INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores

Desenvolvimento de um package em R para Ensemble Feature Ranking – EFR

Henrique Manuel Carvalho Gomes

(Bacharel em Engenharia Eletrónica e de Telecomunicações)

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor Nuno Miguel Soares Datia
Doutora Matilde Pós-de-Mina Pato

Júri:

Presidente: Doutor Carlos Jorge de Sousa Gonçalves

Vogais: Doutor Artur Jorge Ferreira
Doutora Matilde Pós-De-Mina Pato

Novembro, 2021

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores

Desenvolvimento de um package em R para Ensemble Feature Ranking – EFR

Henrique Manuel Carvalho Gomes

(Bacharel em Engenharia Eletrónica e de Telecomunicações)

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Doutor Nuno Miguel Soares Datia
Doutora Matilde Pós-de-Mina Pato

Júri:

Presidente: Doutor Carlos Jorge de Sousa Gonçalves

Vogais: Doutor Artur Jorge Ferreira
Doutora Matilde Pós-De-Mina Pato

Novembro, 2021

Agradecimentos

A minha participação no Mestrado em Engenharia Informática e de Computadores no Instituto Superior de Engenharia de Lisboa foi um regresso à escola que muito me ajudou na minha carreira profissional. Devo, por isso, um obrigado ao ISEL.

Os meus agradecimentos aos meus orientadores, Professor Doutor Nuno Miguel Soares Datia e Professora Doutora Matilde Pós-de-Mina Pato, no seu apoio e orientação, na definição do conteúdo e estrutura do trabalho e também na disponibilização de algumas ferramentas de teste que reduziram em muito o tempo de implementação.

Os meus agradecimentos a todos os professores, colegas e amigos que, de uma forma direta ou indireta, me ajudaram a atingir este objetivo.

Os meus agradecimentos à minha família, que me apoiou nesta decisão difícil de, após tantos anos de dedicação intensa a uma atividade profissional, decidir ocupar parte destes últimos três anos na concretização de mais este objetivo.

Resumo

Em Mineração de Dados e Aprendizagem Automática, o processo de Seleção de Atributos ou Seleção de Características, corresponde à tarefa de eliminar do conjunto de dados original, as características irrelevantes e redundantes, ou seja, aquelas que pouco contribuem como informação preditiva. O processo de Seleção de Características, para além de ser fundamental à otimização e viabilização da geração de modelos preditivos, contribui diretamente para o processo de Extração de Conhecimento. Dada a diversidade do domínio do problema, aplicação e estrutura dos dados a analisar, a generalização e automatização do processo seleção de características é extremamente difícil. Sendo o esforço e tempo atribuído ao pré-processamento dos dados, uma parte substancial do esforço total atribuído a um processo de mineração de dados, uma contribuição na eficiência do processo de seleção de características é relevante para todo o processo mineração de dados. Entre as muitas técnicas e publicações efetuadas sobre o seleção de características, o algoritmo de Avaliação de Características, *Ensemble Feature Ranking (EFR)*, tal como publicado em 2014 no artigo *Ensemble feature ranking applied to medical data*, tem o mérito de poder enquadrar no mesmo processo de seleção de características, um conjunto de diferentes métodos por filtragem, conjugado com um número arbitrário de execuções sobre partições do conjunto de dados com um número reduzido de instâncias, o que o tornam eficiente e adequado a conjuntos de dados de dimensionalidade elevada. Tendo como base o algoritmo EFR, pretende-se a reimplementação mais genérica, eficiente e automatizada desse algoritmo, disponibilizada num package em R, que possa ser reutilizado de forma simples e mais integrada num processo de mineração de dados.

Palavras-chave: Seleção de Características, Mineração de Dados, Avaliação de Características, Filtros, Conjunto de Filtros

Abstract

In Data Mining and Machine Learning, Feature Selection process corresponds to the task of removing from the original data dataset, the irrelevant or redundant attributes, that is, those that present little predictive information. The feature selection process in addition to being fundamental to the optimization and in some cases to enable predictive models, contributes by itself to the Knowledge Discovery in Data. Given the problem domain diversity in data mining (application scope and structure and data types) on feature subset findings, the generalization of the process is extremely difficult. Since the effort and time allocated to data pre-processing is a substantial part of the total effort allocated to a data mining process, any contribution to the efficiency of the feature selection process is relevant for the entire data mining process. Among the many techniques and publications carried out on feature selection, the feature ranking algorithm "Ensemble feature ranking (EFR)", as published in 2014 in the article *Ensemble feature ranking applied to medical data*, has the merit of being able to fit in the same feature selection process, a combined set of different filtering methods executed over an arbitrary number of random small size dataset partitions. Based on the Ensemble feature ranking (EFR) algorithm, the aim is to have a more generic, efficient and automated reimplementation of the algorithm, available in a R Package for Ensemble feature ranking, which can be reused in a simple and more integrated way in data mining processes.

Keywords:

Feature Selection, Data Mining, Feature Ranking, Filters, Filter Ensemble

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Abreviaturas e Siglas

AUC Area Under the ROC Curve. 21, 22, 40, 41, 49, 50

BER Balanced Error Rate. 22, 40, 41, 49, 50

CFS *Correlation-based Feature Selection*. 16, 46, 55
Co Consistência, do inglês *Consistency*. 16
CRISP-DM *Cross-industry standard process for data mining*. 6, 7
CS *Qui*-Quadrado, do inglês *Chi-Squared*. 13, ii

DM Mineração de Dados, do inglês *Data Mining*. 1, 5, 6, 7, 8, 19, 31, 53

EEFR Enhanced Ensemble Feature Ranking. xii, xvii, 3, 5, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56
EFR Ensemble Feature Ranking. vii, ix, 1, 2, 3, 4, 5, 18, 19, 20, 22, 26, 27, 33, 36, 39, 41, 43, 45, 49, 50, 53, 54, 55, i, ii, iv

FR Avaliação de Características, do inglês *Feature Ranking*. 1, 2, 3, 9, 11, 12, 13, 16, 18, 19, 20, 21, 22, 24, 25, 27, 31, 40, 45, 48, 49, 51, 53, 54, i
FS Seleção de Características, do inglês *Feature Selection*. xv, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 21, 22, 23, 29, 31, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 53, 54, 55, 56, ii, iv

GR *Gain Ratio*. 13, 14, ii, iv

HW *Hardware*. 35

IG *Information Gain*. 13, 14, ii, iv

KDD Extração de conhecimento, do inglês *Knowledge Discovery in Databases*. 5, 6

MI Informação Mútua, do inglês *Mutual Information*. 14, 17, 18

ML Aprendizagem Automática, do inglês *Machine Learning*. 1, 7

mRMR Minimal-Redundancy-Maximal-Relevance criterion. 16, 17

Re Relief. 13, 15, 16, 48, 49

RFI *RForest.importance*. 13, 14, 15

RR Eliminação de Redundância, do inglês *Redundancy Removal*. 20, 45, 46, 47

SEMMA processo standard de Mineração de Dados, desenvolvido pelo SAS. 6

SU *Symmetrical Uncertainty*. 13, 14, ii, iv

SW *Software*. 2, 3

1

Introdução

Com a intensificação da produção, disponibilização e utilização dos dados, as técnicas de mineração de dados (DM) [10], tendo como base a aprendizagem automática (ML) na modelização dos dados, estão cada vez mais presentes e integradas nas muitas atividades sociais [40]. Como parte integrante do processo de DM, a **Seleção de Características, do inglês *Feature Selection* (FS)** [14, 23] é fundamental para o desempenho e eficiência de todo o processo. O processo de FS [23] é focado na identificação do subconjunto das características dum conjunto de dados original, com maior **relevância** e com menor **redundância** entre si. O FS funciona como pré-processo à análise e aprendizagem automática dos dados (ML), com vista ao aumento da sua eficiência e desempenho. Entre os muitos métodos de FS disponíveis, os **métodos de FS por filtragem**, por serem de baixa complexidade, são os que apresentam maior desempenho em conjuntos de dados de dimensionalidade elevada, apesar de não garantirem o melhor subconjunto de características para determinado classificador [5, 14, 23]. Entre os muitos métodos de FS por filtragem existentes, os métodos por **Avaliação de Características, do inglês *Feature Ranking* (FR)** [14] são os de menor complexidade. Os métodos de Avaliação de Características (FR), são considerados métodos de FS por avaliarem e ordenarem as características por ordem decrescente de relevância [5, 14]. O algoritmo **Ensemble Feature Ranking (EFR)** tal como apresentado no âmbito do trabalho de Santos et al. [37], corresponde a um método de FR, mais elaborado, o qual pode utilizar diversos filtros (*ensemble*), na avaliação das características (*ranking*), a partir da qual se pode selecionar um subconjunto das k melhores características. Tal como indicado na publicação original do EFR e como veremos no Capítulo 3, este algoritmo de FS

caracteriza-se pela utilização dum *ensemble* de filtros, funções de medida da relevância relativa das características, e múltiplas amostragens de dados, partições de instâncias retiradas aleatoriamente do conjunto de dados original. Estas características do EFR permitem melhorar a eficiência dos modelos de classificação, gerados sobre conjuntos de dados de dimensionalidade elevada, sem comprometer o nível de desempenho de todo o processo, devido a um potencial incremento da computação do pré-processo associado ao FS.

Entende-se por: (i) conjunto de dados, um conjunto de n ocorrências / instâncias num espaço multidimensional de dimensionalidade d, sendo d o número de características. (ii) Por dimensionalidade elevada, um conjunto de dados com centenas ou milhares de características. (iii) Por desempenho, a capacidade do algoritmo de FS em executar em tempo útil [11]. (iv) Por complexidade, a evolução dos tempos de execução do algoritmo de FS em função da dimensionalidade d do conjunto de dados. (v) Por *Ensemble*, neste contexto de FS, a utilização de um conjunto de diferentes funções de medida (filtros) na avaliação das características do conjunto de dados, por analogia à utilização de diferentes modelos de classificação na fase de aprendizagem automática.

1.1 Motivação

O algoritmo EFR original, pode ser melhorado em diversos aspetos, entre os quais: A função W, agregadora dos resultados obtidos pelos diversos filtros, inclui valores de atribuição de constantes no código, os quais se pretendem eliminar por dependerem do conjunto de dados utilizado. Tal como outros métodos de FR, o EFR devolve como resultado uma lista de avaliação das características ordenada inversamente pela sua relevância (*ranking*), sem selecionar, de forma automática, o subconjunto das k características mais relevantes. Pretende-se adicionar ao algoritmo, uma opção que determine, automaticamente, sem necessidade de atribuição de valores arbitrários, um subconjunto de k características que de uma forma genérica, otimize a qualidade dos modelos de aprendizagem. O algoritmo utiliza um *ensemble* de filtros, muito semelhantes entre si. Como forma de generalização, pretende-se procurar um conjunto de filtros, mais abrangente que o incluído por omissão no EFR, que apresente melhor média de resultados quando aplicado a múltiplos modelos de classificação e tipos de dados. A estrutura e documentação do *Software* (SW) de implementação, pode ser melhorada por implementação de um pacote R, de forma a disponibilizar o algoritmo de FS de uma forma mais simples, documentada e mais próxima das práticas utilizadas pela comunidade R.

Este trabalho pretende contribuir para a simplificação e verificação do processo de FS, a partir da reimplementação do *Ensemble Feature Ranking (EFR)* num novo algoritmo *Enhanced Ensemble Feature Ranking (EEFR)*, mais automático e abrangente aos diversos tipos de dados e modelos de classificação. Essas contribuições incluem: (i) alteração da função de medida da relevância das características, evitando valores de parametrização arbitrários e otimizando os resultados obtidos; (ii) a seleção automática do subconjunto das características relevantes a partir do FR; (iii) determinação de lista de filtros mais abrangente, na avaliação das características; (iv) otimização do código em R para uma estrutura de SW mais funcional; (v) a implementação de um pacote de software (*software package*) que implemente o SW do EEFR, de uma forma mais simples, documentada e com o desempenho de execução necessário, tendo como alvo principal o pré-processamento de conjuntos de dados de elevada dimensionalidade.

Resumindo, os grandes objetivos deste trabalho, são: (i) automatização — adicionando à funcionalidade de avaliação de características (*feature ranking*) a seleção de um subconjunto das melhores características (*feature selection*), reduzindo a necessidade de afinação por tentativas; (ii) generalização — encontrando um conjunto de filtros a utilizar por omissão, que resultem numa seleção de características mais independente do conjunto de dados a analisar; (iii) simplificação — produzindo um package em R de utilização simples e bem documentado; (iv) desempenho — mantendo um nível de baixa complexidade do algoritmo e otimizando o seu código de modo a adequá-lo à utilização em conjuntos de dados de elevada dimensionalidade.

1.2 Organização do documento

Este documento descreve o problema, os objetivos, o contexto teórico e a abordagem seguida na implementação do EEFR: (i) O Capítulo 1, Introdução, identifica o problema assim como os motivos e objetivos globais do trabalho. (ii) O Capítulo 2, Avaliação de Características por Filtragem, desenvolve os conceitos de FS, FS por filtragem e FR com a apresentação de exemplos para as diferentes abordagens e contextualiza os algoritmos EFR e EEFR dentro dos muitos e diferentes tipos de FS. (iii) O Capítulo 3, *Enhanced Ensemble Feature Ranking*, descreve as alterações e adições efetuadas ao algoritmo original, em linha com o definido no Capítulo 1. (iv) O Capítulo 4, Implementação do EEFR num pacote R e Testes, descreve a implementação dos pacotes de SW, *EEFRanking* e *pEEFRanking* para execução paralela e descreve o conjunto de testes efetuados e resultados obtidos na aplicação e verificação das funcionalidades descritas no Capítulo 3. (v) O Capítulo 5, Conclusões, apresenta um resumo das conclusões. (vi) O

Anexo A descreve a implementação original do Ensemble Feature Ranking (EFR), o seu interesse e aspetos diferenciadores, assim como a identificação das alterações / otimizações a implementar de modo a se atingirem os objetivos.

2

Avaliação de Características por Filtragem: Conceitos e Definições

Num processo de Mineração de Dados (DM), o pré-processo de FS, como parte integrante da fase de preparação dos dados, é fundamental para a obtenção de conhecimento, pela análise das características mais relevantes [14, 23]. A seleção de características por meio dos métodos de FS, tem como objetivo aumentar a eficiência dos classificadores e o desempenho computacional de todo o processo de DM.

Dentro dos vários métodos de FS, os métodos por filtragem, tomam especial relevo no caso de conjuntos de dados de dimensionalidade elevada. O EFR é um exemplo desses métodos por filtragem.

Este capítulo será focado no tema FS, resumindo e classificando os vários métodos e técnicas mais utilizados, destacando os métodos de FS por filtragem de forma a melhor contextualizar o algoritmo EFR e o novo algoritmo EEFR que pretendemos implementar.

2.1 Mineração de Dados

Parte integrante do processo de obtenção de conhecimento a partir dos dados, tal como definido em KDD [10], o processo DM, que inclui um conjunto de técnicas utilizadas na análise de grandes volumes de dados, é um processo demorado, especializado e com

forte envolvimento humano. De modo a normalizar o processo de DM, tornando-o sistemático e acessível a diferentes indústrias e setores de investigação, apareceram várias iniciativas de sistematização no processo de modelação, tais como, o *Cross-industry standard process for data mining* (CRISP-DM) [39], o processo sequencial *sample, explore, modify, model, assess*, processo standard de Mineração de Dados, desenvolvido pelo SAS (SEMMA) [38] e o processo KDD [10]. Tomando como exemplo o CRISP-DM, o qual teve origem num consórcio de várias empresas e corresponde a um dos mais utilizados no mercado [7], este modelo, tal como ilustrado na Figura 2.1 corresponde a um processo de DM visto como uma cadeia de 6 fases: (i) Entendimento do Negócio – corresponde à parte em que se decide quais os objetivos a atingir com a análise dos dados. (ii) Entendimento dos Dados – identificação do conjunto de dados a analisar, qualidade dos mesmos, dimensão, tipo, entre outros. (iii) Preparação dos Dados – limpeza do conjunto de dados e seleção de características (FS). (iv) Modelagem – a partir do conjunto de dados inicial (treino) e utilizando algoritmos de aprendizagem / classificação, gerar modelos preditivos. (v) Avaliação – utilizando conjunto de instâncias (conjunto de dados de teste) medir a qualidade no modelo a partir da sua capacidade de predição. (vi) Disponibilização – dependendo dos objetivos inicialmente definidos, disponibilização do modelo para utilização em ações de predição ou no apoio à decisão. Na evolução observada nos últimos anos da *mineração de dados* para a *ciência de dados*, esta última mais focada na exploração dos dados, o CRISP-DM continua a ser um dos modelos de referência de todo o processo DM [1].

Este trabalho é focado na fase de preparação dos dados, fase (iii) do modelo de referência, considerada da maior importância uma vez que consome cerca de 60% [27] de todo o tempo atribuído ao processo de DM. As fases de entendimento dos dados (ii), de modelagem (fase iv) e avaliação (fase v), do modelo de referência DM, são igualmente importantes e aplicadas durante a fase de testes do trabalho, mas unicamente como processos auxiliares.

A seleção de características (FS), parte integrante da fase de preparação dos dados, é fundamental a todo o processo de DM, acarretando vários ganhos ao processo, nomeadamente [14, 23]: (i) Na redução do tempo e dos recursos utilizados no processo de aprendizagem dum modelo, nalguns casos fundamental à viabilização do mesmo; (ii) na compreensão do modelo final como parte importante na obtenção do conhecimento (identificação de padrões); (iii) na redução de situações de sobre-ajuste do modelo ao conjunto de dados de aprendizagem, aumentando assim a generalização do modelo obtido; (iv) aumento da exatidão (*accuracy*) do modelo de classificação; (v) minimizar os efeitos ao fenómeno da *maldição da dimensionalidade* (do inglês, *curse of dimensionality [6]*), expressão de *Richard E. Bellman* relativamente a problemas na análise

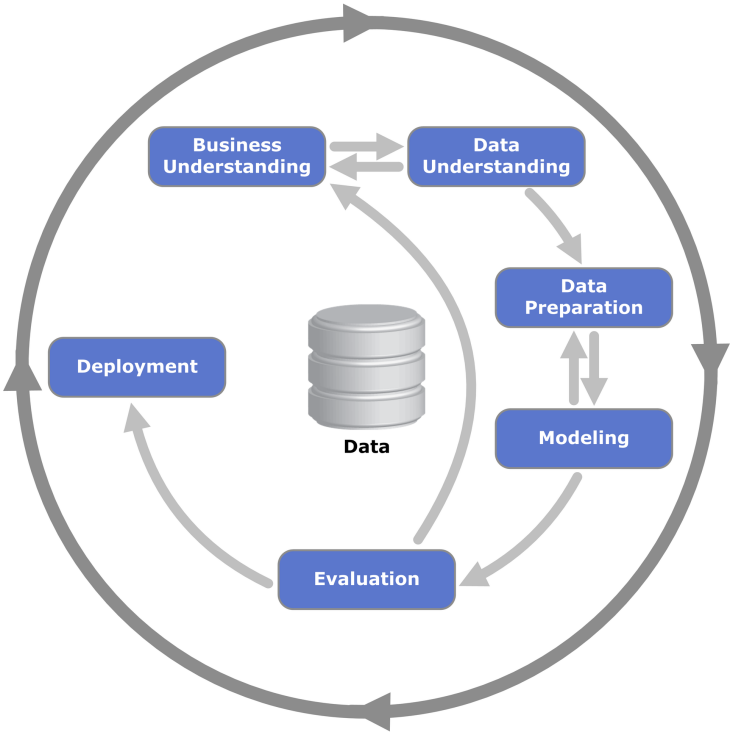


Figura 2.1: Diagrama do processo de mineração de dados CRISP-DM e relação entre as fases [39]

de dados de grande dimensionalidade. Este fenómeno de dimensionalidade ocorre em domínios de DM e de Aprendizagem Automática, do inglês *Machine Learning* (ML), em que em situações de aumento da dimensionalidade, o volume do espaço aumenta de tal forma que os dados disponíveis se tornam esparsos. Essa dispersão é problemática para qualquer método que requeira significância estatística. Para obter um resultado estatisticamente sólido e confiável, a quantidade de dados (instâncias) necessária para suportar o resultado geralmente aumenta exponencialmente com a dimensionalidade, situação conhecida por *fenómeno de Hughes* [19].

