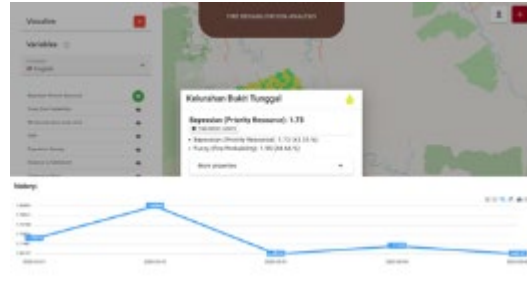


Figure 52: Over time Data

3.3. Component integration

We assume integration involves two main components: deployment and app integration. For deployment, we utilize Docker Container, providing a user-friendly and efficient approach to deploying our app. Detailed documentation for this process can be found on the Silvanus GitHub: <https://github.com/silvanus-prj/fire-probability-analytics-back-end>.

Regarding app integration, pilots can easily integrate with our system by requesting HTTP access through our REST API. The API documentation, available on Postman (**Allocation Resource Analysis** section). The provided preview is presented below:

1. Get Available Factors:

This request returns a detailed overview of all the factors. The results encompass Bayes analysis using both the Area Wide (Montecarlo) technique and Fuzzy Logic, involving a total of 14 variables.

```

{{url}}/factors
Response
Body Headers (6) 200 OK
json
{
  "data": [
    {
      "description": "Hasil perhitungan analisa alokasi sumber daya.",
      "geojsonProp": "4scale",
      "id": 1,
      "is_factor": false,
      "name_text": "Bayessian (Prioritas Sumber Daya)",
      "state": false
    }
  ]
}
View More

```

2. List of Maps

This request returns the list of analyzed maps.

```
{{url}}/available-map
```

Response

Body Headers (6)

200 OK

```
json
{
  "data": [
    {
      "country": "Indonesia",
      "id": 1,
      "map_date": "2015-01-01",
      "name": "Kerang Bengkirai - 2015",
      "process_id": 1,
      "province": "Kalimantan Tengah"
    }
  ]
}
```

3. Map Detail

This request offers comprehensive details about the specific area of the map data.

```
{{url}}/result/{{process_id}}
```

Response

Body Headers (1)

200 OK

```
json
{
  "data": {
    "features": [
      {
        "geometry": {
          "coordinates": [
            [
              [
                113.67155802200000,
                -2.4116666666666666
              ]
            ]
          ]
        }
      }
    ]
  }
}
```

4. Show Timeseries Graph

This request offers the chronological sequence of variable changes over time.

```
{{url}}/process/{{process_id}}/row/{{row}}/column/{{column}}/property/{{geojsonProp}}
```

Response

Body Headers (1) 200 OK

```

json
{
  "data": [
    {
      "4scale": null,
      "file_url": "uploads/M4GXFLBov5FEYL4RpoJd2e/d7b64e581228d5c5c18fc50a75bcd417d0...",
      "id": 1,
      "map_date": "2015-01-01",
      "maps_id": null,
      "process_id": 1,
      "user": "M4GXFLBov5FEYL4Rpo..."
    }
  ]
}

```

View More

Further detailed API documentation, available on Postman (**Allocation Resource Analysis** section): <https://s.id/amikom-silvanus-api>, offers comprehensive information and guidelines for seamless integration.

3.4. Data Fusion Future Plan

An extended data fusion methodology exploiting the above results is depicted in Figure 73Figure 53 and will be describes below.

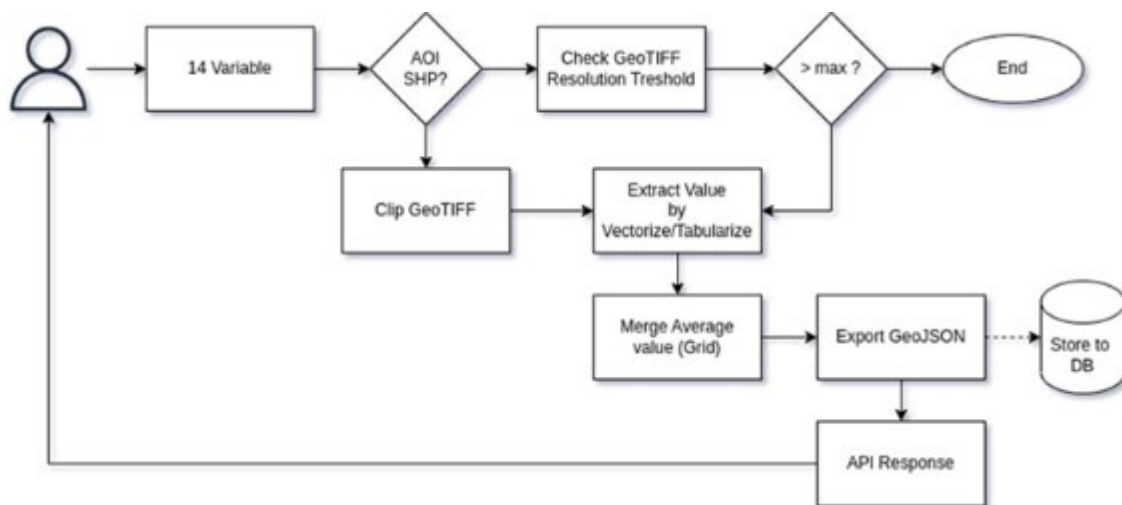


Figure 53: Extended process for data fusion

3.4.1. Replacement of ArcGIS with our own apps

The plans for future development for the Data Fusion App are in regard with ArcGIS replacement by our own code with steps, as follows:

- Users input 14 variables with certain format files.
- The System reads the input files,
- Describe the certain area under considerations and must avoid unnecessary area due to wasteful resources.
- The system read the availability of the area under consideration (whether available or not). The decision

- False: If user does not prepare the area border, then the system will read the satellite image with default resolution. If the size of the default resolution is greater than the threshold (too big), the process will stop.
 - True: If the user prepares the area border, the system will clip the satellite image.
- e. The process will continue only if the size of the image satellite resolution is under the threshold. Otherwise, the process of another step (downsizing) will be mandatory.
 - f. The system will extract the image into certain parameter values or variables.
 - g. The extracted parameter values will be joined into Grid (50x50).
 - h. The results will have three functions:
 - The system will export into GeoJSON format that contains the spatial joints values.
 - The system will store DB in certain locations.
 - The system will respond to the JSON to the users as API Response.

3.4.2. Weights Calculation for 14 variables in different area

The idea for calculating all 14 variables is that they have a correlation to the dependent variable (burned/unburned). The measurement of the strength of linear correlation between dependent and independent variables is used Pearson's correlation (r). The value of the Pearson's correlation is $-1 < r < 1$, where a positive correlation means that the variables move in the same direction. Put another word, it means that as one variable increase so does the other, and conversely, when one variable decrease so does the other. A negative correlation means that the variables move in opposite directions.

4. Resource allocation of response teams

4.1. Concept of operation

The main objective of this component is to assist commanders take optimal decisions regarding the resource allocation of response teams in the field depending on the evolution of a wildfire incident and the status of the available response teams. For instance, it may suggest to assign additional teams to a specific area that is at high-risk (where “risk” will be defined below). The main goal is to perform an optimization process for the coverage of the area under consideration and the use of the available resources upon the data collected by the monitoring components. Hence, the proposed platform is aligned with the real needs maximizing the adopted mitigation actions.

The high-level diagram is depicted on Figure 54 and presents the general flow-chart of this Decision Support System task. The main inputs of the process are the wildfire spread projection, the population distribution, and the initial unit distribution, as well as additional parameters that may contribute to the impact of a specific area that may be affected by the approaching wildfire front (e.g. valuable infrastructure) and the properties of different types of fire-fighting units (e.g. capacity, cost of operation etc.). The output is a newly proposed unit distribution taking into account these inputs. Obviously, there are many parameters that need to be taken into account in this process raising the need to address the trade-offs when trying to optimize the unit resource allocation leading to a multi-objective optimization problem. Thus, the resource optimization process is a mathematical model developed for management of wildfires with the objective, first to minimize the number of people that may be exposed to risk (and above all life loss) in the high-risk areas and second, to minimize the total cost of fire containment and property losses. The model optimizes the resource allocation decisions given limited capacity and availability while considering the fire spread model and the population distribution.

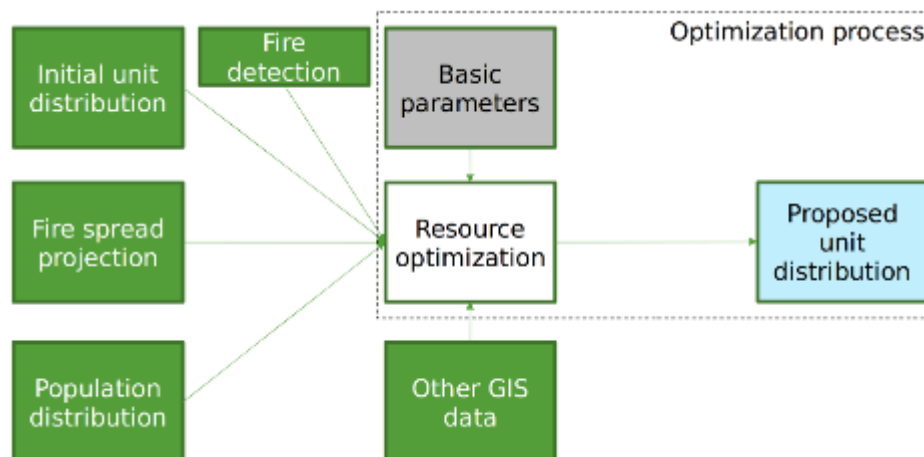


Figure 54 - Resource Allocation Diagram

4.1.1. Data Structures

The aforementioned inputs and outputs of the resource allocation model are obviously geospatial data. The two primary types of geospatial data are raster and vector data. Raster data is stored as a grid of values which are rendered on a map as pixels. Each pixel value represents an area on the Earth’s surface. The value of a pixel can be continuous or categorical. Vector data structures represent specific features on the Earth’s surface and assign attributes to those features.

In our case, we use raster data as depicted on Figure 55, because it includes the spatial information that connects the data to a particular location like the raster's extent and cell size, the number of rows and columns, and its coordinate reference system (or CRS).

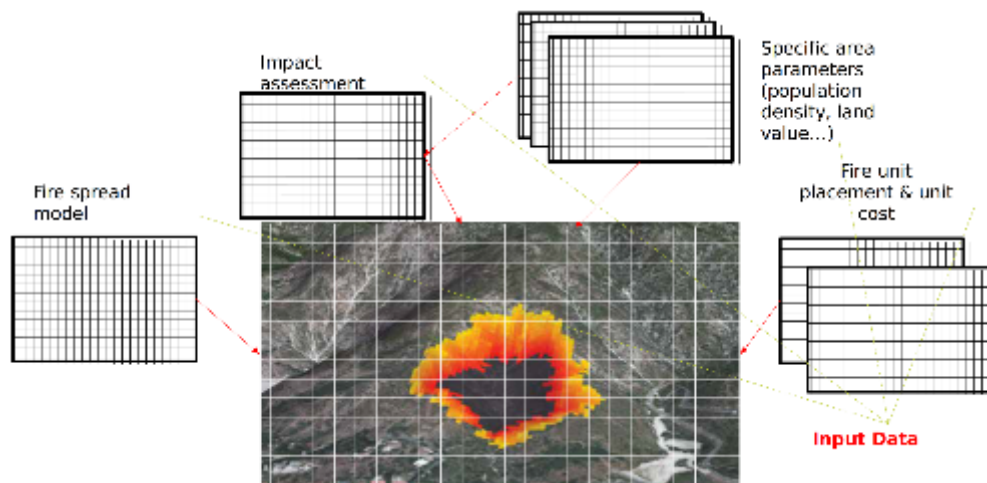


Figure 55 - Resource Allocation Input Raster

4.2. Software implementation and results

4.2.1. Population based geographical distribution of impact risk estimation

Following the above approach, and assuming that specific details about the land properties or the firefighting unit location and capabilities are not available (either not available in a usable format or not considered reliable to include in the DSS) a simple direct approach is to relate the information of the wildfire spread with the population density in a specific area in order to derive a first level information that can be georeferenced and visualized improving the situational awareness of decision makers. By combining these two types of input data, we can define what we call **criticality index** which represents an expected impact severity as a function of the fire intensity estimated to reach a specific area and the population density of the potentially affected area.

Specifically, for the extraction of a sample wildfire spread input, we used in a Proof of Concept (PoC) implementation based on FlamMap 6.2 which is a wildfire analysis desktop application that simulates potential fire behavior characteristics (spread rate, flame length, fire-arrival, etc.), fire growth and spread and conditional burn probabilities under constant environmental conditions (weather and fuel moisture). For the population density input, we are using the open spatial demographic data of WorldPop⁸, which provides different types of gridded population count datasets (Tatem, A, 2017).

The validation of our PoC implementation was done for an area of the Rocky Mountain in Boulder, Colorado in the US as shown in Figure 56. The area under examination extended in a rectangle of 13,500m by 14,760m and a grid of pixels-cells with a size of 90m was assumed.

⁸ <https://www.worldpop.org/>

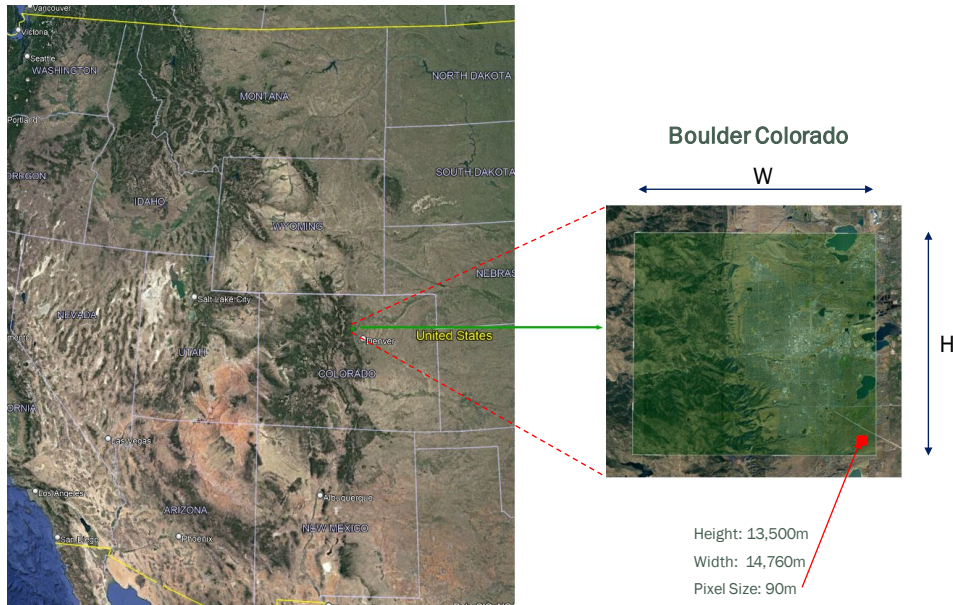


Figure 56 - PoC validation scenario

A wildfire ignition incident with an initial fire front as shown in Figure 57 was simulated on FlamMap and its extension over time was extrapolated against the inhabited surrounding areas in order to generate the criticality index per pixel-cell.

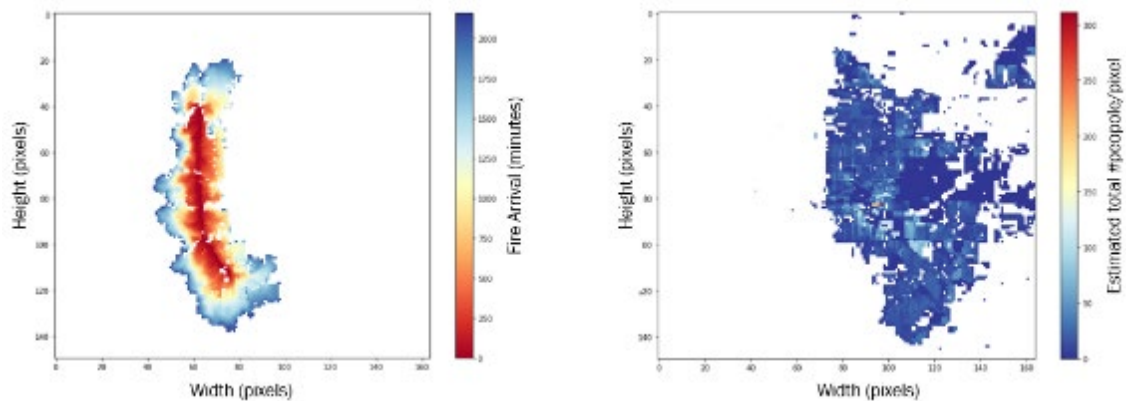


Figure 57 - PoC Example Inputs

In Figure 57, the wildfire arrival in minutes (left diagram) and the estimated total number of people (right diagram) are depicted per pixel – cell.

Regarding the criticality index calculation algorithm, initially, for every grid cell that according to FlamMap the wildfire will arrive in the next 36h, we calculate a population criticality index within a radius of approximately 500 meters. Then, by calculating a weighted average of the wildfire arrival criticality and the population criticality, a final criticality index between 1 - 6 is created for each pixel. When an update of the wildfire spread model is received, the algorithm will recalculate the critical areas. The algorithm and the criticality thresholds should be adjusted to the granularity (forecast time scales, geographical grid) and data types expected as input from the wildfire spread model, as well as to the time interval between wildfire spread model updates.

In our previous example from Figure 57, the wildfire arrival criticality Index is:

- Low: > 24h
- Medium: 12h - 24h

- High: < 12h

while the population criticality index is:

- Low: Uninhabited
- Medium: 0 - 10
- High: > 10

According to these thresholds, the algorithm calculates the final critical areas as depicted in Figure 58. From the below outputs we can conclude that as it was obvious, areas with higher population density and where the wildfire will arrive quickly in the next minutes are more critical than the others.

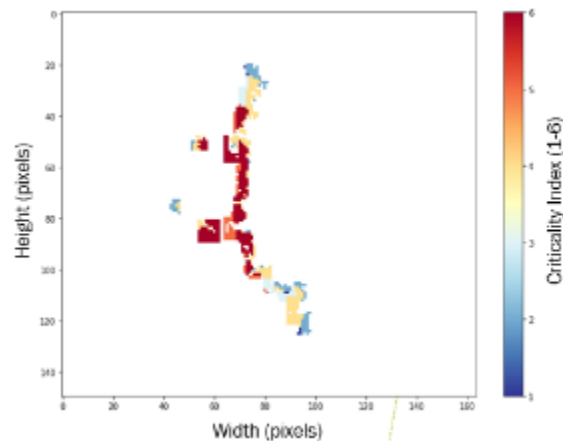


Figure 58 - PoC Example Output (per cell criticality index)

4.2.2. Multi-objective optimization based Resource Allocation (MORA)

The population based geographical distribution of impact risk estimation described above as already mentioned only provides high -level information to increase situational awareness of commanders and cannot lead to specific recommendations for actual Resource Allocation. However, as explained in section 4.1, when more data can be provided regarding the fire-fighting capacity and geographical distribution of firefighting units as well as additional land properties that should be taken into consideration for impact minimization (e.g. critical infrastructure), then more elaborate recommendations can be generated to solve a more complex optimization problem to improve the effectiveness of resource allocation and reduce the overall expected impact taking all the above inputs into consideration. We call this approach multi-objective optimization based Resource Allocation (MORA) and the operation and implementation of this methodology will be described in the following subsections.

4.2.2.1. Model Assumptions and Data Sources

We approach the spatial resource allocation problem as a constrained-optimization problem (COP). In mathematical optimization, constraint optimization is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables. In this case, there are three objective functions that need to be taken into account in the formulation of the overall cost minimization function:

- The total population in risk.
- The total land - property losses.
- The total cost of resource units

Obviously, the first objective function is of higher importance than the second and third objective function, which is considered in the algorithm.

Wildfire suppression is a process where different types of resource units become available at different time periods. Once a fire starts, it is assumed that it will reach a specific cell according to the wildfire spread model (fire arrival parameter). So, each cell has a fire arrival value when the wildfire is estimated to start burning this cell and becomes high-risk according to the wildfire spread model. In this stage, a detailed wildfire suppression plan is made for this time period. Decisions in this stage include identifying high-risk areas for allocating resources by their availability. Resource availability means some of the resources cannot join wildfire suppression until a specific period. For instance, non-local resources or aerial firefighting becomes available after some time.

The model assumes that all the resource allocation decisions are based on the assessment of wildfire condition so that more resources are allocated to fires that are substantial. The wildfire condition affects whether an area is determined as high-risk, completely *burned*, or *treated* (under the protection of units that have been dispatched to the area). By keeping track of fire conditions in each area (as provided by the fire front projection derived by the fire spread model), resource allocation and evaluation of property losses are determined.

In this model, geographical areas are segmented in *cells* to demonstrate the wildfire spread process. Cell sizes are associated with the fire spread rate and length of each period. In other words, the cell size in a fast-spreading fire scenario is larger than the cell size in a slow-spreading wildfire scenario. For different cell sizes in different scenarios, cell population density and cell land value are generated using publicly available data sets. In our case, the datasets of WorldPop research group are used for the geospatial data on population distributions.

At the beginning of a wildfire, according to the spread model, the initial conditions of cells, including the wildfire starting point are updated. These initial cell conditions are determined as:

- High-risk: The cell is about to be affected by the fire or has been partially on fire.
- Burnt: The cell has been fully burnt and cannot be treated.
- Treated: The cell is treated by response teams.

There are three prerequisites for a cell to become a new high-risk cell:

1. It should have at least one burnt cell neighbor.
2. It should not have been a burnt cell before.
3. It should have never been treated before.

In the following periods, suppression efforts based on resource allocation decisions will affect these conditions and result in an update. These updates are made to minimize the total population densities in high-risk cells and the total cost of resources and land values losses.

4.2.2.2. *Model formulation*

The model includes the following sets, parameters, decision variables, objective functions, and constraints:

Sets

A	Set of cells in consideration, indexed by a .
I	Set of resource types, indexed by i .
T	Set of discrete time periods, indexed by t .

Parameters

B	Budget for resource units (<i>optional</i>)
RS	Minimum required suppression rate for each cell (<i>chains/hour</i>)
DEN_a	Population Density at cell a .
VAL_a	Property and land value at cell a .
SUP_i	Suppression rate of resource i .
CST_i	Unit cost of resource i .
CAP_{it}	Capacity (availability) of resource i in period t .
ARV_{at}	Binary parameter defining whether cell a will be high-risk in period t .

Decision Variables

x_{iat}	Number of resources i sent to cell a in period t .
μ_{at}	Binary variable defining whether cell a becomes high-risk in period t .
k_{at}	Binary variable defining whether cell a becomes fully burnt in period t .
ϕ_{at}	Binary variable defining whether cell a starts to be treated in period t .

Objective Functions

$$MinObj_1 = \sum_{a \in A} \left(DEN_a \cdot \sum_{t \in T} \mu_{at} \right)$$

$$MinObj_2 = \sum_{a \in A} VAL_a \cdot \sum_{t \in T} k_{at}$$

$$MinObj_3 = \sum_{i \in I} \sum_{a \in A} \sum_{t \in T} x_{iat} \cdot CST_i$$

Obj_1 :	Total number of people in fire risk.
Obj_2 :	Total land - property losses.
Obj_3 :	Total cost of resource units.

Constraints

1) Total budget for resources must not be exceeded.

$$\sum_{i \in I} \sum_{a \in A} \sum_{t \in T} x_{iat} \cdot CST_i \leq B$$

2) For each period and for each resource type the total number of resources must not exceed the capacity (availability).

$$\sum_{a \in A} x_{iat} \leq CAP_{it} \forall i \in I, \forall t \in T$$

3) All resources sent to each cell are sufficient to contain the fire in this cell.

$$\sum_{i \in I} x_{iat} \cdot SUP_i \geq \phi_{at} \cdot RS \forall a \in A, \forall t \in T$$

4) Only high-risk cells will be treated.

$$\phi_{at} \leq \mu_{at} \forall a \in A, \forall t \in T$$

5) If a cell is a high-risk in t-1 and is not treated in t-1, then it will become a fully burnt cell in t.

$$k_{at} \leq \mu_{a,t-1} - \phi_{a,t-1} \forall a \in A, \forall t \in (1, T)$$

6) Each cell can become a high-risk cell, a burnt cell, a treated cell at most once.

$$\sum_{t \in T} \mu_{at} \leq 1, \sum_{t \in T} k_{at} \leq 1, \sum_{t \in T} \phi_{at} \leq 1 \forall a \in A, \forall t \in T$$

7) Each burnt cell or treated cell needs to be a high-risk previously.

$$\sum_{t \in T} k_{at} \leq \sum_{t \in T} \mu_{at}, \sum_{t \in T} \phi_{at} \leq \sum_{t \in T} \mu_{at} \forall a \in A$$

8) If a cell has fully burnt, it would never be treated in the future and the opposite.

$$\sum_{t \in T} k_{at} + \sum_{t \in T} \phi_{at} \leq 1 \forall a \in A$$

9) Each high-risk cell must become a treated cell or a burnt cell in the end.

$$\sum_{a \in A} \sum_{t \in T} \phi_{at} + \sum_{a \in A} \sum_{t \in T} k_{at} \leq \sum_{a \in A} \sum_{t \in T} \mu_{at}$$

10) A cell to become high-risk must have at least one fully burnt cell neighbor.

$$10 \cdot \mu_{at} \geq ARV_{at} \cdot \sum_{a' \in Neighbors(a)} k_{a't} \forall a \in A, \forall t \in (1, T)$$

$$10 \cdot (1 - \mu_{at}) \geq 1 - \left(ARV_{at} \cdot \sum_{a' \in Neighbors(a)} k_{a't} \right) \forall a \in A, \forall t \in (1, T)$$

11) Nonnegativity of number of resources.

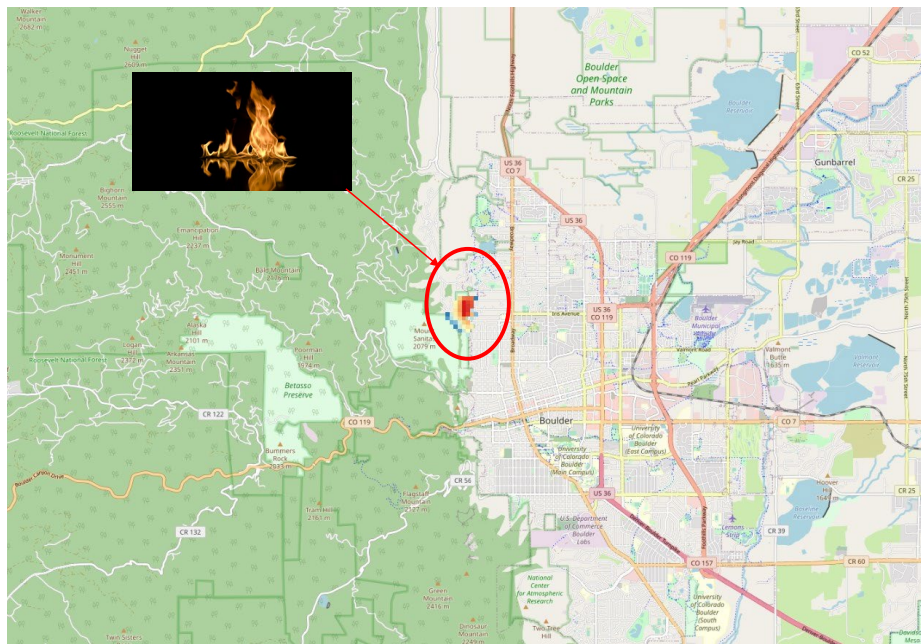
$$x_{iat} \geq 0 \forall i \in I, \forall a \in A, \forall t \in T$$

12) Binary variables limitations.

