A.J. Fernández-García, L. Iribarne, A. Corral, J. Criado, J.Z. Wang. (2019): A Recommender System for Component-based Applications using Machine Learning Techniques. Knowledge-Based Systems Journal, Volume 164, 15 January 2019, Pages 68-84, Elsevier. ISSN: 0950-7051.

A recommender system for component-based applications using machine learning techniques

Antonio Jesús Fernández-García[a], Luis Iribarne[a], Antonio Corral[a], Javier Criado[a], J. Z. Wang[b]

[a]*Applied Computing Group, University of Almeria, Spain*
[b]*College of Information Sciences and Technology, The Pennsylvania State University, USA*

Abstract

Software companies are striving more and more to create software that adapts to their users' requirements. To this end, the development of component-based interfaces that users' can compound and customize according to their needs is increasing. However, the market success of these applications is highly dependent on the users' ability to locate the components useful for them. In our work, we propose an approach to address the problem of suggesting the most suitable component for each user at each moment, by creating a recommender system using intelligent data analysis methods. Once we have gathered the interaction data and built a dataset, we address the problem of transforming an original dataset from a real component-based application to an optimized dataset to apply machine learning algorithms through the application of *Feature Engineering* techniques and *Feature Selection* methods. Moreover, many aspects, such as contextual information, the use of the application across several devices with many forms of interaction or the passage of time (components are added or removed over time) are taken into consideration. Once the dataset is optimized, several machine learning algorithms are applied to create recommendation systems. A series of experiments that create recommendation models are conducted applying several machine learning algorithms to the optimized dataset (before and after applying *Feature Selection* methods) to determine which recommender model obtains a higher accuracy. Thus, trough the deployment of the recommendation system that has better results, the likelihood of success of a component-based application in the market is increased, by allowing users' to find the most suitable component for them, enhancing their user experience and the application engagement.

Keywords: Machine learning, Recommender Systems, Feature engineering, Feature Selection, Component-based interfaces, interaction information acquisition

1. Introduction

Launching a new software application requires an in-depth study of many variables in order to successfully achieve a good position in the market. Some of these variables are related to accomplishing a level of design and usability good enough to become established in an increasingly competitive market.

Hundreds of projects fail when going to market to the failure of users to embrace them, even when the software application could help them to improve their daily task performance, offering them a service that covers theirs needs. Whether or not companies spend a lot of time on design, consume a lot of resources enhancing usability or spend large amounts of money on marketing campaigns, if users do not quickly find the features they need, the software launch is likely to fail. This is more evident today since software applications greatly improved the interfaces that adapt themselves to the user's requirements.

The popularity of modern component-based web applications motivates the creation of interfaces that, frequently have a vast amount of components to be discovered. Examples of these interfaces are: Google Now [19], a kind of personal assistant where each component (called "card") offer information, answers questions or helps perform actions that users may need; Netflix [29], a streaming video-on-demand platform where each component offers information on a video's content [11, 25]; Flipboard [14], a news and social media aggregation system presented in a magazine format; or Fitbit Dashboard [13], an activity tracker brand that provides a dashboard where users can monitor their activity, exercise, food, weight and sleep patterns.

Due to the vast number of components that applications may have, the market success of a component-based web application depends, not only, on the ease with which the user may find components useful to them but also, on the ability of the system to identify the right components at the right time and properly organize them so they can be at the users' disposal. If users do not find useful components, they will probably discard the use of the web application, partly because of their lack of interest in the components they know.

The success of a web application can be partially achieved by predicting the components that are most suitable for each user at each moment. This can be solved by using simple methods such as identify the most frequently used components. Also,

Email addresses: ajfernandez@ual.es (Antonio Jesús Fernández-García), luis.iribarne@ual.es (Luis Iribarne), acorral@ual.es (Antonio Corral), javi.criado@ual.es (Javier Criado), jwang@ist.psu.edu (J. Z. Wang)

https://doi.org/10.1016/j.knosys.2018.10.019

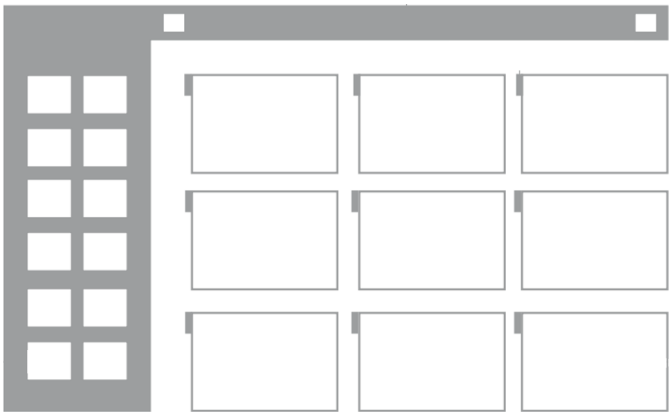

Figure 1: Component-based applications morphology

more elaborate techniques may be applied, for instance, through computational intelligence methods. This is an area yet to be explored since as far as we know, there are no related works that explore the component-based application usage by end-users that analyzes behavior to enhance user experience and maximize the software's probabilities of success in the market.

However, this is not an easy task for several reasons. With the passage of time, new components are added to web applications and users should be able to discover these in event that a newly added component may be useful for their interests. The problem is more compounded when applications have to make suggestions to new users without prior knowledge of their activity. In these scenarios, suggestions should be made based on others, similar user activities, that could be difficult to identify without bias.

The problem is even more challenging today because users interact with web applications across several devices (smartphones, tablets, etc.) using several forms of interactions (touch, gesture, voice, etc.). The devices that users handle and the way they interact with them clearly affect users' behavior. Often it is not enough to identify which component may be suitable for each situation but also whether the usage of that component is appropriate to the form of interaction and the device that a user is handling at that specific time.

In addition, more aspects may be considered to be in predicting the suitability of components to users: a) The nature of the user interacting may be taken into consideration (users categorization); b) Information may be extracted from the surrounding environment (context awareness); and c) other relevant external information related to the web application may be identified that may enrich the prediction data.

This paper addresses the problem of creating a useful recommendation system [31] that would be able to forecast the use of components in cross-device component-based applications with multiple forms of interactions. We intend to create a recommender system that can help users to discover the components most suitable for them, thereby improving their user experience of the applications. The success of the recommendation system will likely increase the possibilities of a web application project being successfully introduced into the market.

To validate the effectiveness of the recommendation system and the data analysis methods applied to suggest an example set of components to users, we have performed an empirical study on a real component-based web application with tracked data of its previous users. The name of the application is ENIA (Environmental Information Agent) [1], a component-based user interface for environmental management used by the Andalusian Environmental Information Network (REDIAM, Spain) [3].

In this case study, a practical methodology is applied that in early stages, obtains a raw dataset from the ENIA application. To this dataset, *Feature Engineering* techniques such as deleting features, filtering instances, transforming features datatypes, and merging or splitting features, among others are applied. This is fundamental for the proper application of machine learning, because the features representation has a great impact on improving prediction models. After that, the *Mutual Information Score* and *Chi-Squared Statistics* methods are applied to obtain a subset of the dataset previously transformed by the *Feature Engineering* techniques. These *Feature Selection* methods automatically select the most relevant features in the dataset and simultaneously, together with the *Feature Engineering* techniques, reduce a dimensionality problem that appears in our case study dataset.

The rest of the article is organized as follows. Section 2 reviews some related projects that involve creating forecasting systems or model-based recommendations models. Section 3 describes the problem of suggesting components to users of components-based user interfaces in depth and details the methodology followed summarizing how the solution to the problem is addressed. Section 4 details everything from gathering raw data to the creation of optimal datasets. The main topics discussed in this section are: a) gathering and describing the raw dataset obtained from ENIA web application; b) applying feature engineering techniques to have transform features with better representation; and c) applying feature selections approaches to identify the most significant features. Section 5 describes the algorithm used to build the recommendation models and analyzes the experimental results through relevant metrics. Finally, some conclusions and further considerations are summarized in Section 6.

2. Related Work

The evolution of data mining and big data involves an increasing interest in forecasting the demand for a product, the needs of a user or the possibility of a series of events happening in general. Accurately forecasting, precisely identifying trends and the discovery of behavior patterns clearly optimizes resource usage or consumption as well as generating new knowledge in science and research facilities; enabling faster and better decisions in politics, retail, weather, sport, science, research, real estate, sports or health-care among many others fields.

For these reasons, the use of recommendation models [31] to forecast the use of resources, anticipate user needs, or to make recommendations based on their behavior (and that of other users, including a situation context) are becoming more frequent. A compilation of some forecasting and recommender

https://doi.org/10.1016/j.knosys.2018.10.019

models using supervised machine learning algorithms found in the literature are featured below.

In some works, such as [41], the authors presented a framework to improve user recommendation in social networks. They capture user preferences involving both interest and social factors to create a model using *Matrix Factorization*[5] techniques, based on the principle that users who have shown similar interests in the past will likely have similar interests in the future.

