A Parallelized Spark Based Version of mRMR

Reine Marie Ndéla Marone[1(✉)], Fodé Camara[2], and Samba Ndiaye[1]

[1] Department of Mathematics, Cheikh Anta Diop University, Dakar, Senegal
reine.marie.marone@ucad.edu.sn
[2] Department of Mathematics, Alioune Diop University, Bambey, Senegal
fode.camara@uadb.edu.sn

Abstract. Nowadays, we are surrounded by enormous large-scale high dimensional data called big data and it is crucial to reduce the dimensionality of data for machine learning problems. That's why feature selection plays a vital role in the process of machine learning because it aims to reduce high-dimensionality by removing irrelevant and redundant features from original data. However some characteristics of big data like data velocity, volume and data variety have brought new challenges in the field of feature selection. In fact, most of existing feature selection algorithms were designed for running on a single machine (centralized computing architecture) and do not scale well when dealing with big data. Their efficiency may significantly deteriorate to the point of becoming inapplicable. For this reason, there is an increasing need for scalable yet efficient feature selection methods. That's why we present here a distributed and effective version of the mRMR (Max-Relevance and Min-Redundancy) algorithm to face real-world problems of data mining and evaluate the empirical performance of the proposed algorithms in selecting features in several public datasets. When we compared the efficiency and the scalability of our parallelized method in comparison with the centralized one we have found out that our parallelized method have given better results.

Keywords: Feature selection · Filter method · Parallel computing
Apache Spark · mRMR · SVM

1 Problematic and Related Works

1.1 Introduction

Feature selection is a fundamental preprocessing step that aims to reduce input dimensionality in machine learning and pattern recognition [1]. Many domains use feature selection: for example in bioinformatics it is an important topic because it is critical to define informative features from complex high dimensional biological in disease study, drug development, etc.

Unfortunately, most feature selection algorithms are designed for centralized computing and do not scale well with large-scale datasets [3]. To tackle these problems, distributed computing framework like Apache Hadoop, which implements the MapReduce model, can be a solution [3].

C. M. F. Kebe et al. (Eds.): InterSol 2018, LNICST 249, pp. 187–198, 2018.
https://doi.org/10.1007/978-3-319-98878-8_18

But the Apache Hadoop is not adapted to feature selection because it because it lacks built-in support for the iterative process [4]. So, an alternative to Hadoop has been presented to overcome these problems. It is Spark, a memory-based iterative computation framework that improves the IO read/write performance issue by processing intermediate data in-memory [4].

In regard to that, in this paper, we propose a parallel version of the centralized mRMR algorithm that we have named SFS-mRMR (for Spark Feature Selection method based on mRMR), on the framework Spark to ameliorate its efficiency. The choice of mRMR is motivated by the fact that minimum-redundancy-maximum-relevance (mRMR) selector is considered one of the most relevant method for dimensionality reduction due to its high accuracy.

The results that we obtained show that our algorithm is scalable and outperforms the classical mRMR feature selection method.

The rest of the paper is structured as follows:

Section 2 reviews previous works. Section 3 deals with the formulation of the problem. Section 4 presents the centralized mRMR. Section 5 gives the metrics we used in our proposal. Section 6 consists of the presentation of our algorithm. Section 7 describes the working environment. Section 8 presents and analyzes the results of the experiments. Section 9 concludes and gives futures researchs.

1.2 Related Works

Feature selection is a fundamental preprocessing step to reduce input dimensionality.

There are 3 general categories of feature selection methods: Filter, Wrapper and embedded [5].

Filters methods use some criterion to score each feature and provide a ranking to evaluate the features which determine their relevance or discriminant powers with the outcome variable [5].

In the wrapper methods the accuracy of classifier is estimated to select the features [5]. Although computationally expensive the wrapper is the best feature selection method for accuracy [5].

In Embedded methods a given model is used to guide the feature selection process, and select the most relevant features when building the model [5].

