Towards Knowledge-Driven Automatic Service Composition for Wildfire Prediction *

Hela Taktak[1,2], Khouloud Boukadi[1], Chirine Ghedira Guégan[2], Michael Mrissa[3], and Faïez Gargouri[1]

[1] Sfax University, MIRACL Laboratory, FSEG Sfax, Tunisia
[2] Lyon University, Lyon 3 University, LIRIS UMR5205, France
[3] InnoRenew CoE, University of Primorska, Slovenia
{hela.taktak1, chirine.ghedira-guegan}@univ-lyon3.fr
khouloud.boukadi@fsegs.usf.tn
michael.mrissa@innorenew.eu
faiez.gargouri@isims.usf.tn

Abstract. Wildfire prediction from Earth Observation (EO) data has gained much attention in the past years, through the development of connected sensors and weather satellites. Nowadays, it is possible to extract knowledge from collected EO data and to learn from this knowledge without human intervention to trigger wildfire alerts. However, exploiting knowledge extracted from multiple EO data sources at run-time and predicting wildfire raise multiple challenges. One major challenge is to provide dynamic construction of service composition plans, according to the data obtained from sensors. In this paper, we present a knowledge-driven Machine Learning approach that relies on historical data related to wildfire observations to guide the collection of EO data and to automatically and dynamically compose services for triggering wildfire alerts.

Keywords: Machine Learning · Fire Prediction · Service Composition

1 Introduction

Wildfire, or wildland fire, is a regularly spotted critical phenomenon that can massively damage human lives, infrastructures, agriculture, and forest ecosystems. It has negative implications on air and water quality, and soil integrity. Recent estimations based on Earth Observation (EO) satellites of the global burned area is around 420 Mha [9]. Several advances have been made in fire

* This work was financially supported by the "PHC Utique" program of the French Ministry of Foreign Affairs and Ministry of higher education and research and the Tunisian Ministry of higher education and scientific research in the CMCU project number 17G1122. The authors acknowledge the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European regional Development Fund).

observations based on physics-based simulators and remote-sensing technologies, such as satellites (e.g. NASA TERRA), and fire detection sensors (e.g. Visible Infrared Imaging Radiometer Suite (VIIRS)), that continuously monitor vegetation distribution and changes. Our PREDICAT (PREDIct natural CATastrophes) project[4], collects EO data from several data sources, such as the National Oceanic and Atmospheric Administration (NOAA) which focuses on climate and oceans, and the Observatory of Sahara and Sahel (OSS), which rely on remote sensing technologies and Internet of Things (IoT) devices for climate parameters sensing. Exploiting EO data rises multiple challenges, ranging from data collection through service composition[5] to triggering wildfire alerts. The issue is that triggering wildfire alerts, is generally realized, once the collection of all the EO data is available. Existing wildfire detection and prediction systems mainly focus on accurately building feature observations and manually defining domain-rules. Indeed, several Web service composition approaches have been developed to implement data collection and trigger alerts, based on optimization, decision-making methods [10, 12], and semantics [2, 8]. While producing candidate solutions, most service composition techniques neglect knowledge from previously called services or from the application domain. Existing approaches start from a user request and try to find out a set of optimal services, based on functional and non-functional parameters. However, the service composition process for wildfire prediction should be data-driven, which means that the service calls alternatives are chosen at run-time depending on the data received from the previously called services.