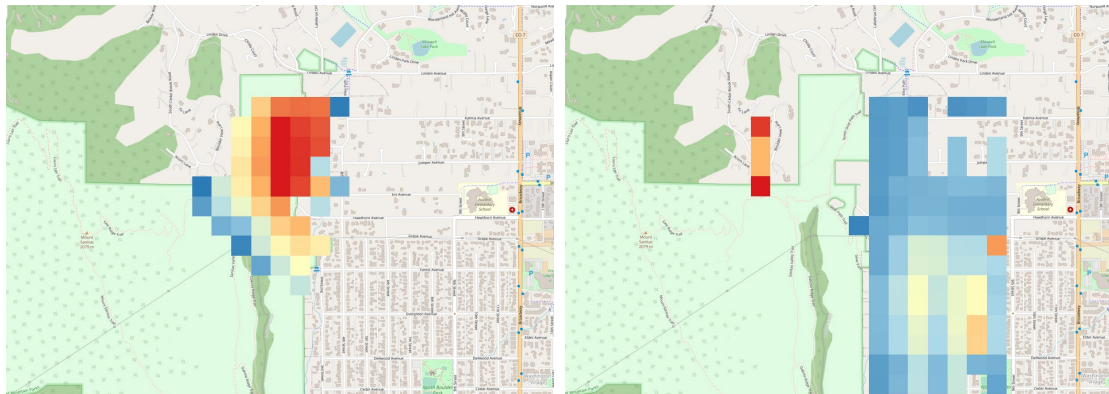
$$\mu_{at}, k_{at}, \phi_{at} \in \{0,1\} \forall a \in A, \forall t \in T$$

4.2.2.3. Model validation and demonstration

The validation of the MORA implementation was done as in the case presented in section 4.2.1 again for an area of the Rocky Mountain in Boulder, Colorado in the US as shown in Figure 59. The area under examination in this case extended in a rectangle of 1,710m by 1,350m and a grid of pixels-cells with a size of 90m was assumed.



(a)



(b)

(c)

Figure 59 – MORA validation scenario (a) area under examination (b) fire spread model (c) population distribution

A simple scenario for an area of 5x5 cells is used to demonstrate the model results over three time periods. Each period has a length of one hour. In this experiment, the wildfire spread rate is assumed to be stable per hour. We assume that the spread rate does not change over time, so that cell size stays the same in each period. Thus, wildfire arrivals and population densities are fixed in each cell. Cell indexes, wildfire arrivals from the spread model and population densities are shown in Figure 60. According to fire arrivals, the fire starts at cells with index 6,7,11 and 12.

0	1	2	3	4				1	2	0	0	0	2	3
5	6	7	8	9		0	0	1	2	0	0	0	4	5
10	11	12	13	14		0	0	1	2	0	0	2	3	6
15	16	17	18	19		1	1	1	2	0	2	3	4	7
20	21	22	23	24		2	2	2	2	0	2	4	7	7

(a)

(b)

(c)

Figure 60 – Cell parameters for the area under evaluation: (a) cell indexes (b) fire arrival time per cell (in multiples of the basic period) (c) population densities

In this simple example, we assume that there are three different resource types of response teams (A, B, C) with different parameters as shown in Figure 61. The minimum suppression rate to contain the fire in each cell is 10, while the total budget is 100.

Resource Type	Suppression Rate	Cost	Capacity
A	2	5	t=0 : 4 t=1 : 4 t=2 : 4
B	5	15	t=0 : 3 t=1 : 2 t=2 : 4
C	10	20	t=0 : 0 t=1 : 0 t=2 : 1

Figure 61 - Resource Types Example

The MORA algorithm will try to allocate the available response teams in the field for each period in order to minimize the total population in high-risk cells and the total cost of resources and land-property value losses. To simplify the process in this test example, it was assumed that the land value is the same and equal to one throughout the whole area. This

means that the land value losses are equal to burnt areas. The optimal solution is expected to lead to the minimum number of burnt areas (cells).

The optimal solution for this simple constrained optimization problem is depicted in the Figure 62 below. High-risk cells are shown with red color, treated cells with grey color and burnt cells with black color. In $t=0$, MORA recommends to allocate to cell with index 7 two units from type B and to cell with index 12 three units from type A and one unit from type B. There are no available resources to contain the fire in the other two high-risk cells (index 6 and 11) in this scenario (an inevitable case in most realistic scenarios). So, in $t=1$ these cells are going to be fully burnt and the fire continues to the neighbor cells. According to the availability of resources, two units from type B are proposed to be allocated to cell with index 17. So, in $t=2$ cell with index 16 will be fully burnt and cells with indexes 21 and 22 will become high-risk. Now, there are available resources to be allocated to these high-risk cells to contain the fire. Specifically, two units from type B in cell 21 and one unit from type C in cell 22 suffice to suppress the fire in these cells. One unit from type C is preferred than two units from type B because it costs less.

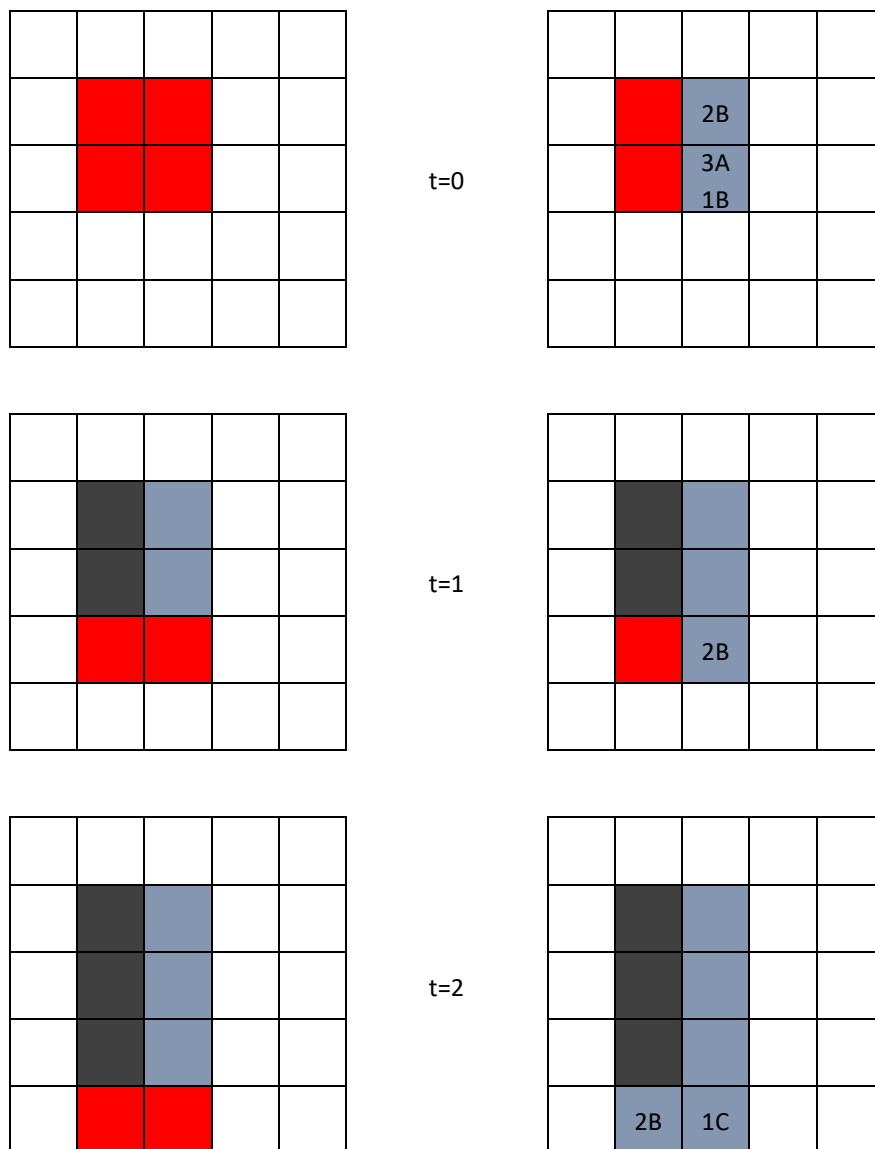


Figure 62 - Resource Allocation Optimal Solution

In conclusion, the MORA recommends the allocation of available resources of response teams in specific cells to minimize the total population in risk and the total burnt areas. It directs the wildfire only to the bottom cells and not to the right cells where the population distribution is denser. The result of this simple simulation is depicted in Figure 63, where the left table shows the fire spread according to fire arrival (t=0 red, t=1 orange, t=2 yellow) and the right table shows the area after the treatment of the response teams (burnt areas with black and treated areas with grey).

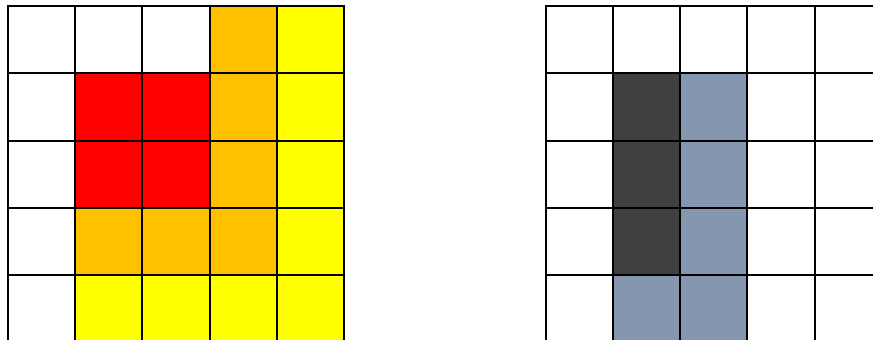


Figure 63 – Input and Output of Resource Allocation DSS

4.3. Component integration

In the context of application integration, the Storage Abstraction Layer (SAL) incorporates our system into its operations. This integration is facilitated by utilizing SAL's message broker to consume data pushed by other components. The data our application pulls from the SAL is the wildfire spread model and specifically the wildfire arrival time raster. Following this, the resource allocation algorithm makes a transformation preprocess and fetches the appropriate open spatial demographic data of WorldPop, as mentioned in section 4.2.1. The results of the algorithm can be easily stored in the SAL too. To ensure ease of deployment, portability, scalability, the resource allocation algorithm is deployed as a Docker container. The containerized integration promises the consistency that the algorithm will run similarly on any system or environment. The resource allocation component will also be deployed in the Silvanus Kubernetes cloud platform for testing and validation. Additional in-depth documentation about the integration process will be also available on the Silvanus GitHub repository: <https://github.com/silvanus-prj/resource-allocation>.

4.4. Plans for future extensions

One of the most important future extensions that needs to be implemented is the calculation of the availability of resources based on their current location. By now, the availability of resources of response teams is assumed to be static and known beforehand; thus, it is not calculated by the MORA algorithm.

Furthermore, the multi-objective optimization algorithm can be extended to incorporate more parameters in its cost function. For instance, land-property value can be retrieved through publicly available datasets and GIS data repositories. For the initial model implementation, it is assumed static and equal to one to avoid biasing the areas.

Finally, an extension that can be implemented in the future is to add some constraints of resource types of response teams in specific locations. For example, in a canyon, the ground

firefighting resources cannot reach the fire, or the aerial firefighting resources cannot drop water too close to houses.

The above extensions are going to be assessed and the selected extensions will be implemented in the final version of the resource allocation of response teams' MORA algorithm.

5. Stakeholder notification decision on fire incidents using multilingual textual framework

Multilingual social media sensing plays a vital role in detecting wildfires, as it allows for a comprehensive understanding of online discussions and their potential risks. With the inclusion of languages such as Indonesian, English, Italian, Spanish, and Slovak, social media platforms can monitor and analyse content in real-time, spanning a wide range of global communities. By employing natural language processing and Machine Learning techniques, algorithms can identify keywords, sentiments, and geographical references related to wildfire hazard. This multilingual approach enables efficient and timely identification of dangerous situations, allowing authorities and relevant organizations to respond swiftly and mitigate the risk of wildfires. Through multilingual social media sensing, the world can proactively prevent and combat this destructive environmental threat.

These systems can extract geolocation information, keywords, and context from social media posts, allowing stakeholders to quickly pinpoint the exact location of the fire. When the system detects the number of posts related to wildfire in a specific location (threshold number being configurable during system initialization), a notification will be sent to the stakeholder.

Stream data from social media is classified into “fire” or “not fire” in the first phase. If the data is fire, then, the system continue to detect the location based on the neural language processing approach. The system supports multi language for wildfire and location detection. The labelled social data for each language should be trained to make robust models for classification and named entity (location detection model).

Information on wildfire locations based on social media data needs to be conveyed quickly to stakeholders. When the text data sent by netizens exceeds the set threshold, a notification needs to be sent. Notifications must be sent to people who really need the information. Too many notifications that are not of interest to the user will make the person not care and not be sensitive to what the system informs them about. Therefore, it is necessary to decide who should receive the notification.

5.1. Concept of operation

This framework employed Twitter data to train the model in several languages. The pipeline of the framework started from collected data related to forest fires in Indonesian (Id), English (En), Italian (It), Spanish (Es), and Slovak (Sk). We annotated relevant datasets with three labels: language types, fire or not, and location and time entities. Social media sensing framework is shown in Figure 64.

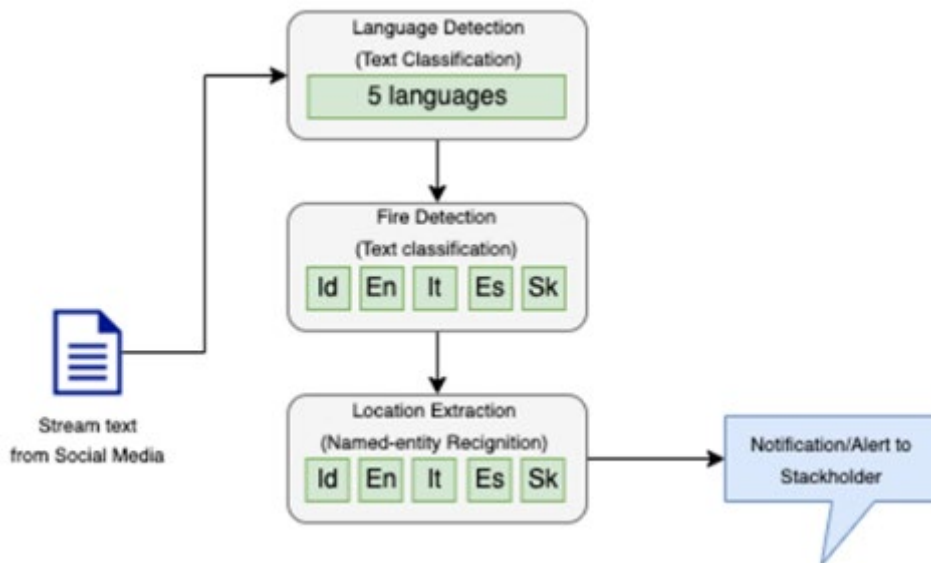


Figure 64: Framework of Social media sensing

5.2. Software implementation and results

5.2.1. Datasets

In the context of social sensing, datasets enable the detection of emerging issues, sentiment analysis, and the identification of patterns and trends. Additionally, these datasets facilitate the development and validation of predictive models, aiding in early warning systems for potential disasters like forest fires. As social media continues to be a prominent source of real-time information, the utilization of datasets for media social sensing becomes an indispensable tool for understanding the world and making informed decisions.

We used several datasets for language detection, wildfire detection and location extraction in 5 languages. Table 6 describes the source of dataset we used for each language detection model.

Table 6: Datasets for language detection models

Models	Languages	Source
Language Detection	Indonesian	Collecting data from Twitter
	English	Provided by partner (CERTH)
	Italian	Provided by partner (CERTH)
	Spanish	Provided by partner (CERTH)
	Slovak	Provided by partner (UISAV)
Fire detection	Indonesian	Collecting data from Twitter
	Italian	Provided by partner (CERTH)
	Spanish	Provided by partner (CERTH)
Location extraction	Indonesian	Web scraping from forest-fire articles

1. Indonesian Dataset

We have two datasets for different aims. First, a dataset for classification (Twitter collection) and a dataset for location extraction (article scraping).

We have collected data from Twitter using the Tweepy library, starting from June 22, 2022, to July 28, 2022. From this work, we got 1048 rows of data related to forest fires in Indonesia. We used several keywords, such as kabutasap, kabakaranhutan, karhutla, sesaknafas, daruratasap. The dataset is available online in this following url, <https://github.com/ariflaksito/forest-fire-dataset/>

Then we did the pre-processing phase to make the data compatible for the following phase. We used Python libraries to perform the following pre-processing tasks:

- a) Regular expression: remove unused character, lower-case, remove hashtag, username and retweet text.
- b) Sastrawi: remove unused word in Indonesian languages and stemming word.

The snippet code below shows the used of Sastrawi and Regular expression for cleaning the data.

Manual labelling was conducted by three participants to classify the text into six classes. According to the labelled process, there are 136 data labelled for forest fire, 458 data for prevention, 54 data for rehabilitation, 127 data for mitigation, 164 data for countermeasures, and 90 remainders of the data are labelled as unknown.

Figure 65 and Figure 66 show the number of data items for each class and the distribution of terms in the dataset. The dataset has 13,167 words in total. Each tweet has an average of 12.57 words, with a maximum word length of 36.

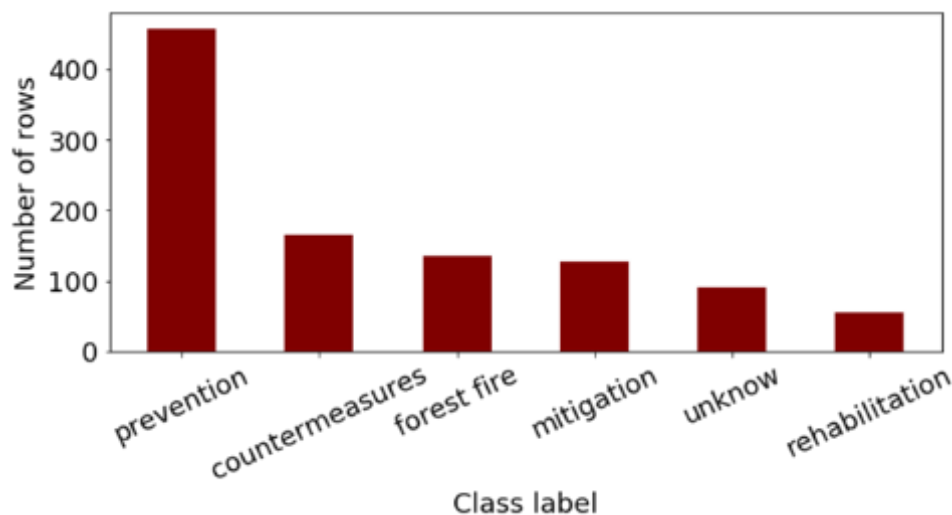


Figure 65: Data distribution each class

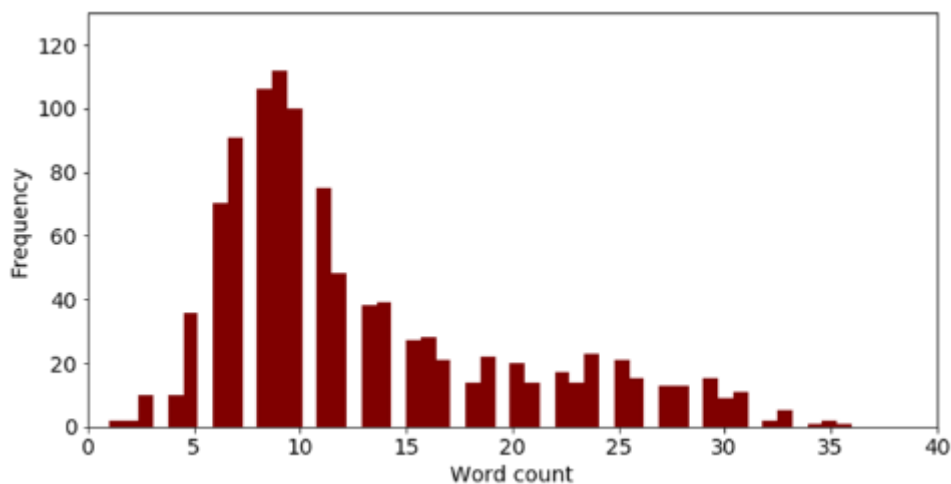


Figure 66: Word length distribution

Due to the requirement of building fire prediction for each language we are using 2 classes (fire and not-fire), so we re-train the model using 2 classes by modifying the class label in Indonesian dataset.

We modified 6 classes into 2 classes by merging labels that is not wildfire (prevention, countermeasure, mitigation, unknow, rehabilitation) in label 0 (not-fire) and wildfire label in label 1 (fire). As result, the number of data returned for the fire class is 136 and for not-fire is 911.

b. Indonesian NER Dataset

The second dataset aims to extract location in the Indonesian language. We collected 96 articles in Indonesia news and labelled using BioTagging.

BioTagging, short for "Beginning-Inside-Outside Tagging," is a critical concept in the field of Named Entity Recognition (NER), a subtask of Natural Language Processing (NLP). NER involves identifying and classifying named entities, such as names of persons, organisations, locations, dates, and more, within a text. BioTagging is a labelling scheme used to annotate tokens (words or characters) in a text corpus to indicate their positions and roles within named entities.

We obtained 1,103 sentences and 19,844 words from the Indonesian articles. We labelled the term in the first phase using five different types: place, time, organisation, quantity, and person. Location appears in 1075 words (B-geo and I-geo), time in 611 words (B-tim, I-tim), organisation in 790 words (B-org, I-org), quantity in 670 words (B-qty, I-qty), and person in 387 words (B-per, I-per). Table 7 describes the number of BioTagging word in the dataset.

Table 7: The number of word in the BioTagging label

Entity	BioTagging	Number of words
Location	B-geo	703
	I-geo	372
Time	B-tim	341
	I-tim	270
Organisation	B-org	303

	I-org	487
Quantity	B-qty	286
	I-qty	384
Person	B-per	237
	I-per	150

2. English Dataset

The English dataset was provided by CERTH in a .csv file format. The data specifically is used for named-entity recognition. The number of sentences is 366 with a total of 5597 words. In this dataset, the entity label is separated into place, organisation, person and other. Location appears in 414 words, organisation in 34 words, person in 44 words, and others in 122 words (Figure 67).

Unnamed: 0	Sentence	Word	Tag
0	0	1 Millions	O
1	1	1 under	O
2	2	1 flood	O
3	3	1 threat	O
4	4	1 in	O
5	5	1 US	B-geo
6	6	1 states	O
7	7	1 https://t.co/tu8YiANAxN	O
8	8	2 The	O
9	9	2 Arctic	B-geo
10	10	2 is	O
11	11	2 melting	O
12	12	2 —	O
13	13	2 and	O
14	14	2 South	B-geo
15	15	2 Florida	I-geo

Figure 67: NER English dataset preview

3. Italian Dataset

The Italian dataset was provided by CERTH in a .csv file format. The data is available for forest-fire prediction in Italian. We did the pre-processing phase to make the data compatible for the following phase. We used Python libraries for pre-processing, as follows:

- Regular expression: remove unused character, lower-case, remove hashtag, username and retweet text.
- Natural Language Toolkit (NLTK): remove unused word in Italian languages and stemming word.

The snippet code below shows the use of NLTK and Regular expression for cleaning the data.

```

import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

# Define a function to perform text pre-processing
def text_preprocessing(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove URLs, user mentions, and hashtags
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\S+', '', text)
    text = re.sub(r'#\S+', '', text)

    # Remove punctuation and special characters
    text = re.sub('[^a-záéíóúñ]+', ' ', text)

    # Tokenize the text into words
    words = nltk.word_tokenize(text)

    # Remove stopwords
    stop_words = stopwords.words('italian')
    words = [word for word in words if word not in stop_words]

    # Stem the words using a Snowball stemmer
    stemmer = SnowballStemmer('italian')
    words = [stemmer.stem(word) for word in words]

    # Join the words back into a string
    text = ' '.join(words)

    return text

```

Labels are already available on datasets from third parties where the dataset is divided into two groups. In this dataset there are 1000 data labelled False and 526 data labelled True.

Figure 68 and Figure 69 show the number of data items for each class and the distribution of terms in the dataset. The dataset has a total of 22,019 words. Each tweet has an average of 15.44 words, with a maximum word length of 41.

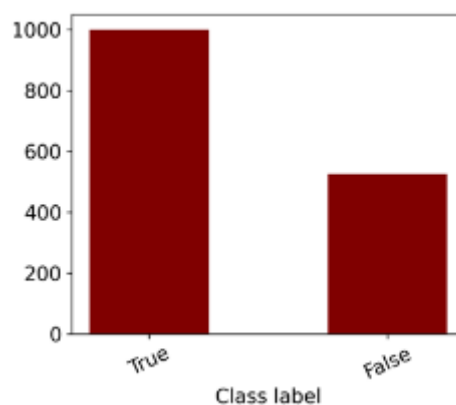


Figure 68: Data distribution each class

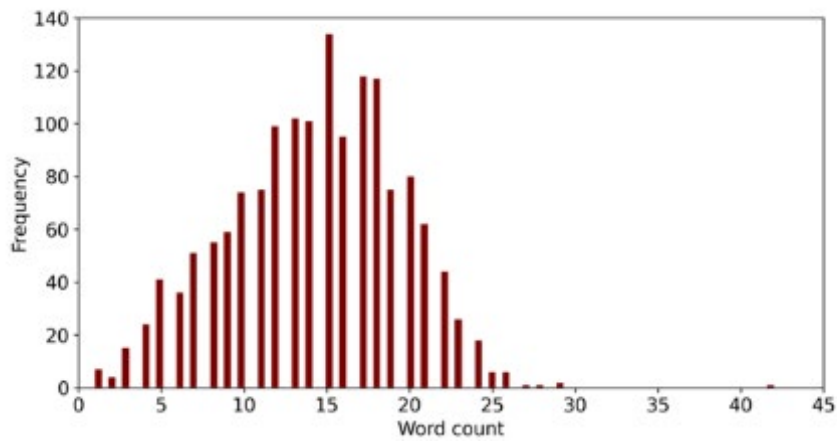


Figure 69: Word length distribution

4. Spanish Dataset

The Spanish dataset provided by CERTH in file csv. We did the pre-processing phase to make the data compatible for the following phase. We used Python libraries for pre-processing.

a) Regular expression: remove unused character, lower-case, remove hashtag, username and retweet text.

b) NLTK: remove unused word in Spanish languages and stemming word.

The snippet code below shows the used of NLTK and Regular expression for cleaning the data.


```

import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

# Define a function to perform text pre-processing
def text_preprocessing(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove URLs, user mentions, and hashtags
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'\@S+', '', text)
    text = re.sub(r'#\S+', '', text)

    # Remove punctuation and special characters
    text = re.sub('[^a-záéíóúñ]+', '', text)

    # Tokenize the text into words
    words = nltk.word_tokenize(text)

    # Remove stopwords
    stop_words = stopwords.words('italian')
    words = [word for word in words if word not in stop_words]

    # Stem the words using a Snowball stemmer
    stemmer = SnowballStemmer('spanish')
    words = [stemmer.stem(word) for word in words]

    # Join the words back into a string
    text = ' '.join(words)

    return text

```

Labels are already available on datasets from third parties where the dataset is divided into two groups. In this dataset there are 1618 data labelled False and 983 data labelled True.

Figure 70 and Figure 71 show the number of data items for each class and the distribution of terms in the dataset. The dataset has a total of 51,833 words. Each tweet has an average of 20.18 words, with a maximum word length of 55.

