A New Feature Selection Technique Applied to Credit Scoring Data
Using a Rank Aggregation Approach Based on: Optimization, Genetic Algorithm and Similarity

Bouaguel Waad\textsuperscript{1}, Bel Mufti Ghazi\textsuperscript{2}, Limam Mohamed\textsuperscript{1,3}
\textsuperscript{1} LARODEC, ISG, University of Tunis
\textsuperscript{2} LARIME, ESSEC, University of Tunis
\textsuperscript{3} Dhofar University, Oman

March 31, 2014

Abstract

Credit scoring has been developed as an essential tool especially in the credit departments of banks that have to deal with a huge sum of credit data. A credit scoring model makes loaning process faster. However, it’s nearly unfeasible to analyze this large amount of data, the feature selection techniques has been used to address this issue. Feature selection, is the process of selecting a subset of relevant features for use in model construction. Unlike existing feature selection techniques that causes bias by using distinct statistical properties of data for feature evaluation, feature selection based on rank aggregation are more robust, thus reducing this bias. Over the past years, rank aggregation has emerged as an important tool. Despite its numerous advantages, rank aggregation may be a deep problem. In fact, rankings provided by the different filters may be in many cases incomplete and similar features may be given disjoint ranking. We first consider the rank aggregation problem as an optimization problem, in which we aim to find an optimal list that approximates all the aggregated lists. Then we concentrate on the problem of disjoint ranking, subsequently we perform a new algorithm that eliminate disjoint ranking for similar features and remove the features that bring less information to the target concept. The performance of our approach was tested using four credit datasets and compared
to three individual filters and four well known aggregation techniques. The result indicates that
the proposed technique is more robust across a wide range of classifiers and has higher accuracy
than other traditional feature selection techniques.

1 Introduction

Credit scoring is the term used to describe a statistically derived numeric expression of an appli-
cant’s creditworthiness that is used by credit institutions to access the possibility that this applicant
will refund his credit. For years, creditors have been using credit scoring systems to determine if
a credit applicant will be a good risk for credit (Zhang et al. 2010). Information about the credit
applicant and his credit experiences, like the number and type of accounts he have, whether he
pay his debts by the due, the age of his accounts, etc, is collected from the applicant in a credit
report, generally, using a statistical program and then lenders compare this information to the loan
repayment history of consumers with similar profiles (Tsai and Wu 2008).

The on hand collection of booked loans is used to build a credit scoring model that would be
used to identify the associations between the applicant’s characteristics and how good or bad is the
credit worthiness of the applicant. In order to have parsimonious credit models, we should only
consider a limited number of features (Thomas 2009). By using only a few features to decide upon
credit approval, the scorecard builder will gain more insight into the model and better understand
its working (Rodriguez et al. 2010).

Choosing the appropriate set of features is one of the most interesting and difficult tasks that
have a key effect on the performance of credit scoring models. Typically, a feature selection tech-
nique looks for a suitable subset of features from the original features set. Feature selection algo-
rium can be divided into two categories: filter and wrapper. Filter methods use general character-
istics of the data independently from the classifier for the evaluation process (Forman 2008). The
obtained results are generally a ranked list of features, where the top ranked features are the most
relevant and features at the bottom are not so relevant or totally unwanted. Wrappers, on the other
hand, search for an optimal feature subset by exploiting the resulting classification performance of
a specific classifier. Therefore, a wrapper’s result is a subset of the most relevant features rather
than an ordered list of all the features as given by a filter. Although effective, the exponential number of possible subsets places computational limits on wrapper algorithm which make filter methods more suitable to our study (Wu et al. 2009).

Since to find the best filtering method is usually intractable in real application, an alternative path is to fuse the results obtained by different filtering methods. Combining preference lists from those individual rankers into a single better ranking is known as rank aggregation. Rank aggregation methods have emerged as an important tool for combining information in credit scoring case. The problem of combining the ranked preferences of many filters is a deep problem. The rankings provided by the different filters may be in many cases incomplete, or even disjoint. In fact in the majority of rankings involve a set of similar feature, but despite the similarity between this features they are not ranked similarly which additionally to the problem of incomplete rankings may lead to noisy ranking.

In this chapter we propose to deal with the rank aggregation problem as an optimization problem in which, we aim to find the best list, which would be the closest as possible to all individual ordered lists all together. We use Genetic algorithms (GAs) as a powerful tool for solving the proposed optimization problem. Once a aggregated list is obtained from GAs, we perform a simple approach that extend GAs results to consider similarity between items in the various ranked lists, in addition to their rankings. The intuition is that similar items should receive similar rankings, given an appropriate measure of similarity.

The reminder of the chapter is organized as follows. Section 2 and Section 3, present respectively an overview of the credit scoring and the financial data proprieties, then an overview of filter feature selection framework and rank aggregation and their issues. In Section 4, we develop a method to combine a set of ordered lists of feature based on an optimization function and genetic algorithm. then we present a framework to extend previous methods of rank aggregation based on a similarity study applied to credit scoring. Section 5 gives experimental investigations on four credit datasets. We conclude this chapter with Section 6.
2 The business challenge of credit scoring

According to Volk (2012) credit risk is one of the major issues in financial researches. Over the past few years, many companies fall apart and were forced into bankruptcy or to a significantly constrained business activity, because of deteriorated financial and economic situation. When banks are unprepared to a variation in the economic activity they will probably suffer from huge credit losses. In fact it is very obvious that credit risk increases in economic depression. However, this effect might be augmented when banks experts under or over estimate the creditworthiness of credit applicants. Expressing why some companies or individuals do default while others don’t and what are the main factors that drive credit risk and how to build robust credit model is very important for financial stability.

2.1 background of credit scoring

Credit scoring consists of the evaluation of the risk related to lending money to an organization or a person. In the past few years, the business of credit products increased enormously. Approximately every day, individual’s and company’s records of past lending and repaying transactions are collected and then evaluated (Hand and Henley 1997). This information is used by lenders such as banks to evaluate an individual’s or company’s means and willingness to repay a loan. The set of collected information makes the managers task simple because it helps determine: whether to extend credit duration or to modify a previously approved credit limit, to quantify the probability of default, bankruptcy or fraud associated to a company or a person. When assessing the risk related to credit products, different problems arise, depending on the context and the different types of credit applicants. S. M. Sadatrasoul (2013) summarize the different kind of scoring as follows:

- **Application scoring:** it refers to the assessment of the credit worthiness for new applicants. It quantifies the risks, associated with credit requests, by evaluating the social, demo-graphic, financial, and other data collected at the time of the application;

- **behavioral scoring:** it involves principles that are similar to application scoring, with the difference that it refers to existing customers. As a consequence, the analyst already has
evidence of the borrower’s behavior with the lender. Behavioral scoring models analyze the consumer’s behavioral patterns to support dynamic portfolio management processes;

- **collection scoring:** collection scoring is used to divide customers with different levels of insolvency into groups, separating those who require more decisive actions from those who don’t need to be attended to immediately. These models are distinguished according to the degree of delinquency (early, middle, late recovery) and allow a better management of delinquent customers, from the first signs of delinquency (30-60 days) to subsequent phases and debt write-off;

- **fraud detection:** fraud scoring models rank the applicants according to the relative likelihood that an application may be fraudulent.

no matter what kind of credit is used, data is at the heart of everything in any credit organization does and, over past years, many techniques and expertise in the interpretation and use of credit bureau and clients’ customer data have been implemented for every aspect of the client relationship. In fact analytics turns the collected data into information, which enables organizations to predict how applicants and customers will behave in the future. The business processes in the market of leading industry should be composed of four major parts: the origination of a credit application, customer management, collections & recoveries and the estimation the default probability.

