Gestion des règles basée sur l'indice de puissance pour la détection de fraude : Approches supervisées et semi-supervisées

Leopold Ghemmogne Fossi

NNT : 2019LYSEI079

DOCTORAL THESIS

Submitted in cotutelle-procedure

Institut National des Sciences Appliquees de Lyon

ECOLE DOCTORALE ***ED 512*** *– INFORMATIQUE ET MATHEMATQUES DE LYON*

SPECIALITE INFORMATIQUE

DIRECTOR: PROF. HAMAMACHE KHEDDOUCI

Università degli Studi di Milano

DEPARTMENT OF COMPUTER SCIENCE "GIOVANNI DEGLI ANTONI"

CORSO DI DOTTORATO IN INFORMATICA *(XXXI CYCLE) –* ***INF/01***

COORDINATOR: PROF. PAOLO BOLDI

Defended on 30 September 2019, by:

Léopold GHEMMOGNE FOSSI

Power-index based Management of Fraud Detection Rules: Supervised and Semi-supervised Approaches

Supervisors:	Prof. Lionel BRUNIE	INSA de Lyon
	Prof. Ernesto DAMIANI	Università degli Studi di Milano
Cosupervisors:	Dr. Habil. Elöd EGYED-ZSIGMOND	INSA de Lyon
	Dr. Gabriele GIANINI	Università degli Studi di Milano

EXAMINATION COMMITTEE:

Reviewers:	Prof. Bogdan GABRYS	University of Technology Sydney, Australia
	Dr. Paul YOO	Cranfield University, UK
	Dr. Alberto TROMBETTA	Università degli Studi dell'Insubria, Italy
Examiners:	Prof. Sylvie CALABRETTO	INSA de Lyon, France
	Prof. Mariagrazia FUGINI	Politecnico di Milano, Italy
	Prof. Elvinia M. RICCOBENE	Università degli Studi di Milano, Italy
	Prof. Jacques SAVOY	Université de NEUCHÂTEL, Switzerland

Abstract

This dissertation, entitled "Power-index based Management of Fraud Detection Rules: Supervised and Semi-supervised Approaches", deals with credit card fraud detection. According to the European Central Bank, the value of fraud using cards issued in the Single Euro Payments Area (SEPA) amounted to €1.8 billion in 2016 [1]. For financial institutions, it is, therefore, a big challenge on how to reduce fraud on credit cards. In general, fraud detection systems consist of an automatic system made by *if-then-else* rules which control any transaction and trigger an alert when the transaction is considered as suspicious. Then human experts check the alert and decide whether the alert is a true or false positive. The criteria used to select the rules to be kept operational are traditionally based mostly on the performance of individual rules. This approach indeed disregards the non-additivity of the rules.

We propose a novel approach using power indices developed within Coalitional Game Theory (CGT). This approach assigns to the rules a normalized score which quantifies the rule influence on the overall performance of the pool. As indices, we use Shapley Value (SV) and Banzhaf Value (BV). The main applications of such scores are: 1) the support of the decision of whether to keep or drop a rule from the pool; 2) the selection of the k top-ranked rules, so as to work with a more compact rule-set. Using real-world credit card fraud data containing approximately 300 rules and 3.5×10^5 transaction records, we show that: 1) This approach fare better in granting the performance of the pool than the one assessing the rules in isolation. 2) The performance of the whole pool can be achieved, keeping only one-tenth of the rules. We then observe that the latter application can be re-framed in terms of a Feature Selection (FS) task for a classifier: we show that our approach is comparable w.r.t benchmark FS algorithms. Also, we observe that it presents an advantage for the rule management, consisting of the assignment of a normalized score to each rule. This is not the case for most FS algorithms, which only focus on yielding a high-performance feature-set solution.

In another contribution, we propose a new version of Banzhaf Value, i.e., *k-Banzhaf*; this new version outperforms concerning the original one in term of computation time and has comparable performance. While for a set $\mathcal{N}$ of N elements, the normal Banzhaf computes $2^N - 1$ differences, the k-Banzhaf computes only $\binom{n-1}{k-1}$ differences. Finally, we implement a self-training process (a kind of bootstrap) to reinforce the learning process in a machine learning algorithm (Random Forest Classifier). We compare the latter with our three power indices to perform classification on the real-world credit card fraud data used in the first part of the manuscript. As a result, we observe that power indices-based feature selection has comparable results w.r.t benchmark FS algorithms also in self-training process.

[1] https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport201809.en.html#toc1

Keywords: Credit card fraud detection, Coalitional Game Theory, Power Indexes, Shapley Value, Banzhaf Index, Restricted Banzhaf Index, Semi-supervised Learning, Supervised Learning, Self-training.

Résumé

Cette thèse, intitulée "Une approche basée sur la théorie des jeux pour la sélection de fonctionnalités pour une prise de décision multicritère efficace: Quelques cas d'utilisation de la classification", traite de la détection de fraude par carte de crédit. Selon la Banque Centrale Européenne, la valeur des fraudes utilisant des cartes émises dans l'espace unique de paiements en euros (SEPA) en 2016 s'élevait à 1,8 milliard d'euros. Ainsi le défis pour les institutions financières est celui de réduire la fraude sur les cartes de crédit. En règle générale, les systèmes de détection de la fraude sont consistués d'un système automatique construit à base de règles "si-alors" qui contrôlent toutes les transactions en entrée et déclenchent une alerte si la transaction est considérée suspecte. Ensuite, un groupe de personel expert vérifie l'alerte et décide si cette dernière est un vrai posifit ou un faux positif. Les critères utilisés dans la sélection des règles maintenues opérationnelles sont principalement basés sur la performance individuelle des règles. Cette approche ignore en effet la non-additivité des règles.

Nous proposons une nouvelle approche utilisant des indices de puissance, concept développé dans le cadre de la théorie des jeux cooperatifs (CGT). Cette approche attribue aux règles un score normalisé qui quantifie l'influence de la règle sur les performances globales du groupe de règles. Les indice que nous utilisons sont le Shapley Value (SV) et le Banzhaf Value (BV). Les principales applications de ces indices sont: 1) l'aide à la décision de conserver ou de supprimer une règle du groupe; 2) la sélection du nombre k de règles les mieux classées, afin de travailler avec un ensemble de règles plus compact. En utilisant des données réelles de fraude par carte de crédit contenant environ 300 règles et $3,5 \times 10^5$ transactions, nous montrons que: 1) Cette approche permet de mieux exécuter les performances du groupe que celle qui évalue les règles isolément. 2) La performance de l'ensemble des règles peut être atteinte en conservant un dixième seulement des règles. Nous observons ensuite que cette application peut être comsidéré comme une tâche de sélection de caractéristiques pour un classificateur: nous montrons que notre approche est comparable aux algorithmes courants de référence en sélection des caractéristiques (FS). De plus, il présente un avantage dans la gestion des règles, en ce sens qu'il attribue un score normalisé à chaque règle. Ce qui n'est pas le cas pour la plupart des algorithmes de sélection des caractéristiques, qui se concentrent uniquement sur une solution d'ensemble pour obtenir des fonctionnalités hautes performances.

Dans une autre contribution, nous proposons une nouvelle version du Banzhaf Value, à savoir le k-Banzhaf; cette nouvelle version surclasse la première en terme de temps de calcul et possède des performances comparables. Alors que pour un ensemble de N éléments, le Banzhaf normal calcule $2^N - 1$ différences, le k-Banzhaf quant à lui calcule seulement $\binom{n-1}{k-1}$. Enfin, nous mettons en œuvre un processus d'auto-apprentissage (sorte de bootstrap) afin de renforcer le processus d'apprentissage dans un algorithme d'apprentissage automatique (Random Forest Classifier). Nous comparons ces derniers avec nos trois indices de puissance pour effectuer une classification sur les données de fraude par carte de crédit du monde réel utilisées dans la première partie du manuscrit. En conclusion, nous observons que la sélection de caractéristiques basée sur les indices de puissance a des résultats comparables avec les algorithmes de référence en FS ainsi que dans le processus d'auto-apprentissage.

Keywords: Détection de Fraud à la Carte, Théorie des Jeux de Coalition, Indice de Pouvoir, Valeur de Shapley, Indice de Banzhaf, Indice de Banzhaf restreint, Apprentissage Semi-supervisé, Apprentissage supervisé , Auto-apprentissage.

Table of Contents

Page

List of Tables

Table **Page**

List of Figures

Figure **Page**

Part I

Introduction

CHAPTER 1

INTRODUCTION

In the latest years, enterprises and financial institutions are facing the ever-growing presence of credit card payment fraudulent activities. Due to the massive volume of online transactions and due to the high dimensionality of the corresponding records, human experts are not able to check for anomalies among the transactions. Systems for online automatic fraud detection are used to contrast those corrupt activities. To achieve better results, fraud detection systems should be fast and accurate. The less time it takes to identify the scam, the more likely it is, indeed, to reduce the economic damage. This introductory chapter is structured as follows: after discussing the statement of the problem (Section 1), we describe the rule governance context (Section 2); then, we define and describe the feature selection in the self-training context (Section 3); finally, we decline the thesis contribution and outline (Section 4).

1.1 Statement of the problem

Generally, in automatic fraud detection systems, a specific engine scans all the incoming transactions for suspected patterns. Such patterns are encoded under the form of rules, an example of the rule can be: *"If a cardholder runs a transaction for a given amount in a given country and, within the next day, (s)he runs another transaction for another given amount in another given country, then trigger an action"*. So, detecting such a pattern, the system would generate an alert. Rules can be created by human experts or from historical data. Indeed there are two types of rules: expert-driven rules, formulated by credit card fraud experts, that are meant to be specific for a given fraud scenario; and data-driven rules, they are learned from historical data through Machine Learning methods.

If at least one of the rules detects a suspicious transaction, then an alert is generated, i.e., the aggregator of the set of rules is the OR operator. After the alert generation, the system sends a record to expert investigators. Their task consist of choosing suspected transactions from the set of received alarms, carrying on a rapid investigation and assessing the alert as a false or a true one: in the latter case, the corresponding credit card is blocked.

The alert generation process is a specific kind of classification process geared towards providing a suitable input to the following investigation process. As such, it can afford to trade a higher recall at the price of a lower precision since the investigators will complete the assessment using human expert judgment: On the other hand, the volume of alerts has not to be exceedingly high. Therefore, rules with high recall but a very low precision should be avoided. The trade-off among the different requirements of the process needs to be managed through calibration, tuning and to be continuously monitored.

For those reasons, in addition to the automatic alert generation process and the human expert alarm investigation process, a further method has to be carried on in real operational environments: the rule governance. Governance performs the maintenance of the rule pool to keep the desired performance. In practice, the current method establishes that based on reports about the performance of the rule (based on single rule precision and recall) human experts – the rules managers – decide to remove/update an existing rule or to add a new one to cover a novel fraudulent scenario. Rule governance is the focus of the present work.

1.2 The rule governance context

In this section, we describe a real-world fraud detection system similar to the one routinely used by the industrial partner that produced the data set, and the corresponding rule governance process.

1.2.1 The fraud detection process

The credit card payment process triggers the fraud detection system: in correspondence to a card payment attempt, the detecting process of credit card fraud is activated in two situations: the first situation is before each authorization for payment, and the second is after each authorization for payment. In the first case, one speaks of *real-time* (RT) fraud detection context, and in the second case of *near-real-time* (NRT) context (see Figure: 1.1).

By analyzing Figure 1.2, one can see that in the first raw of the main picture corresponding to the real-time fraud detection, we have a little "Data Storage" which collects data coming from : on one side the "Front-office" of the company (which role is to fight against fraud) and, on the other side the "Front-office" of the "Issuer/Acquirer", in this case the platform or the payment terminal of the merchant. These data are typically transactions; then, only "Expert Driven" rules check each transaction and trigger an alert in the case of a suspicious transaction. If it is the case, the transaction is blocked and the card also is blocked.

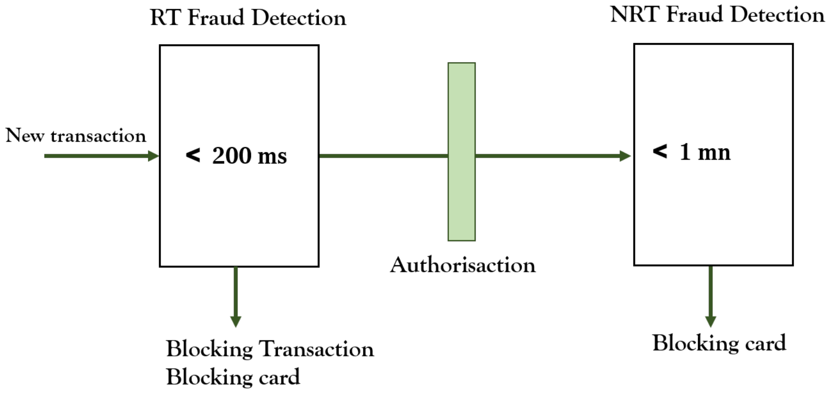

Figure 1.1: Real time (RT) fraud detection : before the authorization is given (200ms to make a decision) and Near-real-time (NRT) fraud detection : after the transaction completed (1mn to make a decision).

While in the second raw corresponding to the near-real-time fraud detection, we have a more prominent "Data storage", which means more data are collected. In the case of near-real-time phase, the data are collected not only from the "Front-office" of the company and the "Issuer/Acquirer", but also from the "Back-office", – data coming from the Back-office are historical data. – These data/transactions would pass through an extra step (Data Management) in which the transactions are categorized (aggregates, profiles and segmentation). Then, in the "Detection " phase, transactions are checked by "Expert Driven" rules and "Data-Driven" rules. Each group of rules can generate an alert in the case of a suspicious transaction. Alerts generated in this phase are potentially shown to the human experts through the "Case Management Tool" for the final decision. But also, during the alert generation, the owner of the card can receive either an email, a short message or a phone call to check if he is the author of that transaction.

By doing fraud detection before the authorization, one has the opportunity to block a fraudulent payment before it is accepted. To allow effective use of RT fraud detection, the system must be: i) fast (RT fraud detection must typically be done in $\sim 200ms$) and ii) precise (a false positive imply that we refuse a legitimate transaction, thus causing an inconvenience to the customer). Rules in an RT fraud detection engine are typically *if-then(-else)* rules designed by the human investigators to block payment requests that are fraud attempts. Due to speed constraint, these rules use mainly information that is available at the time of the transaction request and do little use of the cardholder profile. In practice, several RT rules are simultaneously executed, and a transaction firing any of these rules is refused, i.e., rules are aggregated with an OR operator. The card associated with an allegedly fraudulent transaction is also automatically blocked to prevent future frauds.

All transactions passing the RT fraud detection system without an alert are authorized. The fraud detection activity, however, proceeds beyond this point: all authorized transactions are further analyzed by the NRT fraud detection engine (see Figure: 1.2). In the NRT context, some

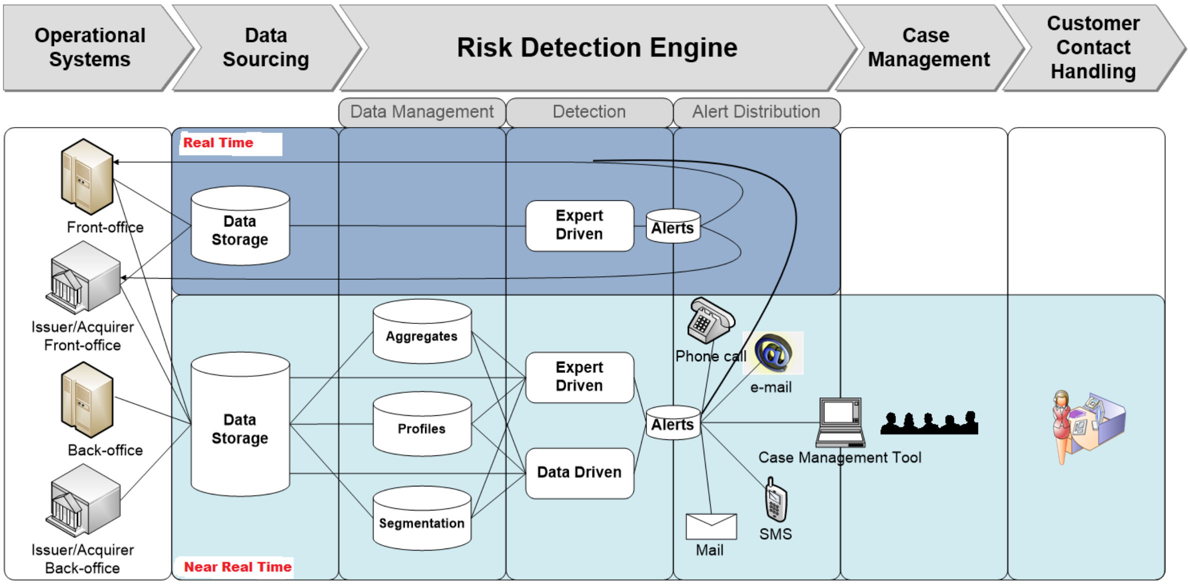

Figure 1.2: The two main phases of the fraud detection process

of the stronger constraints can be relaxed. As the transaction is already accepted, the fraud detection system is allowed more time to make a decision (typically ∼ 1 minute). The system can enrich the transaction data with further features and match the current transaction with the previous purchases and the profile of the cardholder; these can include, simple features, like the average expenditure or the average number of transactions on the same day. It is also possible to add more advanced features see [9], like for example features obtained by graph mining as in [93]. All those aggregated features have the potential to be informative in determining if a transaction is a fraudulent one or not. The features-based rules in NRT context can still be simple *if-then(-else)* rules designed by human investigators; however, they can also be based on advanced machine learning techniques. The NRT rules are simultaneously executed, and any transaction firing at least one of these NRT rules produces an alert (they are aggregated in OR fashion), some rules can also block the card. All alerts generated by the fraud detection system are submitted to the attention of a human investigator, an expert in fraud detection. She estimates with a high degree of confidence whether an alert is a *true positive* or a *false positive*. If an alert was a false positive and blocked the card, the investigator can unblock it. To cover a new fraud scenario, the system uses all the feedback from the investigators i) to automatically retrain the machine learning models if needed, and ii) to generate governance oriented reports: these reports are used by the investigators to manage the pool of rules.

1.2.2 The rule governance process

As mentioned in the previous paragraph, there are two main types of rules, *expert driven* rules which are simple *if-then(-else)* rules and *data driven* rules, obtained by machine learning techniques applied on historical datasets. Both types of rules can be executed in near-real-time. The fraud detection rules pool can contain hundreds of rules; thus, governance is an essential component of the fraud detection process. Based on the governance reports, the investigators can decide to modify/remove an existing rule or to add a new rule in production to cover a new fraud scenario. Classical metrics used in assessing a rule are precision and recall. In particular, other metrics are commonly used like the "*speed-of-detection*" (e.i. the number of fraudulent transactions before an alert is produced) but they are not considered in this work since we cannot compute these metrics with the data provided by our industrial partner. In the standard approach, these metrics are used to assess the rules individually, i.e., in isolation, independently of the performance of the other rules in the pool. We call this approach individual-rule oriented. Within this approach one typically ranks rules according to their performance, measured in isolation and takes removal/update/insertion decisions accordingly: low-rank rules are likely to be deleted, high-rank rules, if already in the pool are preserved, if not in the pool are inserted. This approach assumes implicitly that individual rule optimization obtains the best set of rules. As we are going to demonstrate, this is often not the case.

Individual-level performance and pool-level performance are indeed correlated, but individual rule performances do not add up to the pool-level performance, because a rule can be redundant with another rule or a set of other rules. The point we pursue in this work is that one should not require from a rule to be the best with respect to a performance metrics computed assessing the rule in isolation: one should instead require the rule to best contribute to the rule-pool performance metrics. If one can rank the rules according to their contribution to the pool performance, the elimination of lower-ranked rules, and the insertion of higher-ranked rules will be more effective in improving the pool performance. The fact that the rules that are individually best do not add up to the best rule-pool is intrinsically tied to a fundamental property of the system: the non-additivity of the contributions to the performance metrics, more specifically, in this case, sub-additivity. Given a set of rules, different performance metrics may display different degrees of non-additivity, however, in general, the performance metrics used in practical assessment (precision, recall, F-score, ROC area, etc.) are all non-additive. But let first recall the primary performance metrics.

1.2.3 Classification performance metrics

Given a set of instances that belong either to the class *true* or to the class *false*, and a classifier that assigns to the instances either the class *positive* or the class *negative*,

- a true positive (TP) is an instance classified as positive that belongs to the class true,
- a true negative (TN) is an instance classified as negative that belongs to the class false,
- a false positive (FP) is an instance classified as positive that belongs to the class false,

- a false negative (FN) is an instance classified as negative that belongs to the class true.

Given a set of instances with their class labels and with the classifier tag, one can count the cardinality of each of the four categories:

- the number of true positives is denoted by $\#TP$
- the number of true negatives by $\#TN$
- the number of false positives by $\#FP$
- the number of false negatives by $\#FN$.

The total number of positively tagged instances is denoted by $\#P \equiv (\#TP + \#FP)$
The total number of instances of the class true is denoted by $\#T \equiv (\#TP + \#FN)$
The precision p of the classifier on a data set is defined by $p \equiv \frac{\#TP}{\#P}$
The recall r of the classifier on a data set is defined by $r \equiv \frac{\#TP}{\#T}$
The F-score f of the classifier on a data set is defined by the harmonic average of precision and recall

$$\frac{1}{f} \equiv \frac{(1/p + 1/r)}{2}, \qquad i.e. \qquad f = \frac{2pr}{(p+r)}$$

By construction p, r and f take values in the interval $[0,1]$.

By going further with F-score also called F-measure or F1-score we can say the general term is f_β measure, and the general formula for positive real β is :

$$f_\beta = (1+\beta^2)\frac{p.r}{(\beta^2 p) + r}$$

f_β mesure reaches it best score with $\beta = 1$ and it worst score with $\beta = 0$. In term of errors of type one and errors of type two we have:

$$f_\beta = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + (\beta^2 FN) + FP}$$

Another F-measures are: F_2 measure and $F_{0.5}$ measure which respectively place more emphasis and attenuates the influence of false negatives (FN) .

1.2.4 Non-additivity

To fix the ideas, we focus initially on recall performance metrics. These metrics count the proportion of true positives $\#TP$ identified and retrieved by the rule out of the total number $\#T$ of instances of the true class observed by the rule. Suppose two rules; we call R_1 and R_2, are presented with the same set of instances and that their recalls are respectively.

$$r_{R_1} = \frac{\#TP(R_1)}{T} \quad \text{and} \quad r_{R_2} = \frac{\#TP(R_2)}{T}$$

Consider now the aggregated rule $R_{(1OR2)} \equiv (R_1 \cup R_2)$, defined by the OR of the two. It is straightforward to see that the recall of the aggregated rule is the sum of the two individual rules recalls only in the special case in which the two sets of instances that were retrieved independently by the two rules, do not intersect: in this case, the rules are said orthogonal. Since in the general case for a generic pair of rules orthogonality is not granted, we have that in general, the recall is non-additive, or more specifically sub-additive:

$$r(R_1 \cup R_2) \leq r(R_1) + r(R_2)$$

We observe in passing, that recall is monotonically increasing: adding a rule in OR to a set of rules cannot make the recall decrease: it can only stay the same or increase:

$$r(R_1 \cup R_2) \geq r(R_1)$$

It is easy to show that non-additivity is also a property of precision. Furthermore precision is non-monotonic: OR-ing into a well-performing rule set a poorly performing rule produces an aggregated classifier that is less than well-performing. The non-additivity of r and p and the non-monotonicity of p transfer to their harmonic average, the F-score f.

1.2.5 Quantifying individual contributions in non-additive contexts

The problem of quantifying the contribution of a rule to the performance of the pool is non-trivial and admits several solutions. In an additive context, one could define this contribution as the difference between the pool containing the rule and the pool without only that specific rule: this difference is the added value of the rule with respect to the pool. However, in non-additive contexts, this approach can lead to inconsistencies, as one can see, for instance from the following stylized example. Consider a particular pool in which every rule is redundant with another subset of rules and overall has a positive performance: removing one rule will not change the performance of the pool; thus the rule contribution would be zero.

However, since this holds for all the rules, apparently the performance of the pool receives no contributions from its composing rules, and yet it is positive. This inconsistency makes it clear that the added values of individuals with respect to the whole pool is inappropriate metrics, for non-additive contexts. The problem of quantifying the contribution of individuals to the achievements of an ensemble in a way satisfying some necessary requirements has been faced in Coalitional Game Theory as a part of the problem of assigning a fair share of the surplus produced by coalitions (fair division problem), or as a part of the problem of quantifying the power of individuals in coalitions (power assessment problem or power index problem). We will talk widely about power index in chapter 2, but let say that a solution to the fair division problem fulfilling a basic set of axioms was formulated in 1953 by Lloyd Shapley [80], then used also as a power index, the Shapley-Shubik power index [81].

The solution consists of quantifying the contribution of an individual to a grand-coalition (in our case the contribution of a rule to the whole pool) by taking a weighted average of the added values to all the possible coalitions, where a combinatorial factor weights each added value – taking into account the size of the coalition – In the first part of the contribution, we propose the use of the Shapley value, also known as Shapley-Shubik power index [81], for measuring the contribution of each rule to the global performance of the OR-ed pool as a whole and show that the management of the rules-based on this power index is more efficient than the one based on the performance of the individual rules, assessed in isolation.

1.3 Self-training context and Feature Selection

Since we are in non-additive context, and we would like to reduce the number of operational rules without reducing the performance in term of recall and precision, we have to select the best set of features which in collaboration satisfy our requirement. For this purpose, in this section, we introduce the feature selection activity. We also introduce the self-training concept used in the second part of this dissertation. These two concepts are studied more in detail in Chapter 3. But before, let spend some words saying that the goal is to implement a Shapley value-based feature selection and then use it in the self-training process hopping to improve the result.

1.3.1 Feature Selection

Feature Selection is a process which selects a subset of features defined by one of three approaches: the subset with a specified size that optimizes an evaluation measure, the subset of smaller size that satisfies a specific restriction on the evaluation measure, and the subset with the best compromise among its size and the value of its evaluation measure (general case). In recent years, the application of feature selection methods in many data sets fields has dramatically increased. So, the challenging task in feature selection is how to obtain an optimal subset of relevant and non-redundant features which will give an optimal solution without increasing the complexity of the modelling task.

A suitable feature selection process should not change the original feature set; instead, it should select a subset by eliminating all features whose presence in the dataset do not affect the learning model positively. Thus it preserves the semantics of the characteristics which makes it easily interpretable. It is therefore imperative to select an optimal subset that will represent the original set. Selecting a subset of optimal, relevant, and non-redundant features is a challenging task. If too many features are selected, the classifier has a high workload that can decrease the accuracy of the classification. On the other hand, if only a few features are selected, it is possible to eliminate features that would have increased the accuracy of the classification. It is, therefore, an objective obtain an optimal subset of relevant and non-redundant features that will provide an optimal solution without, reducing the accuracy of the classification.

1.3.2 Self-training

The selection of the best features can help improving the learning process. In the second part of the thesis, we will apply feature selection considering a context in which one has a large amount of data, but only a few percentage are labelled. In this case, one can think to learn from his few labelled data by using self-training, a semi-supervised learning method. We know that supervised learning methods are effective when there is a large number of labelled instances. However, many applications, such as object detection, categorization of documents and Web pages, have very few labelled instances. Those instances are difficult to find, costly or time-consuming to obtain because they require empirical research or experienced annotators. Semi-supervised learning algorithms use also unlabeled data to build a classifier. The goal of semi-supervised learning is to use unlabeled instances and combine the information contained in the unlabeled data with the explicit classification information of the tagged data to improve the performance of the classification. The main difficulty of semi-supervised learning is how to exploit the information from unlabeled data. Some Algorithms of semi-supervised learning have been presented (the Expectation-Maximization (EM) based algorithms, self-training, co-training, Transductive Support Vector Machine (TSVM), Semi-Supervised SVM, graph-based methods, and boosting based semi-supervised learning methods).

Self-training is a method used for semi-supervised learning in many areas, such as object detection and recognition, language processing. A self-training algorithm uses its predictions to assign labels to unlabeled data. Then, a set of data such as the newly labelled ones, which we call a set of high confidence predictions, is selected to be added to the learning set for the next iterations. Self-training uses an iterative method for semi-supervised learning which wraps around a base learner. The performance of the self-training algorithm strongly depends on the newly labelled data selected at each iteration of the learning procedure. This selection strategy is mostly based on a trust in predictions, and it is therefore vitally important for self-learning that the confidence of the prediction is correctly measured.

There is a difference between learning algorithms that produce a probability distribution, such as logistic regression, Bayesian methods, neural networks, margin classifiers, and algorithms usually considered to produce only a classification model like decision trees. Therefore most current approaches to self-training use learning algorithms that produce a probability distribution as a primary learner.

1.4 Thesis contribution and outline

The contribution of the thesis hinges around the following points divided into two main parts:

In the first part:

- We show that a whole rule pool oriented approach can improve the individual rule-oriented approach. To an individual rule, it would better not ask what its performance in isolation is, but rather, what is its contribution to the performance of the rule pool. A single rule effect can be redundant concerning another rule or another subset of rules (other rules can shade a rule), in which case the manager could take steps towards reducing redundancy, thus improving performance.

- The contribution of the individual rule to the rule pool cannot be quantified employing the added value to the pool – computed as the difference of the pool including that rule and the pool obtained removing that rule – because the performance is non-additive (we develop this point further in Chapter 2), thus a suitable metrics should be used for the assessment, that takes into account rules interactions such as redundancy and complementarity: we propose to use the Shapley Value, a *power index* developed in Coalitional Game Theory and defined by a suitably weighted average of the added values of a rule concerning all the possible coalitions of rules [80].

- Computing the exact value of Shapley is exponential, but we adopt an approximated method proposed by Mann and Shapley itself: the Monte Carlo approximation method [62]. It is also the first approximation method. The method is based on Monte Carlo simulation, and it estimates the Shapley value from a random sample of coalitions. Shapley will be studied in detail in chapters two and four.

- The rule managers adopting the Shapley value of a rule as a guide can increase the efficiency of the overall rule pool: for instance, if instead of sorting the rules based on their performance the rule managers sort the rules based on their Shapley value with respect to the performance of the rule pool, and then remove the lowest ranking rules it can be shown that both the performance of the rule pool (precision and recall) improves and the efficiency of the overall process increases. Adopting the Shapley Value ranking allows reducing the number of operational rules from about 300 to about 30, by achieving the same results in terms of precision and recall. Also, it allows assigning a score to the individual rule, which is useful in the monthly revision process, where the performance of each rule is considered by the rule managers that have to make the decision keep/drop.

In the second part:

- We study another power index especially the Banzhaf power index [10]. It was introduced around two decades after the Shapley Index by Jonh F. Banzhaf. While Shapley Index analyses all possible permutations in a coalition, the Banzhaf index considers all combinations into the coalition.

- We propose an improved version of the Banzhaf index called "*restricted Banzhaf index" (k-Banzhaf)*. Given the computation of the Banzhaf value is expensive when the cardinality of the set is large, we introduce this method to overcome the high complexity of the Banzhaf value calculation. We verified that this much quicker approach preserves the feature selection quality. For instance, given a set of N elements, one needs to compute 2^N combinations while our version is based on the cardinality of the subset of interest. Chapter 5 will go more in deep to the concept.

- We propose a power index-based feature selection procedure approach, using the three power indexes studied in the dissertation and compare them with the greedy methods [16] (greedy forward selection and greedy backward elimination)
- In the end, we apply our feature selection method to a self-training process [67, 98], self-training is based on the idea of using few labelled data to improve the classification of unlabelled data.

We observe in the first part that Shapley Value offers an advantage in the rule management process, consisting of the assignment of a normalized score to the individual rule. This is not the case for most Feature Selection algorithms. Also in the second part, we observe that power indexes-based feature selection methods do allow to rank the rules and to select the top-k rules, in analogy to feature selection and achieve performance comparable to other feature selection techniques. Moreover, such power indexes can be interpreted as a summary score of the usefulness of the rule that can be used to assess the rules individually during the periodic rule assessment process. In particular, we observe that the performance of our restricted Banzhaf index is comparable to other power indexes and that some times it is even better.

The remaining of the dissertation is structured as follows: after a review of the related work on Coalitional Game Theory and power indexes in Chapter 2 and review of the related work on Feature Selection and some Machine Learning techniques in Chapter 3, we study the first well-known power index (the Shapley Value) in a real-world case for managing a pool of rule for credit card fraud detection in Chapter 4; then, in Chapter 5, we study the second power index (the Banzhaf Value), propose a modified version (the restricted Banzhaf index) and apply these three power indexes (Shapley, Banzhaf and restricted Banzhaf) to a feature selection in the self-training context. Finally, in Chapter 6 we conclude the dissertation summarizing the results.

LIST OF PUBLICATIONS AND SUBMISSIONS RELATED TO THIS THESIS

- G. Gianini, L. Ghemmogne Fossi, C. Mio, O. Caelen, L. Brunie, and E. Damiani, **Managing a pool of rules for credit card fraud detection by a game theory based approach**, Furture Generation Computer Systems, (2019).
- L. Ghemmogne Fossi, C. Mio, G. Gianini, and E. Damiani, **Gestione basata su power index di un pool di regole per credit card fraud detection**, in Workshop AI for Finance, Commerce and Legal Issues, Ital-IA, Roma, Italy, March 18-19, 2019.