Filter methods offer better computational complexity but do not take into account the interactions among the variables, which cannot be ignored.

mRMR is one of the most famous filter method. But mRMR is a centralized method and do not scale well with ultrahigh dimensional datasets. So it is fundamental to optimize the mRMR algorithm by using efficient parallelization [7]. That's why, proposals have been made on the parallelization of mRMR algorithm the interest of which is to decrease the training time and ameliorate the accuracy of the machine-learning tasks.

The work in [1] present a parallelization of many methods based on information theory including mRMR in Apache Spark.

The Experimental results show that this methods scale well and efficiently with ultra-high-dimensional datasets.

The work in [8] proposes to extend mRMR by using a number of approaches to better explore the feature space and build more robust predictors. To deal with the computational complexity of those approaches, authors implement and parallelize functions in C using the openMP Application Programming Interface. These methods show significant gains in terms of run-time.

Authors in [9] present a two-stage selection algorithm by combining ReliefF and mRMR. In the first stage, ReliefF is applied to find a candidate gene set; In the second stage, mRMR method is applied to directly and explicitly reduce redundancy for selecting a compact yet effective gene subset from the candidate set. The experimental results show that the mRMR-ReliefF gene selection algorithm is very effective.

In [10], authors present three implementations of an extension of mRMR named fast-mRMR in several platforms, namely, CPU for sequential execution, GPU (graphics processing units) for parallel computing, and Apache Spark for distributed computing using big data technologies.

In [11], authors combined dynamic sample space with mRMR and proposed a new feature selection method. In each iteration, the weighted mRMR values are calculated on dynamic sample space consisting of the current unlabelled samples. The feature with the largest weighted mRMR value among those that can improve the classification performance is selected in preference. Five public datasets were used to demonstrate the superiority of this method.

It is clear that the methods presented in these different works use a greedy approach by iteratively add or remove features into a set of features. Our method selects a set of relevant and non-relevant features on the dataset using only one iteration. That allows a more significant reduction of the learning time and an improvement of classification accuracy.

1.3 Formulation

Our work focuses on classification with 2 classes. Let c be the class label with 2 possible values 0 or 1. Let S refer to the input dataset with a high number n of features $\{i_1,..,i_n\}$ and m instances. V is an example defined by a vector $(v_1,..,v_n)$, where v_j is the value of the feature i in V. Let $O(S, D)$ denotes the evaluation function. A subset S' of S is evaluated by O with the data D. Let S_1 and S_2 be 2 subset of features in S. $O(S_1, D) > O(S_2, D)$ means that S_1 is more interesting than S_2.

Our proposed algorithm, called SFS-mRMR, is a distributed version of the mRMR method that we based on Spark, a parallel programming framework. Our method aims to find a subset S' of features from S that maximize the function O.

2 Improvement of MRMR

2.1 The Classical MRMR

mRMR means Minimum Redundancy and Maximum Relevance. The concept of mRMR is to select the features so that they are mutually maximally dissimilar and maximally relevant with the class label l [12]. Let i and j represent 2 features of S. The mutual information between i and j is denoted by $M(i,j)$. $M(c,i)$ stands for the mutual information between the class label c and i.

The redundancy among the features in S, is obtained by calculating the mutual information between the features in S as follow:

$$Q_I(S) = \frac{1}{|S|^2} \sum_{i,j \in S} M(i,j) \tag{1}$$

The relevance of features in S with the class label c is defined as

$$R_I(S) = \frac{1}{|S|} \sum_{i \in S} M(c,i) \tag{2}$$

By optimizing (1) and (2), we obtain S^* the set constituted of features that are the most relevant and less redundant in S. It is done as follows:

$$S^* = argmax_{S' \subseteq S}[R_I(S) - Q_I(S)] \tag{3}$$

2.2 Our Proposal

SVM is an extremely powerful machine learning technique and one of the best supervised classification techniques [13]. That's why, SVM is used in combinaison with mRMR in our algorithm SFS-mRMR as proposed by authors in [14]. The features are scored by using this combinaison to obtain more performance. Let $\beta \in [0,1]$ denotes a ratio between SVM scoring and mRMR scoring. The relevancy $R_{F,i}$ of feature i in S is obtained as follow:

$$R_{S,i} = \frac{1}{|S|} \sum_{c} M(c,i) \tag{4}$$

$Q_{F,i}$ the redundancy of i is calculated as

$$Q_{S,i} = \frac{1}{|S|^2} \sum_{i,j \in S} M(i,j) \tag{5}$$

Let ω_i be the SVM weight of i.
The final score d_i of i is obtained as follow:

$$d_i = \beta|\omega_i| + (1-\beta)\frac{R_{S,i}}{Q_{S,i}} \tag{6}$$

3 Our Method

The method that we propose (SFS-mRMR) takes as input:

- a set of data S composed of x attributes and y observations,
- the number T of attributes to select in S
- β, a ratio between SVM scoring and mRMR scoring,
- and z the desired partition number for the dataset.

Let D be the set of features in S.
Our method return S′ the subset constituted of T best features namely the features that have the highest d_i values.
SFS-mRMR follows seven steps:

- **Stage 1:**

1. Construct *classes*={$c_1, .., c_y$} the set of the class labels in S.
2. Construct *values*={{$v_i^1, .., v_i^y$}, i=1 to x }

v_i^j represents value of the i-th feature in the *j-th* observation:

3. Construct *z* subspaces of features SD_t, *t = 1..z* from D.
4. Construct z subspaces sub_t of {{$v_i^1, .., v_i^y$}, i$\in$SD}
5. Send each sub_t at a worker.

- **Stage 2:**

On each worker t:

6. Create several sets for each feature *i* in sub_t by mapping *i* with each other feature *j* in D as follows:

$$i\Rightarrow\{\ i,\ \{v_i^1, .., v_i^y\},\ \{v_j^1, .., v_j^y\}\ ,\ \{c_1, .., c_y\}\}$$

The resulting set constituted of {*i*, $\{v_i^1, .., v_i^y\}$, $\{v_j^1, .., v_j^y\}$, $\{c_1, .., c_y\}$, i=1 to x, j=1 to y } is called *rdd2*.

- **Stage 3:**

In this step, we use each element of rdd2 to calculate mutual information M_{ij} between each feature *i* and another feature *j* of D. We compute also the relevance R_i (mutual information with his class label) of *i*. We proceed as follows:

For each element $e \subset rdd2$

7. $rdd\ [(i, M_{ij}, R_i)] = mapToPair\ (e=>\{\ i,\ M_{ij},\ R_i\ \})$

$M_{ij} = MutualInformation\ (\{v_i^1, ..., v_i^y\},\ \{v_j^1, ..., v_j^y\})$

$R_i = MutualInformation\ (\{v_i^1, ..., v_i^y\},\ \{c_1, ..., c_y\})\ /x$

where $c_{k \in 1..y}$ *is the class label of the observation k.*

End For each

The constituted set of each feature *i*, the mutual information between i and a feature *j of D* and the mutual information R_i between the class label and i, will be called rdd3.

- **Stage 4 :**

To compute redundancy of each feature *i*, the mutual information between i and other features in D are aggregated. It gives us a new set and we name it *rdd4*. Each value in rdd4 consists of $\{i, sumM_{ij}, R_i\}$, where $sumM_{ij}$ is the sum of mutual information between i and other features in D and R_i the mutual information between *i* and the class label.

It is done as follow:

For each $(i, M_{ij}, R_i) \subset rdd3$

8. $rdd\ [(i, sumM_{ij}, R_i)] = reduceByKey\ (\_+\_)$

$$sumM_{ij} = \sum_{i=1}^{x} M_{ij}$$

End For each

- **Stage 5:**

In this step compute for each feature *i*, its SVM weight ω_i as follows:

For each i in *D*

9. $rdd\ [(i, \omega_i)] = map\ (i =>\{\ i,\ \omega_i\})$

ω_i *in* ω *where* ω*=SVMWeight(F)*

End For each

- **Stage6:**

In this step calculate for each feature *i* its ranking measure d_i *and* sent all d_i scores to the master.