- **Origination:** application scoring in the beginning process. With the applicants’ information, decisions can be made about whether to accept or decline an applicant. This view of each individual can help develop a picture of the potential value of an accepted applicant, to inform decisions about the product and terms offered.

- **Customer management:** behavioral scoring is used throughout the life of a customer relationship to inform management strategies for each customer, whether managing bad customers or extending the relationship with good customers. For organizations that have many relationships with their customers, customer-level scoring brings together the different aspects of the relationship into one complete picture.
• **Collections & Recoveries**: behavioral scoring is also used to prioritize collections activities to maximize recoveries and reduce collections costs.

• **Estimation the default probability**: Estimating the borrower’s risk level, namely the probability of default (PD), by assigning a different PD to each borrower is now widely employed in many banks. The PD indicates that a given counterparty will not be able or willing to meet its obligations. The false estimation of PD leads to unreasonable rating, incorrect pricing of financial instruments and thereby it was one of the causes of the recent global financial crisis.

In this work we will address the application scoring problem also known as consumer credit scoring. In this context the term *credit* will be used to refer to an amount of money that is borrowed to a credit applicant by a financial institution and which must be repaid, with interest, in a regular interval of time. The probability that an applicant will default must be estimated from information about the applicant provided at the time of the credit application, and the estimate will serve as the basis for an accept or a reject decision. According to Hand and Henley (1997) accurate classification is of benefit both to the creditor in terms of increased profit or reduced loss and to the applicant in terms of avoiding over commitment. This activity deciding whether or not to grant credit is generally carried out by banks and various other organizations. It is an economic activity which has seen rapid growth over the last 30 years.

Traditional methods of deciding whether to grant credit to a particular individual use human judgment of the risk of default, based on experience of previous decisions. Nevertheless, economic demands resulting from the arising number of credit requests, joined with the emergence of new machine learning technology, have led to the development of sophisticated models to help the credit granting decision.

The statistical credit scoring models, called scorecards or classifiers, use predictor from application forms and other sources to estimate the probabilities of defaulting. A credit granting decision is taken by comparing the estimated probability of defaulting with a suitable threshold. Standard statistical methods used in the industry for developing scorecards are discriminant analysis, linear regression and logistic regression. Despite their simplicity, Tufféry (2007) and Thomas (2009) show
that both discriminant analysis and logistic regression prediction accuracy is in some particular cases erroneous. Hence, other models based on data-mining methods are proposed. These models do not lead to scorecards but they indicate directly the class of the credit applicant (Jiang 2009). Artificial intelligence methods such as decision trees, artificial neural networks, K-nearest-neighbor and support vector machines (SVM) (Bellotti and Crook 2009) can be used as alternative methods for credit scoring. These methods extract knowledge from training datasets without any assumption on the data distributions. In the industry, the predictor variables are typically called features, a terminology which we shall retain here.

2.2 proprieties of Financial Data

Credit scoring portfolios are frequently voluminous and they are in the range of several thousand, well over 100000 applicants measured on more than 100 variables are quite common (Hand and Henley 1997). These portfolios are characterized by noise, missing values, complexity of distributions and by redundant or irrelevant features (Piramuthu 2006). Clearly, the applicants characteristics will vary from situation to situation: an applicant looking for a small loan will be asked for different information from another who is asking for a big loan. Furthermore, the data which may be used in a credit model is always subject to changing legislation.

Based on the initial application form filed by the credit applicants some are accepted or rejected immediately based on some obvious characteristics as the salary. Further information is then collected on the remaining credit applicants using further forms. This process of collection of the borrower’s information, allows banks to avoid losing time on obvious non worthy applicants as well as permitting a quick decision.

The used data in credit scoring are often categorical. Classically, continuous variables are categorized in order to accelerate the credit models. As any classification problem, choosing the number of appropriate features to be included in the credit model is an important task. One might try to use as many features as possible. However, the more the number of features grows the more computation is required and model accuracy and scoring interpretation are reduced (Liu and Schumann 2005; Howley et al. 2006). There are also other practical issues, in fact with too many
questions or a lengthy vetting procedure, the applicants will deter and will go elsewhere. Based on (Hand and Henley 1997) a standard statistical and pattern recognition strategy here is to explore a large number of features and to identify an effective subset (of say 10 ± 12) of those features for application in practice and for building the credit model.

3 Filter Framework

The basic idea of filter methods is to select the best features according to some prior knowledge. Filter feature selection methods can be grouped into two categories, i.e. feature weighting methods and subset search methods. This categorization is based on whether they evaluate the relevance of features separately or through feature subsets. In feature weighting methods, weights are assigned to each feature independently and then the features are ranked based on their relevance to the target variable. Relief is a famous algorithm that study features relevance (Kira and Rendell 1992). This method uses the Euclidean distance to select a sample composed of a random instance and the two nearest instances of the same and opposite classes. Then a routine is used to update the feature weight vector for every sample triplet and determines the average feature weight vector relevance. Then, features with average weights over a given threshold are selected.

Subset search methods explore all possible feature subsets using a particular evaluation measure. The best possible subset is selected when the search stops. According to Yu and Liu (2003) two existing evaluation measures that efficiently decrease irrelevance and redundancy are: consistency measure and correlation measure (Arauzo-Azofra et al. 2008; Bouaguel et al. 2013a). Yu and Liu (2003) recommended two main approaches to measure correlation, the first one is based on classical linear correlation of two random variables and the second one is based on information theory.

Numerous correlation coefficients can be used under to first approach but the most common is the Pearson Correlation Coefficient (PCC). PCC is a simple measure that has been shown to be effective in a wide variety of feature selection methods (Rodriguez et al. 2010). Formally, the PCC is defined as
\[ PCC = \frac{\text{cov}(X_i, X_j)}{\sqrt{\text{var}(X_i)\text{var}(X_j)}} \]  

where \( \text{cov} \) is the covariance of variables and \( \text{var} \) is the variance of each variable. Simple correlation measure in general measures the linear relationship between two random variables, which may be not suitable in some cases. The second approach based on information theory measures how much knowledge two variables carry about each other. Mutual information (MI) is a well known information theory measure that captures nonlinear dependencies between variables. Formally, the mutual information of two continuous random variables \( X_i \) and \( X_j \) is be defined as:

\[ MI(x_i, x_j) = \int \int p(x_i, x_j) \log \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \, dx_i \, dx_j, \]

where \( p(x_i, x_j) \) is the joint probability density function, and \( p(x_i) \) and \( p(x_j) \) are the marginal probability density functions. In the case of discrete random variables, the double integral become a summation, where \( p(x_i, x_j) \) is the joint probability mass function, and \( p(x_i) \) and \( p(x_j) \) are the marginal probability mass functions.

Feature ranking consists on ranking the features with respect to their relevance, one selects the top ranked features, where the number of features to select is specified by the user or analytically determined.

Many feature selection algorithms include feature ranking as a principal selection mechanism because of its simplicity, scalability, and good empirical success (Caruana et al. 2003).

Let \( X \) a matrix containing \( m \) instances \( x_i = (x_{i1}, \ldots, x_{id}) \in \mathbb{R}^d, \ i = 1, \ldots, m. \) we denote by \( y_i = (y_1, \ldots, y_m) \) the vector of class labels for the \( m \) instances. \( F \) is the set of features \( f_j = (f_1, \ldots, f_d). \)

Feature ranking makes use of a scoring function \( H(j) \) computed from the values \( x_{ij} \) and \( y_i \) using one the criteria discussed above (i.e, weighting, consistency and correlation). By convention, we assume that a high score is indicative of a valuable variable and that we sort variables in decreasing
order of $H(j)$.