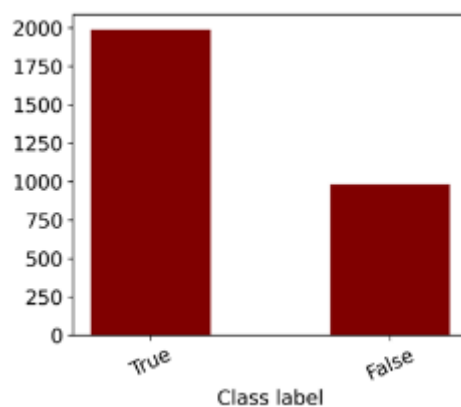


Figure 70: Data distribution each class

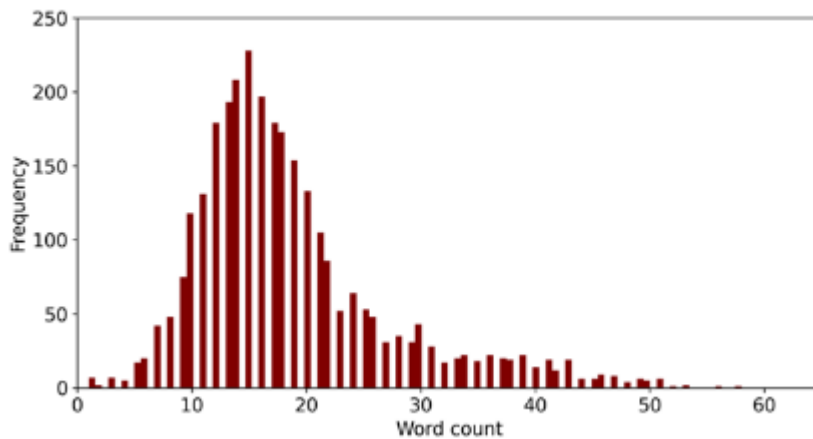


Figure 71: Word length distribution

5.2.2. Training Model

We employed Machine Learning and deep learning for training stages. In the experiments, we used a consumer PC equipped with an Intel i5 (Gen 12) CPU, 16 GB of RAM and RTX2060 GPU.

1. Language Detection

From the five datasets in sub-section 5.2.1, we merged them into a dataset and labelled them based on the language. Figure 72 shows the language distribution of each dataset. The amount of Spanish data is 2568, Italian is 1426, Indonesian is 1047, Slovak is 573 and English is 366. The class seems imbalanced, so we should make it balanced.

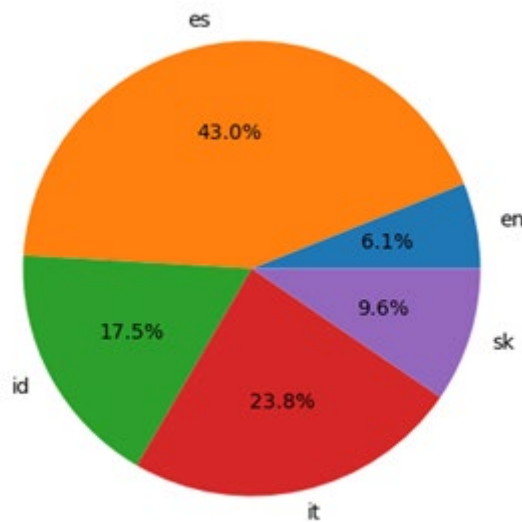


Figure 72: Language datasets distribution

The next process is splitting the dataset to two categories (training data, test data). We used the ratio of 80% training data while 20% of test data. Table 8 depicts the size of the training and test data sets for each class.

Table 8: Size of training and test datasets for each language

Languages	Train data	Test data
Spanish	2054	514
Italian	1141	285
Indonesian	838	209
Slovak	458	115
English	293	73

To deal with the imbalance of the above data, we used the Random Over Sampling technique (ROS) where samples from the minority class are duplicated to balance class distribution. It artificially increases the number of minority class samples, reducing class imbalance. However, this method may lead to overfitting and introduce bias, as it duplicates existing samples without considering the underlying data distribution. After the data balancing technique, the amount of data for each class becomes 2054.

We used Machine Learning algorithms to train the language classification data. Then, we evaluated the classifier models using cross validation using 5 folds. There were 3 Machine Learning algorithms used in this experiment, namely, Naïve bayes, Decision Tree and K-Nearest neighbour.

a. Naïve Bayes

The Naïve Bayes method is a supervised learning technique that depends on conditional probability to apply Bayes' theorem. This algorithm is well-suited for sentiment analysis and text categorization, particularly on social media platforms such as Twitter. The following equation depicts the computation of category probability in the Nave Bayes algorithm:

$$P(Y|X) = \frac{P(X | Y) \times P(Y)}{P(X)} \quad (5.1)$$

b. Decision Tree

Decision tree is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

c. K-Nearest neighbor

K-Nearest neighbour algorithm, a Machine Learning algorithm utilized for regression and classification tasks. It is a straightforward method that stores all existing cases and classifies

new ones by relying on its k neighbours' majority vote. The case is assigned to the class most prevalent among its k nearest neighbours, which is determined through a distance function. One of the distance functions in the KNN algorithm is the Euclidean distance, as formulated in Equation below:

$$d_i = \sqrt{\sum_{i=1}^p (x_{2i} - x_{1i})^2} \quad (5.2)$$

For metrics evaluation, we used the classification report to measure those Machine Learning models. This report offers a comprehensive overview of the performance and effectiveness of language classification models. By assessing metrics such as precision, recall, F1-score, and support for each language class, the classification report helps gauge the accuracy of the model's predictions across different languages. Precision represents the ratio of correctly classified instances within a specific language, while recall measures the proportion of correctly identified instances relative to the actual instances of that language. The F1-score combines both precision and recall, providing a balanced assessment of the model's overall performance. Additionally, the support metric indicates the number of instances present for each language, aiding in identifying potential biases in the dataset. With insights garnered from the classification report, researchers and practitioners can fine-tune their models, address language-specific challenges, and enhance the accuracy of multi-language detection systems powered by Machine Learning algorithms.

Using the three algorithms presented above, Naïve Bayes achieved the best performance for multi-language classification in precision, recall, F1-score and accuracy. The comparison can be seen in Table 9.

Table 9: Comparison performance of models

Model	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	0.99	0.98	0.98	0.98
Decision Tree	0.96	0.94	0.95	0.95
KNN	0.92	0.90	0.91	0.89

From the results above, we built the Naïve bayes model using pickle version 4.0 and save this model to the .sav file.

2. Wildfire Detection

Based on the availability of the dataset, we build 3 models for wildfire detection in Indonesian, Spanish and Italian.

a. Indonesia wildfire detection model

We used the Indonesian dataset with 6 classes for wildfire detection. We also designed a sequential neural network called GRU and LSTM to assess text classification, which was enhanced from an earlier Recurrent Neural Network (RNN). Furthermore, a bidirectional technique was used to compare the performance of both models. Moreover, we examined how the augmented dataset affects the performance of the deep-learning text classifier.

The Indonesian dataset class distribution is not balanced. A challenge with proportionate class sizes in a dataset differing substantially from one another is called imbalanced data

classification. A simple and well-liked set of methods for balancing the class distributions of the training data is sampling methods. By using one of the methods, between oversampling and under sampling, the original data space is balanced. The most common method, oversampling, uses a method called Synthetic Minority Oversampling Technique (SMOTE) to introduce false samples into the data space. In 2002, this technique was first introduced. Random oversampling (ROS) is another technique for handling unbalanced data. The training dataset is oversampled at random by adding duplicate cases from the minority class.

The most popular and successful methods to lessen the disappearing and exploding gradient effects is the Long Short-Term Memory (LSTM) idea. By using gates to control the inputs and outputs, this technique transforms the "sigmoid" or "tanh" hidden unit structure into a memory cell. The hidden state portion of this model resembles a recurrent network employing cell memory, as seen in Figure 73. Gates for input, forget, output, and cell activation make up a cell memory.

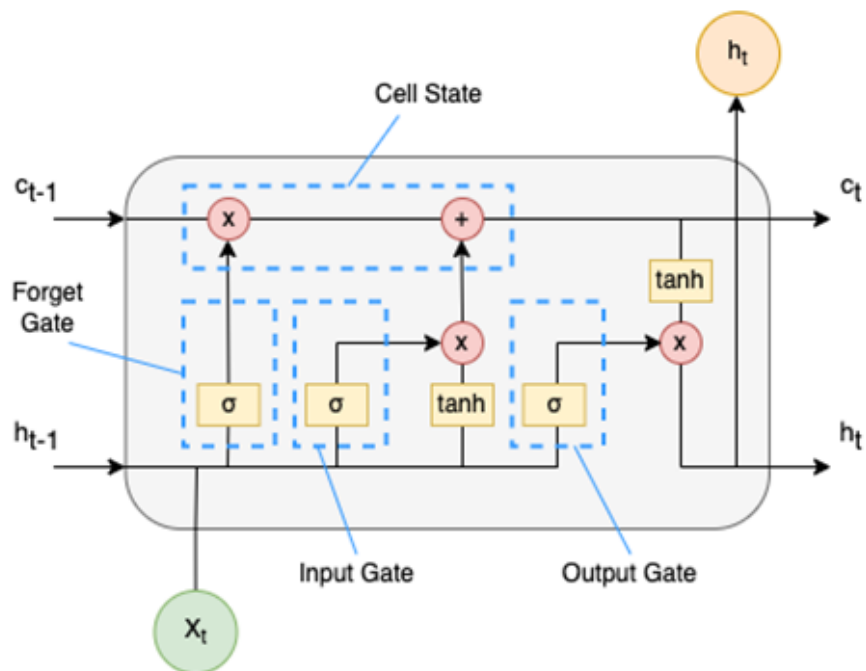


Figure 73: Memory cell of LSTM

Similar in construction to the LSTM, the Gated Recurrent Units (GRU) streamline the architecture by integrating memory and hidden state into a single state vector. The "reset" gate and the "update" gate are two further gates that the GRU employs to manage information flow. The update gate specifies how much of the new state should be remembered, whereas the reset gate decides how much of the prior state should be forgotten. The GRU can update and maintain information from the previous time step using these gates. The unit cell of the GRU is seen in Figure 74.

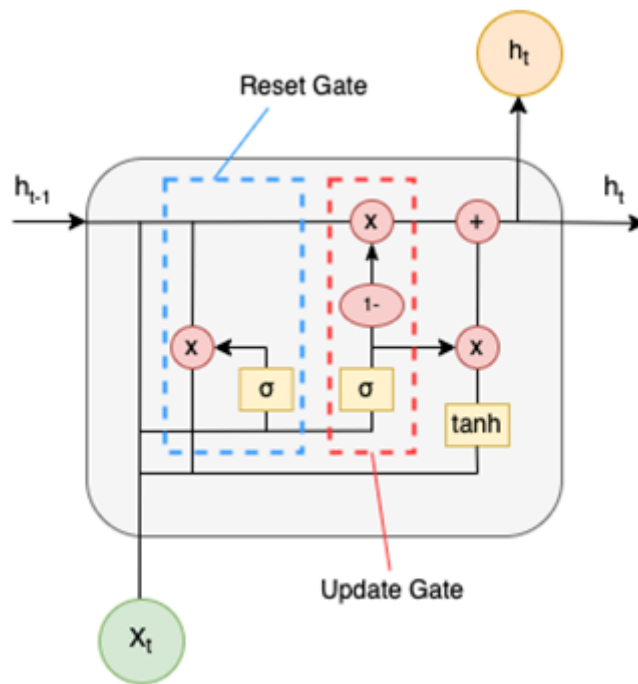


Figure 74: Memory cell of GRU

Traditional RNNs simply take into account the data's prior context during training. While many applications, like voice recognition, just need to look at the prior context, it is also helpful to investigate the future context. Figure 75 shows the structure of Bidirectional RNNs in the unfolded type. As can be seen that output from forward states are not connected to input of backward states, and vice versa.

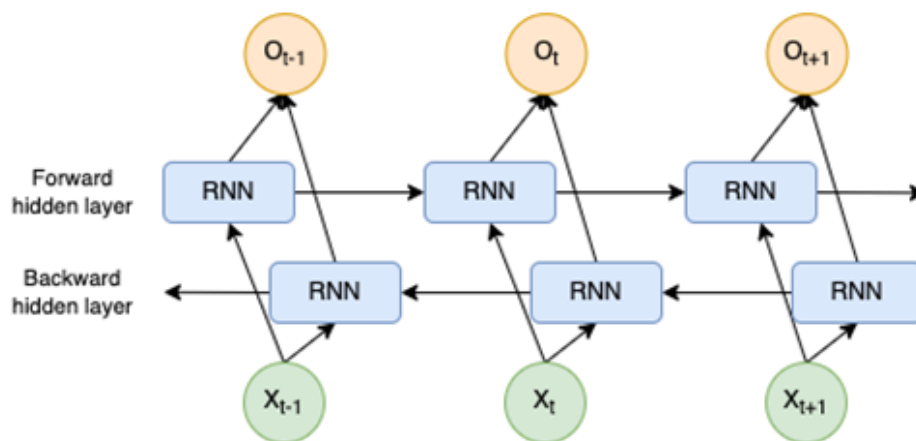


Figure 75: Bidirectional RNNs structure

To get the best model, we compared augmented techniques in several classification models. We evaluated datasets that were not enhanced, datasets that were enhanced using the SMOTE methodology, and datasets that were oversampled randomly. Moreover, as model classifier we used the GRU, LSTM, and the Bidirectional techniques of both models. In this experiment, we used a single-layer architecture for all models, and we trained the model using 100 epochs.

The model's input is text vector array with maximum length of 30 and embedding size 300. The model architecture we used employed 64 units with a dropout of 0.5 followed by a dense

layer and final layer using the softmax activation function for multiclass classification. Then the models are compiled using Adam optimizer with a learning rate of 0.001. We divided the data into training and testing with a comparison of 80% of the training data and 20% of the testing data in order to evaluate the performance of each model using different augmentation methods. Table 10 shows the result of training and testing accuracy for each model and augmented method.

Table 10: Result of Training and Testing Accuracy

Model	Aug.	Testing Acc.	Train Acc.
GRU	Original	0.5043 ± 0.0203	0.8943
	ROS	0.8973 ± 0.0105	0.9754
	SMOTE	0.3282 ± 0.0189	0.8603
LSTM	Original	0.4772 ± 0.0140	0.8870
	ROS	0.8964 ± 0.0117	0.9786
	SMOTE	0.3167 ± 0.0131	0.8617
Bidirectional GRU	Original	0.5063 ± 0.0213	0.8955
	ROS	0.8970 ± 0.0120	0.9804
	SMOTE	0.3173 ± 0.0154	0.8658
Bidirectional LSTM	Original	0.4947 ± 0.0135	0.8967
	ROS	0.8995 ± 0.0119	0.9809
	SMOTE	0.3256 ± 0.0195	0.8644

The SMOTE approach generally resulted in low accuracy for all models. Comparatively, the use of Random oversampling significantly increases the performance of all classifier models. It is proven that the SMOTE-generated synthetic samples could not accurately reflect the minority class in the dataset. Consequently, the accuracy score of training and validation is significantly different, which shows that the models suffer from overfitting. Contrary to the SMOTE method, it is clearly that the ROS method can handle unbalanced data. As can be seen in Table 10, Bidirectional LSTM achieves the best performance compared to other models.

Since each language requires two classes (fire and not-fire), we re-train the model with two classes by modifying the class label in the Indonesian dataset, as described in sub-section 5.2.1 point 1.

We used Bidirectional LSTM and Random Over Sampling method to deal with imbalanced class in the dataset. The input to the model is a text vector array with a maximum length of 30 and an embedding size of 100. For binary classification, we utilised 30 units with a dropout of 0.5, followed by a dense layer and a final layer using the sigmoid activation function. The models are then constructed using the Adam optimizer with a learning rate of 0.01. We separated the data into training and testing and compared 75% of the training data and 25% of the testing data. Finally, we train the model using 100 epochs.

The model performance for wildfire prediction was not satisfied with the F1-score for fire prediction was 0.00 and the f1-score for non-fire prediction was 0.93. Figure 76 depicts the result of the classification report.

Classification Report				
	precision	recall	f1-score	support
0	0.87	1.00	0.93	228
1	1.00	0.00	0.00	34
accuracy			0.87	262
macro avg	0.94	0.50	0.47	262
weighted avg	0.89	0.87	0.81	262

Figure 76: Bidirectional LSTM classification report

To address the aforementioned issue, we employ Machine Learning algorithms for fire prediction in Indonesian. Four Machine Learning methods were used: Nave Bayes, K-Nearest Neighbour, Decision Tree, and Random Forest. Table 11 shows the results of those experiments.

Table 11: Comparison performance of ML algorithms

Algorithm	F1-Score		Accuracy
	Not Fire	Fire	
Naïve Bayes	Not Fire	0.76	0.64
	Fire	0.21	
K-Nearest Neighbour	Not Fire	0.77	0.65
	Fire	0.23	
Decision Tree (neighbours = 5)	Not Fire	0.90	0.82
	Fire	0.34	
Random Forest (estimators = 100)	Not Fire	0.94	0.89
	Fire	0.44	

It seems that Random Forest is better for Fire prediction compared to Bidirectional LSTM and other Machine Learning methods. From the results above, we built the Random Forest model using pickle version 4.0 and save this model to the .sav file.

b. Spanish fire detection model

To train the model prediction for Spanish fire detection, we used Bidirectional LSTM. After pre-processing the data, we separated it into 80:20 train and test datasets. The model was then trained over 200 epochs using the single-layer architecture of Bidirectional LSTM.

The model takes a text vector array with a maximum length of 40 and an embedding size of 300 as input. For binary classification, we utilised a model architecture with 100 units with a dropout of 0.5, followed by a dense layer and a final layer using the sigmoid activation function. The models are then constructed with the Adam optimizer and a learning rate of 0.01. Figure 77 shows model summary using Keras library.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 40, 300)	1386000
bidirectional (Bidirectional)	(None, 40, 200)	320800
global_max_pooling1d (Global)	(None, 200)	0
dense (Dense)	(None, 100)	20100
dense_1 (Dense)	(None, 2)	202
Total params: 1,727,102		
Trainable params: 341,102		
Non-trainable params: 1,386,000		

Figure 77: Keras model summary of Spanish fire prediction

We employed the EarlyStopping strategy during model training to reduce overfitting and improve the model's generalisation performance. The model training was monitored using validation accuracy. When validation accuracy does not increase after a predetermined number of epochs, training is terminated. The patience parameter was set to 20 and the minimum delta was 0.01. That signifies that the training would be terminated if there was no increase in validation accuracy of at least 0.01 after 20 epochs.

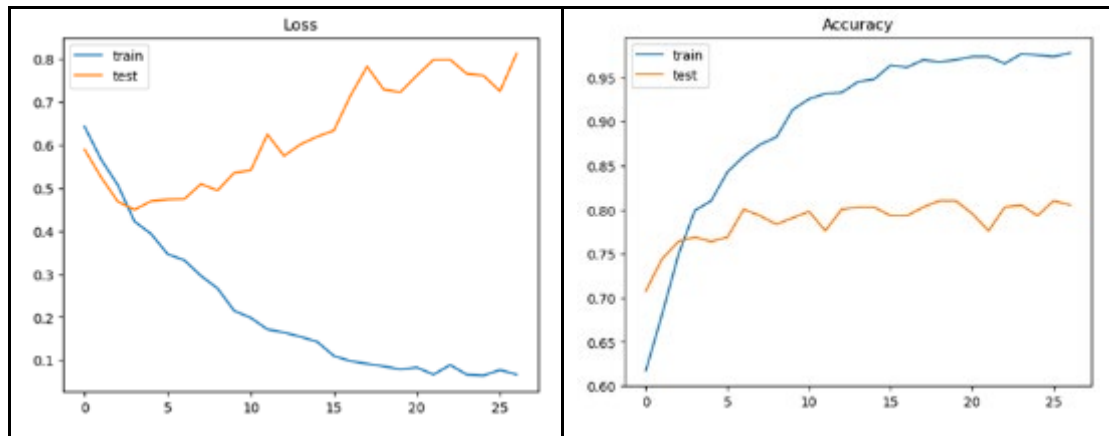


Figure 78: Loss and Accuracy of Spanish fire prediction model training

Figure 78 shows the graphics of loss and accuracy model training. The iteration was stopped at epoch 25, but the model was still slightly overfitting. Future work should be focused on dealing with this issue.

The classification result of class prediction in Spanish is depicted in the Figure 79 below.

Classification Report				
	precision	recall	f1-score	support
not fire	0.83	0.89	0.86	324
fire	0.79	0.69	0.74	190
accuracy			0.82	514
macro avg	0.81	0.79	0.80	514
weighted avg	0.82	0.82	0.81	514

Figure 79: Spanish fire prediction classification report

The precision for the "not fire" class is 0.83, indicating that among the instances the model predicted as "not fire," 83% were correct, while the remaining 17% were false positives. The recall (also known as sensitivity) for this class is 0.89, meaning that the model correctly identified 89% of all actual instances belonging to the "not fire" class. The F1-score, which balances precision and recall, is 0.86, indicating a harmonic mean between these two metrics.

For the "fire" class, the precision is 0.79, signifying that among the instances predicted as "fire," 79% were correct, while 21% were incorrect predictions. The recall for this class is 0.69, indicating that the model identified 69% of all actual instances of the "fire" class. The F1-score for this class is 0.74.

The overall accuracy of the model is 0.82, meaning it correctly classified 82% of all instances in the dataset. The macro average F1-score, which considers the F1-scores of both classes equally, is 0.80. The weighted average F1-score, which accounts for class imbalances, is 0.81.

In conclusion, the classification report provides insights into the model's performance for both classes. While it exhibits strong precision and recall for the "not fire" class, it slightly struggles with the "fire" class, particularly in terms of recall. Further efforts may be needed to improve the model's ability to identify instances of the "fire" class correctly.

c. Italian Wildfire detection model

We did a similar model architecture of Spanish to train the model prediction for Italian wildfire detection. After pre-processing the data, we separated it into 80:20 train and test datasets. The model was then trained over 200 epochs using the single-layer architecture of Bidirectional LSTM. Figure 80 shows model summary using Keras library.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 40, 300)	838800
bidirectional (Bidirectional)	(None, 40, 200)	320800
global_max_pooling1d (Global)	(None, 200)	0
dense (Dense)	(None, 100)	20100
dense_1 (Dense)	(None, 2)	202
Total params: 1,179,902		
Trainable params: 341,102		
Non-trainable params: 838,800		

Figure 80: Keras model summary of Italian fire prediction

The provided sequential model architecture consists of several layers designed for text classification tasks. The model starts with an embedding layer, which transforms input tokens into dense vectors of size 300. These vectors represent the semantic meaning of the words. The subsequent bidirectional layer employs a bidirectional LSTM (Long Short-Term Memory) with 200 units to capture contextual information from both directions in the input sequence. The global max pooling1D layer extracts the most important features from the sequences. Following this, a dense layer with 100 units utilizes these features to learn higher-level representations. Lastly, the output layer, a dense layer with 2 units, generates the final classification results. The model has a total of approximately 1.18 million parameters, with

341,102 of them being trainable. This architecture leverages the strengths of bidirectional LSTM and pooling techniques for text analysis while efficiently managing its parameters.

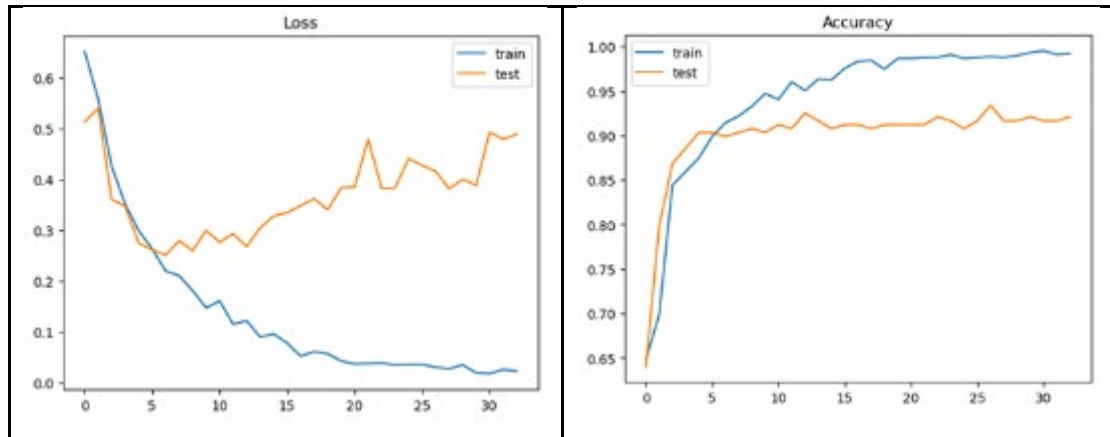


Figure 81: Loss and Accuracy of Italian fire prediction model training

Figure 81 depicts the loss and accuracy model training graphics. The iteration was terminated at epoch 31, but the model remained slightly overfitting. This issue should be addressed in future work.

Classification Report				
	precision	recall	f1-score	support
not fire	0.91	0.95	0.93	184
fire	0.89	0.83	0.86	102
accuracy			0.91	286
macro avg	0.90	0.89	0.90	286
weighted avg	0.91	0.91	0.90	286

Figure 82: Italian fire prediction classification report

The model's precision for the "not fire" class is 0.91, signifying that out of the instances it predicted as "not fire," 91% were accurately classified, while 9% were false positives. The recall for the "not fire" class is 0.95, indicating that the model correctly identified 95% of all actual instances belonging to the "not fire" class. The F1-score, which balances precision and recall, is 0.93 for the "not fire" class.

For the "fire" class, the model's precision is 0.89, implying that among the instances predicted as "fire," 89% were indeed correct, while 11% were incorrect predictions. The recall for the "fire" class is 0.83, denoting that the model identified 83% of all actual instances of the "fire" class. The F1-score for this class is 0.86.

The overall accuracy of the model is 0.91, indicating that it correctly predicted fire occurrences in Italy with an accuracy of 91%. The macro average F1-score, which considers the F1-scores of both classes equally, is 0.90. The weighted average F1-score, which accounts for class imbalances, is also 0.90.

In summary, the classification report outlines the model's effectiveness in predicting fire occurrences in Italy. It demonstrates good precision and recall values, especially for the "not fire" class, suggesting that the model is adept at identifying non-fire instances. However, there

is a slightly lower recall for the "fire" class, implying room for improvement in correctly identifying fire incidents. Overall, the model shows promising performance in fire prediction for the Italian context.

3. Location Extraction

a) Indonesian Model

Location Extraction plays important role to identify location for fire detection. In this work, we employed Named-Entity Recognition (NER), a natural language processing (NLP) technique that involves identifying and classifying named entities within text, such as names of people, organizations, dates, and locations to get location from text. Location extraction using NER specifically focuses on identifying and extracting geographical places, addresses, landmarks, and other location-related information from text data.

In this experiment, we used Bidirectional LSTM and CRF (Conditional Random Forest) to extract location from text. In the context of location extraction, Bidirectional LSTM helps the NER model understand the dependencies between words in a sentence, enabling it to identify location-related patterns more effectively.

Conditional Random Fields (CRF) is a probabilistic modelling technique that considers the sequence of predicted labels when making predictions. In NER, CRF acts as a post-processing layer on top of a neural network-based model. It considers the contextual relationships between adjacent entities to ensure the predicted named entity labels follow a coherent pattern.

We used the BioTagging dataset as seen in the Table 7. Further, the architecture of BiLSTM and CRF in the python code can be seen in Figure 83.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 13)	0
embedding_1 (Embedding)	(None, 13, 20)	117020
bidirectional_1 (Bidirection	(None, 13, 200)	96800
time_distributed_1 (TimeDist	(None, 13, 100)	20100
crf_1 (CRF)	(None, 13, 14)	1638

Total params: 235,558
Trainable params: 235,558
Non-trainable params: 0

Figure 83: BiLSTM and CRF for location extraction

The provided deep learning architecture is designed for sequence labelling tasks, particularly for Named Entity Recognition (NER). It takes sequences of length 13 as input. The architecture includes an embedding layer, which converts input tokens into dense vectors of size 20, allowing the network to capture semantic representations of the tokens.

The Bidirectional layer employs Bidirectional Long Short-Term Memory (BiLSTM) with 200 units to capture context from both directions in the input sequence. This is followed by a TimeDistributed layer that applies a fully connected operation to each time step individually, generating a sequence of vectors with dimensionality 100.

The architecture concludes with a Conditional Random Field (CRF) layer, which enhances sequence labelling tasks by considering the dependencies between adjacent labels. The CRF layer outputs sequences of dimension 13 with 14 possible label classes. Overall, this architecture has approximately 235,558 trainable parameters and is suitable for tasks like NER, where it learns to label each token in a sequence with an appropriate entity class label, benefiting from both the BiLSTM's contextual understanding and CRF's label coherence considerations.