It corresponds to the following instructions:

For each element (i, $sumM_{ij}$ *,* R_i*)* ⊏ *rdd4*

10.rdd [(i, d_i*)]= mapToPair* ({*i,* $sumM_{ij}$ *,* R_i } **=>**{*i,* d_i })

Q_i= $sumM_{ij}$ */(n*n);*

d_i =β + ω_i+((1-β)* (R_i /Q_i));

/* ω_i *is SVM weight of feature i*/*

End For each

11.All workers send d_i values *to the master*

- **Stage 7:**

In this stage the features are collected and ordered by the master. Master then returns the *T* attributes with the best scores d_i.

This corresponds to the following sentences:

On the master:

12.Collect and ordered

13.Return S′: the set of T attributes that have the best d_i *values.*

4 Experimental Setup and Results

4.1 Data Description

The classifier used is SVM (support vector machine) and the data used for the experiments are in LibSVM format.

Datasets used here, are from mldata.org [15].

Table 1 gives us details of the datasets.

Table 1. Characteristics of benchmark datasets

Name	Number of features	Number of instances
Colon-cancer	2000	62
Colon-tumor	2000	60

For our experiments we have used a cluster of 4 nodes then a cluster of 6nodes. Each node has 8 cores and run at 2.60 GHz, with 56 GB memory and a 382 GB disk, Each node run at the Linux-based HDInsight (Spark) cluster.

4.2 Performance Evaluation

In this part, we will first discuss the scalability of our solution then we will compare the execution time of our proposal with the one of centralized mRMR.

Figures 1, 2 and 3 show respectively how the execution time varies according to the number of nodes when we select 25%, 50% or 75% of the dataset.

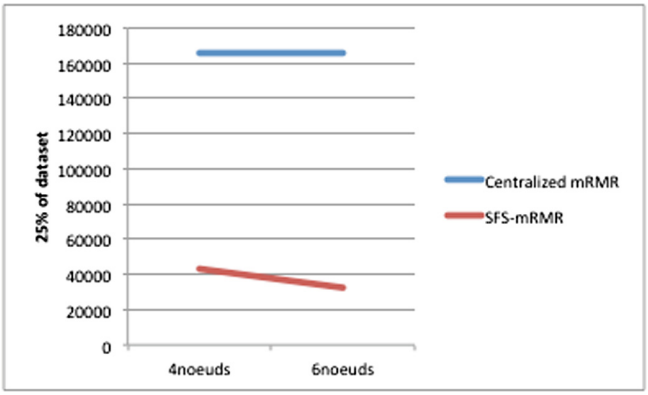

Fig. 1. Scalability of SFS_mRMR and classical mRMR with 25%.

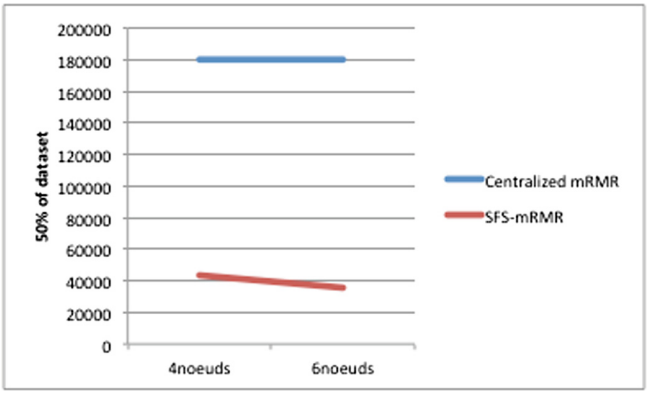

Fig. 2. Scalability of SFS_mRMR and classical mRMR with 50%.