Even when feature ranking is not optimal, it may be preferable to other feature subset selection methods because of its computational and statistical scalability: computationally, it is efficient since it requires only the computation of $d$ scores and sorting the scores; Statistically, it is robust against overfitting because it introduces bias but it may have considerably less variance (Hastie et al. 2001).

### 3.1 Selection Trouble and Rank Aggregation

Given the variety of filter based techniques, it is difficult to identify which of the filter criteria would provide the best output for the experiments. The question is then, how to choose the best criterion for a specific feature selection task? In (Wu et al. 2009) we call this problem a selection trouble. There exists no universal solution for this problem unless to evaluate all existing methods and then establish a general conclusion, which is an impossible solution. The best approach is to independently apply a mixture of the available methods and see what results it will yield.

Combining preference lists from those individual rankers into a single better ranking is known as rank aggregation. Rank aggregation methods have emerged as an important tool for combining information in credit scoring case (Bouaguel et al. 2013b; Bouaguel et al. 2013). Ensemble feature selection techniques (i.e rank aggregation) use an idea similar to ensemble learning for classification (Dietterich 2000). In a first step, a number of different feature selectors (i.e rankers) are used, and in a final phase the output of these separate selectors is aggregated and returned as the final ensemble result.

Ensemble methods have been the most applied to bring together a set of classifiers for building robust predictive models. It has been shown that these ensemble classifiers are competitive with other individual classifiers and in some cases are superior. Recently, there have been studies applying the ensemble concept to the process of feature selection (Dittman et al. 2013). Rank aggregation might be used to improve the robustness of the individual feature selection techniques. Different rankers may yield to different ranking lists that can be considered as local optima in the space of feature subsets, and ensemble feature selection might give a better approximation to the
optimal ranking of features. Also, the representational power of a particular feature selector might constrain its search space such that optimal subsets cannot be reached. Ensemble feature selection could help in alleviating this problem by aggregating the outputs of several feature selectors (Saeys et al. 2008).

As we discussed before rank aggregation have many merits. However, with ensemble feature selection the question is how to aggregate the results of individual rankers. A number of different rank aggregation methods have been proposed in the literature: some are easy to set up like the mean, median, highest rank or lowest rank aggregation, and some are less so.

All rank aggregation methods assume that the ranked lists being combined assign a value to each feature, from 1 to \( d \), where the rank 1 is assigned to the most relevant feature, the second best feature is 2, and so on until the least relevant feature is assigned \( d \). Simple rank aggregation technique use straightforward way to find the final aggregated list, in all cases, once each feature has been given a single value based on the mean, median, highest, or lowest value, all features are ranked based on these new values (Dittman et al. 2013). For example, mean aggregation simply finds the mean value of the feature’s rank across all the lists and uses this as that feature’s value. Likewise, median finds the median rank value across all the lists being combined, using the mean of the middle two values if there are an even number of lists. Highest rank and lowest rank use related strategies: either the highest (best, smallest) or lowest (worst, largest) rank value across all the lists is assigned as the value for the feature in question.

Simple ranking methods are easy to set up. However, in many cases it is possible for two features to end up tied, even if this was not the case in any of the lists being combined and even when these features does not have any tie of similarity. Recent work in the area of rank aggregation techniques has centered on developing unique and innovative approaches. These new techniques can focus on different aspects of the ranking process, including comparing results to randomly generated results. Kolde et al. (2012) proposed an approach that detects features that are ranked consistently better than expected under null hypothesis of uncorrelated inputs and assigns a significance score for each feature. The underlying probabilistic model makes the algorithm parameter free and robust to outliers, noise and errors. Other researches focus on giving more weight to top ranking features
or combining well known aggregation techniques in order to enhance each other. In this work we approach rank aggregation from another perspective. In fact we aim to find the best list, which would be the closest as possible to all individual ordered lists all together and this can be seen as an optimization problem. More details will be given in Section 4.

### 3.2 Incomplete Ranking and Disjoint Ranking for Similar Features

According to Sculley (2007) The rankings provided by the different filters may be in many cases incomplete, or even disjoint. In fact Sculley pointed out that incomplete rankings may come in two forms. In the first form, the different filters or some of them may each provide rankings for only the $k$ best features and ignore the remaining features provided in the beginning. In the second form, the used filters may provide complete rankings over a limited subset of available features, due to incomplete knowledge.

Incomplete rankings are common in many financial applications, but still not the only problem with rank aggregation. In fact in the majority of rankings involve a set of similar feature, but despite the similarity between this features they are not ranked similarly which additionally to the problem of incomplete rankings may lead to noisy ranking. Let us give an illustrative example. Assume we have 7 features \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}, were $f_2$ and $f_5$ are highly similar, but not identical. We consider the two following rank lists from two different filters: list one is given by \{f_3, f_2, f_7, f_5\} and list 2 is given bye \{f_2, f_7, f_3, f_4, f_1\}.

If we have no preference of one filter over the other, then standard methods of rank aggregation may interrupt the rankings in the following ways: \{f_2, f_3, f_7, f_5, f_1\} . In this standard aggregation the features $f_2$ and $f_5$ are given divergent rankings, in spite of their high similarity, which make this kind of aggregation unacceptable. A more acceptable ranking that verify our vision and give a closer ranking to similar features, will be \{f_3, f_2, f_5, f_7, f_4, f_1\}. The last aggregation agrees with our intuition that $f_5$ should be ranked behind $f_2$ because they are highly similar so they should have a close ranking. To avoid disjoint ranking for similar features, we present in the Section 4.3 a simple approach that extend any standard aggregation technique in order to take similarity into account.
4 New Approach

In this section we propose a novel approach for filter feature selection. We consider building a two-stage filter feature selection model. In the first step, an optimization function and genetic algorithm are used to resolve the selection trouble and the rank aggregation problem and to sort the features according to their relevance. In the second step, a standard algorithm is proposed in order to resolve the problem of disjoint ranking for similar features and to eliminate the redundant ones.

4.1 Optimization Problem

Rank aggregation provides a mean of combining information from different ordered lists and at the same time, to set their weak points. The aim of rank aggregation when dealing with feature selection is to find the best list, which would be the closest as possible to all individual ordered lists all together.

This can be seen as an optimization problem, when we look at argmin\(_D,\sigma\), where argmin gives a list \(\sigma\) at which the distance \(D\) with a randomly selected ordered list is minimized. In this optimization framework the objective function is given by:

\[
F(\sigma) = \sum_{i=1}^{m} w_i \times D(\sigma, L_i),
\]

where \(w_i\) represent the weights associated with the lists \(L_i\), \(D\) is a distance function measuring the distance between a pair of ordered lists and \(L_i\) is the \(i^{th}\) ordered list of cardinality \(k\) (i.e number of features in each list \(L_i\)). The best solution is then to look for \(\sigma^*\) which would minimize the total distance between \(\sigma^*\) and \(L_i\) given by

\[
\sigma^* = \text{argmin} \sum_{i=1}^{m} w_i \times D(\sigma, L_i).
\]

Measuring the distance between two ranking lists is classical and several well-studied metrics are known (Carterette 2009; Kumar and Vassilvitskii 2010), including the Kendall’s tau distance and the Spearman footrule distance. Before defining this two distance measures, let us introduce some
Let $S_i(1), \ldots, S_i(k)$ be the scores coupled with the elements of the ordered list $L_i$, where $S_i(1)$ is associated with the feature on top of $L_i$ that is most important, and $S_i(k)$ is associated with the feature which is at the bottom that is least important with regard to the target concept. All the other scores correspond to the features that would be in-between, ordered by decreasing importance. For each item $j \in L_i$, $r(j)$ shows the ranking of this item. Note that the optimal ranking of any item is 1, rankings are always positive, and higher rank shows lower preference in the list.