It is then trained using the provided training data (X_train and y_train) with a batch size of 32 and over 25 epochs. During training, 10% of the training data is used for validation. The "verbose" parameter set to 1 indicates that training progress will be displayed.

The accuracy and loss of model training are shown in Figure 84 below.

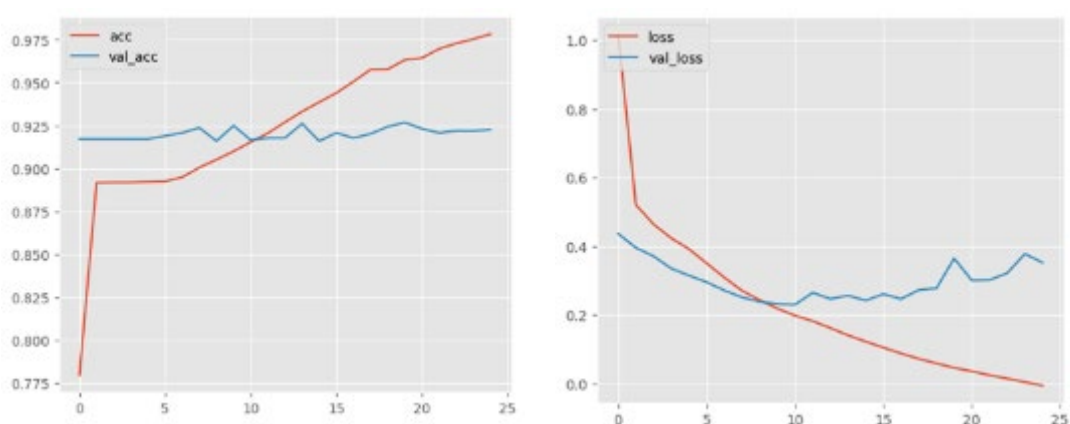


Figure 84: Accuracy and loss training

In this scenario, we divided the data for training and testing using a 90:10 ratio. The test outcome is depicted in Figure 85.

	precision	recall	f1-score	support
geo	0.31	0.41	0.35	29
org	0.00	0.00	0.00	6
per	0.00	0.00	0.00	10
qty	0.56	0.25	0.34	20
tim	0.28	0.36	0.31	22
micro avg	0.28	0.29	0.28	87
macro avg	0.23	0.21	0.20	87
weighted avg	0.30	0.29	0.28	87

Figure 85: Classification report of NER using BiLSTM and CRF

The provided classification report presents the performance evaluation of a Named Entity Recognition (NER) model on multiple named entity categories, including "geo" (geographical locations), "org" (organizations), "per" (persons), "qty" (quantities), and "tim" (time expressions).

In this NER evaluation, the results show variations in precision, recall, and F1-score across different named entity categories. As we focused on location extraction, the "geo" category

achieved a precision of 0.31, indicating that around 31% of instances predicted as geographical locations were correct. The recall for this category is 0.41, indicating that the model identified 41% of all actual geographical locations. The F1-score for "geo" is 0.35, representing a harmonic mean between precision and recall.

b) English Model

We used the similar model architecture for English location detection model (BiLSTM + CRF). Figure 86 below shows the architecture and the number of total and trainable parameters.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 13)	0
embedding_1 (Embedding)	(None, 13, 20)	48360
bidirectional_1 (Bidirection	(None, 13, 200)	96800
time_distributed_1 (TimeDist	(None, 13, 100)	20100
crf_1 (CRF)	(None, 13, 9)	1008
=====		
Total params: 166,268		
Trainable params: 166,268		
Non-trainable params: 0		

Figure 86: English location detection architecture

In this scenario, we divided the data for training and testing using a 90:10 ratio. The test outcome is depicted in Figure 87.

	precision	recall	f1-score	support
geo	0.36	0.43	0.39	23
org	0.00	0.00	0.00	2
oth	0.00	0.00	0.00	8
per	0.00	0.00	0.00	2
micro avg	0.32	0.29	0.30	35
macro avg	0.09	0.11	0.10	35
weighted avg	0.23	0.29	0.26	35

Figure 87: Classification report for English NER model

The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance, especially when there is an imbalance in class distribution. For geographical locations (geo), the F1-score is 0.39. The NER classification model performs reasonably well in identifying geographical locations with moderate precision and recall, although there's room for improvement.

5.2.3. Implementation Model

Multilingual fire detection and location extraction models can be implemented in two forms, standalone Webapp API and integrated with the system models. We have already implemented both methods and in the remainder we will call as first version the Standalone Webapp API and as second version the Integrated Model Systems.

We built the first implementation of our fire detection models in a Webapp form, where they can be accessed one-by-one through the API endpoint /isfire for wildfire detection and /ner for location extraction. Further explanations are shown below (Figure 88).

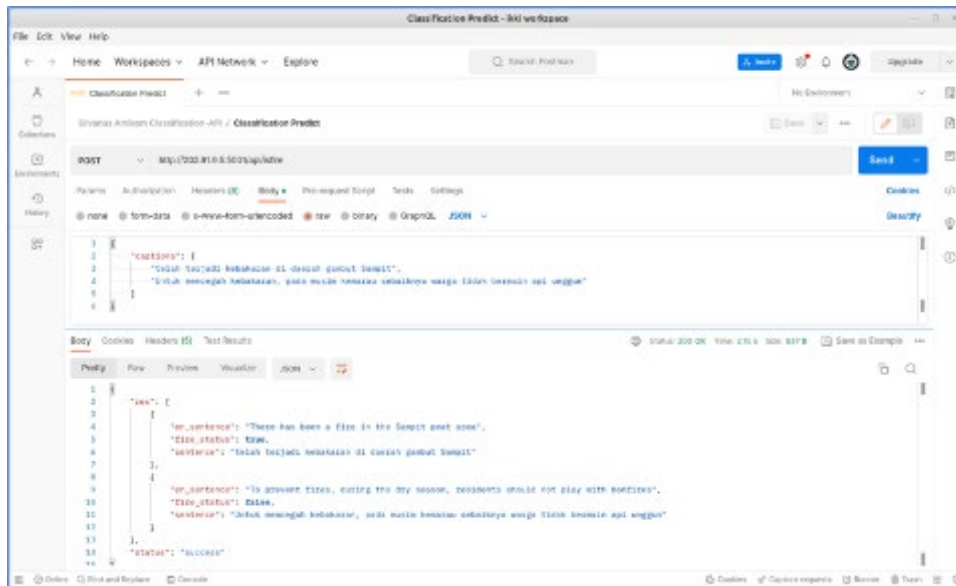


Figure 88: Api Classification for fire

Input

JSON string as an input with the key “captions” contains a list of text that will be classified.

```
curl --location 'http://localhost:5001/api/isfire' \
--header 'Content-Type: application/json' \
--data '{
  "captions": [
    "telah terjadi kebakaran di daerah gambut Sampit",
    "Untuk mencegah kebakaran, pada musim kemarau sebaiknya warga
tidak bermain api unggun"
  ]
}'
```

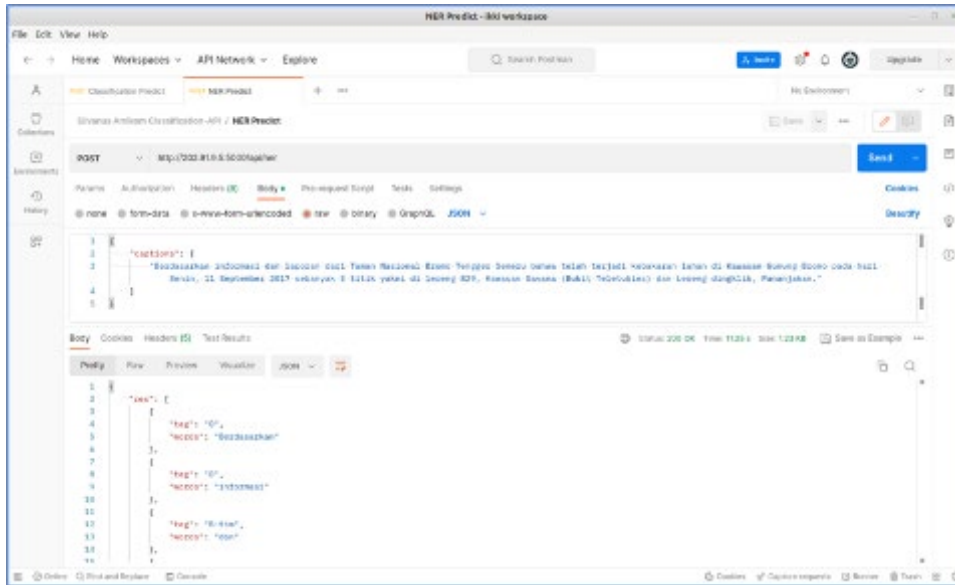


Figure 89: Entity recognition API

Input

JSON string as an input with the key “captions” contains a list of text that will detect the locations.

```
curl --location 'http://localhost:5000/api/ner' \
--header 'Content-Type: application/json' \
--data '{
  "captions": [
    "Berdasarkan informasi dan laporan dari Taman Nasional Bromo Tengger Semeru bahwa telah terjadi kebakaran lahan di Kawasan Gunung Bromo pada hari Senin, 11 September 2017 sebanyak 3 titik yakni di lereng B29, Kawasan Savana (Bukit Teletubies) dan Lereng dingklik, Pananjakan."
  ]
}'
```

The standalone Webapp Api technique that we are building needs several things to be improved, one of which is the NER model's ability to apply (deploy) in production areas and its accuracy in recognising places contained in sentences.

We are currently working on the implementation of the 2nd version, which will offer multilanguage model selection capabilities, allowing our system to be connected with other pilot pipeline systems over the message bus. Based on the pilots we work with; we use the language detection approach to choose the wildfire detection model based on the language in the text input. Now, our model with several five different languages: Indonesian, English, Spanish, Italian, and Slovak.

An implementation concept diagram for the second version of the multilingual social media sensing application is shown in Figure 90 below.

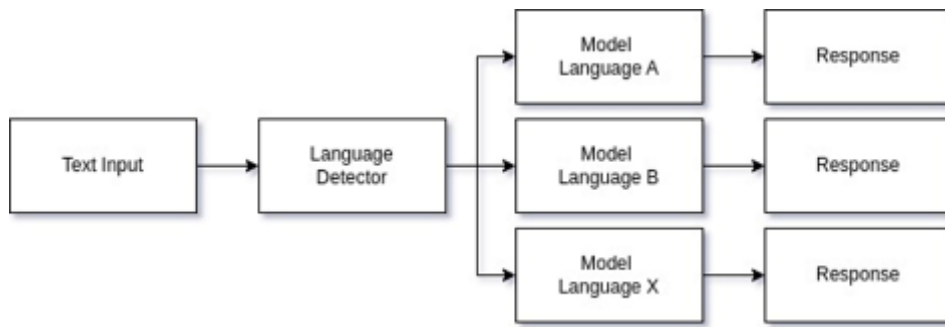


Figure 90: Diagram concept of social media sensing

The following explains how the multi-language social media sensing model flows:

1. The System accepts the request from string/text and process language detection.

```

def predict_lang(self, sentence):
    predict_res = self.model.predict(self.vectorizer.transform([sentence]).toarray())
    detected = predict_res[0]

    return detected
  
```

2. Then the system selects the model based on the language from point 1.

```

def _selected_model(self, lang):
    model_dir = os.path.join(config.MODEL_ROOT_PATH, "models/classification/")
    if lang=='id':
        model_path = os.path.join(model_dir, "id/id_fire_model.h5")
    elif lang=='en':
        model_path = os.path.join(model_dir, "en/themodel.h5")
    elif lang=='it':
        model_path = os.path.join(model_dir, "it/it_fire_model.h5")
    elif lang=='es':
        model_path = os.path.join(model_dir, "es/es_fire_model.h5")
    elif lang=='sk':
        model_path = os.path.join(model_dir, "sk/themodel.h5")

    return model_path
  
```

3. The selected model produces the response in JSON format.

```

[
  {
    "sentence": "telah terjadi kebakaran di daerah gambut Sampit,",
    "en_sentence": "there has been a fire in the Sampit peat area.",
    "fire_status": true,
    "lang": "id"
  },
  {
    "sentence": "Untuk mencegah kebakaran, pada musim kemarau sebaiknya warga tidak bermain api unggun",
    "en_sentence": "To prevent fires, during the dry season, residents should not play with bonfires",
    "fire_status": false,
    "lang": "id"
  }
]
  
```

Because it will be integrated with the message bus in I/O pipelining, the second version of the model will not directly relate to the user as the first did through the Webapp API.

5.3. Component integration

At the outset, our system taps directly into the dynamic stream of Twitter, employing filters based on wildfire related keywords. This mechanism ensures both real-time data acquisition and a focus on relevant tweets. Following this step, the curated tweets are directed to a dedicated topic, named "silvanus_ff", in RabbitMQ, a robust message-brokering service that guarantees an uninterrupted data flow and facilitates efficient consumption by subsequent services.

The Social Media Sensing Service, developed on the Node.js platform for optimal concurrency, consumes the "silvanus_ff" topic from RabbitMQ and orchestrates the concurrent execution of three Python scripts. The Language Detector identifies the tweet's language, ensuring its appropriate processing and classification. The Forest Fire Classifier determines if the tweet indeed reports a wildfire incident. The Named Entity Recognition (NER) examines the tweet's content to pinpoint keywords that hint at the exact location of the potential forest fire.

Once locations are identified, these undergo transformation to geom format, a standardized representation of a spatial location suitable for subsequent processes. All the curated and processed information is then systematically stored in a MySQL database. This database serves two essential functions. Firstly, it facilitates the generation of a wordpool graph, a visual representation of word frequency, providing insights into commonly reported regions or report intensity. Secondly, it supports the Early Warning System's trigger mechanism. This system continuously evaluates data patterns and, upon identifying a significant number of reports from the same location within a predefined period, activates the "silvanus_ews" topic in RabbitMQ. This topic serves as an alert channel, consumed by external applications to notify relevant stakeholders promptly.

To ensure portability, scalability, and ease of deployment, the entire system is encapsulated within a Docker container. Whether implemented on local setups or cloud-based infrastructures, this Docker integration promises seamless functionality across platforms. Additionally, this system seamlessly integrates into larger monitoring frameworks or can function as a standalone early warning unit.

5.4. Plans for future extensions

To reach the final goal of the system, we plan to complete the following additional tasks:

1. Creating a stakeholder notification system that is elaborate with the user interface.
2. Creating a docker image to the Silvanus Cloud so that it can be deployed and consumed by the user interface part.
3. Adding variant languages
4. Adding dataset for fire prediction in English and Slovak
5. Processing location detection in Slovak, Spanish, and Italian

6. Health Impact Component

6.1. Concept of operation

During a wildfire incident, one of the greatest risks faced by the firefighters and people in nearby areas is air quality. As expected, an environmental crisis situation increases the level of gasses, as well as particles that are harmful to people within and around the danger zone. If these changes are not quickly observed to initiate corresponding actions, the concentration of harmful gasses and particles can become dangerously high. Therefore, the development of an air quality monitoring system is of paramount importance.

With the Health Impact Component, we aim to develop a data flow management system from wireless nodes that monitors air quality in the affected area, collects and stores this information, and provides access to both rescuers and services (firefighting, rescue teams, forestry, etc.) while supporting the provision of alerts. Air quality monitoring is carried out by nodes composed of gas sensors for Carbon Monoxide (CO), Sulfur Dioxide (SO₂), Nitrogen Dioxide (NO₂), Ozone (O₃), and particulates PM_{2.5} and PM_{10.0}, connected to a Raspberry Pi microcomputer. For data collection and storage, a web server is developed with a database and a RESTful application. The application communicates with the database for data storage and retrieval. Users can access these functionalities through a set of secure endpoints. The health impact component is a part of the SILVANUS project framework that reflects the potential impact of wildfire emissions in human health, while it generates several spatial and temporal events/alerts towards the control room in order to avoid negative consequences on first response teams and the nearby population.

Figure 91 illustrates a basic use case for the health impact component. Regarding our initial objective of air quality monitoring and the health impact aspect, our approach involves equipping frontline first responders, specifically firefighters, with a compact device housing a Raspberry Pi and five sensors. Presently, wildfire emissions are seamlessly recorded in a MONGODB through a PUT request made to an appropriate endpoint. Interested stakeholders, who share a vested interest in air quality and health impact, can retrieve the most recent emissions data along with the air quality index (AQI) and spatiotemporal information through straightforward GET requests directed at the designated endpoints. It is essential for these partners to undergo authentication using the credentials provided by us to effectively utilize our resources. Moreover, collaborating partners within SILVANUS can retrieve the above data from the SILVANUS Storage Abstraction Layer.

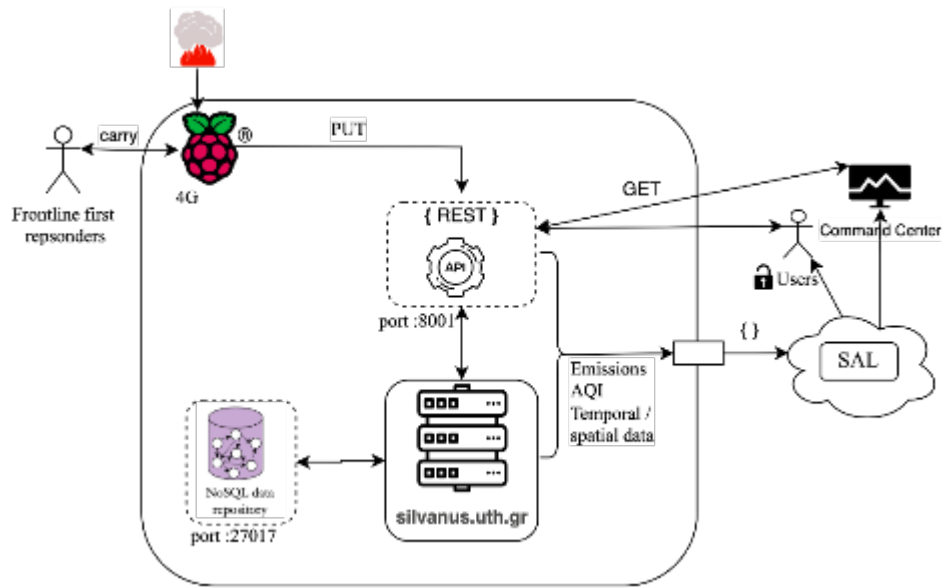


Figure 91: Health impact component – use case

6.2. System implementation and results

6.2.1. Air Quality Monitoring System

6.2.1.1. Air Pollutants

The monitoring system constitutes the backbone of the health impact component as the air quality is observed in real-time. Air quality is calculated based on the quantity of harmful pollutants present in the atmosphere. Special sensors are used to measure these pollutants through a microcomputer, which supplies the sensors with voltage, provides the data transmission channel and sends the data to a remote server. To better understand air quality during and after a wildfire incident we compared several Air Quality Indexes according to type of the pollutants and the air quality levels, as presented in Table 12 and Table 13. Certain countries are omitted from the comparison (e.g., Thailand) as the formula that is used to calculate AQI is very close to the US AQI formula, and actually even more strict on the breakpoints level.

Table 12: Air Quality Indexes - Pollutants Comparison

Pollutants	US AQI	EAQI	AQHI	NAQI	WA DER AQI
Ozone (O ₃)	X	X	X	X	X
Particulate Matter (PM ₁₀)	X	X	X	X	X
Fine Particulate Matter (PM _{2.5})	X	X	X	X	X
Carbon Monoxide (CO)	X			X	X
Sulfur Dioxide (SO ₂)	X	X		X	X
Nitrogen Dioxide (NO ₂)	X	X	X	X	X
Ammonia (NH ₃)				X	
Plumbum - Lead (Pb)				X	

Table 13: Air Quality Indexes - Levels Comparison

	US AQI	EAQI	AQHI	NAQI	WA DER AQI
Number of levels	6	6	4	6	5
Level Characterization	Good Moderate Unhealthy for Sensitive Groups Unhealthy Very Unhealthy Hazardous	Good Fair Moderate Poor Very Poor Extremely Poor	Low Risk Moderate Risk High Risk Very High Risk	Good Satisfactory Moderate Poor Very Poor Severe	Good Fair Poor Very Poor Extremely Poor

The 8 more critical pollutants that are emitted into the air during a wildfire are presented in Table 12. We can observe that every index adopts O3 and PM2.5 while the majority of the indexes also adopts NO2. CO is considered only by the US AQI and WA DER AQI, while NH3 and Pb are considered only by the NAQI. According to the scheme presented in Table 13, AQIs use four up to six air pollution levels. In the health impact component we use six levels (similar to EAQI) with the following level characterizations:

- Good,
- Fair,
- Moderate,
- Poor,
- Very Poor and
- Extremely Poor.

Figure 92, demonstrates the measurements of the five key pollutants supported by our model to determine the index level that describes the current air quality. In general, PM2.5 and particles PM10.0 concentrations are measured as units of µg/m3. In the health impact component we use the parts per million (ppm) unit of measurement.

Particles less than 2.5 µm (PM _{2.5})	0-10	10-20	20-25	25-50	50-75	75-800
Particles less than 10 µm (PM ₁₀)	0-20	20-40	40-50	50-100	100-150	150-1200
Nitrogen dioxide (NO ₂)	0-40	40-90	90-120	120-230	230-340	340-1000
Ozone (O ₃)	0-50	50-100	100-130	130-240	240-380	380-800
Sulphur dioxide (SO ₂)	0-100	100-200	200-350	350-500	500-750	750-1250

Figure 92: Table of European Air Quality Index (based on pollutant concentrations in µg/m3)

6.2.1.2. *Sensors – Equiped by people*

6.2.1.2.1. Gas detection sensors

To detect gasses and particles, sensors from DFRobot's Gravity⁹ series are utilized. These sensors offer significant advantages that can be leveraged in the development of the current system. Firstly, the component design is modular, allowing for easy disassembly and reassembly of the sensors. This provides the flexibility for sensors to be disassembled and reassembled, and for one sensor to utilize the components of another without altering the initial outcome or encountering compatibility, recognition, communication, and other issues. Secondly, the sensors have a standardized plug-and-play interface, eliminating the need to download and install additional drivers. The system can read the sensors upon connection. Thirdly, the company provides an open-source electronic toolkit that covers the basic needs for selecting the data transmission protocol, risk threshold, and basic input/output (I/O) actions.

The gas detection sensors utilized in this component are electrochemical sensors that come pre-calibrated from the factory. Each sensor consists of three components: the gas detector (Sensor Probe), a signal conversion board for electrochemical factors (Signal Conversion Board), and a 4-pin cable for connectivity (4pin Cable). These sensors support three data transmission protocols: analog, I2C, and UART. This versatility in connectivity enables us to select the most suitable protocol based on our specific requirements. It allows us to efficiently handle multiple sensors in a circuit and ensures synchronous data transmission over the same channel or in customized configurations depending on the project's needs. The selection of the sensors is methodical rather than random. These specific sensors offer several advantages for the development of such a system.

- Specifically, they are small in size and lightweight, so firefighters are not burdened with wearing specialized suits that would slow them down during walking and firefighting.
- Furthermore, their operating conditions are satisfactory, as the temperatures of their packaging do not exceed the temperature at which they operate.
- Moreover, their operating voltage allows them to use a 5V battery, making the system relatively small in size and therefore easy to transport.
- Additionally, their cost is not high.
- The sensors contribute to the calculation of the Air Quality Index (AQI)

The sensors have 32 programmable I2C addresses, a built-in temperature compensation algorithm, and a threshold alarm function. They are compatible with most commercial microcontrollers such as Arduino, ESP32, and Raspberry Pi. The operating voltage range is from 3.3V DC to 5.5V DC, and the current consumption should be less than 5 mA. They operate within a temperature range of -20°C to 50°C and a humidity range of 15% to 90% RH without condensation. The sensors have a lifespan exceeding 2 years. In total, there are 12 different gas detectors available in the market: CO, O₂, NH₃, H₂S, NO₂, HCL, H₂, PH₃, SO₂, O₃, CL₂, and HF. They are used and provide data in the same format and manner. The dimensions of the sensors are 37×32mm. Figure 95 presents information about the sensors regarding the four gases we measure, while Figure 94 and Figure 95 present the components of such a sensor.

⁹<https://www.dfrobot.com/product-1272.html>, <https://www.dfrobot.com/product-2508.html>

Gas	Detection Range	Resolution	Response Time(T90)
CO	0-1000 ppm	1 ppm	$\leq 30s$
NO ₂	0-20 ppm	0.1 ppm	$\leq 30s$
SO ₂	0-20 ppm	0.1 ppm	$\leq 30s$
O ₃	0-10 ppm	0.1 ppm	$\leq 120s$

Figure 93: Gas Sensors Specifications



Figure 94: Gas Sensor Components



Figure 95: Gas Detection Sensor

6.2.1.2.2. Particles detection sensors

The PM2.5 particle detection sensor is a digital sensor that measures the concentration of particles and can be used to determine the number of suspended particles in an air volume ranging from 0.3 to 10 micrometers, corresponding to the particle concentration. It provides digital output and can also extract qualitative data about the introduced particles. The sensor operates using laser technology and follows the principles of light scattering theory. It consists of three parts: the laser dust sensor, the sensor adapter, and the sensor adapter cable. The sensor operates within a voltage range of 4.95V to 5.05V. It can handle a maximum current of 120 mA. The particle size (diameter) is divided into three ranges since the sensor is designed to measure all three types of particles: [0.3, 1.0], [1.0, 2.5], and [2.5, 10] micrometers. The measurement range is [0, 999] $\mu\text{m}/\text{m}^3$. The response time is less than 10 seconds. It operates within a temperature range of -20°C to 50°C . The humidity range for operation is [0, 99]%RH. The dimensions of the sensor are 65×42×23 mm. It has a lifespan of more than 5 years. The data transmission protocol used is UART. Figure 96 contains all the components of the detection sensor for PM1.0, PM2.5 and PM10.0 particles.