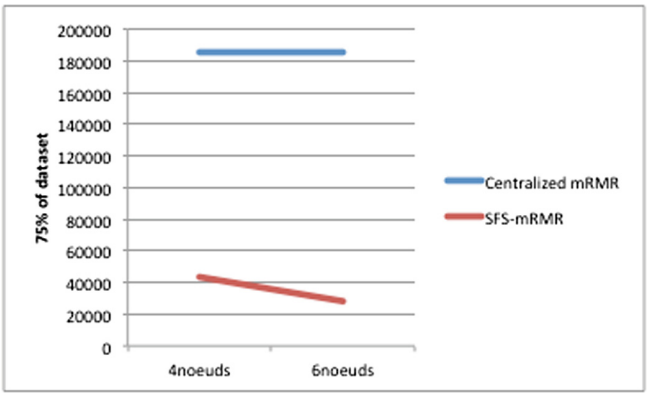

Fig. 3. Scalability of SFS_mRMR and classical mRMR with 75%.

The concerned figures clearly show that the execution time of our proposal considerably decreases when the number of nodes increases whereas the time taken by classical mRMR remains constant.

We have used 4 then 6nodes for the scalability. And for every case we have run the tests using the same environment.

For every dataset we first select 25% then 50% and after 75% of features.

- **Colon-cancer**

Figures 4 and 5 shows the time taken by our method comparatively to the one of centralized mRMR for respectively 4 and 6nodes.

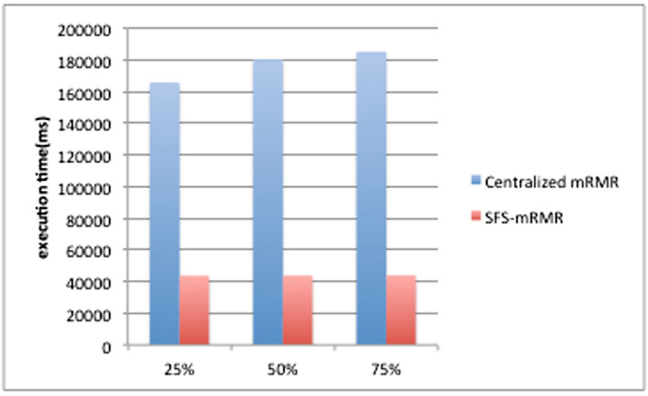

Fig. 4. Time taken for colon-cancer with 4nodes

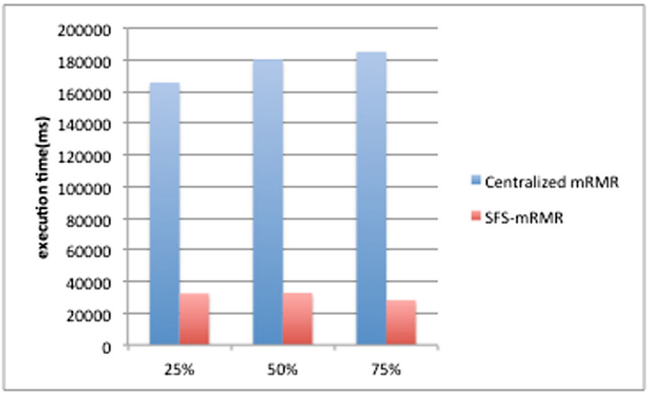

Fig. 5. Time taken for colon-cancer with 6nodes

As we can notice, the execution time of our method is at least 4 times shorter compared to the one of centralized mRMR.