Given the difficulty of modeling a real-world recommendation system, sometimes it is useful to create hybrid models that combine a model and knowledge of the problem addressed. In [42] the authors created a hybrid model-based job recommendation system using *Statistical Relational Learning* [17]. They also took into account tuning the algorithm to satisfy the requirement to give more weight to the more desirable classes (or difficult ones).

There are also approaches that work with similar algorithms used in this article such as *Decision Trees* [32], *Logistic Regression* [27] or *Artificial Neural Networks* [34]. In [7] they use data analysis approaches for predicting the sales of newly published books and in [24] they use contextual information as a way of recognizing human activities and, subsequently make music streaming recommendation accordingly. There are more works that focus on multiple purposes such as [44] in which the authors address the problem of detecting cell mitosis using deep neural networks, or [8] that deal with the detection of malicious webmail attachments or [15] that focus on predicting intervals for solar energy forecasting with neural networks.

In our study, we also make use of context-aware information as we consider it to be valuable data. Works that deal with context-aware information to create recommendation models can also be found in the literature. In [39] the authors propose that contextual features may be considered to be organized in an understandable and intuitive hierarchy and they exploit this hierarchical structure to improve the quality of their recommendation models. The exploitation of context information to improve models is frequently used [38]. For instance, in [35] the authors incorporate contextual situations (e.g., geographical location, special date or activities of interest) to improve the precision of a retailing recommendation system using a collaborative filtering approach. In [20] the authors make use of smartphone's context-aware capabilities to create an adaptive and personalized mobile learning system, which aims to support the semi-automatic adaptation of learning activities and helps students to successfully complete the learning activities of an educational scenario.

Often, before the creation of a recommendation model, *Feature Selection* techniques are used to improve the dataset quality (among other purposes). We also apply feature selection techniques to improve our model's accuracy, deal with high-dimensional spaces and to solve problems such as the sparsity of data. The literature shows how many works use feature selection techniques before building models. There are examples worthy of comment in different areas such as spam detection [43], fraud detection [21] or diagnosis of mechanical faults [40], among others.

Thus, given the related works mentioned, it makes sense to use machine learning algorithms to create a recommendation model to suggest the most suitable component for users on component-based interfaces, that enhances user experience.

3. Methodology and Problem Description

The steps followed to create the recommendation models on component-based web applications are graphically described in Figure 2. These steps do not represent a formal methodology, they are a guideline we have followed in carrying out the work described in this paper. The application of these steps is completely optional, subject to the nature of the problem presented, and the data scientists knowledge.

The first step, understanding the problem, is already briefly discussed in the introduction. It can be summarized as follows: we need to create a recommendation model that assists users in component-based web applications to discover useful components, optimizing the user engagement with the application. The aim of the recommendation model is to maximize the potential market success of a component-based web application.

In order to really understand the problem of the ENIA component-based application case study, there are important facts to be considered (which can be extrapolated to other component-based applications):

(a) The set of components offered by the web application are in continuous growth, new components can be added (or removed) over time.

(b) New users are registered in the web application with a degree of frequency.

(c) Users are categorized according to how they are expected to use the application, e.g. Tourist or Farmer.

(d) Users interact with the interface from different devices (Smartphones, Tablets, Wearables, Laptops, Desktops PC, etc.) with different forms of interaction (mouse and keyboard, touch screen, voice, etc.).

(e) The context that surrounds the users' interactions with the web application is relevant to the way they behave, e.g. Weather.

The second step is gathering the data from the component-based web application [12]. Fortunately, an increasing number of applications have data repositories and a growing number of organizations and companies now store data, sometimes leaving it freely available to everyone (OpenData). If this were not the case, this step would be tedious since it is necessary to access the information system and manually collect the data. In this case, the ENIA application has a service that stores all of the information related to each interaction performed over its user interface in a MySQL database [30]. A CSV file containing all the interaction can be requested and will be generated using the stored data in the ENIA database.

The third step consists of processing the raw data obtained in the previous step. Finding out the relevant attributes that

https://doi.org/10.1016/j.knosys.2018.10.019

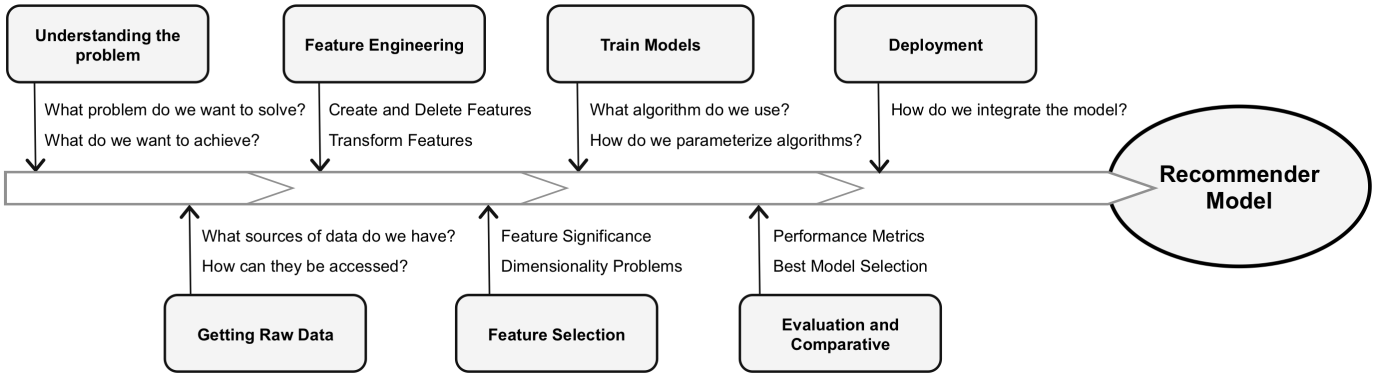

Figure 2: Methodology applied to create the recommendation model

describe an interaction is vital to ensure the success of the the recommendation system. These attributes are called *features* and the set of features is called a *dataset*. This step includes cleaning features and transformation, as well as an important and laborious process known as *Feature Engineering* that we will address further in this paper.

The fourth step consists of applying *Feature Selection* methods that, as we shall see later in this paper, handle many issues related to data such as the sparsity of dat, in addition to reducing the computing resources needed by the machine learning algorithms in order to create the models.

The fifth step consists of applying machine learning algorithms to model the recommendation system we are looking for. There are a wide range of algorithms that can be applied. In this work, we make use of *decision trees*, *logistic regression* and *artificial neural networks*.

Finally, the sixth step consists of validating the recommendation model built by analyzing the accuracy of the algorithms used. For this, a comparison of the experimental results of the applied algorithms is made to determine whether the recommendation models created are suitable to be deployed in the component-based web application and which recommendation model should be deployed.

The following sections describe the steps commented above. Section 4 covers the step required to create the datasets that include gathering and processing, apply *Feature Engineering* techniques and *Feature Selection* methods. Section 5 describes the algorithms applied to create the recommendation model and thoroughly explains the results and validity of the experiments.

4. Creating Optimal Datasets

In this section, we focus on creating the optimal datasets for applying machine learning algorithms. In this way, we shall maximize the possibilities of suggesting the proper component to users in every moment. Thus, the recommendation system that will be built in further steps will be truly useful to end users.

First of all, we obtain raw data from the ENIA component-based user interface. Following this, we use of *Feature Engineering* techniques to help machine learning algorithms to create accurate and simple prediction models, providing better results [23]. Finally, we apply *Feature Selection* methods to create a subset with the more meaningful features of the processed dataset which will imply shorter training times to build the models by removing redundant or irrelevant features.

4.1. Gathering Raw Data

In order to create a recommendation system on component-based applications and suggest useful components to users, it is necessary to create datasets with rich enough attributes to model the problem in the first place. As a starting point, we have a raw dataset provided by our case study component-based application, ENIA. The data contained in the dataset has been directly extracted from the component-based web application through a data acquisition system that is incorporated in the proprietary software application [9, 10] powered by `COSCore`, a modular and scalable service infrastructure approach for the management of component-based architectures [36, 37].

When the dataset was obtained, it consisted of 27702 interactions performed (instances) with a set of 18 features. The set of features describing each interaction can be seen in Table 1.

4.2. Feature Engineering

Table 1 describes the data obtained from the component-based web application. It contains data too raw for direct application of data analysis techniques. In agreement with the REDIAM agency and their experts in the field, some transformations on the dataset and its features have been made using *Feature Engineering* techniques.

Feature Engineering is a process that transforms raw data to create features that have better representation and therefore better to create predictive models [33]. The quality of the prediction models created with machine learning algorithms depends on the *Feature Engineering* approach that has been followed, where usually better features mean better prediction models.