Por motivos de desempenho, a automatização do pré-processo de FS é desejável. Contudo, a generalização dum determinado método de FS alargado a diferentes domínios e modelos de classificação, é de difícil realização [14, 23]. As causas para essa dificuldade são diversas, nomeadamente: (i) a multiplicidade de aplicações, onde o DM e *data science* estão a ser utilizados; (ii) a diversidade do tipo e da dimensionalidade dos dados em análise; (iii) a multiplicidade de métodos de FS disponíveis [2]; (iv) e o facto do melhor subconjunto de características obtido por determinado método de FS, não ser independente do método de classificação ou da medida de avaliação do modelo (exatidão, precisão), utilizados. No entanto, a automatização do pré-processo de FS, é importante, uma vez que, para além de contribuir para o aumento do desempenho

de todo o processo DM, reduz o risco de enviesamento do modelo e resultados obtidos, caso a seleção de características seja pouco cuidada por subjetividade humana na seleção das características ou na definição de parâmetros do algoritmo de FS [26].

2.2 Avaliação e Seleção de Características

Em Mineração de Dados a redução de dimensionalidade do conjunto de dados é fundamental para que os algoritmos de aprendizagem sejam mais rápidos e eficientes [10], existindo várias formas de redução de dimensionalidade, nomeadamente, as de projeção (exemplo principal component analysis - PCA), de compressão e a seleção de características FS [23]. Contrariamente a outras formas de redução de dimensionalidade, os métodos de FS, mantêm a semântica original do conjunto de dados e correspondem à procura de um subconjunto mínimo dos atributos / características do conjunto de dados que representem as propriedades relevantes do mesmo, removendo as características com informação irrelevante e / ou redundante. Considerando o conjunto de dados com d características, classificar e testar todos os 2^d subconjuntos de características possíveis, de modo a encontrar aquele que apresente o máximo de exatidão torna o processo de FS intratável computacionalmente. Assim, os muitos e diferentes métodos de FS disponíveis, são subótimos no sentido em que, o subconjunto de características encontrado pode não ser o que garante a exatidão máxima pelo preditor ou que o subconjunto encontrado não seja único [5]. Este trabalho estará focado nos métodos de FS para aprendizagem supervisionada e mais especificamente, num problema mais recente que é o pré-processamento de conjuntos de dados de dimensionalidade elevada.

Para este trabalho entre as muitas referências utilizadas, destacamos os seguintes artigos e livros: (i) Guyon and Elisseeff 2003 [14], Yu and Liu 2004 [43], Liu and Motoda 2012) [23], Chandrashekar and Sahin 2014 [5], Jović et al. 2015 [20] os quais nos guiam sobre o tema FS em geral, atual classificação e enquadramento dos diferentes algoritmos, vantagens e desvantagens das diferentes aproximações; (ii) Dash and Liu 2003 [8], Peng et al. 2005 [17], Pinoand and Morell 2013)[30], Hoque et al. 2018 [18], Urbanowicz et al. 2018 [32], Stief et al. 2018 [41], Bommert et al. 2020 [2] sobre o tema mais específico do FS por filtragem, exemplos de abordagens e o tema do FS em conjuntos de dados de dimensionalidade elevada; (iii) Khaire and Dhanalakshmi 2019 [21] foca o tema da estabilidade dos algoritmos FS, definida como a propriedade do algoritmo apresentar o mesmo subconjunto ou a mesma ordem de avaliação das características mais importantes, em diferentes amostragens do conjunto de dados. Apesar do tema

da estabilidade não fazer parte do trabalho, ele é importante em futuros testes de avaliação dos algoritmos de FS.

Existem inúmeros métodos de FS os quais, de uma forma geral, enquadram-se na seguinte classificação [5, 14, 23]:

Métodos *Wrapper* Processo de seleção de um subconjunto de características mínimo, que maximize uma medida de qualidade ("accuracy" ou outra) dum modelo preditivo / classificador. O método de procura do subconjunto de características pode ser efetuado de diversas formas, a partir de um subconjunto inicial, adicionando características — *forward*, ou eliminando características — *backward*, utilizando diversos algoritmos de pesquisa.

Métodos de Filtragem O método de filtragem é aplicado como pré-processamento dos dados, anterior e independente ao processo de classificação. Tipicamente, os métodos de filtragem seguem um processo de avaliação de características (FR) por métodos estatísticos, os quais medem a relevância relativa de cada característica, seguido da sua ordenação por ordem decrescente das medidas e seleção do subconjunto com as primeiras k. Existem, no entanto, outros métodos de FS por filtragem, mais complexos, com funcionamento próximo ao dos *wrapper*, avaliando subconjuntos de características no seu todo e selecionando aquele que atingir determinado limiar (*threshold*).

Métodos Embutidos A eliminação das características é efetuada, durante e pelo processo de modelagem. Durante a fase de treino do modelo, em cada iteração, as características que mais contribuem para o modelo, vão sendo selecionadas.

2.2.1 Métodos *wrapper*

Tipicamente os mecanismos de FS incluem no seu funcionamento, 4 funções principais (ver Figura 2.2) [8]: (i) Geração — A função geração, usa uma estratégia de procura para gerar subconjuntos de características, para avaliação. A partir dum subconjunto inicial (vazio, com todas as características, ou aleatório), os subconjuntos vão sendo produzidos iterativamente por adição ou remoção de características, ou produzidos aleatoriamente. (ii) Avaliação — A função de avaliação avalia um subconjunto gerado pela função de geração, se a avaliação do novo subconjunto for superior ao anterior, este substitui o anterior. (iii) Critério de paragem — Dependente da função de geração utilizada, existem vários critérios de paragem da procura, tais como, número de características atingido, número de iterações máximo atingido, obtenção do valor objetivo

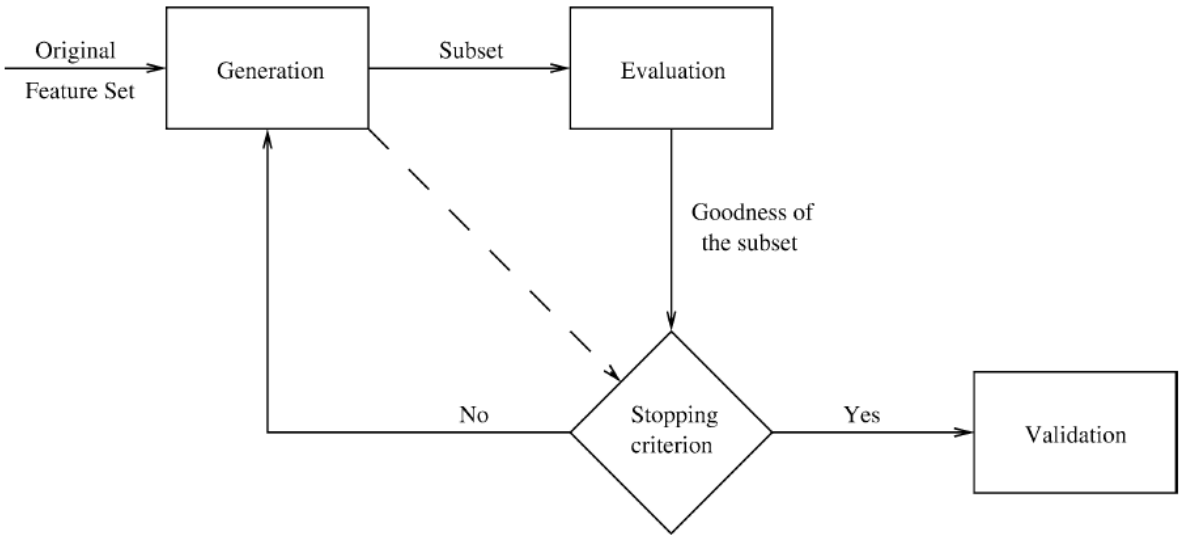


Figura 2.2: Processo de Seleção de Características com validação [8]

da função de avaliação. (iv) Validação — esta função final verifica se o subconjunto gerado é válido.

Os métodos *wrapper* utilizam o classificador (modelo preditivo) e teste como função de avaliação e efetuam uma procura iterativa sobre subconjuntos de características até encontrar um que, após a aprendizagem do modelo e teste, maximize uma medida, tipicamente a exatidão do modelo preditivo. A forma cíclica utilizada de aprendizagem e teste do modelo, torna este método pesado e pouco adequado para conjuntos de dados de elevada dimensionalidade. Apesar de saber selecionar o subconjunto ótimo de características para um classificador específico, o "*wrapper*" pode potencialmente excluir algumas características relevantes [14] para outros modelos de classificação.

Por não necessitarem de aprendizagem de modelo de classificação e teste, cíclico por subconjuntos de características, os métodos por filtragem provam ser de desempenho superior aos do tipo *wrapper* [16] quando aplicados a *conjunto de dados* de grande dimensionalidade [11]. Este capítulo e o trabalho em geral, são focados nos métodos por filtragem.

2.2.2 Métodos por Filtragem

De um modo geral, os métodos de FS por filtragem, dividem-se em: (i) avaliação de características (FR), que a partir de medidas estatísticas uni-variável avaliam a relevância relativa entre as características em relação à classe; e (ii) seleção de características (FS),

que a partir de medidas estatísticas multivariável, avaliam subconjuntos de características combinando medidas de relevância com medidas de redundância entre características. Na definição de característica relevante ou irrelevante, utilizamos a seguinte citação: *"A feature can be regarded as irrelevant if it is conditionally independent of the class label"* [5], ou seja, características sem influência sobre a classe, podem ser descartadas. Os métodos de FS por filtragem são *subótimos*, por não garantirem o melhor subconjunto de variáveis para determinado classificador, particularmente nos métodos FR se as variáveis são redundantes, mas apresentam um conjunto de características importantes. Eles podem ser combinados com qualquer modelo de aprendizagem, independente do algoritmo de classificação, reduzindo o tempo de execução de algoritmos de aprendizagem automática [2]. O foco deste trabalho é centrado nos métodos de FR, de menor complexidade e mais adequados a conjuntos de dados de elevada dimensionalidade. No entanto, a seleção de características por relevância, por si só, não é eficiente em conjuntos de dados de elevada redundância [43], pelo que existem outros métodos de FS por filtragem, mais complexos, que combinam a seleção de características baseada na relevância e na redundância.

No contexto dos métodos por filtragem, de forma a clarificar diferentes conceitos, os quais se combinam nas várias implementações de filtragem, estabelecemos as seguintes definições:

Definição 2.1 *Padrão (Tuplo) — Uma ocorrência ou instância X_l definida por um conjunto de valores x_{lk} atribuídos às d variáveis independentes, características F_k ($k = 1...d$).*

Definição 2.2 *Conjunto de dados — n ocorrências ou instâncias $\{X_l , y_l\}$ ($l = 1...n$), cada uma composta pelo tuplo X_l e pela variável dependente y_l (classe).*

Definição 2.3 *Subconjunto de características — Uma das 2^d possíveis combinações das F_k ($k = 1...d$) características que compõem o conjunto de dados original.*

Definição 2.4 *Filtro — medida de filtragem efetuada por uma função estatística que a partir dum conjunto de instâncias $\{X_l , y_l\}$ ($l = 1...n$) avalia as características, devolvendo um vetor de medidas (weights), cujas componentes $weights_k$ ($k = 1...d$) correspondem à relevância relativa de cada característica F_k, atribuída por essa medida.*

Definição 2.5 *Ranking — vetor com as posições relativas de cada uma das característica F_k ($k = 1...d$), considerando-as por ordem decrescente do valor das relevâncias.*

Definição 2.6 *Avaliação de Características, do inglês Feature Ranking (FR) por filtragem — função que, utilizando as medidas de filtragem, devolve um vetor com todas as características, ordenadas por ordem decrescente da sua relevância.*

Definição 2.7 *Seleção de Características, do inglês Feature Selection (FS) por filtragem — função que, a partir da avaliação de características (FR) por filtragem, ou das medidas de filtragem, devolve um vetor com o subconjunto das k características mais relevantes.*

As medidas de filtragem agrupam-se pelos seguintes tipos [8]:

informação medidas baseadas no ganho de informação duma determinada característica, ou seja, a diferença de incerteza na predição da classe atribuída a determinado padrão, antes e após conhecer a característica em avaliação. Quanto maior a diferença, maior a relevância da característica. Define-se padrão como um tuplo de valores atribuídos ao conjunto (ou subconjunto) das características. O cálculo da incerteza é baseado na entropia da informação H (equação 2.1) Shannon [9];

$$H(X) = -\sum_{i=1}^{n} p_i \times log(p_i) \tag{2.1}$$

onde p_i representa a probabilidade da variável aleatória X apresentar o valor x_i.

distância mede a distância entre as probabilidades condicionais da ocorrência de cada classe antes e após conhecer o valor da característica. Quanto maior a distância entre as duas probabilidades, mais relevante é a característica;

dependência medida de correlação entre uma característica e a classe, a qual mede a capacidade preditiva da classe em função da característica;

consistência forma de medida multivariável, ou seja, mede um subconjunto de características no seu todo. Utiliza a inconsistência entre instâncias como forma de medida, ou seja, a diferença entre o número de ocorrências de um determinado tuplo de valores das características e o máximo número de ocorrências do mesmo tuplo com determinada classe.

Uma das dificuldades da avaliação das características (FR) é a partir do *ranking* obtido, determinar qual o valor de k, para a seleção do subconjunto das k primeiras características a utilizar no processo de aprendizagem. Tipicamente, o valor de k ou a percentagem do número total de características a selecionar, é arbitrário ou determinado

de uma forma heurística. No entanto, existem outros métodos de obtenção do subconjunto, tais como, eliminando as características com medidas atribuídas pelo filtro inferiores a um determinado limiar [43], ou ainda, calculando o valor acumulado das medidas das características ordenadas inversamente pelo valor das medidas e eliminando todas aquelas cujo valor de acumulação seja superior a um limiar [11]. Ambos os casos exigem a parametrização de valores arbitrários e o respetivo teste cíclico de afinação do parâmetro, impedindo a automatização do pré-processo de FS. Outros métodos de FS por filtragem, utilizam um processo idêntico ao da Figura 2.2, neste caso, fazendo uso de medidas de filtragem multivariável como função de avaliação, em vez da exatidão dum determinado classificador e terminando a seleção quando determinado subconjunto de características atingir limiar pré-definido. Neste capítulo incluímos alguns exemplos de medidas de filtragem utilizadas em FR, assim como alguns destes métodos de filtragem FS mais complexos.

Existem inúmeros tipos de filtros e métodos de filtragem. De entre os muitos artigos comparativos entre os diferentes métodos [2, 29], o artigo *Bommert et al. 2020* [2] descreve os testes e comparação entre 22 deles. Este artigo, inclui no final, a seguinte conclusão: *Feature selection is a key part of data analysis, machine learning and data mining. There are many methods for feature selection, but it is unclear which of these methods perform best. In this analysis we focused on the comparison of filter methods for feature selection. We analyzed 22 filter methods based on 16 high-dimensional classification data sets from various domains*. Este estudo faz uma comparação entre os diferentes filtros e respetivas similaridades. A conclusão de que não existe um filtro melhor que outros, aponta para a utilização de um novo método de FS *ensemble* composto por uma combinação de vários filtros. A análise de similaridades entre filtros será útil para a escolha dos filtros do algoritmo de FS, no caso de se desenhar o algoritmo de FS com combinação entre diferentes filtros, que corresponde a um dos objetivos deste trabalho. Um dos critérios associados aos tipos de filtros é a sua complexidade em termos de tempo de execução. Neste trabalho, focado em conjuntos de dados de elevada dimensionalidade, a ordem de complexidade, refere-se à evolução dos tempos de execução como função da dimensionalidade d. Da lista de filtros, tal como indicado em (*bommert2020*) [2], descrevemos aqueles que foram utilizados nos testes do Capítulo 4: *Qui*-Quadrado, do inglês *Chi-Squared* (CS), *Information Gain* (IG), *Gain Ratio* (GR), *Symmetrical Uncertainty* (SU), *RForest.importance* (RFI) e Relief (Re).

***Qui*-Quadrado, do inglês *Chi-Squared* (CS)** O algoritmo avalia as características com base no teste *qui*-quadrado entre cada característica e a classe, a partir dum conjunto

de dados de características e classe categóricas. O *qui*-quadrado avalia cada característica pela relação entre as frequências de ocorrência de cada categoria da classe e a sua frequência expectável, assumindo independência entre elas. O *qui*-quadrado da característica k é calculado por:

$$\chi_k^2 = \sum_{i=1}^{2}\sum_{j=1}^{s}(o_{ij}^{(k)} - e_{ij}^{(k)})^2/e_{ij}^{(k)} \tag{2.2}$$

onde $o_{ij}^{(k)}$ representa, por característica k, o no. de observações com classe i na categoria j, e $e_{ij}^{(k)}$ o no. expectável de ocorrências da classe i na categoria j, assumindo independência entre a classe e a característica. O método de filtragem utiliza a estatística χ^2 para dar pesos (*weights*) às características.

