	**Document:**	IMAS practical work
	Course:	ETSE-URV, 2021-22

Table of contents

General overview 2

Machine learning methods: Weka 3

Requirements 4

Tasks 4

1. Design activity 4

2. Implementation – preliminary revision 5

3. Final documentation and implementation 5

4. Teamwork 5

Useful links and recommended readings 6

General overview

The main goal of the practical work of this course is to create an **agent-based decision support system (A-DSS)** in teams of 4 or 5 students.

The system will process data about one particular domain to solve a classification problem in a collaborative way by means of multiple agents. The data will be picked up from a well-known repository as UCI Machine Learning[1]. In particular, we will consider the dataset AUDIT[2]. It consists of 767 instances, with 25 attributes defining each example. The goal is to help the auditors by building a classification model that can predict if a firm is fraudulent based on the present and historical risk factors. All instances were classified as normal (N) or altered (O).

We will use a set of M classifier agents, who will work using a decision tree algorithm. For simplicity all of them will use the same algorithm, named J48 (see details below).

Each agent will represent a different firm that collected a particular and private set of auditing data. To simulate this setting, each agent will use a subset of 300 instances to build a decision tree classifier (a 25% of the training subset will be used for validation). Moreover, each agent will consider only a subset of 6 attributes, simulating that each company was able to collect different features. Performance values of the model should be computed using the validation set, and stored in the agent database (i.e., precision, recall, F1 score, or others).

A subset of 50 instances will be separated at the beginning in a different test file, and they will be used to test the A-DSS. The user will launch a query to the A-DSS with a set of 15 instances with 20 of the 25 attributes for each one of these instances. An agent will be only able to run its classifier if it receives all the values for its 6 attributes.

The final A-DSS system will use the collaboration between these agents to provide a prediction of fraud to the user. As the prediction can differ from one agent to another, we will require to find some collective answer by using a coordination mechanism between the classifier agents. The performance of each classifier can be used in the coordination mechanism.

[1] https://archive.ics.uci.edu/ml/index.php
[2] https://archive.ics.uci.edu/ml/datasets/Audit+Data

Machine learning methods: Weka

Weka[3] is a collection of *open source machine learning algorithms* for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. It could be used as a standalone application, or by calling it through its API functions. You can learn/test the features of the machine learning algorithms with the standalone mode, but the A-DSS will use the Java-based API calls. Weka can be included in the Java project using Maven[4].

First, Weka uses a particular format of data called ARFF, which includes two main parts, the first one that defines the set of attributes of the dataset, and the second one, lists the set of data rows (see figure 2). You will find the AUDIT dataset ARFF file on the URV virtual campus site, Moodle.

Weka implements a set of classifier methods, such as decision trees (J48), nearest neighbour (IBk), neural networks (MLP), as well as it permits creating new ones. In this practical work, we will only work with one machine learning method, which will be the decision trees J48. Decision tree J48 is the implementation of algorithm C4.5, a well-known algorithm defined by Quinlan and based on entropy gain.

```
@attribute Sector_score numeric
@attribute                                                                LOCATION_ID
{LOHARU,NUH,SAFIDON,1,2,3,4,5,6,7,8,9,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,27,28,29,30,31,3
2,33,34,35,36,37,38,39,40,41,42,43,44}
@attribute PARA_A numeric
@attribute Score_A numeric
@attribute Risk_A numeric
@attribute PARA_B numeric
@attribute Score_B numeric
@attribute Risk_B numeric
@attribute TOTAL numeric
@attribute numbers numeric
@attribute Risk_C numeric
@attribute Money_Value numeric
@attribute Score_MV numeric
@attribute Risk_D numeric
@attribute District_Loss numeric
@attribute PROB numeric
@attribute RiSk_E numeric
@attribute History numeric
@attribute Risk_F numeric
@attribute Score numeric
@attribute Inherent_Risk numeric
@attribute CONTROL_RISK numeric
@attribute Detection_Risk numeric
@attribute Audit_Risk numeric
@attribute Risk {0,1}

@data
3.89,23,4.18,0.6,2.508,2.5,0.2,0.5,6.68,5,1,3.38,0.2,0.676,2,0.2,0.4,0,0,2.4,8.574,0.4,0.5,17.148,1
3.89,6,0,0.2,0,4.83,0.2,0.966,4.83,5,1,0.94,0.2,0.188,2,0.2,0.4,0,0,2,2.554,0.4,0.5,0.5108,0
3.89,6,0.51,0.2,0.102,0.23,0.2,0.046,0.74,5,1,0,0.2,0,2,0.2,0.4,0,0,2,1.548,0.4,0.5,0.3096,0
```