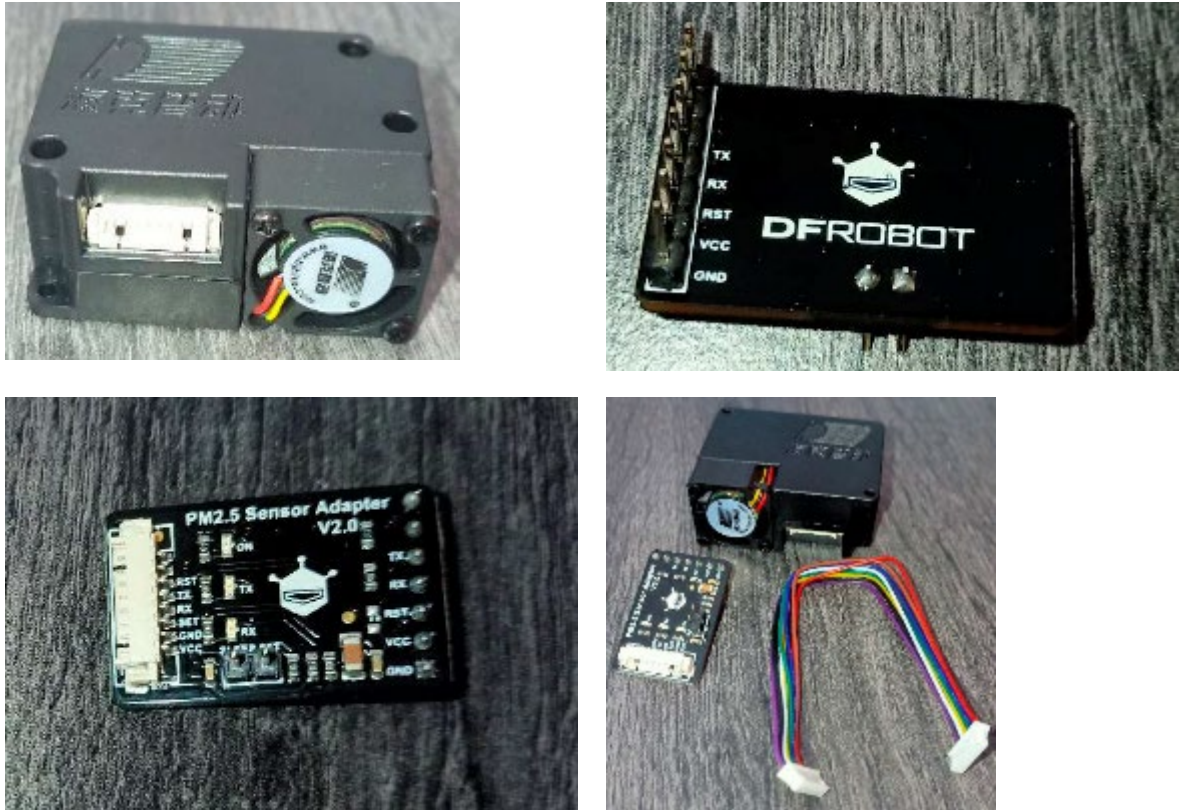


Figure 96: (a) Back View of the Laser Sensor, (b) Front view of the adaptor, (c) Back view of the adaptor, (d) Overall components

6.2.2. Raspberry-Pi

In the health impact component we utilized a Raspberry Pi 4 Model B, the latest model of the latest generation of Raspberry Pi. Like its predecessors, this model features a System on Chip (SoC), providing the Raspberry Pi with a general-purpose processing unit, graphics performance, and I/O capabilities. The SoC used is the Broadcom BCM2711, which includes a quad-core 64-bit Cortex-A72 processor operating at 1.8 GHz. The model is equipped with 8GB LPDDR4-3200 SDRAM, supports 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless networking, has 40 GPIO Pins, and is powered by 5V DC via a USB Type-C adapter or GPIO header. The Raspberry Pi has a set of pins. These pins are called General Purpose Input-Output (GPIO) pins. The GPIO pins are located at the upper left of the board. There are a total of 40 pins for the model we are using and generally in the newer models (older models had 26), arranged in two columns. Each GPIO pin serves a specific purpose. Figure 97 shows the pins with their names (in lowercase letters) and their roles (indicated in parentheses), along with their numbering (in black letters). We observe that there are four pins for power supply, two for 5V and two for 3.3V, eight for grounding, current-carrying pins for data exchange, four for pulse code modulation configuration, and thirteen for general purposes. GPIO pins without specific roles can be used either as a 3.3V voltage source, grounding, or input.

3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

Figure 97: GPIO Pins

According to Figure 98, there are three types of data transmission channels: I2C, UART, and SPI. Among these three, we will focus only on I2C and UART (as the two channels are defined by protocols that are synonymous, here the terms channel and protocol are considered mostly interchangeable). Both of them are serial communication channels. In computer-to-peripheral device communication and data exchange, data flow occurs in two forms: serial and parallel. In parallel communication, each bit of the data channel is transmitted through a separate channel (physical cable), while in serial communication, the bits must be transmitted through a single channel. In serial communication, both ends need to know the start and end of data transmission, as well as ensure the correct transmission of bits. Therefore, additional signals are required to inform these details to the receiver. If these signals are on the same channel as the exchanged digits, the communication is called asynchronous. If a separate channel is used, then the signals are clock pulses, and the communication is called synchronous. The communication protocol I2C (Inter-Integrated Circuit) defines synchronous serial communication using two signal channels or lines: SCL, which stands for Serial Clock, and SDA, which stands for Serial Data. The UART (Universal Asynchronous Receiver/Transmitter) communication protocol defines asynchronous serial communication, as its name suggests.

6.2.3. Sensors configuration

Since Raspberry Pi and gas sensors support multiple, common data communication protocols and particles sensor support only UART protocol, the protocol that is used for communication between Raspberry Pi and particles sensor is UART and for communication between Raspberry Pi and gas sensors is I2C. For the particle sensor, the baud rate is set to 9600 (default value). For the gas sensors, the bus that is used is the first available and four different addresses are used, one for each sensor, from the 32. The connectivity is such that SDA and SCL is common for all sensors, which allows the communication through the same bus.

6.2.4. Raspberry Pi implementation

A gas sensor, prior to its initial operation, needs to establish data acquisition in a passive manner (i.e., data is not continuously transmitted from the sensor but upon request from the Raspberry Pi), incorporates temperature compensation into its calculations, and ultimately identifies the specific gas it detects. During data collection, the data is stored within a structure along with additional information about the node, such as its identifier and location, which will be sent to the server in the form of a JSON object. After data collection and their transformation in the JSON format, the node communicates with the server.

6.2.5. Complete air quality monitoring system

6.2.5.1. Sensors connected to the Raspberry Pi

Figure 98 illustrates an assembled final node of the air quality detection system.

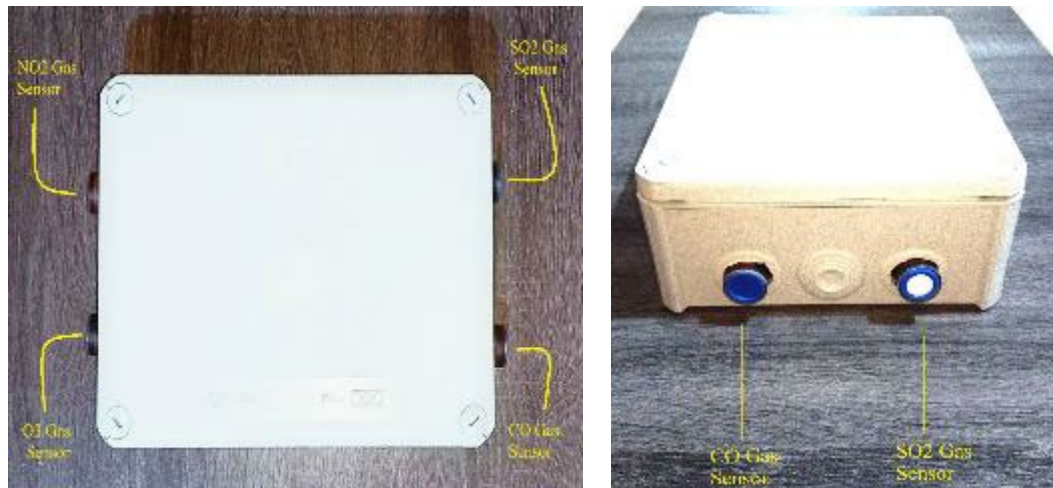


Figure 98: Air Quality Observation System

6.2.5.2. Additional sensors and materials

Figure 99 shows the inside of the sensor system and its components. Besides the gas and particle sensors and Raspberry Pi, the system consists of a WaveShare Solar Power Manager (B)¹⁰, the SIM7600E-H 4G HAT¹¹ module with a 4G antenna and a GPS antenna, together with the breadboard and the cables. With these additional components, the system is capable of being powered, communicating with the server wherever it is located and acknowledging its location.

The SIM module, 4G and GPS antenna form a single unit responsible for the provision of telecommunications services, like dial-up, telephone call, SMS, mail, transfer protocols like TCP, UDP, DTMF, HTTP, FTP, etc., GPS, BeiDou, Glonass and LBS base station positioning. The SIM module contains 40 PIN GPIO extension header for connecting Raspberry Pi and SIM card slot, alongside with other features that are not used in this system, like earphone jack, USB and UART interfaces, TF card slot etc. This module works like a hat to the Raspberry Pi, and does not limit its function.

The solar power bank provides the Raspberry with stored power from an external solar power source (not included in this component). It contains an embedded 10000 mAh Li-Po battery and supports 6V~24V solar panels. It can be charged by either solar panel or type-C power adapter. Due to its features (Maximum Power Point Tracking function, multi protection

¹⁰ <https://www.waveshare.com/solar-power-manager-b.htm>

¹¹ https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT

circuits, operating temperature in the range $-40^{\circ}\text{C}\sim 80^{\circ}\text{C}$, 5V USB input) it is considered suitable for use in this system.

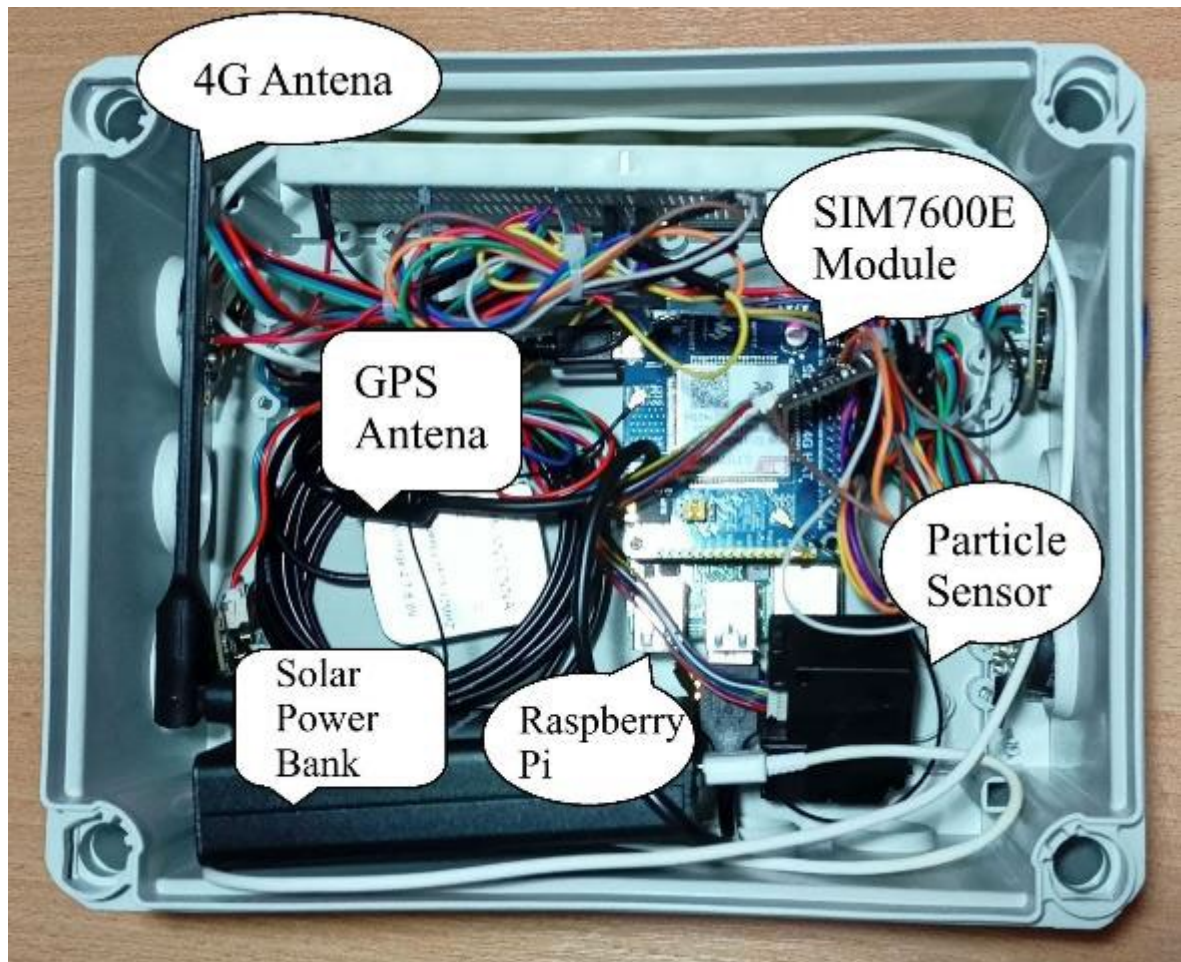


Figure 99: Interior System and Components

6.2.5.3. *RESTful API*

6.2.5.3.1. RESTful API overview

The data collected from the sensors are saved to a web application. This is because the data must be located in one place and not distributed throughout multiple computers, so the end-user will be able to easily access them. Since multiple Raspberry Pis are used, this requirement is crucial for the development of the end system. Also, the Raspberry Pis are not capable of holding multiple data, because their resources are limited and this could undermine its ability to collect real-time data. This application is able to communicate not only with the Raspberry Pi, but also with users that want to access and read the data and the AQI. It is developed on a REST API, providing other applications with the ability to exchange data with the server. Additionally, the information received from the nodes is stored and organized using a NoSQL database. To gain access to API's endpoints, users must be authenticated. The system supports an authentication system, using Basic Authentication with Roles.

The RESTful application is implemented in Python, using the Flask framework¹². Flask is a lightweight framework which provides almost all aspects of a framework, like web template system and URL mapping. The database management system used to store the data was MongoDB¹³. MongoDB is a NoSQL database management system. These systems are distinguished by their widespread duplication and division of data among geographically distributed nodes, flexible system state and the model of ultimate consistency across replicas.

6.2.5.3.2. Deployed VM

The web application must run continuously and receive requests. For this reason, a web server is used. The server is responsible for handling files and requests. The web server consists of 2 server programs: Gunicorn¹⁴ and Nginx¹⁵. Gunicorn server provides the Python Web Server Gateway Interface (WSGI). The WSGI is established in order to allow the different Python web applications to communicate with the web server program. In general, a web server program communicates with a backend application using an interface. For back-end applications written in Python, the standard interface for the servers is WSGI. It is built-in to the Gunicorn, so no further actions are required. Gunicorn is not capable enough to handle issues that may arise, such as network congestion, load balancing, etc. and is not scalable. The Nginx server is able to address these issues. Is also able to scale the application and address security issues. Figure 100 presents the specifications and the integrated software components e.g., frameworks, web server and database of the VM where the health impact component is implemented.

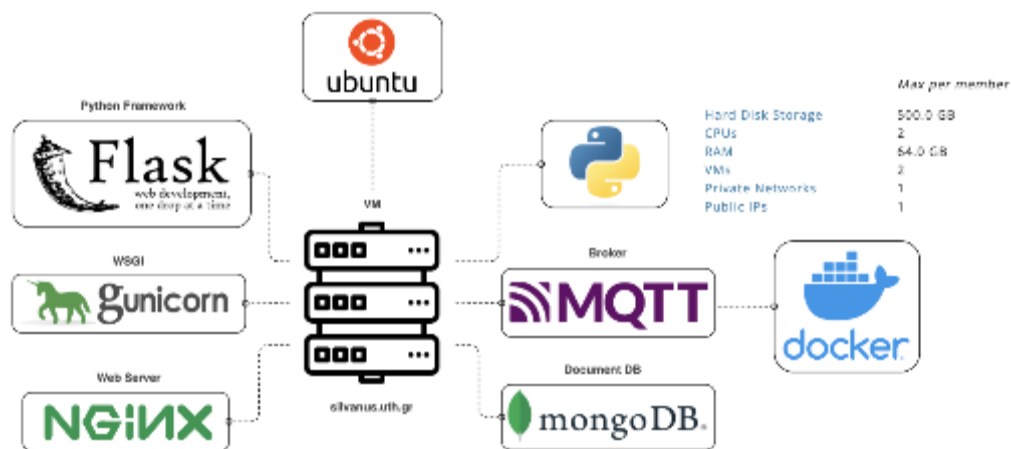


Figure 100: VM characteristics and integrated components

6.2.5.3.3. RESTful API endpoints

The REST API consists of endpoints that are responsible for accessing the database. Through them, users can add or read data directly from the database. The data are in JSON form. These endpoints are not accessible by everyone verified, but from users with specific roles, which also differs for every endpoint. Table 14 describes the endpoints.

¹² <https://flask.palletsprojects.com/en/2.3.x/>

¹³ <https://www.mongodb.com/>

¹⁴ <https://gunicorn.org/>

¹⁵ <https://www.nginx.com/>

Table 14: Endpoints description.

ENDPOINT	HTTP METHOD	DESCRIPTION	PARAMETERS
/insert-data	POST	Send data to the server.	<ul style="list-style-type: none"> - JSON file - Authentication Credentials
/get-latest-data	GET	Get the latest added data.	<ul style="list-style-type: none"> - Number of data - Authentication Credentials
/aqi	GET	Get the most recent AQI measurement.	<ul style="list-style-type: none"> - Authentication Credentials
/health-impact-data-metadata	POST	emissions, AQI and metadata	<ul style="list-style-type: none"> - SILVANUS Credentials

Endpoint: insert-data

This endpoint is periodically called from Raspberry Pis to send the data they have collected to the web server. They contain the concentration values of the observed emissions, geographical coordinates and a unique id. These data are just collected and not processed in the Raspberry Pi. The processing is performed on the server. The data are sent in JSON form, as shown in Figure 1010. The web application then registers these in two collections, one which contains only the concentrations of the emissions and one which contains the calculated European Air Quality Index (EAQI).

```
{
  "Emissions": {
    "CO": 0.0,
    "SO2": 0.0,
    "NO2": 0.0,
    "O3": 0.0,
    "PM1.0": 0,
    "PM2.5": 0,
    "PM10.0": 0,
  },
  "Pi_id": 1,
  "Coordinates": [18.5, 51.2]
```

Figure 101: /insert endpoint sent JSON file

Endpoint: get-latest-data

Using this endpoint, authenticated users can access the database and retrieve the latest chronologically registered data, added from the Raspberry Pi. In order to pass the number of data that will be retrieved, users insert it into the URL as an HTTP header argument in the form ?emissions=<number>. The data that each user requests are returned to each one as a list of JSON files in chronological order. Figure 102 presents the list of JSON files with the three most recently registered data.

```
[
  {
    "metadata": {
      "sensorId": 0,
      "position": {
        "Longitude": 22.4387,
        "Latitude": 38.8749
      }
    },
    "timestamp": "2023-08-29T09:59:40.675506Z",
    "Emissions": {
      "CO": 0.0,
      "SO2": 0.0,
      "O3": 0.0,
      "NO2": 0.0,
      "PM1.0": 0,
      "PM2.5": 0,
      "PM10.0": 0
    }
  },
  {
    "metadata": {
      "sensorId": 0,
      "position": {
        "Longitude": 22.4387,
        "Latitude": 38.8749
      }
    },
    "timestamp": "2023-08-29T09:58:49.429824Z",
    "Emissions": {
      "CO": 0.0,
      "SO2": 0.0,
      "O3": 0.0,
      "NO2": 0.0,
      "PM1.0": 0,
      "PM2.5": 15,
      "PM10.0": 0
    }
  },
  {
    "metadata": {
      "sensorId": 0,
```

```
    "position": {
      "Longitude": 22.4387,
      "Latitude": 38.8749
    },
    "timestamp": "2023-08-29T09:52:06.662253Z",
    "Emissions": {
      "CO": 0.0,
      "SO2": 0.0,
      "O3": 0.0,
      "NO2": 0.0,
      "PM1.0": 0,
      "PM2.5": 0,
      "PM10.0": 0
    }
  }
}
```

Figure 102: /get-latest-data endpoint retrieved JSON

Endpoint: aqi

This endpoint is used to retrieve the most recently registered AQI. Authenticated users calling this endpoint, will receive a JSON file with information for the AQI, as shown in Figure 103. The AQI is retrieved from the collection with the calculated EAQI.

```
{
  "AQI": "Good",
  "position": {
    "Latitude": 38.8749,
    "Longitude": 22.4387
  },
  "radius": 2,
  "sensorId": 0,
  "timestamp": "2023-08-29T09:59:40.675506Z"
}
```

Figure 103: /aqi endpoint retrieved JSON file

Endpoint: health-impact-data-metadata

The above endpoint periodically posts a multipart http request to SILVANUS Storage Abstraction Layer (SAL). Two json files are ingested into SAL, namely

- A. **data.json** that contains a structured representation of data, specifically related to air quality sensor readings and metadata and
- B. **meta-data.json** that contains metadata information associated with the data.json file.

Figure 105 and Figure 106 present the latter two json files (in-depth description about the JSON files is provided in D5.1).

```
{
```

```
"uuid": "uth-123123-lkasjd82-askjd91230asd",
"sensors_type": [
  "PM1.0",
  "PM2.5",
  "PM10.0",
  "sulfur dioxide",
  "carbon monoxide",
  "nitrogen dioxide",
  "ozone"
],
"timestamp": "2023-04-23T11:29:36.372+00:00",
"location": [
  {
    "placename": "somewhere",
    "geometry": {
      "type": "Point",
      "coordinates": [
        {
          "lat": 35.151688,
          "lon": 33.350244
        }
      ]
    }
  }
],
"area": {
  "radius": 2,
  "unit": "meter"
},
"sensor_id": "raspberry_1",
"sensory_data": {
  "PM1.0": {
    "value": 228.0,
    "unit": "micrograms per cubic meter"
  },
  "PM2.5": {
    "value": 230.0,
    "unit": "micrograms per cubic meter"
  },
  "PM10.0": {
    "value": 527.0,
    "unit": "micrograms per cubic meter"
  },
  "sulfur dioxide": {
    "value": 228.0,
    "unit": "ppm"
  },
  "carbon monoxide": {
    "value": 1.0,
    "unit": "ppm"
  },
  "nitrogen dioxide": {
    "value": 0.2,
    "unit": "ppm"
  }
},
```



```

    "ozone": {
      "value": 0.2,
      "unit": "ppm"
    }
  },
  "AQI": "Extremely Poor"
}

```

Figure 104: data.json file

```

{
  "descriptor": {
    "uuid": "uth-123123-lkasjd82-askjd91230asd",
    "obj-class": "IoT",
    "format": {
      "type": "json",
      "resolution": "100",
      "output": "json"
    },
    "access": "default",
    "dataset-type": "air-quality",
    "created": "1674574406.7829435"
  },
  "spatial": {
    "bbox": "POLYGON ((16.0295831170816712 41.9183734508349062,
16.0295831170816712 41.8832522880823319, 16.0872243646491313
41.8832522880823319, 16.0872243646491313 41.9183734508349062,
16.0295831170816712 41.9183734508349062))",
    "coordinates": [
      {
        "lat": 41.918373450834906,
        "lon": 16.02958311708167
      },
      {
        "lat": 41.88325228808233,
        "lon": 16.02958311708167
      },
      {
        "lat": 41.88325228808233,
        "lon": 16.08722436464913
      },
      {
        "lat": 41.918373450834906,
        "lon": 16.08722436464913
      }
    ]
  },
  "pilot": "greek"
},
  "temporal": {
    "datetime": "latest",
    "daterange": "from:to"
  },
  "lineage": {
    "source": "[]"
  }
}

```

```
    "processing": "raw"
  },
  "tag": {
    "device": "raspberry pi",
    "sensors": [
      "PM1.0",
      "PM2.5",
      "PM10.0",
      "sulfur dioxide",
      "carbon monoxide",
      "nitrogen dioxide",
      "ozone"
    ],
    "AQI": true
  }
}
```

Figure 105: meta-data.json file

6.2.5.4. Data visualization

Another endpoint has been developed, in order to fulfill the need of data visualization in real time. This endpoint renders a dynamic site that plots a bar chart and a line chart, which represent the change to concentrations and AQI in real time, based on the latest data registered. The bar chart represents the change of the value of the AQI as percentage (0 for “Good”, 0.2 for “Average” and so on). The line chart represents the change of the value of the concentration in time. It consists of five lines, each corresponding to an emission that is used in the calculation of the AQI. These plots update when new emissions observations are registered in the database. Figure 106 shows the website’s interface.



Figure 106: Visualization webpage

6.3. Sensors – Attached to vehicles or ground

6.3.1. Smart Spot

Besides the sensors that are used by the firefighters, another set of sensors is used, in order to monitor the wildfire spread in the forest. The sensors that are used are ready products of Libelium, specifically the Libelium Smart Spot model¹⁶ (Figure 107) Libelium offers ready-made solutions for more general IoT applications, like measuring air and noise pollution, air quality pollutants, water quality and detecting fire. This specific model is well-suited for monitoring diverse environmental aspects like air quality, temperature, humidity, noise, and it can also function as a weather station. It's a comprehensive device that incorporates essential sensors for these functions, along with hardware and software that facilitate the gathering and sending of measurements, eliminating the necessity for extra purchases or user-initiated software setup. This feature streamlines the process of installation, administration, setup, and communication. Furthermore, the system's utility extends to remote applications, thanks to its power alternatives like linking to solar panels or high-capacity batteries. Table 15 contains specification information about Smart Spot.

Table 15: Libelium Smart Spot Specifications

OS	FreeRTOS
CPU	Dual Core 240MHz
RAM	16MB
Connectivity	WiFi, NB-IoT
Remote Control	Homard Platform
Energy Consumption	180-300 mA Active
Voltage	5V
Size	300x200x36,7 mm
Operating Temperature	[-30, 60] °C
Weight	1,8kg
Gas Sensors	NO ₂ , H ₂ S, CO, NO, SO ₂ , O ₃ , NH ₃ , CO
Particle Sensors	PM _{1.0} , PM _{2.5} , PM _{10.0}
Wind Parameters	Temperature, Humidity, Pressure

¹⁶ <https://marketplace.eiturbanmobility.eu/products/hopu-smart-spot/>



Figure 107: Smart Spot - IoT device for environmental parameter sensing

6.3.2. 2. Smart Spot configuration

Smart Spot management can be accomplished using the following platform: <https://homard.hopu.eu/> . The platform encompasses diverse functionalities crucial for effective device management. It supports remote diagnostics and monitoring of device status, memory usage, battery health, and more. Connectivity options, security features, and the ability to remotely manage device peripherals are incorporated, along with features like remote locking and wiping for lost or compromised devices. Overall, the platform ensures comprehensive device management for enhanced performance, security, and operational control.

6.3.3. Smart Spot technical dashboard

A visualization dashboard is enabled in <https://iprism.hopu.eu/>. Two panels have been integrated to visually represent the presence of gases (NO₂ and O₃) and particulate matter (PM₁₀, PM_{2.5}, and PM₁) within the proximity of the deployed smart spot devices. Figure 108 and Figure 109 illustrate pollutant measurements for O₃ and PM_{2.5} within a specific geographical zone.