Informação Mútua, do inglês *Mutual Information* (MI) Baseados em medidas de informação, medem o ganho de informação sobre a classe pelo facto de se conhecer o valor da característica, utilizando a entropia de informação H (2.1) como forma de cálculo. Com base na informação mútua entre as características e a classe, existem várias variantes de medidas de relevância das características. Considerando a classe Y, a característica F_i e a entropia conjunta $H(Y, F_i)$, definimos os seguintes métodos de medida:

1. *Information Gain* (IG),

$$IG_i = H(Y) + H(F_i) - H(Y, F_i) \tag{2.3}$$

2. *Gain Ratio* (GR),

$$GR_i = (H(Y) + H(F_i) - H(Y, F_i))/H(F_i) \tag{2.4}$$

3. *Symmetrical Uncertainty* (SU),

$$SU_i = 2 * (H(Y) + H(F_i) - H(Y, F_i))/(H(Y) + H(F_i)) \tag{2.5}$$

***RForest.importance* (RFI)** Este algoritmo utiliza a estrutura do *Random Forest* [3] para avaliar as características por relevância atribuindo-lhes importâncias. Na implementação utilizada, a importância pode ser atribuída de duas formas: (i) Diminuição média da exatidão (*mean decrease in accuracy*) — amostras de instâncias são treinadas num modelo formado por um conjunto de árvores (*Random.forest*). As predições das instâncias não utilizadas no treino do modelo, são utilizadas para cálculo da importância de cada

característica F_i, pela diferença das respetivas exatidões, observadas antes e depois de permutar os valores dessas características nas várias instâncias. (ii) Diminuição média da impureza (*mean decrease in node impurity*) ou *Gini importance* — por cada nó da árvore, é definida a sua impureza associada ao conjunto de tuplos X que passam pelo nó. A importância duma característica está associada à diferença da impureza, antes e depois da divisão do conjunto de tuplos X, por subconjuntos de categorias duma característica. A medida de impureza Gini é definida pela equação (2.6), impureza dum conjunto de tuplos X, e por (2.7) [42], impureza após a divisão do conjunto X tuplos por diferentes categorias associadas a uma determinada característica.

$$Gini(X) = 1 - \sum_{i=1}^{c} p_i^2 \tag{2.6}$$

onde p_i representa a probabilidade de tuplos com a classe $i, (i = 1..c)$ no conjunto de tuplos X.

$$Gini_{split}(X) = \sum_{j=1}^{s} n_j/n * Gini(X_j) \tag{2.7}$$

onde X_j corresponde ao subconjunto de n_j tuplos da categoria $j, (j = 1..s)$ após a classificação do conjunto X (n tuplos) em s categorias. A importância duma característica F_i numa árvore de decisão, corresponde à soma das médias ponderadas das diferenças dos *Gini* dos nós classificados por F_i , após e antes da classificação. As importâncias das características da *random forest* são obtidas pelas médias das importâncias das características das árvores individuais.

Considerando um conjunto de dados com n instâncias e dimensionalidade d, correspondente ao número de características, os tempos de execução deste filtro RFI são superiores aos outros anteriores, mais dependente do número de instâncias $O(nlog(n)d)$. Como veremos no Capítulo 4, empiricamente na implementação utilizada e tendo em conta o foco em conjunto de dados de elevada dimensionalidade, a maior complexidade deste filtro em relação aos outros não é impeditiva deste ser utilizado em operações de FS em conjuntos de dados de dimensionalidade elevada.

Relief (Re) Medida de distância [22]. As avaliações da relevância das características são baseadas na identificação de diferenças de valor de cada característica entre pares de instâncias vizinhas mais próximas. Se existir diferença de valores da característica em duas instâncias vizinhas com a mesma classe, o valor da relevância da característica diminui. Alternativamente, se existir diferença de valores da característica em

duas instâncias vizinhas com classes diferentes, o valor da relevância da característica aumenta. O filtro Re apresenta uma evolução do tempo de execução de ordem $O(d)$ [32] parecendo adequado a ser utilizado em operações de FS em conjuntos de dados de dimensionalidade muito elevada, o que, como veremos no Capítulo 4 não parece acontecer experimentalmente com certas implementações. A variante, *Relief-F* [41] corresponde a uma extensão do *relief* para conjuntos de dados multi-classe.

2.2.3 *Frameworks* de FS por Filtragem

Como referido atrás, o método típico do FS por filtragem corresponde à avaliação de características (FR) utilizando uma medida de filtragem, seguido da seleção do subconjunto das k melhores características. Para além dos muitos métodos de FS que surgem da aplicação do FR sobre as diferentes medidas de filtragem existentes (*gain ratio, info gain, reliefF, ...*), existem outros *frameworks* [30], mais elaborados, que implementam FS por filtragem. Alguns exemplos desses *frameworks*, são: o *CFS* e a *consistência* como exemplos de métodos de FS multivariável conhecidos, o mRMR como método híbrido e o *EFS-MI* e *EFR*, como exemplos de métodos FR utilizando *ensemble* de filtros.

***Correlation-based Feature Selection* (CFS)** O CFS [25], corresponde a um modelo de funcionamento da Figura 2.2. Ele consiste na utilização de um método de procura por *best.first.search* de um subconjunto de características, utilizando como função de avaliação, uma função heurística multivariável que relaciona a correlação entre as características e a classe no cálculo da relevância e a correlação entre as características no calculo de redundância entre elas. O CFS avalia subconjuntos de características com base na seguinte hipótese: **"Os bons subconjuntos de características contêm aquelas que são altamente correlacionados com a classe, mas que não estão correlacionados entre si"** [25]. Tipicamente, o CFS usa um critério de paragem de cinco iterações consecutivas sem melhoria de resultado de avaliação.

Consistência, do inglês *Consistency* (Co) O método por Co [8], funciona da mesma forma como indicado na Figura 2.2, no entanto utiliza a função consistência como função de avaliação. A função consistência, é uma função multivariável, ou seja, avalia todo o subconjunto de características no seu todo em vez da avaliação individual. A forma de avaliação por consistência é a seguinte: (i) Definindo padrão como um tuplo de valores do subconjunto de características, existe inconsistência quando duas ou mais instâncias do conjunto de dados, exibem para o mesmo tuplo, valores da classe

diferentes. (ii) Para cada tuplo, é efetuada a contagem da inconsistência como a diferença entre o número total de ocorrências do tuplo e o máximo de ocorrências com determinado valor da classe. (iii) A taxa de inconsistência de subconjunto de características é dado como a soma de todas as contagens de inconsistência dividida pelo total de ocorrências. Como critério de paragem, um subconjunto de características é considerado consistente quando a sua taxa de inconsistência é inferior a determinado limiar.

Minimal-Redundancy-Maximal-Relevance criterion (mRMR) O mRMR [17, 31] funciona num processo a um ou dois passos. (i) O primeiro passo, correspondente ao mRMR, funciona como indicado na Figura 2.2, onde é utilizado um gerador sequencial de subconjuntos e uma função de avaliação heurística mRMR que combina a medida MI entre as características e a classe (relevância) e a medida MI entre as características (redundância). O subconjunto é inicializado com $S = \{X_k\}$ em que X_k corresponde à característica com o máxima informação mútua com a classe Y, sendo a informação mútua $I(Y;X_k)$ definida por (2.8). Cada iteração adiciona a S uma característica adicional X_k que maximize a função de avaliação definida por (2.9) [2]. (ii) Em algumas implementações, pode ainda existir um segundo passo, em que, sobre o subconjunto de características resultante de (i) é utilizado um método de FS mais elaborado tipo *wrapper* para a seleção final do subconjunto de características.

$$I(Y;X_k) = H(Y) - H(Y|X_k) \tag{2.8}$$

$$I(Y;X_k) - \frac{1}{|S|} \times \sum_{X_i \epsilon S} I(X_k;X_i) \tag{2.9}$$

EFS-MI: *an ensemble feature selection method for classification* O princípio base do EFS-MI [18] está baseado na seguinte afirmação: "*From our empirical study, it has been observed that use of a single filter often fails to provide consistent performance on multiple datasets. So different filters can be ensembled to overcome the biasness or limitations of the identical classifiers and to provide consistent performance over a wide range of applications*" [18]. O algoritmo funciona em dois passos: primeiro, é utilizado um *ensemble* de F filtros, para, cada um selecionar um subconjunto de k características ordenadas inversamente por relevância. Segundo – São verificadas quais as características a ocupar as primeiras posições de cada subconjunto, se for a mesma a ocupar uma dessas posições, esta é selecionada, se não for a mesma, é selecionada para essa posição, a característica mais relevante e menos redundante, ou seja, MI elevado entre as características e a classe e

MI inferior a um valor α entre características.

Ensemble Feature Ranking (EFR) O EFR [37] é mais um exemplo desses *frameworks*, o qual, sendo o algoritmo base na implementação deste trabalho, é apresentado em detalhe no Anexo A. Neste algoritmo, a medida de avaliação das características utilizada, tem por base a soma das medidas parciais resultantes da aplicação dum conjunto de filtros (*ensemble*) sobre um conjunto de partições aleatórias retiradas do conjunto de dados. Cada medida parcial, resulta numa lista com os valores relativos da relevância de cada uma das características. De modo a se poder somar os resultados, de diferentes naturezas, obtidos pelos diferentes filtros e assim determinar o FR final, é utilizada a função W a qual atribui pesos às características em função das posições de *ranking* resultantes das medidas de filtragem. O FR final, corresponde à lista das características ordenadas inversamente pelo valor dos pesos. A seleção dum subconjunto de características a partir do resultado do FR, é obtido pela indicação do número de características pretendido.

O EFR original, representado no Algoritmo 2 do EFR (Anexo A), caracteriza-se pela combinação de $F \times S$ medidas parciais por filtragem (F filtros e S partições de n_p instâncias). Considerando que os filtros utilizados são funções uni-variável, estes apresentam complexidade $O(n)$ com $n =$ número de instâncias do conjunto de dados. Num conjunto de dados com dimensão $n \times d$ (n instâncias e d características), considerando os parâmetros F e S como constantes e com valores muito inferiores a n e d, podemos simplificar a ordem de complexidade do tempo de execução do pré-processamento EFR como sendo $O(n_p d)$.

Em relação ao desempenho, o EFR atenua o efeito dum elevado número de instâncias (n), utilizando um conjunto de partições com um número reduzido de instâncias (n_p) e atenua o efeito dum elevado número de características (d), utilizando um conjunto de filtros de baixa complexidade.

Em relação ao tipo de dados, o EFR está desenhado para o pré-processamento de conjuntos de dados com características numéricas e classe binária. Como método de FR, o EFR não deteta redundância entre características, pouco indicado para conjuntos de dados de elevada redundância. O EFR original não generaliza a função W para diferentes tipos de conjuntos de dados.

3

Enhanced Ensemble Feature Ranking

Este capítulo descreve o algoritmo de seleção de características **Enhanced Ensemble Feature Ranking (EEFR)**. Mantendo como base as características essenciais do método EFR e dos métodos de FS por filtragem (Seção 2.2.2), propomos neste capítulo, um conjunto de novas características / funcionalidades a incluir no EEFR. Estas novas características são apresentadas neste capítulo como hipóteses de melhoria do algoritmo, as quais são testadas no Capítulo 4.

O EEFR é um algoritmo de FS por filtragem baseado na Avaliação de Características, do inglês *Feature Ranking* (FR), ou seja, obtenção duma lista ordenada por ordem decrescente de relevância, de todas as características ou, a partir daí, dum subconjunto das k primeiras características. Como objetivos principais, o algoritmo EEFR deve ser o mais independente possível do tipo de conjunto de dados a pré-processar e deve selecionar um subconjunto das características que torne o processo de DM o mais eficiente possível, independentemente do algoritmo de classificação utilizado. Tendo como alvo principal a avaliação e seleção de características em conjuntos de dados de dimensionalidade elevada, o EEFR deve ser de baixa complexidade de modo a garantir um bom desempenho neste tipo de conjuntos de dados. Outro objetivo a ter em conta na implementação do EEFR, é o da automatização do pré-processamento de FS. Sem eliminar ou reduzir a possibilidade de otimização por reparametrização, ou seja, o EEFR deve ser o mais automático possível utilizando a configuração por omissão.

Tal como referido na Seção 2.2, a avaliação de características FR tem dois problemas

face a outros métodos de FS do tipo *wrapper*, mais completos mas também mais pesados do ponto de vista de processamento: (i) O primeiro é que não detetam / eliminam redundância entre características, (ii) o segundo é que não garantem a seleção do melhor subconjunto de características para um determinado classificador. O EEFR, como variante dos métodos FR, não resolve o primeiro ponto por si só. No entanto, este problema pode ser minimizado, combinando o EEFR com um processo de Eliminação de Redundância, do inglês *Redundancy Removal* (RR), o qual por utilizar métodos estatísticos simples, apesar de ter complexidade $O(d^2)$ é rápido (ver testes na seção 4.3.3), não comprometendo em demasiado o princípio de simplicidade e rapidez dos métodos por filtragem. A segunda desvantagem é reduzida com a alteração do algoritmo, introduzida na Seção 3.2, onde foi adicionado um novo mecanismo que calcula automaticamente um subconjunto das k melhores características. Como veremos nos testes efetuados, o subconjunto de características obtido desta forma, apesar de não garantir o melhor subconjunto de características para um determinado classificador, é um bom ponto de partida, provavelmente com um valor de k mais elevado, mas que não compromete a capacidade preditiva dos classificadores. Com este mecanismo de seleção automática do subconjunto das k melhores características, pretendemos obter resultados semelhantes a outros métodos de FS mais complexos, sem perda de desempenho, aspeto fundamental a considerar em conjuntos de dados de dimensionalidade elevada. A funcionalidade de eliminação de características redundantes, não foi integrada neste algoritmo, de modo a mantermos o alinhamento com o algoritmo original e com os métodos FR.

3.1 Classificação das características FR

Tal como referido no Apêndice A, o EFR e agora o EEFR, caracterizam-se pela combinação múltipla de $F \times S$ medidas parciais por filtragem, utilizando um *ensemble* de F tipos de filtros e S partições de instâncias do conjunto de dados. A avaliação final atribuída a cada característica é calculada pela soma dos pesos parciais de cada combinação, atribuídos pela função / vetor de peso W. Dadas as diferentes medidas de filtragem do *ensemble*, gerarem resultados de naturezas diferentes, é utilizada a função W para a normalização dos diferentes resultados das medidas de forma a torná-los adicionáveis. A função W atribui um peso à característica em função do seu *rank* em cada medida de filtragem parcial. Em cada medida de filtragem, define-se por *rank* de uma determinada característica, a sua posição relativa por ordenação inversa dos valores obtidos, ou seja, o $rank_k = 1$ é atribuído à característica k que obteve o maior valor nessa medida de filtragem.

Tal como referido no apêndice A, o algoritmo EFR, apresenta como função de peso W, uma função descontínua constituída por dois segmentos lineares, a qual privilegia as características nas primeiras posições do *ranking*, mas que obriga a testes de otimização de forma a encontrar o ponto adequado de viragem da tendência da curva. A Figura 3.1 da esquerda, apresenta a forma da curva da função $peso = W(rank)$ original para um exemplo dum conjunto de dados de dimensionalidade $d = 20$ características. A Figura 3.1 da direita, uma curva contínua alternativa (equação 3.1), a qual também favorece as primeiras posições do *ranking* e resolve o problema anterior, eliminando a necessidade de definir um ponto de viragem da curva. Entre as várias alternativas, a determinação da curva contínua alternativa, foi efetuada, heurística e experimentalmente, tendo em conta os seguintes critérios: (i) semelhança com a curva da função W original, (ii) eliminação da necessidade de atribuição de valores arbitrários, (iii) benefício das características de maior relevância, primeiras posições do *ranking*, (iv) simplicidade de cálculo.

$$W(rank) = \frac{(d - rank + 1)\sqrt{(d - rank + 1)}}{\sqrt{d}} \tag{3.1}$$

onde d =no. de características e $rank \in [1...d]$.

O peso final atribuído a cada característica k é assim, definido por:

$$finalWeight_k = \sum_{i=1}^{F \times S} W(rank_{ik}) \tag{3.2}$$

onde $d =$ no. de características; $W(rank)$ definido por (3.1); $rank_{ik}$ corresponde à posição, atribuída à característica k, na medida de filtragem parcial i: $partialWeight_i$, com i a variar entre 1 e $F \times S$ (F filtros e S partições de dados).

De forma a verificar o comportamento do FR e FS, usando a curva dada por (3.1), foram efetuados diversos conjuntos de testes, cada um dos testes composto pela seleção do subconjunto das características, divisão em duas partições de treino e de teste, classificação e medição da qualidade do modelo de dados obtido. Cada um dos conjuntos de testes é efetuado sobre, 6 conjuntos de dados diferentes e utilizando 3 tipos de classificadores, num total de 18 testes. Os conjuntos de dados, numéricos com classe binária, foram selecionados entre os mais comuns, sem critério especial para além de tentar cobrir um leque alargado de dimensionalidades. O conjunto de classificadores foi selecionado entre os mais conhecidos, sem critério especial, para além de terem como objetivo a construção de modelos de classificação binária. O resultado de cada conjunto de testes corresponde aos dados estatísticos das 18 medidas de *Area Under the*

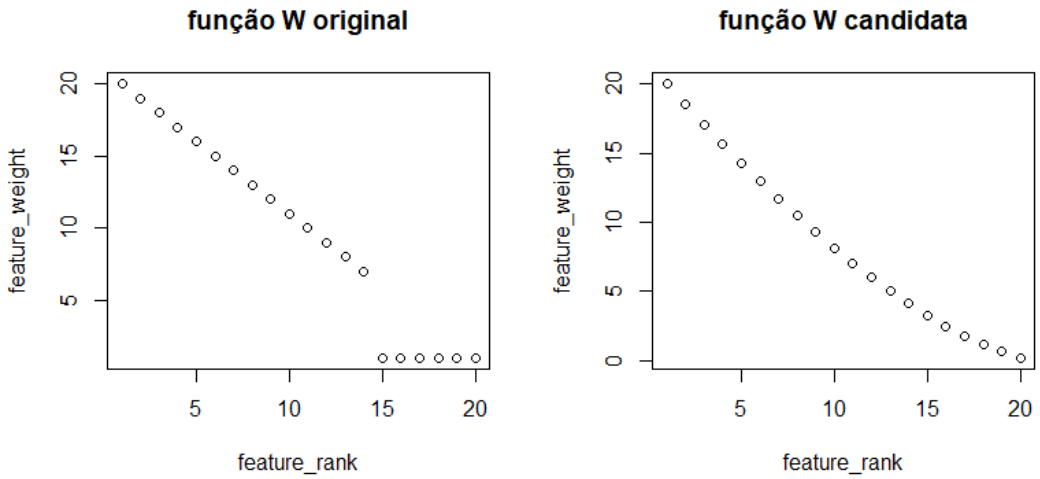


Figura 3.1: Função de Pesos W

ROC Curve (AUC) e *Balanced Error Rate (BER)* obtidas. De forma a verificar o comportamento da curva (3.1) foram comparados os resultados de 3 conjuntos de 18 testes, por cada uma das seguintes seleções de características: (i) sem qualquer eliminação de características, (ii) subconjunto de características obtido por FS utilizando uma simulação do EFR e, (iii) subconjunto de características obtido por FS utilizando EEFR com a função de peso W contínua, não linear, definida por (3.1).

Uma vez que o EFR não seleciona automaticamente um subconjunto de características, de forma a tornar o conjunto dos 18 testes EFR exequível, utilizámos uma simulação do mesmo, recorrendo ao algoritmo EEFR com uma função de peso W linear ($W(rank) = d - rank + 1$).

Apesar do tema da determinação do FR, medição e ordenação das características por ordem de relevância (Seção 3.1), ser distinto do tema do FS, determinação do subconjunto das melhores características (Seção 3.2), optámos por efetuar os testes conjuntos, apresentados na Seção 4.3.