Although *Feature Engineering* is usually treated lightly, it plays an important role in the success of a machine learning experiment. These techniques optimize the performance of the dataset, improve the data analysis algorithms's results, and create simple and reliable prediction models that perform well and accurately [23].

https://doi.org/10.1016/j.knosys.2018.10.019

Feature	Description
idInteraction	Unique Interaction Identification
dateTime	Exact moment in which the interaction took place
operationPerformed	Kind of interaction performed in the application
latitude	Latitude in which the interaction took place
longitude	Longitude in which the interaction took place
idUserClient	Unique User Identification
Name	Name of the user that performs the interaction
Surname	Surname of the user that performs the interaction
userType	Type of user that performs the interaction (tourist, farmer, etc.)
userSubType	Subtype of user that performs the interaction (tourist, farmer, etc.)
birthDate	Birth date of the user that performs the interaction
countryOrigin	Country of origin of the user that performs the interaction
areaOrigin	Area/State of the Country of the user that performs the interaction
email	Email of the user that performs the interaction
deviceType	Type of device on which the interaction was performed (laptop, desktop, smartphone, etc.)
interactionType	Form of interaction on which the interaction was performed (mouse&keyboard, touch, voice, etc.)
idSession	Unique Session Identifier
actionComponent	Component over which the interaction has been performed

Table 1: Set of features describing an interaction directly obtained from the ENIA web application

The following *Feature Engineering* transformations have been applied to the original dataset:

(a) **No transformation**. There are features that have good significance in the form they appear in the original dataset. For this kind of features, no type of transformation is necessary. For this reason, the `actionComponent`, `deviceType`, `interactionForm`, `countryOrigin`, `areaOrigin` and `userType` features have been left unaltered, the value they add to the dataset is significant enough as is.

(b) **Deleting Features**. There are features that do not correlate well with the objective field (label). Some are easy to find because they have no knowledge associated with the field. Others are more difficult to find because they are dependent on other features and to identify them domain-specific knowledge is required. It is important to identify these features and delete them. The `idSession`, `idUserClient` and `idInteraction` features have been discarded because they are not relevant to the accuracy of the model, they are simply auto-generated identifiers created by the ENIA database. The `idUserSubType` feature has also been discarded because it is an optional field that several users miss. According to the REDIAN experts, ENIA do not provide a good categorization of users subtypes. The `areaOrigin` and `areaInteraction` features have also been deleted. On this occasion, because they have many possible cases, i.e., a large amount of values can be assigned to these features (17 and 12 respectively) and we already have the `countryOrigin` and `countryInteraction` features that are similar and have more homogeneous values. In the future, with a higher number of instances, they could be kept.

(c) **Filtering Instances**. Often, the whole set of dataset instances are not required because the problem is circumscribed to a fraction of it. On these occasions, it is necessary to obtain a subset of the original dataset according to a rule that can isolate the required instances that are useful to create the model. In our case study dataset, the `operationPerformed` feature has 13 possible values: {`Add`, `Group`, `AddGroup`, `Ungroup`, `Delete`, `UngroupGroup`, `UngroupDelete`, `Resize-Bigger`, `ResizeSmaller`, `ResizeShape`, `Move`, `Maximize`, `Minimize`}. These are the possible operations that users can perform in the ENIA component-based application user interface. Since our purpose is to recommend components to users, we are interested in filtering the `operationPerformed` feature to keep the instances where the `operationPerformed` values are {`Add`, `AddGroup`}. These values of the `operationPerformed` feature bring together the operations where users begin using a component.

(d) **Processing Features**. There are features that, according to domain-specific experts, can be useful but algorithms may find them difficult to manage. These features should be processed to adapt them to more suitable datatypes, or representations that increase their significance and may be easily processed using data analysis algorithms. In our case study, the `birthDate` feature is too complex to obtain significance from it, thus it has been transformed to a new feature: the age the user was when he/she performed the interaction (`age`), which performs better.

(e) **Creating new features by splitting existing ones**. Sometimes there are features rich enough to be broken down into more than one feature because they contain a great deal of information in a specific domain. The ENIA original dataset comes with the `dateTime` feature, which contains the exact time when an interaction was performed. It has been divided into two features: `season` and `partOfTheDay`, which allows us to explore two different aspects that surround the interaction and may influence the user's behavior.

(f) **Creating new features by merging existing ones**. In contrast to the previous case, sometimes there are features that by themselves do not contribute to creating better prediction models. However, by merging them with

https://doi.org/10.1016/j.knosys.2018.10.019

others features, it is possible to create meaningful features that improve the overall performance. The `latitude` and `longitude` features do not add relevant information to the recommendation systems by themselves, but they can be transformed into two interesting new features: `countryInteraction` and `areaInteraction`.

(g) **Adding relevant features**. Often, there is meaningful data that is not contained in the dataset but can be obtained from existing features. Because we are talking about a component-based web application developed for an Environmental Agency, it is considered relevant, according to the experts, to know the weather conditions at the time that the interactions have been carried out. A piece of code has been implemented that connects to the `OpenWeatherMap` [2] service and using the `latitude`, `longitude` and `dateTime` data features as inputs can retrieve the weather conditions. So, two new features have been inserted in the dataset: `weatherDescription` and `Temperature`.

(h) **Discretizing Features**. Many machine learning algorithms produce better prediction models using discretized values. By discretizing, the impact that small variations have on the data is reduced, which justifies the loss of information [16]. We performed the following discretizations, assisted by the experts:

— The numeric values of the `age` feature are calculated (as described before) and categorized into the following values:

{*<18, 18-25, 25-35, 35-50, 50-65, >65*}.

— The values of the `season` feature have been categorized according to the values:

{*Fall, Summer, Spring, Winter*}

— The values of the `partOfTheDay` feature have been categorized according to the values:

{*Morning, Afternoon, Evening, Night*}

Table 2 synthesizes the transformation that has been taken place in each feature.

4.3. Reducing the Number of Classes

Classification algorithms in machine learning can be categorized as binary or multiclass, according to the number of classes they can predict. In this study, we are trying to forecast the most feasible components to suggest to users in specific situations. Thus, the algorithm must decide between the available components which one is the most suitable. This is a multiclass problem. It would be different if the model were to foresee whether a component is suitable or not, then it would be a much easier binary problem.

After the *Feature Engineering* process, our dataset consists of 27702 instances with a set of 11 features (including the objective field). The possible values that can be assigned to each feature and its data type is described in Table 3 and summarized as follow:

```
Dataset = {
  weatherDescription (4),
  temperature (7),
  countryInteraction (4),
  countryOrigin (4),
  userType (4),
  age (6),
  season (4),
  partOfTheDay (4),
  deviceType (3),
  interactionType (3),
  actionComponent (33)
}
```

The number of cases in the `actionComponent` feature, that is the objective field, is too high. With 33 different components that can be suggested and 27702 instances to train the model, it is highly probable that the forecasting would not be accurate. According to the number of cases of each feature, the total number of possible cases is more than 50 million ($4*7*4*4*4*6*4*4*3*3*33 = 51.093.504 \approx 5*10^9$). If we had not deleted the `areaOrigin` and `areaInteraction` features the total number of possible cases would have been more than 10 american billion ($4*7*4*12*4*17*4*6*4*4*3*3*33 = 10.423.074.816 \approx 10^9$). With this action the problem was drastically reduced, but not enough.

For this reason, the number of components to be suggested has been limited. A reduction of the 33 possible classes of the objective field has been made, reducing the number of classes to 8. We have selected the components that are the most frequently used. The recommendation system is then set up to suggest only these 8 components to users. It is remarkable that the appearances of these 8 cases selected account for 21882 instances ($\approx$ 78%). The possible cases of the `actionComponents` feature that can be predicted are `BioReserves`, `Clock2`, `CuttleRoads`, `GeoParks`, `Heritage`, `Twitter`, `Weather` and `Wetlands`. The number of instances of each class and its frequency can be seen in Table 4. Thus, the recommendation system is a hybrid between computational intelligence methods and a simpler method that selects the 8 components most frequently used by users of ENIA.

Even after the objective field classes reduction, the number of possible cases is more than 12 million ($4*7*4*4*4*4*6*4*4*3*3*8 = 12.386.304 \approx 12*10^6$). It remains a large number but, given the high number of occurrences of these classes, and the high appearances frequency of the selected classes, it is enough. The domain-specific experts, i.e., employees of the environmental agency that make use of the application to perform their tasks, agree with this simplification. Besides, following their advice, the recommendation model will not suggest components to users in every situation, it will only suggest on the occasions that the likelihood of using a component is sufficiently high to make a good recommendation. For example, if the recommender system suggests a component with a likelihood of 37%, the recommendation is not shown to users because we have set a threshold of 50% of likelihood.

https://doi.org/10.1016/j.knosys.2018.10.019

Before FE	After FE	Consideration
idInteraction	~~idInteraction~~	Deleted
idUserClient	~~idUser~~	Deleted
idSession	~~idSession~~	Deleted
operationPerformed	~~operationPerformed~~	Deleted. Instances filtered by add or addGroup
name	~~name~~	Deleted
surname	~~surname~~	Deleted
email	~~email~~	Deleted
userType	userType	Kept
userSubType	~~userSubType~~	Deleted
birthDate	Age	Processed
deviceType	deviceType	Kept
interactionType	interactionType	Kept
dateTime	season	New Feature (Splitting)
	partOfTheDay	New Feature (Splitting)
	temperature	New Feature (from external sources)
	weatherDescription	New Feature (from external sources)
latitude	Country Interaction	New Feature (Merging)
longitude		
countryOrigin	countryOrigin	Kept
areaOrigin	~~areaOrigin~~	Deleted
actionComponent	actionComponent	Objective field