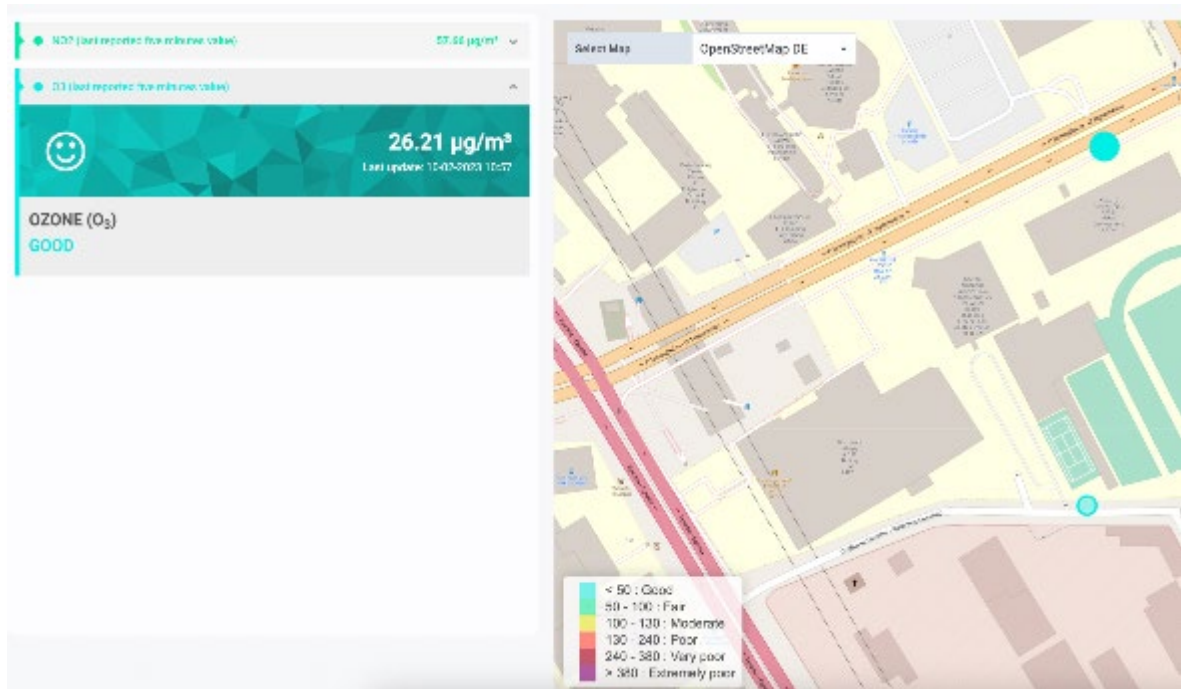


Figure 108: Visualization dashboard (O3)

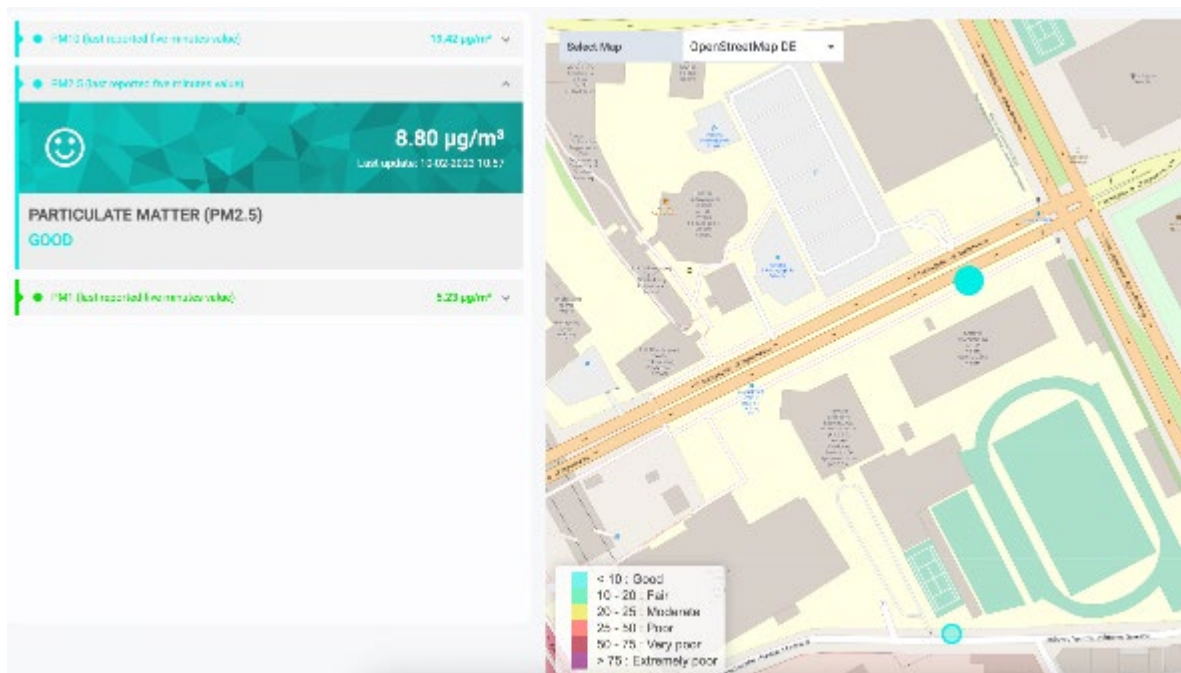


Figure 109: Visualization dashboard (PM2.5)

6.3.4. Smart Spot MQTT Broker

For the communication between Smart Spot sensors and end users the MQTT protocol is used. Message Queuing Telemetry Transport (MQTT) protocol provides connection between devices and networks with middleware and applications. Multiple clients connect and exchange information with each other through a broker, which is responsible for information

distribution. The protocol uses the publish-subscribe model, where a publisher (one of the clients) publishes messages to topics, and clients subscribed to those topics (subscribers) can read these messages. This architecture enables many-to-many communication as well as communication between machine-to-machine, machine-to-server, and server-to-server. Transmission relies on TCP/IP but can also extend to TSL/SSL. In this work we select the Mosquitto¹⁷ broker which implements the MQTT protocol. In general, the Mosquitto system has three main parts: the Mosquitto broker, the functions that let clients publish and subscribe to messages, and the library for setting up the client's role. Mosquitto is designed to work with simple message systems, especially for devices that don't have a lot of power. It's compatible with the latest MQTT protocol, it's open-source, and it offers features like dynamic topics, support for web connections, bridges and security through user verification.

MQTT broker deployment and exploitation is presented in Figure 110. Initially, we set up a Mosquitto MQTT broker within a Docker¹⁸ container hosted on our virtual machine. Subsequently, we configured each equipment to publish data at five-minute intervals to two specified topics, respectively (lib1attns and lib2attns). Stakeholders who possess an interest in air quality, subscription to our broker is achievable through MQTT clients such as MQTT Explorer or the Paho¹⁹ package in Python. Portainer.io²⁰ is also installed to our VM as a container management tool. The operational procedure of the MQTT Explorer client is illustrated in Figure 111. In this instance, a client has been successfully subscribed to the topic "lib1attns," and at regular 5-minute intervals, it receives a JSON file, as depicted in Figure 112. We have also developed a sample Python code that users can easily employ (Figure 113). This code demonstrates the setup of an MQTT client to connect to the specified broker, subscribe to a topic, and handle incoming messages.

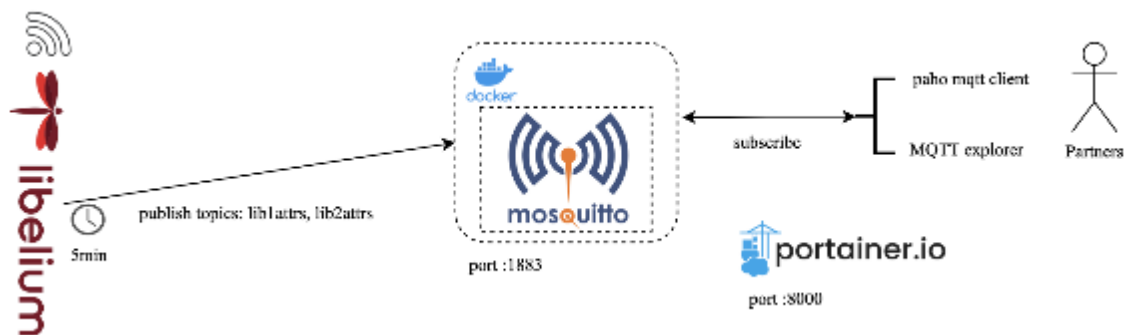


Figure 110: Broker deployment and configuration

¹⁷ <https://mosquitto.org/>

¹⁸ <https://www.docker.com/>

¹⁹ <https://eclipse.dev/paho/index.php?page=clients/python/index.php>

²⁰ <https://www.portainer.io/>

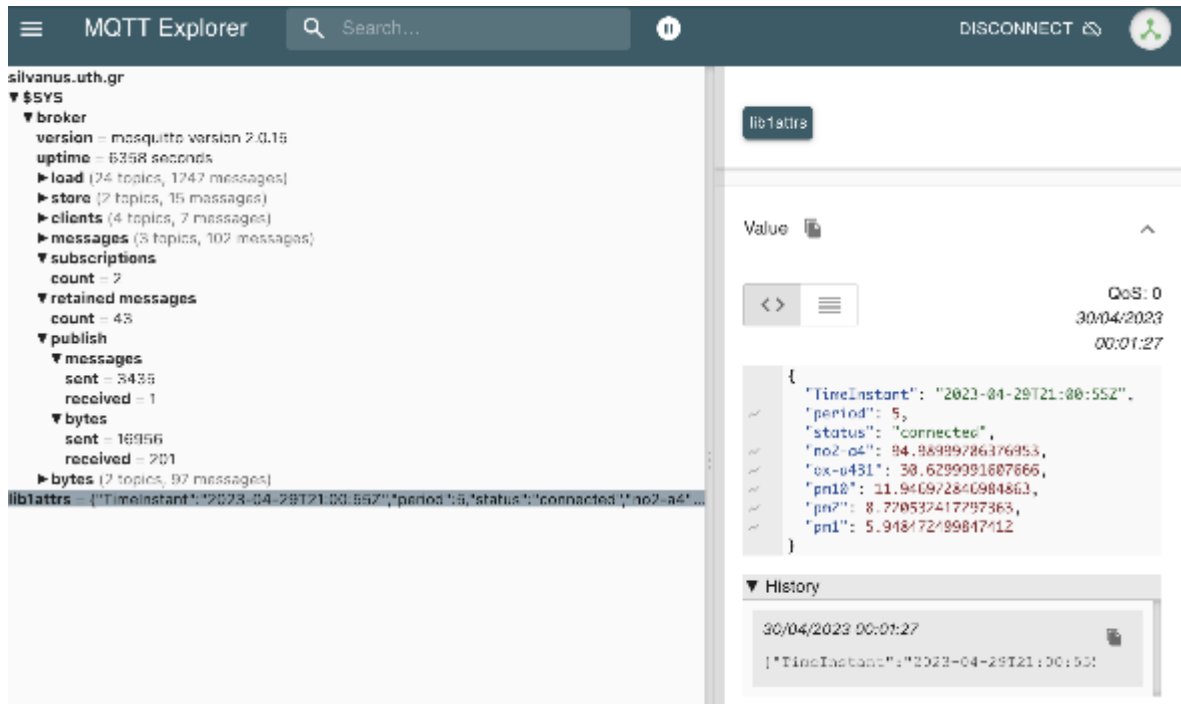


Figure 111: MQTT Explorer Client

```
{
  "TimeInstant": "2023-04-30T18:10:00Z",
  "period": 5,
  "status": "connected",
  "no2-a4": 94.830001831054688,
  "ox-a431": 29.280000686645508,
  "pm10": 15.006196975708008,
  "pm2": 10.773552894592285,
  "pm1": 7.3032760620117188
}
```

Figure 112: JSON file

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("liblattrs")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
    print("Received message: ", str(msg.payload.decode("utf-8")))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("silvanus.uth.gr", 1883, 60)
```

```
client.loop_forever()
```

Figure 113: Python's paho package

6.4. Component integration

In the context of application integration, pilots can straightforwardly incorporate our system into their operations. This is achieved by initiating HTTP access via our REST API. Detailed information about the API is available in the previous section (6.2.5.3) while information can be also accessible on Postman using the following link: <https://documenter.getpostman.com/view/29042250/2s9Xy2QCLs>.

Regarding the data ingestion from the raspberry sensors into the SILVANUS Storage Abstraction Layer (SAL), a comprehensive description has already been outlined in D5.1. This documentation provides an in-depth description about two JSON files, namely "data.json" and "meta-data.json" as well as for the accompanying Python code which is responsible to produce/manage. File data.json contains a structured representation of data, specifically related to air quality sensor readings, while meta-data.json contains metadata information associated with the data.json file. Both files successfully ingested into SAL using a multipart POST request. Pollutant measurements obtained from the smart spot device can be accessed through the MQTT protocol, as outlined in section (6.3.4). A Mosquitto broker is operational within a Docker container, enabling devices to publish data to designated topics, while users can subscribe to these topics to retrieve valuable information. For deployment, we use the Virtual Machine described in section (6.2.5.3.2). Detailed documentation about the deployment can be found on the Silvanus GitHub: <https://github.com/silvanus-prj/health-impact>.

6.5. Plans for future extensions

In the first place of our future agenda is to a) demonstrate another air quality component that can seamlessly attached to ground vehicles or buildings. This particular component will include a Smart Spot solution from Libelium and will operate through communication with an MQTT broker, b) conducting practical testing of the showcased health impact component during a real wildfire incident.

7. Evacuation route planning

7.1. Concept of operation

The evacuation paths planning module aims to ensure the safe movement of citizens and fire responders away from an area affected by a wildfire incident. This goal will be achieved by developing functional synergies and interactions with other modules of the SILVANUS platform and some open external tools.

The module runs continuously on a Virtual Machine (VM) and is accessible to authorized users through straightforward HTTP GET requests (further information is provided in Section 7.2.1.2). Moreover, the module will be triggered when a new fire-related message is published to the broker or when a new event is added to the knowledge graph, signaling the need for a safe evacuation of an area in danger. Upon request, the module retrieves suitable contextual data from multiple sources (e.g., wildfire location, weather, wildfire front, firefighters' location and burned area). Next, the module interacts with a) the Health Impact Component to obtain valuable insights about the air quality and characterize the neighboring sub-areas of interest in terms of their potential health implication and b) with a smoke dispersion module to obtain informations regarding the dispersion of the smoke in the area. Additionally, data regarding population characteristics and locations are necessary and are fetched after contacting the SILVANUS Storage Abstraction Layer (SAL). Once the module gathers all the necessary inputs, it estimates the safe areas and paths using a greedy reward-based algorithm adopted in OpenRouteService. The results of the algorithm are then sent to the Command and Control center and may be distributed to other interested actors, e.g., citizens and first response teams.

7.2. SW implementation and results

To operate the evacuation route planning module, certain data inputs are essential, including fire spread projections, burnt area information, firefighters' locations, smoke dispersion details, plume height data, and population distribution data. Unfortunately, the majority of these inputs are currently unavailable. As a result, our evaluation proceeds using synthetic data that closely approximates realistic fire behavior. Specifically, for simulating smoke dispersion, we've implemented a Gaussian plume model to estimate pollutant concentrations in the atmosphere and accurately simulate smoke movement. The rest of this section is organized as follows: we begin by introducing the smoke dispersion model, followed by an analysis of the fundamental components of the evacuation path planning module and the presentation of various evacuation scenarios based on different data inputs.

7.2.1. Smoke dispersion

Smoke from wildfires has the potential to cause various temporal and spatial effects. Land managers are tasked with the challenge of reconciling concerns related to human health, nuisance smoke, and transportation hazards with matters pertaining to wildlife management, forest health and safety, and ecosystem restoration (Achtemeier et. Al, 1998).

Models designed to forecast smoke impacts of wildfires consist of several components. The first component involves a comprehensive description of the emissions source, which should include both pollutants and heat release. The second component requires the determination of plume rise, which involves an assessment of the atmospheric stability and wind profile, as well as the fire-source rate of heat release, to determine the vertical extent of the plume. The third component, which partially overlaps with the plume rise element, entails the actual movement of the smoke (namely, the transport and dispersion) by the ambient wind (Goodrick et. Al2012).

Most smoke dispersion models estimate the concentration of pollutants downwind of a wildfire source utilizing input data on the following (Good Practice Guide, 2004):

- rate of emission of pollutants
- features of the emissions source
- topographical characteristics of the local region
- meteorological conditions of the surrounding area
- ambient or background concentrations of pollutants.

A comprehensive depiction of the utilization of this data in a smoke dispersion model is presented in Figure 114.

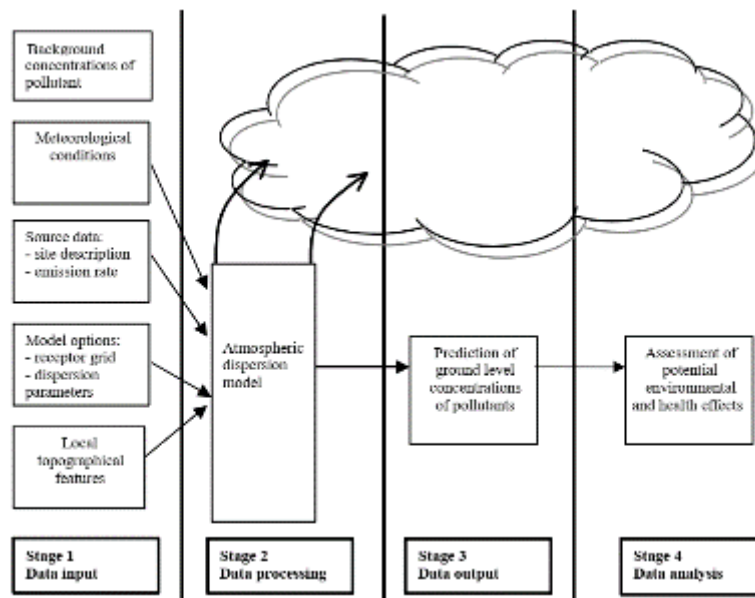


Figure 114: Smoke dispersion modeling process (Good Practice Guide, 2004)

The outcomes of dispersion modeling can furnish an approximation of impacted areas, ambient concentrations, and guide the selection of protective measures suitable for application in the occurrence of a burning event. These protective measures may encompass the evacuation for citizens and first responders located in the downwind direction.

7.2.1.1. Gaussian-plume models - Theoretical Background

Gaussian-plume models are extensively implemented, well understood, and effortlessly applicable, and have been recognized globally until recently, for smoke dispersion estimation. The formula adopted is simple and describes the concentration field generated in three dimensions by a point source under conditions of stationary meteorology and emission. It must be noted that these models operate on certain assumptions:

- the dispersion of the plume follows a Gaussian distribution in both the horizontal and vertical dimensions, with corresponding standard deviations, and
- complete reflection of the plume occurs at the surface of the Earth (i.e., no reaction or deposition) (Zannetti et. Al, 1990), (Turner et. Al. 1970).

The general equation to estimate the steady state concentration of an air pollutant is given as follows:

$$C(x, y, z; H) = \frac{Q}{2\pi\sigma_y\sigma_z u} \exp\left[-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2\right] \left\{ \exp\left[-\frac{1}{2}\left(\frac{z-H}{\sigma_z}\right)^2\right] + \exp\left[-\frac{1}{2}\left(\frac{z+H}{\sigma_z}\right)^2\right] \right\}$$

where:

C = mean concentration of a pollutant at a point (x,y,z) [$\mu\text{g}/\text{m}^3$]

H = effective source height [m] - physical source height and plume rise

Q = uniform pollutant emission rate [mass/s]

σ_y = standard deviation of plume concentration distribution in the horizontal [m]

σ_z = standard deviation of plume concentration distribution in the vertical [m]

u = mean wind speed [m/s] - in downwind direction

Figure 115 and Figure 116 illustrate the aforementioned concepts.

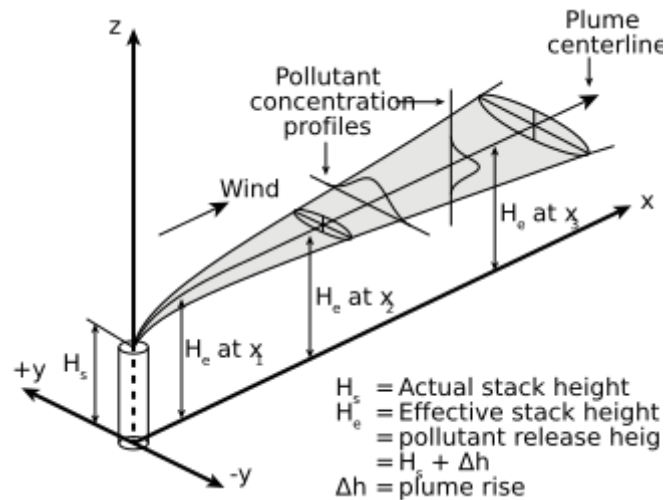


Figure 115: Gaussian air pollutant dispersion plume and coordinate system (from en.wikipedia.org)

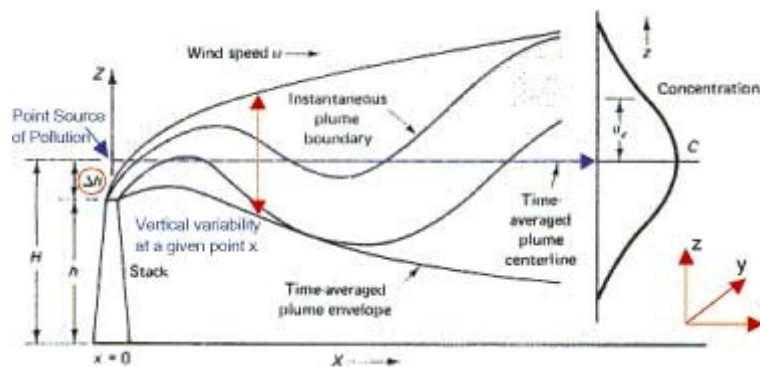


Figure 116: Plume rise (from http://www-personal.umich.edu/~weberg/eff_stack_height.htm)

Values for σ_y and σ_z depend on the stability class of the atmosphere, which in turn can be determined by assessing wind speed at a height of approximately 10 meters, and by examining incoming solar radiation during the day, or cloud cover during the night. They can be estimated at a certain downwind distance x as (Turner et. Al. 1970), (Gifford et. Al. 1961) , (Gifford et. Al, 1980):

$$\sigma_y(x) = \frac{k_1 x}{[1 + (\frac{x}{k_2})]^{k_3}} \quad \sigma_z(x) = \frac{k_4 x}{[1 + (\frac{x}{k_2})]^{k_5}}$$

where the constant values are given in Table 16.

Table 16: Values of the constants for the estimation of σ_y and σ_z

Stability Class	<i>k1</i>	<i>k2</i>	<i>k3</i>	<i>k4</i>	<i>k5</i>
A	0.250	927	0.189	0.1020	-1.918
B	0.202	370	0.162	0.0962	-0.101
C	0.134	283	0.134	0.0722	0.102
D	0.0787	707	0.135	0.0475	0.465
E	0.0566	1,070	0.137	0.0335	0.624
F	0.0370	1,170	0.134	0.0220	0.700

Pausquill's atmosphere stability categories (Pasquill, F. 1961) (A - Extremely unstable; B - Moderately unstable; C - Slightly unstable; D - Neutral; E - Slightly stable; F - Moderately stable) are presented in Table 17.

Table 17: Atmospheric Stability Classes

Surface Wind Speed (at 10 m) [m/sec]	Day			Night	
	Incoming Solar Radiation			Thinly Overcast	
	Strong	Moderate	Slight	≥ 4/8 Cloud	≤ 3/8 Cloud
< 2	A	A - B	B		
2 - 3	A - B	B	C	E	F
3 - 5	B	B - C	C	D	E
5 - 6	C	C - D	D	D	D
> 6	C	D	D	D	D

The curves of σ_y and σ_z as produced as a function of the downwind distance from the source, x , are presented in Figure 117. The crucial aspect to consider is that, in unstable conditions, the standard deviations increase rapidly in the downwind direction, whereas in stable conditions, the standard deviations remain modest in the downwind direction.

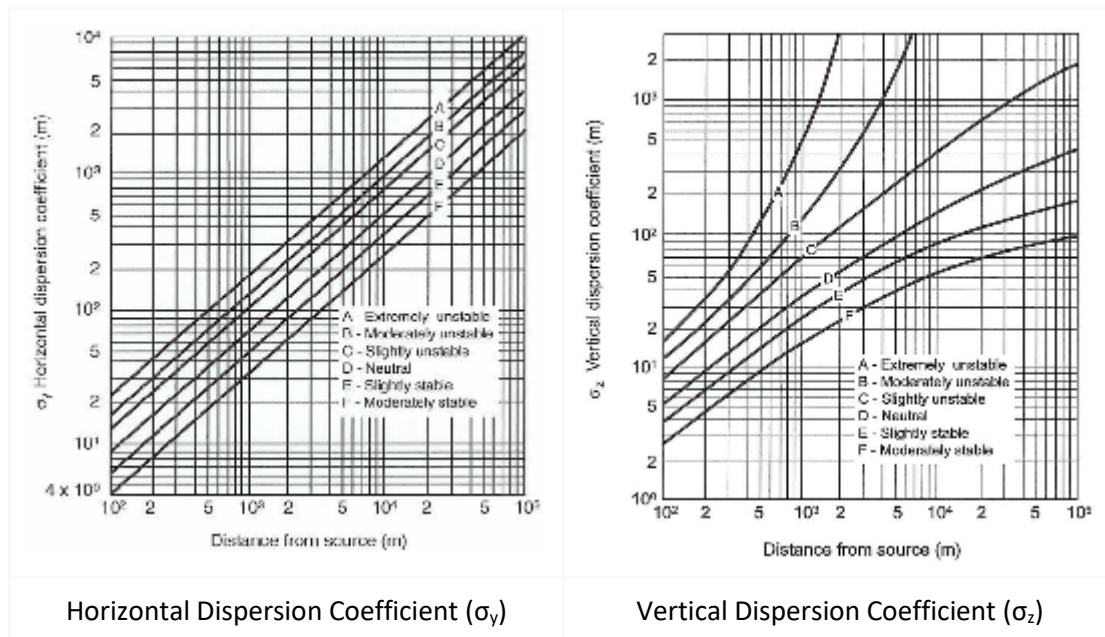


Figure 117: Curves of σ_y and σ_z for stability classes as a function of distance from the source (Varma, M. S. A. K. 2014).

Gaussian plume models operate under the assumption that smoke moves in a straight line during steady-state, homogeneous conditions. Steady-state models are capable of calculating concentrations for every hour by utilizing an emission rate as well as meteorological conditions that remain consistent throughout the entire modeling domain. Consequently, the models are able to simulate average concentrations on an hourly basis (Good practice, 2004). However, regions with varying weather conditions may violate these assumptions and decrease the reliability of the outcomes. Nonetheless, one benefit of plume models is that they do not demand a comprehensive meteorological input and are highly advantageous when there is a shortage of meteorological information available (Goodrick et. Al, 2012).

7.2.1.2. Implementation

A Gaussian plume model has been developed in Python to predict hourly smoke dispersion, leveraging the educational material provided in (Connoly, 2023). This model allows to capture the variation of effects caused by wind fluctuations and speed, vertical stability, and existence of multiple fire sources on smoke behavior and ground level concentrations. This Gaussian plume model requires the input of values related to the number, coordinates and effective height of the fire sources, the pollutants mass in grams / second emitted by each source and the meteorological conditions in the field, like the speed and direction of the current wind blowing. The following are illustrative examples that illustrate the operation of the proposed model.

Differing wind fluctuations

In the first set of experiments, the impact of assumptions regarding wind direction on the spread of pollutants is demonstrated. Normally, wind speed and direction are derived from observational data or a product of a forecast model. However, for the purposes of this set of experiments, we generate a synthetic dataset using one of three methods:

- a constant wind direction (135°),
- a completely random wind direction, or
- a prevailing wind direction (135°) with some variability on either side

The wind speed is set to 10 m/s. It is also assumed that the effective height of the wildfire source is 50 m and 40 g/s particulate matter PM10 are released. These scenarios are tested for neutral vertical stability and the burning event takes place in northern Euboea. The respective hourly outcomes are presented in Figure 118 . The colors correspond to European Air Quality Index²¹ levels.

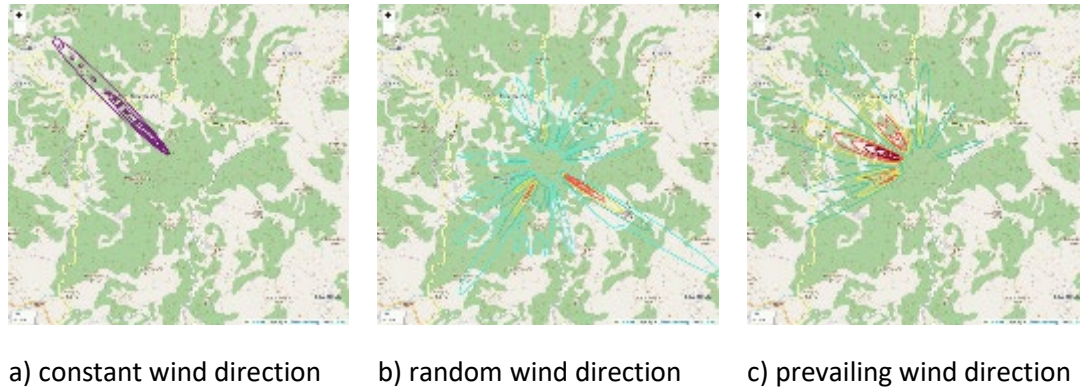


Figure 118: Differing wind fluctuations

Multiple wildfire sources

The effects of multiple wildfire sources on the ground level concentrations are then investigated. In this case, the prevailing wind is adopted as a basis for study, where the presence of two or three different sources exhibiting identical effective height and quantity of pollutants released (i.e., $H = 50\text{m}$ and $Q = 40\text{ g/s}$), is thoroughly examined. The rest of the input values are kept as above. The effects that including two or three sources has on ground level concentrations of pollutant (PM10) are shown in Figure 119.

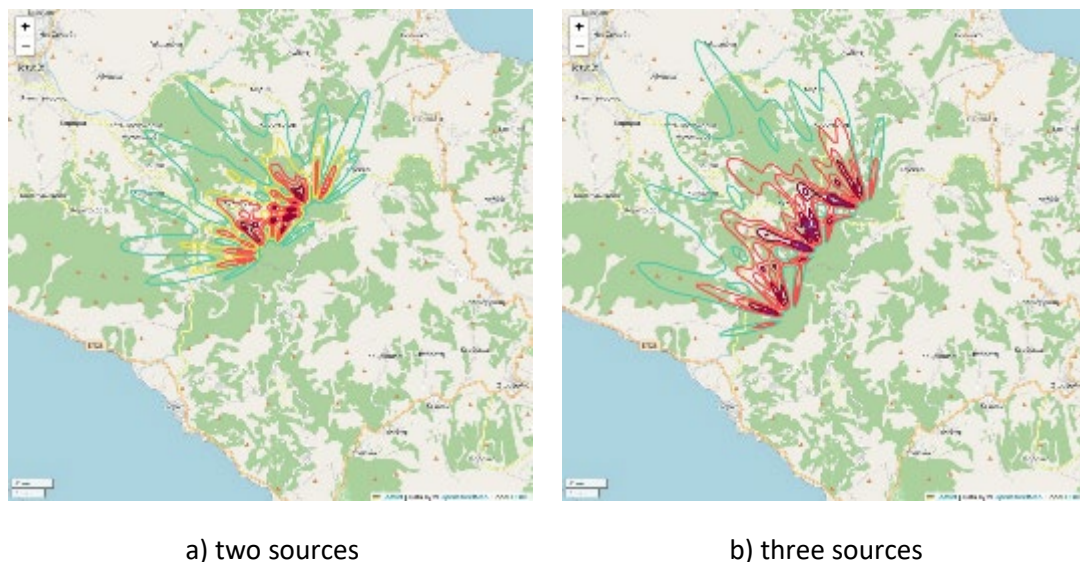


Figure 119: Multiple wildfire sources

Vertical stability

Finally, we want to investigate the effects of the vertical stability of the atmosphere on the ground level concentrations of pollutants. In this set of experiments, a wildfire source is

²¹ <https://airindex.eea.europa.eu/Map/AQI/Viewer/>

adopted ($H = 50\text{m}$ and $Q = 40\text{ g/s}$), and the vertical stability takes all possible values according to Pausquill (i.e., A - F), under the same meteorological conditions. Figure 120 demonstrates the outcomes of the proposed model.

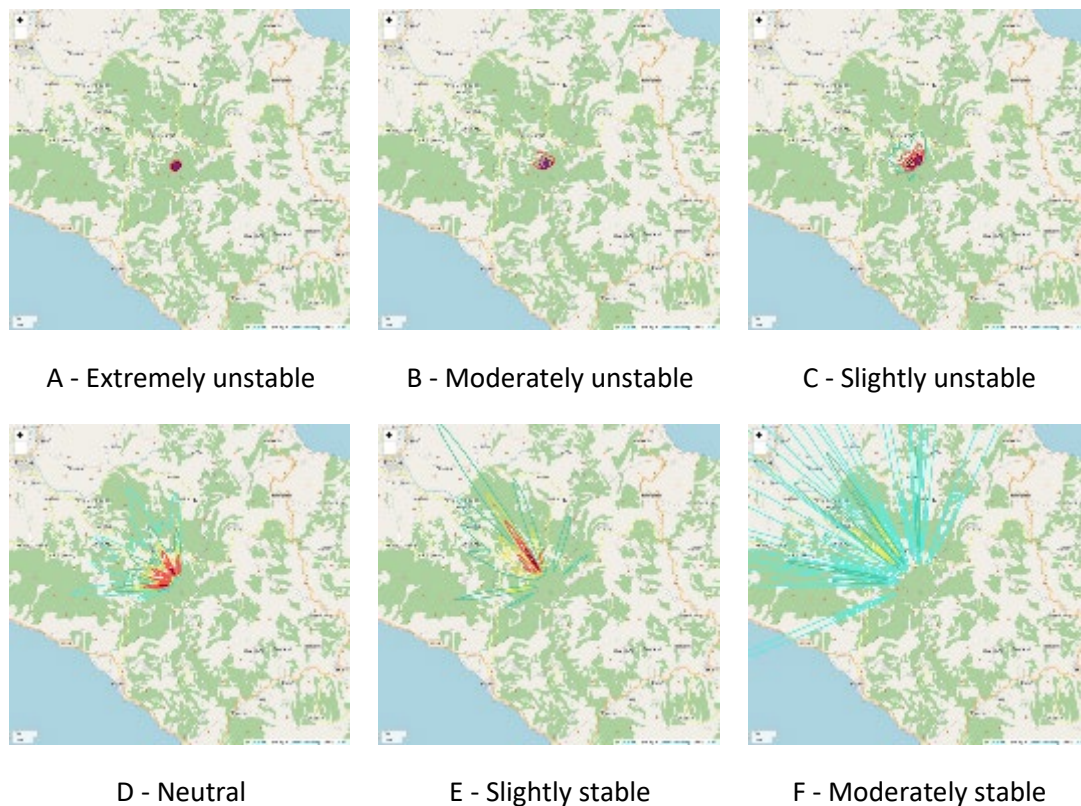


Figure 120: Vertical stability effects

7.2.2. Evacuation route planning

In this section, we outline the process of creating a strategy to safely relocate individuals from hazardous zones to designated safety areas. This comprehensive planning process encompasses the following key components: **a) Risk Assessment:** Evaluating potential hazards and threats; **b) Population Analysis:** Assessing the demographic characteristics of the affected population; **c) Definition of Safe Areas:** Identifying secure locations for evacuees; **d) Route Selection:** Determining the most suitable evacuation routes; **e) Simulation Testing:** Conducting simulated evacuation tests for evaluation.

7.2.2.1. Risk assessment - Module triggering

The organized preventive removal of citizens (evacuation) is an action that can be implemented promptly in areas where a catastrophic event is anticipated. This measure is taken to protect the lives and health of the citizens. The risk of staying within these vulnerable areas outweighs the risk of relocating to a safe place. The command centers based on the collected data from the appropriate repositories and modules concerning the fuel characteristics and the topography of the affected area, the meteorological conditions, the fire spread forecast, the ambient air quality and the smoke dispersion forecast assess the wildfire event intensity and the subsequent wildfire hazard. The decision to formulate evacuation plans is determined by the estimated levels of risk and hazard. When necessary, the command centers trigger (by placing a new emergency message on the queue) the corresponding module to design safe and appropriate evacuation routes (Figure 121).

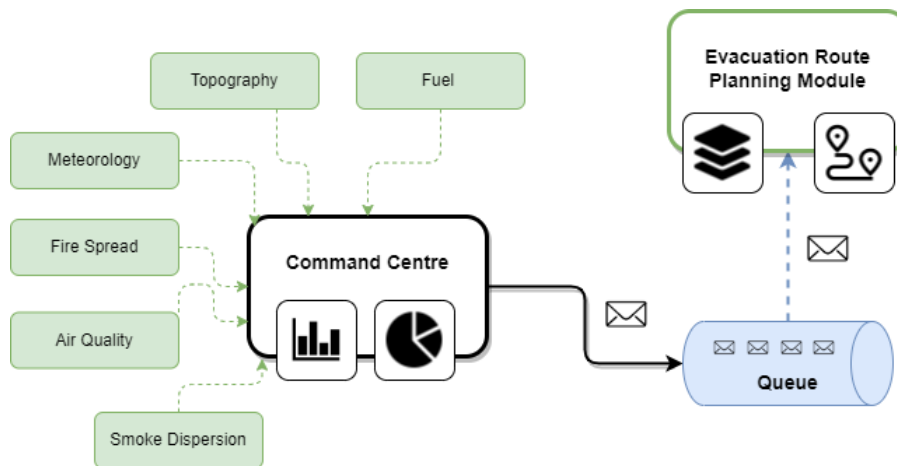


Figure 121: Evacuation Route Planning Module Triggering

7.2.2.2. Module overview

In this section the Evacuation Route Planning module's inputs, operation and outputs are thoroughly demonstrated. A comprehensive description is presented in Figure 122.

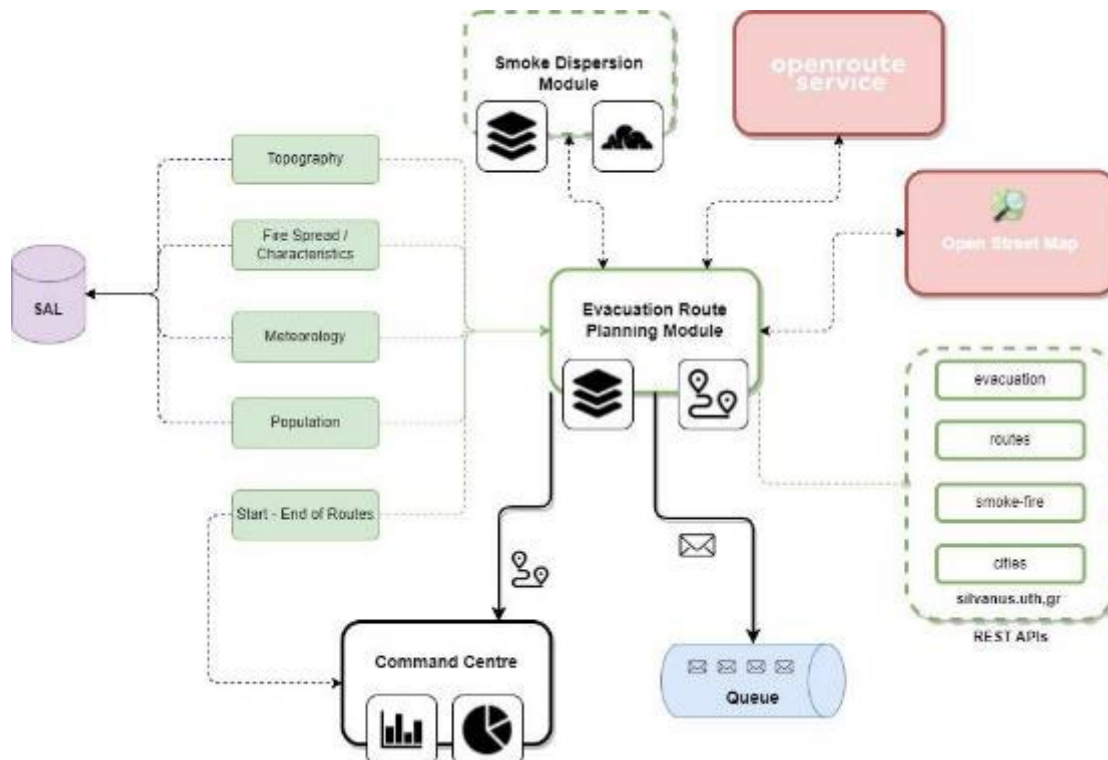


Figure 122: Evacuation Route Planning Interactions

7.2.2.2.1. Module inputs

When an emergency message is produced, the evacuation route planning component is activated, initiating a search within the SILVANUS Cloud to retrieve all the necessary inputs. The following list depicts all the required inputs, along with their corresponding values, used in our experimental evaluation (if an input is not currently available in the SAL we create synthetic data).

1. Ambient Area Features

An accurate planning of evacuation routes requires the detailed description of the forms and characteristics of the area where the wildfire phenomenon takes place. Usually, these

features are depicted in the form of maps. For demonstration purposes, we have chosen the island of Euboea (SILVANUS pilot site) and collected various physical features such as roads and buildings from OpenStreetMaps (OSM). Selecting an island as the primary input for evaluating the evacuation planning component offers several distinct advantages since **a)** Euboea provides a controlled and isolated environment, ideal for modeling and analyzing evacuation scenarios without significant external interference, **b)** Euboea presents unique challenges related to evacuation due to their geographical features, such as limited access points and varying population densities, and **c)** Euboea has a unique escape route to the rest of Greece, allowing us to study congestion phenomena when multiple residents need to evacuate promptly. Implementation is performed in Python and the folium library is used to create several types of Leaflet maps.

2. Smoke Dispersion and Fire Behaviour - Meteorology

In addition, it is vital to determine the hazards and threats to the lives and health of nearby citizens and first responders as a result of wildfires. The risk posed to individuals arises from the fact that their bodies are subjected to extreme heat caused by the burning of biomass, as well as from the release of combustion derivatives that diffuse into the atmosphere. The smoke produced consists of a variety of suspended particles. The composition of the mixture is determined by the type of biomass that is burning. The health effects of breathing in these particles vary according to their characteristics, the degree and duration of exposure, the state of the surrounding air, the overall level of health of individuals, as well as how each person's body reacts to them (Guidelines, 2013).

In order to estimate the wildfire and smoke behavior, the *evacuations/smoke-fire* endpoint has been developed. This endpoint implements a typical wildfire spread scenario and the subsequent estimation of the smoke dispersion adopting the aforementioned Gaussian plume model and taking into account the inputs concerning the required meteorological conditions (i.e., direction and speed of wind, and humidity level). When called, it receives the arguments and executes the calculations. The result is then saved as a Features Collection of GeoJSON format and the response with the JSON file is sent back to the caller. A demonstrative example is presented in Figure 132.

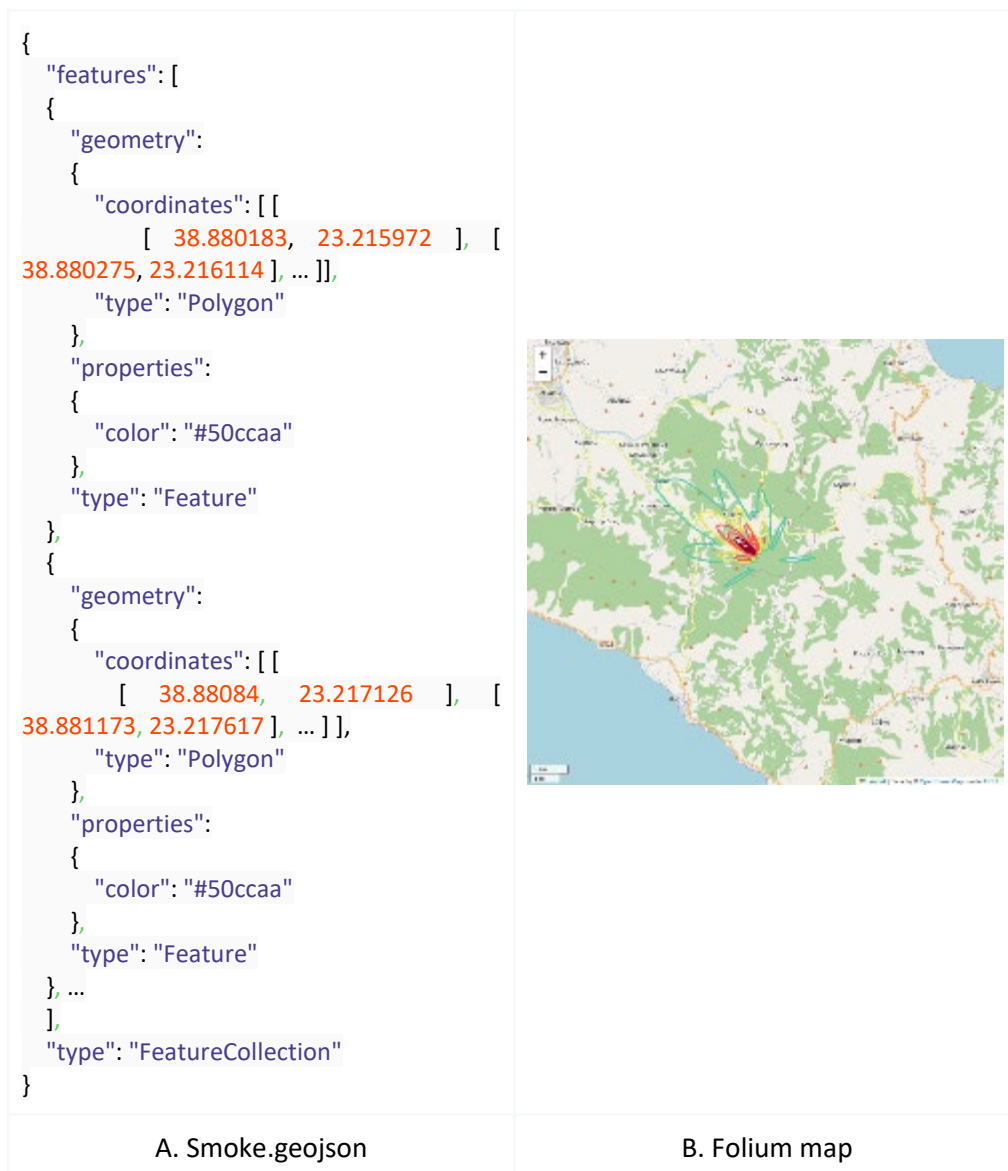


Figure 123: GeoJSON output example with corresponding map

In order to access this endpoint, the user must make a POST request, passing the needed arguments to the body of the request, as shown in the cURL in Figure 133. The arguments that the user needs to pass are shown in Table 20. There are some optional parameters, which take specific values and set to a default value if none is set. *stability* takes values from the interval 1 to 6, corresponding to how normal the atmosphere is. *wind* takes one of the values “constant_wind”, “fluctuating_wind” and “prevailling_wind”. *days* takes any non-negative integer