Therefore, in this paper, we propose a bottom-up approach to guide the EO data collection from IoT devices based on a knowledge-driven process, and dynamically compose services accessing IoT data for wildfire prediction. We apply Machine Learning (ML) techniques [1], to guide the service composition process for EO collection and fire prediction by learning from the data itself and from historical data related to fire observations. Our solution exploits a service-based combination of ML and knowledge engineering methodologies, to automatically and dynamically compose services for wildfire prediction. Our contribution can be summarized as follows: we describe our knowledge-driven service composition approach through a system architecture for wildfire prediction, including (1) a dynamic and knowledge-driven construction of services composition scheme to organize the flow of services, based on a Prediction Module. (2) a classification of the EO historical data collections, by a Learning Module. (3) EO data collection, according to the most important feature related to fires to alert the scientists, by an Awareness Module. (4) the generation of alert risk patterns, by the Prediction Module. The remainder of this paper is structured as follows: Section 2 overviews related works dealing with Web service composition. Section 3 describes the global system architecture for prediction. Section 4 defines the pre-

[4] `https://sites.google.com/view/predicat/predicat`

[5] Service composition is the combination of a set of the smallest services forming a more complex service to meet users' complex requirements.

diction flow. Section 5 demonstrates the applicability of our approach. Finally, we conclude and present our future work in Section 6.

2 Related work

Web service composition for IoT environments has been often employed to access multiple IoT devices [18, 19]. Authors in [21] proposed a distributed social network approach for IoT management and service composition. An agent-based middleware was proposed in [22], to handle service composition of logistics services in IoT. Asghari et al. [20] proposed a systematic literature review on service composition approaches in IoT. They aimed to analyze and categorize IoT service composition methods into two main categories focusing on functional and non-functional properties. Other efforts were oriented towards semantic Web service composition [8], [2], [3], and QoS-aware web service composition [10–12]. Many approaches employed Evolutionary Computing (EC) to automate the generation of composition solutions and the optimization of the QoS web service composition by handling the large search spaces of services. Genetic Algorithm (GA) is used in [10], to propose a composition solution. Genetic Programming is employed in [13] and [11] to find near-optimal solutions using a fitness function. Although these solutions are a promising direction, efforts are still needed for enforcing composition constraints and also, for optimizing the quality of solutions. Another group of approaches for Qos-aware Web service composition employ the Particle Swarm Optimisation (PSO), which searches for near-optimal solutions by avoiding producing invalid solutions [12]. All these QoS-aware service composition approaches do not consider the semantic matchmaking quality of service compositions. Therefore, another group of researches focused on the semantic Web service composition using ontology-based semantics, such as OWL-S, WSML, and SAWSDL [14], to semantically represent the knowledge conveyed in these Web services [5]. To sum-up, although a large number of approaches for semantic web service composition and QoS-aware service composition exist, they all propose solutions based on the expression of a user query and not on domain knowledge. Furthermore, these approaches do not dynamically adapt the composition schema when new knowledge is learned, to guide the data collection process in a knowledge-capable manner.

3 System Architecture for Prediction

Figure 1 depicts our layered prediction system architecture [4] that includes five layers: Semantic layer, Knowledge layer, Application layer, Service Composition layer, and Service layer. The Semantic layer contains the domain and source ontologies. It aims to semantically describe the domain services and their related data sources. Hence, the domain ontology represents the environmental domain concepts related to services for EO data collection. Furthermore, the