Table 2: Features transformation carried out in the *Feature Engineering* process

Feature	Cases (Possible Values)	Type
`weatherDescription`	Clouds, Clear, Fog, Mist	Categorical Feature
`temperature`	<-10, -10-0, 0-10, 10-20, 20-30, 30-40, >40	Categorical Feature
`countryInteraction`	Not limited possibilities. When creating the dataset: 4 cases	Categorical Feature
`countryOrigin`	Not limited possibilities. When creating the dataset: 4 cases	Categorical Feature
`userType`	Tourist, Farmer, Technical, Politician	Categorical Feature
`age`	<18, 18-25, 26-35, 36-50, 51-65, >65	Categorical Feature
`season`	Spring, Summer, Fall, Winter	Categorical Feature
`partOfTheDay`	Morning, Afternoon, Evening, Night	Categorical Feature
`deviceType`	Computer, SmartPhone, Tablet	Categorical Feature
`interactionType`	MouseKeyboard, Touch, Voice	Categorical Feature
`actionComponent`	Not limited possibilities. When creating the dataset: 33 cases. Limited to 8.	Categorical Label

Table 3: Set of features describing an Interaction after processing the original raw dataset

Class	Instances	Frequency
`GeoParks`	5572	0.25463
`Heritage`	4760	0.21753
`CuttleRoads`	3038	0.13883
`Twitter`	2618	0.11964
`Weather`	2282	0.10428
`Wetlands`	1400	0.06397
`BioReserves`	1302	0.05950
`Clock2`	910	0.04158
Total	21882	1

Table 4: Most frequently used components

4.4. Feature Selection

The set of attributes of a dataset should be as significant as possible to describe the nature of the reality they represent. It may seem that a large number of features can better describe a problem and (with them) better predictive models can be built, but this is not entirely true. *Feature Selection* methods can help to reduce overfitting [4] and avoid high-dimensional spaces and the sparsity of data [22] that datasets with a large set of attributes may have. Also, a feature subset of a dataset implies shorter training times building the models by removing redundant or irrelevant features.

In this work, we make use of feature selection methods that support all data type features, specifically the *Mutual Information Score* and *Chi-Squared Statistic* methods available in the Microsoft Azure Machine Learning Studio (AzureML) [28].

4.4.1. Mutual Information Score

The *Mutual Information Score* method is defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y)\ log\left(\frac{p(x,y)}{p(x)\ p(y)}\right); \quad (1)$$

where $p(x,y)$ is the joint probability function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively [18].

Table 5 shows the results of applying the *Mutual Information Score* selection method to the ENIA dataset and Figure 3 illustrates the relevance of each feature.

Feature	MR	MO	CR	CO
`userType`	0.078864	1	383.256110	1
`age`	0.036541	2	170.996097	2
`countryInteraction`	0.022832	3	106.796770	4
`weatherDescription`	0.020199	4	91.611703	5
`partOfTheDay`	0.018090	5	73.870284	6
`countryOrigin`	0.017928	6	121.178845	3
`deviceType`	0.003489	7	14.995899	7
`interactionType`	0.002720	8	11.730771	8
`season`	0.001766	9	7.233112	9
`temperature`	0	10	0	10

Table 5: *Mutual Information Score* and *Chi-Squared Statistics* results (MR: Mutual Information Result; MO: Mutual Information Order; CR: Chi-Squared Result; and CO: Chi-Squared Order)

https://doi.org/10.1016/j.knosys.2018.10.019

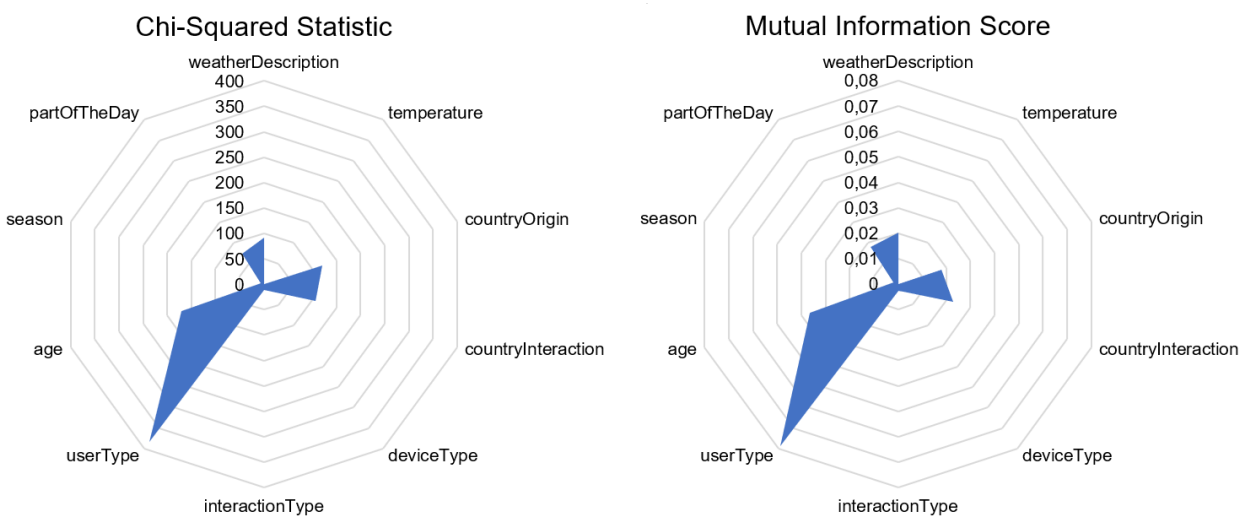

Figure 3: *Mutual Information Score* and *Chi-Squared Statistic* Features Relevance

The relevance of the `userType` feature is far higher than any other feature, followed by the `age`, `countryInteraction` and `weatherDescription` features, that also have considerable weight. Conversely, the `deviceType` and the `interactionType` features have little relevance, probably because the vast majority of users interactions has been performed in desktop computers by means of mouse and keyboard. The `temperature` feature has no relevance, probably because that feature can be highly correlated with the `season` and `partOfTheDay` features.

4.4.2. Chi-Squared Statistic

The *Chi-Squared Statistic* method is defined as:

$$X^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\left(O_{(i,j)} - E_{(i,j)}\right)^2}{E_{(i,j)}} \; ; \tag{2}$$

where $O_{(i,j)}$ is the observed value of two nominal variables and $E_{(i,j)}$ is the expected value of two nominal values. The expected value can be calculated with the following formula:

$$E_{(i,j)} = \frac{\sum_{i=1}^{c} O_{(i,j)} \sum_{k=1}^{c} O_{(k,j)}}{N} \; ; \tag{3}$$

where $\sum_{i=1}^{c} O_{(i,j)}$ is the sum of the i_{th} column and $\sum_{k=1}^{c} O_{(i,j)}$ is the sum of the k_{th} column [26].

Table 5 shows the results of applying the *Chi-Squared Statistics* selection method to the ENIA dataset and Figure 3 illustrates the relevance of each feature.

When we observe Figure 3 and Table 3 we can appreciate that there is not very much difference. Both *Mutual Information Score* and *Chi-Squared Statistic* methods have thrown similar results. The main difference is that the weight of the `countryOrigin` feature in the *Chi-Squared Statistic* method is higher than the `countryInteraction` and `weatherDescription` features in comparison with the *Mutual Information Score* method.

4.4.3. Features Selected

The features selected to create a subset of the original dataset are:

```
Dataset = {
  userType,
  age,
  countryOrigin,
  countryInteraction,
  weatherDescription,
  partOfTheDay,
  deviceType,
  interactionType,
  actionComponent (label)
}
```

Taking into account the finding of the *Feature Selection* methods, the `deviceType` and `interactionType` features would have been discarded, but we have kept them since we expect that soon enough, ENIA web application user access from smartphones or tablets will increase and we want this recommendation system to automatically evolve over time. When creating the models, each is built using both datasets, the original and the subset of the original with the most significant features. The results are evaluated and their performance compared.

5. Recommendation Model and Experiments

This section analyses the machine learning algorithms used to create the recommendation models and presents the results of the experiments conducted. We evaluate the recommendation model's accuracy and performance based on relevant metrics obtained from the empirical study, to determine which recommendation model is more suitable to deploy. By selecting the more accurate model the recommendation system will provide ENIA users with the most suitable components, thereby improving their user experience.

https://doi.org/10.1016/j.knosys.2018.10.019

5.1. Recommendation Model

There are plenty of algorithms available in the literature that can be applied in order to create recommendation models. Likewise, by adjusting the parameters of these algorithms, there are many more configurations possible that directly affect their performance and accuracy.

In this case, it is worth noting that we are facing a multiclass classification problem, where each instance can be classified into more than two classes, specifically, in as many classes as components are available to suggest to users. Although the possibility exists of turning binary classification algorithms into multiclass classifiers, we've decided to make use of algorithms that naturally allow decisions between more than two classes.

We make use of four multiclass classification algorithms available in Microsoft Azure Machine Learning Studio [28]. We make use of the AzureML implementation because of the ease of setting up web services over trained models to deploy them that this platforms offers. We also value positively the fact that readers can easily access these algorithms as anonymous guests on the platform for free and reproduce the experiments conducted in this paper.