- G.Gianini, C.Mio, L.Ghemmogne Fossi, E. Egyed-Zsigmond, L. Brunie, Greedy methos vs.Power index based methods in Feature Selection, Submited to International Workshop on Applied Computational Intelligence (ACI) at SITIS 2019 (The 15th International Conference on Signal Image Technology and Internet based Systems), Sorrento, 26-29 November 2019.

LIST OF PUBLICATIONS NOT DIRECTLY RELATED TO THIS THESIS

- G. Gianini, C. Mio, L. Ghemmogne Fossi, and E. Egyed-Zsigmond, **A watermark inspection game for IoT settings**, in 2019 IEEE World Congress on Services, SERVICES 2019, Milan, Italy, July 8-13, 2019, pp 29-34.

- R.Gonbin, D. Lefeuvre, A. Alhamzeh, J. Mitrovic, E.Egyed-Zsigmond, and L. Ghemmogne Fossi, **Bots and gender profiling using a multi-layer architecture**, in Working Notes of CLEF 2019- Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019,.

- G. Gianini, C. Mio, L. Ghemmogne Fossi, and A. Rizzi, **Gradient attenuation as an emergent property of reset-based Retinex models**, in 2019 The 11th International Conference on Management of Digital EcoSystems MEDES 2019, Cyprus 12-14, 2019.

Part II

Related literature

CHAPTER 2

Power indexes and Coalitional Game Concepts

Problems of decision making and cost-sharing are crucial issues in general, and particularly in the industrial area (introduction of shareholding in a new market, questions about a tactical or strategic decision regarding merging, acquisitions or firms' control). Power indexes are leading indicators for decision-making, at normative and operative levels. Hence in this chapter, we will introduce some concepts and definitions to help to understand coalitional games and power indexes; namely, Shapley index and Banzhaf index which are our focus in this dissertation. We will come back on these two key concepts in the Chapters 4 and 5. But before that, we can notice that coalitions are usually the result of human relationships, feelings, aversions, external influences, and psychological attitudes than of pure strategic computations. We can also anticipate that Power index is a concept within cooperative game theory used to measure someone's a priori power in certain coalitional games. Moreover, Computing power indexes turn to be a great challenge, so many researchers have been working on to propose an efficient method either computing the exact value or computing an approximated value. In this chapter, we provide some useful tools to better understand Coalitional Game Theory in general and power indexes in particular.

2.1 Coalition analysis concepts in Game Theory

Coalitional (or cooperative) games are a branch of game theory in which one can model cooperation or collaboration among agents. In a coalitional game, we focus on what groups of players can achieve rather than on what individual players can do. Intuitively, stability of a coalitional game means that the game outcome is immune to deviations by groups of players, i.e., no subset of players can unilaterally improve their outcome.

2.1.1 Coalitional games with transferable utility (TU games)

Let a set $\mathcal{N}$ containing $N = |\mathcal{N}|$ agents. A coalition is defined as follows.

Definition 1 (coalition)**.** *A* ***coalition*** *C is a set of agents: $C \in 2^{\mathcal{N}}$ where $2^{\mathcal{N}}$ notes all the subsets of $\mathcal{N}$.*

In Coalitional Game Theory, two main classes of games are considered:
Transferable Utility Games and Non-Transferable Utility Games.

Definition 2 (Transferable Utility Game (TU game))**.** *A* ***Transferable Utility Game*** *is a game where*

- *Two agents can* ***compare*** *their utility*
- *Two agents can* ***transfer*** *some utility*
- *In the transferable utility assumption, the way of* ***sharing the payout*** *of the cooperation is a central issue.*

Definition 3 (Non Transferable Utility game (NTU game))**.** *A* ***Non Transferable Utility Game*** *is a game where*

- *It is not always possible to compare the utility of two agents*
- *It is not always possible to transfer the utility (e.g., no price tags) between two agents. Agents have preference over coalitions.*

A task allocation problem can be use as an informal example of NTU game:

- Consider a set of tasks which require different expertises to be performed
- In the case, tasks may be decomposed
- In the case Agents do not have enough resource on their own to perform a task
- In the case it is necessary to find complementary agents to perform the tasks

In this thesis, we focus on Transferable Utility (TU) games. In a TU game, it is assumed that the earnings of a coalition can be expressed by one numerical value. One may think of this number as an amount of value generated by the process, which can be distributed among the actors in any conceivable way - including negative payments - if the grand coalition is actually formed. In general terms, this number is an *amount of utility* and our assumption in the following is twofold: (i) individual utilities of the business process participants can be expressed in monetary terms (ii) it makes sense to transfer/share this utility among the participants.

Definition 4 (Valuation)**.** *A* ***valuation function*** *μ (or* ***characteristic function****) is a set function that associates a real number $\mu(C)$ to any coalition, i.e.*

$$\mu : \ 2^{\mathcal{N}} \to \mathbb{R}$$

Definition 5 (TU game)**.** *A* ***TU game*** *is a pair $(\mathcal{N}, \mu)$, where $\mathcal{N}$ is a set of agents and μ is a valuation function.*

We are especially interested in games where the collaboration produces a surplus (in a sense they are the non-trivial collaborative games). This idea is captured by the concept of *essential games*.

Definition 6 (Essential game)**.** *A game $(\mathcal{N}, \mu)$ is essential if $\mu(\mathcal{N}) > \sum_{i \in \mathcal{N}} \mu(i)$.*

In an essential game, there is a positive difference between the minimum values that each player can attain individually and the total value that can be attained by the whole grand coalition players: in a TU game this extra wealth is exactly the utility that can be allocated among the players.

2.1.2 Issues in TU games

In coalitional game theory the following two main issues regarding TU games ave to be addressed:

- what coalitions will form or which, once formed, will be stable?
- how to reward each member of a coalition when a task is completed?

2.1.3 Definition and properties of valuations

In algebra (in particular in algebraic geometry or algebraic number theory), a valuation is a function on a field that provides a measure of size or multiplicity of elements of the field. Here, we will consider valuations over the lattice of subsets of a given set, also known as its Boolean or power-set algebra.

2.1.3.1 Measure or capacity

Definition 7 (Measure or Capacity)**.** *A* measure *is a map from the power-set algebra of a set $\mathcal{N}$ to a scalar field, in our case $\mathbb{R}$,*

$$\mu : C \in 2^{\mathcal{N}} \to \mathbb{R}$$

vanishing at the empty set, i.e. such that

1) $\mu(\emptyset) = 0$ *(0-normalization)*

Since no *additivity* or *positivity* is assumed, this map is more soundly called *non-additive signed measure*, or **capacity**.

It is important to remark that a **valuation** in TU game theory is an example of such a measure. Every measure can be taken to represent the characteristic function of a game, thus define unequivocally a game.

In our application field we are mostly interested in *non-negative measures*.

Definition 8 (Non-negativity). *A measure is said* non-negative *if*

2) $\mu(S) \geq 0 \qquad \forall S \in 2^{\mathcal{N}}$ ***(non-negativity)***

hereafter, when using the term *measure* or *capacity*, will always assumed the non-negativity condition.

Definition 9 (Normalization). *A measure is said* normalized*, or* 1*-normalized, if*

3) $\mu(\mathcal{N}) = 1$ ***(normalization)***

If a measure is both 0-normalized and 1-normalized, it is said $(0,1)$-normalized. For instance, standard probability measures are $(0,1)$-normalized.
We are now ready for introducing a simplifying notion, i.e. the one of $(0,1)$-normalized games. For essential games, it is always possible to transform a game $(\mathcal{N},\mu)$ from the non-normalized form to a $(0,1)$-normalized form, through the following transformation.

Proposition 2.1 ($(0,1)-$Normalization of an essential game). *If* $(\mathcal{N},\mu)$ *is an essential game, then it is equivalent to a* $(0,1)$*-normal game* $(\mathcal{N},\mu^*)$ *defined by*

$$\mu^*(C) = \frac{\mu(C) - \sum_{i\in C}\mu(i)}{\mu(\mathcal{N}) - \sum_{j\in\mathcal{N}}\mu(j)}$$

2.1.3.2 Monotonicity

Definition 10 (Monotonicity). *A measure is said to be increasing (decreasing)* monotone *if the measure of a set cannot be smaller (larger) than the measure of one of its subsets.*

4.a) $S \subseteq T \Rightarrow \mu(S) \leq \mu(T)$ ***(increasing monotonicity)***

4.b) $S \subseteq T \Rightarrow \mu(S) \geq \mu(T)$ ***(decreasing monotonicity)***

Strict monotonicity has the signs < and >.

A monotonic measure can be used as the seed for a parametric family of measures satisfying probabilistic inequalities. For instance:

Example 1 (Monotonic measures).

- *the* ***max*** *measure* $\mu(S) \equiv \max_{i\in S}(\mu(i))$ *is monotonically increasing*

- *the **min** measure $\mu(S) \equiv \min_{i \in S}(\mu(i))$ is monotonically decreasing*
- *Probability Measures are monotonically increasing*

For monotonically increasing measures the following inequalities hold

- $\mu(A \cup B) \geq \max(\mu(A), \mu(B))$, since $A \cup B$ is a super-set of both A and B
- $\mu(A \cap B) \geq \min(\mu(A), \mu(B))$, since $A \cap B$ is a subset of both A and B

Definition 11 (Capacity). *A capacity is a measure holding the following properties:*

- $(0,1)$-*normalized*
- *monotone*

Example 2 (Capacities). *For instance*

- *The "Glove Game" defined as:*
  Let L and R be two disjoint companies (players), who make gloves; the company (players) L make only left gloves, while the company R make only right gloves. The value of a coalition depends on the number of pairs the companies L and R are able to form. We assume that each company endowed with a single glove. Supposing that L = {1, 2}; R = {3} and the value of each pair is 1, the resulting game is:

  - $N = \{1,2,3\}$
  - $\mu(\emptyset) = \mu(1) = \mu(2) = \mu(3) = \mu(12) = 0,$
  - $\mu(13) = \mu(23) = \mu(123) = 1$

- *Similarly, the "Airport Runway Game" introduced by S. C. Littlechild and G. Owen [58] in 1973 is defined as*
  An airport who needs to build a runway for 4 different aircraft types, Hence the building cost associated with each aircraft respectively A, B, C, D is 8, 11, 13, 18. In this case, we have:

  - $N = \{A,B,C,D\}$
  - $\mu(\emptyset) = 0, \mu(1) = 8/18, \ \mu(2) = 11/18, \ \mu(3) = 13/18, \ \mu(4) = 18/18$
  - *and* $\mu(S) = \max_{i \in S} \mu(i)$

2.1.3.3 Additivity, non-additivity, super- and sub-additivity

Definition 12 (Additivity). *A measure is said to be* additive *if, for every disjoint pair of sets S, T, it maps their union into the sum of their individual measures, i.e. if*

5) $\mu(S \cup T) = \mu(S) + \mu(T) \qquad \forall S, T \text{ s.t. } S \cap T = \emptyset$ *(**additivity**)*

A measure is said to be non-additive if *at least for a pair of subsets* the above condition does not hold.

Example 3 (Additive and non-additive measures). *For instance*

- *Let a measure count the number of pairs in a set* $\mu(A) = |A|(|A|-1)$*, then it is non-additive.*
- *The classical probability measures are examples of additive measures.*

If a coalitional game is defined by an additive measure, then it is said additive or **inessential**, since it is trivial from the game theoretic point of view (in an inessential game, forming a coalition does not bring any benefit to the participants).
It is easy to see that **additivity** implies **monotonicity**, but not the other way round. Non-additive monotonic measures generalize the additive ones.

The requirement of additivity (countable or finite) of classical measures are based on the assumption that disjoint sets are **non-interactive** with respect to the measured property. This assumption is too restrictive in some application contexts.

A monotone measure is able to capture the following situations relating to two disjoint sets A and B:

a) $\mu(A \cup B) > \mu(A) + \mu(B)$ (positive or cooperative interaction, synergy)

b) $\mu(A \cup B) = \mu(A) + \mu(B)$ (non-interactivity of A and B)

c) $\mu(A \cup B) < \mu(A) + \mu(B)$ (negative or inhibitory interaction)

An additive theory, such as probability theory, based on classical measure theory, is capable of capturing only situation (b). The theory of monotone measures provides a considerably broader framework.
The wide range of application of non-additive measures is due to the fact that in any situation with more than minimal complexity the effect of the ensemble of parts is different from the sum of the effects of the parts taken individually.
Notice that in a specific setting there could be at the same time positive interactions between some sets and negative interactions between other sets. They allow representing interaction among the distinct elements.
For example, we might have

- $\mu(A \cup B) > \mu(A) + \mu(B)$ (positive interaction between A and B) and
- $\mu(A \cup C) < \mu(A) + \mu(C)$ (negative interaction between A and C).

Nonetheless, in some remarkable games, the interaction is always cooperative, while in other remarkable games the interaction is always inhibitory.

Definition 13 (Super-additivity). *A measure is said **super-additive** if for all the pairs of disjoint sets the measure of the union is not smaller than the sum of the measures of the individual sets:*

6) $\mu(S \cup T) \geq \mu(S) + \mu(T) \quad \forall S, T \text{ s.t. } S \cap T = \emptyset$ *(**super-additivity**)*

Example 4 (Super-additive measures). *For instance*

- *A measure counting the number of pairs in a set* $\mu(A) = |A|(|A|-1)$.
- *The Glove Game measure is super-additive.*

Definition 14. *A measure is said **sub-additive***
if for all the pairs of disjoint sets the measure of the union is not greater than the sum of the measures of the individual sets:

7) $\mu(S \cup T) \leq \mu(S) + \mu(T) \quad \forall S, T \text{ s.t. } S \cap T = \emptyset$ *(**sub-additivity**)*

Example 5 (Sub-additive measures). *For instance*

- *The* min *function:* $\mu(S \cup T) \equiv \min(\mu(S), \mu(T))$.
- *The* max *function:* $\mu(S \cup T) \equiv \max(\mu(S), \mu(T))$.
- *The Airport Runway Game is sub-additive.*

Sub-additive measures and super-additive measures are two special types of monotone measures. Other special types include classical, additive measures (i.e. probability measures), the classical (crisp) possibility and necessity measures. Each special type of monotone measures can be used for formalizing a certain type of uncertainty. A remarkable type of uncertainty can be formalized based on a property stronger than super-additivity: modularity.

2.1.3.4 Modularity, super- and sub-modularity

While additivity/non-additivity represent the behaviour of a measure with respect to the union of **disjoint sets**, *modularity*, refers to the union of **any pairs of sets**.

Definition 15 (Modularity). *A measure is said to be* modular *if the following inclusion-exclusion equation holds for any pairs of sets*

- $\mu(S \cup T) = \mu(S) + \mu(T) - \mu(S \cap T) \qquad \forall S,T \in 2^{\mathcal{N}}$ *(modularity)*

It is easy to see that if a measure is **additive** it is also **modular**. We are now ready to define super-modularity

Definition 16 (Super-modularity of order 2, a.k.a. convexity, a.k.a. 2-monotonicity)**.** *A measure is said super-modular if in the inclusion-exclusion relation the = sign is replaced by ≥:*

- $\mu(A \cup B) \geq \mu(A) + \mu(B) - \mu(A \cap B) \qquad \forall A,B \in 2^{\mathcal{N}}$ *(super-modularity)*

Since it involves pairs of sets it is also called *super-modularity of order* 2.

- For monotone measures, super-modularity is a condition stronger than super-additivity, i.e. super-modular measures are monotone non-decreasing. Super-modular measures of order 2 are also called 2-monotone measures.

- Super-modularity, captures the *increasing returns* property: for any set $X \subseteq Y \subseteq \mathcal{N}$, and any $i \in \mathcal{N}$, the inequality

$$\mu(Y \cup i) - \mu(Y) \ \geq \ \mu(X \cup i) - \mu(X)$$

  expresses the fact that the marginal contribution of an element i to a set is a non-decreasing function of the set. Rearranging the definition one gets $\mu(Y \cup i) \ \geq \ \mu(Y) + \mu(X \cup i) - \mu(X)$, i.e the previous expression if one takes $A = Y$ and $B = (X \cup i)$.

- In Game Theory, the super-modularity of a game (a.k.a. *convexity*), captures the intuitive notion that the game incentives for joining a coalition increase as the coalition grows.

Definition 17 (Sub-modularity (of order 2))**.** *A measure is said Sub-modular if in the inclusion-exclusion relation the = sign is replaced by ≤:*

- $\mu(S \cup T) \leq \mu(S) + \mu(T) - \mu(S \cap T) \qquad \forall S,T \in 2^{\mathcal{N}}$ *(sub-modularity)*

- For monotone measures, sub-modularity is a condition stronger than sub-additivity, i.e. sub-modular measures are monotone non-increasing.

- Sub-modularity, can be used to capture the *diminishing returns* property: for any set $A \subseteq B \subseteq \mathcal{N}$, and any $i \in \mathcal{N}$, the inequality

$$\mu(B \cup i) - \mu(B) \ \leq \ \mu(A \cup i) - \mu(A)$$

  expresses the fact that the marginal value of an element i to a set A is a non-increasing function of the set A.

- In coalitional Game Theory, the sub-modularity of a game (a.k.a. *concavity*), captures the intuitive notion that the game incentives for joining a coalition decrease as the coalition grows.

This property can be easily generalized to higher orders. For instance, super-modularity of order 3 is defined as follows.

Definition 18 (Super-modularity of order 3, also known as 3-monotonicity)**.**

$$\begin{aligned}\mu(A\cup B\cup C) \;\geq\; & \mu(A)+\mu(B)+\mu(C)\\ & -\mu(A\cap B)-\mu(A\cap C)-\mu(B\cap C)\\ & +\mu(A\cap B\cap C) \qquad\qquad \forall A,B,C\in 2^{\mathcal{N}}\end{aligned}$$

Every super-modular measure of order 3 is also super-modular of order 2. The above expression can be rewritten more compactly by considering the families of 3 sets and denoting the members of each family by $A_1,A_2,A_3 \in 2^{\mathcal{N}}$

$$\mu\left(\bigcup_{i=1}^{3} A_i\right) \geq \sum_{\emptyset\neq I\subset\{1,2,3\}} (-1)^{(|I|+1)}\mu\left(\bigcap_{i\in I} A_i\right) \qquad A_1,A_2,A_3\in 2^{\mathcal{N}}$$

where I is a set of indexes. The corresponding order 3 sub-modularity can be defined through the above expressions, by changing $\geq$ into $\leq$.

All the property described above can be generalized to any order $k \leq n$ by considering the families of k sets and denoting the members of each family by $A_1,A_2,\ldots,A_k \in 2^{\mathcal{N}}$. For the sake of brevity we consider only super-modularity, the corresponding sub-modularity definitions can be obtained by changing $\geq$ into $\leq$.

Definition 19 (Super-modularity of order k, a.k.a. k-monotonicity)**.** *A non-additive measure is said to be order k super-modular if for all collections of k subsets the following relation holds*

$$\mu\left(\bigcup_{i=1}^{k} A_i\right) \geq \sum_{\emptyset\neq I\subset\{1,2,\ldots,k\}} (-1)^{(|I|+1)}\mu\left(\bigcap_{i\in I} A_i\right) \qquad A_1,A_2,\ldots,A_k\in 2^{N}$$

where I is a set of indexes.

Every k-monotone measure of order $k>2$ is also of order $k' \leq k$, but, a capacity of order k is not necessarily a measure of any higher order of monotonicity.

Definition 20 (Totally monotone measures)**.** *In a finite universe of size $N = |\mathcal{N}|$ a k-monotone measure with $k = N$ is said* totally monotone *(or ∞-monotone).*

2.1.4 Coalition stability concepts

We are now ready to study allocations that determine stable coalitions. Specifically, let the characteristic function μ of a game (the $(2^N - 1)$ *valuations* for all the possible non-empty coalitions) be given. Consider for the sake of simplicity the grand coalition $\mathcal{N}$. We adopt the convention that the valuation of such a coalition is equal to one: $\mu(\mathcal{N}) = 1$. This value represents the total returns earned by the coalition in this specific game. One can assign to each player $i \in \mathcal{N}$ of the grand coalition as a payoff a share x_i of the total returns, and ask if the proposed solution has determinate properties.

Definition 21 (Feasibility)**.** *A solution is feasible if it does not exceed the total worth of the grand coalition:*

$$\sum_{i \in \mathcal{N}} x_i \leq \mu(\mathcal{N})$$

Definition 22 (Efficiency)**.** *A solution is efficient if the payoff distribution is an allocation of the entire worth of the grand coalition to all agents:*

$$\sum_{i \in \mathcal{N}} x_i = \mu(\mathcal{N})$$

Definition 23 (Anonymity)**.** *A solution is independent of the names of the players.*

Definition 24 (Individual rationality)**.** *A solution is individually rational if each agent obtains at least its self-value as a payoff.*

$$\forall i \in \mathcal{N} \qquad x(i) \geq \mu(\{i\})$$

Definition 25 (Imputation)**.** *An imputation is a payoff distribution x that is*

- efficient
- *and* individually rational

Definition 26 (Coalitional rationality/group rationality)**.** *A solution is group-wise rational if each group obtains at least its self-value as the sum of its component's payoffs*

$$\forall C \subseteq \mathcal{N} \qquad \sum_{i \in C} x_i \geq \mu(C)$$

Indeed, on one hand, if $\sum_{i \in C} x_i < \mu(C)$, no group would be ready to accept to be part of C.

Group rationality ensures individual rationality as a special case.

Occasionally, hereafter, we will use the shorthand notation $x(C) \equiv \sum_{i \in C} x_i$: in those terms efficiency is expressed by the condition $x(N) = \mu(N)$, while group rationality is expressed by the condition $x(C) \geq \mu(C), \;\; \forall C \subseteq \mathcal{N}$.

2.1.5 The Core

Let us consider a TU game $(\mathcal{N},\mu)$ and its grand coalition. The notion of Core, first introduced by Gillies [32], is a natural way to define stability: a payoff distribution lies in the Core when no sub-group of agents has any incentive to form a different coalition.

Definition 27 (Core)**.** *The core is the set of payoff allocations satisfying* efficiency *and* coalitional rationality*. The core is a stability concept were no agents prefer to deviate to form a different coalition.*

$$Core(\mathcal{N},\mu) \equiv \left\{ x \in \mathbb{R}^N \,\middle|\, \left(\sum_{i\in\mathcal{N}} x_i = \mu(\mathcal{N}) \right) \wedge \left(\forall C \subseteq \mathcal{N} \qquad \sum_{i\in C} x_i \geq \mu(C) \right) \right\}$$

Equivalently one can say that the core is the collection of group-rational imputations.

Example 6 (Core of some 2-player games)**.** *For instance*

- *Let us consider the following two-player game* $(\{1,2\},\mu)$ *where* $\mu(1)=5$*,* $\mu(2)=5$*, and* $\mu(1,2)=20$*. The core of the game is a segment defined as follows:*

  $$Core(\{1,2\},\mu) = \{(x_1,x_2) \in \mathbb{R}^2 | x_1 \geq 5, x_2 \geq 5, x_1+x_2 = 20\}.$$

  Notice incidentally that the segment contains the point $(10,10)$*: due to the symmetry of the two players' contributions this can be taken as a fair allocation, while all the other allocations in the core are, to some degree, unfair.*

- *Let us consider the "glove game"* $(\{1,2,3\},\mu)$ *where* $\mu(1)=\mu(2)=\mu(3)=0$*,* $\mu(1,2)=\mu(1,3)=1$*,* $\mu(2,3)=0$ *and* $\mu(1,2,3)=1$ *. The core of the game is defined by:*
  $Core(\{1,2,3\},\mu) = \{(x_1,x_2,x_3) \in \left(\mathbb{R}^+\right)^3 \mid x_1+x_2 \geq 1, x_1+x_3 \geq 1, x_1+x_2+x_3 = 1\} = (1,0,0)$*. Here the core is a single point. Notice incidentally that the corresponding allocation is not fair.*

The core condition for coalition stability is so strong (the condition $\sum_{i\in C} x_i \geq \mu(C)$ has to be true for all the subsets C of $\mathcal{N}$), that some games may have an empty core: not all the players and groups of players can be satisfied simultaneously. The Core is empty when at least one player is dissatisfied by the payoff allocation. Such player can be said to "block" the coalition; in other words, she can raise an objection against the formation of such coalition, or threaten to leave the coalition if the coalition is already in place. When focusing on the grand coalition, the core is the set of allocations such that the grand coalition cannot be blocked.

The literature on TU games provides several sufficient conditions for the core of a game to be non-empty, or for the core of a game to be empty (for a summary see for instance Owen [37] or Peleg and Sudhölter [71]). For instance, the core of inessential games, i.e. those games such that $\sum_{i\in\mathcal{N}} \mu(i) > \mu(\mathcal{N})$, is trivially empty; on the other hand convex games have a non-empty core. The Bondareva and Shapley Theorem [79] provides a characterization of the non-empty core games using the concept of balancedness of the game (see [71]).

2.1.6 Games with coalitional structure

So far we focused on the formation of the grand coalition, i.e. the one formed by all participants. This corresponds to the practical situation in which the super-additivity of the valuation function is either explicitly stated or implicitly assumed. However, when the valuation function is not super-additive, players may have an incentive to form a different partition. Furthermore, sometimes some coalitions are excluded by necessity (e.g. impossibility to measure or communicate) or due to an exogenous choice.

To model these cases one has to consider *coalitional structures* (CS): the concept was introduced by Auman and Dreze [4]. A coalitional structure S is a partition of the grand coalition: S is a CS if $S = \{C_1, C_2, \dots, C_m\}$, with $\cup_{k=1}^{m} C_k = \mathcal{N}$ and $\forall i \neq j \, C_i \cap C_j = \emptyset$. e.g., $\{\{1,2,4\},\{3,6\},\{4\},\{5,8\}\}$ is a coalition structure for $n = 8$ agents.

Definition 28 (Game with coalitional structure)**.** *A game with coalitional structure is a triplet $(\mathcal{N}, \mu, S)$, where $(\mathcal{N}, \mu)$ is a TU game and S is a specific CS. In addition, transfer of utility is allowed only within coalitions of S and not between coalitions of S.*

Notice that the problems of deciding which coalition to form and how to share the coalition's revenue so as to achieve stability are independent and decoupled. Also in the previous section, the CS was fixed: it consisted of the trivial partition defined by the grand coalition. As before, here we address the problem of the stability of individual coalitions, by examining the payoff distributions within each coalition and checking efficiency and group rationality conditions.

Definition 29 (Feasible payoff)**.** *Let $(\mathcal{N}, \mu, S)$, be a TU game with CS. The set of feasible payoff distributions is*

$$X_{(\mathcal{N},\mu,S)} = \{x \in \mathbb{R}^N | \forall C \in S, x(C) \leq \mu(C)\}$$

In analogy to the previous definition of efficiency (where the condition was $x(N) = \mu(N)$), but restricted to the coalitions allowed by the CS we define the following.

Definition 30 (Efficiency with respect toa CS)**.** *A payoff distribution x is efficient with respect to a CS S when*

$$\forall C \in S \qquad x(C) = \mu(C)$$

In the context of CS, a payoff distribution is an imputation if it is efficient with respect to the current CS and it fulfils individual rationality (i.e. $\forall i \in \mathcal{N}, x_i \geq \mu(i)$). We denote set of all the imputations for a CS by $Imp(S)$. The definition of the core in TU games with CS is the following.

Definition 31 (Core of a game with CS)**.** *Let $(\mathcal{N}, \mu, S)$, be a TU game with CS. The core of the game is the set of imputations $x \in Imp(S)$ that are group rationale for the coalitions of the CS.*

2.2 Some definitions related to the Shapley Value concept

Actors in our coalitional setting are not only concerned with their individual economic advantage; rather they are typically sensitive to comparative distributional justice. In other words they value fairness, and their behaviour is negatively affected when they experience unfair treatment. Lack of fairness can be considered as a cause for dysfunctional behaviour of an actor. We will use perceived unfairness towards a group of actors to study also the possibility of dysfunctional behaviour of coalitions. Thus we recall the main concepts and introduce the necessary extensions. The definitions of fairness referring to revenue distributions are several. A specific set of requirements broadly accepted as a definition of fair redistribution, and taken as axioms has been shown by Shapley [80] to correspond, given a game, to exactly one allocation: this allocation is called Shapley Value. Hereafter we recall the axioms and the Shapley solution.
We point, in passing, to the fact that in general, the allocation x corresponding to the Shapley Value is not in the core. It belongs to the core for particular classes of games, such as convex games.

2.2.1 Quantifying individual contributions in non additive contexts

As we already discussed in Section 1.2.5, the problem of quantifying the contribution of a rule to the performance of the pool is non-trivial and admits several solutions. In an additive context, one could define this contribution as the difference between the pool containing the rule and the pool without only that specific rule: this difference is the added value of the rule with respect to the pool. However, in non-additive contexts, this approach can lead to inconsistencies, as one can see, for instance from the following stylized example. Consider a particular pool in which every rule is redundant with another subset of rules and overall has a positive performance: removing one rule will not change the performance of the pool; thus the rule contribution would be zero.

We are interested in optimizing the performance of coalitions of rules, aggregated in OR. To that purpose, we would like to assess the contribution of individual rules to the performance of the coalition. The performance metrics we consider are precision, recall and F_β-score.

2.2.2 Marginal contributions

The problem of quantifying the contribution of a rule to the performance of the pool is non-trivial and admits several solutions. In an additive context one could simply define this contribution as the *marginal contribution* (also known as added value) of the rule with respect to the pool, i.e., the difference between the pool containing the rule and the pool without only that specific rule. However, in non-additive contexts, this approach can lead to inconsistencies, as one can see, for instance, from the following stylized example. Consider a special rule pool, which, overall, has a positive performance and in which every rule is made redundant by another subset of rules. Removing one rule will not change the performance of the pool, thus the marginal value of each individual rule would be zero; since this holds for all the rules, apparently the performance of the pool receives no contributions from its composing rules: and yet the performance is positive. This inconsistency makes it clear that the added values of individuals with respect to the whole pool is an inappropriate metric, in non-additive contexts. Nonetheless, *marginal contributions* are essential parts of the definition of more appropriate assessment metrics. We can define them formally as follows.

Given a set function, or measure, $\mu : \mathscr{C} \in 2^{\mathscr{R}} \to \mu(\mathscr{C}) \in \mathbb{R}$, mapping any coalition $\mathscr{C}$ of rules into a real value $\mu \in \mathbb{R}$, representing a performance index (precision, recall or F_β-score), we can define the *marginal contribution* $\Delta_i^{[\mu]}(\mathscr{C})$ of a rule i to a rule coalition (with respect to that measure) as

$$\Delta_i^{[\mu]}(\mathscr{C}) \equiv \Delta^{[\mu]}(\mathscr{C}, \{i\}) \equiv \mu(\{\mathscr{C} \cup i\}) - \mu(\mathscr{C}) \tag{2.1}$$

2.2.3 Fairness Axioms (Efficiency, Symmetry, Dummy player)

Given a super-additive game with characteristic function/measure μ, we look for an allocation of values v_i such that

- **Efficiency** – It must hold
$$\sum_{i=1}^{n} v_i = \mu(N)$$
- **Symmetry** - If the two players i and j are *interchangeable*, i.e. if $\mu(S \cup i) = \mu(S \cup j)$ for every set not containing neither i nor j, it holds
$$v_i = v_j$$
- **Dummy player** – i is a dummy player if the amount that he contributes to any coalition is exactly the amount that i is able to achieve alone: if i is a dummy player he has to get a payment equal to exactly the amount that he would achieve on its own
$$v_i = \mu(i)$$
In other words, there is no synergy between the player i and the other players.

2.2.3.1 Linearity

Consider two different coalitional games, defined by two different characteristic functions μ_1 and μ_2, involving the same set $\mathscr{N}$ of agents.

- **Additivity** – If we re-model the setting as a single game in which each coalition S achieves a payoff of $\mu_1(S) + \mu_2(S)$, the agents' payments in each coalition should be the sum of the payments they would have achieved for that coalition under the two separate games.

$$v_i(N, \mu_1 + \mu_2) = v_i(N, \mu_1) + v_i(N, \mu_2)$$

where the game $(N, \mu_1 + \mu_2)$ is defined by

$$(\mu_1 + \mu_2)(S) = (\mu_1)(S) + (\mu_2)(S)$$

for every coalition S

2.3 Power indexes

In this section, we discuss why power indexes are so important, why do we need them. We study some power indexes that have been invented over time, formally, we expose the two well-known power indexes used in the thesis.

2.3.1 Why Power indexes

More than two decades ago, power indexes have been massively applied to political institutions [89], for instance, weighted voting has become popular. However, a widely shared intuition was that voting weights are not a good proxy for the impact the various decision-makers have on the outcome. Even if, weighted voting has been applied to the election of the president of the United States [81], (1954), the United Nation's Security Council [19, 81], (1954, 1971), and in the board of Governors of International Monetary Fund [26, 78] , (1980, 1982), it is the Council of Ministers of European Union and its status in the process of decision making that brought weighted voting on the more general research agenda. Nowadays, if we talk about cost-sharing and game-theoretic, Shapley value is proposed as a predominant solution (see [90, 92]). Moreover, in the application of coalitional games, most the time, the players are persons or groups of persons, for instance, nations, labour unions, Companies, towns, nations, etc. However, in some game-theoretic models of economic problems the players may not be persons. They may be factors of production, objectives of an economic project, or some other economic variables of the situation of interest. Power indexes have been applied in the literature on feature selection in few cases (see [17, 18]).