3.2 Subconjunto das k melhores características FS

Na secção anterior, analisámos a problemática do FR, ou seja, obtenção da lista de todas as características do conjunto de dados ordenadas por ordem decrescente da sua relevância. Nesta seção, analisamos um novo mecanismo que, a partir do FR, calcula de forma automática, um subconjunto com as k melhores características. Após as características ordenadas por relevância e da seleção das primeiras k características, a

otimização por variação do número de características a selecionar em torno de k, é algo subjetivo, uma vez que depende, do compromisso entre a taxa de redução do número de características pretendida e da maximização da qualidade do modelo medido por uma determinada métrica. Tal como podemos verificar nos testes, nalguns dos conjuntos de dados, selecionando as primeiras características da lista ordenada, o k pode variar sem grande alteração da qualidade do modelo. A introdução do cálculo automático de k, no algoritmo EEFR, não tem como ambição a determinação do subconjunto de características ótimo, mas sim a obtenção dum subconjunto de características, de forma simples e rápida, que não comprometa a qualidade do modelo a utilizar. O subconjunto de características obtido desta forma, serve também como ponto de partida para selecionar outros valores de k mais adequados, por afinação de parâmetros, com um ganho imediato de desempenho pela redução no tempo de afinação do pré-processo de FS.

O mecanismo de seleção utilizado é baseado nos seguintes princípios:

1) Durante a execução do algoritmo EEFR[1], à medida que o número de iterações (medidas de filtragem parciais) vai aumentando de 1 até $F \times S$, por ação da função de peso W, as características menos relevantes vão sendo distribuídas nas últimas posições do *ranking* com valores tendencialmente semelhantes (Figura 3.2 esquerda). Esta situação, faz aumentar a distância vertical entre os pesos médios obtidos pelos dois grupos de características, os mais e os menos relevantes, permitindo a sua separação. Para este efeito, contribuem as diferenças entre os filtros do *ensemble* e as partições de instâncias retiradas aleatoriamente do conjunto de dados.

2) Uma vez que a função de peso W se limita a efetuar permutações dos valores definidos por (3.1) em função do *rank* das características em cada filtragem parcial, a soma total dos pesos finais das características de um conjunto de dados de dimensionalidade d, equação (3.3), só depende de d e é constante para um determinado conjunto de dados.

$$\sum_{k=1}^{d} finalWeight_k = Constante \tag{3.3}$$

onde $finalWeight_k$ é definido em (3.2)

3) No caso limite em que todas as características apresentam igual relevância, ou seja, todas as características com o mesmo valor de peso, não se justifica a seleção descriminada das características, por não existirem melhores nem piores. Assim, definimos

[1]O pseudocódigo pode ser encontrado no Algoritmo 1

esse valor como sendo o valor mínimo de relevância a considerar para fins de seleção de características (equação (3.4)).

$$peso_{min} = \frac{1}{d} \times \sum_{k=1}^{d} finalWeight_k \tag{3.4}$$

Definimos $peso_{min}$, como o valor do peso final, abaixo do qual podemos considerar as características como irrelevantes. A utilização deste critério, resulta na seleção dum subconjunto de características com as k melhores características, calculada automaticamente, tal como indicado na linha 15 do pseudocódigo da função *EnsenbleFeatureRanking* do Algoritmo 1.

De notar que a eficiência, desta forma simplificada de selecionar as características, está diretamente associada ao EEFR (com múltiplas medidas de filtragem e múltiplas partições de instâncias) e à aleatoriedade dos *ranks* atribuídos às características irrelevantes, pelas medidas parciais, durante as $F \times S$ iterações. Essa aleatoriedade dos *ranks* é gerada pelo facto das medidas serem efetuadas com filtros diferentes e sobre partições aleatórias do conjunto de dados. Um número $F \times S$ elevado de medidas e o tipo de filtros diversificado, favorecem a eficiência do algoritmo. Para além da aleatoriedade associada à utilização de diversos filtros e diversas partições, o código foi alterado de forma a fomentar aleatoriedade (distribuindo aleatoriamente as características antes de cada medição parcial e o embaralhar dos *ranks* em caso de igualdade de valores).

Como forma visual demonstrativa deste mecanismo, a Figura 3.2 representa dois exemplos de distribuição final dos pesos de todas as características. O gráfico da esquerda é relativo a um conjunto de dados artificial de classe binária e com 200 instâncias e 20 características numéricas. Às características foram atribuídos valores aleatórios (distribuição uniforme), 3 delas com relevância, por adição de correlação à classe ($F_i = F_i + \alpha \times class$, característica F_i e coeficiente de correlação α). O gráfico da direita é relativo a um conjunto de dados da mesma dimensão do anterior, mas com todas as 20 características sem relevância. Quanto maior for o número $F \times S$ de iterações, maior aleatoriedade na atribuição dos *ranks* parciais, mais a curva de distribuição das características não relevantes, tenderá para uma reta horizontal, aumentando o número de características abaixo do valor limite $peso_{min}$. Apesar do número reduzido de iterações de filtragens $F \times S$ deste exemplo, podemos observar no lado esquerdo da Figura 3.2, a separação vertical dos pesos das 3 ou no máximo 4 características mais relevantes, o que não acontece na distribuição do lado esquerdo da Figura 3.2.

O pacote R `FSelector`, que utilizamos na implementação do EEFR, inclui 3 formas de seleção dum subconjunto de características a partir do FR: `cutoff.k(attrs, k)`,

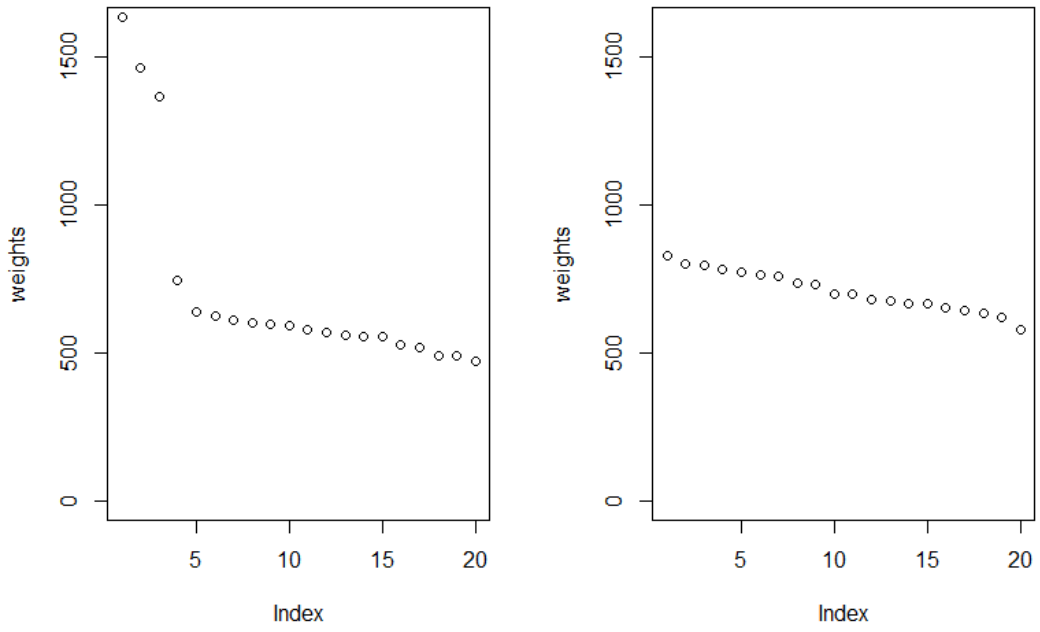


Figura 3.2: FR sobre um conjunto de dados artificial por EEFR

seleciona as k primeiras, `cutoff.k.percent(attrs, x)`, seleciona as k primeiras ($x = \frac{k}{d}$) e `cutoff.biggest.diff(attrs)`, seleciona o grupo das primeiras até àquela que apresente maior degrau de decréscimo de relevância. Por analogia, denominámos esta nova forma de seleção automática do subconjunto de características a partir do FR por `cutoff.by.contrib(attrs)`, função opcional incluída no EEFR (ver Algoritmo 1).

Os testes conjuntos do EEFR, medição e ordenação das características por ordem de relevância e seleção automática do subconjunto das k melhores características, são apresentados na Seção 4.3.

3.3 *Ensemble* de Filtros

Tal como referido na Seção 2.2, existem inúmeros métodos de filtragem, não existindo um melhor para todas as situações. O objetivo desta secção, é a de encontrar um *ensemble* de filtros o mais abrangente possível, a diferentes tipos de conjuntos de dados, a diferentes classificadores e a diferentes tipos de medida a maximizar, mantendo o desempenho adequado para conjuntos de dados de elevada dimensionalidade d. De entre os muitos artigos comparativos entre vários métodos de filtragem (e.g. [2, 29]), o

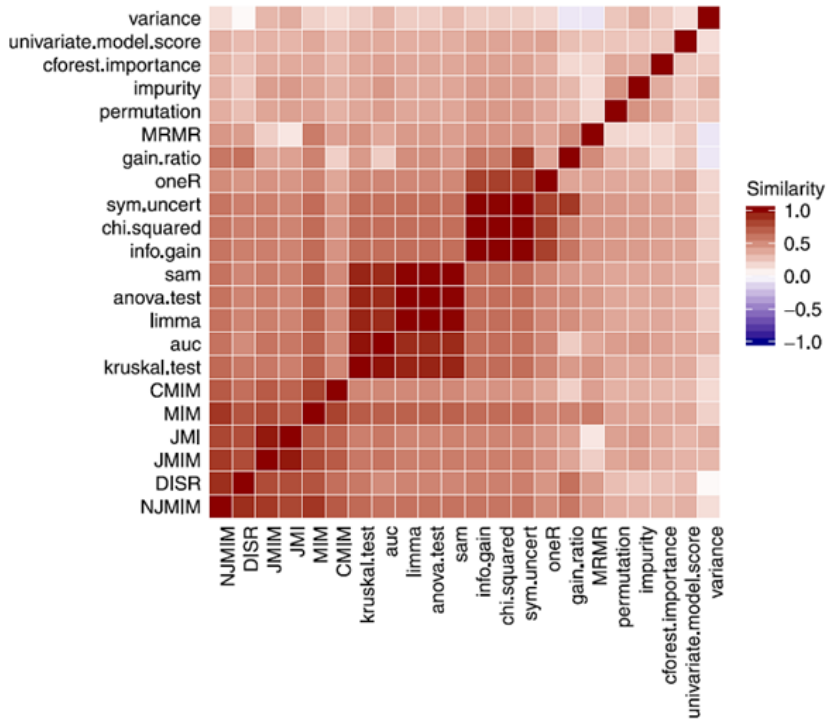


Figura 3.3: Similaridade entre métodos de filtragem[2]

artigo de *Andrea Bommert* [2] compara 22 métodos de filtragem e inclui um mapa comparativo das similaridades, tal como indica a Figura 3.3. Nesta figura, as zonas mais escuras, indicam grupos de filtros com elevada semelhança entre eles.

O algoritmo original EFR apresenta o *ensemble* de filtros por omissão constituído por: (i) *information.gain*, (ii) *gain.ratio*, (iii) *symmetrical.uncertainty*, (iv) *chi.squared* . Como se pode observar na Figura 3.3, os métodos (i), (ii) e (iii) são todos baseado em informação mútua e muito semelhantes entre eles. Este *ensemble* de filtros pode servir para o pré-processamento de um conjunto de dados específico, tal como indicado no artigo original, mas como veremos nos testes efetuados na Seção 4.2 existem outros conjuntos de filtros mais abrangentes a tipos de conjuntos de dados diferentes. Uma maior diversificação dos tipos de filtros no *ensemble*, espera-se que aumente a abrangência do EEFR, ou seja, que ajude à construção de modelos menos dependentes do tipo de conjunto de dados ou do tipo de classificador utilizado.

Dando seguimento aos tipos de filtros analisados na Seção 2.2 e de forma a analisar os comportamentos de diversos *ensembles* e a estabelecer um *ensemble* de filtros a utilizar por omissão, mais adequado, efetuaram-se testes do EEFR com diversos *ensembles* de filtros, sobre diversos conjuntos de dados. Os testes estão descritos na Seção 4.2.

3.4 Características diferenciadoras do EEFR

Dentro do âmbito dos métodos de avaliação de características FR e alinhado com o EFR, tentámos implementar um novo algoritmo, o EEFR, mais automático, mais genérico a diferentes tipos de dados e com o desempenho necessário ao pré-processamento de conjuntos de dados de dimensionalidade elevada. À principal característica do EFR, utilização dum duplo *ensemble* de filtros e de partições na avaliação de características, adicionámos um conjunto de novas características com as quais pretendemos atingir os objetivos para o EEFR. Assim, a utilização da nova função W proposta, a seleção automática do subconjunto das k melhores características, a utilização dum conjunto de filtros mais diversificado e um código mais eficiente, apresentam-se como hipóteses de melhoria do algoritmo cuja implementação e verificação se inclui no Capítulo 4.

Algoritmo 1 Enhanced Ensemble Feature Ranking (EEFR)

Input:
data - The dataset for feature ranking, represented as a matrix;
N - The number of iterations;
sampleSize - The size of the sample for each iteration;
algo - A vector of feature ranking functions (filters);
k - cutOff: k best features ordered by -rank, k=0 (all features), k=-1 (automatic k calculation)

Output: Feature names vector ordered by *-rank*

function ENSEMBLEFEATURERANKING(*data*, *N*, *sampleSize*, *algo*, *k*)
  $W \leftarrow vector(nColumns(data])$ ▷ a vector initialized with the weights for each rank position
  $localWeights \leftarrow vector(length(algo))$ ▷ A matrix of weights of each filter measurement
  for each fun in *algo* **do** ▷ m x n measurements, m filter functions, n samples
    for 1 to N **do**
      $randomData \leftarrow SamplingWithReplacement(data, sampleSize)$
      $localWeights$.add($fun(randomData)$)
    end for
  end for
  $weights \leftarrow calculateWeightedRank(localWeights, W))$
  $rank \leftarrow$ names($dataset$)[$order(rank(-weights))$]
  switch k : **do**
    case (> 1 and $\leqslant$ length(W)) : *return* cutoff.k($weights, k$)
    case 0 : *return* cutoff.k($weights$, lenght(W))
    case *other* : *return* cutoff.by.contrib($weights, W$, length($algo$) $* N$)
end function

function CALCULATEWEIGHTEDRANK(weights,W)
  $weightsRanks \leftarrow$ vector(length(W)) ▷ A matrix of ranks of each filter measurement
  $weightsTotal \leftarrow$ vector(length(W)) ▷ A vector with the final weight of each feature
  set to 0 in all $weightedRank[i]$
  for $i \leftarrow 1$ to $nRows(weights)$ **do** ▷ replace measurements weights by measurements ranks
    $weightsRanks[i] = rank(-weights[i], randomties)$
  end for
  for $j \leftarrow 1$ to $nColumns(weights)$ **do** ▷ sum of feature ranks weighted by W
    $weightsTotal[j] \leftarrow$ Sum(W indexed by $weightsRanks[j]$)
  end for
  return $weightsTotal$
end function

function CUTOFF.BY.CONTRIB($weights, W, measurementsNum$)
  $minContrib \leftarrow measurementsNum *$ Sum(W)/length(W)
  return cutoff.k($weigths[\geqslant minContrib]$)
end function

4

Implementação do EEFR num pacote R e Testes

Este capítulo descreve a implementação do algoritmo EEFR e os testes relativos às diferentes funcionalidades indicadas no Capítulo 3. A Seção 4.1 descreve a implementação do algoritmo EEFR num pacote em linguagem R. As seções 4.2 a 4.5 descrevem os testes e resultados, de cada uma das funcionalidades sugeridas no Capítulo 3: A Seção 4.2 trata dos testes efetuados sobre diferentes conjuntos de filtros e a determinação do melhor conjunto a utilizar por omissão. A Seção 4.3 inclui os testes relativos à avaliação de características utilizando a nova função W proposta e os testes relativos à seleção automática do subconjunto das k melhores características. A Seção 4.4 inclui testes comparativos do EEFR com outros métodos de FS. A Seção 4.5 conclui com a descrição dos testes da configuração final do EEFR a utilizar por omissão no pré-processo de FS sobre um determinado conjunto de dados.

4.1 Implementação do EEFR num pacote em R

Como implementação do algoritmo EEFR, em linha com as indicações do capítulo 3, foi criado o pacote na linguagem R, `EEFRanking`. O pacote R, segue as orientações do *CRAN - The Comprehensive R Archive Network* para implementação de pacotes em R [34] e respetiva documentação com base no *Roxygene2* [33].

No desenvolvimento foi utilizado o R Studio (*integrated development environment (IDE) for R*) e o R versão 3.6.2. O código do EEFR e respetivo ambiente de teste, está distribuído em três projetos R principais:

1. `EEFRanking package`, objeto principal e produto final deste trabalho, que contém a implementação do EEFR;

2. `pEEFRanking package`, pacote alternativo ao `EEFRanking`, com a mesma funcionalidade mas utilizando processamento paralelo, indicado para conjuntos de dados de dimensionalidade muito elevada;

3. `EEFRTests`, o qual não fazendo parte do trabalho, inclui todos os módulos relativos aos testes realizados, tal como apresentados no capítulo 3. Este projeto inclui ainda os módulos `importData` que são utilizados na importação, análise e limpeza dos diferentes conjuntos de dados utilizados nos testes.

O R, orientado para a manipulação, estatística, análise e visualização de dados, está em constante evolução, em grande parte devido à dinâmica duma comunidade que continuamente cria e implementa extensões para o mundo R. Uma forma de contribuir para essa comunidade está na implementação e disponibilização de novos pacotes R. A implementação dum projeto na forma de pacote R tem, entre outras, as seguintes vantagens: (i) organização do código em módulos individuais associados a funções específicas e sua integração num projeto conjunto; (ii) disponibilização fácil das funcionalidades do pacote R, após a sua publicação num repositório, CRAN ou GitHub; (iii) documentação incluída no pacote R, acessível através da instrução `help()`.

A implementação dum pacote R, duma maneira geral, segue os seguintes passos [34]: (i) Verificar a inexistência do nome do pacote R (*EEFRanking* e *pEEFRanking*) no CRAN e GitHub utilizando o pacote auxiliar em `library(available)`; (ii) Estabelecer a estrutura do pacote R inicial, utilizando o *template* do RStudio (`File > New Project.. > New Directory > R Package`) tendo em conta opção `GitHub`; (iii) Completar o ficheiro `DESCRIPTION` com os dados do projeto e com a indicação dos pacotes externos a importar (`Imports: FSelector`); (iv) Activar as opções, gerar documentação `Roxygene` e verificar código `Check Package (-as-cran)` utilizando o *template* do RStudio (`Build > Configure Build Tools...`); (v) Organizar o código por módulos funcionais específicos e inclui-los na diretoria R; (vi) Utilizando o formato *Roxygene2* [33], inserir em cada um dos módulos de código, a descrição da função, os parâmetros de entrada e de saída e se for o caso, incluir o código do exemplo de demonstração; (vii) Incluir o conjunto de dados utilizado na demonstração na diretoria

data; (viii) No RStudio, criar o pacote e instalar (`Build > Clean and Rebuild`); (ix) Publicar o pacote R no GitHub.