The well known classification algorithms proposed are *Multiclass Decision Forest*, *Multiclass Decision Jungle*, *Multiclass Neural Network* and *Multiclass Logistic Regression*.

5.1.1. Multiclass Decision Forest (DF)

Decision trees algorithms build a tree-like structure where each node represents a question over an attribute. The answers to that question create new branches to expand the structure until the end of the tree is reached, being the leaf node the one that indicates the predicted class. The *Decision Forest* (DF) algorithm, graphically illustrated in Figure 4, creates several decision trees and votes the most popular output of them. The implementation used in this paper does not directly count the output of them but sum the normalized frequency ($\hat{f}$) of each output in each tree to get the label with more "probability":

$$\hat{f} = \frac{1}{T}\sum_{t=1}^{T} f_t(x); \tag{4}$$

where T is the number of trees and x the probability of each class.

The parametrization followed in this experiment built 8 decision trees with a maximum depth of 32 levels each. The amount of 128 splits are generated per node to select the optimal split. In order to generate a leaf node, just a sample is required. To resample the dataset for each decision tree the *Bagging* method, also known as *bootstrap aggregation* is applied [6].

The bagging method consists of duplicating the original dataset D to a new dataset D' for each decision tree. D and D' have the same size n. When creating and evaluating models the original dataset is divided in two parts: *training* and *evaluation*. The instances used for *training* and for *evaluation* are different in each new dataset D' since they are selected randomly with replacement.

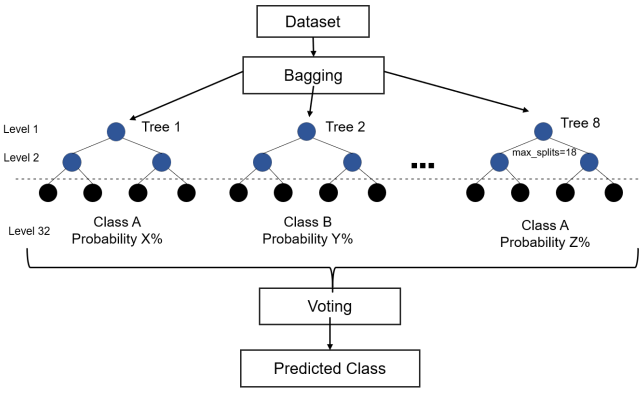

Figure 4: Decision Forest Parametrization

5.1.2. Multiclass Decision Jungle (DJ)

The *Decision Jungle* (DJ) algorithm is an extension of the *Decision Forest* algorithm where each tree is replaced by a DAG (directed acyclic graph). The structure of a DAG is illustrated in Figure 5. It is more memory-efficient because it eliminates the need for repeating leaf nodes and allows branches to merge but, as a disadvantage, takes more computing time.

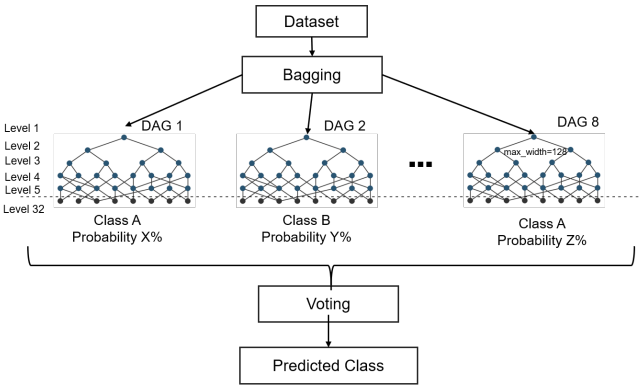

Figure 5: DAG Parametrization

The parametrization followed in this experiment built 8 decision DAGs with a maximum depth of 32 levels each. The maximum width of each decision DAG is 128 and the number of steps for each level optimization of the graph is 2048. As in the case of the *decision forest* algorithm, to resample the dataset for each decision DAG the *Bagging* method is applied.

5.1.3. Multiclass Artificial Neural Network (ANN)

The *Artificial Neural Network* (ANN) algorithm creates a set of interconnected levels, where each level consists of a set of nodes (neurons) that receives input and produces weighted outputs. The nodes of layer 1 are the inputs, the nodes of the last layer are the output and the nodes in between are called "hidden nodes". A neural network can be seen as a weighted directed acyclic graph.

The parametrization followed in this experiment built an ANN with a fully connected structure -i.e., every node of each

https://doi.org/10.1016/j.knosys.2018.10.019

layer is connected with every node of the previous and next layer. There is only 1 hidden layer with 100 nodes. The instances are processed a maximum of 100 times if the ANN has not converged yet. The learning rate is set to 0.1 and the momentum (weight applied to nodes from previous iterations) is set to 0, with these values we want the model to converge fast but avoid local minimums.

5.1.4. Multiclass Logistic Regression (LR)

The *Multiclass Logistic Regression* (LR) algorithm or *Multinomial Logistic Regression*), generalizes logistic regression with more than two possible classes to predict. It predicts the output probability of a class by fitting data to a logistic function. LR aims to build a function that describes the relationships between the input features and the class label.

The parametrization followed in this experiment to build the logistic regression model set the *Optimization Tolerance* $1 * 10^{(-6)}$. This parameter set the threshold that each iteration has to surpass to continue fitting the model to the dataset. The $L1$ and $L2$ regularization weights that deal with the sparsity of data and with no sparse data respectively, are set to 1.

5.2. Experiment Results

In this subsection, we present the results of the experiments conducted that aim to build a useful recommendation model to suggest the most feasible components to users of a component-based interface software application, thereby enhancing the user experience and improving their engagement with the interface. Due to the fact that the number of classes to predict was too high according to the number of instances of our dataset, we reduced the number of components to suggest from 33 to 8. Thus, the recommendation model will suggest only the 8 most used components to users in the ENIA application, which are: `BioReserves`, `Clock2`, `CuttleRoads`, `GeoParks`, `Heritage`, `Twitter`, `Weather` and `Wetlands`.

To build the model, we have selected the four classification algorithms previously discussed, which are *Decision Trees*, *Decision Jungle*, *Artificial Neural Networks* and *Logistic Regression*. We have applied each of these algorithms to the whole dataset after applying the *Feature Engineering* techniques discussed in section 4.2. We have also applied these algorithms to a subset of the dataset we have extracted using the *Mutual Information Score* and *Chi-Squared Statistic Feature Selection* methods described in Section 4.4.

Figure 6 shows the results of the experiments by plotting the accuracy of the models created with each algorithm, with and without reducing the original dataset applying *Features Selection* methods. The figure clearly indicates that the model built with *Artificial Neural Networks* gets higher accuracy than the model built using the other algorithms, while the *Decision Forest* and *Decision Jungle* tree-like algorithms present the worst results, especially the *Decision Jungle*.

Figure 6 also shows that the results are better when applying *Feature Selection* methods to the dataset than using the whole dataset. The exact numerical results can be seen in Table 6. We could foresee that a features subset selection of the original dataset could get better results, given the number of instances and the number of possible hypothesis. A high sparsity of data in high-dimensional spaces does not only computationally penalizes the creation of a model but also affects to its efficiency, decreasing the accuracy of the model.

Algorithm	FS	OA	AA
`Decision Forest`		0.604878049	0.901220
`Decision Forest`	✓	0.646341463	0.911585
`Decision Jungle`		0.398747390	0.849687
`Decision Jungle`	✓	0.434237996	0.858559
`Logistic Regression`		0.718978102	0.929745
`Logistic Regression`	✓	0.740875912	0.935219
`Neural Network`		0.788321168	0.947080
`Neural Network`	✓	0.802919708	0.950730

Table 6: Overall and Average Accuracy (FS: Feature Selection; OA: Overall Accuracy; AA: Average Accuracy)

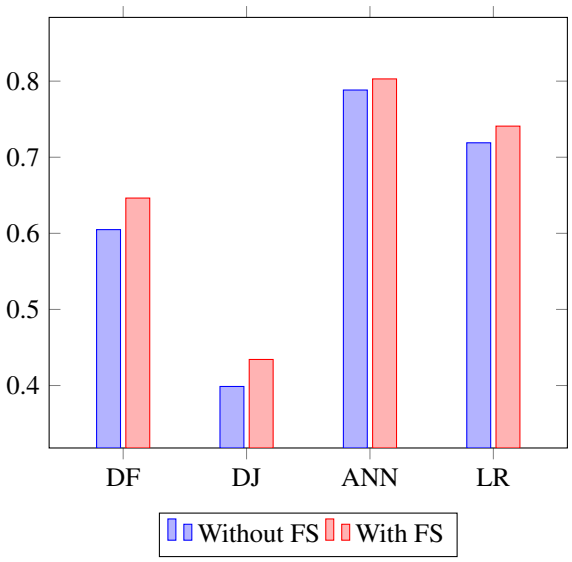

Figure 6: Overall Accuracy

Nevertheless, the sparsity of data itself, is not necessarily a problem if there are insights or knowledge to infer from data, as happens in this scenario. In ENIA, users' behavior clearly follows some patterns, as is borne out by the high accuracy of the models created. Inferring that patterns would be difficult in high-dimensional spaces with the lack of data that our dataset may have but, by reducing the number of features of the dataset with a subset of the most relevance features, we can avoid the problems derived from the curse of dimensionality in high-dimensional spaces. Thus, creating models using a subset of features selected by *Feature Selection* methods we can get greater accuracy. The models we have created obtain good results, reaching 80% accuracy in the case of the *Artificial Neural Network* model. This happens for the models built using every algorithm, obtaining better results in all cases with a subset of the features of the original dataset.