```

curl -X 'POST' \
  'http://silvanus.uth.gr/evacuations/smoke-fire' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d
'lat=38.892851059647484&lon=23.23213064855875&list_source_x=0.0&list_source_x=0.0&list_source_y=0.0&list_source_y=2000.0&list_Q=40.0&list_Q=40.0&list_H=50.0&list_H=50.0&ph_wind_speed=10.0&ph_wind_dir=135&rh=0.9&&stability=4&wind=prevailing_wind&days=1'

```

Figure 124: POST request to /evacuations/smoke-fire endpoint (2 sources)

Table 18: Parameters of /evacuations/smoke-fire endpoint

Parameter Name	Type	Description
lat	float	The latitude of the observation point.
lon	float	The longitude of the observation point.
list_source_x	float array	List of the x-coordinates of fire outbreak points relative to the observation point in meters.
list_source_y	float array	List of the y-coordinates of fire outbreak points relative to the observation point in meters.
list_Q	float array	List of masses emitted per unit time for each fire outbreak in grammars per second.
list_H	float array	List of effective source heights for each fire outbreak in meters.
ph_wind_speed	float	The speed of the wind in meters per second.
ph_wind_directions	int	The direction of the wind in degrees.
rh	float	The relative humidity of the atmosphere.
stability	integer, optional	The stability of the atmosphere. Default value is 4.
wind	string, optional	The wind field. Default value is "prevailing_wind".
days	integer, optional	The number of the days for which the model calculates.

3. Population

Smoke from a wildfire could potentially induce ocular and respiratory tract irritation, bouts of coughing, bronchitis, as well as aggravate or precipitate asthma symptoms, alongside instances of respiratory insufficiency, particularly marked by dyspnea. Furthermore, it can potentially contribute to cardiac arrhythmias and, in severe scenarios, culminate in cardiovascular incidents resulting in death in people with underlying diseases. Hence, in situations where there is a perceived need to protect individuals from the effects of smoke emitted by biomass burning, specifically those who are vulnerable, such as individuals with respiratory problems or cardiovascular diseases, as well as the elderly, children, infants, and pregnant, it is imperative for them to promptly relocate from the affected area. Individuals are confronted with an additional hazard, namely the potentiality of thermal burns, resulting from the body's direct exposure to flames or hot gasses. It is evident that attempting to move away, while being exposed to elevated temperatures and noxious composition of the wildfire products in an open area along its trajectory, reduces an individual's likelihood of survival, regardless of age and health state (Guidelines, 2023).

To provide a demonstration of the overall module operation, the examination of the demographic composition of the towns and villages in the northern region of Euboea is undertaken, with the information being derived from the most recent census carried out in this area. It is necessary to point out that data could well be drawn from WorldPop²² project repositories as well.

4. Safe Gathering Places - Refuge Destinations

A crucial aspect of the evacuation route planning process is the identification of safe and suitable places both for gathering individuals who are confronted with wildfire hazard and for seeking refuge, finding temporary accommodations, and receiving administrative support. It is imperative to ascertain, from the command center, the starting and ending points of the alternative evacuation routes.

During the experiments, numerous combinations of these two distinct points from the northern Euboea region are tested.

7.2.2.2.2. Module operation

The calculation of the evacuation routes is done using the OpenRouteServices API²³. OpenRouteServices offers a variety of routing services, like directions, isochrones, time-distance matrices etc, in the form of endpoints. From these services, the directions endpoint is used for the calculation of the evacuation routes. This service offers the capability to get the optimal route from a given set of points as well as a list of optional parameters for the surround. The set of points consists of two or elements: the *initial point* and *one or more points of destinations*. The results are returned by default in a custom JSON format, but it can also be set to either GeoJSON or GPX. Since the most flexible format was GeoJSON, we chose it as the main format for returned files.

Figure 134 shows the response from a simple call to the endpoint. The initial point is Istiaia and the destination point is Aidipsos. The field "coordinates" inside "geometry" contains all the points of the linestring representing the route.

²² <https://www.worldpop.org>

²³ <https://openrouteservice.org/>

```
{
  "type": "FeatureCollection",
  "metadata": {
    "attribution": "openrouteservice.org | OpenStreetMap contributors",
    "service": "routing",
    "timestamp": 1694606498382,
    "query": {
      "coordinates": [
        [
          23.1515895799512,
          38.952002738068195
        ],
        [
          23.043416098102462,
          38.878264801957656
        ]
      ],
      "profile": "driving-car",
      "format": "geojson"
    },
    "engine": {
      "version": "7.1.0",
      "build_date": "2023-07-09T01:31:50Z",
      "graph_date": "2023-09-03T10:10:37Z"
    }
  },
  "bbox": [
    23.039025,
    38.87848,
    23.151579,
    38.956633
  ],
  "features": [
    {
      "bbox": [
        23.039025,
        38.87848,
        23.151579,
        38.956633
      ],
      "type": "Feature",
      "properties": {
        "transfers": 0,
        "fare": 0,
        "segments": [
          {
            "distance": 18635.8,
            "duration": 1524.9,
            "steps": [
              {
                "distance": 38.6,
                "duration": 9.3,
                "type": 11,
                "instruction": "Head west on Δήμαρχος Ιωάννου Αντωνίου",
                "name": "Δήμαρχος Ιωάννου Αντωνίου",
                "way_points": [
```

```

        0,
        2
    ]
  },
  {
    "distance": 158.3,
    "duration": 35.6,
    "type": 1,
    "instruction": "Turn right",
    "name": "-",
    "way_points": [
      2,
      8
    ]
  }, ...
]
}
],
"way_points": [
  0,
  380
],
"summary": {
  "distance": 18635.8,
  "duration": 1524.9
}
},
"geometry": {
  "coordinates": [
    [
      23.151579,
      38.95192
    ],
    [
      23.151424,
      38.951931
    ], ...
  ],
  "type": "LineString"
}
}]
}

```

Figure 125: GeoJSON response from simple directions endpoint call.

The case above is a very simple call, with only the initial point and point of the destination. This case does not consider any restriction that the user might face. More specifically, in the case of a wildfire, it doesn't consider the smoke and the fire, the profile of the vehicle or the preferred route (fastest or shorter) and so on. In order to use this information, the directions endpoint provides a list of customized parameters.

Figure 135 shows the response of a more complex call. The initial point and the point of the destination are the same as before, but additional parameters have also been included: the profile has been set to driving car, the preferred route is set to shortest and the instructions have been disabled. The most important added parameter is an object to be avoided.

Specifically, a list of points representing the obstacle is added as a parameter in the endpoint call. This list shall be taken into account by the route's estimation, producing a route that avoids that obstacle.

```
{
  "type": "FeatureCollection",
  "metadata": {
    "attribution": "openrouteservice.org | OpenStreetMap contributors",
    "service": "routing",
    "timestamp": 1694611018512,
    "query": {
      "coordinates": [
        [
          23.1515895799512,
          38.952002738068195
        ],
        [
          23.043416098102462,
          38.878264801957656
        ]
      ],
      "profile": "driving-car",
      "format": "geojson",
      "preference": "shortest",
      "options": {
        "avoid_polygons": {
          "coordinates": [
            [
              [
                23.04273091613799,
                38.93712480960088
              ],
              [
                23.096165358787243,
                38.94679002083246
              ],
              ...
            ]
          ],
          "type": "Polygon"
        }
      }
    },
    "engine": {
      "version": "7.1.0",
      "build_date": "2023-07-09T01:31:50Z",
      "graph_date": "2023-09-03T10:10:37Z"
    }
  },
  "bbox": [
    23.04301,
    38.859552,
    23.152158,
    38.952458
  ],
  "features": [
```

```

{
  "bbox": [
    23.04301,
    38.859552,
    23.152158,
    38.952458
  ],
  "type": "Feature",
  "properties": {
    "transfers": 0,
    "fare": 0,
    "way_points": [
      0,
      1115
    ],
    "summary": {
      "distance": 27575.2,
      "duration": 4116.1
    }
  },
  "geometry": {
    "coordinates": [
      [
        23.151579,
        38.95192
      ],
      [
        23.151424,
        38.951931
      ],
      ...
    ],
    "type": "LineString"
  }
}
]
}

```

Figure 126: GeoJSON response from customized parameters.

In general, the second showcase of the destination's endpoint call is the approach that is taken, due to better performance in calculations and ease of use. It expands the route calculation, providing the capability of obstacle avoidance, a very important feature that is needed, in order for the population to evacuate safely, without facing the fire or the smoke.

When calling the endpoint like that, it is important to clarify what the obstacles are. The obstacles that need to be avoided are the wildfire and the smoke. When the smoke or fire cover the route, the passage of vehicles becomes impossible and someone who is within the fire zone is in immense danger. Their representation is done using polygons. The polygons consist of a list of points representing its corners, with each point having specific longitude and latitude values. These polygons are passed as parameters in the endpoint call, along with the other OpenRouteServices parameters. For example, in Figure 136, the field "coordinates" of the field "avoid_polygons" contains a list of points that represent a simple polygon that blocks the shortest route between Istaia and Aidipsos.

Besides these two endpoints, another two endpoints have been developed. The first one is responsible for returning information about a city. Specifically, its purpose is to return a GeoJSON format file with multiple information about the requested city. Currently, the file contains information about city's perimeter, center and population. The access to this endpoint is done, using a GET request to `/evacuations/cities?city=<city_name>`. On a successful request, it returns the proper status code along with the GeoJSON file. If it fails, an error message is returned to the caller, informing him about their possible mistake. Figure 136 demonstrates a cURL call to this endpoint.

```
curl -X 'GET'  
'http://silvanus.uth.gr/evacuations/cities?city=Taxiarhis'  
-H 'accept: application/json'
```

Figure 127: cURL call to `/evacuations/cities` endpoint

The second endpoint is responsible for acquiring the route between two cities. This endpoint returns a GeoJSON file that contains information about the route between two cities. We assume that the direction of the route doesn't affect the route itself, so going from one city to another is the same as going in reverse. This means that the order of the cities doesn't change the final route. The access to this endpoint is done, using a GET request to `/evacuations/routes?from=<initial_city>&destination=<destinated_city>`. These two parameters are used to find an already estimated route between them that is located within the server. If the query was successful, the endpoint returns the proper status code with the GeoJSON file with the route. If it fails, it returns an error message, informing the caller about the error. Figure 137 represents a call to this endpoint.