Power indexes can be divided into two main groups: the well-known power indexes which are our focus in this thesis:

- the Shapley-Shubik index [82] (1954), introduced by Lloyd Shapley and Martin Shubik in the 1950s based on the Shapley value introduced by Shapley [80] in 1953.
- the Banzhaf index [10], introduced by John F. Banzhaf in 1965

and the less well-known power indexes

- the Johnston index [48] published in 1978 by R.J. Johnston, the idea behind Johnston index is quite similar to the one standardized by Banzhaf index. The difference between the two indexes is the way they determine if a player is crucial in a group.

- Deegan-Packel index (1978) proposed by Deegan and Packel [49]. This power index account only the coalitions in which each agent is critical, while Johnston includes the coalitions in which at least one agent is critical. Both indexes divide the unitary power among the coalitions considered; then the power assigned to each coalition is equally shared among its critical agents.

- the Holler index or public good index (1982), first proposed in [45]. It introduces the Public Good index, stating that the worth of a coalition is a public good. with this consideration, the members of the winning decisive coalitions, (groups in which all the agents are critical), have to enjoy the same relevance; so, the power of an agent is in proportion to the number of winning decisive sets he belongs to.

- the Coleman index [19] (1971), proposed in the early 1970s by James S. Coleman. He defined three different power indexes, one for voting a collectivity as a whole and two others for individual voters.

2.4 The Shapley Value

Shapley Value, since its introduction in 1953, has generated a wide literature [65], where it was alternatively interpreted as a solution of the fair division problem [80], as a power index [81], as a centrality measure [85] or as a transform endowed of desirable properties within the Dempster-Shafer evidence theory [86] (a.k.a. theory of belief functions).

Non-additive measures play a key role in several fields [36], for instance, in Game Theory, $\mathscr{R}$ can take the meaning of the set of players, while $2^{\mathscr{R}}$ is the family of all coalitions and a measure μ of a coalition $\mathscr{C}$ represents the worth of the coalition $\mathscr{C}$. Any $\mu : 2^{\mathscr{R}} \to \mathbb{R}$, with $\mu(\emptyset) = 0$, defines a distinct coalitional game.

The problem of quantifying the contribution of individuals to the achievements of an ensemble – in a way satisfying some basic requirements – has been faced within Coalitional Game Theory, as a part of the problem of assigning a fair share of the surplus produced by coalitions (the so-called *fair division problem*), or as a part of the problem of quantifying the power of individuals in coalitions (the so-called *power assessment problem*): the value associated to a player in the former context is interpreted as the fair share to which that actor is entitled, in the latter context, it is interpreted as the power of an actor in determining the achievements of the coalition.

A solution to the fair-division problem, fulfilling a basic set of axioms, has been formulated in 1953 by Lloyd Shapley [80] and then used also as a power index in voting games, the Shapley-Shubik index [81]. The problem it solves can be formulated as follows.

An appropriate definition for the importance of a rule should fulfil a set of desirable properties. One such set of properties or axioms is the following: the whole value of the grand coalition is redistributed among players, without losses (*efficiency* axiom); players who bring the same contribution receive the same share (*symmetry* axiom); those who bring nothing (no added values) receive no shares of the surplus (*dummy player* axiom); the more a player brings, the more he/she gets (*strong monotonicity* axiom). The latter axiom can be stated as follows: suppose that there are two games, played by the same set of players, and that a player in the second game brings to each coalition a marginal contribution greater or equal to the contribution brought to the first, then the player should receive in the second game an amount greater or equal than the one she gets in the first game. Lloyd Shapley demonstrated in [80] that there is a unique value which fulfils those desirable properties and provided the corresponding expression, a specific weighted average over all the possible coalitions, later called Shapley Value.

Given a set $\mathscr{R}$ and a measure μ (i.e. given a game $\left(\mathscr{R},\mu\right)$), the Shapley Value is defined as an array of payoffs: a payoff $Sha[\mu]_i$ for each actor $i \in \mathscr{R}$. The value of $Sha[\mu]_i$ is obtained by computing a weighted average of the $\Delta_i^{[\mu]}(\mathscr{C})$ over all the coalitions $\mathscr{C} \in 2^{\mathscr{R}}$; the combinatorial weight $w(\mathscr{C})$, depends on the game size $R \equiv |\mathscr{R}|$ and on the coalition size $C \equiv |\mathscr{C}|$

$$Sha[\mu]_i = \sum_{\mathscr{C} \in 2^{\mathscr{R}}} w(R,C)\Delta_i^{[\mu]}(\mathscr{C})$$

with

$$w(R,C) = \left(\frac{R!}{(R-C)!(C-1)!}\right)^{-1}$$

The same combinatorial weight can also be obtained by means of a procedure based on the computation of the added values of players forming a permutation. Suppose, that all the players are arranged in some order, all orderings being equally likely: each ordering of the actors corresponds to a marginal value sequence, achieved by the members; for each order, one has to take note of the marginal value introduced by the actor i; then the Shapley Value for the actor i, with respect to the measure μ, is computed as the average of its added values over all permutations

$$Sha[\mu]_i = \frac{1}{R!}\sum_{\pi} \Delta_i^{[\mu]}(\pi) \tag{2.2}$$

where π runs over the set Π of all the $R!$ permutations of R objects and $\Delta_i^{[\mu]}(\pi)$ is the marginal value of the player i with respect to the coalition made by the players that preceded him in the permutation.

By analogy to its interpretation as a power index, it has been used to assess the importance of components in a composite entity (a system or a process); among the recent examples, we can mention its application in algorithm portfolio selection [30], tag sense disambiguation [54], neural network pruning [88]. It has also been used in feature selection [7, 18] and model interpretation [57, 60].

Theorem 1 (Shapley Theorem)**.** *There is a unique efficient payoff division (of the full payoff of the grand coalition) that satisfies the Symmetry, Dummy player and Additivity axioms:*

$$v_i^{Shapley} = \frac{1}{N!}\sum_{\sigma}\Big(\ \mu(\{C \cup i\}) - \mu(C)\ \Big)$$

where the index σ runs over all the $N!$ permutations of the N elements of $\mathcal{N}$.

An alternative expression is the following.

$$v_i^{Shapley} = \sum_{C \subseteq N} \frac{c!(N-c-1)!}{N!}\Big(\ \mu(\{C \cup i\}) - \mu(C)\ \Big) \tag{2.3}$$

Where the C's are the coalitions of $\mathcal{N}$ (the subsets of $\mathcal{N}$, i.e. $C \in 2^{\mathcal{N}}$) and $c = |C|$.

It is important to introduce here a remark on the Additivity axiom. In the words of Airiau [2], the additivity axiom, or ADD, is harder to motivate in some cases. If the valuation function of a TU game is interpreted as an expected payoff, then ADD is desirable (as you want to be able to add the value of different states of the world). Also, if we consider cost-sharing games and that a TU game corresponds to sharing the cost of one service, then ADD is desirable as the cost for a joint-service should be the sum of the cost of the separate services. However, if we do not make any assumptions about the games $(\mathcal{N}, \mu_1)$ and $(\mathcal{N}, \mu_2)$, the axiom implies that there is no interaction between the two games. In addition, the game $(\mathcal{N}, \mu_1 + \mu_2)$ may induce a behaviour that may be unrelated to the behaviour induced by either $(\mathcal{N}, \mu_1)$ or $(\mathcal{N}, \mu_2)$, and in this case ADD can be questioned.

For this reason, several other equivalent axioms have been considered. A parsimonious set of axioms is due to Myerson [66] and it is based on a concept called balancing of contributions. Myerson considers the N restrictions of the original game μ, each defined by a single player leaving the game: a definition of value applied to each of those games has the balancing property if for every pair of players the amount that each player wins or loses if the other leaves the game is the same. Let us denote by $\mu \setminus i$ the game $(\mathcal{N} \setminus i, \mu_{\setminus i})$, where $\mu_{\setminus i}$ is the restriction of μ to $\mathcal{N} \setminus i$.

Definition 32 (Balanced contribution axiom)**.** *A value function v satisfies the balanced contribution axiom if for all $(i,j) \in \mathcal{N}^2$*

$$v_i(\mu) - v_i(\mu \setminus j) = v_j(\mu) - v_j(\mu \setminus i)$$

The new characterization of the Shapley Value due to Myerson is the following.

Theorem 2 (Shapley Value Theorem 2)**.** *The Shapley value is the unique value function that satisfies the balanced contribution axiom.*

2.4.1 Shapley Value Computation

Methods for finding the Shapley Value can be divided into two types. The first type computes the exact value when the second provides an approximation value. The approach and the computing requirements vary with the method. Notice that none of the methods is universally ideal. Due to the number of features of our use case, we are interested in approximation methods. Here we will list some methods classified as exact and then we will see methods that have been proposed to approximate the Shapley value; particularly the method used in this thesis.

2.4.1.1 Exact methods

Four main methods are classified as exact:

- **Direct enumeration**: this method uses equation 2.3 to compute the Shapley value for the player i.

  Given a set ($N = \{1, 2, ...n\}$) of players the number of subset of players is 2^n. So evaluating the Shapley value for player i has time complexity $O(2^n)$. This method has the disadvantage that time complexity is exponential.

- **Generating functions** [62]: this method finds the exact Shapley value based on the coefficients of a polynomial generated by a function (generating functions are more detailed in [3]). The advantage of this method is that it has polynomial time complexity.

- **Conitzer and Sanhom's method** [20]: This method is used only if the characteristic function is represented in a specific form and it does not scale well.

- **Ieong and Shohem's method** [46]: this proposed method assume that the value of Shapley of a component for a given coalitional game is given by an oracle. Based on this assumption, it aggregates these values to compute the value of the overall game. This method has the advantage that his time complexity is polynomial, but to be used, the coalitional game should be represented as "marginal contribution net".

Hence all these methods compute the exact Shapley value, but they also have important disadvantages. To overcome these issues, some approximation methods were developed.

2.4.1.2 Approximation methods

The main methods proposed to approximate the Shapley value are the following:

- **The Multi-linear extension (MLE) method** [37] was proposed by Owen, his advantage is his linear time complexity. The interesting consequence of this result is the fact that it is much easier to approximate the Banzhaf value. It requires evaluation of the partial derivatives at just a single point. The Shapley value, for example, requires evaluation along the entire diagonal, plus an integration.

- **The modified MLE method** [62] is the extension of the MLE method. In order to improve the error of approximation, the modified MLE method trades-off computational time. More specifically, the modified MLE method merges the essential features of direct enumeration and MLE to improve the accuracy of the MLE method.

- **The Random permutation method** [101] was proposed by Zlokin, in this random permutation mechanism, the players form the full coalition, one player after another, choosing a random permutation. The utility of a player is equal to its contribution to the coalition at the time of joining it. The requirement of this method is that the players should agree on the all-or-nothing deal. The method' time complexity is linear.

- Finally, **The Monte Carlo simulation method**: is the earliest approximation method, the one used in the thesis, it was proposed by Mann and Shapley itself. Here we just mention that the method is based on Monte Carlo simulation, and its estimates the Shapley value from a random sample of coalitions. Then we develop it more in detail in the next section.

2.4.2 Statistical estimate of the Shapley Value

The main drawback of computation of the Shapley value consists of its exponential complexity. This was pointed out since the publication of the work by Shapley, and approximate computation methods were soon devised. A review of the various methods used to compute the exact or the approximate Shapley Value can be found in [8, 29, 61]. Among the simplest approximate methods is the one proposed by Mann and Shapley [62], based on random sampling from the space of permutations. Since, for our purposes, it is not necessary to compute the exact value of the Shapley index, but rather an approximated value which allows us to find a ranking of the rules, we can resort to one such sampling-based computations, whose time complexity is polynomial.

The rationale behind the use of this method is the following: the Shapley value is defined as an expected value: it is the sum of terms consisting of the product of the marginal value of a coalition times its probability (within the permutation-based process described above). Such a probability can be modelled as a Bernoulli parameter and thus estimated by sampling and taking the appropriate count ratios.

Consider the set Π of the $R!$ permutations of the R elements in the set $\mathscr{R}$. Consider a subset $\Theta \subset \Pi$ of permutations, obtained by sampling Π uniformly at random, and let $\theta = |\Theta|$ the cardinality of the subset. We can compute an estimate of the Shapley value as follows:

Monte Carlo Estimate of the Shapley value.

- Consider the sample Θ consisting of $\theta \ll R!$ permutations

- For each permutation $\pi \in \Theta$ compute the marginal contribution of each element i, with respect tothe measure μ, using the expression

$$\begin{aligned} \Delta_i^{[\mu]}(\pi) &\equiv \mu\left(\, \pi(1),\ldots,\pi(i-1),\pi(i)\,\right) \\ &- \mu\left(\, \pi(1),\ldots,\pi(i-1)\,\right) \end{aligned}$$

  where $\pi(i)$ is the player at the i-th position of permutation π.

- Average over the sample Θ of permutations to obtain the estimate $\widehat{Sha}[\mu]_i$ of the Shapley Value of each element

$$\widehat{Sha}[\mu]_i = \frac{1}{\theta!}\sum_{\pi}\Delta_i^{[\mu]}(\pi)$$

The quantity thus obtained is an unbiased estimator whose error decreases with $1/\sqrt{\theta}$. The complexity of this algorithm (for estimating the Shapley Value of all the R elements of a set) is $O(R\theta)$, times the complexity of the computation of the measure μ.

2.5 The Banzhaf Value

It is normal to meet in our everyday life situations of decision-making where individuals or groups of individuals must make collective decisions. In parliaments, for example, legislators collectively decide whether a bill is accepted or not. It may not have specific rules for decision-making if the electorate or decision-making body is informal. However, as a rule in the formal decision-making bodies, there are particular rules for decision-making (in the case of parliaments for the acceptance of laws). As a decision rule, the quota or voting threshold determines the number of votes be in favour of a proposal that must be guaranteed acceptance. It exits a multiplicity of majority rules according to the importance of a proposal of laws in the national parliaments. Three well-known majority rules are the simple majority (more than half of the deputies), the qualified majority of the 2/3, and more rarely one finds for important decisions, the unanimity (the whole assembly).

2.5.1 Definitions

Definition 33 (voting body)**.** *A voting body is formally represented as a set N, which contains every member in the voting body. The set $\mathscr{W}$ includes all winning subsets (i.e., that can ensure acceptance of the proposal), some restrictions are applied to $\mathscr{W}$, (see [89])*
(i) $\emptyset \notin \mathscr{W}$
(ii) $N \in \mathscr{W}$
(iii) if $S,T \subset N$ with $S \in \mathscr{W}$ and $S \subset T$ then $T \in \mathscr{W}$

A simple game, introduced by Von Neuman and Morgenstern [95] in 1944 is a n-person game which can be define as a pair $(N,\mathscr{W})$ satisfying condition (i)- (iii)

Definition 34 (Weighted voting games)**.** *Weighted voting games are special classes of simple game where each voter might not have the same amount of votes, for instance, party groups into a parliament. A specified quota of votes is necessary to approve. We use the symbol* $[q : s_1, s_2, ..., s_n]$ *to represent a weighted voting game where the numbers* s_i *are the voting "weights" of the n players and* q *is the quota need to win. In mathematical terms,* $S \in \mathscr{W} \equiv \sum_{i \in \mathscr{W}} s_i \geq q$

The idea behind voting power is that the weight of a voter is not a good measure of power instead, voting power analyses what a voter can do with his weight.

Let just take a simple real-life example of voting power concerning the first Council of EU (1958-1973). The Council consist of six members states, France, Germany, Italy, Belgium, the Netherlands and Luxembourg. These countries had the following voting weights regarding proposals which could be decided upon using a qualified majority $[12 : 4, 4, 4, 2, 2, 1]$ it is easy to conclude slipping into a fallacy that the big three (4) had four times as much as the last (1) or twice as much voting power as Belgium or Netherlands (2). Voting power theory analyses what the countries can do with their votes. Three big countries or two big countries plus two of the middle size ones can form a winning coalition and Luxembourg votes does not make any difference in any winning coalition so he has no voting power at all, and we cannot conclude that the ratio between the middle size countries and the three big ones is 1:2.

Definition 35 (Concept of Swing)**.** *We can analyze all possible voter combinations of i's voter to measure his voting power. using a characteristic function* v*, we say that a voter* i *has a negative swing in a coalition* C *if* $v(C) = 1$ *but* $v(C \setminus \{i\}) = 0$*, i.e. without his support, i can turn (swing) a winning coalition into a losing one. Positive swing is when* $v(C) = 0$*, but* $v(C + \{i\}) = 1$*, i.e., taking into account her support i can turn a losing coalition into a winning one.*

Banzhaf Index [10] was introduced after the Shapley index by lawyer John F. Banzhaf (1965). The index calculates voter i's swings like shapley. While Shapley-Shubik analyses all the possible voter permutations, Banzhaf index considers each distinct coalition. So it concentrates on voter combinations.

The absolute Banzhaf index value is the sum of voter i's swings divided by the number of subsets (2^{n-1}).

$$\beta_i = \frac{1}{2^{n-1}} \sum_{S \subseteq N} [v(S) - v(S \setminus \{i\})]$$

2.6 Summary

In this chapter, we provided definitions and conceptual tools for reasoning on coalitional games and power indexes. After mentioning the importance of power indexes, we decline the most used and well-known power indexes in coalitional games. We also define some less well-known power indexes. We analyse in Section 2.1 the two main categories of cooperative games (Transferable

Utility and Non-Transferable Utility games) and their key concepts, providing also some properties issues. Then we focus on Shapley Value, one of the two power indexes of our interest going through definitions, properties, some examples and computation methods.

In the end, Shapley Value, since its introduction in 1953, has generated a wide literature [65], where it was alternatively interpreted as a solution of the fair division problem [80], as a power index [81], as a centrality measure [85] or as a transform endowed of desirable properties within the Dempster-Shafer evidence theory [86] (also known as theory of belief functions).

By analogy to its interpretation as a power index, it has been used to assess the importance of components in a composite entity (a system or a process); among the recent examples, we can mention its application in algorithm portfolio selection [30], tag sense disambiguation [54], neural network pruning [88]. It has also been used in feature selection [7, 18] and model interpretation [57, 60]. In Section 2.5, we define the second power index, the Banzhaf index with an example and see the formula. All this will help on one hand to apply the Shapley Value in Chapter 4, and on the other hand to develop more the Banzhaf index in Chapter 5.

CHAPTER 3

Feature Selection and Machine Learning Techniques

In the two last decades, the dimensionality of datasets involved in data mining and machine learning process has increased enormously. Data has become ubiquitous in many domains, such as e-commerce, social media, bioinformatics, health care, transportation, etc. When applying data mining and machine learning algorithms on high-dimensional data, a critical issue is known as the curse of dimensionality [42]. Bellman coined this term in 1961; it refers to the problems associated with the multivariate data analysis when the dimensionality increases. Moreover, data of high dimensionality significantly increases the memory storage requirements and computational costs for data analytics. Therefore, dimensionality reduction has become a necessary step to make the analysis more manageable and to extract useful knowledge. The thesis deal with a rule selection in credit card fraud data, we re-frame the problem in terms of Feature Selection (FS) task for a classifier. For this reason, we study in this chapter the dimensionality reduction or feature selection in the first part and some varieties of machine learning techniques in the second part. We compare our method with some standard feature selection algorithms from machine learning; the greedy methods.

3.1 Feature Selection

3.1.1 Dimensionality reduction

Feature extraction and feature selection are the two approaches available to reduce dimensionality, and they are capable of improving learning performance, lowering computational complexity, building better generalizable models, and decreasing required storage. Feature selection selects a meaningful subset of the original features, hence discarding those irrelevant and redundant ones [38]. Some well-known feature selection techniques include Greedy methods (Greedy forward

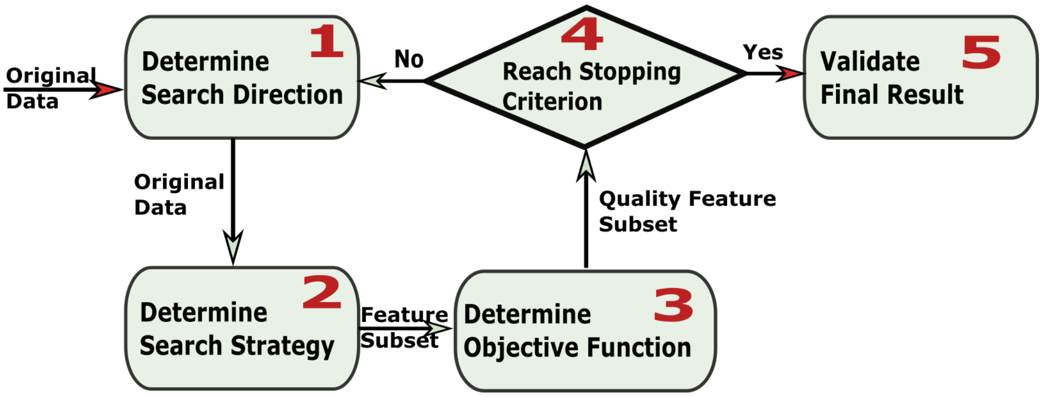

Figure 3.1: Steps of feature selection

selection and greedy backward elimination) as refers by R. Caruana and D.Freitag in [16]. Moreover also by George H John et al. in [47]. We will refer to these methods as a benchmark in the first part. On the other hand, feature extraction consists of mapping the original feature space to a new space with lower dimensions [96]. The constructed feature space is usually a linear (or nonlinear) combination of the original feature space. Some examples of feature extraction methods include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Canonical Correlation Analysis (CCA). Feature selection, on the other hand, selects a meaningful subset of the original features, hence discarding those irrelevant and redundant ones [38]. There are other techniques such as Lasso - *Least Absolute Shrinkage and Selection Operator*, it was first formulated by Robert Tibshirani [77] in 1996, Information Gain (2012) as refers in [6], and Laplacian Score (2005) as refers by He et al. in [43]. Both the feature extraction and feature selection have the advantages of improving learning performance, increasing computational efficiency, decreasing memory storage requirements, and building more general models. Since feature extraction builds a set of new features, further analysis is problematic as we cannot get the physical meaning of these features in the transformed space. In contrast, by keeping some original features, feature selection maintains physical meanings of the original features and gives models better readability and interpretability. Therefore, feature selection is known to be very effective in the context of high dimensional classification problems, enabling to improve predictive performance as well as to obtain faster and more cost-effective predictors.

3.1.2 General categories of feature selection techniques

Feature selection methods are generally classified into three categories: supervised selection [87, 97], unsupervised [27, 63], and semi-supervised selection [99]. The selection process of the critical or essential features among all the features can be divided into five main steps described in the following, see Figure 3.1 The decision made at each step is necessary for the performance of the feature selection.

- **Step 1:** Determine the search direction

  The first step determines the starting point and the search direction. If the search process starts with an empty set, then it is followed by iteratively adding new features into the collection, then we have forward search strategy. If on the contrary, the search process starts with the full set, then the features are iteratively eliminated from the collection, the strategy refers to the backward elimination process. The alternative is to begin by simultaneously adding and removing features in each iteration in this case; we have bi-directional search.

- **Step 2:** Determine the search strategy

  Given N the cardinality of the original feature set and M the cardinality of the target feature subset (when it is known) with $M < N$, an exhaustive evaluation of feature subsets involve $\binom{N}{M}$ combinations for a fixed value of M and 2^N combinations if M must be optimized as well. We know that this number of combinations is unfeasible for significant or even moderated values of M and N. Therefore a search strategy must be used to direct the process as it explores the space of the possible combination of features. A good search strategy should provide a rapid convergence to a possible optimal solution, excellent computational efficiency, and good search capability [31]. Search strategies are generally grouped into three types (Exponential search or complete search [70], Randomized search [25, 33, 34, 84], and sequential search or greedy hill-climbing search), we will develop the latest one in the following section. It is the one we use for the comparison. Its complexity is polynomial w.r.t the number of features; moreover, it is sensitive to feature iteration [31]. The common sequential strategies are Sequential Forward Search also called Greedy Forward Search (GFS) and Sequential backward Search a.k.a Greedy Backward Elimination (GBE).

- **Step 3:** Determine the objective function

  The objective function intends to evaluate candidate subsets and returns an evaluation of their "goodness," an indication used by the search strategy to select new candidates. Originally, there are four types of objective functions in feature selection: filter, wrapper, embedded and hybrid [55]. They are developed respectively in Sections 3.1.3.2, 3.1.3.3, 3.1.3.4, and 3.1.3.5.

- **Step4:** Define stopping criteria

  This step defines when the feature selection process should stop. A good stopping criterion typically leads to an efficient process, for example by avoiding over-fitting (the modelling error observed when a function is too closely fit a limited set of data). By doing so it produces a suitable feature subset with normal computational complexity. Some standard stopping criteria are number of iterations, number of features.

- **Step 5:** Validate the result

  There exist many error estimation methods for the evaluation of the effectiveness of a feature set in a context of classification or prediction. The most common is cross validation and confusion matrix for performance measures. Others, less common are Jaccard Index [52, 56] or Analysis of variance (ANOVA). In the cross-validation method, we have a training set to train the classifier and a testing set for the final evaluation. The evaluation is repeated on many samples so that the average error somehow approximates the expected error across all possible samples [12].

3.1.3 Search strategies for feature selection

3.1.3.1 Greedy selection methods

The Greedy methods exploit an algorithmic paradigm based on the heuristic of intuitive judgment, or merely common sense to seek answers which are hopefully close to the global optimum. In this approach, the decision is made based on currently available information without worrying about the effect of the current choice in the future. Greedy algorithms build a solution part by part, choosing the next part in such a way, that it gives an immediate benefit. This approach never reconsiders the choices taken previously. It is mainly used to solve optimization problems. The greedy method is easy to implement and quite efficient in most of the cases. When a Greedy Algorithm can solve a problem, then it is generally among the best methods to solve that problem as it is in general, more efficient than other techniques like Dynamic Programming. There are two main greedy selection methods: Greedy Forward Selection (GFS) and Greedy Backward Elimination (GBE).

3.1.3.2 Filter methods

Filter methods rely on specific characteristics of data to assess the importance of features. They are independent of any learning algorithms and they are also more efficient than wrapper methods. The lack of a specific learning algorithm to guide the feature selection phase makes the selected features not so useful for the target learning algorithms. Filter methods consist mainly of two steps (see Figure 3.2): (1) feature importance is ranked by a feature score according to some feature evaluation criteria. The feature importance evaluation process can either be univariate or multivariate. In the univariate case, each feature is ranked individually regardless of other features, while the multidimensional scheme ranks multiple features simultaneously. (2) The lowly-ranked features are filtered out, and the remaining features are kept. Some advantages of the filters: **fast execution:** generally, they involve a non-iterative computation on the dataset, so the implementation is faster than a classifier training session; **generality:** the filter results exhibit more generality since they evaluate the intrinsic properties rather than the interactions of the data. A disadvantage of filters is **the tendency of selecting large subsets:** their objective function is generally monotonic, so filters tend to output the full feature set as the optimal solution.

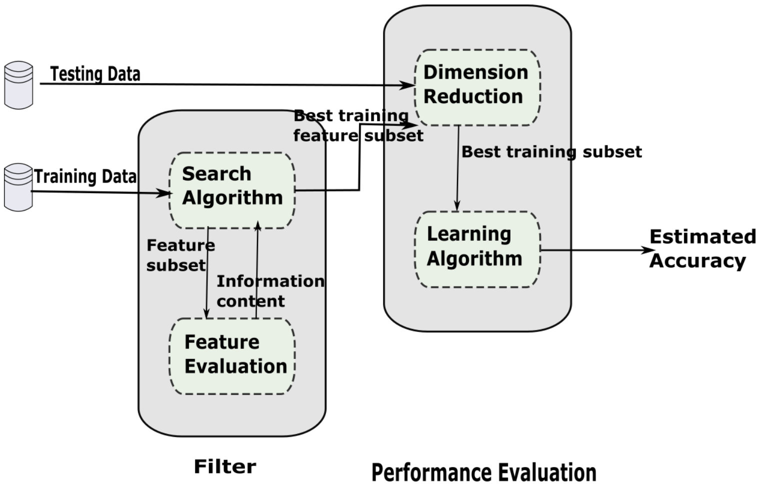

Figure 3.2: Filter Method

3.1.3.3 Wrapper methods

Wrappers are methods that rely on the predictive performance of a predefined learning algorithm to evaluate the quality of the selected features. A wrapper method, given a specific learning algorithm performs in two steps (see Figure 3.3): First, it searches for a subset of features; then, it evaluates the selected features. It repeats the two steps until some stopping criteria are satisfied or the desired performance is obtained. It can be observed on the workflow that the searching component first generates a subset of features, and then the learning algorithm acts as a black box to evaluate the quality of the features based on the learning performance. The whole process goes on iteratively until the highest learning performance is achieved. The feature subset with the most top learning performance is retained as the selected features. Unfortunately, a known issue of wrapper methods is that the search space for d features is 2^d, which makes the exhaustive search impractical when d is large. Therefore, many different search strategies such as sequential search [38], best first search [51], genetic algorithms [34] proposed to yield a local optimum learning performance. However, the search space remains huge for high-dimensional datasets. Consequently, wrapper methods are seldom used in practice. Wrapper advantages: **generalization ability:** since wrappers typically use cross-validation measure of predictive accuracy, they can avoid over-fitting; **better accuracy:** since they have specific interaction between the classifier and the dataset, they achieve better recognition rates than filters. Some drawback: **no generalization:** the solution of the wrapper is tied to the bias of the classifier used in the evaluation function. Selected features are specific to the classifier under consideration. **Execution speed:** if cross-validation is used, the wrapper must train several classifiers for each feature subset, by doing so, the method can be unfeasible for some intensive techniques.

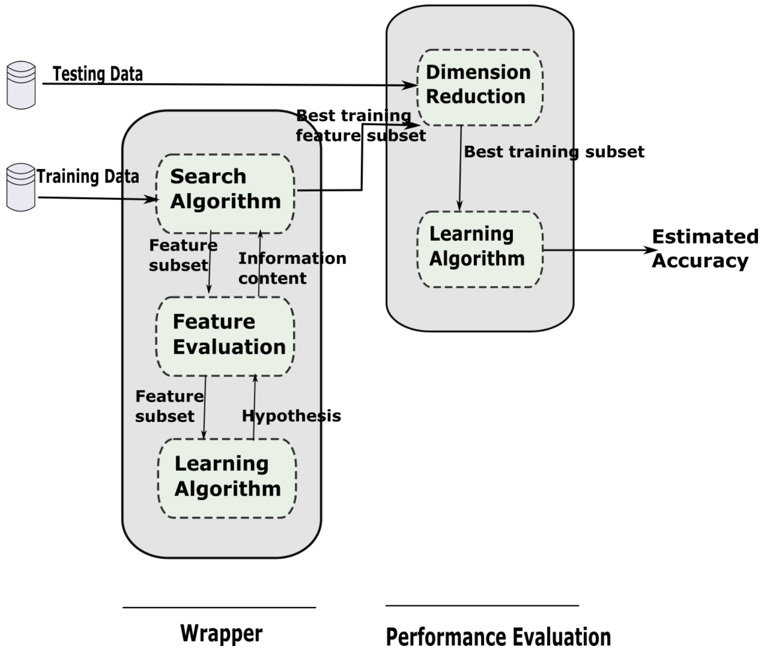

Figure 3.3: Wrapper Method

3.1.3.4 Embedded methods

Even if filter methods are computationally efficient, (they select features that are independent of any learning algorithms), they fail to consider the bias of the learning algorithms, and the selected features may not be optimal for specific learning tasks. Wrapper methods, on the contrary, evaluate the importance of features for a given learning algorithm and can obtain better predictive accuracy. Due to the exponential search space, however, it is computationally intractable when the feature dimension is very high. Embedded methods are a sort of trade-off solution between filter and wrapper methods. They embed the feature selection with model learning (see Figure 3.4). Thus they inherit the merits of both wrapper and filter methods: first by including the interactions with the learning algorithm; and then, they are also more efficient than the wrapper methods since they do not need the iterative feature evaluation. The most well-known used embedded methods are the regularization models (LASSO (Least Absolute Shrinkage and Selection Operator) [77] or Ridge Regression [44]) which target to fit a learning model by minimizing the fitting errors and forcing the feature coefficients to be less or equal than zero. Afterwards, both the regularization model and the selected feature sets represent the results.