The good results that every model offers, independently of the algorithm used to build it, is remarkable. Even the *Decision Jungle* algorithm applied to the whole dataset, which is

https://doi.org/10.1016/j.knosys.2018.10.019

the model that presents least accuracy, throws an accuracy of 0.398747390. If one of the 8 possible classes were to be chosen randomly, the probability of guessing would be 0.125. It means that the worst model built by applying machine learning techniques triplicates the possibilities of suggesting a suitable component to users when they need it, in the case that components were to be chosen randomly.

In our case study, there is a class imbalance, i.e., the most frequent case `GeoParks` has six times the number of occurrences than the less frequent case `Clock2`, as previously shown in Table 4. The class imbalance is not extremely high but it needs to be considered. For this reason, the *Overall Accuracy* measure is not enough, we also need the *Average Accuracy* measure to really determine the performance of the models. The *Overall Accuracy* indicates the correctly predicted instances (number of correctly predicted items / total of items to predict). The *Average Accuracy* measures the ability to predict classes with few occurrences, difficult to predict (sum of the accuracy for each class predicted / number of classes). Table 6 presents both measures and we can see that given the performance of the *Average Accuracy* the models created using the *Decision Forest* and *Decision Jungle* algorithms have problems predicting the less frequent classes and the *Artificial Neural-Networks* and *Logistic Regression* models behave more consistently across all classes to predict.

Data can be analyzed in more detail by studying the *Confusion Matrix*. A *Confusion Matrix* is a table layout where each row represents the number of instances of each class and each column represents the class that has been predicted by the model. In this table, we can obtain a reliable performance of the model beyond the global accuracy. We have created a customized *Confusion Matrix* (Table 7) that has been divided into two parts. The upper part shows the results of the models created using the whole dataset and the lower part shows the results of the models created using a subset of features. Each cell shows 4 values, corresponding to each algorithm used.

The detailed values of the accuracy of each model predicting each class are shown in Table 7. We have also created a surface chart representing the Confusion Matrix, shown in Figure 7, that helps to easily visualize the results thrown by the models created. As can be noted from this chart, the *Decision Forest* and *Decision Jungle* algorithms, regardless of using the whole dataset or a subset of it, show a high concentration of occurrences that are not correctly predicted at the bottom part of their chart. The bottom part of the graphic corresponds to the instances where the predicted class is `GeoParks`, the most common label. Additionally, the *Decision Jungle* algorithms have serious problems predicting the `Weather`, `Heritage` and `Clock2` components. It looks like these algorithms tend to predict the most common label `GeoParks`. Conversely, the *Artificial Neural Networks* and *Logistic Regression* algorithms present better results, specifically the model created using *Artificial Neural Networks* is particularly suitable for creating the recommendation system that we aim to create in this research. The surface chart shows that this model is very consistent accurately predicting all classes as well as being highly accurate predicting the most uncommon classes. These attributes ensure

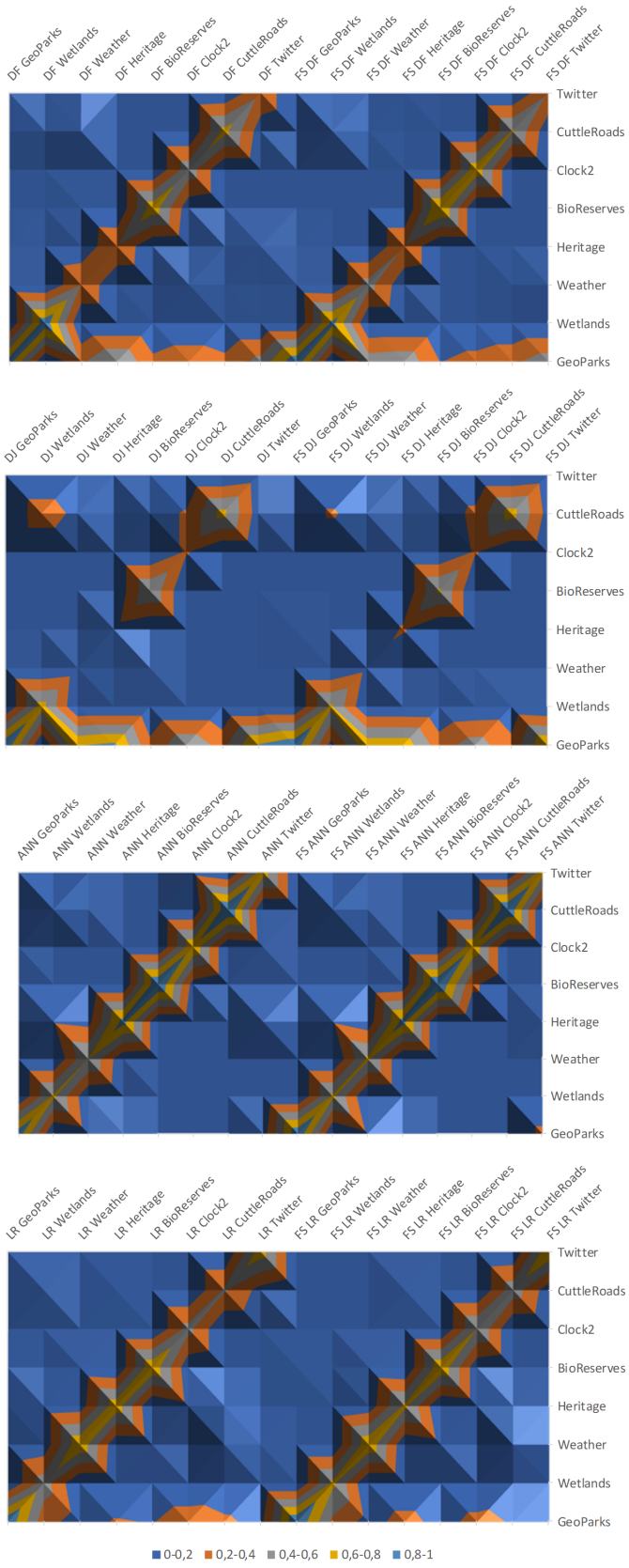

Figure 7: Surface Chart representing the Confusion Matrix

https://doi.org/10.1016/j.knosys.2018.10.019

	BR	C2	CR	GP	HT	TW	WT	WL	
BR	**0.72**	0.00	0.00	0.25	0.02	0.00	0.00	0.00	DF
	0.58	0.00	0.11	0.25	0.00	0.00	0.04	0.00	DJ
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	ANN
	0.75	0.00	0.00	0.19	0.00	0.00	0.06	0.00	LR
C2	0.07	**0.52**	0.00	0.34	0.00	0.00	0.07	0.00	DF
	0.00	**0.22**	0.22	0.56	0.00	0.00	0.00	0.00	DJ
	0.12	**0.75**	0.00	0.00	0.00	0.12	0.00	0.00	ANN
	0.07	**0.47**	0.00	0.33	0.13	0.00	0.00	0.00	LR
CR	0.00	0.00	**0.67**	0.26	0.03	0.03	0.03	0.00	DF
	0.00	0.00	**0.67**	0.33	0.00	0.00	0.00	0.00	DJ
	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	ANN
	0.00	0.00	**0.42**	0.24	0.15	0.00	0.15	0.03	LR
GP	0.00	0.00	0.04	**0.90**	0.02	0.02	0.03	0.00	DF
	0.00	0.00	0.02	**0.97**	0.00	0.00	0.01	0.00	DJ
	0.00	0.04	0.00	**0.92**	0.04	0.00	0.00	0.00	ANN
	0.02	0.00	0.00	**0.97**	0.02	0.00	0.00	0.00	LR
HT	0.00	0.00	0.00	0.60	**0.39**	0.00	0.01	0.00	DF
	0.00	0.00	0.01	0.85	**0.11**	0.00	0.03	0.00	DJ
	0.00	0.00	0.00	0.09	**0.87**	0.00	0.04	0.00	ANN
	0.00	0.00	0.02	0.22	**0.72**	0.00	0.04	0.00	LR
TW	0.00	0.00	0.09	0.47	0.07	**0.36**	0.02	0.00	DF
	0.00	0.00	0.11	0.82	0.00	**0.08**	0.00	0.00	DJ
	0.00	0.05	0.00	0.10	0.15	**0.70**	0.00	0.00	ANN
	0.00	0.00	0.13	0.13	0.00	**0.71**	0.03	0.00	LR
WT	0.02	0.00	0.13	0.37	0.05	0.05	**0.40**	0.00	DF
	0.00	0.00	0.14	0.79	0.00	0.00	**0.07**	0.00	DJ
	0.00	0.00	0.07	0.10	0.13	0.07	**0.60**	0.03	ANN
	0.04	0.00	0.00	0.10	0.10	0.00	**0.76**	0.00	LR
WL	0.00	0.00	0.07	0.00	0.00	0.00	0.00	**0.93**	DF
	0.00	0.00	0.32	0.00	0.00	0.00	0.00	**0.68**	DJ
	0.00	0.00	0.18	0.00	0.18	0.00	0.00	**0.64**	ANN
	0.00	0.00	0.07	0.29	0.07	0.00	0.07	**0.50**	LR