Figure 1: Sample of an ARFF file

[3] https://www.cs.waikato.ac.nz/~ml/weka/index.html

[4] https://mvnrepository.com/artifact/nz.ac.waikato.cms.weka/weka-stable/3.8.5

Requirements

The A-DSS is open to include different design decisions. However, the system must fulfil list of minimum requirements to accept the practical work:

- The system will work in two phases:
    1. The first phase consists of creating and training the classifier agents. An XML or Java properties configuration file must be used to define the information needed to create and train the classifiers (i.e., number of classifiers, the dataset file, number of attributes used for the training, etc).
    2. The second phase is creating a user agent, which will also be configured through an XML or Java properties file. The user agent is the agent in charge of making a request to classify the instances of the test set.
- A minimum of 10 classifier agents trained using 6 attributes are required.
- Classifier agents must register to the Directory Facilitator providing the information about the services they can provide.
- After being trained, the performance metrics of the classifier must be computed using the validation set. Each agent will keep record of its evaluation metrics.
- The A-DSS should provide a coordination mechanism for the classifiers in order to compose the final decision in the prediction stage.
- Other types of agents can be added if necessary.

- It is mandatory to document all changes on the practical work using a Git server. You can use GitHub (https://github.com/github), GitLab (https://about.gitlab.com/) or Bitbucket (https://bitbucket.org/). The teacher must have read-only access to the repository to check the evolution of the project. Use the following email address to add the teacher to the Git repository: jordi.pascual@urv.cat.

Tasks

1. Design activity

Description: In this activity, a first analysis of the practical exercise must be performed. An architecture for the multi-agent system must be designed and its characteristics defined. The design decisions include the types and properties of the agents in the MAS system, the functions of each agent, the overall architecture of the system. Write a detailed justification for those decisions.

Delivery content: PDF report with your proposed design for the MAS and PDF with about 7-8 slides for a presentation of up to 8-10 minutes.

Delivery deadline: October, 17th, 2021.

Grade weight: documentation and oral presentation 10%

2. Implementation – preliminary revision

Description: The main goal of this activity is to begin the implementation of the practical work. The first set of basic functionalities such as the initialization of the system, and creation of the agents is required.

Delivery content: You do not need to deliver any file. The lecturer will check the implementation in the laboratory class.

Delivery deadline: the revision will be done in the lab in November 10th or 17th , 2021.

Grade weight: code 5%

3. Final documentation and implementation

Description: In this activity, the complete implementation of the practical work must be delivered. All the required agents must be implemented, as well as the necessary communication and coordination mechanisms among the agents.

A README file must be included with brief instructions on how to execute the project. The XML or Java properties configuration files must also be included in the project.

Delivery content: PDF report with the details of the MAS system, PDF with 7-8 slides for a presentation of up to 8-10 minutes and the source code of the project in a ZIP file.

Delivery deadline: January, 9th ,2022.

Grade weight: 20% documentation and oral presentation + 20% implementation

4. Teamwork

Team work will be evaluated too as one of the advantages of multi-agent systems is to facilitate working in a distributed way. The team must write minutes of the meetings of the group using the forum task available in the Moodle campus. The minutes must indicate how the work is planned and distributed into the different team members, replanning actions, changes introduced during the project, and any other important issue related with the group work. This forum is only accessible by the group and the lecturers.

You can assign different roles in the team if you want: person in charge of the minutes (i.e. secretary), person in charge of reporting, leader, etc.

If someone abandons the course, please write it in the minutes, but also send an email to the lecturers as soon as possible.

Grade weight: 5%

Useful links and recommended readings

Information about Weka:

- In this link, you can find an introduction to all different concepts introduced in Weka: https://www.cs.auckland.ac.nz/courses/compsci367s1c/tutorials/IntroductionToWeka.pdf
- Documentation about how to use/include Weka in your Java code, https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/ .

The configuration files could be written using an XML-based notation or using java-like properties:

- In this link, you can find information about how to handle properties' files, https://www.mkyong.com/java/java-properties-file-examples/.
- There are some alternatives to handle XML files. In previous courses of IMAS, the JAXB was used (https://docs.oracle.com/javaee/7/api/javax/xml/bind/JAXBContext.html).

As you work is a team, it is useful to use well-known and widely used tools such as:

- As IDE, IntelliJ (https://www.jetbrains.com/idea/)
- JDK 1.8 (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).
- It is mandatory the use of Maven (https://maven.apache.org/) to create the workspace. Weka (https://mvnrepository.com/artifact/nz.ac.waikato.cms.weka/weka-stable/3.8.5) and JADE (https://jade.tilab.com/developers/maven/) are supported.
- It is highly recommended to use a logging library to trace all the entities of the A-DSS. Following, two alternatives are listed:
    - The Apache Log4j 2 (https://logging.apache.org/log4j/2.x/).
    - The Java Logger (https://examples.javacodegeeks.com/core-java/util/logging/java-util-logging-example/).