Após o pacote `EEFRanking` (ou `pEEFRanking`) estar disponível no CRAN ou GitHub, utilizando o R, este pode ser instalado numa determinada máquina com a instrução `install.packages()` ou `install.github()`. Qualquer aplicação de DM em R, que pretenda pré-processar os dados com EEFR, deve incluir no início do código, a instrução `library(EEFRanking)` e invocar o algoritmo EEFR a partir da instrução `ensemble.feature.ranking()`. A mesma situação relativamente ao pacote R alternativo `library(pEEFRanking)` para execução paralela.

4.1.1 Pacote *EEFRanking*

O `EEERanking` utiliza o pacote externo `FSelector` [13] do repositório CRAN, para a disponibilização das funções de filtragem utilizadas no *ensemble*. A documentação está acessível pela instrução `help(package = "EEFRanking")` e uma demonstração de pré-processamento por EEFR, sobre um conjunto de dados artificial, pode ser invocada pela instrução `example("ensemble.features.ranking")`. O repositório do `EEFRanking` pode ser encontrado em: `https://github.com/matpato/EEFR.git`

A operação de pré-processamento FS ou de FR sobre um determinado conjunto de dados, é executada chamando a função `ensemble.feature.ranking()` a qual utiliza os seguintes parâmetros de entrada:

conj. de dados: `data.frame` com as **d** colunas das características do tipo numérico e a coluna da classe do tipo categórica binária. O nome da coluna da classe deve ser `"class"`;

nTries: número S de amostras do conjunto de dados a utilizar (nTries = 10, corresponde ao valor por omissão).

nRows: número n_p de instâncias do conjunto de dados por amostra ($nRows = d/2$, corresponde ao valor por omissão);

cutOff: tipo de resultado a obter, (i) `cutOff=0`, lista FR com todas as características, ordenadas por ordem decrescente de relevância. (ii) `cutOff=k` lista FS com as **k** melhores características, com **k** atribuído pelo valor do parâmetro `cutOff` e ordenada por ordem decrescente de relevância. (iii) `cutOff=-1` vetor FS com as **k** melhores características, com **k** calculado automaticamente e ordenadas por ordem decrescente de relevância, (`cutOff=-1`, corresponde ao valor por omissão);

methods: lista das S funções de filtragem (*filtros* a utilizar). Esta implementação utiliza algumas das funções de filtragem disponíveis no *FSelector*. No entanto, podem ser utilizadas outras funções, desde que recebam como parâmetro de entrada um *data.frame* com os valores numérico das **d** características e os valores categóricos binários da classe. Como resultado, a função deve devolver um *data.frame* com uma coluna com os valores de relevância das **d** características, com os nomes das linhas correspondente aos nomes das características. O valor por omissão do parâmetro `methods` corresponde à seguinte lista de filtros: *methods = list(gain.ratio, symmetrical.uncertainty, chi.squared, random.forest.importance)*.

A utilização do EEFR no pré-processamento dum conjunto de dados na sua forma mais simples, `features = ensemble.feature.ranking(dataset)`, deverá devolver um subconjunto de características aceitável para aprendizagem de um modelo, sem necessitar de afinamentos exaustivos por alteração e teste de diferentes parâmetros. No entanto, existem casos que necessitam de afinamento dos parâmetros, tais como: Controlo do tempo de execução, caso o conjunto de dados contenha um número muito elevado de instâncias, por redução do número de linhas das partições de dados (n_p) compensado pelo aumento do número de partições (S). Redução da dimensionalidade do subconjunto de características, por manipulação do parâmetro `cutOff=k`, indicando o número de características a selecionar. Aumento da eficiência do algoritmo, aumentando o número de iterações ($F \times S$) por alteração dos parâmetros `nRows` e `nTries`.

De forma a documentar o pacote R com um exemplo de utilização, foi incluído no pacote, um conjunto de dados artificial, com 20 colunas de caraterísticas e 200 linhas de instâncias com valores aleatórios de distribuição uniforme, uma coluna com o fator classe, com valores binários aleatórios e três das características (F3, F7 e F17) alteradas de forma a estarem correlacionadas com a classe. O pacote permite, a partir da instrução `example("ensemble.features.ranking")`, executar o exemplo de teste incluído no `EEFRanking`.

A Listagem 4.1 contém um exemplo simples do código em linguagem R, de como utilizar o EEFR com os parâmetros de entrada por omissão, no pré-processamento do conjunto de dados artificial, seguido dum exemplo simples de classificação e teste. A Listagem 4.2 exemplifica a utilização do EEFR, agora com redefinição dos parâmetros de entrada. A título exemplificativo, o exemplo da Listagem 4.2 devolve a lista de características:

```
[1] "F17" "F7"  "F3"  "F5"  "F10" "F19" "F4"
```

Listagem 4.1: Exemplo de seleção de características com o EEFR

```
# Carregamento do conj. de dados
# Data.frame artificial "allDataProb" incluído no pacote EEFRanking
library(EEFRanking)
rm(list=ls(all=TRUE))
# load("data/allDataProbe.RData")
data(allDataProbe)

# Seleção de características utilizando o EEFR
set.seed(134)
features <- ensemble.features.ranking(allDataProbe)
print(features)

# Exemplo simplificado de classificação e teste
library(caret)
allDataProbe <- allDataProbe[, c(features, 'class')]
trainIndex <- createDataPartition(allDataProbe$class, p = .8, list=FALSE)
trainset <- allDataProbe[trainIndex,]
testset <- allDataProbe[-trainIndex,]
model <- train(class ~ .,data=trainset,method="gbm",verbose=FALSE)
predictions <- predict(model, testset[,features])
confusionMatrix(predictions, testset[, 'class'])
```

correspondente a 1/3 da dimensionalidade de conjunto de dados artificial, ordenadas inversamente por relevância. O exemplo Listagem 4.1 calcula automaticamente o subconjunto das características mais relevantes:

```
[1] "F17" "F3"  "F7"
```

Relativamente ao código, as principais alterações em relação ao EFR têm a ver com, a alteração da curva de pesos W, a adição da função de seleção automática das características, a substituição de alguns *loops* por funções `apply()` e `lapply()` e a alteração dos parâmetros de entrada e respetivos valores por omissão. Foi também adicionada a função `RForestImportance2` de forma a permitir a opção da variante, importância por *accuracy* do `forest.importance` do `FSelector`, no *ensemble*. O pacote R inclui os seguintes módulos de código: `EnsembleFeaturesRanking` — módulo principal do EEFR, `getRowSamples` — seleção das S partições de n_p instâncias do conjunto de dados, `weightsSampling` — aplicação duma medida de filtragem a uma partição do conjunto de dados, `rankSampling` — calcula os pesos finais das características, em função dos *ranks* obtidos em cada medida de filtragem e devolve a lista de características por ordem inversa dos pesos. `cutoff.by.contrib` — seleção automática dum subconjunto das melhores características.

Listagem 4.2: Exemplo de seleção de características com o EEFR com parametrização

```
# Carregamento do conj. de dados
# Data.frame artificial "allDataProb" incluído no pacote EEFRanking
library(EEFRanking)
rm(list=ls(all=TRUE))
# load("data/allDataProbe.RData")
data(allDataProbe)

# Seleção de características utilizando o EEFR
library(FSelector)
set.seed(134)
nRows <- nrow(allDataProbe)/4
nTries <- 20
k <- 1/3*ncol(allDataProbe)
methods <- list(gain.ratio, symmetrical.uncertainty, chi.squared)
features <- ensemble.features.ranking(allDataProbe, nRows=nRows, nTries=
    nTries, cutOff=k, methods=methods)
```

4.1.2 Pacote *pEEFRanking*

Sendo o algoritmo EEFR vocacionado para o pré-processamento de seleção de características em conjuntos de dados de dimensionalidade elevada, a possibilidade de processamento em paralelo é especialmente importante por razões de desempenho, por permitir utilizar toda a capacidade da máquina ou das máquinas. Assim, criámos um novo pacote `pEEFRanking` em tudo idêntico ao `EEFRanking` mas que extrai algumas funcionalidades de paralelização do pacote `future` [35]. O repositório do `pEEFRanking` pode ser encontrado em: `https://github.com/matpato/EEFR-parallel.git`

O pacote `pEEERanking` utiliza os pacotes externos `FSelector` [13] para a disponibilização das funções de filtragem utilizadas no *ensemble* e o `future` [35] para a funcionalidade de processamento paralelo. O pacote `future` possibilita iniciar partes do código com processamento paralelo, nas suas diversas formas: `clustering`, `multicore`, `multisession`, e voltar à forma sequencial após terminar. O pacote `future_apply` [36] inclui a coleção de funções do tipo `future_apply()` as quais utilizam as funcionalidades disponibilizadas pelo `future`, permitindo o processamento em paralelo. As funções `apply()`, `lapply()`, `sapply()`, ..., típicas na linguagem R, são mapeáveis univocamente por funções da coleção `future_apply`, sem alteração funcional e com propriedades de processamento paralelo. Utilizando o `future`, a ativação de processamento paralelo numa parte do código, é iniciada pela instrução `plan("multisession")` e terminado pela instrução `plan("sequential")`.

O EEFR, na sua execução, por cada um dos F filtros do conjunto, executa S medidas

de filtragem sobre as S diferentes partições retiradas aleatoriamente do conjunto de dados. Esta parte do código do módulo `weightsSampling`, é a que consome mais tempo de execução pelo que optámos pela sua execução em paralelo.

O `pEEFRanking` tem um *setup* mais pesado que o `EEFRanking`, pelo que a sua utilização só se justifica quando o processamento do `EEFRanking` não se executa em tempo útil. A opção tomada foi a de criar dois pacotes R independentes sem e com processamento paralelo, tornando a opção base mais ligeira, sem necessidade de importar o pacote R `future`. No entanto é possível ter um único pacote R com ambas as funcionalidades de execução, selecionáveis por parametrização ou de forma automática em função da dimensão do conjunto de dados em termos de número de instâncias e de características.

A documentação e demonstração sobre a utilização do `pEEFRanking`, está incluída no pacote R e acessível a partir das instruções `help()` e `example()`.

4.1.3 Desempenho do *EEFRanking* versus *pEEFRanking*

Recorrendo ao mesmo ambiente de testes utilizado na Seção 4.3, fomos calcular e comparar os tempos de execução do pré-processamento EEFR utilizando o pacote original `EEFRanking` e a versão com ambiente de execução paralela `pEEFRanking`. Dos resultados obtidos e apresentados na tabela 4.1, verifica-se que para conjuntos de dados com dimensionalidade a partir da centena de características (conjunto de dados: *Arcene, Dexter e CancerMass*), se justifica a utilização da versão paralela. Nestes conjuntos de dados e para um HW (computador pessoal) limitado em termos de recursos, obtém-se uma redução dos tempos de execução entre 50 a 60%.

Como referido na seção anterior, a zona de execução paralela incide na execução das medidas de cada uma função de filtragem sobre um número parametrizável de partições de instâncias do conjunto de dados original. Esse facto, torna a versão paralela especialmente interessante, em termos de desempenho, quando o número de instâncias (n_p) ou o número de partições (S) são elevados. Ou seja, caso o número de partições utilizado pelo EEFR seja da ordem das dezenas, a redução dos tempos de processamento será substancialmente superior ao obtido, uma vez que o processamento de cada partição é efetuado em paralelo e não em sequência. Com outro tipo de *Hardware* (HW) menos limitado, a redução do tempo de processamento seria na ordem de S (número de partições de dados). Por outro lado, pelos resultados obtidos e apresentados na tabela 4.1, verifica-se que para os conjuntos de dados com dimensionalidade reduzida (conjunto de dados: *SPAM, DBC e Slice*), a solução de processamento paralelo não

Tabela 4.1: Tempos de execução (s) do pré-processamento FS

conj. de dados	trainset d/n	EEFRanking elapsed time	pEEFRanking elapsed time
SPAM	56/3066	100.44	70.94
DBC	30/379	4.67	44.67
Arcene	10000/100	2165.44	934.81
CancerMass	117/68195	6074.91	3071.53
Dexter (esparso)	20000/300	1925.71	750.94
Splice	60/1000	27.89	50.53

se mostra eficiente. Os tempos de execução ou aumentam ou são semelhantes aos da versão sequencial.

Durante a execução dos testes verificou-se que, no caso específico da medida de filtragem `rforest.importance`, contrariamente ao referido na documentação do pacote R `future_apply`, o resultado da medida difere ligeiramente entre a versão original e a paralela. Esse facto é conhecido e referido na Seção 5.1.

4.2 Testes EEFR sobre *ensemble* de Filtros

De forma a analisar os diferentes comportamentos do EEFR com diversos conjuntos de filtros, efetuámos um conjunto de testes do EEFR utilizando os *ensembles* identificados na Tabela 4.2, sobre um conjunto diversificado de conjuntos de dados. Relativamente à Tabela 4.2, o conjunto E1, tem a ver com o facto deste ser o utilizado por omissão no EFR; E2 tem a ver com uma alternativa a E1, reduzindo a semelhança dos primeiros 3 filtros do conjunto e a dependência em medidas baseadas em entropia de informação de 75% para 50%; E3, E4 e E5, têm a ver com a avaliação individual destes 3 filtros, pertencentes a diferentes grupos de semelhança.

O EEFR configurado com cada um dos conjuntos de filtros, E1, E2, E3, E4 e E5, foi testado sobre vários conjuntos de dados e vários classificadores tal como indicado na

Tabela 4.2: Lista de Ensembles de Filtros

Ensembles de Filtros	Teste
information.gain, gain.ratio, symmetrical.uncertainty, chi.squared	E1
gain.ratio, symmetrical.uncertainty, chi.squared, forest.importance	E2
gain.ratio,	E3
chi.squared,	E4
forest.importance	E5

Tabela 4.3: Lista de testes de Ensemble de Filtros

conjunto de dados	Dim d	SVM	rForest	logReg	sampling	E1 d	E2 d	E3 d	E4 d	E5 d
SPAM	56	X	X	X	Original	27	28	26	27	27
DBC	30	X	X	X	Original	16	15	16	16	18
Arcene	10000	X	X		Original	4199	4280	4202	4195	4526
Bankrrupcy	93	X		X	OverS	49	45	50	46	50
CancerMass	117	X	X	X	UnderS	56	53	57	54	53
Dexter (sparce)	20000	X	X		Original	3998	3890	3998	3998	3154(*)
Splice	60	X	X	X	Original	14	19	14	14	23

Tabela 4.3. O critério de escolha dos conjuntos de dados e classificadores teve unicamente a ver a diversificação. Como medida comparativa, utilizámos os indicadores de qualidade dos modelos obtidos, *AUC* e o *BER*. Os testes foram efetuados utilizando o algoritmo EEFR com a opção de seleção automática das *k* melhores características (Seção 3.2). Os resultados obtidos de redução de dimensionalidade pelo EEFR para cada conjunto de filtros, estão indicados pelas colunas E1d a E5d na Tabela 4.3.

Dado verificarmos a existência duma relação da eficiência dos modelos de dados produzidos, entre os conjuntos de filtros utilizados e a dimensionalidade dos conjuntos de dados, dividimos os resultados estatísticos dos testes por 3 grupos diferentes de dimensionalidades: (i) todos os conjuntos de dados da Tabela 4.2; (ii) conjuntos de dados, com dimensionalidade inferior ou igual a 100 características; (iii) conjuntos de dados, com dimensionalidade superior a 100 características. A Figura 4.1 representa os dados estatísticos dos testes efetuados sobre todos os conjuntos de dados. Como se pode visualizar, não existem grandes diferenças entre os vários conjuntos E1 a E5: as medianas e os primeiros quartis relativos ao *AUC* são muito semelhantes entre si.

As Figura 4.2 e Figura 4.3, representam os dados estatísticos dos testes efetuados, separando os conjuntos de dados com (i) dimensionalidade <= 100 e (ii) dimensionalidade > 100. Neste caso, verificamos que: (i) tendencialmente o teste E2 que utiliza o conjunto de filtros: *gain.ratio, symmetrical.uncertainty, chi.square, forest.importance*, apresenta os melhores ressultados nos conjuntos de dados de elevada dimensionalidade, ou seja, o *AUC* mais elevado e o *BER* mais reduzido, (ii) Por outro lado o conjunto formado unicamente pelo *forest.importance* é o que melhor se comporta em conjuntos de dados de menor dimensão. De notar ainda que o filtro *forest.importance* é do conjunto de filtros listados, aquele que apresenta maiores tempos de execução, tanto em função da dimensionalidade *d*, como do número de instâncias.

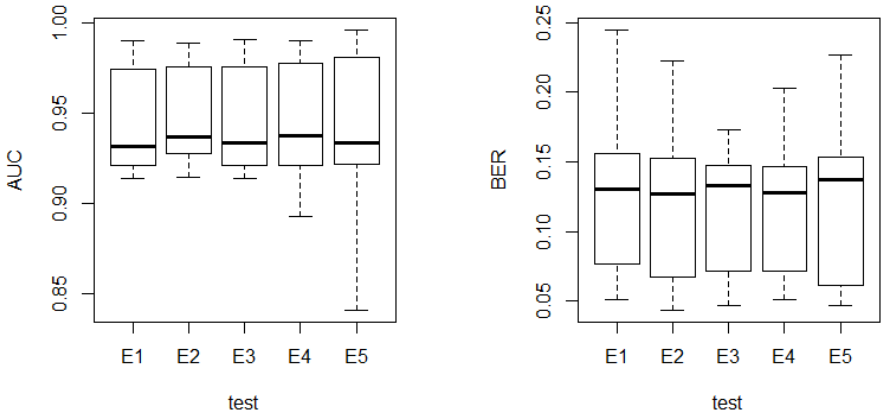


Figura 4.1: Comparativo entre *ensemble* de filtros, sobre conjuntos de dados com múltiplas dimensionalidades

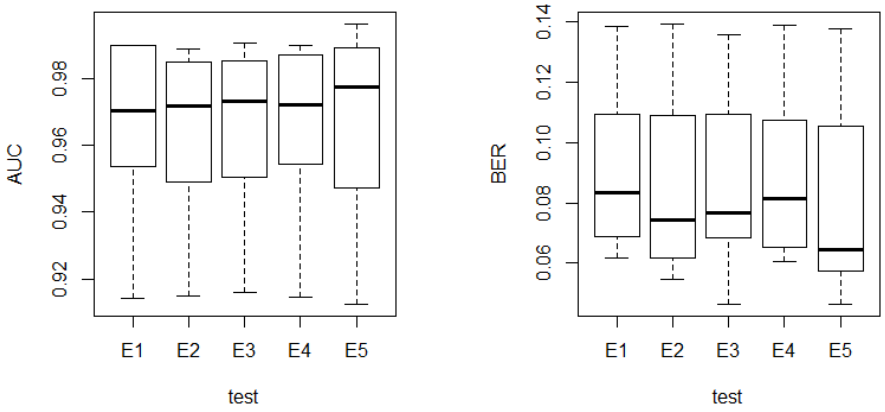


Figura 4.2: Comparativo entre *ensemble* de filtros, sobre conjuntos de dados de baixa dimensionalidade

Considerando as observações atrás e considerando que o EEFR tem como alvo, as operações de FS em conjuntos de dados de grande dimensionalidade, definimos o conjunto de filtros *E2* da Tabela 4.2, a aplicar por omissão no EEFR, caso não seja especificado qualquer outro conjunto.