(a) Without Feature Selection

	BR	C2	CR	GP	HT	TW	WT	WL	
BR	**0.81**	0.00	0.00	0.19	0.00	0.00	0.00	0.00	DF
	0.63	0.00	0.07	0.30	0.00	0.00	0.00	0.00	DJ
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	ANN
	0.75	0.00	0.00	0.13	0.00	0.00	0.13	0.00	LR
C2	0.00	**0.72**	0.00	0.28	0.00	0.00	0.00	0.00	DF
	0.00	**0.22**	0.25	0.50	0.00	0.00	0.00	0.03	DJ
	0.25	**0.75**	0.00	0.00	0.00	0.00	0.00	0.00	ANN
	0.07	**0.60**	0.00	0.27	0.07	0.00	0.00	0.00	LR
CR	0.00	0.00	**0.62**	0.36	0.00	0.00	0.03	0.00	DF
	0.00	0.00	**0.69**	0.27	0.00	0.00	0.00	0.04	DJ
	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	ANN
	0.00	0.00	**0.52**	0.18	0.06	0.03	0.15	0.06	LR
GP	0.00	0.00	0.00	**0.94**	0.02	0.01	0.03	0.00	DF
	0.00	0.00	0.03	**0.96**	0.01	0.00	0.00	0.00	DJ
	0.00	0.00	0.00	**0.96**	0.04	0.00	0.00	0.00	ANN
	0.02	0.00	0.00	**0.98**	0.00	0.00	0.00	0.00	LR
HT	0.00	0.00	0.00	0.55	**0.39**	0.01	0.05	0.00	DF
	0.00	0.00	0.02	0.74	**0.22**	0.00	0.01	0.01	DJ
	0.04	0.00	0.00	0.09	**0.87**	0.00	0.00	0.00	ANN
	0.00	0.00	0.04	0.28	**0.64**	0.00	0.04	0.00	LR
TW	0.00	0.00	0.07	0.51	0.00	**0.38**	0.04	0.00	DF
	0.00	0.00	0.14	0.80	0.00	**0.06**	0.00	0.00	DJ
	0.00	0.00	0.00	0.25	0.05	**0.70**	0.00	0.00	ANN
	0.00	0.00	0.08	0.08	0.00	**0.76**	0.08	0.00	LR
WT	0.00	0.00	0.02	0.51	0.00	0.00	**0.48**	0.00	DF
	0.00	0.00	0.11	0.74	0.00	0.00	**0.14**	0.01	DJ
	0.00	0.00	0.07	0.17	0.07	0.07	**0.63**	0.00	ANN
	0.04	0.00	0.00	0.12	0.08	0.00	**0.76**	0.00	LR
WL	0.00	0.00	0.14	0.00	0.00	0.00	0.00	**0.86**	DF
	0.00	0.00	0.23	0.14	0.00	0.00	0.00	**0.64**	DJ
	0.00	0.00	0.18	0.00	0.18	0.00	0.00	**0.64**	ANN
	0.00	0.00	0.00	0.36	0.00	0.00	0.00	**0.64**	LR

(b) With Feature Selection

Table 7: Confusion Matrix (Components {BR: BioReserves; C2: Clock2; CR: CuttleRoads; GP: GeoParks; HT: Heritage; TW: Twitter; WT: Weather; WL: Wetlands} — ML Algorithms {DF: Decision Forest; DJ: Decision Jungle; ANN: Artificial Neural Network; LR: Logistic Regression})

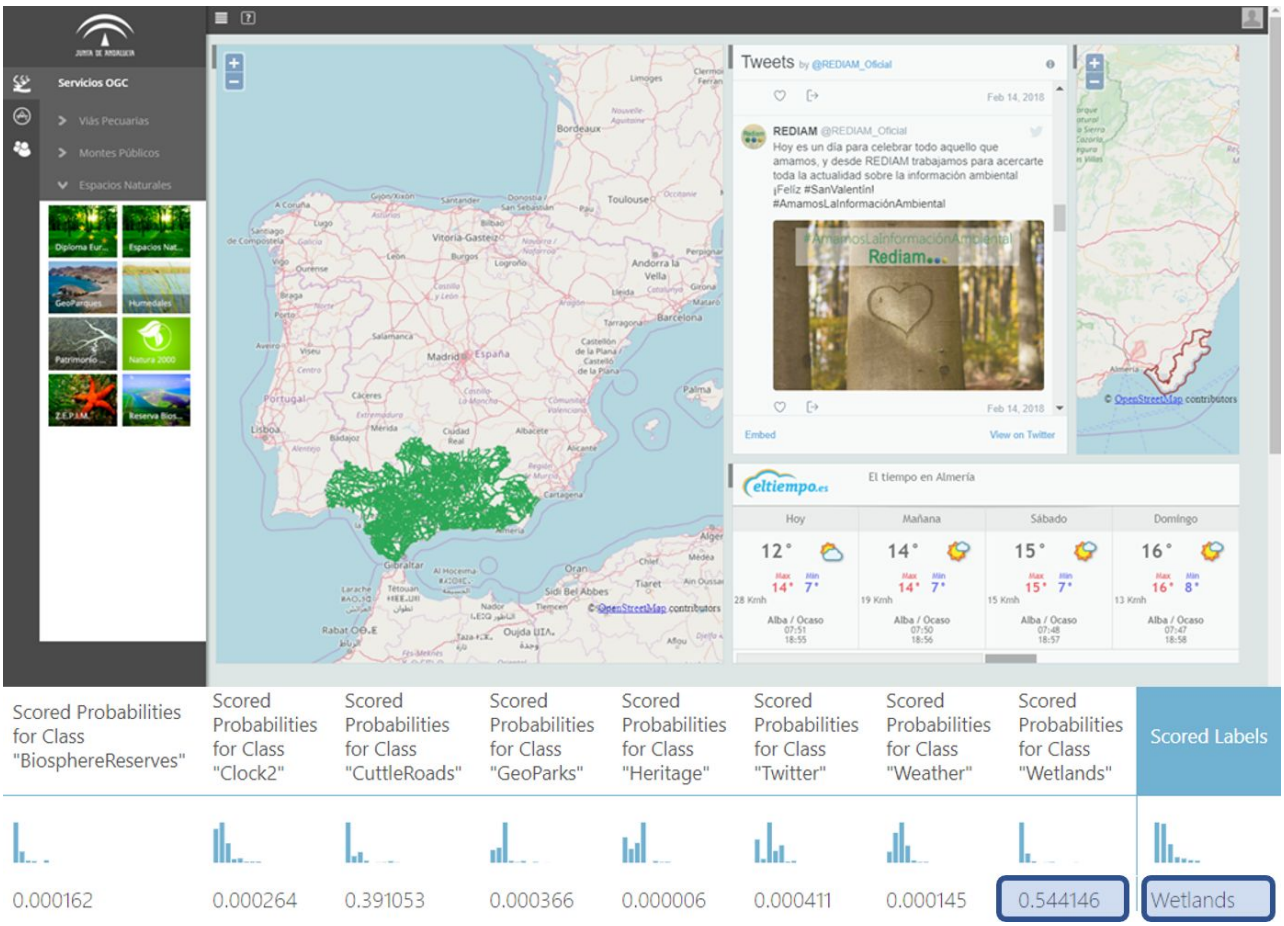

Figure 8: Recommendation model output for a concrete scenario in ENIA

https://doi.org/10.1016/j.knosys.2018.10.019

that the model is reliable and we can conclude that the model created using the *Artificial Neural Networks*, trained with a subset that contains the most significant features extracted using the *Mutual Information Score* and *Chi-Squared Statistic Feature Selection* methods, is the best model created to suggest the most suitable component to users in component apps.

5.3. Deployment

Once the best model is selected, it has to be deployed in the component-based application. The real output of the recommendation system is the list of possible components to suggest, along with the grade of certainty of the recommendation, expressed as a percentage.

In order to avoid the risk of forecasting a component that is not good for a user in a concrete scenario, in ENIA we do not directly suggest the most suitable component to a user according to the output of the model. Instead, we evaluate if the likelihood of the component suggested is higher than 50%. If so, the component will be suggested to the end-user.

Figure 8 illustrates the output of the model for a concrete scenario in the ENIA component-based web application. The scored label is the class with higher probability, in this case, the `Wetlands` component, is the class with the higher score (0.544146). As the likelihood of this class is higher than 50% we proceed to suggest the `Wetlands` component to the end-user in this scenario.