### 4.1.1 Spearman Footrule Distance

Spearman footrule distance between two given rankings lists $L$ and $\sigma$, is defined as the sum, overall the absolute differences between the ranks of all unique elements from both ordered lists combined. Formally, the Spearman footrule distance between $L$ and $\sigma$, is given by

$$Spearman(L, \sigma) = \sum_{f \in (L \cup \sigma)} |r_L(f) - r_{\sigma}(f)|$$  \hspace{1cm} (5)

Spearman footrule distance is a very simple way for comparing two ordered lists. The smaller the value of this distance, the more similar the lists. When the two lists to be compared, have no elements in common, the metric is $k(k + 1)$.

### 4.1.2 Kendall’s Tau Distance

The Kendall’s tau distance between two ordered rank list $L$ and $\sigma$, is given by the number of pairwise adjacent transpositions needed to transform one list into another (Dinu and Manea 2006). This distance can be seen as the number of pairwise disagreements between the two rankings. Hence, the formal definition for the Kendall’s tau distance is:

$$Kendall(L, \sigma) = \sum_{i,j \in (L \cup \sigma)} K,$$  \hspace{1cm} (6)
where

\[
K = \begin{cases} 
0 & \text{if } r_L(i) < r_L(j), r_\sigma(i) < r_\sigma(j) \\
& \text{or } r_L(i) > r_L(j), r_\sigma(i) > r_\sigma(j) \\
1 & \text{if } r_L(i) > r_L(j), r_\sigma(i) < r_\sigma(j) \\
& \text{or } r_L(i) < r_L(j), r_\sigma(i) > r_\sigma(j) \\
p & \text{if } r_L(i) = r_L(j) = k + 1, \\
r_\sigma(i) = r_\sigma(j) = k + 1
\end{cases}
\] (7)

That is, if we have no knowledge of the relative position of \( i \) and \( j \) in one of the lists, we have several choices in the matter. We can either impose no penalty (0), full penalty (1), or a partial penalty (0 < \( p < 1 \)).

### 4.1.3 Weighted Distance

In case, the only information available about the individual list is the rank order, the Spearman footrule distance and the Kendall’s tau distance are adequate measures. However the presence of any additional information about the individual list may improve the final aggregation. Typically with filter methods, weights are assigned to each feature independently and then the features are ranked based on their relevance to the target variable. It would be beneficial to integrate these weights into our aggregation scheme. Hence, the weight associated to each feature consists of taking the average score across all of the ranked feature lists. We find the average for each feature by adding all the normalized scores associated to each lists and dividing the sum by the number of lists. Thus, the weighted Spearman’s footrule distance between two list \( L \) and \( \sigma \) is given by

\[
\sum_{f \in (L \cup \sigma)} |W(r_L(f)) \times r_L(f) - W(r_\sigma(f)) \times r_\sigma(f)|.
\]

= \[
\sum_{f \in (L \cup \sigma)} |W(r_L(f)) - W(r_\sigma(f))| \times |r_L(f) - r_\sigma(f)|.
\] (8)

15
Analogously to the weighted Spearman’s footrule distance, the weighted Kendall’s tau distance is given by:

$$WK(L, \sigma) = |W(r_L(i)) - W(r_L(j))|K.$$  

(9)

4.2 Solution to Optimization Problem Using Genetic Algorithm

The introduced optimization problem in Section 4.1 is a typical integer programming problem. As far as we know, there is no efficient solution to such kind of problem. One possible approach would be to perform complete search. However, it is too time demanding to make it applicable in real applications. We need to look for more practical solutions.

The presented method uses a genetic algorithm for rank aggregation. Genetic algorithms (GAs) were introduced by Holland (1992) to imitate the mechanism of genetic models of natural evolution and selection. GAs are powerful tools for solving complex combinatorial problems, where a combinatorial problem involves choosing the best subset of components from a pool of possible components in order that the mixture has some desired quality (Clegg et al. 2009). GAs are computational models of evolution. They work on the basis of a set of candidate solutions. Each candidate solution is called a ”chromosome”, and the whole set of solutions is called a ”population”. The algorithm allows movement from one population of chromosomes to a new population in an iterative fashion. Each iteration is called a ”generation”. GAs in our case proceeds in the following manner:

4.2.1 Initialization

Once a set of aggregation rank lists are generated by several filtering techniques, it is necessary to create an initial population of features to be used as starting point for the genetic algorithm, where each feature in the population represents a possible solution. This starting population is then obtained by randomly selecting a set of ordered rank lists.

Despite the success of genetic algorithm on a wide collection of problems, the choice of the population size stills an issue. Gotshall and Rylander (2000) proved that the larger the population size, the better chance of it containing the optimal solution. However, increasing population size
also causes the number of generations to converge to increase. In order to have great results, the population size should depend on the length of the ordered lists and on the number of unique elements in these lists. From empirical studies, over a wide range of problems, a population size of between 30 and 100 is usually recommended by Gotshall and Rylander (2000).

4.2.2 Selection

Once the initial population is fixed, we need to select new members for the next generation. In fact, each element in the current population is evaluated on the basis of its overall fitness (the objective function score given in Equation (3)). Depending on which distance is used, new members (rank lists) are produced by selecting high performing elements.

4.2.3 Cross-over

The selected members are then crossed-over with the cross-over probability CP. Crossover randomly select a point in two selected lists and exchange the remaining segments of these lists to create a new ones. Therefore, crossover combines the features of two lists to create two similar ranked lists.

4.2.4 Mutation

In case only the crossover operator is used to produce the new generation, one possible problem that may arise is that if all the ranked lists in the initial population have the same value at a particular rank, then all future lists will have this same value at this particular rank. To come over this unwanted situation a mutation operator is used. Mutation operates by randomly changing one or more elements of any list. It acts as a population perturbation operator. Typically mutation does not occur frequently so mutation is of the order of 0.001 (Gotshall and Rylander 2000).

4.3 A rank aggregation based on similarity

Once an initial ranking InitialR is obtained from the proposed rank aggregation algorithm in Section 4.2, we move to solving the problem of disjoint ranking for similar features. So, we first propose to perform a simple algorithm that give a new ranking that incorporate similarity knowledge. Then
we move to the elimination of redundant feature and then the final feature set is obtained by comparing the relevance of each pair of redundant feature using the relevance to the target class (see Figure 2).

4.3.1 Solution to disjoint ranking for similar features

We take as staring point, the ranked list InitialR, where each item ri of this list, presents the rank of a feature fi. remind that we always give 1 as optimal ranking to the most significative feature, rankings are always positive, and lower rank shows higher preference in the list.

In each iteration we study the similarity between the first feature (i.e the feature with the most
relevance to the target concept and where \( r_{fi} = 1 \) and the remaining features in the aggregated list, for that we use the function \( \text{SIM} \). This function uses \( MI \) as a similarity measure given its efficiency (see Section 3, Equation 2). At first a \((m \times m)\) symmetric positive semi definite matrix describing the similarity among variables is computed using \( MI \) (see Figure 3). Then the function \( \text{SIM} \) compare the similarity between the features using this matrix. If the feature in hand have 80% of mutual information with any of the features in the list \( Initial_R \), the function \( \text{SIM} \) return 'true' elsewhere it returns 'false'. In case the value 'false' is obtained that means that the feature doesn’t have any strong connection with any other features in the list \( Initial_R \) and that this feature is in its appropriate place in the aggregated list. In this case this feature is removed from the initial aggregated list and automatically added to the final list. In case the returned value is 'true' we proceed by a set of steps in order to move the similar features closer and resolve the problem of divergent rankings.