Para além dos filtros: *information.gain*, *gain.ratio*, *symmetrical.uncertainty*, *chi.squared* e *forest.importance* foi analisado o *relief*. Este filtro não parece ser edequado a ser utilizado no *ensemble* do EEFR, por apresentar complexidade elevada em termos dos tempos de execução observados. Tendo como alvo o pré-processamento de conjuntos de dados de dimensionalidade elevada, o EEFR requer filtros de baixa complexidade temporal.

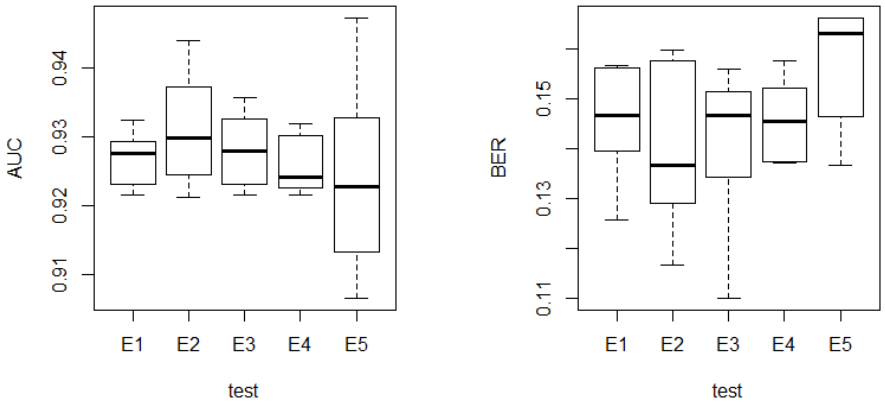


Figura 4.3: Comparativo entre *ensemble* de filtros, sobre conjuntos de dados de elevada dimensionalidade

Para fins comparativos, a (Figura 4.4) apresenta a evolução dos tempos de execução dos diferentes filtros em função do número de características (fNum). Este teste utiliza um conjunto de dados artificial com 200 instâncias e dimensionalidade variável a partir das 20 características.

Tal como referido na Subseção 2.2.3, o EFR apresenta uma ordem de complexidade $O(n_p d)$ com n_p = número de instâncias atribuídas às partições e d = dimensionalidade do conjunto de dados. Esta situação é válida tanto para o EFR como para o EEFR, desde que os filtros utilizados sejam de ordem $O(n)$. Caso o conjunto de filtros inclua o *rforest.importance*, como proposto para o EEFR, a ordem de complexidade passa para $O(n_p log(n_p) d)$ o que torna o tempo de processamento do pré-processo de FS mais lento no caso de conjuntos de dados com número de instâncias elevado. Este problema é atenuado reduzindo o comprimento das partições n_p e no caso específico do EEFR utilizando a opção processamento paralelo.

4.3 Testes EEFR sobre as funcionalidades de FR e FS

De forma a se analisar o EEFR com as alterações propostas nas Seções 3.1 (nova função W) e 3.2 (obtenção automática do subconjunto das k melhores características) , foram efetuados diversos conjuntos de testes: (i) Seção 4.3.1, testes do EEFR sobre conjuntos de dados de diferentes dimensionalidades; (ii) Seção 4.3.2, testes do EEFR sobre conjuntos de dados de dimensionalidade elevada ($d > 100$); (iii) Seção 4.3.3, evolução da eficiência de modelo, em função da redução de dimensionalidade.

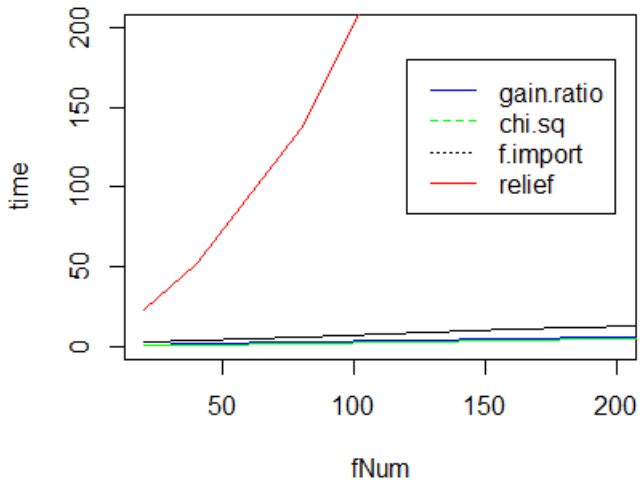


Figura 4.4: Evolução dos tempos de execução de diferentes funções de filtragem

Tabela 4.4: Ambiente de teste do EEFR

conj. de dados	conj. de treino d/n	SVM	rForest	logReg	sampling	atrib	class
SPAM	56/3066	X	X	X	Original	Numeric	Binary
DBC	30/379	X	X	X	Original	Numeric	Binary
Arcene	10000/100	X	X		Original	Numeric	Binary
CancerMass	117/68195	X	X	X	UnderS	Numeric	Binary
Dexter (sparce)	20000/300	X	X		Original	Numeric	Binary
Splice	60/1000	X	X	X	Original	Categorical	Binary

Todos os testes têm por base os mesmos conjuntos de dados e os mesmos classificadores tal como indicado na Tabela 4.4. As medidas de avaliação utilizadas nos testes são as estatísticas dos Area Under the ROC Curve (AUC) e Balanced Error Rate (BER) as quais medem a qualidade dos modelos de classificação [24] obtidos. A tabela 4.4 identifica os conjuntos de dados utilizados nos testes, as respetivas dimensões (nº de características / número de instâncias) e os algoritmos de classificação. A coluna *sampling* indica o tipo de transformação aplicado ao subconjunto dos dados obtido após o pré-processamento FS, de forma a balancear o número de instâncias por valor da classe: *original* – sem transformação; *underSampling* – redução das instâncias com valor da classe dominante; *oversampling* – aumento do número de instâncias com a classe em minoria.

Na comparação do efeito das diversas formas de FR / FS, foram avaliados os dados estatísticos do *AUC* e do *BER* obtidos pelos 18 testes efetuados por tipo de FS:

- FS tipo T1: Sem FS,
- FS tipo T2: Com FS, utilizando uma aproximação do EFR (curva *W* linear),
- FS tipo T3: Com FS, utilizando o EEFR (com a curva *W* proposta, Equação (3.1)).

O teste T2, utilizando uma aproximação do EFR, tem a ver com o facto do EFR avaliar sem selecionar diretamente um subconjunto das *k* melhores características. Face à quantidade de testes envolvidos, seria pouco prático determinar manualmente o valor de *k* por tentativas para cada um dos testes.

De modo a compararmos os efeitos introduzidos no EEFR pela nova função *W* e pelo cálculo automático de *k*, independente doutras alterações, ambos os testes T2 e T3 utilizam o mesmo conjunto de filtros (*gain.ratio, symmetrical.uncertainty, chi.squared, forest.importance*). A opção por este conjunto de filtros e não outro, é explicada na Seção 4.2. A opção pela utilização de *AUC* e *BER* como medidas de qualidade dos modelos obtidos, tem a ver com o facto destas serem comuns noutras avaliações de FS [15]. O AUC (área da curva da taxa de verdadeiros positivos em função da taxa de falsos negativos) e o BER (média das taxas de falsos positivos e taxa de falsos negativos), são mais abrangentes e menos sensíveis ao desequilíbrio do número de instâncias por valor da classe observada em alguns dos conjuntos de dados, do que outros indicadores, como por exemplo a exatidão (ACC) ou a sensibilidade.

4.3.1 *EEFR* sobre múltiplos conjuntos de dados

A Tabela 4.5 apresenta as dimensionalidades dos subconjuntos de dados de treino, obtidos após os pré-processamentos de FS. A Figura 4.5, apresenta em forma visual os dados estatísticos principais das medidas *AUC* e *BER*, resultantes dos testes efetuados sobre todos os conjuntos de dados indicados na Tabela 4.4.

Como se pode verificar pelas medidas observadas, as diferenças entre os testes T1, T2 e T3, não são significativas, no caso do AUC, $H0 : \mu(T1) = \mu(T2)$, $valor\_p = 0.47$, $H0 : \mu(T1) = \mu(T3)$, $valor\_p = 0.41$. Da Figura 4.5, verifica-se: medianas muito próximas, no entanto com o primeiro quartil a favorecer ligeiramente o teste T3. Situação semelhante no *BER*, neste caso com a mediana a favorecer ligeiramente T3.

Por outro lado, observa-se uma redução significativa (de 57% no caso de T2 e de 60% no caso de T3) do número de características selecionadas para aprendizagem, sem degradação da qualidade dos modelos. De notar que esta redução de dimensionalidade

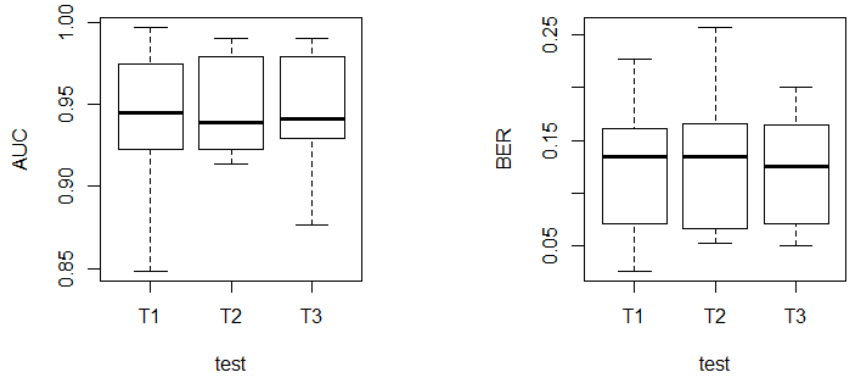


Figura 4.5: Teste EEFR múltiplos conjuntos de dados

tem por base a supressão das características nulas ou de menor relevância. Como veremos nos testes (Seção 4.3.3), a inclusão da eliminação de características redundantes irá aumentar a redução de dimensionalidade aqui apresentada.

Resumindo, a utilização do EEFR mostra-se eficiente pela obtenção da redução da dimensionalidade, mas não são observadas diferenças significativas dos indicadores de qualidade dos modelos obtidos a partir das 3 diferentes formas de seleção de características (T1, T2 e T3).

4.3.2 *EEFR* sobre conjuntos de dados de elevada dimensionalidade

Em contraste com os resultados anteriores apresentados na Figura 4.5, a Figura 4.6, apresenta em forma visual os dados estatísticos principais das medidas *AUC* e *BER*, resultantes dos testes efetuados sobre os conjuntos de dados da tabela 4.4 com dimensionalidade d superior a 100 características. O valor de 100, algo arbitrário, divide os conjuntos de dados utilizados nos testes, em dois grupos simétricos, cada um com o mesmo número de conjuntos de dados. Utilizámos como referência (Canedo 2014) [4],

Tabela 4.5: Dimensionalidade dos subconjuntos de dados, após FS

conj. de dados	**T1**	**T2**	**T3**
SPAM	56	29	26
DBC	30	15	15
Arcene	10000	4601	4307
CancerMass	117	60	55
Dexter	20000	4082	3890
Splice	60	15	14

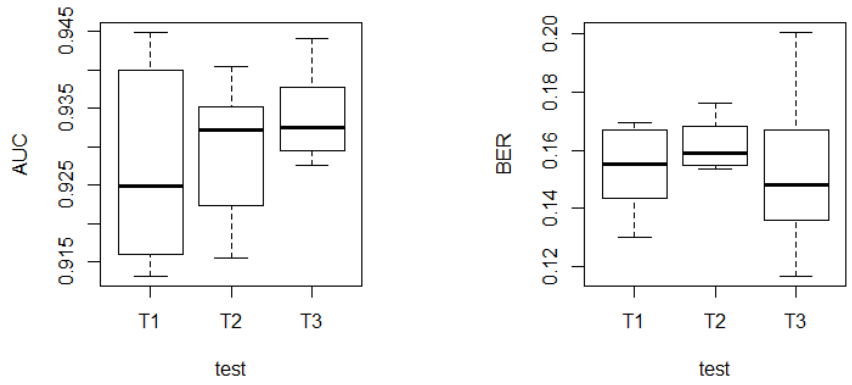


Figura 4.6: Teste EEFR para conjuntos de dados de dimensão elevada

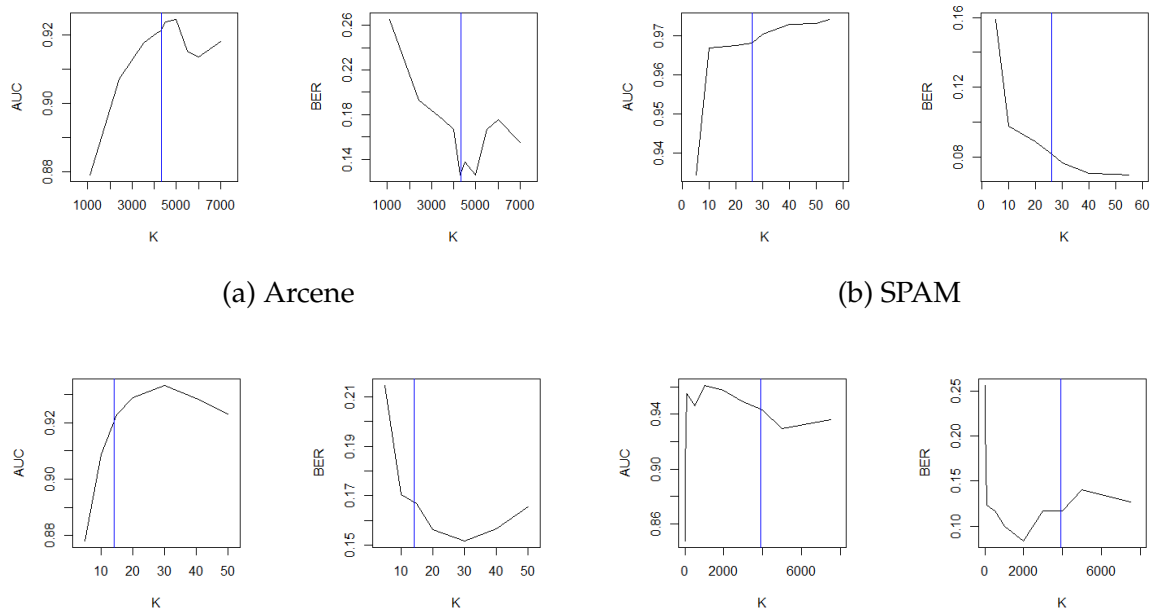
que define um número de características superior a 100 como dimensionalidade elevada e um número de características superior a 10000 como dimensionalidade muito elevada.

Pelos resultados, dos testes efetuados, conclui-se que, apesar das diferenças entre T1, T2 e T3 ainda não serem estatisticamente significativas, no caso do AUC, $H0 : \mu(T1) = \mu(T2)$, $valor\_p = 0.45$, $H0 : \mu(T1) = \mu(T3)$, $valor\_p = 0.34$, tendencialmente, a eficiência com a utilização do EEFR aumenta com a dimensionalidade dos conjuntos de dados. Da Figura 4.6, em relação ao AUC verifica-se: medianas, quartis e variância a favorecerem o teste T3 e no BER: a mediana e o primeiro quartil a favorecer ligeiramente T3. À medida que a dimensionalidade d dos conjuntos de dados é maior, o teste T3, também apresenta uma maior redução da dimensionalidade (de uma média de 60% em todos os conjuntos de dados, no teste anterior, para os 63% selecionando os de dimensionalidade elevada).

Resumindo, para conjuntos de dados de grande dimensionalidade (centenas ou milhares de características), os modelos de aprendizagem, com pré-processamento FS utilizando o EEFR com a curva da função W proposta, apresenta melhor comportamento que no caso de ausência de FS ou de utilização do EFR baseado na curva W linear.

4.3.3 *EEFR* em função de *k*, efeito da redundância

Em relação à redução de dimensionalidade observada na Tabela 4.5, comparativamente a outros estudos de FS com os mesmos conjuntos de dados, verifica-se que nalguns casos, os valores obtidos estão ainda longe do valor de redução possível [15].



(a) Arcene (b) SPAM

(c) Splice (d) Dexter

Figura 4.7: Teste EEFR em função do subconjunto das melhores k características

De forma a verificar a adequabilidade do valor *k* obtido de forma automática (Seção 3.2), fomos calcular a eficiência dos modelos obtidos a partir do subconjunto de dados obtido por EEFR variando o número das *k* primeiras características selecionadas. Fixando a lista de todas as características do conjunto de dados, ordenadas por ordem inversa de relevância, fomos testar a evolução da qualidade do modelo (*AUC* e *BER*) em função de *k*, subconjunto com as *k* primeiras características. Utilizando alguns dos conjuntos de dados anteriores e o algoritmo de classificação *SVM*, obtivemos os resultados da Figura 4.7 em que a linha vertical azul representa o valor do *k* calculado anteriormente de forma automática. Pelos resultados apresentados, a opção do cálculo automático de *k*, parece apresentar valores válidos (próximo do valor máximo do *AUC* e do valor mínimo do *BER*), com excepção da sua aplicação ao conjunto de dados *Dexter* Figura 4.7d, o qual apresenta o valor de *k* mais distante dos valores máximos de *AUC*. Esta situação deve-se ao elevado nível de redundância entre as características deste conjunto de dados. De forma a reduzir o efeito da redundância, efetuámos o mesmo teste sobre o conjunto de dados *Dexter*, mas agora com o pré-processamento FS composto por duas etapas, eliminação de redundância seguido de EEFR com calculo automático de *k*. Como se pode verificar pelos resultados obtidos, apresentados na Figura 4.8, a redução de dimensionalidade e respetivo valor de *k*, fica muito mais próximo do valor que maximiza os indicadores de qualidade do modelo obtido.

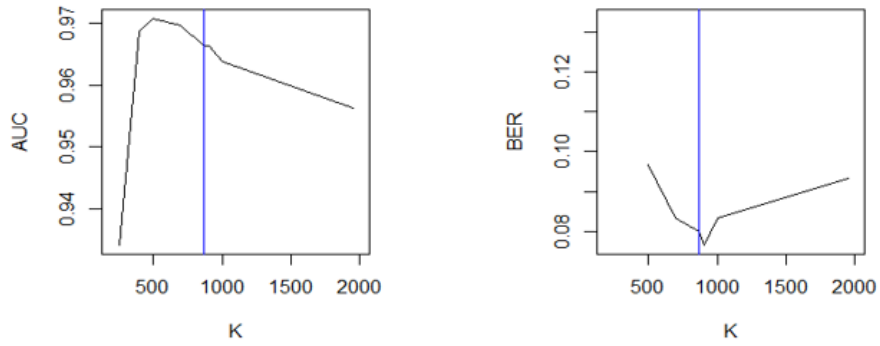


Figura 4.8: Teste RR+EEFR em função do k no conjunto de dados *dexter*

Tal como referido no Capítulo 2, os métodos por FR não eliminam características redundantes entre si e daí as reduções de dimensionalidades observadas serem, nalguns casos, bastante inferiores às produzidas por outros métodos. Este problema pode ser minimizado através de um processo em cadeia, com a eliminação de redundância das características RR seguido do EEFR. Fomos verificar o impacto do RR nos tempos de execução do RR + EEFR em todos os conjuntos de dados da tabela 4.4, utilizando como RR a função *rcorr* do package R *Hmisc*. Verificámos que no caso de conjuntos de dados sem redundância (*SPAM* e *Splice*), o incremento no tempo de execução do RR é inferior a 16%. No caso dos conjuntos de dados com redundância (*DBC*, *Arcene*, *Dexter* e *CancerMass*) os tempos de execução do RR + EEFR são muito inferiores aos do EEFR pelo facto do pré-processamento FS eliminar as características redundantes antes de executar o EEFR. Apesar da redução de dimensionalidade do RR + EEFR ser superior à do EEFR e não existir degradação no desempenho no pré-processo de FS, pretendermos manter a separação das funções de FR por relevância das de FS por relevância e redundância, pelo que a funcionalidade RR não foi incluída no EEFR.