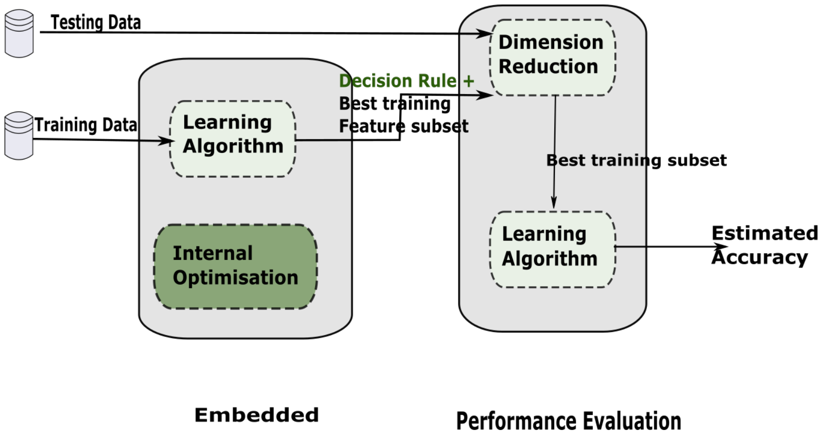

Figure 3.4: Embedded Method

3.1.3.5 Hybrid methods

Hybrid and the ensemble methods can represent the latest developments in feature selection methods. The hybrid method is formed by combining two different methods of the same criterion such as filter and wrapper, or two feature selection approaches to inherit the advantages of both methods (see Figure 3.5). In doing so, the hybrid method combines the complementary strengths of both methods [64] and try to improve the efficiency and prediction performance by using different evaluation criteria. A popular hybrid method is the combination of filter and wrapper method [72].

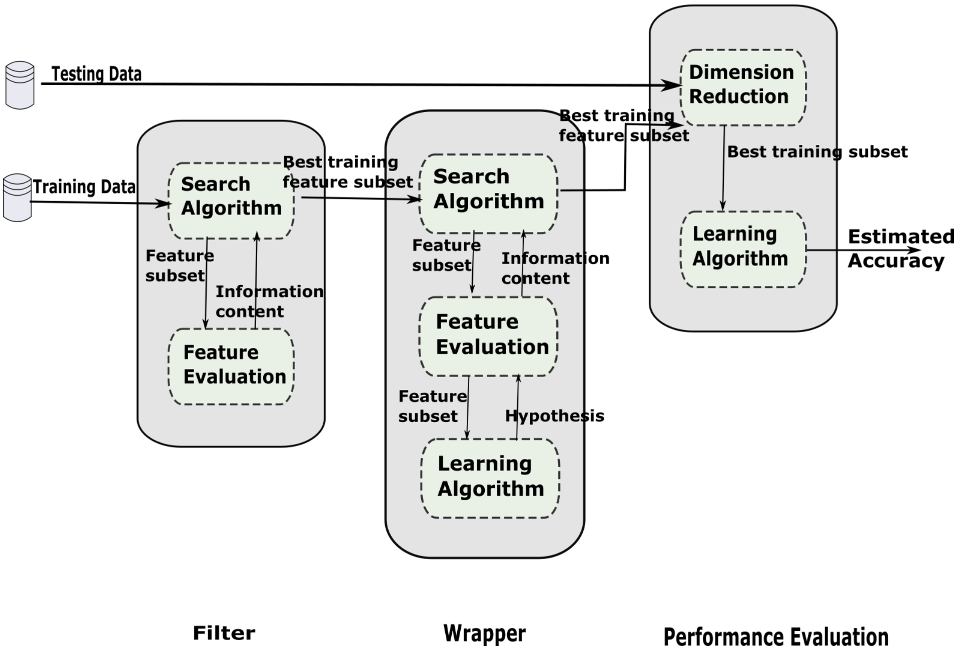

Figure 3.5: Hybrid Method

3.1.3.6 Ensemble methods

Ensemble methods are methods that aim to produce an aggregated result constructed from a group of feature subsets [83]. The purpose of this method is to tackle perturbation and the instability issues in many feature selection algorithms. Hence the method is based on different sub-sampling strategies where a particular feature selection method is run on several sub-samples, and the output features are merged to form a more stable subset see Figure 3.6. It is more flexible and robust when dealing with high dimensional data giving that the performance of feature selection no longer depends on a single selected subset. Moreover, ensemble methods provide a better approximation to the optimal subset or ranking of features by aggregating the outputs of several feature selectors. Further details on the ensemble feature selection can be found in [5].

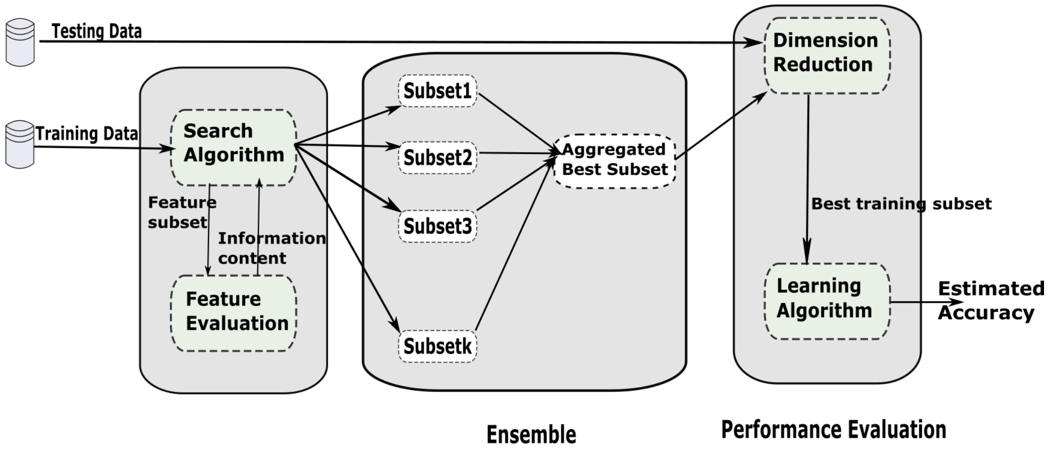

Figure 3.6: Ensemble Method

3.2 Some Machine Learning techniques

Machine learning is, at a high-level the study of teaching a computer program or algorithm how to improve upon a set of a given task. While on the theoretical side, machine learning can be viewed through the lens of the mathematical modelling of how the process works. In practical terms, it is the study of how to build applications that exhibit an iterative improvement. In this chapter, we expose the different types of machine learning algorithms before seeing in detail the Semi-supervised machine learning (SSL). Machine Learning (ML) was born from pattern recognition and the idea that computers can learn (without being programmed) to perform specific tasks. As models are exposed to new data, the iterative behaviour of ML is essential because they can independently adapt, earning from previous computations to produce repeatable, reliable decisions and results. The interest in ML is due to some factors like availability of a massive volume of data, more powerful and cheaper computational processing added to and affordable data storage. Supervised learning and unsupervised learning are two of the most widely adopted machine learning methods, but there is also the Semi-Supervised Learning method (SSL) which is the main object of our concern in the second part of this thesis.

3.2.1 Supervised Learning

The most popular paradigm for machine learning is Supervised Learning; It is easy to understand: we feed a learning algorithm with a given data in the form of example-label pairs. So the learning algorithm receives a set of inputs with their corresponding correct outputs. Through regression or classification techniques, the algorithm will learn and find the exact nature of the relationship between examples and their labels when fully trained. The supervised learning algorithm will then be able to observe a new, never-before-seen example and predict the excellent label. It is highly focused on a single task, feeding more cases to the algorithm until it can correctly perform on that task; for this reason, it often characterized as task-oriented. This learning type is mostly encountered for the reason that it is exhibited in many standard applications for instance (Advertisement popularity, spam classification or face recognition).

3.2.1.1 Supervised feature selection

For classification or regression problems, supervised feature selection is generally used — this for selecting a subset of features that are discriminative w.r.t the classes.- The feature relevance is usually assessed due to the existence of class labels, via its correlation with class labels. The training phase of the classification depends on feature selection. After splitting the data into training and testing sets, classifiers are trained based on a subset of features selected by supervised feature selection. The feature selection phase can be independent of learning algorithms (filter methods), or it may take advantage of the learning performance of a classifier to assess the selected features (wrapper methods). Finally, the trained classifier predicts class labels of samples in the test set on the selected features.

3.2.2 Unsupervised Learning

Unsupervised learning is quite the opposite of the supervised learning; in this case, the algorithm would be fed with a massive amount of data, there are no output categories based on which the algorithm can model relationships. The algorithm uses some techniques on the input data to mine for rules and try to detect patterns. This type of learning works well with transactional data; for example, identifying segments of customers who have similar attributes. Given that unsupervised learning is based on the data and its properties, we say that it is data-driven. Some areas we might see unsupervised learning are: recommender systems, buying habits.

3.2.2.1 Unsupervised feature selection

For clustering problems, unsupervised feature selection is generally used. Since acquiring labelled data is particularly expensive, unsupervised feature selection on unlabelled data has recently gained considerable attention. Unsupervised feature selection methods seek alternative criteria, due to the lack of label information to evaluate feature importance. Such alternatives criteria are data similarity and local discriminative information to define feature relevance. Unsupervised feature selection, differently from supervised feature selection, uses all instances that are available in the feature selection phase. Also, the feature selection phase is independent of the unsupervised learning algorithms (filter methods), or it relies on the learning algorithm to better select the features (wrapper methods). After the feature selection phase, it outputs the cluster structure of all data samples on the selected features by using a clustering algorithm.

3.2.3 Semi-supervised Learning (SSL)

Semi-supervised learning, as the name suggests, is somewhere between unsupervised and supervised learning. The strategies of semi-supervised learning are based on either supervised or unsupervised learning, including some additional information from the other learning paradigm. Semi-supervised learning has tremendous value, in fact, in most of the situations, labelled data are tough to collect, by contrast, in the real-world projects, unlabelled data are easier to access than labelled ones because they require less expertise, effort and time- consumption. Therefore semi-supervised learning is an extension of both supervised and unsupervised learning by including additional information.

3.2.3.1 Semi-supervised feature selection

Unsupervised feature selection algorithms do not need any label information. We know that in many real-world applications, we often have a small number of labelled samples and a large number of unlabelled samples. In this context, both supervised and unsupervised feature selection algorithms do not work very well. The small number of labelled samples in the supervised methods may be insufficient to provide correlation information of features; while unsupervised methods ignore class labels which could provide useful information to discriminate different classes. So, it could be helpful to develop semi-supervised methods by exploiting both labelled and unlabelled samples. A general framework of semi-supervised feature selection is similar to the framework of supervised feature selection except that in semi-supervised methods only partial label information is available.

3.2.4 Semi-Supervised learning problems

It comes natural asking the following question: is semi-supervised learning meaningful? In other words: can we expect a more accurate prediction compared to a supervised algorithm using only labelled data, taking into account non-labelled points? The general answer is "yes" to an essential prerequisite: the dissemination of examples, which untagged data will help to elucidate, will be relevant to the classification problem. In a mathematical formulation, one could say that the knowledge on $p(x)$ that one gain through untagged data must carry useful information into the inference of $p(y|x)$, otherwise, semi-supervised learning will not produce any improvement over supervised learning. Besides, it may even happen that the use of unlabelled data misleads the inference thus degrading the accuracy of the forecast; One should not be too surprised that for semi-supervised learning to work, certain assumptions need to be verified. Moreover, supervised learning must also be based on assumptions. Some of the most popular assumptions are:

- The smoothing hypothesis: it can be formulated as follows: If two points $x1$, $x2$ are close, it should be the same corresponding outputs $y1$, $y2$. The assumption is that the marking function is smoother in high-density regions than in the low-density areas. By transitivity, this hypothesis implies that if a path of high density connects two points, their outputs will probably be close. If, on the other hand, they are separated by a region of low density, their outputs do not necessarily have to be close.

- The cluster hypothesis: If the points belong to the same cluster, they are likely to be of the same class. Note that the cluster assumption does not imply that each class forms a single compact cluster: it merely means that we typically do not observe the objects of two distinct classes in the same cluster. The equivalence of this assumption states that: the decision limit must be in a low-density region. Moreover, this equivalence is easy to see: a decision boundary in a high-density area would cut a cluster into two different classes; many objects of different classes in the same cluster would require the decision limit to cut the cluster, that is, to go through a high-density region.

- The different assumption: it is a different but related hypothesis which forms the basis of several semi-supervised learning methods. It can be formulated as follows: High dimensionality data are (approximately) on a small variety.

Clearly, without such assumptions, it would never be possible to generalize from a finite training set to a set of infinitely unseen test cases.

3.2.5 Self-training

Self-training is probably the earliest idea about using unlabelled data in classification; it is also known as self-learning because the learning process uses its own predictions to teach itself (see Figure. 3.7). Depending on the nature of his predictor, self-training can be either inductive or transductive. Self-training can be considered as one way to address the scarcity of labelled data. The main idea is to first train the predictor f on the labelled data (L). The function f is then used to predict the labels for the unlabelled data (U). A subset S of the unlabelled data, those predicted with higher confidence, together with their predicted labels, are then selected to augment the labelled data. Regularly, S consists of the few unlabelled instances. The function f is then re-trained on the now more extensive set of labelled data, and the procedure repeats. It is possible to S being the complete unlabelled dataset, and the assigned labels on unlabelled instances can vary from iteration to iteration.

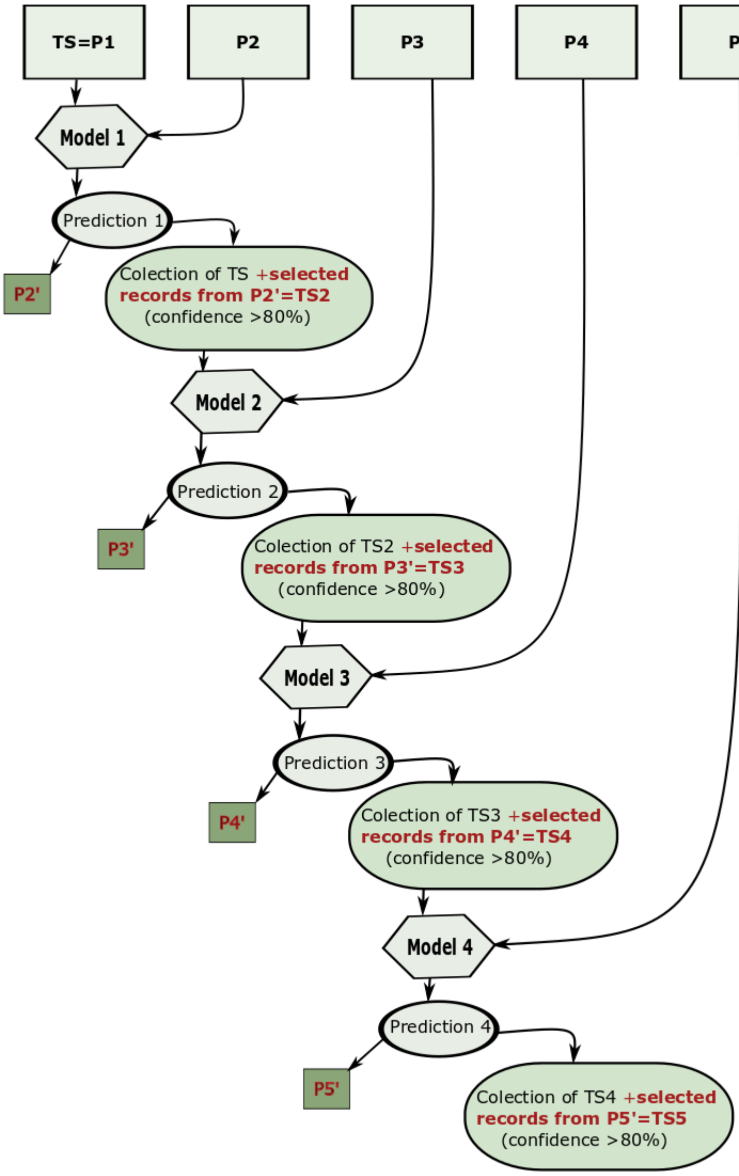
TS=P1
P2
P3
P4
P5
Model 1
Prediction 1
P2'
Colection of TS +selected records from P2'=TS2 (confidence >80%)
Model 2
Prediction 2
P3'
Colection of TS2 +selected records from P3'=TS3 (confidence >80%)
Model 3
Prediction 3
P4'
Colection of TS3 +selected records from P4'=TS4 (confidence >80%)
Model 4
Prediction 4
P5'
Colection of TS4 +selected records from P5'=TS5 (confidence >80%)

Figure 3.7: Self-training process.

The significant self-training advantages are due to his simplicity and the fact that it is a wrapper method. It means that the choice of the learner for the predictor is entirely free. The learner can be a simple SVM algorithm or another classifier. The procedure 'wraps' around the learner without changing its inner workings. It is essential for many real-world tasks. On the other hand, an early mistake made by the predictor can reinforce itself by generating incorrectly labelled data with the consequence that re-training with this data will lead to an even worse predictor in the next iteration. Some heuristics have been proposed for this problem. Semi-supervised learning does not always help; it is at least what multiple researchers have informally noted. Little is written about, except a few papers like [21]. 'Publication bias' could be the reason: given that negative results tend not to be published. So we need further study to understand when semi-supervised learning works. A well-known early example of self-training is Yarowsky's word sense disambiguation algorithm [98]. For specific learning algorithms, there are some theoretical analyses of self-training [22, 39]. However, in general, self-training is challenging to analyze. Here are some example applications of self-training [74, 75].

3.3 Summary

In this chapter, we provided the different categories and methods of feature selection, providing in the same occasion tools to better understand feature selection, which is one of the key concepts of this thesis. Like we can expect, feature selection will come back in our discussion in Chapters 4 and 5. In these cases, we will apply it to our use case dataset on fraud detection.

We also define some machine learning techniques in order to introduce self-training, this semi-supervised machine learning technique we will use in Chapter 5 to improve the classification when ones have a large dataset with just a few proportions of this dataset annotated.

Part III

Power Index and Feature Selection : Application

CHAPTER 4

Managing a pool of rules for credit card fraud detection by a Game Theory based Approach

This chapter exposes the first contribution based on real credit card fraud transactions. In general, fraud detection systems consist of an automatic system made by (*if-then-else* rules) which control any transaction and trigger an alert in the case of suspicious transaction. These automatic credit card transaction classification are divided into two phases (the real-time phase and the near-real-time phase). In the Real-Time (RT) phase the system decides quickly, based on the bare transaction information, whether to authorize the transaction; in the subsequent Near-Real-Time (NRT) phase, the system enacts a slower *ex-post* evaluation, based on a larger information context. The classification rules in the NRT phase trigger alerts on suspicious transactions, which are transferred to human investigators for final assessment. The management criteria used to select the rules, to be kept operational in the NRT pool, are traditionally based mostly on the performance of individual rules, *considered in isolation*; this approach disregards the non-additivity of the rules (aggregating rules with high individual precision does not necessarily make a high-precision pool). In this work, we propose to apply, to the rule selection for the NRT phase, an approach which assigns a normalized score to the individual rule, quantifying the rule influence on the overall performance of the pool. As a score we propose to use a power-index developed within Coalitional Game Theory, the Shapley Value (SV) introduced in Chapter 2, summarizing the performance in collaboration. Such score has two main applications: 1) it can be used, within the periodic rule assessment process, to support the decision of whether to keep or drop the rule from the pool; 2) it can be used to select the k top-ranked rules, so as to work with a more compact rule set. Using real-world credit card fraud data containing approximately 300 rules and 3×10^5 transactions records, we show that: 1) this score fares better – in granting the performance of the pool – than the one assessing the rules in isolation; 2) that the same performance of the whole pool can be achieved keeping only one tenth of the rules – the top-k SV-ranked rules. We observe that the latter application can be re-framed in terms of Feature Selection (FS) task for a classifier: we show that our approach is comparable with respect to benchmark FS algorithms, but argue that it presents an advantage for the management, consisting of the assignment of a normalized score to the individual rule. This is not the case for most FS algorithms, which only focus on yielding a high-performance feature-set solution.

4.1 Introduction

4.1.1 Context and problem

In the latest years, enterprises and financial institutions have been faced with an ever-growing presence of credit card payment fraudulent activities. Since the huge volume of online transactions, and the high dimensionality of the corresponding records, makes it unfeasible for human experts to check for anomalies, systems for online automatic fraud detection are used to contrast this phenomenon. In those systems, an automatic fraud detection engine scans all the incoming transactions for suspicious patterns. Such patterns are encoded under the form of rules. Here is an example of rule *" if a cardholder runs a transaction for a given amount in a given country and, within the next day, (s)he runs another transaction for another given amount in another given country, then trigger an action"*.

In the typical online automatic fraud detection system there are two phases: the Real-Time (RT) phase and the Near-Real-Time (NRT) phase. In the RT phase, the system decides quickly whether to block a transaction based on the bare transaction information. In the subsequent NRT phase, the system enacts a slower ex-post evaluation based on a larger information context, including linked data. The classification rules in the NRT phase can trigger an alert on suspicious transactions: those transactions are transferred to human investigators for final assessment. The investigators' task consists of 1) choosing those among the submitted transactions to investigate and 2) carrying out a rapid investigation on them (this may include a phone call or a text message to the card holder), so as to assess the alert as a legitimate transaction or a fraudulent one. If the transaction is judged fraudulent, the corresponding credit card is blocked. We focus on the management of the rules for the NRT phase.

The NRT phase rules can be created either on the basis of expert knowledge or on the basis of historical data. *Expert driven* rules are formulated by credit card fraud experts and are meant to be specific for a given fraud scenario, whereas *data driven* rules are learned from the historical data, through Machine Learning methods. The rules that are operational at a given moment in time are gathered in a *rule-pool* and run simultaneously on every incoming transaction. If at least one of the rules detects a suspicious transaction, an alert is generated (in other words, the set of rules are combined by means of the OR operator).

It is very important to point out that, although the *alert generation* process is a *classification process*, it is a peculiar one: it is not aimed at saying the final word on the transaction but is rather geared towards providing a suitable input to the subsequent investigation process. As such, it can afford unusual trade-offs: it will try to achieve a *high recall* level at the price of a *lower precision*. Indeed the investigators will complete the assessment: using human expert judgment, they will improve the precision up to very high levels. The traditional approach is the following: try to build the rule pool with the rules that individually have high precision, and aggregate them by an OR operator. The higher the number of rules, the higher, in general, the coverage of the fraud cases, i.e. the higher the recall. Of course, the volume of alerts has not to be too high, because it would exceed the processing capacity of the experts, hence rules with low precision and too high recall should be avoided.

It is also intuitive that – due to the natural shift in the fraudulent patterns – the composition of the rule pool needs to be continuously monitored and re-tuned. This pertains to the *rule management process* and is enacted by human experts that periodically take care of rule assessment, rule addition, rule removal, etc., so as to maintain or achieve the desired classification performance and efficiency. The assessment is carried on considering a rule at a time. For this reason, the number of rules should not be too high.

In short, the overall process requirements demand a good computational efficiency, agility in the management and a peculiar trade-off in the classification performance of the OR-ed rule pool, aimed at making the downstream expert assessment more effective. The equilibrium among those different requirements can be maintained only by continuous monitoring and re-tuning.

4.1.2 Contributions

In this work, we focus neither on the design nor on the discovery of new rules, nor on the computational efficiency of the individual rules, but rather on supporting the efficiency of the process of *rule management* for the NRT phase.

The current praxis establishes that the rule-managers decide to remove an existing rule or to add a new one to cover a novel fraudulent scenario based on reports about the rule precision and recall computing using recent data. It is important to notice that in this assessment process the managers consider the performance of the rule as if it were run in isolation: the fact that in correspondence to a fraudulent transaction the rule was the only one triggering an alert or that there were some other rules firing, is not taken into account.

We propose a rule evaluation and selection procedure that quantifies the performance of a rule "in collaboration" with the others, taking into account possible redundancy. This kind of performance is quantified by a score. Given the same performance, such score will be higher if the rule is non-redundant, lower if it is redundant with other rules, or sets of rules.

Notice that the contribution of the individual rule to the rule pool cannot be quantified correctly by means of the simple *added value* (a.k.a. *marginal contribution*) of the rule to the pool (defined as the difference between the performance of the pool *including* that rule and the performance of the pool *without* that rule) because the performance is *non-additive*.

The relevant performance metric we propose is inspired by Coalitional Game Theory (CGT) and interprets each rule of the pool as a player of a collaborative game: the metric quantifies the contribution of the player in attaining the goal of the OR-ed pool (measured in terms of precision, recall or F-score). Such a CGT inspired metric is the Shapley Value introduced in Chapter 2 and is defined by a suitably weighted average of the added values of a rule with respect to all the possible coalitions of rules (see Section 2.2.1) for the detailed definition.

Such score has several advantages. First, it provides the rule managers with a synthetic account of the actual usefulness of the rule in the fraud detection process. Second, it can be used to select the k top-ranked rules, so as to work with a performing and more compact rule set: a lower number of operational rules requires a lower maintenance effort.

We observe that the problem of selecting the set of rules operational at a given time – then to be OR-ed for flagging a transaction as suspicious – can be mapped into a special version of a Feature Selection (FS) problem: if we consider that the OR aggregator acts as a *fixed-by-design* classifier), each boolean rule can be considered a boolean-input feature for such classifier. To this task, one could apply the state-of-the-art feature selection algorithms. We use the Greedy Forward Selection (GFS) algorithm as a benchmark and show that our method has comparable performance: we argue that the advantage of our method is that it associates a normalized score to the individual "features", whereas, typically, FS algorithms focus on providing an optimal feature set, and do not yield normalized importance scores.

Notice, in passing, that the exact computation of the Shapley value is exponential, and that one can replace such a computation with a sampling-based estimate. We show that also the computational cost of this approach is competitive even with respect to one of Greedy Forward Selection.

We validate our approach on a real-world credit card fraud transaction dataset consisting of about 3.5×10^5 transactions provided by an industrial partner (Atos Worldline) [1] . Each record contains only a generic incremental ID for the transaction, its class (fraudulent/legitimate) and a Boolean value for each rule telling whether it had been triggered. No other attributes of the transactions were available.

4.1.3 Credit Card fraud detection

The automatic detection of frauds in credit card transactions has been subject to extensive research. The literature mostly focuses on the discovery of new rules from data (see for instance [11], or [23] and references therein). Even the data from our industrial partners were also used, in other recent works, for studies oriented to the discovery of fraudulent patterns, for instance, in the structure of transaction sequences [50], using Hidden Markov Models [59], exploiting semantic knowledge, inserted using graph embedding [100], using active learning [15]. Most of these cited work above have been done by the members of our team. Also, looking for the optimal fusion of classifiers [14]. The optimal fusion of classifiers in credit card fraud detection was also investigated by Vergara et al. in [76, 94].

The contribution of the present work does not address the problem of fraud discovery, nor the optimal fusion of the different discovered rules (in the NRT phase they are always aggregated in OR), but rather the problem of supporting the rule management with suitable summary information about individual rules. We are not aware of any research paper discussing such a pool oriented approach to the process of rule management for credit card fraud detection.

4.1.4 Feature Selection

In the present work, we argue that power indexes can be used to rank the rules and help to reduce the size of the rule pool. Selecting the rules is analogous to the problem of Feature Selection (FS) for classifiers, provided that one assimilates a rule to a feature and the OR aggregator to a classifier. Power indexes have been already used for supporting FS, in some works: the Shapley Value was used for FS by Cohen, Dror and Ruppin [17, 18]. Their works, however, differ from ours under several aspects. Their papers focus on the general task of FS, aimed at optimizing the process of learning classifiers (they experiment with SVMs, random forest, Bayesian Classifiers and so on), whereas our classifier is fixed (the OR of Boolean output rules); they do not use the basic Shapley Value computation and ranking, but rather a Greedy Backward Selection algorithm, the Contribution Selection Algorithm (a backward elimination algorithm); their algorithm works by maximizing the Shapley Value of the *accuracy*, so as to achieve high *accuracy* in the final classifier; on the contrary, we work with the Shapley Value with respect to the *precision* and, by aggregation in OR, we build a high *recall* classifier. To reduce the computational complexity, they make the apriori assumption that the size d of significant interactions between features is much smaller than the number of features, n, i.e. they use $d << n$. We do not make such an assumption. Variations around these ideas can be found in [35].

In this thesis, we compare the effect of the Shapley Value in supporting Feature Selection to other standard FS algorithms to show that its results are comparable to those algorithms in terms of performance, but stress that most FS algorithms do not provide a normalized score to the feature, which could be used for feature management.

4.1.5 Interpretation of model prediction

The Shapley Value has been used also in the context of explanation and interpretation of prediction models. The earliest work using the Shapley Value to address such problem of interpretability is the one made by Lipovetsky and Conklin in [57], who use the SV to quantify the relative importance of the predictors in linear regression and show that the approach is robust with respect to the presence of collinearity.

Lundberg and Lee in [60] address the problem of interpretability of the results of a prediction model. They consider an explanation model g to be a simple approximation of a prediction model f and focus on local additive explanation models: the local aspect corresponds to the fact that they aim at explaining a prediction $f(x)$ based on a single input x; the additivity implies that the explanation model attributes an effect to each feature and summing the effect of each feature attribution one gets the approximation of the original model; they show that the Shapley Value allows deriving a class of additive importance measures fulfilling a set of desirable requirements. The accent of the paper is on formulating explanation models: the authors define a class of explanation models which unifies six major methods (the class is named additive feature attribution methods) and they validate their work by means of a user study showing that the approach is aligned with human intuition.

An earlier work by Strumbelj and Kononenko [91] also address the problem of the explanation of prediction models. They focus on the situational importance of a feature: this is defined as the difference between what a feature contributes in average and what it contributes when it takes a specific value (the average contribution represents the contribution when the value of a feature is not known or missing). Such a concept represents a useful explanation only if the prediction model is additive. The authors find that a convenient explanation model for non-additive cases is provided by a weighted average of the situational importance with respect to all the possible sets of missing features. The solution turns out to correspond to the Shapley Value of the situational importance.

Both the works by Lundberg and Lee and by Strumbelj and Kononenko are far from ours in that they focus on local explanations (in our setting this would be equivalent to explaining the *predicted class* of a single transaction based on the individual rules): we, on the contrary focus on the average impact of the features on the overall performance (the impact on precision and recall). The work by Lundberg and Lee does not address the problem of editing the set of input features and has no explicit link with the fraud detection domain; no suggestion is made about exploiting the explanation for feature selection purposes.

4.2 The fraud detection process and its management

In this section, we describe a real-world fraud-detection system – similar to the one routinely used by the industrial partner that provided the dataset – and the corresponding rule management process.

4.2.1 The fraud detection process

A credit card transaction involves mainly two actors: a cardholder and a merchant; for each of them, third parties are responsible for managing the transactions. For instance, issuing banks and acquiring banks are respectively involved at the cardholder side and the merchant side of the transactions. The detection of fraud is performed in two cases: fraud detection at the issuing side aims at protecting the cardholder from misuse of information from his/her card and from an unauthorized payment; fraud detection at the acquiring side aims at protecting merchants from stolen and counterfeit card frauds. We focus on the issuing side, but the management strategy proposed in this research could easily be applied in fraud detection by the acquiring side. The fraud detection system is triggered by the credit card payment process: in correspondence to a card payment attempt, the credit card fraud detection process is activated twice: before the authorization for payment and after. In the first case, one speaks of *real time* (RT) fraud detection phase, in the second of *near real time* (NRT) phase. (see Figure 1.1).

By running a fraud detection process before the authorization, one has the opportunity to block a fraudulent payment before it is accepted. To allow effective use of RT fraud detection, the system must be: i) fast (RT fraud detection must typically be done in ~ 200ms) and ii) precise (a false positive implies that we refuse a legitimate transaction, thus causing an inconvenience to the customer).

Rules in an RT fraud detection engine are typically *if-then(-else)* rules designed by human investigators to block payment requests that are clearly fraud attempts. Due to the speed constraint, these rules use mainly information that is available at the time of the transaction request and do little use of cardholder profiles. The card associated with an allegedly fraudulent transaction is automatically blocked to prevent future frauds. All transactions passing the RT fraud detection phase without an alert are authorized.

The fraud detection process, however, proceeds beyond this point: all authorized transactions are further analyzed by the NRT fraud detection engine. As the transaction is already accepted, the NRT fraud detection system is allowed more time to make a decision (typically ~ 1 minute). The system can augment the transaction data with further features and match the current transaction with the previous purchases and the profile of the cardholder. The augmented transaction data can include features like the average expenditure or the average number of transactions on the same day. It is also possible to add advanced features [9] for instance obtained by graph mining [53] or extracted from Long Short-Term Memory (LSTM) models [50].

The features-based rules in the NRT context can still be simple *if-then(-else)* rules designed by human investigators, or they can be the result of data analytics/machine learning. The NRT rules are simultaneously executed, and any transaction firing at least one of these NRT rules produces an alert i.e., (they are aggregated in OR).

Each alert generated by the fraud detection system is submitted to the attention of a human investigator, an expert in fraud detection. The experts estimate with a high degree of confidence whether an alert is a *true positive* or a *false positive*.

All the feedback from the investigators is used by the system i) to automatically retrain the machine learning models, if needed, and ii) to generate management oriented reports: these reports are used by the investigators to manage the pool of rules.

4.2.2 The management process for the NRT phase rules

As mentioned in Chapter 1, Section 1.2 , there are two main types of rules, *expert driven* rules which are simple *if-then(-else)* rules and *data driven* rules, obtained by machine learning techniques applied on historical data-sets. The fraud detection rule pool can contain hundreds of rules, thus management is an important component of the fraud detection process. Based on the management oriented reports, the investigators can decide to modify/remove an existing rule or to add a new rule in production in order to cover a new fraud scenario.

Classical metrics used in assessing a rule are precision and recall and F_β-score (a weighted harmonic average of the former two). Other metrics are commonly used like the *speed of detection* (i.e. the number of fraudulent transactions before we produce an alert), but they are not considered in this paper, due to the fact that one cannot compute these metrics with the data provided by our industrial partner.