source ontology represents the quality of the data sources accessed by their related services[6]. The Knowledge layer consists of collecting the EO data from diverse environmental data sources, provided by the OSS. The environmental data sources include data from several devices (i.e: sensors, connected objects, satellites, etc.). In our case, the knowledge-base contains the main features of interest of the environmental domain related to the fire occurrences.
The Application layer which constitutes the main focus of this paper, sums-up our vision of how to predict wildfire occurrences based on knowledge and learning methodologies. The Application layer uses the EO data from the knowledge-base. This layer encompasses the Prediction Module, which is composed of two sub-modules: the Learning Module and the Awareness Module. The Prediction Module produces alert risk patterns. The Learning Module builds the prediction model by applying a classification ML algorithm. It takes as inputs the set of features of interest that shapes the fire (i.e: temperature, wind speed, humidity, wind direction, etc.) and produces as output the class of fire danger (i.e: Moderate, Low, High, Very High, Extreme, Catastrophic, etc.). The supervised ML algorithm in the Learning Module generates the decision tree (DT), which is the fire prediction model. This model determines the danger classes of fire, in the leaf-nodes of the DT. The Awareness Module handles the search for the important features and the traversal branches of the generated DT. It consists of traversing all the necessary branches of the DT, from the important feature node till the leaf-node determining the danger class fire. The first step in the Awareness Module is to determine the most important feature, upon a classification task of all the features of interest. The most important feature is determined by computing the highest score among all the features of interest based on RF feature selection method. More details about this method are given in section 5.1. In the Service layer, the most important feature is mapped to an abstract service, which is a class interface representing its functionality and designed independently from particular implementations of services. The abstract service of the most important feature is then mapped to its service instance. This service is invoked, executed, and returns a value. The returned service value is then, compared to the node threshold of the most important feature. Upon this comparison, the set of the traversed branches, from the most important feature node to the leaf-node, is determined, as a second step in the Awareness Module. The set of the traversed branches determines whether to traverse the left sub-tree or the right sub-tree of the most important feature node. Thus, the traversed path is guided by the most important feature. If the most important feature is present in multiple nodes in the prediction model, then, we have multiple paths traversing each of these nodes. Hence, we have multiple services composition schemas in the Service Composition layer, each of which is built according to the following processes. Each node in the constructed path traversing the most important feature node are mapped to abstract services. These abstract services constitute one of the abstract composition scheme, in the Service Composition

[6] Qualities of the data sources and services are out of the scope of this paper.

layer. In order to instantiate these services, within the Service layer, the Features Matching Module, maps each abstract service representing a feature node, with its suitable service description from the service registry. Thus, all the abstract composition schema in the Service Composition layer are mapped to schema composition with concrete services. In fact, the service registry encompasses services descriptions expressed semantically as detailed in our previous work [6]. Thus, all the invoked services from the service registry, in the Service layer, are enhanced with EO linked-data. The orchestration of these services execution follows the ordered set of decision nodes traversed from the root until the leaf-node, and passing by the most important feature node. Otherwise, each value of a feature node is determined by the execution of the suitable environmental Web service accessing to its related environmental data source within the PREDICAT project.

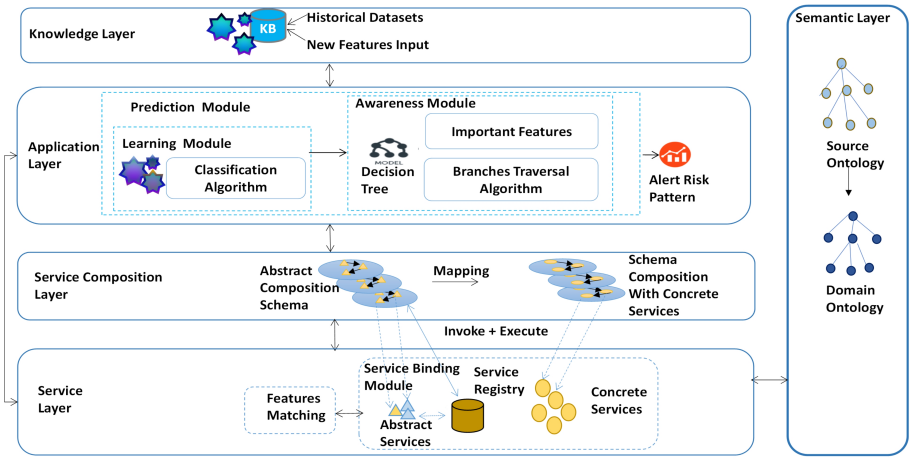

Fig. 1. The Prediction System Architecture.