![Similarity Matrix](image)

**Figure 3:** Example of the matrix of similarity produced by the \( \text{SIM} \) function

In order to make the rank of similar features closer in the aggregated list we take the feature in the top of the list and we study the distance in terms of rank between this feature and the feature with the next rank. We also use the function \( \text{PLUS-SIM} \) to study the distance between the feature with the next rank and the feature with the highest similarity with the first feature. More details are given in Algorithm 1, given below with a detailed description of the different functions used in this approach.
Algorithm 1

Require: \( Initial_R\): Initial rank aggregation.
Ensure: \( Final_R\): Final Rank List.

1: \textbf{while} \( Initial_R = \emptyset \) \textbf{do} 
2: \hspace{1em} \( Var = Initial_R[1] \).
3: \hspace{1em} \( Var_list = \text{SUBLIST}(Initial_R, 2) \).
4: \hspace{1em} \textbf{if} \ \text{SIM}(Var, Var_list) == \text{FALSE} \ \textbf{then}
5: \hspace{2em} \( Final_R = \text{CONCAT}(Final_R, Var) \).
6: \hspace{2em} \( Initial_R = Var_list \).
7: \hspace{1em} \textbf{else}
8: \hspace{2em} \( Var_{next} = Var_list[1] \).
9: \hspace{2em} \textbf{if} \ \text{VAR}_{next} == \text{PLUS-SIM}(Var, Var_list) \ \textbf{then}
10: \hspace{3em} \( Final_R = \text{CONCAT}(Final_R, Var) \).
11: \hspace{3em} \( Initial_R = Var_list \).
12: \hspace{2em} \textbf{else}
13: \hspace{3em} \textbf{while} \ \text{VAR}_{next} == \text{PLUS-SIM}(Var, Var_list) \ \textbf{do}
14: \hspace{4em} \textbf{if} \ \text{DIST-POS}(\text{VAR}_{next} , \text{PLUS-SIM}(Var, Var_list), Var_list) > 1 \ \textbf{then}
15: \hspace{5em} \text{PERMUTE}(\text{VAR}_{next}, Var, Initial_R) \).
16: \hspace{4em} \textbf{else}
17: \hspace{5em} \text{PERMUTE}(\text{PLUS-SIM}(Var, Initial_R), Var_{next}, Initial_R) \).
18: \hspace{4em} \textbf{end if}
19: \hspace{3em} \textbf{end while}
20: \hspace{2em} \textbf{end if}
21: \hspace{1em} \textbf{end if}
22: \hspace{1em} \textbf{end while}
23: \hspace{1em} \textbf{Return} \ Final_R.

- \textbf{SIM}(E, L): return : false, true
  
  Takes a parameter list L and a feature E and verify if the feature E has a similarity with one of the elements of the list L. If the similarity with one of the elements of the list is superior to 80 %, the function returns true elsewhere false.

- \textbf{CONCAT} (L, E): return : list
  
  Takes a parameter list L to be concatenated and appends the second argument E into the end of the list L.

- \textbf{POS}(E,L): return: number
  
  Searches for the feature E in the List L, and returns its position in the list L, or zero if the
feature E was not found in L.

- **PLUS-SIM(E, L):** return : feature
  Searches for a feature in the list L with the biggest similarity to the feature E.

- **SUBLIST(L, P):** return : list
  Returns a list of the elements in the list L, starting at the specified position P in this list.

- **REMOVE(E,L)***
  Remove the element E given as argument from the list L.

- **DIST-POS(E1,E2,L):** return : number
  Count the number of position between two given elements E1 and E2 in the list L.

- **PERMUT(E1,E2,L):**
  swap the position of two feature E1 and E2 in the list L.

### 4.3.2 Removing unwanted features

Once the selection trouble is resolved and a consensus list of mutual features is obtained, we come across the issue of choosing the appropriate number of features to retain. In fact a list of sorted features doesn’t provide us with the optimal features subset. In general a predefined small number of features is retained from the consensus list in order to build the final model. If the number of used features is relatively small or big, then the final classification results may be degraded.

Despite the fact that most of the features that had a disjoint ranking in Section 4.3.1 are relevant (see Algorithm 1), the underlying concepts can be concisely captured using only a few features, while keeping all of them has substantially detrimental effect on the credit model accuracy. So while we solve the problem of disjoint ranking, we use a marker to mark each pair of treated feature as similar items. A matrix $S$ is then created in order to stock each pairs of similar item, were each row of $S$ contains a feature and their similar items. Then we study each row of $S$ by looking into the computed \( MI \) in order to identify the feature that supplies the most information about the target class. As a result the feature with the highest \( MI \) is kept and the others similar
features are removed from the aggregated list. Let’s take the illustrative example that we used in Section 3.2. We suppose that after dealing with the problem of disjoint ranking we obtain this list \( \{f_3, f_2, f_5, f_7, f_4, f_1\} \), as we introduced before the features \( f_2 \) and \( f_5 \) are highly similar, looking into the results of \( MI \) we observe that \( f_5 \) obtained the highest \( MI \), consequently \( f_2 \) is removed from the list.

5 Empirical Study

5.1 Datasets

The adopted herein datasets are four real-world datasets: two datasets from the UCI repository of machine learning databases: Australian and German credit datasets, a dataset from a Tunisian bank and the HMEQ dataset. Table 1 displays the characteristics of the datasets that have been used for evaluation.

- Australian presents an interesting mixture of attributes: 6 continuous, 8 nominal and a target attribute with few missing values. This dataset is composed of 690 instances where 306 are creditworthy and 383 are not. All attribute names and values have been changed to meaningless symbols for confidentiality.

- German covers a sample of 1000 of credit consumers where 700 instances are creditworthy and 300 are not. For each applicant, 21 numeric input variables are available, i.e. 7 numerical, 13 categorical and a target attribute.

- HMEQ is composed of 5960 instances describing recent home equity loans where 4771 instances are creditworthy and 1189 are not. The target is a binary variable that indicates if an applicant is eventually defaulted. For each applicant, 12 input variables were recorded where 10 are continuous features, 1 is binary and 1 is nominal.

- Tunisian covers a sample of 2970 instances of credit consumers where 2523 instances are creditworthy and 446 are not. Each credit applicant is described by a binary target variable and a set of 22 input variables were 11 features are numerical and 11 are categorical.
5.2 Experimental setting

As discussed in Section 3, in general MI computation requires estimating density functions for continuous variables. For simplicity, each variable is discretized, knowing that discretization of continuous features depends on the context. In this study, we are in the supervised learning context. The discretization step should be performed prior to the learning process. Several tools can be used for that, and we selected Weka 3.7.0 for its simplicity (Bouckaert et al. 2009).

Our feature selection ensemble is composed by three different filter selection algorithms, Relief algorithm, PCC and MI (see Section 3). These algorithms are available in Weka 3.7.0 machine learning package (Bouckaert et al. 2009).