In the standard approach, these metrics are used to assess the rules individually, i.e. in isolation, independently of the performance of the other rules in the pool. Within this approach, one typically ranks rules according to their performance, measured in isolation and takes the removal/update/insertion decisions accordingly: low-rank rules are likely to be deleted, high-rank rules, if already in the pool, are preserved, and if not in the pool, are inserted.

This approach implicitly assumes that the best set of rules is obtained by individual rule optimization. As we are going to see, this is often not the case. Individual-level performance and pool-level performance are indeed correlated, but individual rule performances do not add up to the pool-level performance, because a rule can be redundant with respect to another rule or a set of other rules.

We argue that one should not require from a rule to be the best with respect to a performance metric computed assessing the rule in isolation: one should, rather, require the rule to best contribute to the rule-pool performance metric. If one can rank the rules according to their contribution to the pool performance, the elimination of lower ranked rules and the insertion of higher ranked rules will be more effective in improving the pool performance.

The fact that the rules that are “individually best” do not add up to the “best rule-pool” is intrinsically tied to a fundamental property of the system: the non-additivity of the contributions to the different performance metrics. Given a set of rules, different performance metrics may display different degrees of non-additivity; however, in general, the performance metrics used in practical assessment (precision, recall, F-score, etc.) are all non-additive.

4.3 The Shapley Value approach to the rule management problem

We discussed the approach to quantifying individual contributions based on the simple marginal contribution in section 2.2.2, hereafter we discuss the approach based on the Shapley Value, a computationally costly metrics, which, however, can be estimated accurately enough by a permutation sampling method. Among the problems faced by rule managers during their activities are those that can be epitomized in the comparison of a rule to the other rules in terms of some assessment metrics, so as to take actions and edit the rule pool. For instance, if a newly created rule – not yet operational – turns out to be – according to the chosen metric (computed on historical data) – relatively good, the rule managers can consider inserting the rule into the pool. On the other hand, monitoring the performance of the operational rules has a considerable time cost, thus the pool should not be too extended: if an operational rule turns out to be worse than other rules the managers can consider the possibility of removing it from the pool.

All those choices are based on relative comparisons among rules.

4.3.1 Traditional approach: individual performance ranking

According to the current practice, the rules are ranked according to their individual precision p, individual recall r, and individual F_β-score, the latter being defined by

$$F_\beta = (1+\beta^2)\frac{p\,r}{\beta^2 p + r}$$

as introduce in Section 1.2.3, when $\beta^2 = 1$, the same importance is attributed to precision and recall (i.e., to type I and type II errors): this metric is called F_1-score or simply F-score; by convention when $\beta^2 = 2$ or $\beta^2 = 0.5$ one talks respectively of F_2-score and $F_{0.5}$-score.

Typically, thus, the rule is run on a representative historical transaction dataset and the performance metrics are computed. After this assessment, decisions are taken about whether to use the rule or not in the pool.

Wishing to create a performing rule pool with a subset of k rules, within the traditional approach one would rank the rules based on the individual performance and then one would keep the top-k performing rules. The algorithm $1R$ for such a pool-building prescription is illustrated by the pseudo-code Algorithm 1. There, the base-metric μ represents either p, or r, or F_β. The algorithm simply computes the performance metrics of each rule (used in isolation), ranks the rules based on such performance and keeps the set of k top-performing rules.

4.3.2 Shapley Value based ranking

The approach we propose is different in the following respects. We suggest

- to assess the performance of a rule not based on its individual rule performance, but rather on its contribution to the pool performance
- to quantify such a contribution by means of the rule's Shapley value with respect to the pool performance
- to estimate such a Shapley Value by permutation sampling

Algorithm 1: Algorithm 1R to build a pool of k rules by assessing of the rules performance metric μ in isolation

```
function: 1R ($D, R, k, \mu$)
input   : D the dataset with labels
input   : R the rules
input   : k the desired size of the pool
input   : $\mu$ the reference metric

for $i \in R$ do
    $\mu[i] \leftarrow$ compute the performance metric $\mu$ of rule $i$ on $D$
                                   ▷ Computing the performance metrics of each rule
end
$S \leftarrow$ sort the rule base on their $\mu$    ▷ Ranking the rules on the base of the computed performance metrics
$T \leftarrow S[1..k]$                ▷ keeping the set of k top-performing rules
return $T$
end function
```

- to rank the current operational rules according to the estimated Shapley Value

Once this is done, one can choose one of two options. The first consists of choosing the size k of the pool by making a trade-off between the F-score of the OR-ed pool and the size k (by pruning rules with the lowest Shapley Value). Alternatively one can set a predefined threshold k and keep only the highest performance k rules. To demonstrate our approach we develop the latter alternative.

The algorithm, denoted by SV, is illustrated by the pseudo-code Algorithm 2. The algorithm estimates for each rule, by means of permutation sampling, the Shapley Value with respect to the performance metrics, ranks the rules based on their Shapley Value and keeps the set of k top-performing rules.

4.3.2.1 Forward Selection based on the Shapley Value

A subtle point to consider with respect to the Shapley Value is the following. When the "game" is played by a pool of n players/rules, a rule i is endowed with a given Shapley Value. However, if we drop from the pool the rule endowed with the least Shapley Value, the game changes: at that point, in general, also does so the Shapley Value of rule i in the new game. If this is the case, also the ranking of the rules may change: the set of the top-k Shapley Value rules in the whole n-rule pool, is not the same that one would obtain, by dropping the least-SV rules one at a time and recomputing every time the SV.

Since we are aiming at creating the best-SV team of k rules, to be conservative, we can replace the one-time estimate of the SV by a repeated re-computation of the SV and dropping of the rule with lowest SV, i.e. by a backward elimination algorithm, based on SV. Similarly, we could replace the one-time estimate of the SV by an incremental construction of the pool structured as

Algorithm 2: Algorithm SV to build the rule pool based on the Shapley Value of the rules with respect to performance metric μ

```
function : SV ($D,R,k,\mu,N$)
input    : D the dataset with labels
input    : R the rules
input    : k the desired size of the pool
input    : $\mu$ the reference metric definition
input    : N number of permutations used in the sampling

output   : $T$ list of original rules indexes, sorted by their $Sha$
output   : $Y$ $Sha$ values for the rules in T

for $l \in 1..N$ do
    $\pi \leftarrow$ a random Permutation of $R$ ▷ Permutation sampling (for random permutation generation, see algorithm 3)
    for $i \in 1..R$ do
        $total\_score\_of[i] \leftarrow 0$
    end
    $oldscore \leftarrow 0$
    $U \leftarrow \emptyset$
    for $j \in 1..R$ do
        $i \leftarrow \pi[j]$
        $U \leftarrow U \cup i$
        $newscore \leftarrow$ compute the classification score $\mu_U$ ▷ Computing classification score for each permutation
        $\Delta \leftarrow newscore - oldscore$
        $tot\_score\_of[i] \leftarrow tot\_score\_of[i] + \Delta$ ▷ Computing marginal contribution of each rule in each permutation
        $oldscore \leftarrow newscore$
    end
end
for $i \in 1..R$ do
    $Sha[i] \leftarrow tot\_score\_of[i]/N$ ▷ Computing the Shapley for each rule
end
$S \leftarrow$ sort the rules based on their $Sha$ ▷ Ranking the rules on the base of the computed performance metrics
$T \leftarrow S[1..k]$ ▷ Sorting rules indexes by their performance metrics
$Y \leftarrow Sha[T]$ ▷ Sorting performance metrics of the rules
return $T,Y$
end function
```

a forward selection process. We focus on the latter procedure. This procedure, denoted by SVFS (Shapley Value based Forward Selection), is formalized in Algorithm 4. Notice that the algorithm has the same general structure that of the Greedy Forward Selection (GFS) algorithm described below, in Algorithm 5, except that in SVFS, at each round, the chosen rule is not the one that determines the highest increase in pool performance, but rather the one that has the highest Shapley Value in the game determined by the current rule pool.

Computing the random permutation We use the Knuth shuffle algorithm [69] for generating random permutation in the range [1..n]

```
Algorithm 3: RandomPermutation ($n$)
    function : RandomPermutation($n$)
    input    : $n$ number of elements in the permutation
    output   : a random permutation

P ← [1,...,$n$]
for $i \in [1, n-1]$ do
    $j$ ← RandomInteger($i, n$)      ▷ Generation of random integer in the range i..n
    swap($P_i$, $P_j$)
end
return $P$
end function
```

Our findings, reported in the next section, show that the results obtained using SV (Algorithm 2) and those obtained using SVFS (Algorithm 4) on our dataset, yield essentially the same results, whereas they perform generally better than 1R (Algorithm 1).

4.3.3 Computational complexity and execution time

In terms of computational complexity and actual execution time, the above-described algorithms are very different. We will see that SV is not the fastest algorithm. However, execution times are less important for this kind of application, since, in practical settings, the reassessment of the rules in the pool is typically performed once or twice a month.

Let us denote by $C_\mu(k)$ the cost for evaluating the performance μ of an OR-ed k feature set. We list the algorithms in increasing order of complexity.

1R The algorithm based on the ranking by performance computed in isolation (Algorithm 1) has the lowest complexity among the considered algorithms. Its complexity $\mathscr{C}_{1R}(k,n)$ is

$$\mathscr{C}_{1R}(k,n) \propto n\,C(1) + n\log n$$

where $n\log n$ is the complexity of a single sort, typically negligible with respect to other component, therefore in first approximation the complexity is $\mathscr{C}_{1R}(k,n) \propto n\,C(1)$, i.e. in practice, it does not depend on k.

Algorithm 4: Incremental construction of the rule pool based on the Shapley value with respect to performance metric μ

```
function : SVFS $(D,R,k,\mu,N)$
input    : D the dataset with labels
input    : R the rules
input    : k the desired size of the pool
input    : $\mu$ the reference metric definition
input    : N number of permutations used in the sampling

$U \leftarrow \emptyset$                         ▷ indexes vector
$V \leftarrow R$                                 ▷ Rules vector
for $l \in 1..k$ do
    $max \leftarrow -1$
    for $i \in V$ do
        $X \leftarrow U \cup i$
        $T,Y = SV(D,X,\mu,N) \leftarrow$ Computed metric $\mu$ for $OR-ed$ pool $X$   ▷ Computing the performance metrics for X
        $t \leftarrow$ position of $i$ in $T$
        $Sha[i] = Y[t] \leftarrow$ Extracting from $Y$ the Shapley Value of $i$ in the game $U \cup i$   ▷ Extracting the Shapley of the rule i in X
        if $Sha[i] > max$ then
            $i_{max} \leftarrow i$
            $max \leftarrow Sha[i]$
    end
    $U \leftarrow U \cup i_{max}$
    $V \leftarrow V \setminus i_{max}$
end
return $U$
end function
```

GFS Greedy Forward Selection (Algorithm 4) has complexity $\mathscr{C}_{GFS}(k,n)$ given by

$$\mathscr{C}_{GFS}(k,n) \propto n\,C(1) + (n-1)C(2) + \cdots + (n-k+1)C(k)$$

If we assume that the complexities for evaluation are approximately the same, i.e.
$C(1) \approx C(2) \approx \cdots \approx C(k)$, and call that value C,
then the complexity of GFS is proportional to $\mathscr{C}_{GFS}(k,n) \propto C \times k(n+(n-k))/2$, which for $k << n$ is approximately $\mathscr{C}_{GFS}(k,n) \propto k\,n$.

SV The complexity of SV is proportional to the number of evaluations made during the processing of a permutation, times the number N of permutations, plus the cost for sorting the values, i.e.

$$\mathscr{C}_{SV}(k,n) = \mathscr{C}_{SV}(n) \propto (C(1) + C(2) + \cdots + C(n))N + n\log n$$

disregarding the latter and assuming the costs of the evaluation independent of k, we have $\mathscr{C}_{SV}(k,n) \propto n\,N$.

SVFS The complexity of SVFS results from the addition of the different calls to SV by the Forward Selection algorithm. Adopting again, for the sake of simplicity, the assumption $C(k) \approx C \; \forall k$ we have

$$\mathscr{C}_{SVFS}(k,n) \propto \mathscr{C}_{SV} \times k(n+(n-k))/2$$

i.e.

$$\mathscr{C}_{SVFS}(k,n) \propto nNk(n+(n-k))/2 \simeq Nkn^2$$

so, while SV is linear in n, SVFS is quadratic.

Exhaustive Search We also mention for reference the exhaustive search algorithm: finding the best feature-set of size k out of n features has complexity

$$\mathscr{C}_{Eshaustive} \propto \binom{n}{k} C(k) \simeq \binom{n}{k} C$$

Comparison Clearly, in typical conditions, i.e. $1 \ll k \ll n$, the ordering of the complexities is the following

$$\mathscr{C}_{1R} < \mathscr{C}_{GFS} < \mathscr{C}_{SV} < \mathscr{C}_{SVFS} \ll \mathscr{C}_{Exhaustive}$$

4.4 Validation of the method

Hereafter, first, we describe our contributions, then describe the real world data set used for the investigation, and finally report the results of the validation: we compare our rule-pool-based approach relying on the Shapley Value 1) first to the individual-rule-based approach (Section 4.4.3), then 2) to the Greedy Forward Selection feature selection approach (Section 4.4.4). The results confirm the validity of the proposed approach. We will show the data support the following two contributions:

1. The SV-based approach fares better than the traditional individual-rule-based 1R in building the OR-ed rule pool;
2. The SV-based approach is comparable to other feature selection techniques (we use the *Greedy Forward Selection for comparison)*.

The Greedy Forward Selection approach for rule pool building – which is formalized by the pseudo-code in Algorithm 5 – proceeds building the pool incrementally: the algorithm starts from the empty set, and adds to the current rule set one rule at time: each time it chooses, among the rules not yet selected, the one yielding the largest increase in pool performance (where the performance μ can be precision, recall or F-score); the algorithm stops when the target size k of the pool is reached.

Algorithm 5: Rule pool building by Greedy Forward Selection

```
function: GFS ($D$,$R$,$k$)
input   : D the dataset with labels
input   : R the rules
input   : k the desired size of the pool
input   : $\mu$ the reference metric

$U \leftarrow \emptyset$                      ▷ indexes vector
$V \leftarrow R$                      ▷ Rules vector
for $l \in 1..k$ do
    $max \leftarrow -1$
    for $i \in V$ do
        $X \leftarrow U \cup i$
        $current = \mu(X)$          ▷ Computing the performance metric for X
        if $current > max$ then
            $i_{max} \leftarrow i$
            $max \leftarrow current$
    end
    $U \leftarrow U \cup i_{max}$
    $V \leftarrow V \setminus i_{max}$
end
return $U$
end function
```

4.4.1 The dataset

For the experiments, we consider a real-world data-set composed of $N = 359,862$ observations/rows made available by our industrial partner. It concerns a set of cases in credit card fraud detection generated in a specific business line where 288 rules are in production (this represents a subset of the actual operational rules: for confidentiality purposes, the complete collection of rules was not made available). Each observation represents a transaction and consists of a vector (X,Y) of 289 binary values where $X \in \{0,1\}^{288}$ and $Y \in \{0,1\}$. Each column variable X_j in the vector X (with $j \in [1,\ldots,288]$) correspond to a rule. The value of this variable tells whether the transaction has triggered this rule: a "1" (Resp. "0") means that the corresponding rule has (Resp. "has not") generated an alert for this transaction. The binary variable Y equals 1 if the transaction was actually a fraudulent ($Y = 1$) transaction. In our data-set, we have a total of 136,223 fraudulent transactions.

The 288 rules are aggregated by the OR operator. Let $\mathscr{X}$ be the output of the fraud detection system, the output of the system is $\mathscr{X} \equiv \max_i(X_i)$. In other words, if at least an alert has been produced on this transaction by the fraud detection system $\mathscr{X} = 1$, while if no alert has been generated $\mathscr{X} = 0$. Under this settings, we can define respectively the *true positives* (TP) as those for which ($\mathscr{X} = 1, Y = 1$), the *false positives* (FP) as those for which ($\mathscr{X} = 1, Y = 0$) and the *false negatives* (FN) those for which ($\mathscr{X} = 0, Y = 1$). Note that in our data-set we have no information about the *true negative* cases (non-fraudulent transactions which generated no alerts). The cardinality of the three available categories was found to be the following: $\#TP = 42210$, $\#FP = 219390$ and $\#FN = 94013$.

As each individual rule is an *if-then(-else)* rule specific to a given fraud scenario, most of the rules are designed to have individually a good precision, but tend to have a low recall. We computed the *average individual rule performance*: the average individual rule precision is equal to 0.29 and the average recall is equal to 0.0016.

The performance measures *of the whole* OR-ed rule-pool containing the $n = 288$ rules turned out to be the following: precision $p = 0.16$, recall $r = 0.31$, F-score $f = 0.21$. Notice that a low precision is not uncommon in this phase, since, in the following phase, the investigators will improve the performance of the overall system by taking into account further domain knowledge. The real goal of the system is to have a high recall, and precision not to low (so that the experts do not spend too much time investigating cases that turn out to be false positives).

During the explorative analysis, we noticed that the original dataset contained rules that had very little impact on our results since they were triggered very few times. For instance, 5 rules were triggered only once. Only 51 rules were triggered more than 360 times: this fraction corresponds approximately to $1/1000th$ of the dataset size. We limited our analysis to those 51 rules.

Execution times The actual execution time of the algorithms implemented in Python, on a 6-core +HT host (Intel i7 8850H, 2.6GHz, 64Gb RAM) using the whole $\simeq 360000$ record dataset and $\simeq 50$ rules is of the order of one minute for $1R$, same for GFS, about ten minutes for SV, about one hour for $SVFS$, using $N = 250$ permutations. As we show below, there is essentially no difference in the outputs of SV, and $SVFS$. In any case, the execution time of the algorithms is affordable even using higher N and longer historical records, if, as it happens in practice, the typical frequency of the rule assessment procedure is of the order of once a month.

4.4.2 Preliminary observations and findings

In order to assess the generality of the results, we used 3-fold cross-validation: i.e. we split the dataset, by choosing the transactions at random, into three chunks of $119,954$ records and in turn used two of them as the Training set and the remainder as a Test set. The performance values, shown in Figures 4, 5 and 6, are obtained averaging the test set performance of the three folds.

The estimate of the Shapley Value to be used for ranking the rules has been carried on using the permutation sampling method described in the previous section and formalized by Algorithm 2.

A first observation We have set the number of permutations to use for the estimate to $N = 250$ on the account that, adopting a much larger N does not change the results significantly. This is demonstrated in Figure 4.1: Performance of the OR-ed top-k rules classifier. We have from left to right: *precision* (left), *recall* (center) and *F-score* (right). The composition of the top-k set was established based on the top-k Shapley Values of the F-score, respectively estimated using $N = 250$ permutations (blue lines) and estimated using $N = 1000$ permutations (red lines). One can observe that the using only $N = 250$ does not influence significantly the curve: for this reason we used $N = 250$ in the following experiments.

.

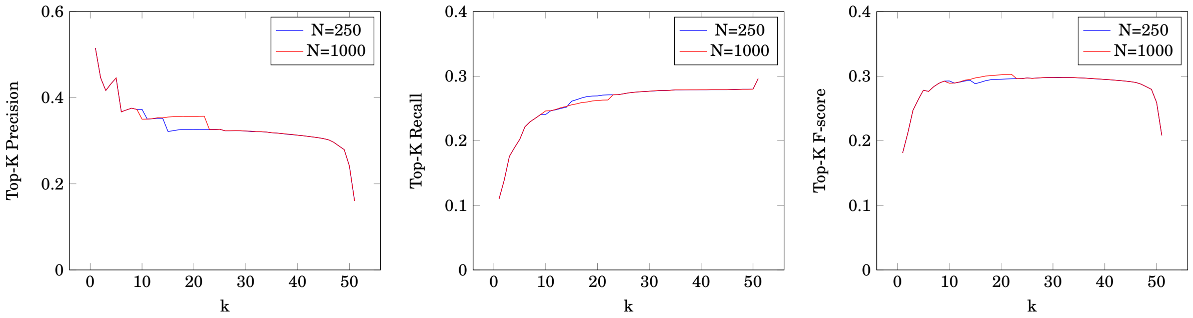

Figure 4.1: Performance of the OR-ed top-k rules classifier

A second observation The SV-based ranking and the SVFS-based ranking yield the same performance of the rules, as can be seen in Figure 4.2. Performance of the OR-ed top-k rules classifier using as *rule ranking metric* the Shapley Value with respect to same metric computed on the pool (red lines) and the SVFS with respect to same metric computed on the pool (green lines). Left: the comparison of the ranking metric SV with respect to precision and SVFS with respect to precision in terms of assessment metric: top-k classifier precision. Center: the comparison of the ranking metric SV with respect to recall and SVFS with respect to recall in terms of assessment metric: top-k classifier recall. Right: the comparison of the ranking metric SV with respect to F-score and SVFS with respect to F-score in terms of assessment metric: top-k classifier F-score. One can observe that the use of the (more costly) incremental method SVFS does not bring any significant improvement: for this reason we used the SV method.

Thus, for comparison with 1R and GFS, we use only the SV-ranking, which has a comparatively much lower complexity with respect to SVFS-ranking.

4.4.3 Validation of the first contribution

In this section we show that the rule-pool-based approach – assessing the rules based on their contribution to the performance of the OR-ed pool, quantified by the Shapley Value – fares

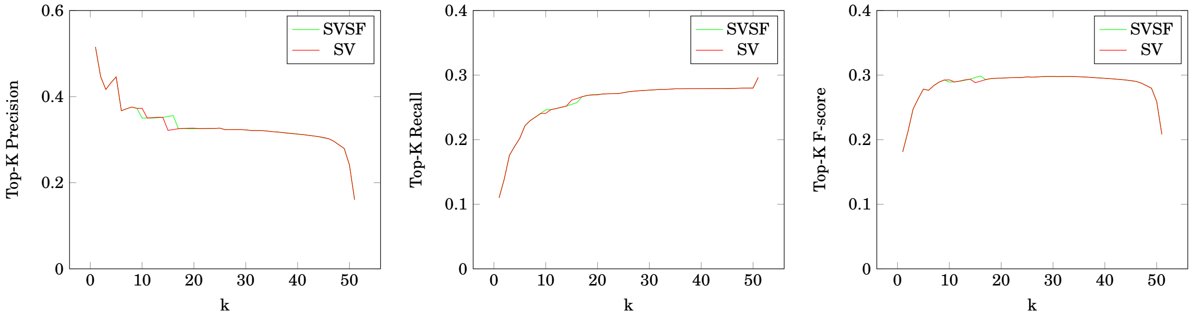

Figure 4.2: Performance of the OR-ed top-k rules classifier using as *rule ranking metric* the Shapley Value with respect to same metric computed on the pool (red lines) and the SVFS with respect to same metric computed on the pool (green lines).

better than the traditional individual-rule-based one – where rules are assessed based on their performance in isolation.

For the sake of clarity, we recall and highlight the distinction between (1) the metric, say v, used for ranking the rules and (2) the metric, say μ to evaluate the performance of the OR-aggregated ensemble of rules:

1. the first kind of metric, v, is used by an algorithm as a reference for establishing a ranking, we call it the *rule-ranking metric*; it allows to evaluate the usefulness of the rule, and, optionally, to extract, from the whole pool of n rules, the k top-ranked rules (e.g. SV, added value...)

2. the second kind of metric, μ, is used as a *pool assessment metric* for the OR-aggregated rule pool classifier (the one obtained aggregating in OR the set of rules in the highest k ranks according to a given *rule-ranking metric*) (e.g. precision, recall ...).

Pool-assessment metric The choice of the second metric, μ, is dictated by the business case. The one at hand dictates that pool assessment metric should privilege high recall over high precision, still not disregarding precision: it could be defined as a weighted average of the two, the F_β-score, however, in practice, there is no fixed relative weight that can be used as a sharp reference. The rule managers evaluate the trade-off case by case. We considered also another performance metric, a variant of the Area Under the Precision-Recall Curve: in the NRT-phase application context it is customary to compute the area under such a curve at the point where the recall reaches the 20% level, which is more representative of the processing capacity of the subsequent expert investigator assessment. We denote this metrics by $AUC\text{-}PR_{0.2}$.

Rule-ranking metric The choice of the rule ranking metric, v, ought to be made so as to *maximize* the pool performance in terms of the pool assessment metrics μ. Traditionally, the rule-ranking metric used was the individual rule precision, $v = p$, and one would count on the fact that a large set of moderately-high precision rules OR-ed in a pool, would – all together, – provide also a high recall. We propose to use not the *individual rule precision* but of the *contribution of the rule to the overall pool precision* as a rule-ranking metric: such contribution is quantified by the Shapley Value of the rule with respect to the pool precision, i.e. by $Sha[p]$.

In our experimentation we made the following comparisons of *rule ranking metrics* v:

a) individual rule precision vs. SV of the rule in contributing to the pool precision, i.e. $[p]$ vs. $Sha[p]$ (see Figure 4.3)

b) individual rule recall vs. SV of the rule in contributing to the pool recall, i.e. $[r]$ vs. $Sha[r]$ (see Figure 4.4)

c) individual rule F-score vs. Shapley Value of the rule in contributing to the pool F-score, i.e. $[f]$ vs. $Sha[f]$ (see Figure 4.5)

From each pair of ranking criteria, we obtained the corresponding pair of top-k rules; we put the rules in OR and obtained a pair of rule-pool classifiers; then we compared the performances of the two elements of the pair by a *pool assessment metric*; the assessment metric μ considered were precision, recall, F-score (they correspond to distinct columns of the figures) and $AUC\text{-}PR_{0.2}$.

The overall results of the comparisons of the pairs of rule ranking metrics are shown in Figures 4.3, 4.4 and 4.5: the 1R performance is represented in blue, the SV performance is represented in red.

In Figure 4.3, one can observe that the performance of the SV-based ranking, in the case of the pool assessment metrics recall and F-score, is remarkably higher than the traditional one for the OR-ed top-k rules until $k \simeq 30$. On the contrary, in the case of the pool assessment metric precision, it is the traditional approach that prevails. However, as mentioned above, in the business case at hand, it is preferable to have a large recall or a large F-score than a large precision at the expense of recall. Further inspection of the data confirmed that the high-precision points corresponded to irrelevant or negligible pool recall.

In Figure 4.4 one can observe that the performance of the different ranking criteria (recall and SV with respect to recall) are almost identical. In fact, by closer inspection of the rule rankings (not shown here), one would observe that the two metrics produce nearly *co-monotonic* rankings for the rules. The substantial equivalence of the ranking with the recall, and the Shapley value of the recall can be explained by the particularly small recall of individual rules as designed in this application domain: each rule is made to detect a specific kind of fraud. As a consequence rules have little overlap in terms of true detections, but a large overlap in terms of false alerts. The first fact makes the recall set function *nearly additive*: as a consequence, recall, SV of recall – and as we observe later, Greedy Forward Selection based on recall – rank the rules nearly in the same way. On the other hand, the observed overlap in terms of false alerts, makes the rules interact with one another: this is the reason why the precision is *remarkably not-additive* and we observe differences between the ranking based on precision and one based on the Shapley Value with respect to the pool precision.

In Figure 4.5 one can see that the SV-based rule ranking yields much better pool precision and F-score than the 1R rule ranking, whereas for recall the results are comparable.

The $AUC\text{-}PR_{0.2}$ achieved respectively by ranking the rules based on individual rule performance (Algorithm 1), and by ranking the rules based on the Shapley Value of the performance (Algorithm 2) are the following: when the base performance metric is precision we have 0.25 for 1R and 0.24 for SV; when the base performance metrics is recall we have 0.20 for both; when the base performance metrics is the F-score we have 0.20 for 1R and 0.24 for SV.

In short, by ranking the rules by the Shapley Value of a performance metric, one fulfills the business requirements better than by ranking them by the performance metrics itself.

4.4.4 Validation of the second contribution

To compare the SV- based rule ranking to other Feature Selection algorithms we chose as a reference the Greedy Forward Selection (GFS) algorithm (Algorithm 5). The results of the comparisons are shown in Figures 4.3, 4.4 and 4.5: the performance of the SV-based rule ranking in represented in red, the one of GFS is represented in green.

In Figure 4.3, where the base metric is the precision, one can see that in terms of pool precision GFS prevails over SV, but in terms of recall and F-score SV outperforms GFS. We have: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules; different line colors correspond to different rule ranking metrics based on precision: individual rule precision (blue lines), Shapley Value with respect to the pool precision (red lines) and Greedy Forward Selection (GFS) based on precision (green lines). One can see that sorting the rules according to the SV of the precision provides better performance for the recall for $k \leq 30$ and for the F-score over all the values of k: this meets the requirements of the NRT fraud detection phase, where one looks for good recall and moderate precision (the precision will be improved by human experts in the subsequent investigation phase).

In Figure 4.4, where the base metric is the recall, SV and GFS have nearly the same performance. As already noted, this can be explained by the fact that the rule recall is nearly additive. we have: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules; different line colors correspond to different rule ranking metrics based on recall: individual rule recall (blue lines), Shapley Value with respect to the pool recall (red lines) and Greedy Forward Selection (GFS) based on recall (green lines). All the methods based on recall yield approximately the same results, since their rankings are nearly co-monotonic. Indeed, individual rules are designed to capture specific fraud cases, so the true detections have little overlap, which makes the recall *nearly additive*. For an additive set function, the value of the function at the atom and the SV of the atom coincide, furthermore, the ranking by SV and by GFS would be the same.

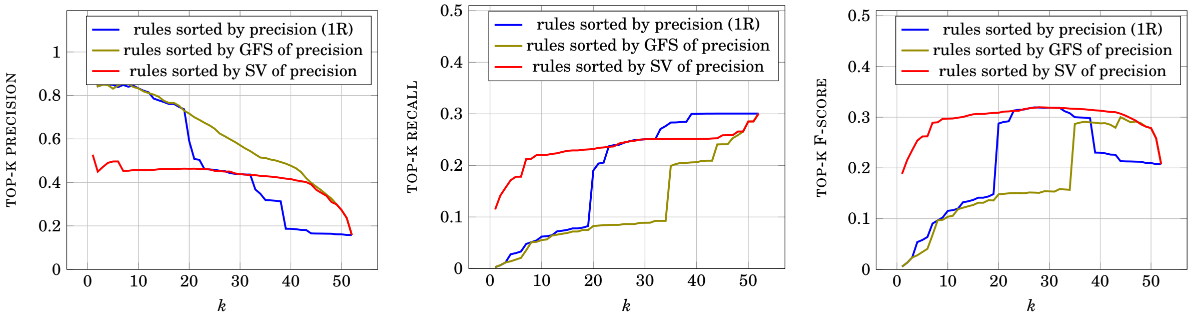

Figure 4.3: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules

In Figure 4.5, where the base metric is the F-score, SV and GFS are not too far from one another: GFS provides better pool precision and F-score, SV provides better pool recall. The pool F-scores provided by the two algorithms are remarkably close. We have: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules; different line colors correspond to different rule ranking metrics based on the F-score: individual rule F-score (blue lines), Shapley Value with respect to the pool F-score (red lines) and Greedy Forward Selection (GFS) based on F-score (green lines). Here the SV rule ranking yields much better pool precision and F-score than 1R rule ranking, whereas for recall the results are comparable. Also the pool recall and F-score determined by the GFS rule ranking and the SV rule ranking are comparable.

In terms of $AUC\text{-}PR_{0.2}$, we found that the two ranking methods provide nearly identical results. Rounding the values to the second decimal figure we have the following: when the base performance metrics is precision both algorithms yield a metric value of 0.24; when the base performance metric is recall both algorithms yield a metric value of 0.20 (as expected); when the base performance metrics is F-score SV yields 0.24; and GFS 0.23.

In short, Shapley Value ranking and Greedy Forward Selection ranking are nearly equivalent. However, as remarked in a previous section, the SV-based ranking has the advantage of providing a normalized score which summarizes the performance of the rule in terms of contribution to the pool and is suitable to support rule management decisions one rule at a time.