4 The Wildfire Prediction Flow

Figure 2 presents the proposed prediction flow related to wildfire alerts along with its different phases. In fact, the novelty of our proposed flow is that it allows to trigger alerts to scientists, transparently and without any human intervention nor a user request. The first phase in this flow is to collect the EO data stored in the knowledge-base. 1-EO data-acquisition phase is realized by the IoT devices that belong among others to the OSS system. The second phase consists in 2-Data Preparation, which takes as input the data from the knowledge-base and splits it into two sets: the training data-set and the testing data-set. The Model Input Data consists of the set of features of interest, to which is applied the ML algorithm. The third phase is 3-Prediction Model Building one, which consists of performing the ML algorithm and produces the prediction model, which is the decision tree (DT). This phase relates to the Learning Module, detailed in

the previous section. A DT is a tree structure that consists of multiple internal-nodes and leaf-nodes [15]. Each internal node represents a single category; each branch of a node represents one possible value or a set of possible values of the category, and each leaf-node represents a class label. DT uses a tree structure to represent the rules between independent and dependent variables. Each node has a threshold compared with (i.e: <=, >).
The Random Forest (RF) ML classifier is an ensemble learning method developed by constructing multiple DTs [16]. In the training process, an RF applies a bagging technique to bootstrap instances and selects a random subset of features. A set of DTs is then constructed based on each set of bootstrap instances with a subset of features. Once the set of trees is constructed, a prediction regarding unseen samples can be generated by selecting the majority class of individual trees. Once the prediction model is generated, all the needed features of interest defined in the decision nodes are ready to be extracted. In the fourth phase 4-Prediction Model Deployment, each extracted feature of interest represents an abstract service that will be part of the abstract composition schema. The design of the abstract service composition schema is handled by the Awareness Module which is detailed in section 5.1. Furthermore, it is worthy to note that we can have multiple designed abstract composition schema as far as there are multiple nodes representing the most important feature of interest in the DT, determined by the Awareness Module, and guiding the EO data collection in the model prediction generated upon the Prediction Model Building Phase. See section 5.1 for additional details given in the presented algorithms. A Feature matching process is applied to each extracted feature, and which is mapped to an abstract service. The list of services along with their related descriptions are stored in a service registry. In the fifth 5-Service Composition phase, each abstract service is instantiated and executed. This execution is handled by the Composition Execution Engine. The orchestration of the execution of these services follows the order of traversing the decision nodes defined by the traversal algorithm, in the Awareness Module. Otherwise, each executed decision node, based on its value, determines which next decision node and its service instance to be executed. Furthermore, the execution of multiple instantiated composition schema is realized in parallel. Upon the execution of the schema composition, an alert is then, transmitted to the scientist.

5 Implementation and Evaluation

In this section, we present the feasibility of our fire model prediction in the guidance of the building of the service composition scheme. In particular, we demonstrate the effectiveness of our model. First, we provide an implementation example of our model with the Random Forest (RF) algorithm to perform the fire prediction. Then, we provide some algorithms showing details of the DT traversal from the most important feature of interest in the model prediction, which will guide the search for the rest of the features, in the tree. Thus, the generated paths across the DT are mapped to service composition schemas. Second, we focus on the evaluation of our built-model, by comparing it with other classifier algorithms, through the examination of the different performance indicators'.

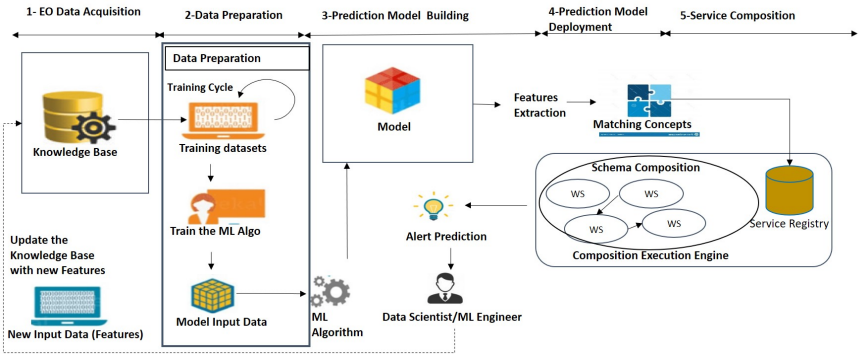

Fig. 2. The Prediction Flow.