The aggregation of these filters is first performed by our genetic algorithm approach with Kendall and Spearman distances (i.e GA-K and GA-S) and then compared to the mean, median, highest rank or lowest rank aggregation (see Section 3.1), these aggregation techniques were tested using a Matlab implementation of the R package “RobustRankAggreg” written by Kolde et al. (2012), and also to the results given by the individual approaches. We use in this study three different classifiers, namely Decision trees (DT), Support vector machines (SVM) and the logistic regression (LR). These three classifiers are available in Weka 3.7.0 machine learning package (Bouckaert et al. 2009).

The classification models were created and tested using a 10-fold cross validation, where the different models were trained using 9 of the folds. Then the resulting models were validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure). The performance measure reported by 10-fold cross-validation is then the average of the values computed in the loop. The experiment was performed 30 times.

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Names} & \text{Australian} & \text{German} & \text{HMEQ} & \text{Tunisian} \\
\hline
\text{Total instances} & 690 & 1000 & 5960 & 2970 \\
\text{Nominal features} & 6 & 13 & 2 & 11 \\
\text{Numeric features} & 8 & 7 & 10 & 11 \\
\text{Total features} & 14 & 20 & 12 & 22 \\
\text{Number of classes} & 2 & 2 & 2 & 2 \\
\hline
\end{array}
\]
5.3 Performance Metrics

The performance of our system is evaluated using the standard Information retrieval (IR) performance measures: Precision, Recall and F-measure metrics. The Precision, also known as specificity, measures how often the system is correct. It is calculated as the ratio of the number of credit applicants correctly identified by the model as positives ($TP$) to the total number of credit applicants. The total number of credit applicants is the number of applicants correctly identified as positives plus the number of incorrectly classified applicants ($FP$). Precision is given by

$$Precision = \frac{|TP|}{|TP| + |FP|}. \tag{10}$$

The Recall, also known as true positive rate or sensitivity, measures how often the system correctly finds the right class to a credit applicant. It is defined as the proportion of true positives against potential correct examples. The total number of potential correct examples is the number of correct examples ($TP$) plus the count of applicants that should have been output but not ($FN$). Recall is given by

$$Recall = \frac{|TP|}{|TP| + |FN|}. \tag{11}$$

The F-measure combines recall and precision into a global measure and it is given by

$$F\text{-measure} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{12}$$

5.4 Results

This section contains the results from our experiment regarding the classification performance given in Section 5.3 of four classifiers.

Tables 2 and 3 summarize the results obtained using four datasets, where the best results are shown in bold. Column’s 2, 3 and 4 in Tables 2 and 3 present the precision, recall and F-measure achieved by the different feature selection techniques using respectively the Australian and the HMEQ datasets. Column’s 5, 6 and 7 present the precision, recall and F-measure achieved by the
different feature selection techniques using respectively the German and the Tunisian datasets.

Overall we remark that when we train the different algorithms over the feature subset obtained by the proposed approach, we obtain in the majority of cases, the highest precision rates. In general if we have an algorithm with high precision, we can trust the classification judgments made by it. And we can conclude that the selected features are worthy to be investigated.

Table 2: Results Summary for the Australian and German datasets.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relief</td>
<td>0.885</td>
<td>0.926</td>
<td>0.905</td>
<td>0.556</td>
<td>0.511</td>
<td>0.533</td>
</tr>
<tr>
<td>MI</td>
<td>0.929</td>
<td>0.873</td>
<td>0.902</td>
<td>0.612</td>
<td>0.534</td>
<td>0.572</td>
</tr>
<tr>
<td>PCC</td>
<td>0.926</td>
<td>0.924</td>
<td>0.926</td>
<td>0.721</td>
<td>0.500</td>
<td>0.591</td>
</tr>
<tr>
<td>mean</td>
<td>0.927</td>
<td>0.934</td>
<td>0.931</td>
<td>0.781</td>
<td>0.586</td>
<td>0.656</td>
</tr>
<tr>
<td>median</td>
<td>0.925</td>
<td>0.937</td>
<td>0.931</td>
<td>0.778</td>
<td>0.591</td>
<td>0.671</td>
</tr>
<tr>
<td>highest rank</td>
<td>0.929</td>
<td>0.940</td>
<td>0.934</td>
<td>0.770</td>
<td>0.600</td>
<td>0.674</td>
</tr>
<tr>
<td>lowest rank</td>
<td>0.896</td>
<td>0.975</td>
<td>0.933</td>
<td>0.765</td>
<td>0.602</td>
<td>0.673</td>
</tr>
<tr>
<td>GA-K</td>
<td>0.931</td>
<td>0.953</td>
<td>0.941</td>
<td>0.821</td>
<td>0.706</td>
<td>0.759</td>
</tr>
<tr>
<td>GA-S</td>
<td>0.929</td>
<td>0.883</td>
<td>0.905</td>
<td>0.819</td>
<td>0.708</td>
<td>0.759</td>
</tr>
<tr>
<td><strong>SVM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relief</td>
<td>0.795</td>
<td>0.898</td>
<td>0.843</td>
<td>0.517</td>
<td>0.511</td>
<td>0.514</td>
</tr>
<tr>
<td>MI</td>
<td>0.931</td>
<td>0.870</td>
<td>0.900</td>
<td>0.603</td>
<td>0.534</td>
<td>0.566</td>
</tr>
<tr>
<td>PCC</td>
<td>0.918</td>
<td>0.935</td>
<td>0.927</td>
<td>0.705</td>
<td>0.489</td>
<td>0.577</td>
</tr>
<tr>
<td>mean</td>
<td>0.923</td>
<td>0.943</td>
<td>0.928</td>
<td>0.766</td>
<td>0.552</td>
<td>0.627</td>
</tr>
<tr>
<td>median</td>
<td>0.921</td>
<td>0.945</td>
<td>0.932</td>
<td>0.756</td>
<td>0.560</td>
<td>0.643</td>
</tr>
<tr>
<td>highest rank</td>
<td>0.933</td>
<td>0.940</td>
<td>0.936</td>
<td>0.762</td>
<td>0.623</td>
<td>0.685</td>
</tr>
<tr>
<td>lowest rank</td>
<td>0.894</td>
<td>0.980</td>
<td>0.935</td>
<td>0.708</td>
<td>0.602</td>
<td>0.650</td>
</tr>
<tr>
<td>GA-K</td>
<td>0.945</td>
<td>0.921</td>
<td>0.933</td>
<td>0.823</td>
<td>0.812</td>
<td>0.817</td>
</tr>
<tr>
<td>GA-S</td>
<td>0.943</td>
<td>0.942</td>
<td>0.943</td>
<td>0.812</td>
<td>0.799</td>
<td>0.805</td>
</tr>
<tr>
<td><strong>DT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relief</td>
<td>0.786</td>
<td>0.917</td>
<td>0.846</td>
<td>0.682</td>
<td>0.555</td>
<td>0.669</td>
</tr>
<tr>
<td>MI</td>
<td>0.930</td>
<td>0.870</td>
<td>0.900</td>
<td>0.516</td>
<td>0.534</td>
<td>0.525</td>
</tr>
<tr>
<td>PCC</td>
<td>0.932</td>
<td>0.86</td>
<td>0.905</td>
<td>0.737</td>
<td>0.477</td>
<td>0.579</td>
</tr>
<tr>
<td>mean</td>
<td>0.931</td>
<td>0.890</td>
<td>0.910</td>
<td>0.750</td>
<td>0.542</td>
<td>0.612</td>
</tr>
<tr>
<td>median</td>
<td>0.931</td>
<td>0.888</td>
<td>0.909</td>
<td>0.750</td>
<td>0.545</td>
<td>0.613</td>
</tr>
<tr>
<td>highest rank</td>
<td>0.920</td>
<td>0.943</td>
<td>0.931</td>
<td>0.788</td>
<td>0.605</td>
<td>0.684</td>
</tr>
<tr>
<td>lowest rank</td>
<td>0.900</td>
<td>0.902</td>
<td>0.901</td>
<td>0.700</td>
<td>0.642</td>
<td>0.669</td>
</tr>
<tr>
<td>GA-K</td>
<td>0.946</td>
<td>0.923</td>
<td>0.934</td>
<td>0.792</td>
<td>0.701</td>
<td>0.743</td>
</tr>
<tr>
<td>GA-S</td>
<td>0.952</td>
<td>0.950</td>
<td>0.951</td>
<td>0.756</td>
<td>0.697</td>
<td>0.725</td>
</tr>
</tbody>
</table>