4.4 Testes de Comparação do EEFR com outros métodos de *FS*

O objetivo deste trabalho não tem a ver com a implementação dum novo algoritmo de FS mas sim a generalização e otimização do EFR, pelo que a análise mais relevante será a comparação entre estes dois métodos. No entanto, de modo a melhor posicionar o EEFR entre as várias opções de FS, optámos por efetuar também a sua comparação com os métodos *CFS (Correlation-based Feature Selection)* [12, 13, 25] e *Relief* [22] por estes serem sobejamente conhecidos e referenciados na literatura em outros testes

comparativos.

4.4.1 *EEFR* versus *CFS*

O *CFS*, tal como referido na Seção 2.2.2, consiste na procura por *best.first.search* de um subconjunto de características, com base na deteção da relevância por correlação entre as variáveis independentes (características) e a variável dependente (classe), e na redundância por correlação entre as variáveis independentes, eliminando as características irrelevantes e redundantes.

De forma a entendermos melhor o comportamento do *CFS*, testou-se este método de FS sobre um conjunto de dados artificial, matriz com 200 $\times$ 20 valores numéricos aleatórias com distribuição uniforme distribuídos por 200 instâncias e 20 características e classe binária aleatória balanceada em 50%. Algumas das características foram alteradas de forma a estarem correlacionadas com a classe: $F_i = F_i + \alpha \times classe$, característica F_i e coeficiente de correlação α. Dos testes observados, verifica-se que o *CFS* apresenta o melhor comportamento, em tempo de execução e taxa de redução de dimensionalidade, para situações com elevada correlação entre as variáveis independentes e / ou elevada correlação entre as variáveis independentes e a dependente. Contudo, verifica-se que, se as variáveis independentes apresentarem um coeficiente de correlação baixo com a variável dependente, o *CFS* torna-se ineficaz, uma vez que essas variáveis são descartadas. Exemplo: com um valor de $\alpha = 0.01$, utilizando, o subconjunto de características obtido pelo *CFS* e o classificador *SVM*, obtemos uma exatidão de 52% no teste do modelo obtido, ou seja, nível de predição nulo, o que contrasta com o mesmo teste, com a pré-seleção efetuada com EEFR, em que se obtém níveis de exatidão de 97,5%.

Efetuando a comparação entre o EEFR e o *CFS*, em conjuntos de dados reais (Tabela 4.6), e utilizando a combinação RR + EEFR, por forma a efetuarmos testes comparáveis (RR - eliminação das características com redundância superior a 98% entre elas), obtemos os resultados apresentados na Figura 4.9 e na Tabela 4.6. A Figura 4.9 compara os dados estatísticos da qualidade dos modelos obtidos após o pré-processamento FS pelo EEFR e CFS. A Tabela 4.6 apresenta nas colunas, Dim, Dim RR+EFR e Dim CFS, a dimensionalidade original do conjunto de dados, a dimensionalidade após FS utilizando RR+EEFR e a dimensionalidade após FS utilizando CFS.

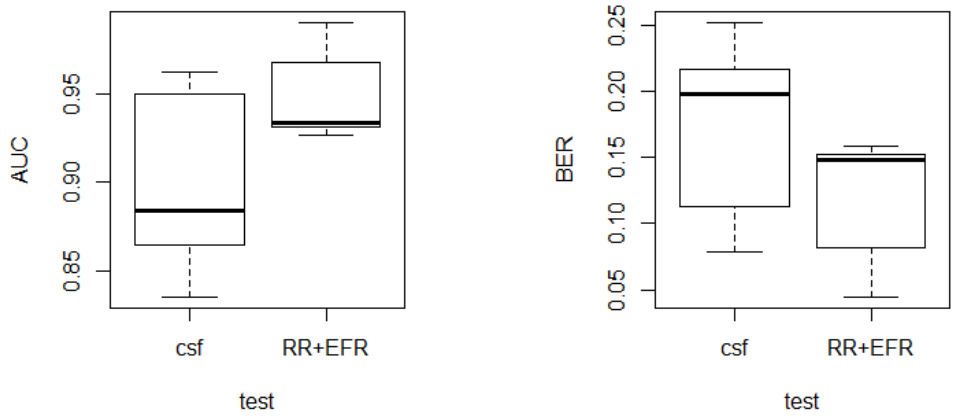


Figura 4.9: Teste CSF versus RR+EEFR

Tabela 4.6: Teste comparativo entre os algoritmos CFS e RR+EEFR para múltiplas combinações de conjuntos de dados e classificadores

conj. de dados	**Dim**	**SVM**	**rForest**	**logReg**	**sampling**	Dim **RR+EEFR**	**Dim CFS**
SPAM	56	X	X	X	Original	25	17
CancerMass	117	X	X	X	UnderS	27	9
Splice	60	X	X	X	Original	17	5

Numa primeira análise, efetuando os testes comparando o EEFR com o *CSF*, o método *CSF* apresenta maior desempenho e maior taxa de redução da dimensionalidade (Tabela 4.6). No entanto os testes evidenciam as seguintes particularidades: (i) em conjuntos de dados reais, os valores obtidos de *AUC* e *BER* para diferentes conjuntos de dados e classificadores (Figura 4.9), são melhores com a FS utilizando EEFR. As diferenças nas medidas do *AUC* apresentam significância estatística, $H0 : \mu(CFS) = \mu(RR + EEFR)$, $valor\_p = 0.011$; (ii) a taxa de redução da dimensionalidade d, é superior utilizando o *CFS*, mas à custa de alguma redução do AUC, (iii) pelo facto do *CFS* ter uma evolução de tempos de execução de $O(d^2)$ [28], torna-o pouco operacional no caso de dimensionalidades de milhares de características, tal como verificado nos testes, sendo impraticável os testes nos conjuntos de dados *Dexter e Arcene*; (iv) para situações de baixa correlação entre características e a classe, o *CFS* é ineficiente o que obriga a testes de verificação e otimização num processo que pretendemos otimizar; (v) para conjuntos de dados de baixa / média dimensionalidade, em que o ponto anterior não se aplique, o *CFS* é mais eficiente que o RR + EEFR.

4.4.2 *EEFR* versus *Relief*

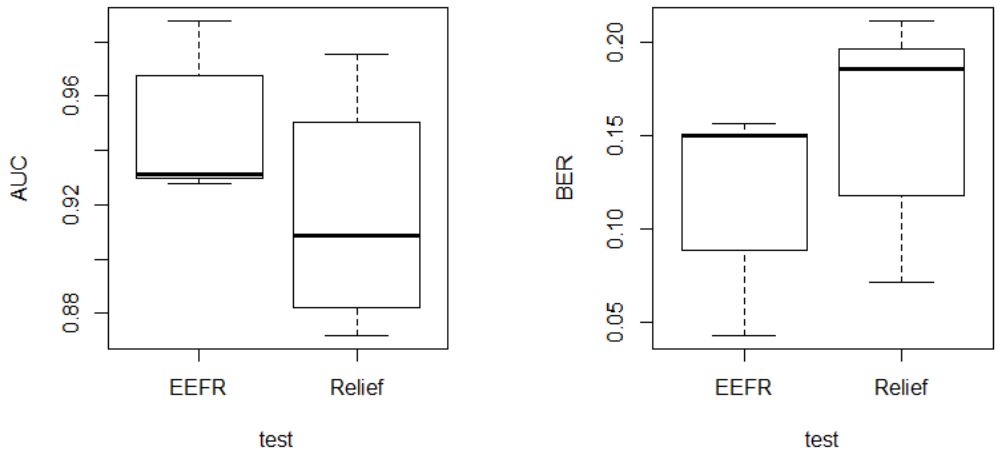


Figura 4.10: Teste Relief versus EEFR

Tal como o EEFR, o Relief (Re) é um método de avaliação de características FR, que efetua a avaliação de todas as características devolvendo uma lista das mesmas por ordem decrescente da sua relevância. Na implementação do Re utilizámos o pacote *Biocomb*. Na comparação destes dois métodos, efetuámos a avaliação das características FR em diversos conjuntos de dados (Tabela 4.7), selecionando em ambos os casos, as primeiras k características com $k = 1/3 * d$, com d = dimensionalidade do conjunto de dados. Após o pré-processamento de FS com EEFR e Re, com o objetivo de comparar qual dos métodos selecionou o melhor subconjunto do primeiro $1/3$ das características, efetuámos a modelagem e o teste com vários classificadores de cada um dos subconjuntos de características obtidos. De uma forma geral, com os conjuntos de dados e classificadores utilizados, verifica-se pela Figura 4.10 que o EEFR apresenta melhores valores de AUC e BER tanto do ponto de vista das medianas como do primeiro quartil no AUC e terceiro quartil no BER. Para maior detalhe, a Tabela 4.7 indica os valores obtidos de AUC, BER e tempo de execução do FR. Em relação aos tempos de execução verifica-se o mau comportamento do Re quando a dimensionalidade d do conjunto de dados sobe aos milhares de características, daí não se apresentarem resultados para os conjuntos de dados *Arcene* e *Dexter* por dificuldade da execução do Re em tempo útil. Em relação ao comportamento em conjuntos de dados com um número de instâncias S muito elevado (dezenas ou centenas de milhares de ocorrências), o EEFR tem a vantagem de poder jogar com os parâmetros, número de instâncias da partição n_p e número de partições (ver Subseção 4.1.1) e utilizar o pacote de processamento paralelo. No

caso do conjunto de dados *CancerMass*, alterando os valores dos parâmetros do EEFR para $n_p = n/6$ (valor por omissão $n_p = n/2$) e número de partições $S = 20$ (valor por omissão $S = 10$), utilizando o pacote *pEEFRanking*, obtemos um tempo de execução muito inferior ao do Re e com valores de *AUC* e *BER* substancialmente melhores.

Tabela 4.7: Teste comparativo entre os algoritmos Relief e EEFR para múltiplas combinações de conjuntos de dados e classificadores

Conj de Dados	Dim d/n	Alg FS	FS tempo(s)	Classificador	AUC	BER
SPAM	56/3066	EEFR	99	SVM	0.968	**8.9%**
				rForest	0.976	6.3%
				logReg	0.965	9.3%
		Relief	77	SVM	0.950	**12.0%**
				rForest	0.973	7.9%
				logReg	0.950	11.8%
CacerMass	117/68195	EEFR	**1407**	SVM	**0.931**	**15.4%**
				rForest	0.928	15.0%
				logReg	0.930	15.1%
		Relief	**3404**	SVM	**0.882**	**19.7%**
				rForest	0.872	21.2%
				logReg	0.881	20.4%
Splice	60/1000	EEFR	27	SVM	0.929	**15.6%**
				rForest	0.988	4.3%
				logReg	0.930	15.0%
		Relief	30	SVM	0.905	**18.7%**
				rForest	0.975	7.1%
				logReg	0.909	18.6%

4.4.3 *EEFR* versus *EFR*

O artigo *Ensemble feature ranking applied to medical data* (Santos et al., 2014) [37] descreve o algoritmo EFR como método de FR enquadrado na modelização dum conjunto de dados específico, *CancerMass* (*Breast cancer - KDD Cup 2008 - SIGKDD*). O EEFR pretende generalizar e otimizar a utilização do EFR a um número mais alargado de aplicações. Como anteriormente referido, as principais alterações funcionais do EEFR em relação ao EFR original, tem a ver com; (i) maior diversificação do conjunto de filtros, (ii) generalização da função W, (iii) opção de cálculo automático das k melhores características, (iv) vetorização do código e (v) opção de processamento em paralelo.

Como resultado dos testes de comparação entre o EEFR e EFR, a Tabela 4.8 apresenta os valores de AUC e BER obtidos após a classificação de diversos conjuntos de dados,

pré-processados por EFR e por EEFR. Na classificação dos dados foram utilizados os algoritmos *SVM C linear*, *Random Forest* e *Logistic Regression*. Uma vez que o EFR não dispõe da funcionalidade (iii), optámos por selecionar o subconjunto constituído pelo primeiro terço das melhores características obtidas em cada teste, deixando de fora o teste da funcionalidade (iii) o qual foi alvo de análise nas Seções anteriores. No caso de EEFR e para os conjuntos de dados de maior dimensão (*CancerMass*, *Arcene* e *Dexter*) utilizámos a opção de processamento paralelo.

De uma forma geral verificam-se valores semelhantes de AUC e BER para o mesmo conjunto de dados e classificador, justificado pelo facto de estarmos a fixar o mesmo número de características para o EEFR e EFR. No entanto, no caso de conjunto de dados de dimensionalidade elevada (*Aecene, Dexter*) verificam-se vantagens para o EEFR nos valores de AUC e BER. Em relação ao desempenho, fica evidente a vantagem na utilização da opção de processamento paralelo do EEFR para casos de conjuntos de dados de elevada dimensão (*CancerMass*, *Arcene* e *Dexter*). No caso específico do conjunto de dados *CancerMass* com amostragem *underSampling*, os valores obtidos de AUC são comparáveis aos valores apresentados nas linhas *SVM L+T1* e *RF+T1* da Tabela 2 do artigo original (Santos et al., 2014) [37].

4.5 Testes do EEFR na configuração final

O desenvolvimento do EEFR foi efetuado de forma iterativa, intercalado com testes funcionais com o objetivo de comparar e verificar diferentes opções e configurações.

Entre os diversos testes de FS realizados sobre os diferentes cenários indicados na Tabela 4.4, foi testado o EEFR com as alterações e novas funcionalidades indicadas na seção 3.1. Os resultados observados nas Figuras 4.5 e 4.6 e na Tabela 4.5 incluem na coluna T3, os resultados obtidos para o EEFR já na sua configuração por omissão, final, incluindo o conjunto de filtros que apresentaram os melhores resultados durante os testes. Estes reultados revelam a eficiência atribuída à utilização do EEFR como pré-processamento de seleção de características, na sua configuração por omissão, a qual inclui, a curva W não linear, o cálculo automático de k e o conjunto de filtros correspondentes à configuração E2 da Tabela 4.2.

Comparativamente com a ausência de FS ou utilizando o EFR, conclui-se que: (i) em média, os modelos de dados gerados a partir do subconjunto de características selecionado pelo EEFR, apresentam a melhor redução de dimensionalidade; (ii) tendencialmente, à medida que as dimensionalidades dos conjuntos de dados são mais elevadas, os indicadores de qualidade dos modelos de dados gerados a partir do subconjunto

de características selecionado pelo EEFR, apresentam os melhores valores, apesar das diferenças não serem estatisticamente significativas; (iii) o EEFR apresenta um melhor desempenho, pela particularidade do subconjunto de características ter sido calculado automaticamente a partir do FR, sem necessidade de testes manuais sobre o número de características a selecionar e pela possibilidade de processamento paralelo;

Tabela 4.8: Teste comparativo entre os algoritmos EEFR e EFR para múltiplas combinações de conjuntos de dados e classificadores

Conj de dados	Dim d/n	Alg FS	FS tempo(s)	Classificador	AUC	BER
SPAM	56/3066	EEFR	99	SVM	**0.968**	**8.9%**
				rForest	0.976	6.3%
				logReg	0.965	9.3%
		EFR	11	SVM	**0.966**	**9.7%**
				rForest	0.975	7.0%
				logReg	0.965	9.9%
CancerMass	117/68195	EEFR	**1407**	SVM	0.931	15.4%
				rForest	0.928	15.0%
				logReg	0.930	15.1%
		EFR	**3404**	SVM	0.931	14.9%
				rForest	0.933	14.0%
				logReg	0.929	14.8%
Splice	60/1000	EEFR	27	SVM	**0.929**	**15.6%**
				rForest	0.988	4.3%
				logReg	0.930	15.0%
		EFR	6	SVM	**0.918**	**17.0%**
				rForest	0.991	4.7%
				logReg	0.919	17.2%
Arcene	10000/100	EEFR	**865**	SVM	**0.920**	**15.3%**
				rForest	0.854	27.7%
				logReg		
		EFR	**1953**	SVM	**0.916**	**15.3%**
				rForest	0.824	24.1%
				logReg		
Dexter	20000/300	EEFR	**714**	SVM	**0.949**	**11.3%**
				rForest	0.958	10.7%
				logReg		
		EFR	**1529**	SVM	**0.920**	**18.0%**
				rForest	0.940	13.7%
				logReg		

5

Conclusões

As aplicações de Mineração de Dados (DM) em conjuntos de dados com centenas ou milhares de características e um número relativamente baixo de instâncias, reforçam a necessidade de Seleção de Características (FS) como pré-processamento à construção do modelo de dados. A dimensionalidade elevada dos conjuntos de dados, obrigam à escolha de algoritmos de FS de baixa complexidade em detrimento da maior eficiência dos algoritmos mais complexos. Dentro do leque alargado de métodos de FS, os métodos FS por filtragem e mais especificamente os métodos por Avaliação de Características (FR), por tipicamente utilizarem medidas estatísticas uni-variável com um nível de computação simples, adequam-se melhor a este tipo de aplicações. O EEFR é um exemplo deste tipo de algoritmos, um pouco mais elaborado por utilizar, no processo de seleção das características, múltiplas medidas estatísticas (filtros) e múltiplas partições aleatórias do conjunto de dados.

A implementação do EEFR teve como principais objetivos, (i) a reimplementação do algoritmo EFR, melhorando-o nalgumas das suas fragilidades de modo a torná-lo mais abrangente, automático e com maior desempenho; (ii) a implementação dum pacote em R, que implemente o EEFR de forma documentada, permitindo às aplicações de DM, uma forma de utilização simples deste algoritmo, no pré-processamento dos dados. Tendo como alvo, o pré-processamento de conjuntos de dados de elevada dimensionalidade, foram consideradas, formas de eliminar ou minimizar a tarefa morosa de afinação cíclica e por tentativas do pré-processo de FS. Essa simplificação é conseguida por redução do número de parâmetros de entrada, definindo um conjunto de filtros

mais diferenciado e finalmente, adicionando um novo mecanismo de seleção automática do subconjunto de características.