6. Conclusions and Future Work

In this article, we address the problem of creating a recommender system that is able to suggest to users of component-based interfaces, which are the most suitable components for them to use at a specific time. By forecasting the component most closely aligned to each situation, we aim to improve the user experience in the software application and thus, optimize the possibilities of successfully achieving a good position in the increasingly competitive software development market.

We have created several models using a dataset that contains the interactions performed by users in component based applications after applying *Feature Engineering* techniques and *Feature Selection* methods. After this, we have evaluated and compared the models created in terms of overall accuracy and average accuracy, as well as analyzing the confusion matrix that offers the specifying accuracy including the deviation that the model may have predicting a class compared with the labeled class. As a conclusion, all the recommender systems get good results achieving an accuracy of up to 40%, with the *Neural Network* models being the one that gets better performance, yielding an accuracy of up to 80%. It is remarkable that *Feature Selection* methods favorably affect to the results, increasing the accuracy by an average of 3% in all cases.

In future works, it would be interesting to improve the recommendation of components in component-based user interfaces by following these practices:

a) Instead of using just the recommendation model with highest accuracy, the creation of a network that agglutinates all the models and votes the most popular output of each of them, considering the normalized accuracy of each, could improve the overall performance.

b) Since there are components that are more frequently used than others, with a huge difference, adding weights that help to better forecast the less common classes would improve the average accuracy of the recommendation model.

c) Given the rapid evolution of users and components in component-based user interfaces and in the software development market, a nice improvement would be the definition of a coherent strategy to continuously update the recommendation model so it can be adapted to the changes in the software application environment.

Through the implementation of these improvements and the continuous optimization of the dataset, adding more instances and significant features, the recommendation system can be properly optimized over time.

Acknowledgements

This work has been funded by the EU ERDF and the Spanish Ministry of Economy and Competitiveness (MINECO) under Projects TIN2013-41576-R and TIN2017-83964-R. A.J. Fernández-García has been funded by a FPI Grant BES-2014-067974. J.Z. Wang was funded by the US National Science Foundation under Grant No.1027854.

References

[1] ENIA. Environmental Information Agent Project. `http://acg.ual.es/projects/enia/ui/`. Online; last accessed 18 December 2017.

[2] Open Weather Map. `https://openweathermap.org/`. Online; last accessed 18 December 2017.

[3] The Andalusian Environmental Information Network (REDIAM). `http://www.juntadeandalucia.es/medioambiente/site/rediam/`. Online; last accessed 18 December 2017.

[4] Salem Alelyani, Jiliang Tang, and Huan Liu. *Feature Selection for Clustering: A Review*. Chapman&Hall, 2013.

[5] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49(Supplement C):136 – 146, 2015.

[6] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[7] Pedro A. Castillo, Antonio M. Mora, Hossam Faris, J.J. Merelo, Pablo García-Sánchez, Antonio J. Fernández-Ares, Paloma De las Cuevas, and María I. García-Arenas. Applying computational intelligence methods for predicting the sales of newly published books in a real editorial business management environment. *Knowledge-Based Systems*, 115(Supplement C):133 – 151, 2017.

[8] Yehonatan Cohen, Danny Hendler, and Amir Rubin. Detection of malicious webmail attachments based on propagation patterns. *Knowledge-Based S.*, 141:67 – 79, 2018.

[9] Antonio Jesús Fernández-García, Luis Iribarne, Antonio Corral, Javier Criado, and James Z. Wang. Optimally storing the user interaction in mashup interfaces within a relational database. In *Current Trends in Web Engineering*, pages 188–195, Cham, 2016. Springer Int. Pub.

[10] Antonio Jesus Fernandez-Garcia, Luis Iribarne, Antonio Corral, and James Z. Wang. Evolving mashup interfaces using a distributed machine learning and model transformation methodology. In *On the Move to Meaningful Internet Systems*, pages 401–410. Springer Int. Pub., 2015.

[11] Eva-Patricia Fernandez-Manzano, Elena Neira, and Judith Clares-Gavilan. Data management in audiovisual business: Netflix as a case study. *Profesional de la Información*, 25(4):568–576, 2016.

https://doi.org/10.1016/j.knosys.2018.10.019

[12] Antonio Jesús Fernández-García, Luis Iribarne, Antonio Corral, Javier Criado, and James Z. Wang. A flexible data acquisition system for storing the interactions on mashup user interfaces. *Computer Standards & Interfaces*, 2018. (In press).

[13] FitBit. FitBit. Fitness products family to monitor activity, exercise, food, weight and sleep. `https://www.fitbit.com`, 2018.

[14] Flipboard. Flipboard. News and Social Media Aggregation Platfrom. `https://flipboard.com/`, 2018.

[15] Inés M. Galván, José M. Valls, Alejandro Cervantes, and Ricardo Aler. Multi-objective evolutionary optimization of prediction intervals for solar energy forecasting with neural networks. *Information Sciences*, 418-419:363 – 382, 2017.

[16] Salvador Garcia, Julian Luengo, Jose A. Saez, Victoria Lopez, and Francisco Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Trans. on Knowl. and Data Eng.*, 25(4):734–750, April 2013.

[17] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[18] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28, 2014.

[19] Google. Google Now. Personal Assistant. `https://www.google.com/intl/es/landing/now/`, 2018.

[20] Sergio Gómez, Panagiotis Zervas, Demetrios G. Sampson, and Ramón Fabregat. Context-aware adaptive and personalized mobile learning delivery supported by uolmp. *Journal of King Saud University - Computer and Information Sciences*, 26(1, Supplement):47 – 61, 2014.

[21] Petr Hajek and Roberto Henriques. Mining corporate annual reports for intelligent detection of financial statement fraud – comparative study of machine learning methods. *Knowledge-Based Syst.*, 128:139 – 152, 2017.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. *International Statistical Review*, 77(3):482–482, 2009.

[23] Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods and Algorithms*. John Wiley & Sons, Inc., 2002.

[24] Wei-Po Lee, Chun-Ting Chen, Jhih-Yuan Huang, and Jhen-Yi Liang. A smartphone-based activity-aware system for music streaming recommendation. *Knowledge-Based Systems*, 131(Supplement C):70 – 82, 2017.

[25] Alex Liu. Javascript and the netflix user interface. *Commun. ACM*, 57(11):53–59, October 2014.

[26] Nathan Mantel. Chi-square tests with one degree of freedom; extensions of the mantel-haenszel procedure. *J. of the American Statistical Assoc.*, 58(303):690–700, 1963.

[27] P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989.

[28] Microsoft Corporation. Microsoft Azure Machine Learning Studios.

[29] Netflix. Netflix. A component-based video-on-demand streaming platform. `https://www.netflix.com/`, 2018.

[30] Oracle Corporation. MySQL Database.

[31] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205 – 227, 2018.

[32] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[33] Karthik Ramasubramanian and Abhishek Singh. *Feature Engineering*, pages 181–217. Apress, Berkeley, CA, 2017.

[34] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., 1996.

[35] C. Sánchez, N.M. Villegas, and J. Díaz Cely. Exploiting context information to improve the precision of recommendation systems in retailing. *Communications in Computer and Information Science*, 735:72–86, 2017.

[36] J. Vallecillos, J. Criado, N. Padilla, and L. Iribarne. A cloud service for COTS component-based architectures. *Computer Standards & Interfaces*, 48:198–216, 2016.

[37] Jesús Vallecillos, Javier Criado, Antonio Jesús Fernández-García, Nicolás Padilla, and Luis Iribarne. A web services infrastructure for the management of mashup interfaces. In *Service-Oriented Computing – ICSOC 2015 Workshops*, pages 64–75. Springer, 2016.

[38] Norha M. Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173 – 200, 2018.

[39] Shaoqing Wang, Cuiping Li, Kankan Zhao, and Hong Chen. Learning to context-aware recommend with hierarchical factorization machines. *Information Sciences*, 409-410:121 – 138, 2017.

[40] Zexian Wei, Yanxue Wang, Shuilong He, and Jia Bao. A novel intelligent method for bearing fault diagnosis based on affinity propagation clustering and adaptive feature selection. *Knowledge-Based Syst.*, 116:1 – 12, 2017.

[41] Ke Xu, Xushen Zheng, Yi Cai, Huaqing Min, Zhen Gao, Benjin Zhu, Haoran Xie, and Tak-Lam Wong. Improving user recommendation by extracting social topics and interest topics of users in uni-directional social networks. *Knowledge-Based Systems*, 140(Supplement C):120 – 133, 2018.

[42] Shuo Yang, Mohammed Korayem, Khalifeh AlJadda, Trey Grainger, and Sriraam Natarajan. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. *Knowledge-Based Systems*, 136(Supplement C):37 – 45, 2017.

[43] Yudong Zhang, Shuihua Wang, Preetha Phillips, and Genlin Ji. Binary pso with mutation operator for feature selection using decision tree applied to spam detection. *Knowledge-Based Systems*, 64:22 – 31, 2014.

[44] Yao Zhou, Hua Mao, and Zhang Yi. Cell mitosis detection using deep neural networks. *Knowledge-Based Systems*, 137:19 – 28, 2017.

https://doi.org/10.1016/j.knosys.2018.10.019