5.1 Implementation

As aforementioned, we focus the implementation details on the two supervised classifiers: The Random Forest (RF) and the Decision Tree (DT). We used the built-in implementation of the RF and the DT algorithms, from the free software ML library "sklearn". Furthermore, we used Python programming language for the implementation and the "export_graphviz" module for the visualization of the generated DT in both classifiers. Each generated DT represents the fire model prediction. We, then, provide an evaluation based on performance indicators to choose the relevant classifier. In the following, we first introduce the used data-sets, then, present the implementation results.

Data-sets. We used data[7] related to weather recorded by the OSS as a set of inputs and stored in the knowledge-base. In particular, we used hourly data recorded between 1st November 2017 and 31st March 2019. This period of time includes a wide range of temperature ($°C$), relative humidity (%), wind speed (km/h), wind direction (°) and, drought factor values relevant to fire weather considerations. Furthermore, we used the McArthur Forest Fire Danger Index (FFDI) and its rating namely, the fire danger rating scale for forest (FFDR) as output used by our ML algorithm. The fire danger index is determined by the calculation of the FFDI according to the equation defined in [17]. FFDR is defined by the following classes: catastrophic (>100), extreme (75–99), severe (50–75), very high (25–49), high (12–24), and low-moderate (0–11). All these classes are considered as output to the ML algorithm and stored in the knowledge-base. We considered about 1500 tuples in our knowledge-base when performing the ML algorithm. We split the data-set into 70% for training and 30% for testing. Furthermore, in order to avoid overfitting and determine the optimal model performances, we used the "GridSearchCV" function from the "sklearn" library, to

[7] https://docs.google.com/spreadsheets/d/1v-46-KMHtErt3IGigFsusk7Fnp61DKvctMs9KMH_a-E/edit?usp=sharing

tune the model hyperparameters for both RF and DT classifiers. It consists in using a subset of the training collection as a validation dataset. We considered the following hyperparameters. For cv=5 in the DT classifier: max_depth=10, criterion='entropy' and, min_samples_split=2. For cv=3 in the RF classifier: criterion='gini', max_depth=10 and, n_estimators=90. Furthermore, we used the Amicus fire knowledge-base[8], which is a free suite of tools to simulate the calculation of the FFDI index.

Learning Module. As aforementioned, the main objective of this module is to learn from the EO data itself and the historical EO data collected by the IoT devices. This module generates the fire model prediction. For the construction of the latter, we chose two decision tree algorithms: the Random Forest (RF) [16] and the Decision Tree (DT) [15]. This choice is explained by the fact that the decision tree algorithms are effective in that they provide human-readable rules of classification. We performed tests on both classifiers to choose the relevant one. Section 5.2 provides evaluation details. After performing an ML classifier algorithm, an extraction of the produced fire model prediction decision tree is depicted in Figure 3. Each classifier produces a fire model prediction, each of which represents a decision tree (DT).

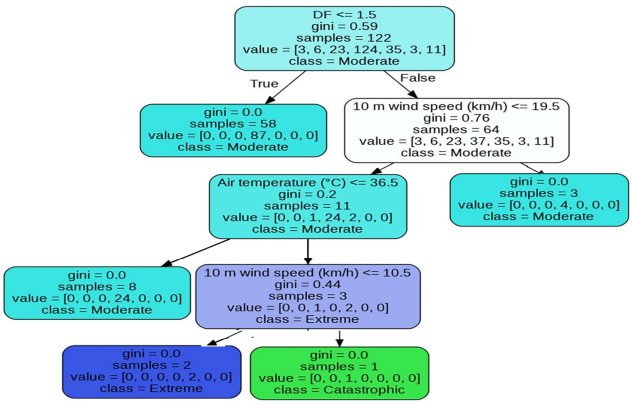

Fig. 3. Extraction from the decision tree.