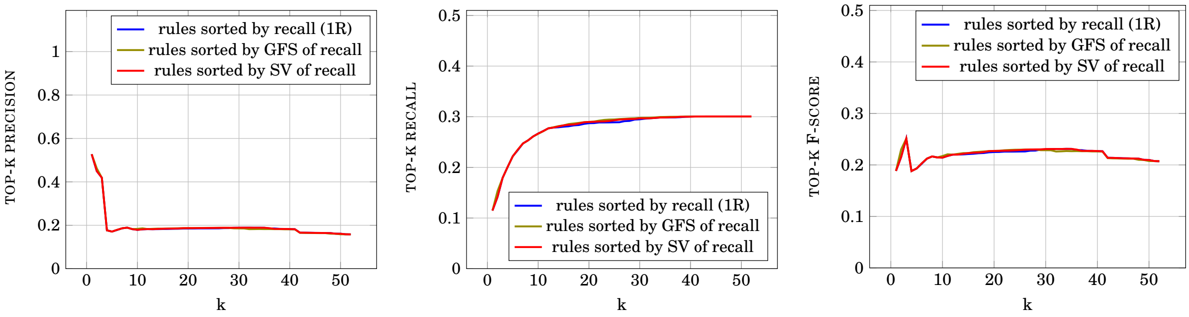

Figure 4.4: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules;

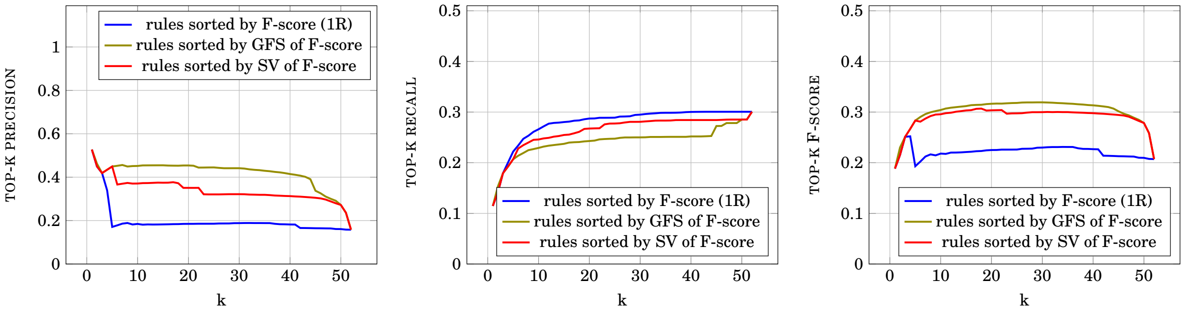

Figure 4.5: Precision (left), recall (center) and F-score (right) of the classifier defined by the OR-ed top-k rules

4.5 Conclusions

The management of a pool of classification rules for credit card fraud detection takes care of updating the set of member rules based on their performance. In the Near-Real-Time (NRT) the classification aims at high recall and moderate precision since the latter is later improved by the human experts. The continuous shift in fraud patterns requires the continuous revision of the set of rules present in the pool. This key activity represents a considerable overhead to the process.

In this work, we addressed the following points: the criteria used to select which rules to keep operational in the NRT pool are traditionally based on the historical performance of the individual rules, *considered in isolation*. This approach disregards the non-additivity of rule composition within the pool. We proposed to use an approach based on estimating the individual rule contribution to the overall pool performance through the Shapley Value (SV). We validated our approach using real-world credit card fraud data containing approximately 3×10^5 records of transactions. The analysis shows that the SV ranking based approach is more effective and efficient than the traditional one. The effectiveness of the approach, when considered as a Feature Selection method, is comparable to other standard FS algorithms. We demonstrated this point using the Greedy Forward Selection algorithm. The assignment of a "performance in collaboration" score to the rule distinguishes this approach to rule management from other approaches aimed at Feature Selection: the latter methods can indeed provide good coalitions rules of a given size k, but are not designed to assign a normalized score to the rules, summarizing their performance in collaboration. This score allows to assess the rule individually, which is useful in the periodic rule assessment process.

This power-index based approach has originated a patent application [13] and has been adopted, in the management of the rules for the NRT phase, by our industrial partners.

4.6 Summary

In this chapter, we introduce the power-index-based approach into the field of credit card fraud detection, that, by providing a synthesis of the usefulness of the rule, that supports the rule-by-rule periodic assessment process. In particular, we apply a Coalitional Game Theory (CGT) concept introduced in Chapter 2 and feature selection task introduced in Chapter 3, on a real transaction dataset coming from a fraud detection context. By doing this, we show that the novel approach we proposed i.e., the Shapley Value based approach support much better the rule management process. Our proposed process focuses on the overall performance of each rule, considering one rule at time and periodically revising the usefulness of the rule (it chooses either to keep or to drop the rule). The Shapley value score provides a synthesis of the rules usefulness through a normalized rule score. The assignment of the normalized score to the rule distinguishes the proposed approach from the other approaches, used for feature selection: for instance, the greedy forward selection used as benchmark can provide good coalition rules of size k, but it does not assign a score to the rule, summarizing its performance in collaboration. Moreover, in principle, the application of the Greedy forward selection during the monthly reassessment could allow the important changes in the composition of the pool, while the Shapley Value score supports a fine-grained control over the composition of such pool. But, first, going from credit card fraud detection, we describe the process and its management in Section 4.2; then, in Section 4.3, we analyze the Shapley Value and its properties. The Section 4.4 validate the results and, finally, the section 4.5 conclude the chapter.

CHAPTER 5

Restricted Banzhaf Index (k-Banzhaf) and Feature Selection

The chapter provides tools and concepts to understand the second power index under our analysis in this dissertation, namely the Banzhaf index. The latter brings us to a new power index developed form the original one. From the original Banzhaf index, we propose a new version called *the restricted Banzhaf index (k-Banzhaf)* and argue that our version is more efficient than the original one. The chapter is composed into two main parts: in the first part, we propose to look into the Banzhaf index, then from its properties, we develop and implement a new power index. The main advantage of the new version of Banzhaf is its reduced computational cost. We are going to see why k-Banzhaf is computationally less expensive than the normal Banzhaf.

In the second part, we implement different power indexes-based feature selection using (Shapley Value, Banzhaf Index and k-Banzhaf Index) and two greedy method-based feature selection (greedy forward selection and greedy backward elimination). We build a self-training process (a variant of bootstrap) based on a selected machine learning classifier, namely Random Forest Classifier (RFC). We make three scenarios of machine learning classification to compare our five feature selection performance (Three power indexes- based and two greedy-based). The performance metric we choose for the comparison is the accuracy.

5.1 Introduction

We wish to extend our study of coalitional game theory approach on credit card fraud detection by including a new power index; the one chosen is the Banzhaf index, the second well-known power index. We discuss Banzhaf index approximation, propose a variant of Banzhaf index (k-Banzhaf) and implement the k-Banzhaf-based feature selection method which we compare with Shapley and Banzhaf-based feature selection ones. So in this chapter, we provide tools and expressions to understand the approximation of the Banzhaf index, the demonstrations provided in the following sections shows that Banzhaf Index is, in general the best approximation based on the mean square error.

A pseudo-boolean function (i.e. a real-value set function) $\mu : S \in 2^N \longrightarrow R$, as shown by Hammer and Rudeanu in [40] can be represented univocally by the following multi-linear polynomial

$$\mu = \mu(S) = \sum_{T \in 2^N} a_T \prod_{i \in T} x_i$$

where $x = x_S = (x_1, x_2, ..., x_n)$ is the boolean characteristic function of a set S ($x_i = 1$ if the element i is contained in S, otherwise $x_i = 0$). The a_T are known as the Moëbius coefficients (or, in Coalitional Game Theory, as *Harsanyi dividends*).

The element $\prod_{i \in T} x_i$ form the so called Unanimity Game basis. We call this expansion the Moëbius representation of μ, and the basis will be called Moëbius basis. The expression of the function μ of a set S in terms of the Moëbius basis is called Moëbius expansion of $\mu(S)$. For functions P such that $\mu(N) = 1$, the a_T's are normalized to 1, i.e. $\sum_{T \in 2^N} a_T = 1$. The degree of μ is defined as the size of the largest set S with a nonzero coefficient. Notice that the coefficient a_T contributes to the sum of $\mu(S)$ if and only if the monomial $\prod_{i \in T}$ evaluates to 1, which happens only if T is a subset of S, so that for a set $S \in 2^N$ one has $\mu(S) = \sum_{T \subseteq S} a_T$. The Moëbius expansion of $\mu(S)$ involves all the subsets of S; no super-sets of S are involved.

The pseudo-Boolean functions with domain 2^N form a linear space, that we denote by $\mathscr{PB}^n$ (since we can point-wise add two functions and we can multiply a function by a real scalar to get another function of the same space). On this space one can define the scalar product of two functions f and g by $\langle f, g \rangle \equiv \frac{1}{2^n} \sum_{T \in 2^N} f(x) g(x)$.

Let us denote by $S_k = S_k(N)$ the family of sets consisting of exactly k elements from N (it represents the k-th level of the Boolean lattice 2^N), i.e.

$$S_k = S_k(N) \equiv \{T \in 2^N : |T| = k\}$$

and let $S_k^{\leq}$ be the family of sets with at most k elements, i.e.

$$S_k^{\leq} = S_k^{\leq}(N) \equiv \{S \in N : |S| \leq k\} = \cup_{j=0}^{k} S_j.$$

The family contains the empty set. The pseudo-boolean functions on $S_k^{\leq}$ form the subspace $\mathscr{PB}^{\leq k}$ of $\mathscr{PB}^n$.

We assume the coefficients a_T to be known, and consider the problem of approximating μ by another pseudo-boolean function g given by a first degree polynomial:

$$g = g^{(1)} \equiv g^{(1)}(x_1, x_2, ..., x_n | \alpha_0, \beta_1, \beta_2, ..., \beta_n) = \alpha_0 + \sum_{i \in N} \beta_i x_i$$

We look for the β_i's that minimize the sum of the squared differences over *a chosen family* $\mathscr{Q} \subseteq 2^N$ of relevant sets. We consider few important cases

- the case $\mathscr{Q} = 2^N$ already studied in [41]
- the case $\mathscr{Q} = S_k$
- the case $\mathscr{Q} = S_{k_{min}} \cup S_{k_1} ... \cup S_{k_j} ... \cup S_{k_{max}}$

For practical purposes it is useful to consider also another family, related to $\mathscr{Q}$: the family of all the subsets of the sets in $\mathscr{Q}$. Indeed, in order to express the measure μ of a set S, one needs in general all the subsets of that set from $\emptyset$; to S itself. Specifically, let $S \in 2^N$: the family of sets which are subsets of S will be called the *down-family* (or the *down-set*) [1] of S and denoted by $\mathscr{D}_S$. Sometimes one does not need only the down-family of a set but rather to consider all the down-families of a collection of sets. Given the collection $\mathscr{Q}$, the union of all the down-families of its sets is called *down-family generated* by $\mathscr{Q}$, and denoted by $\mathscr{D}_{\mathscr{Q}}$:

$$\mathscr{D}_{\mathscr{Q}} = \{T \in 2^N : \exists S \in \mathscr{Q} \; s.t. \; T \subseteq S\}$$

The down-family generated by $\mathscr{Q}$, for the considered cases, corresponds to the following:

- in the case $\mathscr{Q} = 2^N$, we have $\mathscr{D}_{\mathscr{Q}} = 2^N = \mathscr{Q}$
- in the case $\mathscr{Q} = S_k$, we have $\mathscr{D}_{\mathscr{Q}} = S_k^{\leq}$
- in the case $\mathscr{Q} = \cup_{k=k_{min}}^{k_{max}} S_k$, we have $\mathscr{D}_{\mathscr{Q}} = S_k^{\leq}$

5.2 The best linear approximation

We are looking for the linear approximation that minimizes the sum of the squared differences. An important fact is that, once found, we can exploit some special properties of such function. Consider the following. The space $V = \mathscr{P}\mathscr{B}^n$ of the pseudo-boolean functions on N is a Hilbert space (isomorphic to $\mathbb{R}^{2^n}$). Hilbert spaces generalize the properties of the Euclidean space. Linear approximations of functions are projections of this space onto the subspace $U = \mathscr{P}\mathscr{B}^1$ of the functions of degree one (of dimension $n+1$). Among those projections, the projection with minimal distance is the orthogonal projection [2]. Orthogonal projections grant linearity, i.e.

$$g(\eta 1 f 1 + \eta 2 f 2) = \eta 1 g(f 1) + \eta 2 g(f 2).$$

[1] If a down-set is nonempty and closed under $\cup$, it is called an *ideal*. Similar definitions can be adopted for the families of super-sets of a set S, called *upsets*: an upset closed under $\cap$ is called *filter*.

[2] Indeed, the orthogonal projection provides the minimal distance and is unique. Let $v \in V$ and $u \in U$, and let $\pi(v)$ the orthogonal projection of v on u. The vector $v - u$ is the hypotenuse of a right-angled triangle. Then for the Pythagorean theorem

$$\|v-u\|^2 = \|v-\pi(v)\|^2 + \|u-\pi(v)\|^2 \geq \|v-\pi(v)\|^2$$

Recall that projections are idempotent and notice that in the new space $\pi(v-\pi(v)) = \pi(v) - \pi(v) = 0$, which is the minimum distance

Thanks to the linearity we can look individually for the best approximation to elements of the basis, then combine them linearly to get the best approximation of any $f \in \mathscr{P}\mathscr{B}^n$. Let $z_1,...,z_q$ a sequence of q real numbers. Let c be a real number to be determined. It is easy to show that the expression $\sum_{j=1}^{q}(z_j - c)^2$ is minimized by $qc = \sum_{j=1}^{q} z_j$. Now, suppose we have found $g(f)$ the best linear approximation of f and suppose c is a real value to be determined. The expression $\sum_{\mathscr{Q}}((f(x) - g(x)) - c)^2$ is minimized by $c = 0$. Hence $0 = \sum_{\mathscr{Q}}(f(x) - g(x))$ or

$$\sum_{Q\in\mathscr{Q}} f(x) \quad = \sum_{Q\in\mathscr{Q}} g(x) \tag{5.1}$$

Next, for $i \in N$ the function $g(x) + cx_i$ is a linear function: the expression

$$\sum_{Q\in\mathscr{Q}} (f(x) - g(x) + cx_i)^2$$

is minimized by $c = 0$, but c affects only the terms for which $x_i = 1$, i.e. $i \in Q$, so we can say that

$$\sum_{Q\in\mathscr{Q}:i\in Q} (f(x) - g(x) + c)^2$$

is minimized for $c = 0$, hence

$$\sum_{Q\in\mathscr{Q}:i\in Q} g(x) = \sum_{Q\in\mathscr{Q}:i\in Q} f(x)$$

from the two previous equalities, follows also the one for the case $x_i = 0$

$$\sum_{Q\in\mathscr{Q}:i\notin Q} g(x) = \sum_{Q\in\mathscr{Q}:i\notin Q} f(x)$$

and taking the difference of the latest two equalities one gets

$$\sum_{Q\in\mathscr{Q}:i\in Q} \Delta_i g(x) = \sum_{Q\in\mathscr{Q}:i\in Q} \Delta_i f(x) \tag{5.2}$$

Now for $g(x = \alpha_0) + \sum_{i\in N} \beta_i x_i$ one has $\Delta_i g(x) = \beta_i$, so that equation (5.2) can be used to determine β_i after which equation (5.1) is used to determine α_0. For practical purposes, in the latter equation, it is useful to write $Q \in \mathscr{Q} : i \in Q$ as $Q = (X \cup i) : X \in \mathscr{Q}_i$, where we denote by $\mathscr{Q}_i$ the family of those of $\mathscr{Q}$ that do not contain i.

$$\sum_{X\in\mathscr{Q}_{-i}} \Delta_i g(x) = \sum_{X\in\mathscr{Q}_{-i}} \Delta_i f(x) \tag{5.3}$$

We will denote the cardinality of a set by the corresponding lower case letter e.g. $t = |T|$, $x = |X|$. We will often use $m = (n-1)$, $h = (k-1)$.

In the following, besides 2^N the power set of the reference set N, a few families of sets will appear repeatedly: we will be interested to their cardinality in special cases. They are summarized here (we use calligraphic letters):

- $\mathscr{Q} \subseteq 2^N$ the family on which we wish to approximate the function f using g
- $\mathscr{Q}_{-i} \subset \mathscr{Q}$ the family of sets of $\mathscr{Q}$ that do not contain i
- $\mathscr{Q}_i \subset \mathscr{Q}$ the family of sets of $\mathscr{Q}$ that contain the element i (a.k.a the family of supersets of i in $\mathscr{Q}$, upper cone of i in $\mathscr{Q}$) : $\mathscr{Q}_{-i} = \mathscr{Q} \setminus \mathscr{Q}_i$
- $\mathscr{Q}_T$ the family of supersets of a given set T in $\mathscr{Q}$ (upper cone of T in $\mathscr{Q}$)
- $(\mathscr{Q}_{-i})_{(T\setminus i)}$ the family of supersets of a set $(T \setminus i)$ in $\mathscr{Q}_{-i}$ (the filter of T in $(\mathscr{Q}_{-i})$
- $\mathscr{Q}_{-ij} \subset \mathscr{Q}$ the family of $\mathscr{Q}$ that do not contain ij
- $(\mathscr{Q}_{-ij})_{(T\setminus ij)}$ the family of a given set $(T \setminus ij)$ in $\mathscr{Q}_{-ij}$

In the relevant cases, their cardinalities are the following:

- If $\mathscr{Q} = 2^N$

$$|\mathscr{Q}| = 2^n \qquad |\mathscr{Q}_{-i}| = |\mathscr{Q}_i| = 2^{n-1}$$

$$|\mathscr{Q}_T| = 2^{n-t} \qquad \left|(\mathscr{Q}_{-i})_{(T\setminus i)}\right| = 2^{(n-1)-(t-1)} = 2^{n-t}$$

$$|\mathscr{Q}_{-ij}| = |\mathscr{Q}_{ij}| = 2^{n-2} \qquad \left|(\mathscr{Q}_{-ij})_{(T\setminus ij)}\right| = 2^{(n-2)-(t-2)} = 2^{n-t}$$

- If $\mathscr{Q} = S_k$

$$|\mathscr{Q}| = \begin{pmatrix} n \\ k \end{pmatrix} \qquad |\mathscr{Q}_{-i}| = |\mathscr{Q}_i| = \begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$$

$$|\mathscr{Q}_T| = \begin{pmatrix} n-t \\ k-t \end{pmatrix} \qquad |(\mathscr{Q}_{-i})_{(T\setminus i)}| = \begin{pmatrix} (n-1)-(t-1) \\ (k-1)-(t-1) \end{pmatrix} = \begin{pmatrix} n-t \\ k-t \end{pmatrix}$$

$$|\mathscr{Q}_{-ij}| = |\mathscr{Q}_{ij}| = \begin{pmatrix} n-2 \\ k-2 \end{pmatrix} \qquad |(\mathscr{Q}_{ij})_{(T\setminus ij)}| = \begin{pmatrix} (n-2)-(t-2) \\ (k-2)-(t-2) \end{pmatrix} = \begin{pmatrix} n-t \\ k-t \end{pmatrix}$$

- If $\mathscr{Q} = \cup_{k=k_{min}}^{k_{max}}$

$$|\mathscr{Q}| = \sum_{k=k_{min}}^{k_{max}} \begin{pmatrix} n \\ k \end{pmatrix} \qquad |\mathscr{Q}_{-i}| = |\mathscr{Q}_i| = \sum_{k=k_{min}}^{k_{max}} \begin{pmatrix} n-1 \\ k-1 \end{pmatrix} = \sum_{h=k_{min}-1}^{k_{max}-1} \begin{pmatrix} m \\ h \end{pmatrix}$$

$$|\mathscr{Q}_T| = \sum_{k=k_{min}}^{k_{max}} \binom{n-t}{k-t} \qquad \left|(\mathscr{Q}_{-i})_{(T\setminus i)}\right| = \sum_{k=k_{min}}^{k_{max}} \binom{n-t}{k-t}$$

$$|\mathscr{Q}_{-ij}| = |\mathscr{Q}_{ij}| = \sum_{k=k_{min}}^{k_{max}} \binom{n-2}{k-2} \qquad \left|(\mathscr{Q}_{-ij})_{(T\setminus ij)}\right| = \sum_{k=k_{min}}^{k_{max}} \binom{n-t}{k-t}$$

Notice the useful fact that, for all the special families under the scope of this work, the following equality holds

$$|\mathscr{Q}_T| = \left|(\mathscr{Q}_{-i})_{(T\setminus i)}\right| = \left|(\mathscr{Q}_{-ij})_{(T\setminus ij)}\right|$$

5.2.1 Determination of β_i

Now in view of using linearity to approximate any f, let us choose

$$f = f^T = e_T(x) = \prod_{j\in T} s_j$$

i.e. an element of the Moëbius basis, as a function to approximate. We look for the best linear approximator $g^T(x) \equiv g(f^T)$ to the function $f^T(x)$ with

$$g^T(x) = \alpha_0^T + \sum_{i\in N} \beta_i^T x_i$$

Let us consider equation (5.3) that relates the sums of the first derivatives.

5.2.1.1 Left side

The left side (since $\Delta_i g^T(x) = \beta_i^T$) is

$$\sum_{Q\in\mathscr{Q}:i\in Q} \Delta_i g^T(x) = \sum_{Q\in\mathscr{Q}:i\in Q} \beta_i^T = \sum_{X\in\mathscr{Q}_{-i}} \beta_i^T = |\mathscr{Q}_{-i}|\beta_i^T$$

5.2.1.2 Right side

Now let us consider right side. If $i \notin T$, the derivative Δ_i is zero: $\Delta f^T(x) = 0$. Instead for $i \in T$

$$\Delta f^T(x) = \prod_{j\in T} x_j \Bigg|_{x_i=1} - \prod_{j\in T} x_j \Bigg|_{x_i=0} = \prod_{j\in T\setminus\{i\}} x_j - 0 = \prod_{j\in T\setminus\{i\}} x_j$$

hence,

$$\sum_{Q\in\mathscr{Q}:i\in Q} \Delta f^T(x) = \sum_{Q\in\mathscr{Q}_{-i}} \Delta f^T(x) = \sum_{X\in\mathscr{Q}_{-i}} \prod_{T\setminus\{i\}} x_j = \#(supersets\ of (T\setminus\{i\}) \quad in \quad \mathscr{Q}_{-i}) = \left|(\mathscr{Q}_{-i})_{(T\setminus i)}\right|$$

because the chain product is evaluated to 1 only on the superset of $(T \setminus \{i\})$.

5.2.1.3 Two sides

Using the two sides of (5.2) for f^T one has the equation for the β_i^T's:

$$|\mathcal{Q}_{-i}|\beta_i^T = \left|(\mathcal{Q}_{-i})_{(T\setminus i)}\right|$$

We get

$$\beta_i^T = \frac{\left|(\mathcal{Q}_{-i})_{(T\setminus i)}\right|}{|\mathcal{Q}_{-i}|} = \frac{|\mathcal{Q}_T|}{|\mathcal{Q}_{-i}|} \tag{5.4}$$

We observe that, in the considered approximations of f_T , the value of β_i^T does not depend on T directly, but only on its size, i.e. β_i^T is a level dependent coefficient: below we will use occasionally $\beta_i^T = \beta(t)$.

- If $\mathcal{Q} = 2^N$, the equation becomes

$$\beta_i^T = \frac{1}{2^{t-1}} \tag{5.5}$$

- If $\mathcal{Q} = S_k$, we have

$$\beta_i^T = \frac{\binom{n-t}{k-t}}{\binom{n-1}{k-1}}$$

This ratio of binomial coefficients can be written as follows

$$\beta_i^T = \frac{\binom{n-t}{k-t}}{\binom{n-1}{k-1}} = \frac{(k-1)_{t-1}}{(n-1)_{t-1}} \tag{5.6}$$

where $(m)_r$ denotes the (r-factors) falling factorial. Thus

$$\beta_i^T = \frac{(k-1)_{t-1}}{(n-1)_{t-1}} \tag{5.7}$$

- If $\mathcal{Q} = S_{k_{min}} \cup S_{k_1} ... \cup S_{k_j} ... \cup S_{k_{max}}$, the # (supertsets of $(T \setminus \{i\})$ in $\mathcal{Q}_{-i}$) is the sum of the number of supersets at all the levels k_j involved; we have

$$\sum_{k=k_{min}}^{k_{max}} \binom{n-1}{k-1} \beta_i^T = \sum_{k=k_{min}}^{k_{max}} \binom{n-t}{k-t}$$

and

$$\beta_i^T = \frac{\sum_{k=k_{min}}^{k_{max}} \binom{n-t}{k-t}}{\sum_{k=k_{min}}^{k_{max}} \binom{n-1}{k-1}}$$

Notice that those terms k such that $t > k$ (i.e. $x > h$) give zero contribution to the sum.

- In passing, we observe the following. Consider the limiting case $k_{min} = 0$ and $k_{max} = n$. The sum at the denominator, setting $m = (n-1)$, $h = (k-1)$, $p = (m-x)$ and $j = (h-x)$ (boundaries $k = 0 \Longrightarrow h = -1$, $k = n \Longrightarrow h = n-1 = m$) becomes

$$|\mathscr{Q}_{-i}| = \sum_{h=k_{min}}^{k_{max}} \binom{m-x}{h-x} = \sum_{k=0}^{n} \binom{n-1}{k-1} = \sum_{h=-1}^{m} \binom{m}{h} = \sum_{h=0}^{m} \binom{m}{h} = 2^m = 2^{n-1}$$

To evaluate the numerator, set $p = (m-x)$ and $j = (h-x)$

(boundaries $k_{min} = 0 \Longrightarrow h = -1 \Longrightarrow j = -(x+1)$ and $k_{max} = n \Longrightarrow h = n-1 = m \Longrightarrow j = m-x = p$). The numerator becomes

$$(\mathscr{Q}_{-i})_{(t\setminus i)} = \sum_{h=-1}^{m+1} \binom{m-x}{h-x} = \sum_{j=-(x+1)}^{p} \binom{p}{j} = \sum_{j=0}^{p} \binom{p}{j} = 2^p = 2^{m-x} = 2^{(n-1)-(t-1)}$$

Overall

$$\beta_i^T = \frac{\sum_{k=0}^{n} \binom{m-x}{h-x}}{\sum_{k=0}^{n} \binom{n}{k}} = \frac{2^{m-x}}{2^m} = \frac{1}{2^x} = \frac{1}{2^{t-1}}$$

i.e., we recover the Banzhaf coefficient of T.

5.2.2 Determination of α_0

To determine α_0 we use equation (5.1), i.e. the equality of the sums of $g^T(x)$ and $f^T(x)$

$$\sum_{Q \in \mathscr{Q}} \left(\alpha_0^T + \beta(t) \sum_{i \in T} x_i \right) = \sum_{Q \in \mathscr{Q}} f^T(x)$$

$$\sum_{Q \in \mathscr{Q}} \alpha_0^T + \beta(t) \sum_{i \in T} \sum_{Q \in \mathscr{Q}} x_i = \sum_{Q \in \mathscr{Q}} \prod_{j \in T} (x_j)$$

$$|\mathscr{Q}| \alpha_0^T + \beta(t) \sum_{i \in T} \#(supersets\ of\ \{i\}\ in\ \mathscr{Q}) = \#(supersets\ of\ T\ in\ \mathscr{Q})$$

$$|\mathscr{Q}| \alpha_0^T + t\beta(t) |\mathscr{Q}_i| = |\mathscr{Q}_T|$$

$$|\mathcal{Q}|\alpha_0^T + t\frac{\left|(\mathcal{Q}_{-i})_{(T\setminus i)}\right|}{|\mathcal{Q}_{-i}|}|\mathcal{Q}_i| = |\mathcal{Q}_T|$$
$$|\mathcal{Q}|\alpha_0^T + t\left|(\mathcal{Q}_{-i})_{(T\setminus)}\right| = |\mathcal{Q}_T|$$

$$|\mathcal{Q}|\alpha_0^T = -t\left|(\mathcal{Q}_{-i})_{(T\setminus i)}\right| + |\mathcal{Q}_T|$$

and finally, since $\left|(\mathcal{Q}_{-i})_{(T\setminus i)}\right| = |\mathcal{Q}_T|$

$$|\mathcal{Q}|\alpha_0^T = -(t-1)|\mathcal{Q}_T| \tag{5.8}$$

i.e.

$$\alpha_0^T = -(t-1)\frac{|\mathcal{Q}_T|}{|\mathcal{Q}|} \tag{5.9}$$

Hereafter the symbol $\alpha(t)$ is often used in place of α_0^T because the latter does not depend directly on T, but rather on T's size t. In other words also α^T is a level dependent coefficient.

- If $\mathcal{Q} = 2^N$, we have

$$\alpha_0^T = \quad \alpha(t) \quad = -\frac{t-1}{2^t} \tag{5.10}$$

- If $\mathcal{Q} = S_k$, we have

$$\alpha_0^T = -(t-1)\frac{\binom{n-t}{k-t}}{\binom{n}{k}}$$

and finally

$$\alpha_0^T = \quad \alpha(t) \quad = \quad -(t-1)\frac{(k)_t}{(n)_t} \tag{5.11}$$

- If $\mathcal{Q} = S_{k_{min}} \cup S_{k_1} ... \cup S_{k_j} ... \cup S_{k_{max}}$, we have

$$\alpha_0^T = -(t-1)\frac{\sum_{h=k_{min}-1}^{k_{max}-1}\binom{m-x}{h-x}}{\sum_{h=k_{min}-1}^{k_{max}+1}\binom{m+1}{h+1}} \tag{5.12}$$

5.2.3 Wrap up

All together

$$g^T(x) = \alpha(t) + \beta(t) \sum_{i\in T} x_i$$

where

$$\alpha_0^T = \alpha(t) \quad = \quad -(t-1)\frac{|\mathcal{Q}_T|}{|\mathcal{Q}|}$$

$$\beta_i^T = \beta(t) = \frac{|\mathcal{Q}_T|}{|\mathcal{Q}_{-i}|} \tag{5.13}$$

5.2.4 General expression of the linear approximation in the Moëbius basis

The expression for a general function f follows from the above one by linearity, summing over *all the subsets* of the members of $\mathcal{Q}$, i.e. summing over $\mathcal{D}_{\mathcal{Q}}$ (in the Moëbius basis a set S is univocally determined by the Moëbius coefficients of its subsets). The expression for f is

$$f(x) = \sum_{T\in\mathcal{D}_{\mathcal{Q}}} a_T \left[\prod_{j\in T} x_j\right]$$

$$f(x) = \sum_{T\in\mathcal{D}_{\mathcal{Q}}} a_T f^T \tag{5.14}$$

The equation for the approximator is

$$\begin{aligned} g(x) &= \sum_{T\in\mathcal{D}_{\mathcal{Q}}} a_T g^T(x) \\ &= \sum_{T\in\mathcal{D}_{\mathcal{Q}}} a_T \left[\alpha(t) + \beta(t) \sum_{i\in T} x_i\right] \\ &= \sum_{T\in\mathcal{D}_{\mathcal{Q}}} \alpha(t) a_T + \sum_{T\in\mathcal{D}_{\mathcal{Q}}} \beta(t) a_T \sum_{i\in T} x_i \end{aligned}$$

$$g(x) = \sum_{T\in\mathcal{D}_{\mathcal{Q}}} \alpha(t) a_T + \sum_{i\in N}\left[\sum_{T\in\mathcal{D}_{\mathcal{Q}}:i\in T} \beta(t) a_T\right] x_i \tag{5.15}$$

i.e.

$$\alpha_0 = \sum_{T\in\mathscr{D}_{\mathscr{Q}}} \alpha(t)a_T \quad = -\frac{1}{|\mathscr{Q}|}\sum_{T\in\mathscr{D}_{\mathscr{Q}}}(t-1)|\mathscr{Q}_T|a_T \tag{5.16}$$

$$\beta_i = \sum_{T\in\mathscr{D}_{\mathscr{Q}}:i\in T} \beta(t)a_T = \quad \frac{1}{|\mathscr{Q}_{-i}|}\sum_{T\in\mathscr{D}_{\mathscr{Q}}:i\in T}|\mathscr{Q}_T|a_T \tag{5.17}$$

For the cases considered by substituting $\alpha(t)$ and $\beta(t)$ one has the following

- If $\mathscr{Q} = 2^N$ we have

$$g(x) = -\sum_{T\in 2^N}\frac{t-1}{2^t}a_T + \sum_{i\in N}\left[\sum_{T\in 2^N:i\in T}\frac{a_T}{2^{t-1}}\right]x_i$$

  i.e.

$$\alpha_0 = -\sum_{T\in 2^N}\frac{t-1}{2^t}a_T \qquad \beta_i = \sum_{T\in\mathscr{D}_{\mathscr{Q}}:i\in T}\frac{1}{2^{t-1}}a_T \tag{5.18}$$

  Notice that the β_i above coincide with the Banzhaf coefficients of the i's

- If $\mathscr{Q} = S_k$ we have

$$g(x) = -\sum_{T\in S_k^{\le}}\left[(t-1)\frac{(k)_t}{(n)_t}\right]a_t + \sum_{i\in N}\left[\sum_{T\in S_k^{\le}:i\in T}\frac{(k-1)_t-1}{(n-1)_{t-1}}\right]a_Tx_i$$

$$\alpha_0 = -\sum_{T\in S_k^{\le}}\left[(t-1)\frac{(k)_t}{(n)_t}\right]a_T \qquad \beta_i = \sum_{T\in S_k^{\le}:i\in T}\left[\frac{(k-1)_{t-1}}{(n-1)_{t-1}}\right]a_T \tag{5.19}$$

  by analogy we indicate the above as the k-restricted Banzhaf coefficients.

- If $\mathscr{Q} = S_{k_{min}} \cup S_{k_1}\ldots\cup S_{k_j}\ldots\cup S_{k_{max}}$, the derivation of the expression is equally straightforward.