```
curl -X 'GET'  
'http://silvanus.uth.gr/evacuations/routes?from=Agdines&destination=Achladhi'  
-H 'accept: application/json'
```

Figure 128: cURL call to `/evacuations/routes` endpoint

The operation of the model is divided into four phases. In the first phase the cities within the risk area are retrieved from the system. The second phase records all the available routes for all pairs of the cities from the previous phase. The third phase estimates the fire spread and smoke dispersion in the affected area of interest. The fourth phase estimates all the safe routes for evacuating the cities in danger.

The first phase starts by feeding the `/evacuations/cities` endpoint with a list of city names. This action results in a list of GeoJSON files each of which represents a specific city. For each city pair the `/evacuation/routes` endpoint is called and returns information for the route in GeoJSON form. The third phase estimates the progression of the fire and the smoke. It is independent from the previous phases, which can be useful if only the dispersion is needed. For that phase, the parameters discussed in 7.2.2.1 must be used. The Gaussian plume model, as described in 7.2.1, computes the fire spread and smoke dispersion. This phase uses the `/evacuations/smoke-fire` endpoint in order to calculate a GeoJSON file of the model. The fourth phase produces the final output of the component. It produces a list with routes that are characterized as safe or unsafe. Using the wildfire dispersion model from phase three, it forecasts where the wildfire and smoke are directed, and using the routes from phase two, it

evaluates the safety of each route. The latter evaluation is accomplished using the `/evacuations/evacuation` endpoint in GeoJSON form.

7.2.2.2.3. Module outputs

The system provides three outputs. The first output consists of a set of evacuation recommendations for a group of cities, along with suggested evacuation times (timestamps). The remaining two outputs are two lists, each containing sets of safe and unsafe paths in the GeoJSON format. More details are presented in section 7.3.2.3.

7.2.2.3. Simulation testing

7.2.2.3.1. Experimental evaluation

In total, we present 2 experiments. In the first experiment, we consider the following parameters: a) Ignition point at location 38.925252233969054, 23.12973430941135, b) Mass emitted per unit time is 40.0 g/s, c) Source height is 50.0m, d) Wind speed is 10 m/s and e) East Wind. Additionally, we assume that each iteration step is equivalent to one hour, and we forecast the fire behavior for 3 hours from 2023-11-15 09:30:00 to 2023-11-15 12:30:00.

Figure 138a displays the 22 cities utilized in our experiments, while Figure 138b illustrates all the available routes between every pair of cities prior to the fire incident.

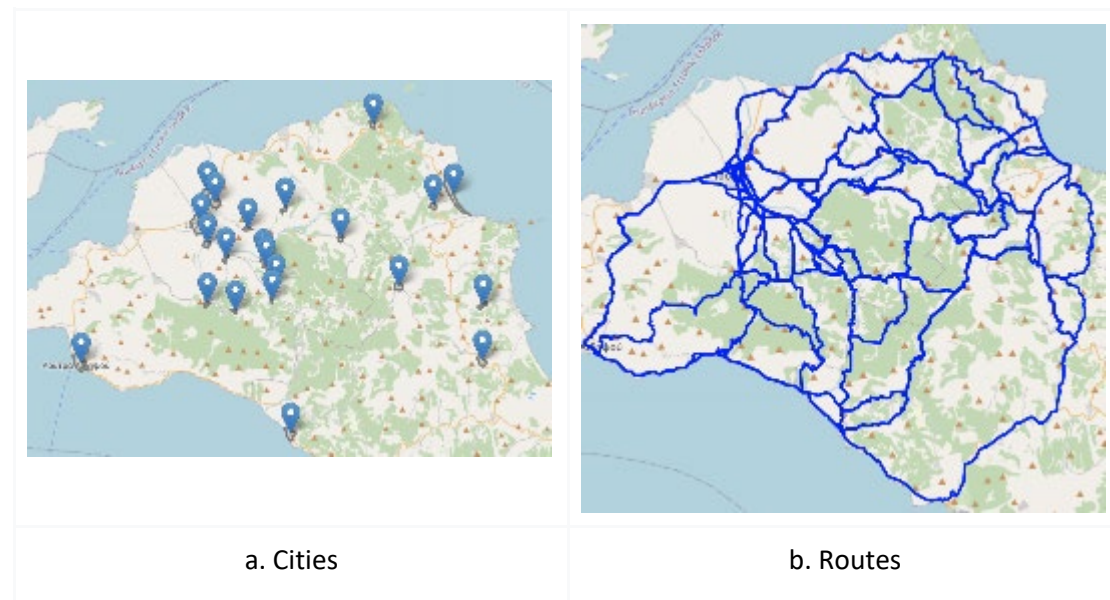


Figure 129: Cities and available routes in Northern Euboea

Figure 139 illustrates the progression of the fire incident and the dispersion of smoke within one hour (Exp.1). The results of the evacuation route planning component for a three-hour simulation execution time are presented in Figure 140, where green and red routes denote safe and unsafe paths, respectively. Additionally, the component identifies the necessity for emergency evacuation in a particular city. This information is generated when there is no safe route available between two cities. The output of the component is as follows:

- Evacuate AgiosGeorgiosIstiaias time: 2023-11-15 09:30:00
- Evacuate Kastaniotissa time: 2023-11-15 09:30:00
- Evacuate Kamaria time: 2023-11-15 10:30:00
- Evacuate Kamatriades time: 2023-11-15 10:30:00
- Evacuate Istiaia time: 2023-11-15 11:30:00
- Evacuate Avgaria time: 2023-11-15 11:30:00
- Evacuate Monokaria time: 2023-11-15 11:30:00

- Evacuate KatoMonokaria time: 2023-11-15 11:30:00

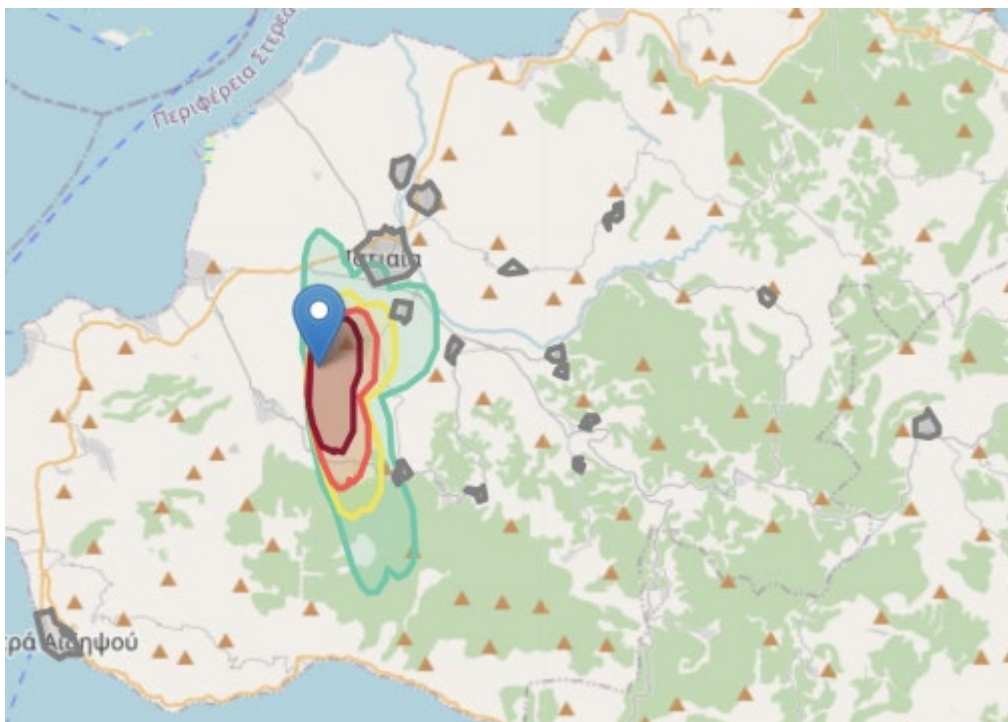


Figure 130: Fire spread and smoke dispersion progression (Exp.1)

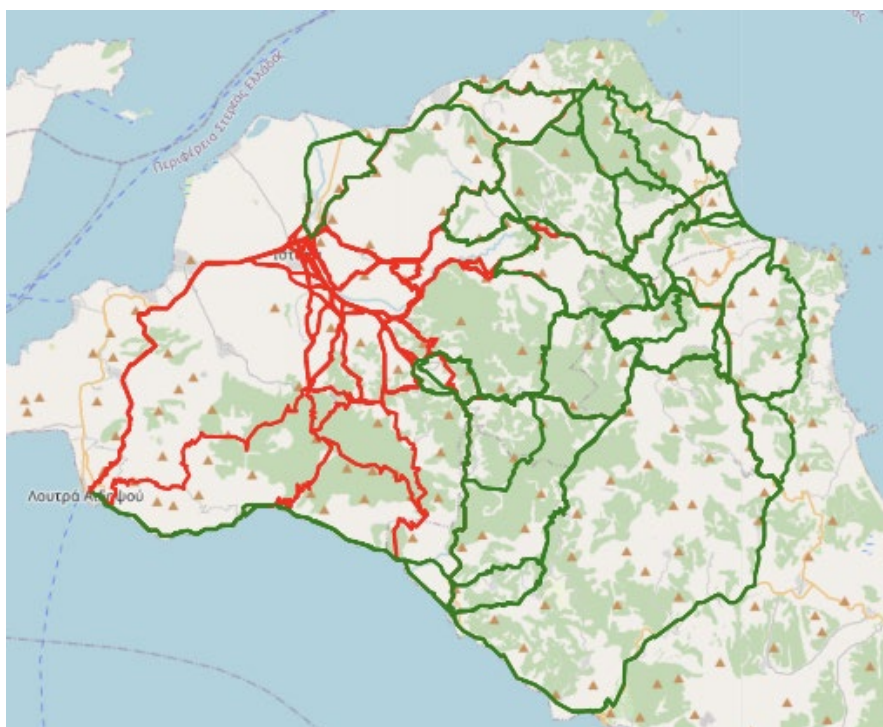


Figure 131: Routes characterization (Exp.1)

In the next experiment (Exp.2), we create a severe wildfire incident and add four more cities as depicted in Figure 141a while the wind direction is changed in an opposite direction - west wind (Figure 141b). The component recommends immediate evacuation for the cities of Agios Georgios and Lichada in the first simulation step, as there is no feasible route to bypass the fire incident. Residents of the remaining two cities can readily evacuate, provided the wind direction changes, by utilizing the green routes illustrated in Figure 142.

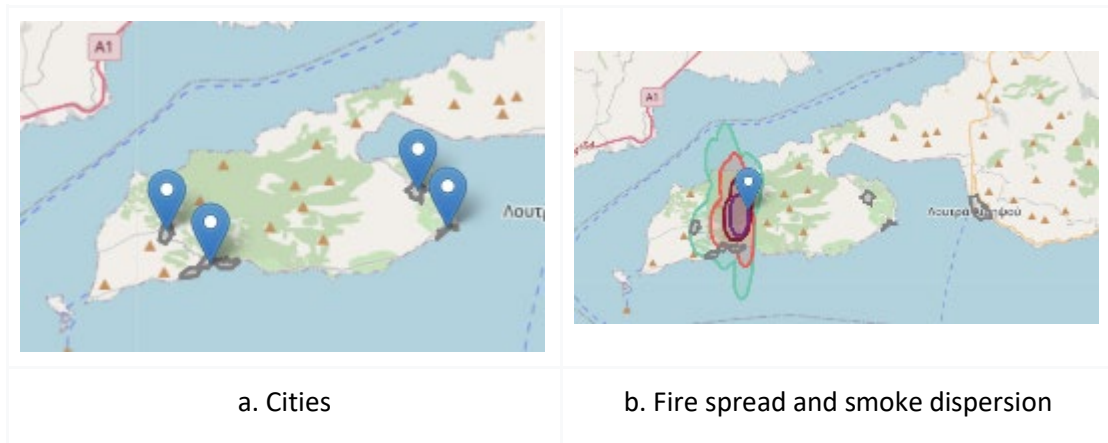
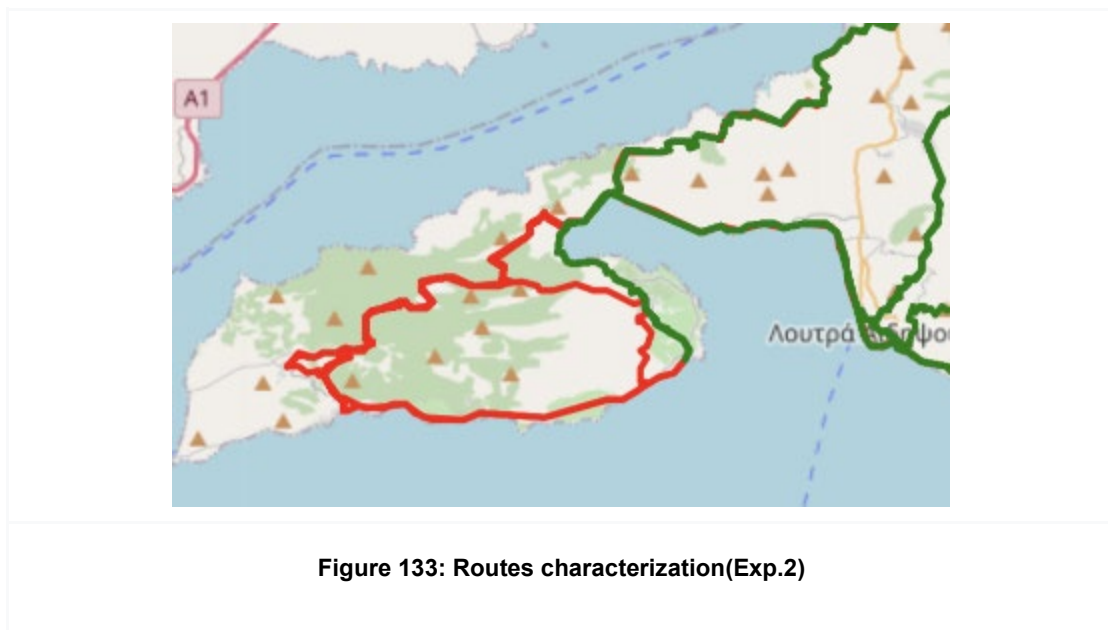


Figure 132: Exp.2 setting



7.2.2.3.2. Output format

Figure 143 displays a sample GeoJSON file for the Gialtra - NeaSinasos route (Figure 143a). This route exhibits numerous similarities with the route obtained from Google Maps (Figure 143b), with both routes having the same average duration and distance.



a. OSM



b. Google Maps

Figure 134: Routes comparison

```

{
  "type": "FeatureCollection",
  "features": [
    {
      "bbox": [
        22.947754,
        38.863502,
        23.16447,
        38.968077
      ],
      "geometry": {
        "coordinates": [[
          22.97426,
          38.863502
        ],
          [
            22.974283,
            38.863558
          ],
          [...]
        ],
        "type": "LineString"
      },
      "properties": {
        "fare": 0,
        "summary": {
          "distance": 30916.6,
          "duration": 3838.2
        },
        "transfers": 0,
        "way_points": [
          0,
          735
        ],
        "start": "Gialtra",
        "end": "NeaSinastos",
        "characterization": "safe",
        "from": "2023-11-15 09:30:00",
        "to": "2023-11-15 12:30:00"
      },
      "type": "Feature"
    }
  ]
}

```

Figure 135: GeoJSON sample of a route

The above GeoJSON file represents an evacuation route from Gialtra to NeaSinastos. The route is characterized as "safe" and is designed for use during the specified time period from "2023-11-15 09:30:00" to "2023-11-15 12:30:00". It includes geographic coordinates that define a LineString geometry, detailing the path of the route. The route has a total distance of 30,916.6 meters and an estimated duration of 3,838.2 seconds. It involves no transfers, making it a direct route. The "fare" is listed as 0, and the "way_points" indicate the start and end points of the route within the coordinate sequence.

7.3. Component integration

Regarding component integration, pilots possess the ability to seamlessly integrate our system into their operations. This accomplishment is attained through the initiation of HTTP access by means of our RESTful APIs. Elaborate information regarding the API can be found in the preceding section, and information can also be obtained through the documentation available in the provided link: <http://silvanus.uth.gr/documentation>. The results of the evacuation route planning can be easily stored in the Storage Abstraction Layer (SAL) for the Greek pilot case. For testing in other pilots, the module can be seamlessly integrated after a preprocessing step, which involves recording cities characteristics and generating the initial available routes before a fire incident occurs.

7.4. Plans for future extensions

In the first place of our future agenda is to conduct practical testing of the showcased evacuation planning component. The testing will be conducted in collaboration with a local fire department, which will orchestrate a simulated fire incident. Also, we will connect the module to the SILVANUS message queue on one or more appropriate brokers and test its performance. In addition, it is anticipated that the module's functionality will be extended to pay specific attention to the management of vulnerable groups (i.e., individuals with respiratory problems or cardiovascular diseases, elderly, children, pregnant, etc.) and load balancing among alternative safe routes, while data will be drawn from WorldPop. Finally, the possibility of integrating data with the SILVANUS Storage Abstraction Layer in the form of json files will be investigated.

8. Conclusions

This deliverable includes an extensive report on the software components that have been developed as part of the SILVANUS platform that contribute to the toolkit development of a decision support system (DSS) to improve situational awareness. The SILVANUS DSS tools aim to assist decision makers to make informed decisions reacting faster to wildfires, optimizing response coordination and allocation of resources reducing wildfire impact first of all on human health and secondly on valuable infrastructures. Decision support is based on automated data gathering and knowledge extraction and management based on intelligent algorithms. The SILVANUS platform facilitates the gathering of such valuable data either from sensors integrated on the system platform or external sources. Communication and interaction of all components is facilitated by the message broker and the Storage Abstraction Layer (SAL), which are part of the SILVANUS platform middleware.

Systematic data processing and semantic annotation for extracting and managing high-level knowledge related to forest management and wildfire incidents is achieved by the Knowledge Base Information Fusion (KBIF) presented in Section 2. KBIF facilitates extraction of knowledge based on fusion through the semantic representation and filtering of data gathered from in-situ sensors, combined with data retrieved from external sources or generated by the SILVANUS platform components at a second level leading to systematic data classification and detection of relationships among the generated data that can lead to improved situational awareness. Stochastic modelling of such data, as presented in Section 3, can lead to improved preparedness and analysis of GIS related data that can present in a user-friendly manner important area characteristics that can be exploited for reacting faster and more efficiently to protect human lives and the environment. The detection of wildfire events that take place in areas under monitoring and inspection apart from in-situ sensors can also be facilitated via the analysis of human descriptions made available over social media that generate a lot of data that can be exploited towards this end. Machine Learning based approaches to achieve the above objective in a multilingual context have been presented in Section 5. Upon detection of a wildfire event response coordination must focus on two main areas: 1) early and efficient distribution of resources to suppress fires and 2) protection of human lives by combating the wildfire and evacuating citizens from dangerous areas. Section 3 presents algorithms to solve a number of multiparametric problems that are related to the resource distribution under several constraints that need to be met simultaneously. Such multi-objective optimization problems are hard to be solved by humans, especially under the pressure of a fire-spreading event and the solution presented in Section 3 assist in this direction. Finally, protection of human lives requires the identification of potentially dangerous for humans environmental conditions and in that case evacuation of citizens in a safe way. Thus, in Section 5 we presented the Health Impact Component that detects potentially poisonous gases that can be harmful to fire-fighters in the field and citizens located in an area and in Section 6 a component that can calculate safe routes on a map that can be used securely as evacuation paths avoiding passing by areas that have been identified as dangerous for human life (either to fire or low air quality conditions).

The above tools have been developed and integrated in the SILVANUS platform. Their operation has been evaluated in different scenarios as presented in detail in this deliverable. The component implementation details have been described, accompanied by demonstration of the operation and the results obtained when using the tools. The plans for future extensions are finally described, defining the next development steps that will be reported in the following planned deliverables of WP5.

9. References

- Achtemeier GL, Jackson W, Hawkins B, Wade D, McMahon C (1998) The smoke dilemma: head-on collision! In 'Transactions of the Sixty-Third North American Wildlife and Natural Resources Conference', 20–24 March 1998, Orlando, FL. (Ed. KG Wadsworth) pp. 415–421. (Wildlife Management Institute: Washington DC)
- Agranat V. and Perminov V., "Mathematical modeling of wildland fire initiation and spread," *Environmental Modelling and Software*, vol. 125, no. January, p. 104640, 2020, doi: 10.1016/j.envsoft.2020.104640.
- Allaire F., Mallet V., and Filippi J. B., "Emulation of wildland fire spread simulation using deep learning," *Neural Networks*, vol. 141, pp. 184–198, 2021, doi: 10.1016/j.neunet.2021.04.006.
- Baines P. G., "Physical mechanisms for the propagation of surface fires," *Math Comput Model*, vol. 13, no. 12, pp. 83–94, 1990, doi: 10.1016/0895-7177(90)90102-S.
- Bao W., et. al., "Estimation of fuel load using remote sensing data in Hulunbuir Grassland," *Natural Hazards Research*, vol. 2, no. 4, pp. 375–383, Dec. 2022, doi: 10.1016/J.NHRES.2022.11.004.
- Chen W., et. al., "Wildfire risk assessment of transmission-line corridors based on naïve bayes network and remote sensing data," *Sensors (Switzerland)*, vol. 21, no. 2, pp. 1–16, 2021, doi: 10.3390/s21020634.
- Connolly, P. (2023, August 30). Gaussian Plume Model. University of Manchester. https://personalpages.manchester.ac.uk/staff/paul.connolly/teaching/practicals/gaussian_plume_modelling.html
- Depicker A., De Baets B., and Marcel Baetens J., "Wildfire ignition probability in Belgium," *Natural Hazards and Earth System Sciences*, vol. 20, no. 2, pp. 363–376, 2020, doi: 10.5194/nhess-20-363-2020.
- Dhall A., Dhasade A., Nalwade A., Mohan M. R., and Kulkarni V., "A survey on systematic approaches in managing forest fires," *Applied Geography*, vol. 121, no. November 2018, p. 102266, 2020, doi: 10.1016/j.apgeog.2020.102266.
- General Secretariat of Civil Protection (2023). Guidelines for the organized preventive evacuation of citizens for reasons of protection from developing or imminent disaster due to forest fires and instructions for the drafting of special plans in the context of the implementation of art. 23 par. 4 of Law 4662/2020. (available in greek)
- Gifford, F. A. (1961). Use of routine meteorological observations for estimating atmospheric dispersion. *Nucl. Safety*, 2, 47-51.
- Glaser J. and Halada L., "On elliptical model for forest fire spread modeling and simulation," *Math Comput Simul*, vol. 78, no. 1, pp. 76–88, 2008, doi: 10.1016/j.matcom.2007.06.001.
- Goodrick, S. L., Achtemeier, G. L., Larkin, N. K., Liu, Y., & Strand, T. M. (2012). Modelling smoke transport from wildland fires: a review. *International Journal of Wildland Fire*, 22(1), 83-94.
- Green, A. E. S., Singhal, R. P., & Venkateswar, R. (1980). Analytic extensions of the Gaussian plume model. *Journal of the Air Pollution Control Association*, 30(7), 773-776.

- Hernández L. Encinas, Hoya White S., Martín del Rey A., and Rodríguez Sánchez G., "Modelling forest fire spread using hexagonal cellular automata," *Appl Math Model*, vol. 31, no. 6, pp. 1213–1227, 2007, doi: 10.1016/j.apm.2006.04.001.
- Kalabokidis K. et al., "Virtual Fire: A web-based GIS platform for forest fire control," *Ecol Inform*, vol. 16, pp. 62–69, 2013, doi: 10.1016/j.ecoinf.2013.04.007.
- Kolanek A., Szymanowski M., and Raczyk A., "Human activity affects forest fires: The impact of anthropogenic factors on the density of forest fires in Poland," *Forests*, vol. 12, no. 6, p. 728, Jun. 2021, doi: 10.3390/F12060728/S1.
- Landau. D. and Binder. K., *A guide to Monte-Carlo Simulation in Statistical Physics-3rd ed.* 2009.
- Ma W., et. Al. , "Identifying Forest Fire Driving Factors and Related Impacts in China Using Random Forest Algorithm," *Forests 2020*, Vol. 11, Page 507, vol. 11, no. 5, p. 507, May 2020, doi: 10.3390/F11050507.
- Marlon J. R. et al., "Climate and human influences on global biomass burning over the past two millennia," *Nature Geoscience* 2008 1:10, vol. 1, no. 10, pp. 697–702, Sep. 2008, doi: 10.1038/ngeo313.
- Ministry for the Environment Manatū Mō Te Taiao, *Good Practice Guide for Atmospheric Dispersion Modelling*, www.mfe.govt.nz, ISBN: 0-478-18941-9 ME:522, Wellington, New Zealand, 2004.
- Navalho I., Alegria C., Quinta-Nova L., and Fernandez P., "Integrated planning for landscape diversity enhancement, fire hazard mitigation and forest production regulation: A case study in central Portugal," *Land use policy*, vol. 61, pp. 398–412, 2017, doi: 10.1016/j.landusepol.2016.11.035.
- Pasquill, F. (1961). The estimation of the dispersion of windborne material. *Meteoro. Mag.*, 90, 20-49.
- Reiter D., *The Monte Carlo Method, an Introduction*, Computational Many Particle Physics, Springer 2008.
- Sevinc V., Kucuk O., and Goltas M., "A Bayesian network model for prediction and analysis of possible forest fire causes," *For Ecol Manage*, vol. 457, no. October 2019, p. 117723, 2020, doi: 10.1016/j.foreco.2019.117723.
- Sivrikaya F. and Küçük Ö., "Modeling forest fire risk based on GIS-based analytical hierarchy process and statistical analysis in Mediterranean region," *Ecol Inform*, vol. 68, no. September 2021, 2022, doi: 10.1016/j.ecoinf.2021.101537.
- Storey M. A., Bedward M., Price O. F., Bradstock R. A., and Sharples J. J., "Derivation of a Bayesian fire spread model using large-scale wildfire observations," *Environmental Modelling and Software*, vol. 144, no. July, p. 105127, 2021, doi: 10.1016/j.envsoft.2021.105127.
- Tatem, A. WorldPop, open data for spatial demography. *Sci Data* 4, 170004 (2017). <https://doi.org/10.1038/sdata.2017.4>.
- Tien D., Van Le H., and Hoang N., "Ecological Informatics GIS-based spatial prediction of tropical forest fire danger using a new hybrid machine learning method," vol. 48, no. April, pp. 104–116, 2018.
- Turner, D. B. (1970). *Workbook on atmospheric dispersion estimates report Ap-26*. Research Triangle Park, NC: US EPA.

- Varma, M. S. A. K. (2014). *Mathematical Modeling of Air Pollution in a Thermal Power Project* (Doctoral dissertation, Sri Venkateswara University Tirupati).
- W. Jiang et al., "Modelling of wildland-urban interface fire spread with the heterogeneous cellular automata model," *Environmental Modelling and Software*, vol. 135, p. 104895, 2021, doi: 10.1016/j.envsoft.2020.104895.
- Yengoh G. T., et al., "Use of the Normalized Difference Vegetation Index (NDVI) to Assess Land Degradation at Multiple Scales," 2016, doi: 10.1007/978-3-319-24112-8.
- Yonezawa C., "Maximum likelihood classification combined with spectral angle mapper algorithm for high resolution satellite imagery," <http://dx.doi.org/10.1080/01431160701373713>, vol. 28, no. 16, pp. 3729–3737, 2007, doi: 10.1080/01431160701373713.
- You W. et al., "Geographical information system-based forest fire risk assessment integrating national forest inventory data and analysis of its spatiotemporal variability," *Ecol Indic*, vol. 77, pp. 176–184, 2017, doi: 10.1016/j.ecolind.2017.01.042.
- Zannetti, P. (Ed.). (1990). *Air pollution modeling: theories, computational methods and available software*. Springer Science & Business Media.
- Zheng Z., Huang W., Li S., and Y. Zeng, "Forest fire spread simulating model using cellular automaton with extreme learning machine," *Ecol Modell*, vol. 348, pp. 33–43, 2017, doi: 10.1016/j.ecolmodel.2016.12.022.
- Zhou G. et al., "A BUFFER ANALYSIS BASED ON CO-LOCATION ALGORITHM," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-3, no. 3, pp. 2487–2490, May 2018, doi: 10.5194/ISPRS-ARCHIVES-XLII-3-2487-2018.