Awareness Module. The Awareness Module encompasses two algorithms: the first one determines the most important feature which indicates its impact on the model compared to the rest of the features, and the second one determines the path to traverse in the DT, from the most important feature node till the leaf-node containing the fire danger class. Figure 4 depicts the relative important features, taking into consideration the set of the used features. We observed that the drought factor feature (DF) has the highest importance value. This value is determined by at first, applying the RF feature selection method [7], which produces a list of scored features within the prediction model. This list is

[8] `https://research.csiro.au/amicus/`

denoted "L" int the Algorithm 1. Second, by applying the maximum equation on these values to determine the most important feature in our model prediction. Algorithm 1[9] is a pseudo-code presenting details about determining the most important feature, which is the feature DF in our model prediction. Furthermore, once the most important feature is determined, the idea is to map the feature tag to its suitable service. This service is executed in order to have the DF value. According to the returned value by the DF service, the DF node will guide the tree traversal to search for the other features in the DT.

Algorithm 1 CMIF+ES.

Begin
Let L ← Search for features importance //List of the important features
DF ← $\max_{i \in L}\{importance(i)\}$ //The most important feature in our model prediction
Val ← Execute the Service Having DF as a tag//Exec of the important Feature service
End.

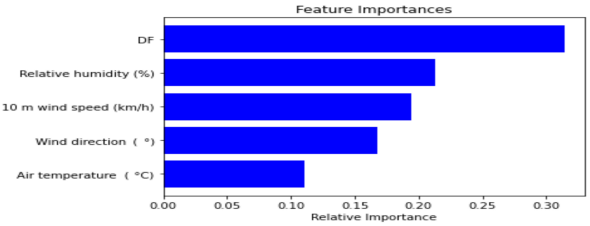

Fig. 4. The set of the important features in the fire model prediction.

Then comes the generation of the tree traversing the DF node which is composed on the one hand, of the path departing from the DF node to the root, and on the other hand, of the sub-tree of the DF node. To do so, Algorithm 2 reuses from Algorithm 1 the most important feature (e.g: DF) and its related value 'Val'. As a first step, in order to search for the DF node in the DT, the algorithm determines the nearest node to the root whose feature is DF. Furthermore, it generates the path from the DF node to the root. As a second step, according to the returned value 'Val', this latter is compared to the 'DF_Threshold' indicated in the chosen DF node in the DT. Thus, the algorithm decides which sub-tree to extract (i.e: the left sub-tree or the right sub-tree). Afterwards, the path and the sub-tree are merged to generate the tree that traverses the DF node. In fact, the DF node, according to its value, guides the ordered connections to the root and to the leaf-nodes. Thus, the DF node impacts on the dynamic generation of the service composition scheme. In case if we have multiple DF nodes in the DT, then we have multiple paths traversing each of these nodes. Therefore, we present Algorithm 3, which defines a pseudo-code managing the generation of multiple paths traversing the multiple DF nodes in the DT. This algorithm reuses the generated binary decision tree of the model prediction and the extracted sub-tree

[9] Computing the most important feature and executing its service.

Algorithm 2 Construction of the tree: path departing from the important feature to the root and, the sub-tree of the important feature.

Input: Tree //The generated decision tree of the model prediction
Output: Tree //Concatenation of the path departed from the important feature to the root and its extracted sub-binary tree
Begin
Node ← Nearest Node to the root whose feature is DF
Path ← Path from DF to the root
if (DF_Threshold ≤ Val) then SubTree ← Left_SubTree of DF
else SubTree ← Right_SubTree of DF
end if
Tree ← Path + SubTree //Fusion of the path and the sub-tree
return Tree
End.

determined according to the DF value in Algorithm 2. The idea, when having multiple DF nodes in the DT, is to only prune the extracted sub-tree from the binary tree and return the new tree. This way, the other non extracted sub-tree will be used, or another path traversing another DF node will be used. The sub-tree pruning is realized 'i' times until reaching a 'Stop_Condition'. The value of the 'Stop_Condition' (e.g: 10) is to be fixed at the beginning of the experimentation by the experimental user of the PREDICAT platform, to define the maximum number of service composition schemas supported to be run in parallel depending on the capacity of the PREDICAT platform. The generated paths constitute the possible constructed abstract service composition schemas. The execution of these composition schema is realized in parallel by instantiating the services at run-time. An alert is, then, triggered. In the next section, we provide the evaluation of the fire prediction model based on comparative performance measures computed from both previously detailed classifiers.