5.2.5 General expression of α and β in terms of f and $\Delta_i f$

5.2.5.1 Expression of β_i in terms of $\Delta_i f$

We recall from equation (5.3) that

$$\sum_{X\in\mathscr{Q}_{-i}}\Delta_i g(x) = \sum_{X\in\mathscr{Q}_{-i}}\Delta_i f(x)$$

On the other hand, by the definition of the linear approximator $\Delta_i g(x) = \beta_i$, so the left side is $|\mathcal{Q}_{-i}|\beta_i$. It follows that

$$(5.20) \qquad \beta_i = \frac{1}{|\mathcal{Q}_{-i}|} \sum_{X \in \mathcal{Q}_{-i}} \Delta_i f(x) \quad = \frac{1}{|\mathcal{Q}_{-i}|} \sum_{X \in \mathcal{Q}_{-i}} [f(X \cup i) - f(X)]$$

- If $\mathcal{Q} = 2^N$, then $\mathcal{Q}_{-i} = 2^{N \setminus i}$

  $$(5.21) \qquad \beta_i = \frac{1}{2^{n-1}} \sum_{X \in 2^{N \setminus i}} \Delta_i f(x) \quad = \frac{1}{2^{n-1}} \sum_{X \in 2^{N \setminus i}} [f(X \cup i) - f(X)]$$

  i.e. we recover the alternative definition of the Banzhaf value.

- If $\mathcal{Q} = S_k$, then $\mathcal{Q}_{-i} = (S_k)_{-i}$

  $$(5.22) \qquad \beta_i = \frac{1}{\binom{n-1}{k-1}} \sum_{X \in (S_k)_{-i}} \Delta_i f(x) \quad = \frac{1}{\binom{n-1}{k-1}} \sum_{X \in (S_k)_i} [f(X \cup i) - f(X)]$$

- If $\mathcal{Q} = S_{k_{min}} \cup S_{k_1} ... \cup S_{k_j} ... \cup S_{k_{max}}$, then $|\mathcal{Q}_{-i}| = \sum_{k=k_{min}}^{k_{max}} \binom{n-1}{k-1}$

  $$(5.23) \qquad \beta_i = \frac{1}{\sum_{k=k_{min}}^{k_{max}} \binom{n-1}{k-1}} \sum_{X \in \mathcal{Q}_{-i}} \Delta_i f(x) \quad = \frac{1}{\sum_{k=k_{min}}^{k_{max}} \binom{n-1}{k-1}} \sum_{X \in \mathcal{Q}_{-i}} [f(X \cup i) - f(X)]$$

Notice that in equation (5.20), singling out the case $X = \emptyset$, if we additionally assume $f(\emptyset) = 0$, we can write

$$\beta_i = \frac{f(i)}{|\mathcal{Q}_i|} + \frac{1}{|\mathcal{Q}|} \sum_{X \in \mathcal{Q}_{-i} : X \neq \emptyset} \Delta_i f(x)$$

5.2.5.2 Expression of α in terms of f

As to α, it can be obtained from the equality of the sums (5.1):

$$\sum_{S \in \mathcal{Q}} g(x_S) = \sum_{S \in \mathcal{Q}} f(x_S)$$

$$\sum_{S \in \mathcal{Q}} \left[\alpha + \sum_{i \in N} \beta_i x_i \right] = \sum_{S \in \mathcal{Q}} f(x_S)$$

$$
\begin{aligned}
|\mathcal{Q}|\alpha + \sum_{S\in\mathcal{Q}}\left[\sum_{i\in N}\beta_i x_i\right] &= \sum_{S\in\mathcal{Q}} f(x_S)\\
|\mathcal{Q}|\alpha &= \sum_{S\in\mathcal{Q}} f(x_S) - \sum_{S\in\mathcal{Q}}\left[\sum_{i\in N}\beta_i x_i\right]\\
|\mathcal{Q}|\alpha &= \sum_{S\in\mathcal{Q}}\left[f(x_S) - \sum_{i\in N}\beta_i x_i\right]\\
\alpha &= \frac{1}{|\mathcal{Q}|}\sum_{S\in\mathcal{Q}}\left[f(x_S) - \sum_{i\in S}\beta_i x_i\right]
\end{aligned}
$$

where the β_i are those given in (5.20)

5.2.6 Example on the Moëbius basis

Consider the function $\mu(x) = 8 - x_1 + 5x_2 - x_1x_5 + 4x_3x_5 - 6x_2x_4x_5 + 2x_1x_2x_3x_4$. We would like to approximate the function μ of degree $n = 5$ at the level $k = 3$, i.e. at approximating by a linear function $g(x)$ the function $f(x) = f^{(k)}(x)$ obtained by truncation

$$f(x) = f^{(k)} = 8 - x_1 + 5x_2 - x_1x_5 + 4x_3x_5 - 6x_2x_4x_5$$

We are interested in comparing the Banzhaf coefficients and the k-Banzhaf coefficients.

Monomial f^T	Banzhaf approx. $g_T^{(\mu)}(x)$	k-Banzhaf approx. $g_T^{(f)}(x)$	n=5, k=3
	$-\frac{t-1}{2^t} + \frac{1}{2^{t-1}}\sum_{i\in T} x_i$	$-\left[(t-1)\frac{(k)_t}{(n)_t}\right] + \frac{(k-1)_{(t-1)}}{(n-1)_{(t-1)}}\sum_{i\in T} x_i$	t
x_1x_5	$-\frac{1}{4} + \frac{1}{2}(x_1 + x_5)$	$-\frac{3}{10} + \frac{1}{2}(x_1 + x_5)$	2
x_3x_5	$-\frac{1}{4} + \frac{1}{2}(x_3 + x_5)$	$-\frac{3}{10} + \frac{1}{2}(x_3 + x_5)$	2
$x_2x_4x_5$	$-\frac{1}{4} + \frac{1}{4}(x_2 + x_4 + x_5)$	$-\frac{2}{10} + \frac{1}{6}(x_2 + x_4 + x_5)$	3
$x_1x_2x_3x_4$	$-\frac{3}{16} + \frac{1}{8}(x_1 + x_2 + x_3 + x_4)$	n.a.	4

Table 5.1: Table of comparison between Banzhaf and k-Banzhaf coefficients

$$
\begin{aligned}
g^{Banzhaf}(x) = 8 - x_1 + 5x_2 &- \left[-\frac{1}{4} + \frac{1}{2}(x_1 + x_5)\right] + 4\left[-\frac{1}{4} + \frac{1}{2}(x_3 + x_4)\right]\\
&- 6\left[-\frac{1}{4} + \frac{1}{4}(x_2 + x_4 + x_5)\right] + 2\left[-\frac{3}{16} + \frac{1}{8}(x_1 + x_2 + x_{x3} + x_4)\right]\\
&= \frac{67}{8} - \frac{5}{4}x_1 + \frac{15}{4}x_2 + \frac{9}{4}x_3 - \frac{5}{4}x_4
\end{aligned}
$$

$$\begin{aligned}
g^{k-Banzhaf}(x) &= 8 - x_1 + 5x_2 - \left[-\frac{3}{10} + \frac{1}{2}(x_1 + x_5)\right] \\
&+ 4\left[-\frac{3}{10} + \frac{1}{2}(x_3 + x_5)\right] - 6\left[-\frac{2}{10} + \frac{1}{6}(x_2 + x_4 + x_5)\right] \\
&= 8 + \frac{3}{10} - \frac{12}{10} + \frac{12}{10} + x_1(-1 + \frac{1}{2}) + x_2(5 - 1) \\
&+ x_3(2) + x_4(-1) + x_5(-\frac{1}{2} + \frac{4}{2} - 1) \\
&= 8 + \frac{3}{10} - \frac{1}{2}x_1 + 4x_2 + 2x_3 - x_4 + \frac{1}{2}x_5
\end{aligned}$$

In terms of common denominators

$$g^{Banzhaf}(x) = const. - \frac{25}{20}x_1 + \frac{75}{20}x_2 + \frac{45}{20}x_3 + \frac{25}{20}x_4$$

$$g^{k-Banzhaf}(x) = const. - \frac{10}{20}x_1 + \frac{80}{20}x_2 + \frac{40}{20}x_3 - \frac{20}{20}x_4 + \frac{10}{20}x_5$$

The ordering of the coefficients has changed: in the first case we have

$$\beta_2 > \beta_3 > \beta_5 = 0 > \beta_1 > \beta_4$$

$$\beta_2 > \beta_3 > \beta_5 > 0 > \beta_1 >= \beta_4$$

In the first case there are two positive coefficients, in the second there are three positive coefficients.

5.2.7 Comparison with Shapley Value and Banzhaf Value

	Value of $\beta(t)$ in $\sum_{T\in\mathcal{Q}_i}\beta(t)a_T$	Weight of $\Delta_i(X)$ in $\sum_{X\in\mathcal{Q}_{-i}}\beta(t)\Delta_i(X)$
Shapley Value	$\frac{1}{t}$	$\frac{1}{n}\frac{1}{\binom{n-1}{t-1}}$
Banzhaf Value	$\frac{1}{2^{t-1}}$	$\frac{1}{2^{n-1}}$
k-Banzhaf	$\frac{\binom{n-t}{k-t}}{\binom{n-1}{k-1}} = \frac{(k-1)_{t-1}}{(n-1)_{t-1}}$	$\frac{1}{\binom{n-1}{k-1}}$

Table 5.2: Advantage of k-Banzhaf

The k-Banzhaf Value has the advantage that his computational cost is reduced. For each i, one has to compute $\binom{n-1}{k-1}$ differences, whereas for both the Banzhaf and the Shapley Value one has to compute $2^n - 1$ differences. Consider also that normally one would like to use a k considerably lower than $\frac{n}{2}$: this makes the evaluation in practice very advantageous. Moreover, several methods for Feature Selection use a greedy forward selection or a backward elimination approach, which implies computing Shapley or Banzhaf repeatedly (for different games) as the selected candidate set is increased/decreased, until a coalition of size k is found. Clearly, the k-Banzhaf is less expensive computationally, than those methods.

5.3 Implemented Algorithms

We implement some algorithms and compare them to k-Banzhaf. As power index-based algorithms, we have (Shapley Value and Banzhaf Value). As greedy-based algorithms, we have (Greedy Forward Selection and Greedy Backward Elimination).

5.3.1 For generation of random subsets

This is the procedure to generate a random subset, consider a set $N = \{0,1,...,n-1\}$, mapping a bit sequence, we consider an element i as part of the set if the corresponding bit position is set to 1, and that the element is not part of the set otherwise. We generate a sequence of n bits where each bit is set to 1 with probability $\frac{1}{2}$ and set to 0 with probability $\frac{1}{2}$, so probability of a sequence with n bits will be $\frac{1}{2^n}$ with 2^n equal the number of subsets for a set of n elements.

Algorithm 6: RandomSubset(n, k)

```
function : RandomSubset(n, k)
input    : $n$ the number of elements in the set
input    : $k$ the cardinality of the subset
output   : $S$ a random set of cardinality $k$

$S \leftarrow \emptyset$
$i \leftarrow$ RandomInteger($1, n$)
while $|S| \neq k$ do
    $r \leftarrow$ RandomReal()
    if $r \geq \frac{1}{2}$ then
        $S \leftarrow S \cup \{i\}$

    $i \leftarrow 1 + (i \mod n)$
end
return $S$
 end function
```

5.3.2 The Greedy approach

Greedy approaches are discussed in Section 3.1.3, they are those choosing the search path based on the direction that seems best at the time in order to achieve the best benefit immediately. This can be an effective strategy; in fact, it may show immediate benefits in a predictive performance that stall out at a locally best setting. A non-greedy approach unlikely would re-evaluate previous feature combinations and would have the ability to change direction, to move in a way that was initially unfavourable if it appears to have a potential benefit the current step. With this behavior the non-greedy approach would escape being trapped in a local optimum.

- **Sequential Forward Selection Algorithm:** It is the classic greedy forward selection. The algorithm selects the best feature to add to the set which will increase the score of the set by the maximum value (see Algorithm 5). His main limitation is that it is not possible to remove obsoleted features after the addition of the new features.

- **Sequential Backward Selection Algorithm:** It is the classic greedy backward elimination. Starting from the full set, the algorithm sequentially remove the feature which has minor importance, so that the score of the set will be reduce by the minimum value. His limitation is its impossibility/inability to reevaluate the usefulness of a discarded feature.

5.3.3 Power indexes

Regarding power indexes-based algorithms, the following approaches can be used:

1. By computing the power index for each feature once and ordering them in ascending or descending order. The exact score is not necessary and it is also expensive: It would need to analyze in the case of n features all the 2^n subsets for Banzhaf and all the $n!$ permutations for Shapley. Sampling is used in practice to approximate (Monte Carlo), the number of sets or permutations used depend on the goodness of the approximation.

2. Power index is in general, a weighted average of the marginal contributions. Hence computing the marginal contribution of each feature can allow us to order the features based on their partial value. By doing so, the order of the most important features become stable after a moment. So if the user is interested in some number of features let say k, it turns useful to compute the partial power index, just until the order of the first k features is stable. The order among the features is not so important in this case, just the stability of the whole set k is important.

- **Shapley Value Algorithm:** we use the same Shapley Value algorithm used in Chapter 4; Algorithm 2). So it remain Banzhaf Value and k-Banzhaf value to be discussed in the following section.

- **Banzhaf Value Agorithm:** uses the approach 1) with the Banzhaf Value as power index (see Algorithm 7)
- **K-Banzhaf Value Algorithm:** uses the k-Banzhaf Value as power index where k is the number of features to select (see Algorithm 8)

5.3.3.1 Banzhaf Index

As we have seen in Section 2.5 of Chapter 2, the Banzhaf index was introduced after the Shapley Value, the former index considers each distinct coalition in a set while the latter analyses all the possible permutations of the element s of the set. Here is the Banzhaf value-based algorithm (see Algorithm 7), his computation is based on subsets. So the approximation of the Banzhaf Value is done by uniformly sampling the subsets of the set of n elements. The user can choose the number of subset to use.

Algorithm 7: Banzhaf Value(μ, n, m)

```
function : Banzhaf ($\mu, n, m$)
input    : $\mu$ the reference metric definition
input    : $n$ the number of elements in the set
input    : $m$ the number of subsets
output   : the vector of Banzhaf Value
$v \leftarrow 0 \in \mathbb{R}^n$                    ▷ Vector of Banzhaf value
$c \leftarrow 0 \in \mathbb{Z}^n$                    ▷ Counter vertor
for $t \in [1, m]$ do
    $T \leftarrow$ RandomSubset($n$)                 ▷ Generating the random subsets in the range 1..n
    for $i \in T$ do
        $S \leftarrow T \setminus i$
        $v_i \leftarrow v_i + \mu(S \cup i) - \mu(S)$     ▷ Computing marginal contribution
        $c_i \leftarrow c_i + 1$
    end
end
$b \leftarrow 0 \in \mathbb{R}^n$
for $i \in [0, n-1]$ do
    $b_i \leftarrow v_i / c_i$                       ▷ Computing the normalized Banzhaf value
end
return $b$
 end function
```

5.3.3.2 k-Banzhaf Index

We develop the k-Banzhaf index from Banzhaf as shown in Section 5.2. We show in Section 5.2.7, that k-Banzhaf, with respect to the shapley Value and Banzhaf Value has reduced computational cost. The k-Banzhaf-based algorithm (see Algorithm 8), derive from the Banzhaf Value, the level k of the computation is chose by the user.

Algorithm 8: k-Banzhaf Value(μ,n,m,k)

```
function : k-Banzhaf ($\mu,n,m,k$)
input    : $\mu$ the reference metric definition
input    : $n$ the number of elements in the set
input    : $m$ the number of subsets
input    : $k$ the subset cardinality
output   : the vector of k-Banzhaf Value

$v \leftarrow 0 \in \mathbb{R}^n$                      ▷ Vector of k-Banzhaf value
$c \leftarrow 0 \in \mathbb{Z}^n$                      ▷ Counter verctor
for $t \in [1,m]$ do
    $T \leftarrow$ RandomSubset($n$, $k$)      ▷ Generatingg the random subsets in the range 1..k
    for $i \in T$ do
        $S \leftarrow T \setminus i$
        $v_i \leftarrow v_i + \mu(S \cup i) - \mu(S)$      ▷ Computing marginal contribution
        $c_i \leftarrow c_i + 1$
    end
end
$b \leftarrow 0 \in \mathbb{R}^n$
for $i \in [1,n]$ do
    $b_i \leftarrow v_i / c_i$                          ▷ Computing the normalized k-Banzhaf value
end
return $b$
 end function
```

5.4 Power indexes-based feature selection applied to self-training

In this section, we compare the performance metric (namely the accuracy of many (five) feature selection-based methods in a classification process using the same dataset used in the previous experiment. The feature selection methods are: (*Greedy Forward Selection (GFS), Greedy Backward Elimination (GBE), Shapley Value (Shap), Banzhaf Value (Banz) and k-Banzhaf Value (k-Banz)*). To achieve this goal, we build three classification scenarios, in each of them, we compare the five feature selection methods also with a benchmark method; a "no feature selection" method (noFS). In the following section, we begin by the experimental protocol.

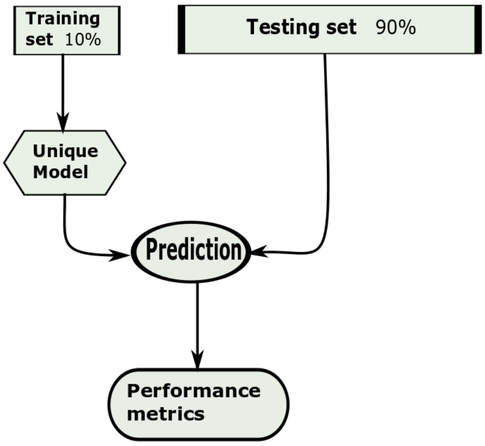

Figure 5.1: No Feature Selection process

5.4.1 Experimentation protocol and classification methods

In this section, we describe the entire process, for each of our three scenarios *(the simple training and testing process, the bootstrap process, and the self-training process)*, we provide step by step, information regarding the experimental protocol. Also, we provide information about the classification methods or feature selection methods we use in the experimentation.

5.4.1.1 Experimentation protocol

Our experiment compares three main scenarios, in each of these scenarios, we split, in the same manner, the data set; for more details on the dataset, (see section 4.4.1). The dataset splitting process has two variants: in the first variant, we split the data set (DS) respectively into 10% for the training set and 90% for the testing set, in the second variant we split it into 5% for the training set and 95% for the testing set. Each of the three scenarios implements six different sub-scenarios divided mainly into two groups: the first group (No Feature selection) is made by only the first sub-scenario, and it is also the benchmark, the second group (Feature Selection) is made by the remaining five sub-scenarios. In the first group (like the name indicates) we do not apply feature selection as opposed to the second group in which we apply different feature selection algorithms to chose the best features which optimize the learning process. In the following, we present the description of the scenarios followed by the description of the sub-scenarios:

- **The first scenario** : Simple training and testing process (1)
  In the first group of the sub-scenario (No Feature Selection), we leave out the feature selection step, so we just split the data set (DS) respecting the indicated proportion, build a model using the selected classifier, predict the labels on the testing set and compute the performance metrics.

In the case of the second group of the sub-scenarios (Feature Selection), we use the corresponding algorithm/method to choose the most critical features, (as we said before, each sub-scenario uses his own algorithm). We use those features to create a new version of the dataset containing only the most important features (DS-v) then split it (DS-v1) into the training set (TS, TS=P1) and the testing set (Testset) observing the proportions indicated in the previous section, e.g.: (TS=10% and Testset = 90%). Then we build a model on the training set with one of the selected classifiers (Random Forest, Linear SVM), and use that model to predict the labels on the test set. In the end, we compute for all the sub-scenarios the performance metrics (in this case the accuracy).

- **The second scenario** : Simple bootstrap process (2)

Definition 36 (Bootstrap)**.** *The bootstrap process is characterized by the successive reinsertion of all the predicted labels (and their corresponding records) into the previous training set to form the next training set and then build the next model (see Figure 5.2).*

This scenario consists of: using the corresponding algorithm to chose the most important features and created with those features a new version of the data set (DS-v) then split the new version data set into the training and testing set according to the proportions indicates before. After splitting the dataset as in the first scenario, in this second scenario we split again the testing set part (Testset) into four parts/chunks (P2, P3, P4, P5) so that $P_i \cap P_j = \emptyset$ and P1 ∪ P2 ∪ P3 ∪ P4 ∪ P5 = DS. Differently, with respect to the first scenario, in this scenario, we make four predictions using four different models. This is how we construct the models: the first model (*model1*) is built on the initial training set (P1), using the same classifier as in the first scenario, then we use the model1 to predict the labels on the first part of the testing set (P2), the predicted labels and their corresponding records are appended to the initial training set to form the training set 2 (TS2). This TS2 is used to build the *model2* following the same process. We use the model2 to predict the labels of P3. The predicted labels on the P3 will be appended with their corresponding records to the TS2 to form the training set 3 (TS3), the TS3 will be used to build the *model3*, this model3 is used to predict the labels on third part (P4) and so on, until the *model4* which is used to predict labels on the last part (P5). In the end, we compute for all the sub-scenarios the chosen performance metrics (in this case, the accuracy).

- **The third scenario** : Self-training process (3)

Definition 37 (Self-training)**.** *Self-training as define by Ningam in [67] "is an incremental algorithm that initially builds a single classifier using a small amount of labelled data. Then it iteratively predicts the labels of the unlabelled examples, rank the examples by*

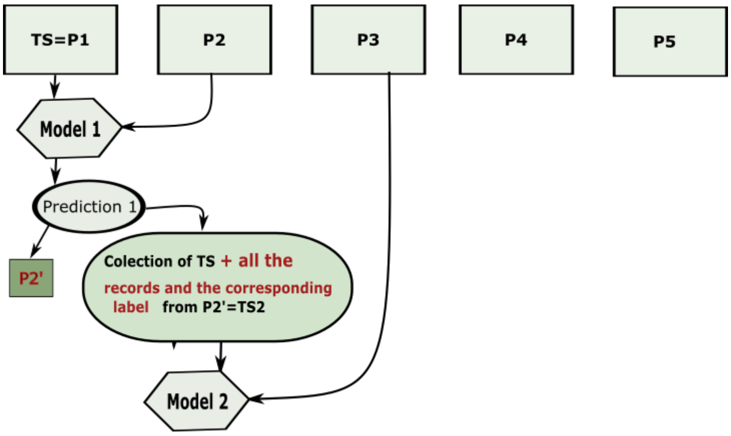
TS=P1
P2
P3
P4
P5
Model 1
Prediction 1
P2'
Colection of TS + all the records and the corresponding label from P2'=TS2
Model 2

Figure 5.2: A step of the Bootstrap scenario

the confidence in their prediction and permanently add the most confident examples into the labelled training set. It retrains the underlying classifier with the augmented training set and the process is repeated for a given number of iterations or until some heuristic convergence criterion is satisfied".

Only in the case the initial and subsequent classifiers correctly label most of the unlabelled examples, then the classification accuracy can be improved over iterations. Unfortunately, adding mislabeling noise is not avoidable. When self-training is applied on linear classifier such as SVM, one drawback is that the majority confident examples often lie away from the target decision boundary (non-informative examples).

This scenario is quite similar to the second, the unique difference is that while in the second scenario the predicted labels are all appended to the previous training set with their corresponding records to form the new training set, in this scenario, only the labels predicted with a certain confidence threshold and their corresponding records are appended to the previous training set (In this specific case, base on the experiments, we notice that the better threshold confidence is 80%. So we chose the labels with a confidence greater than 80%) (see Figure 3.7). More details in Section 5.4.1.1.

5.4.1.2 Classification methods

As we said before, each of the above mentioned scenarios implements the same six methods or sub-scenarios : No feature selection (No-FS), Greedy Forward Search-based Feature Selection (GFS), Greedy Backward Elimination-based Feature Selection (GBE), Shapley Value-based Feature Selection (Shap), Banzhaf Index-based Feature Selection (Banz), Restricted-Banzhaf Index-based Feature Selection (k-Banz) let have a look at each of the sub-scenario.

- **No Feature Selection** (1-0, 2-0, 3-0):

This method uses all the 288 features of the dataset (DS). After splitting the DS into the training set (TS) and testing set (Test-set) it uses respectively the Random Forest Classifier (RFC) and the Linear Support Vector Machine Classifier (SVM) to build a model on the training set make predictions on the testing set and compute all the the chosen performance metrics (in this case, the accuracy).

- **Greedy Forward Search-based Feature Selection** (1-GFS, 2-GFS, 3-GFS):
  This classification method uses Greedy Forward Search (GFS) to select the 150 most essential features, it builds a new data set version (DS-v1) using only the selected features splits this latest data set according to the indicated proportions so that we have P1 as the training set, (only for the scenarios 2 and 3) the remaining testing set is then split equally into four parts P2, P3, P4, P5. The classification process uses these training and testing parts to build models and then predict the labels.

- **Greedy Backward Elimination-based Feature Selection**(1-GBE, 2-GBE, 3-GBE):
  The feature selection algorithm used in this case is the Greedy Backward Elimination (GBE). Therefore, based on this greedy selection method we select the most 150 essential features and build with them a new data set version (DS-v2) and using only the selected features, splits this latest data set according to the indicated proportions so that we have P1 as the training set, (only for the scenarios 2 and 3) the remaining testing set is then split equally into four parts P2, P3, P4, P5. The classification process uses these training and testing parts to build models and then predict the labels.

- **Shapley Value-based Feature Selection**(1-Shap, 2-Shap, 3-Shap):
  In this sub-scenario, the feature selection method used is the Shapley Value. Consequently, based on the Shapley Value algorithm we select the 150 most important features and use it to build a new data set version (DS-v3) using only those critical features. The new version of the data set is then split according to the indicated proportions so that we have P1 as the training set, (only for the scenarios 2 and 3) the remaining testing set is split equally into four parts P2, P3, P4, P5. The classification process uses these training and testing parts to build models and then predict the labels.

- **Banzhaf Index-based Feature Selection**(1-Banz, 2-Banz, 3-Banz):
  Using the Banzhaf Index algorithm in this method, we select the most 150 essential features and use them to build a new version of our data set (DS-v4). The new version of data set is then split according to the indicated proportions so that we have P1 as the training set, (only for the scenarios 2 and 3) the remaining testing set is then split equally into four parts P2, P3, P4, P5. The classification process uses these training and testing parts to build models and then predict the labels.

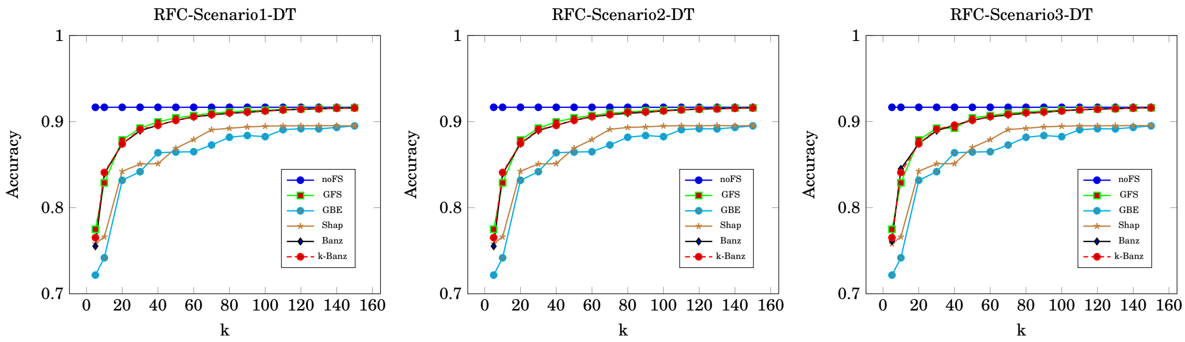

Figure 5.3: Performance of the top-k rules using "Random forest" as ML algorithm and "Decision-Tree" as splitting criteria

- **Restricted-Banzhaf Index-based Feature Selection**(1-k-Banz, 2-k-Banz, 3-k-Banz): Like in the method using the Banzhaf index, this latter method uses the restricted Banzhaf index algorithm to select the 150 most essential features. The process of building a new version of the data set (DS-v5) is the same as in the previous sub-scenarios. The new version of data set is split according to the indicated proportions so that we have P1 as the training set, (only for the scenarios 2 and 3) the remaining testing set is then split equally into four parts P2, P3, P4, P5. The classification process uses these training and testing parts to build models and then predict the labels.

5.4.2 Results with Random Forest Classifier (RFC) as ML algorithm

In this section, we use Random Forest Classifier (RFC) to build, train and test the model from our dataset, then compute the performance metrics, namely the accuracy.

5.4.2.1 Results using "RFC" as ML algorithm and "Decision Tree" as splitting criteria

Here we have the performance of the top-k rules using Decision Tree classifier and using as rule ranking metric the k-Banzhaf value with respect to the same metric computed on the pool(k-Banz) (red lines), the Banzhaf value with respect to the same metric computed on the pool (Banz)(black lines), Shapley value with respect to the same metric computed on the pool (Shap) (brown lines), the Greedy Backward Elimination with respect to the same metric computed on the pool (GBE) (cyan lines), the Greedy Forward Selection computed on the pool (GFS) (green lines with square), and the benchmark with no feature selection (noFS) (blue lines). From the left to the right respectively: the comparison of the ranking metric with respect to (k-Banzhaf, Banzhaf, Shapley, GFS, GBE and noFS) in term of assessment metric into the first, the second and the third scenario, top-k classifier accuracy.

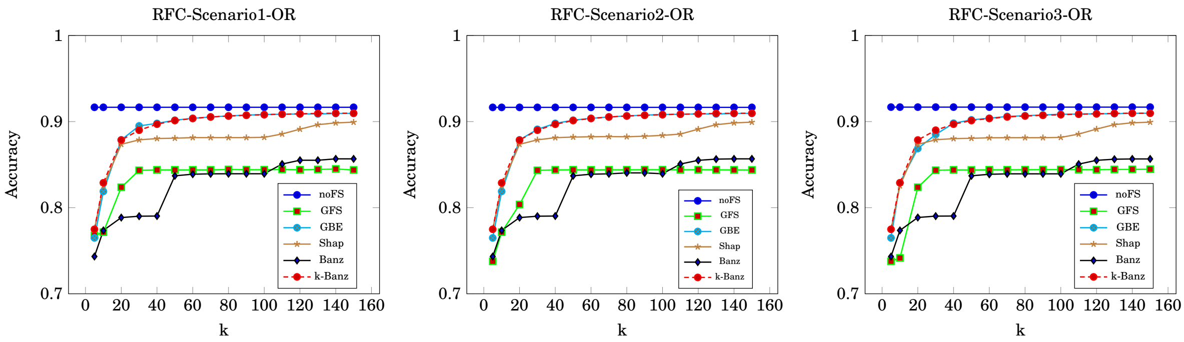

Figure 5.4: Performance of the top-k rules using Random forest as ML algorithm and "OR operator" as splitting criteria

5.4.2.2 Result using "RFC" as ML algorithm and "OR-operator" as splitting criteria

Here we have the performance of the top-k rules using the "OR classifier" classifier and using as rule ranking metric the k-Banzhaf value with respect to the same metric computed on the pool(k-Banz) (red lines), the Banzhaf value with respect to the same metric computed on the pool (Banz)(black lines), Shapley value with respect to same metric computed on the pool (Shap) (brown lines), the Greedy Backward Elimination with respect to the same metric computed on the pool (GBE) (cyan lines), the Greedy Forward Selection computed on the pool (GFS) (green lines with square), and the benchmark with no feature selection (noFS) (blue lines). From the left to the right respectively: the comparison of the ranking metric with respect to (k-Banzhaf, Banzhaf, Shapley, GFS, GBE and noFS) in term of assessment metric into the first, the second and the third scenario, top-k classifier accuracy.

5.4.3 Results with support-vector machine (SVM) as ML algorithm

In this section, we use a support-vector machines to build, train and test the model from our dataset, then compute the performance metrics, namely the accuracy.

5.4.3.1 Result using "SVM" as ML algorithm and "OR-operator" as splitting criteria

Here we have the performance of the top-k rules using the "OR operator" and using as rule ranking metric the k-Banzhaf value with respect to the same metric computed on the pool(k-Banz) (red lines), the Banzhaf value with respect to the same metric computed on the pool (Banz)(black lines), Shapley value with respect to the same metric computed on the pool (Shap) (brown lines), the Greedy Backward Elimination with respect to same metric computed on the pool (GBE) (cyan lines), the Greedy Forward Selection computed on the pool (GFS) (green lines with square), and the benchmark with no feature selection (noFS) (blue lines). From the left to

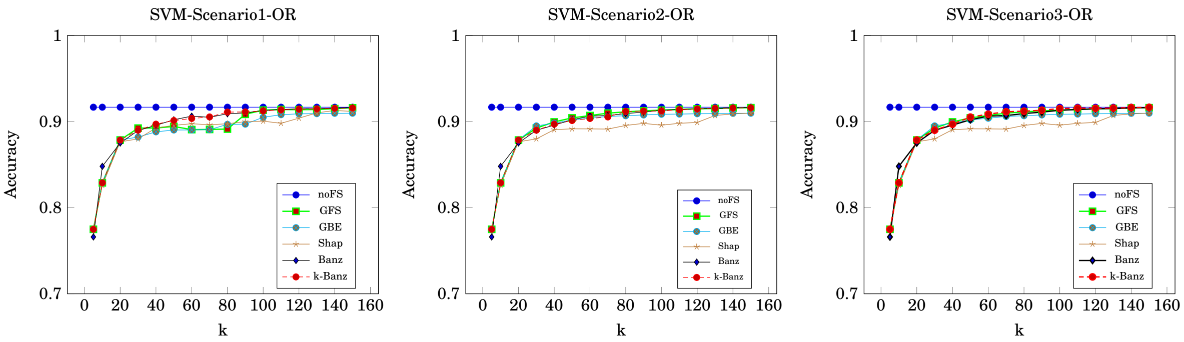

Figure 5.5: Performance of the top-k rules using "SVM" as ML algorithm and "OR operator" as splitting criteria

the right respectively: the comparison of the ranking metric with respect to (k-Banzhaf, Banzhaf, Shapley, GFS, GBE and noFS) in term of assessment metric into the first, the second and the third scenario, top-k classifier accuracy.