Clearly from both the tables above, the feature subset obtained using the GA-K achieved the highest precision value when trained with the SVM classifier. i.e lowest number of false positive errors committed by this classifier. This was also the case for the other classifiers where GA-K achieved 0.931 of precision with LR classifier for the Australian dataset and 0.821 for the German. GA-K also reached the highest precision with LR classifier for the HMEQ and the Tunisian datasets. The DT classifier was not an exception where GA-K was in lead with by 0.792 precision for the
### Table 3: Results Summary for the HMEQ and Tunisian datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relief</td>
<td>0.663</td>
<td>0.715</td>
<td>0.688</td>
<td>0.827</td>
<td>0.847</td>
<td>0.830</td>
</tr>
<tr>
<td>MI</td>
<td>0.681</td>
<td>0.788</td>
<td>0.730</td>
<td>0.822</td>
<td>0.852</td>
<td>0.826</td>
</tr>
<tr>
<td>PCC</td>
<td>0.838</td>
<td>0.974</td>
<td>0.901</td>
<td>0.833</td>
<td>0.850</td>
<td>0.832</td>
</tr>
<tr>
<td>mean</td>
<td>0.850</td>
<td>0.966</td>
<td>0.904</td>
<td>0.875</td>
<td>0.964</td>
<td>0.917</td>
</tr>
<tr>
<td>median</td>
<td>0.848</td>
<td>0.971</td>
<td>0.905</td>
<td>0.881</td>
<td>0.951</td>
<td>0.914</td>
</tr>
<tr>
<td>highest rank</td>
<td>0.842</td>
<td><strong>0.980</strong></td>
<td>0.905</td>
<td>0.901</td>
<td>0.894</td>
<td>0.897</td>
</tr>
<tr>
<td>lowest rank</td>
<td>0.870</td>
<td>0.880</td>
<td>0.875</td>
<td>0.878</td>
<td>0.888</td>
<td>0.887</td>
</tr>
<tr>
<td>GA-K</td>
<td><strong>0.902</strong></td>
<td>0.972</td>
<td><strong>0.935</strong></td>
<td><strong>0.924</strong></td>
<td>0.902</td>
<td>0.912</td>
</tr>
<tr>
<td>GA-S</td>
<td>0.896</td>
<td>0.955</td>
<td>0.924</td>
<td>0.916</td>
<td>0.943</td>
<td><strong>0.929</strong></td>
</tr>
</tbody>
</table>

| **SVM** |
|---------|-----------|--------|-----------|-----------|--------|-----------|
| Relax   | 0.845     | 0.807  | 0.728     | 0.845     | 0.807  | 0.728     |
| MI      | 0.822     | 0.828  | 0.784     | 0.822     | 0.828  | 0.784     |
| PCC     | 0.822     | 0.828  | 0.784     | 0.822     | 0.828  | 0.784     |
| mean    | 0.830     | **0.987** | 0.902 | 0.830     | 0.987  | 0.902     |
| median  | 0.823     | 0.906  | 0.862     | 0.889     | 0.975  | 0.930     |
| highest rank | 0.905 | 0.945  | 0.924     | 0.922     | 0.907  | 0.914     |
| lowest rank | 0.900 | 0.891  | 0.895     | 0.881     | 0.880  | 0.880     |
| GA-K    | **0.966** | 0.933  | **0.949** | **0.967** | 0.952  | **0.959** |
| GA-S    | 0.942     | 0.940  | 0.941     | 0.966     | 0.923  | 0.944     |

<table>
<thead>
<tr>
<th><strong>DT</strong></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Relief</td>
<td>0.747</td>
<td>0.8</td>
<td>0.736</td>
<td>0.876</td>
<td>0.888</td>
<td>0.882</td>
</tr>
<tr>
<td>MI</td>
<td>0.814</td>
<td>0.831</td>
<td>0.801</td>
<td>0.885</td>
<td>0.883</td>
<td>0.884</td>
</tr>
<tr>
<td>PCC</td>
<td>0.818</td>
<td>0.832</td>
<td>0.798</td>
<td>0.876</td>
<td>0.880</td>
<td>0.879</td>
</tr>
<tr>
<td>mean</td>
<td>0.821</td>
<td><strong>0.981</strong></td>
<td>0.887</td>
<td>0.860</td>
<td><strong>0.962</strong></td>
<td>0.913</td>
</tr>
<tr>
<td>median</td>
<td>0.808</td>
<td>0.926</td>
<td>0.863</td>
<td>0.871</td>
<td>0.899</td>
<td>0.884</td>
</tr>
<tr>
<td>highest rank</td>
<td>0.906</td>
<td>0.921</td>
<td>0.913</td>
<td>0.901</td>
<td>0.907</td>
<td>0.904</td>
</tr>
<tr>
<td>lowest rank</td>
<td>0.842</td>
<td>0.922</td>
<td>0.880</td>
<td>0.889</td>
<td>0.902</td>
<td>0.895</td>
</tr>
<tr>
<td>GA-K</td>
<td>0.920</td>
<td>0.921</td>
<td><strong>0.921</strong></td>
<td><strong>0.922</strong></td>
<td>0.912</td>
<td><strong>0.917</strong></td>
</tr>
<tr>
<td>GA-S</td>
<td><strong>0.923</strong></td>
<td>0.912</td>
<td>0.917</td>
<td>0.917</td>
<td>0.908</td>
<td>0.912</td>
</tr>
</tbody>
</table>

German dataset and 0.922 for the Tunisian dataset. The results for the GA-S came in the second place in term of precision except for DT classifier where the subset of feature that was generated by GA-S achieved the best precision with the Australian and HEMQ datasets.

The mean, median and highest rank aggregation almost perform the same with equal number of minimum false positives. Except for the lowest rank aggregation which give less precision but still better than the results given by the individual filters. Looking now on the recall results’ we remark that the proposed approach also achieves high rate both for Kendall and Spearman distances, which justify the high value for F-measure.

As we examine closely the results, we observe that the mean aggregation method achieve four time the best recall (i.e with LR, SVM and DT for the Tunisian dataset and DT for the HMEQ dataset) followed by lowest rank aggregation (i.e with LR and SVM for the Australian dataset),
GA-K (i.e. SVM and DT for the German dataset) and GA-S (i.e. LR for German dataset and DT for the Australian dataset). In the final place come the median and highest rank aggregation that achieve just for one time the highest rate of recall respectively with (SVM for the Tunisian dataset) and (LR for HMEQ dataset).

The computed values or scores of recall, precision, and the F-measures are used to measure the performance of the feature selection techniques. The differences between any two features selection techniques may be due to chance or may be due to a significant difference between the two feature selection algorithms. To rule out the possibility that the difference is due to chance and to confirm our conclusions, statistical hypothesis testing is used.