Algorithm 3 Generating multiple paths traversing multiple DF nodes.

Input: Binary Tree //The generated DT of the model prediction from Algorithm 2
Stop_Condition←10//10 supported generated paths to be executed in parallel in the platform
Output: Tree //The generated tree
Begin
Repeat
Tree ← Binary Tree \{$SubTree(i)$} //Generate a new tree by eliminating the subTree of the important feature extracted in Algorithm 2
Until (i > Stop_Condition) //Stop_Condition is to be fixed limiting the number of the generated //paths traversing the important feature DF.
return Tree
End.

5.2 Evaluation metrics

Several metrics for the evaluation of the performance of the classifier from the literature can be used. In our experiments, we considered four commonly used metrics, which are accuracy, precision, recall, and f1-score. These latter are indicators to measure the performance of our prediction models.

Table 1. System Performance Measures

ML Classifier	Accuracy %	Precision	Recall
Random Forest	86	0.862	0.862
Decision Tree	84.06	0.840	0.840

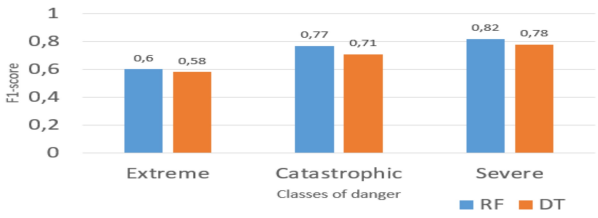

Fig. 5. Classes of fire danger chart.

To assess the evaluation of our fire prediction model, we used the most two popular ML classifiers which allow evaluating the RF classifier against the Decision Tree classifier. Moreover, we used performance measures computed for both ML classifier models. Furthermore, we considered the classes: Catastrophic, Extreme, and Severe as the most important classes of fire danger triggering alerts, and we measured the f1-score related to these classes, for each of the classifiers. According to results in Figure 5, we noticed that f1-score values in the RF classifier, all the danger classes advance those in the Decision Tree classifier. These results related to the most important danger classes show relevant values for prediction. Moreover, according to our experiments on our fire prediction model generated by the RF classifier, in Table 1, showed a high accuracy value, which is in advance to the one in the Decision Tree classifier.

6 Conclusion

In this paper, we proposed an approach that combines ML and knowledge-driven engineering to dynamically compose services from sensor data for wildfire predictions. The predicted alerts help scientists to anticipate and to manage fire in threatened areas. We evaluated our wildfire model prediction through several experiments, which showed relevant values with respect to the most important classes of fire danger. Future work includes exploring optimal services composition along with optimal services selection based on the quality of data and quality of services.

References

1. Mitchell, T.: Machine Learning, ISBN 0070428077, Publisher McGraw-Hill,(1997).
2. Boustil, A., Maamri, R., Sahnoun, Z.: A semantic selection approach for composite web services using OWL-DL and rules. Serv. Oriented Comput. Appl., (2014).
3. Rodriguez-M, P., Pedrinaci, C., Lama, M., Mucientes, M.: An integrated semantic web service discovery and composition framework. IEEE Trans. Serv.Comput. pp. 537–550, (2016).