5.4.3.2 Result using "SVM" as ML algorithm and "Decision Tree" as splitting criteria

Here we have the performance of the top-k rules using Decision Tree classifier and using as rule ranking metric the k-Banzhaf value with respect to the same metric computed on the pool(k-Banz) (red lines), the Banzhaf value with respect to the same metric computed on the pool (Banz)(black lines), Shapley value with respect to the same metric computed on the pool (Shap) (brown lines), the Greedy Backward Elimination with respect to the same metric computed on the pool (GBE) (cyan lines), the Greedy Forward Selection computed on the pool (GFS) (green lines), and the benchmark with no feature selection (noFS) (blue lines). From the left to the right respectively: the comparison of the ranking metric with respect to (k-Banzhaf, Banzhaf, Shapley, GFS, GBE and noFS) in term of assessment metric into the first, the second and the third scenario, top-k classifier accuracy.

5.5 Interpretations and Conclusions

In Section 5.2 we presented the approximation of the Banzhaf Value, then from its properties, we proposed and developed a new version called the *restricted Banzhaf index (k-Banzhaf)*. We compared the latter with the original one and also with the Shapley Value. From this comparison, we showed that the restricted Banzhaf value is better with respect to the Banzhaf Value and Shapley. In Section 5.4, we used all the three power indexes to build power indexes-based feature selection and compare them to the greedy-based feature selection (Greedy Forward Selection

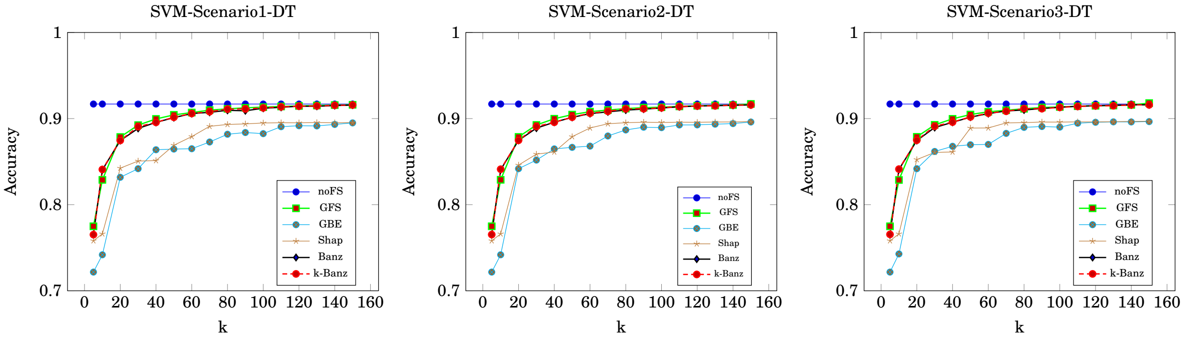

Figure 5.6: Performance of the top-k rules using "SVM" as ML algorithm and "Decision Tree" as splitting criteria

(GFS) and Greedy Backward Elimination (GBE)). We used as a benchmark a model without feature selection. We made all this using three different scenarios: in the first scenarios we train and test our model, in the second scenario we included a bootstrap process, and, in the third scenario we include a self-training process. We used the same dataset from the first part of the dissertation (see Section 4.4.1). We implemented all this using two machine learning algorithms: the Random forest classifier (RFC) and the support vector machine (SVM).

From the results we can see that there is almost no difference between the two machine learning algorithms, apart from that, with the RFC, the performance metrics of the GFS and Banzhaf Value are a little bit lower with respect to the others whereas with SVM it is the GBE and Shapley which fare a little bit less than the others. From our purpose to see how the different methods of Feature Selection influence the accuracy, we observed that in most cases, the accuracy obtained with k-Banzhaf is among the best results. On the other hand, we have seen in 2.5 that the time computation of k-Banzhaf index is more advantageous with respect to the Shapley and the Banzhaf index. So we can conclude that it is better to use restricted Banzhaf than index Banzhaf index.

5.6 Summary

In this chapter, we analyzed another power index and its properties; the *Banzhaf index*, and propose a new version called the **restricted Banzhaf Index (k-Banzhaf)**; developed from the original one (see Section 2.5). This new version outperforms concerning the normal one in term of computation time and has comparable results. Indeed, given a set $\mathcal{N}$ of N elements with $|\mathcal{N}| = N$, while the normal Banzhaf computes $2^N - 1$ differences, the k-Banzhaf computes only $\binom{n-1}{k-1}$ differences. We implemented a self-training process (a kind of bootstrap) to reinforce the learning process in a Machine Learning algorithm (Random Forest Classifier or Support Vector

Machines), (see Section 5.4.1.1). We use all the following components (Shapley Value, Banzhaf Value, k-Banzhaf Value, Self-training) in a classification process in which we build different features selections algorithms (power index-based features selection) and compare them to greedy-based feature selection (see 5.4.1.2). On the other part, we compare the performance metric, namely the accuracy of each features selection method (Shapley Value-based, Banzhaf Value-based, k-Banzhaf Value-based, Greedy forward selection-based, Greedy backward elimination-based) in the self-training process. The experimentation dataset is the one used in the first part of the dissertation.We presented the results in Section 5.4.2 and Section 5.4.3. We observed that power indexes-based feature selection methods do allow to rank the rules and to select the top-k rules, in analogy to feature selection and achieve performance comparable to other feature selection techniques. Moreover, such power indexes can be interpreted as a summary score of the usefulness of the rule that can be used to assess the rules individually during the periodic rule assessment process. The implemented self-training has a comparable result with respect to the bootstrap process and even with respect to a simple training and testing process. So the improvement in the case of the self-training is not significant. But, observed that the performance of our restricted Banzhaf index is comparable to other power indexes and in some cases, it is even better.

Part IV

Conclusion and Future Work

Chapter 6

Conclusion

Joined to the conclusion of this dissertation, let start by summarizing his content and his workflow. In the first chapter, we started observing that financial institutions are facing an ever-growing presence of credit card payment fraudulent activities. Underlining that there already exist automatic fraud detection systems to tackle such activities, we first describe the fraud detection and the rule management process used by our industrial partner (Atos Worldline). We then aimed at proposing an approach which improves the fraud detection systems based on our use case, and using a power index, a concept inspired by Coalitional Game Theory (CGT). Finally, we introduce some concepts as feature selection and self-training, to be used in the following parts of the thesis. In chapters two and three, we presented the related work; briefly, in chapter two, we introduce the game theoretic-concepts, going from definitions to properties of games theory in general, we analyzed in particular coalitional games with transferable utility. Then, we focus on the two power indexes (Shapley Value and Banzhaf Value) explored in the thesis. In chapter three, we discussed some feature selection methods, insisting on greedy methods used as a benchmark in our experiments. We discussed also some machine learning techniques, especially semi-supervised machine learning, from which we implemented a self-training process. In chapters four and five, we presented two experiments using power indexes in the context of fraud detection and based on a real-world dataset. We also propose in chapter five, a new power index (the restricted Banzhaf value (k-Banzhaf)), derived from the Banzhaf Value. Chapter six concludes the dissertation. Regarding the contributions of the thesis, we summarized them in the following paragraphs.

6.1 Summary

Reducing the number of operational rules

In the first experiment based on a real-world dataset made available by our industrial partner containing approximately 300 rules and $N = 359,862$ observations, we conducted some experiments in which we implemented a novel approach (*the Shapley Value-based approach*). This approach proposes a rule evaluation and selection procedure that quantifies the performance of the rule "in collaboration" with the others, taking into account possible redundancy. It brings us to achieve the goal of reducing the number of operational rules from about 300 rules to about 30, by achieving the same result in terms of precision and recall. We showed that our proposed approach fares better than the traditional one; more specifically, we prove that the rule-pool-based approach − assessing the rules based on their contribution to the performance of the OR-ed pool, quantified by the Shapley Value − fares better than the traditional individual rule-based one. The Shapley Value ranking lightens and makes more effective the rule management process.

Assigning a score to the individual rule

The score is useful in the monthly revision process, where the performance of each rule is considered by the rule managers that have to make the decision keep/drop. So, the Shapley Value score provides a synthesis of the rule usefulness. We showed that our approach is comparable to other feature selection techniques, but with the advantage that the assignment of the normalized score to the rule distinguishes the proposed approach (to rule management) from other approaches, used for feature selection. For instance, the Greedy Forward Selection method (used as the benchmark) can provide good coalitions rules, but it does not assign a score to the rule, summarizing its performance in collaboration. Furthermore, in principle, the application of the Greedy Forward Selection during the periodical assessment could allow important changes in the composition of the pool, while the Shapley Value score supports a fine-grained control over the composition of such pool.

The restricted Banzhaf Value

In the second experiment, we studied another power index, the *Banzhaf index*; then we proposed a modified version called the **restricted Banzhaf Index (k-Banzhaf)**. The new version proposed outperforms the normal one in term of computation time and has comparable performance metrics. While for a set $\mathcal{N}$ of N elements with $|\mathcal{N}| = N$, the normal Banzhaf computes $2^N - 1$ differences, the k-Banzhaf computes only $\binom{n-1}{k-1}$ differences. The Banzhaf index and the k-Banzhaf Value are then implemented in the experiment to be compared not only with the Shapley index but also with two greedy approaches: the greedy forward selection and the greedy backward elimination. We included the k-Banzhaf index in three scenarios of which two feature selection scenarios and one without feature selection, each scenario comparing five algorithms (Shapley, Banzhaf, k-Banzhaf, GFS, and GBE). We observed that k-Banzhaf is always among the best results.

For the same experiment, we implement also a self-training process (a kind of bootstrap process) to reinforce the learning process in two Machine Learning algorithms (Random Forest Classifier or Support Vector Machines). We use all these components (Shapley Value, Banzhaf Value, k-Banzhaf Value, Self-training) in a classification process in which we build different features selections algorithms (power index-based features selection) and compare them with greedy-based feature selection on the base of their performance metrics. In the same time, we compare the score of these feature selection methods in the self-training process.

The experimentation dataset is the one used in the first part of the dissertation. We observe that power indexes-based feature selection methods do allow to rank the rules and to select the top-k rules, in analogy to feature selection and achieve performance comparable to other feature selection techniques. Moreover, such power indexes can be interpreted as a summary score of the usefulness of the rule. In particular, we observe that the performance of our restricted Banzhaf index is comparable to other power indexes and that some times it is even better.

6.1.1 First contribution

In detail, regarding the first part of the thesis, we investigated the fraud detection process, particularly on the near-real-time fraud detection scenario. In that scenario, the system enacts ex-post evaluation based on broader information context with respect to the real-time scenario, including linked data. Afterwards, the transactions triggered by the systems are then presented to human investigators for final assessment. The investigators first chose among the submitted transactions those to investigate and decide whether the transaction is fraudulent or legitimate. Cards corresponding to fraudulent transactions are blocked.

We focused on supporting the efficiency of the process of rule management, observing that managers consider the performance of the rule as if it were run in isolation. We proposed a rule evaluation and selection procedure that quantifies the performance of a rule in collaboration with the others, taking into account possible redundancy. The contribution of the rule to the rule pool cannot be quantified correctly through the simple marginal contribution of the rule to the pool because the performance is non-additive.

We mapped the problem of selecting the set of rules operational at a given time for flagging a transaction as suspicious into a version of Feature Selection problem. Applying a state-of-the-art feature selection algorithm, (in this case Greedy Forward Selection) we showed that our method has comparable performance. Also, we argue that our method has the advantage that it associates a normalized score to the individual "feature"/rule whereas typically, feature selection algorithms focus on providing an optimal feature set and do not yield normalized importance scores.

6.1.2 Second Contribution

The second part of the thesis is devised into two main sections: in the first one, the leading power indexes (Shapley Value and Banzhaf Value) are discussed, in particular, the Banzhaf Value is analyzed through an analytical approach. From the properties of the latter, we deduced a new power index which is an improved version of the Banzhaf. We called the new version the *(restricted Banzhaf index (k-Banzhaf))*. In this section, we showed that our improved version fares at least as other power indexes methods (Shapley and Banzhaf index) but also as reference methods used as a benchmark (greedy methods). In terms of computation time, the k-Banzhaf has the advantage to be faster than the two other power indexes.

The second section is about using power indexes to build the power indexes-based features selection algorithms and compare them with the greedy-based feature selection, using the same dataset we used in the first part of the dissertation. For this second experiment, we implemented five feature selection methods (three power index-based and two greedy-based methods). Then we build three scenarios: the first one is a simple scenario (two steps) in which we train a model on the training set, predict on testing set and compute the accuracy score (see Figure 5.1); we called this first scenario **(No feature selection)**. The second scenario, called **Simple Bootstrap** (see Figure 5.2) consist of building many models (four), reinserting the prediction obtained into the training set. The third scenario, the **Self-training** (see Figure 3.7) scenario is like the second but characterized by the fact that only predictions which confidence is more than 80% are reinserted into the training set. We observed from the results that the classification performances are comparable in all the three scenarios; so there is no significant difference between the self-training scenario and the other two scenarios. This can be due to the quality of our dataset, on the other hand, an important observation is that the performance of our proposed power index is always among the best one. So the k-Banzhaf -based feature selection is not only comparable to the other power index-based feature selection, but also it is comparable to the traditional feature selection method (greedy-based methods), and some times it even fares better.

6.2 Future improvements

6.2.1 Limitations

The well-known advantage of the Shapley Value over solution concepts such as kernel and core is that the solution provided by Shapley Value is both fair and unique: the former relates to the way the gains from cooperation are distributed to the coalition members. The latter is quite clear (no ambiguity) for a game, there is only one possible solution, and players know what is the payoff of the game if they play. Despite the uniqueness and the fairness of the Shapley Value, some drawbacks are known; the main ones are: first, for certain classes of games (the weighted voting game), the problem of determining the Shapley Value is # P-complete as shown by X. Deng and C. Papadimitriou in [24]. The class of # P-complete problems are those that, although nobody has been able to prove, they seem intractable[1]. This means computing the Shapley Value for the voting game will be in general intractable. As second drawback, Shapley Value provides a solution with a limited degree of certainty like shown by A. Roth in [28], the uniqueness characteristic of Shapley Value is associated with the risk neutrality. Moreover, Shapley Value is an entailment of the normalization of the utility function used to define the considered game. Therefore, this uncertainty provides an additional dimension for evaluating a player.

6.2.2 Possible solutions

We have seen that the computation of Shapley is at least # P-complete, as well as the Banzhaf index from computational perspective as proved in [73]. Given the computational hardness of finding the exact value of Shapley Value and Banzhaf Value, researchers have developed some approximation methods to overcome the problem as introduced in Section 2.4.1. These methods include the Monte Carlo simulation method [29], the multilinear extension (MLE) method [37], the modified MLE method [62, 68], the random permutation method [101]. Most of these methods are efficient for majority voting games and they vary in terms of their approximation errors and their time complexities. The aim can be to find methods that minimize approximation errors and time complexities in the already existing methods; or finding new efficient methods that fulfill these expectations. For example it may be possible to find good deterministic methods or even good approximation algorithms to achieve better quality of computed values.

[1] A problem is said to be intractable if it takes unreasonably long to solve it. This is the case, for example, if the time taken to solve it increases exponentially in the size of the input to the problem.

APPENDIX A

Appendix A

Main notation used in the dissertation

S	A coalitional structure
C	A coalition
μ	The characteristic function , Performance metrics
$\mu(C)$	A value of the coalition, where μ is the characteristic function
$Shap$	The Shapley Value
$\mathscr{N}$	A set
N	The cardinality of the set $\mathscr{N}$: $N = \lvert\mathscr{N}\rvert$
$2^{\mathscr{N}}$	The power set of $\mathscr{N}$
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integers
Π	A set of permutation
π	A permutation
Θ	A subset of Π
θ	The cardinality of Θ, $\theta = \lvert\Theta\rvert$
$\mathscr{P}\mathscr{B}^n$	A Hilbert space (Space of psoeudo-boolean function on N)
$\Delta_i^{[\mu]}(\mathscr{C})$	The marginal contribution of a rule i to a rule coalition w.r.t the measure μ

BIBLIOGRAPHY

[1] https://worldline.com/.

[2] *http://www.lamsade.dauphine.fr/ airiau/teaching/index.html.*

[3] *Copyright*, in Generatingfunctionology, H. S. Wilf, ed., Academic Press, 1990, p. iv.

[4] R. J. AUMANN AND J. H. DREZE, *Cooperative games with coalition structures*, International Journal of Game Theory, 3 (1974), pp. 217–237.

[5] W. AWADA, T. KHOSHGOFTAAR, D. DITTMAN, R. WALD, AND A. NAPOLITANO, *A review of the stability of feature selection techniques for bioinformatics data*, 08 2012, pp. 356–363.

[6] B. AZHAGUSUNDARI AND A. S. THANAMANI, *Feature selection based on information gain*, in International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN, pp. 2278–3075.

[7] S. B. COHEN, E. RUPPIN, AND G. DROR, *Feature selection based on the shapley value.*, 01 2005, pp. 665–670.

[8] Y. BACHRACH, E. MARKAKIS, E. RESNICK, A. D. PROCACCIA, J. S. ROSENSCHEIN, AND A. SABERI, *Approximating power indices: theoretical and empirical analysis*, Autonomous Agents and Multi-Agent Systems, 20 (2010), pp. 105–122.

[9] A. C. BAHNSEN, D. AOUADA, A. STOJANOVIC, AND B. OTTERSTEN, *Feature engineering strategies for credit card fraud detection*, Expert Systems with Applications, (2016).

[10] J. BANZHAF, *Weighted voting doesn't work: A mathematical analysis*, Rutgers Law Review, 19 (1965), pp. 317–343.

[11] S. BHATTACHARYYA, S. JHA, K. THARAKUNNEL, AND J. C. WESTLAND, *Data mining for credit card fraud: A comparative study*, Decision Support Systems, 50 (2011), pp. 602–613.

[12] U. M. BRAGA-NETO, R. F. HASHIMOTO, E. R. DOUGHERTY, D. V. NGUYEN, AND R. J. CARROLL, *Is cross-validation better than resubstitution for ranking genes?*, Bioinformatics, 20 2 (2004), pp. 253–8.

[13] O. Caelen, G. Gianini, and E. Damiani, *Fr3065558a1, system and method to manage the detection of fraud in a system of financial transactions*.

[14] F. Carcillo, Y.-A. L. Borgne, O. Caelen, Y. Kessaci, F. Oblé, and G. Bontempi, *Combining unsupervised and supervised learning in credit card fraud detection*, Information Sciences, (2019).

[15] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi, *Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization*, International Journal of Data Science and Analytics, 5 (2018), pp. 285–300.

[16] R. Caruana and D. Freitag, *Greedy attribute selection*, in Machine Learning Proceedings 1994, W. W. Cohen and H. Hirsh, eds., Morgan Kaufmann, San Francisco (CA), 1994, pp. 28 – 36.

[17] S. Cohen, E. Ruppin, and G. Dror, *Feature selection based on the shapley value*, in Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI'05, San Francisco, CA, USA, 2005, Morgan Kaufmann Publishers Inc., pp. 665–670.

[18] S. B. Cohen, G. Dror, and E. Ruppin, *Feature selection via coalitional game theory*, Neural Computation, 19 (2007), pp. 1939–1961.

[19] J. S. Coleman, *Control of Collectivities and the Power of a Collectivity to Act..*, 1968.

[20] V. Conitzer and T. Sandholm, *Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains.*, 01 2004, pp. 219–225.

[21] F. G. Cozman, I. Cohen, and M. C. Cirelo, *Semi-supervised learning of mixture models*, in Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML'03, AAAI Press, 2003, pp. 99–106.

[22] M. V. Culp and G. Michailidis, *An iterative algorithm for extending learners to a semi-supervised setting*, 2008.

[23] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, *Credit card fraud detection: a realistic modeling and a novel learning strategy*, IEEE transactions on neural networks and learning systems, 29 (2017), pp. 3784–3797.

[24] X. Deng and C.-t. H. Papadimitriou, *On the complexity of cooperative solution concepts*, Mathematics of Operations Research - MOR, 19 (1994), pp. 257–266.

[25] J. Doak, *An evaluation of feature selection methods and their application to computer security*, 1992.

[26] J. S. DREYER AND A. SCHOTTER, *Power relationships in the international monetary fund: The consequences of quota changes*, The Review of Economics and Statistics, 62 (1980), pp. 97–106.

[27] J. G. DY AND C. E. BRODLEY, *Feature selection for unsupervised learning*, J. Mach. Learn. Res., 5 (2004), pp. 845–889.

[28] A. E. ROTH, *The expected utility of playing a game*, (1988).

[29] S. S. FATIMA, M. WOOLDRIDGE, AND N. R. JENNINGS, *A linear approximation method for the shapley value*, Artificial Intelligence, 172 (2008), pp. 1673 – 1699.

[30] A. FRÉCHETTE, L. KOTTHOFF, T. MICHALAK, T. RAHWAN, H. H. HOOS, AND K. LEYTON-BROWN, *Using the shapley value to analyze algorithm portfolios*, in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, AAAI Press, 2016, pp. 3397–3403.

[31] I. GHEYAS AND L. SMITH, *Feature subset selection in large dimensionality domains*, Pattern Recognition, 43 (2010), pp. 5–13.

[32] D. GILLIES, *Some Theorems on n-Person Games*, PhD thesis, Princeton University, 1953.

[33] F. GLOVER, *Tabu search—part i*, ORSA Journal on Computing, 1 (1989), pp. 190–206.

[34] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st ed., 1989.

[35] S. GORE AND V. GOVINDARAJU, *Feature selection using cooperative game theory and relief algorithm*, in Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions, A. M. Skulimowski and J. Kacprzyk, eds., Cham, 2016, Springer International Publishing, pp. 401–412.

[36] M. GRABISCH, *Set Functions, Games and Capacities in Decision Making*, Springer Publishing Company, Incorporated, 1st ed., 2016.

[37] O. GUILLERMO, *Game theory*, Emerald Group Publishing Limited, 3rd ed ed., october 1995. Original copyright 1995.

[38] I. GUYON AND A. ELISSEEFF, *An introduction to variable and feature selection*, J. Mach. Learn. Res., 3 (2003), pp. 1157–1182.

[39] G. HAFFARI AND A. SARKAR, *Analysis of semi-supervised learning with the yarowsky algorithm*, in Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, UAI'07, Arlington, Virginia, United States, 2007, AUAI Press, pp. 159–166.

[40] P. HAMMER AND S. RUDEANU, *Boolean methods in operations research and related areas*, Ökonometrie und Unternehmensforschung, Springer-Verlag, 1968.

[41] P. L. HAMMER AND R. HOLZMAN, *Approximations of pseudo-boolean functions; applications to game theory*, ZOR - Meth. Mod. of OR, 36 (1992), pp. 3–21.

[42] T. HASTIE, R. TIBSHIRANI, AND J. FRIEDMAN, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[43] X. HE, D. CAI, AND P. NIYOGI, *Laplacian score for feature selection*, in Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS'05, Cambridge, MA, USA, 2005, MIT Press, pp. 507–514.

[44] A. E. HOERL AND R. W. KENNARD, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 12 (1970), pp. 55–67.

[45] M. J. HOLLER, *Forming coalitions and measuring voting power*, Political Studies, 30 (1982), pp. 262–271.

[46] S. IEONG AND Y. SHOHAM, *Multi-attribute coalitional games*, vol. 2006, 01 2006, pp. 170–179.

[47] G. H. JOHN, R. KOHAVI, AND K. PFLEGER, *Irrelevant features and the subset selection problem*, in Machine Learning Proceedings 1994, W. W. Cohen and H. Hirsh, eds., Morgan Kaufmann, San Francisco (CA), 1994, pp. 121 – 129.

[48] R. J. JOHNSTON, *On the measurement of power: Some reactions to laver*, Environment and Planning A: Economy and Space, 10 (1978), pp. 907–914.

[49] J. JR DEEGAN AND E. PACKEL, *A new index of power for simple n-person games*, International Journal of Game Theory, 7 (1978), pp. 113–123.

[50] J. JURGOVSKY, M. GRANITZER, K. ZIEGLER, S. CALABRETTO, P.-E. PORTIER, L. HE-GUELTON, AND O. CAELEN, *Sequence classification for credit-card fraud detection*, Expert Systems with Applications, (2018).

[51] R. KOHAVI AND G. H. JOHN, *Wrappers for feature subset selection*, Artif. Intell., 97 (1997), pp. 273–324.

[52] S. KOSUB, *A note on the triangle inequality for the jaccard distance*, Pattern Recognition Letters, 120 (2019), pp. 36 – 38.

[53] B. LEBICHOT, F. BRAUN, O. CAELEN, AND M. SAERENS, *A graph-based, semi-supervised, credit card fraud detection system*, in Complex Networks & Their Applications V, H. Cherifi, S. Gaito, W. Quattrociocchi, and A. Sala, eds., Cham, 2017, Springer International Publishing, pp. 721–733.

[54] M. Legesse, G. Gianini, and D. Teferi, *Selecting feature-words in tag sense disambiguation based on their shapley value*, 01 2016, pp. 236–240.

[55] Y. Leung and Y. Hung, *A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 7 (2010), pp. 108–117.

[56] M. Levandowsky and D. Winter, *Distance between sets*, Nature, 234 (1971), pp. 34– 35.

[57] S. Lipovetsky and M. Conklin, *Analysis of regression in game theory approach*, Applied Stochastic Models in Business and Industry, 17 (2001), pp. 319–330.

[58] S. C. Littlechild and G. Owen, *A simple expression for the shapley value in a special case*, Management Science, 20 (1973), pp. 370–372.

[59] Y. Lucas, P.-E. Portier, L. Laporte, S. Calabretto, O. Caelen, L. He-Guelton, and M. Granitzer, *Multiple perspectives hmm-based feature engineering for credit card fraud detection*, in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, ACM, 2019, pp. 1359–1361.

[60] S. M. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions*, in Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.

[61] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers, *Bounding the estimation error of sampling-based shapley value approximation with/without stratifying*, (2013).

[62] I. Mann and L. S. Shapley, *Values of Large Games, IV: Evaluating the Electoral College by Montecarlo Techniques*, Santa Monica, Calif.: RAND Corporation, 1960.

[63] P. Mitra, C. A. Murthy, and S. K. Pal, *Unsupervised feature selection using feature similarity*, IEEE Trans. Pattern Anal. Mach. Intell., 24 (2002), pp. 301–312.

[64] M. Monirul Kabir, M. Monirul Islam, and K. Murase, *A new wrapper feature selection approach using neural network*, Neurocomput., 73 (2010), pp. 3273–3283.

[65] S. Moretti and F. Patrone, *Transversality of the shapley value*, TOP, 16 (2008), p. 1.

[66] R. B. Myerson, *Conference structures and fair allocation rules*, International Journal of Game Theory, 9 (1980), pp. 169–182.

[67] K. Nigam and R. Ghani, *Analyzing the effectiveness and applicability of co-training*, in Proceedings of the Ninth International Conference on Information and Knowledge Management, CIKM '00, New York, NY, USA, 2000, ACM, pp. 86–93.

[68] G. OWEN, *Multilinear extensions of games*, Management Science, 18 (1972), pp. P64–P79.

[69] D. O'CONNOR, *A historical note on shuffle algorithms*, 2014.

[70] D. PA AND J. KITTLER, *Pattern recognition: A statistical approach*, (1982).

[71] B. PELEG AND P. SUDHÖLTER, *Introduction to the Theory of Cooperative Games*, vol. 34, 01 2007.

[72] Y. PENG, Z. WU, AND J. JIANG, *A novel feature selection approach for biomedical data classification*, Journal of biomedical informatics, 43 (2009), pp. 15–23.

[73] K. PRASAD AND J. S. KELLY, *Np-completeness of some problems concerning voting games*, International Journal of Game Theory, 19 (1990), pp. 1–9.

[74] E. RILOFF, J. WIEBE, AND T. WILSON, *Learning subjective nouns using extraction pattern bootstrapping*, in Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4, CONLL '03, Stroudsburg, PA, USA, 2003, Association for Computational Linguistics, pp. 25–32.

[75] C. ROSENBERG, M. HEBERT, AND H. SCHNEIDERMAN, *Semi-supervised self-training of object detection models*, in Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1 - Volume 01, WACV-MOTION '05, Washington, DC, USA, 2005, IEEE Computer Society, pp. 29–36.

[76] A. SALAZAR, G. SAFONT, A. SORIANO, AND L. VERGARA, *Automatic credit card fraud detection based on non-linear signal processing*, in 2012 IEEE International Carnahan Conference on Security Technology (ICCST), IEEE, 2012, pp. 207–212.

[77] F. SANTOSA AND W. SYMES, *Linear inversion of band-limited reflection seismograms*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 1307–1330.

[78] A. SCHOTTER, *The paradox of redistribution: Some theoretical and empirical results*, in Power, Voting, and Voting Power, M. J. Holler, ed., Heidelberg, 1982, Physica-Verlag HD, pp. 324–338.

[79] SHAPLEY, *n balanced sets and cores*.

[80] L. S. SHAPLEY, *Additive and non-additive set functions*, Princeton University, 1953.

[81] L. S. SHAPLEY AND M. SHUBIK, *A method for evaluating the distribution of power in a committee system.*, American political science review, 48 (1954), pp. 787–792.

[82] L. S. SHAPLEY AND M. SHUBIK, *A method for evaluating the distribution of power in a committee system*, The American Political Science Review, 48 (1954), pp. 787–792.

[83] Q. SHEN, R. DIAO, AND P. SU, *Feature selection ensemble*, 06 2012, pp. 289–306.

[84] D. B. SKALAK, *Prototype and feature selection by sampling and random mutation hill climbing algorithms*, in Machine Learning Proceedings 1994, W. W. Cohen and H. Hirsh, eds., Morgan Kaufmann, San Francisco (CA), 1994, pp. 293 – 301.

[85] O. SKIBSKI, T. P. MICHALAK, AND T. RAHWAN, *Axiomatic characterization of game-theoretic centrality*, Journal of Artificial Intelligence Research, 62 (2018), pp. 33–68.

[86] P. SMETS, *The Transferable Belief Model for Quantified Belief Representation*, Springer Netherlands, Dordrecht, 1998, pp. 267–301.

[87] L. SONG, A. SMOLA, A. GRETTON, K. M. BORGWARDT, AND J. BEDO, *Supervised feature selection via dependence estimation*, in Proceedings of the 24th International Conference on Machine Learning, ICML '07, New York, NY, USA, 2007, ACM, pp. 823–830.

[88] J. STIER, G. GIANINI, M. GRANITZER, AND K. ZIEGLER, *Analysing neural network topologies: a game theoretic approach*, Procedia Computer Science, 126 (2018), pp. 234 – 243.

Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia.

[89] P. D. STRAFFIN, *Power Indices in Politics*, Springer New York, New York, NY, 1983, pp. 256–321.

[90] P. D. STRAFFIN AND J. P. HEANEY, *Game theory and the tennessee valley authority*, International Journal of Game Theory, 10 (1981), pp. 35–43.

[91] E. ŠTRUMBELJ AND I. KONONENKO, *Explaining prediction models and individual predictions with feature contributions*, Knowledge and information systems, 41 (2014), pp. 647–665.

[92] M. SUZUKI AND M. NAKAYAMA, *The cost assignment of the cooperative water resource development: A game theoretical approach*, Management Science, 22 (1976), pp. 1081–1086.

[93] V. VAN VLASSELAER, C. BRAVO, O. CAELEN, T. ELIASSI-RAD, L. AKOGLU, M. SNOECK, AND B. BAESENS, *Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions*, Decision Support Systems, 75 (2015), pp. 38–48.

[94] L. VERGARA, A. SORIANO, G. SAFONT, AND A. SALAZAR, *On the fusion of non-independent detectors*, Digital Signal Processing, 50 (2016), pp. 24–33.

[95] J. von Neumann, O. Morgenstern, and A. Rubinstein, *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*, Princeton University Press, 1944.

[96] X. Wang and K. K. Paliwal, *Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition*, Pattern Recognition, 36 (2003), pp. 229–239.

[97] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, *Use of the zero-norm with linear models and kernel methods*, Journal of Machine Learning Research, 3 (2003), pp. 1439–1461.

[98] D. Yarowsky, *Unsupervised word sense disambiguation rivaling supervised methods*, in Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics, ACL '95, Stroudsburg, PA, USA, 1995, Association for Computational Linguistics, pp. 189–196.

[99] Z. Zhao and H. Lu, *Semi-supervised feature selection via spectral analysis*, Proceedings of the 7th SIAM International Conference on Data Mining, (2007), pp. 641–646.

[100] K. Ziegler, O. Caelen, M. Garchery, M. Granitzer, L. He-Guelton, J. Jurgovsky, P.-E. Portier, and S. Zwicklbauer, *Injecting semantic background knowledge into neural networks using graph embeddings*, in 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, 2017, pp. 200–205.

[101] G. Zlotkin and J. Rosenschein, *Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains*, Proceedings of the National Conference on Artificial Intelligence, 1 (2000).