Após as alterações do algoritmo definidas no Capítulo 3 e com o objetivo de medir as diferenças de comportamento obtidas com EEFR no pré-processamento do conjunto de dados, foram efetuados diversos testes aos subconjuntos de características selecionados por EEFR. Cada teste individual inclui a classificação e teste do modelo obtido, utilizando as medidas de qualidade AUC e BER, utilizando partições de instâncias distintas para a aprendizagem e teste. Sendo o objetivo melhorar, em média, a eficiência e desempenho do algoritmo EEFR, independente do tipo de conjunto de dados e do classificador, cada teste de conjunto, inclui testes individuais sobre diferentes tipos de conjuntos de dados e diferentes algoritmos de classificação. Os conjuntos de dados foram selecionados sem critério especial para além de serem diversificados: com a classe binária balanceada e desbalanceada, esparsos e não esparsos e com dimensionalidades entre as dezenas e os milhares de características. Com base nas evidências dos resultados dos testes efetuados no Capítulo 4, podemos concluir que existem ganhos na utilização do EEFR no pré-processamento FS de conjuntos de dados de dimensionalidade elevada, comparativamente com a ausência de FS ou com o EFR:

1. Os testes efetuados na Seção 4.2 comparam vários conjuntos de filtros a aplicar no EEFR com seleção automática do número de características. Estes testes mostram que, à medida que a dimensionalidade dos conjuntos de dados aumenta, as diferenças entre os diferentes conjuntos de filtros testados, começam a surgir. A partir dos testes efetuados e tendo em conta o alvo do EEFR em conjuntos de dados de dimensionalidade elevada, é definido o conjunto por omissão a utilizar no EEFR.

2. Os testes efetuados nas Seções 4.3.1 e 4.3.2 comparam as médias da qualidade dos modelos de dados obtidos, sem FS, com o EFR (aproximação) e com o EEFR. Após as alterações introduzidas no EEFR, os resultados evidenciam ganhos de desempenho, pela maior redução de dimensionalidade, pela automatização do processo FS e por uma tendência positiva das medidas de qualidade AUC e BER, dos modelos criados, à medida que a dimensionalidade dos conjuntos de dados aumenta.

3. Os testes efetuados na Seção 4.3.3 têm como objetivo verificar se o número de características selecionadas automaticamente com EEFR é adequado. Os resultados mostram que, como método de FR, o número de características k selecionado pelo EEFR é eficiente. Contudo, o EEFR, tal como outros métodos FR, pelo facto

de não eliminar características redundantes entre si, não garante a menor dimensionalidade do subconjunto gerado.

4. Já fora do âmbito principal traçado para este trabalho, na Seção 4.4.1 analisamos uma forma prática e simples de melhorar a redução da dimensionalidade k, em conjuntos de dados com elevada redundância, por combinação do EEFR com um algoritmo de supressão de redundância.

5. Os resultados dos testes efetuados nas Seções 4.4.1 e 4.4.2 mostram, comparativamente, como o método EEFR pode ser mais competitivo que outras alternativas com algoritmos mais complexos, tais como o CFS ou o *Relief*.

6. Finalmente, os testes efetuados na Subseção 4.4.3 mostram que existe convergência de resultados entre o EEFR e o EFR selecionando o mesmo número de características após o pré-processamento dos dados e mostram vantagens para EEFR nos conjuntos de dados de maior dimensionalidade.

Apesar de todos os testes realizados, estes ainda estão longe de serem exaustivos, sobretudo, em relação aos tipos de filtros testados e à comparação com outros algoritmos. No entanto, face aos resultados favoráveis obtidos, ao nível de automatização do pré-processo de FS, à baixa complexidade do EEFR e à possibilidade de processamento paralelo, os indicadores são favoráveis à utilização do EEFR em conjuntos de dados de dimensionalidade elevada. Dadas as potencialidades observadas, o EEFR parece assim justificar algum trabalho adicional de desenvolvimento e teste, tal como indicado na lista de sugestões da Seção 5.1.

Em termos de implementação, tal como indicado na Seção 4.1, foi criado um pacote R com a implementação do EEFR tornando-o mais acessível, documentado e em linha com as práticas da comunidade R. Tendo o EEFR como alvo, os conjuntos de dados de dimensionalidade elevada, foi implementado um pacote R alternativo, com capacidade de execução paralela. A Seção 4.1.3 demonstra os ganhos obtidos no tempo de execução do EEFR em conjuntos de dados com milhares de características, utilizando processamento paralelo.

5.1 Melhorias futuras a considerar

De modo a melhorar a eficiência do EEFR e como sugestão para futuras evoluções, recomendamos: (i) Alteração do código das medidas de filtragem, utilizadas nas classificações parciais, de modo a melhorar a aleatoriadade do *ranking* das características

sem relevância, aumentando assim, ainda mais, a distância entre os valores médios dos *weights* mais baixos e mais elevados; (ii) Alteração da medida de filtragem *random.forest.importance* de modo a compatibilizá-la com o método de processamento paralelo utilizado *future*; (iii) Alargar os testes com outras medidas de filtragem para além das incluídas no *FSelector package* [13] como por exemplo as incluídas no *Praznik package* [31]; (iv) A análise e testes efetuados relacionados com o EEFR foram limitados a conjuntos de dados para aprendizagem supervisionada com classe binária. Seria recomendável a generalização do EEFR para conjuntos de dados associados a aprendizagem supervisionada multiclass; (v) Com o aumento da dimensionalidade dos conjuntos de dados, o tema da estabilidade do FS passa a ser importante pelo que se recomenda adicionar testes de estabilidade aos testes efetuados ao EEFR. Ambas as sugestões (i) e (ii) requerem a reimplementação dos métodos de filtragem incluídos no *FSelector package* ou no *Praznik package* caso se alargue o leque de filtros utilizáveis a este pacote, implementação adicional não incluída neste trabalho.

Referências

[1] Fernando Martínez-Plumed, Lidia Contreras-Ochando, Cèsar Ferri, José Hernández-Orallo, Meelis Kull, Nicolas Lachiche, María José Ramírez-Quintana & Peter Flach, "CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories", *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, n.º 8, páginas 3048–3061, 2021. DOI: 10.1109/TKDE.2019.2962680.

[2] Andrea Bommert, Xudong Sun, Bernd Bischl, Jörg Rahnenführer & Michel Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data", *Computational Statistics & Data Analysis*, vol. 143, pág. 106 839, 2020.

[3] Breiman Leo, "Random Forests", *Machine Learning*, vol. 45, n.º 1, páginas 5–32, 2001. DOI: 10.1023/A:1010933404324.

[4] Verónica Bolón-Canedo, "Novel feature selection methods for high dimensional data", Tese de Doutoramento, jun. de 2014.

[5] Girish Chandrashekar & Ferat Sahin, "A survey on feature selection methods", *Computers & Electrical Engineering*, vol. 40, n.º 1, páginas 16–28, 2014, 40th-year commemorative issue, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2013.11.024.

[6] "Curse-of-Dimensionality". (jan. de 2021), URL: https://www.wikiwand.com/en/Curse_of_dimensionality.

[7] Antonia Dåderman & Sara Rosander, *Evaluating Frameworks for Implementing Machine Learning in Signal Processing : A Comparative Study of CRISP-DM, SEMMA and KDD*, 2018.

[8] Manoranjan Dash & Huan Liu, "Consistency-based search in feature selection", *Artificial Intelligence*, vol. 151, n.º 1, páginas 155–176, 2003, ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(03)00079-1.

[9] C. E. Shannon, "A mathematical theory of communication", *The Bell System Technical Journal*, vol. 27, n.º 3, páginas 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[10] Usama Fayyad, Gregory Piatetsky-Shapiro & Padhraic Smyth, "From data mining to knowledge discovery in databases", *AI magazine*, vol. 17, n.º 3, páginas 37–37, 1996.

[11] Artur J. Ferreira & Mário A.T. Figueiredo, "Efficient feature selection filters for high-dimensional data", *Pattern Recognition Letters*, vol. 33, n.º 13, páginas 1794–1804, 2012.

[12] "Feature Selection". (jun. de 2021), URL: https://en.wikipedia.org/wiki/Feature_selection.

[13] Patrick Schratz Piotr Romanski Lars Kotthoff, *Selecting attributes*, online: https://cran.r-project.org/web/packages/FSelector/FSelector.pdf, 2021.

[14] Isabelle Guyon & André Elisseeff, "An introduction to variable and feature selection", *Journal of machine learning research*, vol. 3, n.º Mar, páginas 1157–1182, 2003.

[15] Isabelle Guyon, Steve Gunn, Asa Ben Hur & Gideon Dror, "Result Analysis of the NIPS 2003 Feature Selection Challenge", sér. NIPS'04, Vancouver, British Columbia, Canada: MIT Press, 2004, 545–552.

[16] Mark A Hall, "Feature selection for discrete and numeric class machine learning", *DSpace RIS Export, Computer Science, University of Waikato*, 1999. URL: https://hdl.handle.net/10289/1033.

[17] Hanchuan Peng, Fuhui Long & C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, n.º 8, páginas 1226–1238, 2005. DOI: 10.1109/TPAMI.2005.159.

[18] Bhattacharyya Dhruba K Hoque Nazrul Singh Mihir, "EFS-MI: an ensemble feature selection method for classification", *Complex and Intelligent Systems*, páginas 105–118, 2018. URL: https://doi.org/10.1007/s40747-017-0060-x.

[19] Gordon Hughes, "On the mean accuracy of statistical pattern recognizers", *IEEE transactions on information theory*, vol. 14, n.º 1, páginas 55–63, 1968.

[20] Alan Jović, Karla Brkić & Nikola Bogunović, "A review of feature selection methods with applications", em *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, Ieee, 2015, páginas 1200–1205.

[21] Utkarsh Mahadeo Khaire & R. Dhanalakshmi, "Stability of feature selection algorithm: A review", *Journal of King Saud University - Computer and Information Sciences*, 2019, ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2019.06.012.

[22] Kenji Kira & Larry A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm", em *Proceedings of the Tenth National Conference on Artificial Intelligence*, sér. AAAI'92, San Jose, California: AAAI Press, 1992, 129–134, ISBN: 0262510634.

[23] Huan Liu & Hiroshi Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[24] Yangguang Liu, Yangming Zhou, Shiting Wen & Chaogang Tang, "A Strategy on Selecting Performance Metrics for Classifier Evaluation", *International Journal of Mobile Computing and Multimedia Communications*, vol. 6, páginas 20–35, out. de 2014. DOI: 10.4018/IJMCMC.2014100102.

[25] Mark A. Hall, "Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning", 2000.

[26] Brent Daniel Mittelstadt, Patrick Allo, Mariarosaria Taddeo, Sandra Wachter & Luciano Floridi, "The ethics of algorithms: Mapping the debate", *Big Data & Society*, vol. 3, n.º 2, pág. 2 053 951 716 679 679, 2016.

[27] M Arthur Munson, "A study on the importance of and time spent on different modeling steps", *ACM SIGKDD Explorations Newsletter*, vol. 13, n.º 2, páginas 65–71, 2012.

[28] Raul-Jose Palma-Mendoza, Luis de Marcos, Daniel Rodriguez & Amparo Alonso-Betanzos, "Distributed correlation-based feature selection in spark", *Information Sciences*, vol. 496, 287–299, 2019, ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.10.052.

[29] Thu Zar Phyu & Nyein Nyein Oo, "Performance comparison of feature selection methods", em *MATEC web of conferences*, EDP Sciences, vol. 42, 2016, pág. 06 002.

[30] Adrian Pino & Carlos Morell, "Analytical and Experimental Study of Filter Feature Selection Algorithms for High-dimensional Datasets", em *Proceedings of the Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support*, Atlantis Press, 2013/10, páginas 339–349, ISBN: 978-90-78677-86-4. DOI: https://doi.org/10.2991/.2013.42.

[31] Miron B. Kursa, *Tools for Information-Based Feature Selection*, online: https://cran.r-project.org/web/packages/praznik/praznik.pdf, 2020.

[32] Ryan J. Urbanowicz, Melissa Meeker, William La Cava, Randal S. Olson & Jason H. Moore, "Relief-based feature selection: Introduction and review", *Journal of Biomedical Informatics*, vol. 85, páginas 189–203, 2018, ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.07.014.

[33] Gábor Csárdi Hadley Wickham Peter Danenberg, *In-Line Documentation for R*, online: https://roxygen2.r-lib.org/, https://github.com/r-lib/roxygen2, 2020.

[34] Meyer & HAMMERSCHMIDT, *How to write your own R package and publish it on CRAN*, https://www.mzes.uni-mannheim.de/socialsciencedatalab/article/r-package/, 2020.

[35] Henrik Bengtsson, *Unified Parallel and Distributed Processing in R for Everyone*, online: https://future.futureverse.org, https://github.com/HenrikBengtsson/future, 2021.

[36] ——, *Apply Function to Elements in Parallel using Futures*, online: https://github.com/HenrikBengtsson/future.apply, 2021.

[37] Vítor Santos, Nuno Datia & MPM Pato, "Ensemble feature ranking applied to medical data", *Procedia Technology*, vol. 17, páginas 223–230, 2014.

[38] "SEMMA". (dez. de 2020), URL: https://en.wikipedia.org/wiki/SEMMA.

[39] Colin Shearer, "The CRISP-DM model: the new blueprint for data mining", *Journal of data warehousing*, vol. 5, n.º 4, páginas 13–22, 2000.

[40] Thilo Stadelmann, Mohammadreza Amirian, Ismail Arabaci, Marek Arnold, Gilbert François Duivesteijn, Ismail Elezi, Melanie Geiger, Stefan Lörwald, Benjamin Bruno Meier, Katharina Rombach et al., "Deep learning in the wild", em *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, Springer, 2018, páginas 17–38.

[41] Anna Stief, James R. Ottewill & Jerzy Baranowski, "Relief F-Based Feature Ranking and Feature Selection for Monitoring Induction Motors", em *2018 23rd International Conference on Methods Models in Automation Robotics (MMAR)*, 2018, páginas 171–176. DOI: 10.1109/MMAR.2018.8486097.

[42] Guangming Wang & Jinxin Xu, "A Decision Tree Algorithm Based on Coordination Degree and Gini-Coefficient", em *2008 International Conference on MultiMedia and Information Technology*, 2008, páginas 822–824. DOI: 10.1109/MMIT.2008.208.

[43] Lei Yu & Huan Liu, "Efficient Feature Selection via Analysis of Relevance and Redundancy", *J. Mach. Learn. Res.*, vol. 5, 1205–1224, dez. de 2004, ISSN: 1532-4435.

Ensemble Feature Ranking

O algoritmo EFR utiliza uma combinação de F medidas de filtragem e S subconjuntos aleatórios de instâncias do *dataset* original. Para cada uma destas $F \times S$ combinações é calculado um vetor de medidas e respetivos *rankings*. Utilizando a função W, a qual atribui a cada característica um peso parcial em função do seu *ranking*, é calculado o vetor de pesos parciais de todas as características. Finalmente, a avaliação de características (FR) é efetuada a partir da soma vetorial dos $F \times S$ vetores de pesos parciais. A partir do FR pode ser efetuada a seleção do subconjunto das k características mais relevantes.

A.1 Características Principais

O EFR apresenta três atributos principais que o caracterizam: (i) o *ensemble* de filtros que o tornam mais genérico, independente do tipo de *dataset* e do modelo de classificação a utilizar. (ii) a utilização de filtros na pesagem das características, garantindo assim a baixa complexidade e a sua adequação em *datasets* de elevada dimensionalidade. (iii) a utilização de múltiplas amostragens com um número reduzido de instâncias do *dataset*, permite controlar os tempos de execução no caso de *datasets* com número de instâncias n elevado.

A.1.1 Função W

Os filtros atrás descritos têm uma particularidade em comum que é a de devolverem como resultado um vetor de medidas (*weights*), cujas componentes $weights_k$ ($k = 1...d$) correspondem às relevâncias relativas das características. Contudo, sendo o tipo de valores obtidos específico de cada filtro, estes não são diretamente comparáveis de filtro para filtro. De modo a combinar resultados de diferentes filtros e filtragens, é utilizada a função-W associada ao vetor W, cujos componentes w_i correspondem ao peso a atribuir por posição i que cada característica toma no *rank* resultante de cada medida de filtragem.

A.1.2 Implementação

O pseudocódigo original em que está baseado o EFR é apresentado na figura Algoritmo 2. Na implementação do Algoritmo EFR, a função `ensemble.feature.ranking`, devolve diretamente um vetor com o nome de todas (ou de um subconjunto por especificação do k) as características, ordenadas inversamente por relevância.

Resumidamente a implementação em R deste algoritmo, contempla a seguinte configuração:

1. Parâmetros de entrada (inputs): o *dataset* do tipo numérico composto por características e classe do tipo categórico, uma lista de métodos de filtragem, por omissão: IG, GR, SU e CS, número de execuções do algoritmo, por filtro, cada uma associada e uma amostragem de instâncias retirada aleatoriamente do *dataset*, o número de instâncias de cada amostragem e opcionalmente, o *cutoff k* (número de características a selecionar).

2. Resultado da implementação (output): é devolvido um vetor com os nomes de todas as características, ordenado por ordem decrescente da sua relevância, ou o subconjunto das k primeiras, por especificação do valor k.

A implementação do EFR utiliza o *"FSelector Package"* criado por Piotr Romanski em 2009, o qual disponibiliza um conjunto de técnicas de FS organizadas por "feature ranking" e "subset selection". Neste algoritmo são utilizados, por omissão, os seguintes filtros do grupo "feature ranking" [13]: (i) *Information Gain* (IG), (ii) *Gain Ratio* (GR), (iii) *Symmetrical Uncertainty* (SU), (iv) *Qui*-Quadrado, do inglês *Chi-Squared* (CS).

O *package FSelector* inclui, integrado na medida de filtragem, a discretização dos valores das características do *dataset* [23], viabilizando a utilização de alguns dos filtros

Algoritmo 2 Ensemble feature ranking

Input: *data* - The dataset for feature ranking, represented as a matrix;
S - The number of iterations;
sampleSize - The size of the sample for each iteration;
W - A vector containing the weights for each rank position;
algo - A vector of feature ranking functions
Output: A vector with the rank of each feature

```
1: function ENSEMBLEFEATURERANKING(data, S, sampleSize, W, algo)
2:     weights ← vector(length(algo))  ▷ A matrix of weights. Each row i represents the weights of the j features, given each algorithm in algo
3:     for each fun in algo do
4:         localWeights ← vector(S)
5:         for 1 to S do
6:             randomData ← SamplingWithReplacement(data, sampleSize)
7:             localWeights.add(fun(randomData))
8:         end for
9:         weights.add(calculateWeightedRank(localWeights, W))
10:    end for
11:    rank ← vector with the sum of all weights_j
12:    return rank
13: end function
```

```
1: function CALCULATEWEIGHTEDRANK(weights,W)
2:     weightedRank ← vector(length(W))
3:     set to 0 in all weightedRank[i]
4:     for i ← 1 to S do
5:         weightedRank+ = W indexed by weights[i]
6:     end for
7:     return weightedRank
8: end function
```

exigindo características categóricas em *datasets* com características de valores numéricos contínuos.

Algumas fragilidades do EFR: Ambos os filtros IG, GR e SU são baseados em medidas de informação mútua, com resultados muito semelhantes entre eles, como indicado no estudo comparativo [2]. o que revela possíveis otimizações ao algoritmo EFR, com um *ensemble* de filtros mais diversificado. A curva associada à função W contém parâmetros de caráter arbitrário e inscritos no código. A seleção do subconjunto das melhores características não é efetuada automaticamente. Em termos de desempenho, a implementação R utiliza demasiados *loops*, forma pouco eficiente que pode penalizar o tempo de execução do pré-processamento FS e de difícil evolução para o processamento paralelo das $F \times S$ iterações de medidas de filtragem.