4. Masmoudi, M., Taktak, H., Ben Abdallah Ben Lamine, S., Boukadi, K., Karray, M.H., Baazaoui Zghal, H., Archimede, B., Mrissa, M., Guegan Ghedira, C.: PREDICAT: A Semantic Service-Oriented Platform for Data Interoperability and Linking in Earth Observation and Disaster Prediction. Conf. on Service-Oriented Computing Applications (SOCA), Paris, pp. 194–201, (2018).
5. Bartalos, P., Bielikova, M.: Automatic dynamic web service composition: A survey and problem formalization. Computing and Informatics 30, 4, pp. 793–827, (2011).
6. Taktak, H., Boukadi, K., Mrissa, M., Ghedira, C., Gargouri, F.: A Model-Driven approach for semantic Data-as-a-Service generation. IEEE Int Conf on Enabling Technologies : Infrastructure for Collaborative Enterprises - WETICE, France, (2020).
7. Robin Genuer, R., Poggi, J.M., Tuleau-Malot, C.: Variable selection using random forests. Journal of Pattern Recognition Letters, vol. 31, pp. 2225–2236, (2010).
8. Bansal, S., Bansal, A., Gupta, G., Blake, M. B.: Generalized semantic Web service composition. Journal of Service Oriented Computing and Applications, (2016).
9. Giglio, L., Boschetti, L., Roy, D. P., Humber, M. L., Justice, C. O.: The Collection 6MODIS burned area mapping algorithm and product. Remote Sensing of Environment, 217 , pp. 72–85, (2018).
10. Gupta, I.K., Kumar, J., Rai, P.: Optimization to quality-of-service-driven web service composition using modified genetic algorithm. International Conference on Computer, Communication and Control, pp. 1–6, (2015).
11. Ma, H., Wang, A., Zhang, M.: A hybrid approach using genetic programming and greedy search for QoS-aware web service composition. In: ameurlain, A., Küng, J., Wagner, R., Decker, H., Lhotska, L., Link, S. LNCS, vol.8980, pp. 180–205, Springer, Heidelberg, (2015).
12. Sawczuk da Silva, A., Mei, Y., Ma, H., Zhang, M.: Particle swarm optimisation with sequence-like indirect representation for web service composition. In: Chicano, F., Hu, B., Garcia-Sanchez, P. (eds.) EvoCOP 2016. LNCS, vol. 9595, pp. 202–218.Springer, Cham, (2016).
13. Yu, Y., Ma, H., Zhang, M.: An adaptive genetic programming approach to QoSaware web services composition. In: 2013 IEEE Congress on Evolutionary Computation, pp. 1740–1747, (2013).
14. Petrie, C.J.: Web Service Composition. Springer, Heidelberg, (2016).
15. Murthy, Sreerama K.: Automatic construction of decision trees from data: A multidisciplinary survey. J.Data mining and knowledge discovery, pp. 345–389, (1998).
16. Breiman, L.: Random Forests. Journal of Machine Learning, vol. 45, (2001).
17. J. J. Sharples, R. H. D. McRae, R. O. Weber, and A. M. Gill.: A simple index for assessing fire danger rating. Environ. Model. Softw., pp. 764–774, (2009).
18. Urbieta, A., González-B, A. Mokhtar, S., Hossain, M., Capra, L.: Adaptive and context-aware service composition for IoT-based smart cities. Future Generation Com.Syst, (2017).
19. Deng, S., Xiang, Z., Yin, J., Taheri, J., Zomaya, A.Y: Composition-driven IoT service provisioning in distributed edges. IEEE Access, pp.54258–54269, (2018).
20. Asghari, P., Rahmani, A., Javadi, H.H.S.: Service composition approaches in IoT: A systematic review. J. of Network and Computer Applications, pp. 61–77, (2018).
21. Chen, G., Huang, J., Cheng, B., Chen, J.: A Social Network Based Approach for IoT Device Management and Service Composition. IEEE World Congr. Serv., pp. 1–8, (2015).
22. Yang, R., Li, B., Cheng, C.: Adaptable service composition for intelligent logistics: A middleware approach. Conf. Cloud Comput. Big Data, pp. 75–82, (2015).