A key element in hypothesis testing is obviously the hypothesis. Statistics can be used to reject a hypothesis. In statistical hypothesis testing, it is important to compose the hypothesis in such a way as to be able to use statistics to reject it and to thereby be able to conclude that something of interest is true. The hypothesis formulated to be rejected is called the null hypothesis.

We are interested in determining whether two feature selection techniques are significantly different in their performance. To conclude that a set of feature selection techniques are significantly different, we would formulate null hypotheses of the following form: The absolute value of the difference between a feature selection technique A’s overall Precision, Recall and F-measure score for the features filtering task and a feature selection technique B’s overall Precision, Recall and F-measure score for the same features filtering task is approximately equal to zero. If this null hypothesis can be rejected, then we can conclude that the results obtained by the different feature selection techniques are significantly different.

The ANOVA test is used here in order to evaluate the null hypothesis. The P values are calculated from the ANOVA table. If the computed P value between two feature selection techniques is large, the data do not give any reason to conclude that the means differ and then that the proposed results are ineffective. When the p-value turns out to be less than 0.05, such a result indicates that the observed result would be highly unlikely under the null hypothesis. Tables 5.4-3(c) summarize.

The mean of the dependent variable differs significantly among the levels of Precision, Recall and F-measure.
### (a) P-value for Precision

<table>
<thead>
<tr>
<th>Method</th>
<th>Relief</th>
<th>MI</th>
<th>PCC</th>
<th>Mean</th>
<th>Median</th>
<th>Highest</th>
<th>Lowest</th>
<th>GA-K</th>
<th>GA-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>GA-K</td>
<td>0.022722917</td>
<td>0.020182</td>
<td>0.007056333</td>
<td>0.003680917</td>
<td>0.003839333</td>
<td>0.004955</td>
<td>0.003501583</td>
<td>0.011539571</td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.017023816</td>
<td>0.009612013</td>
<td>0.0032657</td>
<td>0.005461467</td>
<td>0.002424667</td>
<td>0.002553667</td>
<td>0.011539571</td>
<td>0.02789775</td>
</tr>
<tr>
<td>SVM</td>
<td>GA-K</td>
<td>0.021602667</td>
<td>0.02132425</td>
<td>0.00771825</td>
<td>0.004181583</td>
<td>0.005117667</td>
<td>0.006336917</td>
<td>0.00763625</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.004034917</td>
<td>0.004630917</td>
<td>5E-05</td>
<td>0.004332917</td>
<td>0.019224025</td>
<td>0.0311406964</td>
<td>0.000253125</td>
<td>0.08464917</td>
</tr>
<tr>
<td>DT</td>
<td>GA-K</td>
<td>0.00657825</td>
<td>0.03474025</td>
<td>0.00695025</td>
<td>0.005713667</td>
<td>0.006122</td>
<td>0.003724917</td>
<td>0.0067255</td>
<td>0.022012032</td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.004854667</td>
<td>0.007860667</td>
<td>0.028150875</td>
<td>0.0345845</td>
<td>0.0336275</td>
<td>0.01729225</td>
<td>0.022012032</td>
<td></td>
</tr>
</tbody>
</table>

### (b) P-value for Recall

<table>
<thead>
<tr>
<th>Method</th>
<th>Relief</th>
<th>MI</th>
<th>PCC</th>
<th>Mean</th>
<th>Median</th>
<th>Highest</th>
<th>Lowest</th>
<th>GA-K</th>
<th>GA-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>GA-K</td>
<td>0.027019333</td>
<td>0.027275</td>
<td>0.036550917</td>
<td>0.04200425</td>
<td>0.032535</td>
<td>0.025606667</td>
<td>0.017866667</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.011867583</td>
<td>0.013164917</td>
<td>0.046369</td>
<td>0.022896113</td>
<td>0.040682987</td>
<td>0.031192744</td>
<td>0.025650421</td>
<td>0.018225338</td>
</tr>
<tr>
<td>SVM</td>
<td>GA-K</td>
<td>0.028463583</td>
<td>0.024108</td>
<td>0.037638</td>
<td>0.04460025</td>
<td>0.037279</td>
<td>0.023948917</td>
<td>0.026810917</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.016668167</td>
<td>0.031775604</td>
<td>0.023649986</td>
<td>0.018399648</td>
<td>0.005918926</td>
<td>0.009758333</td>
<td>0.008803</td>
<td>0.007265333</td>
</tr>
<tr>
<td>DT</td>
<td>GA-K</td>
<td>0.003965667</td>
<td>0.004696667</td>
<td>0.013053736</td>
<td>0.025723444</td>
<td>0.01298225</td>
<td>0.007036545</td>
<td>0.025861488</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.03291025</td>
<td>0.02436025</td>
<td>0.045858667</td>
<td>0.034193</td>
<td>0.032955667</td>
<td>0.029795667</td>
<td>0.026238917</td>
<td>0.014836917</td>
</tr>
</tbody>
</table>

### (c) P-value for F-measure

<table>
<thead>
<tr>
<th>Method</th>
<th>Relief</th>
<th>MI</th>
<th>PCC</th>
<th>Mean</th>
<th>Median</th>
<th>Highest</th>
<th>Lowest</th>
<th>GA-K</th>
<th>GA-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>GA-K</td>
<td>0.007409583</td>
<td>0.020246333</td>
<td>0.008911125</td>
<td>0.001485125</td>
<td>0.0019845</td>
<td>0.010473625</td>
<td>0.020246333</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.006533583</td>
<td>0.029646125</td>
<td>0.014959625</td>
<td>0.04118675</td>
<td>0.01520425</td>
<td>0.023471125</td>
<td>0.023385667</td>
<td>0.0418295</td>
</tr>
<tr>
<td>SVM</td>
<td>GA-K</td>
<td>0.014440917</td>
<td>0.03224932</td>
<td>0.02222875</td>
<td>0.016583333</td>
<td>0.004739583</td>
<td>0.004339667</td>
<td>0.018614917</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.004633722</td>
<td>0.011598292</td>
<td>0.133370875</td>
<td>0.010661458</td>
<td>0.020266917</td>
<td>0.03773648</td>
<td>0.01654153</td>
<td>0.02222875</td>
</tr>
<tr>
<td>DT</td>
<td>GA-K</td>
<td>0.0478675</td>
<td>0.010467583</td>
<td>0.016347907</td>
<td>0.003160125</td>
<td>0.019938625</td>
<td>6.6125E-05</td>
<td>0.005983437</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GA-S</td>
<td>0.00965825</td>
<td>0.030219</td>
<td>0.02191025</td>
<td>0.021646</td>
<td>0.018894917</td>
<td>0.013582</td>
<td>0.01251025</td>
<td>0.008242917</td>
</tr>
</tbody>
</table>

Table 4: The P-values between the proposed rank aggregation approach and a set of others feature selection techniques.
6 Conclusion

Feature selection is a fundamental step in building robust and simple models for credit scoring. However choosing the appropriate feature selection technique is not an easy task. We investigate in this study the effect of the fusion of a set of ranking techniques. Our work was conducted on two parts. First, we conducted a preliminary study in which, the issue of rank aggregation was first presented as an optimization problem that we resolved using genetic algorithm and distance measures. Second we investigated on resolving the problem of disjoint ranking for similar features and choosing the right number of features from the final ranked list, for that we relate the similarity of the feature to their ranking. We evaluated the proposed approach on four credit datasets, results shown that there is a generally beneficial effect of aggregating feature rankings as compared to the ones produced by single methods and four other well known aggregation methods.

References