- **Colon-Tumor**

For colon-tumor the results obtained with 4nodes are the following ones given in Fig. 6:

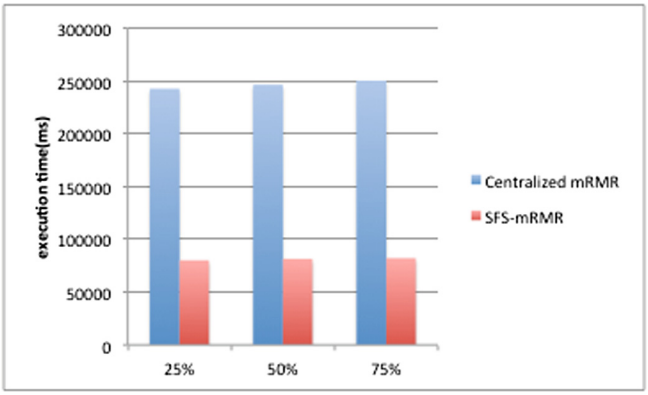

Fig. 6. Time taken for colon-tumor with 4nodes

With a cluster of 6nodes the execution time is stated in Fig. 7.

As for the colon-cancer we can notice that the execution time of our method SFS-mRMR is also 4 times shorter at least.

Therefore, we can conclude from these experiments that our solution outperforms the centralized mRMR method in terms of execution time. Moreover, the more we increase the number of nodes, the shorter the execution time becomes in our method whereas the one of centralized mRMR remains constant.

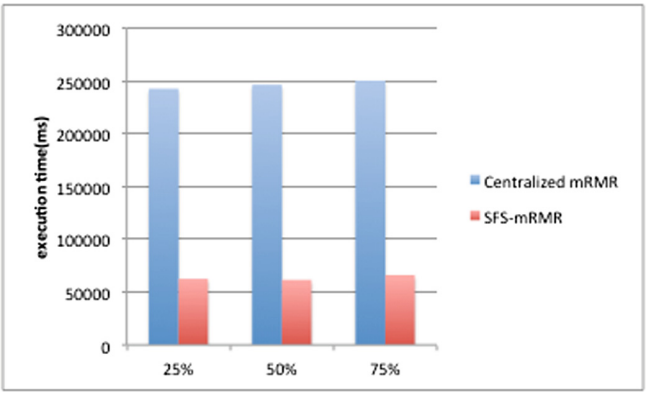

Fig. 7. Time taken for colon-tumor with 6nodes

In our experiments we have used datasets limited to 2000 features, because beyond that number, the centralized mRMR takes too much time to run. For example, for certain datasets above 2000 mRMR can take days to run completely.

5 Conclusion

In this paper, we have proposed a parallel and scalable version of a centralized feature selection method named mRMR that we developed with the Spark framework.

In our method, a score is given to each feature to evaluate its redundancy with the others features of the dataset and its relevance relatively to the class label. Then the features presenting the highest score are returned.

Performance evaluation of our method demonstrates that our parallel algorithm can improve the accuracy of classification and reduce the time taken by instance selection. The performance results also show that our method scale well and efficiently with big data.

In the future, we plan to compare our method with other parallelized methods in the litterature.

Acknowledgment. In this work, Microsoft Azure has sponsored us and we would like to take this occasion to express them our thanks.

Without their help we would not have been able to test our algorithms in a cluster and we would not have reached our goals.

References

1. Ramırez-Gallego, S., et al.: An information theory-based feature selection framework for big data under apache spark. J. Latex Class Files **13**(9) (2014)
2. Chahar, V., Chhikara, R., Gigras, Y., Singh, L.: Significance of hybrid feature selection technique for intrusion detection systems. Indian J. Sci. Technol. **9**(48) (2016). https://doi.org/10.17485/ijst/2016/v9i48/105827

3. Zhao, Z., Cox, J., Duling, D., Sarle, W.: Massively parallel feature selection: an approach based on variance preservation. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012. LNCS (LNAI), vol. 7523, pp. 237–252. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33460-3_21
4. Singh, D., Reddy, C.K.: A survey on platforms for big data analytics. J. Big Data **2**(1), 8 (2015)
5. Liu, C., Wang, W., Zhao, Q., Konan, M.: A new feature selection method based on a validity index of feature subset. Pattern Recogn. Lett. **92**(1), 1–8 (2017)
6. Wenyan, Z., Xuewen, L., Jingjing, W.: Feature selection for cancer classification using microarray gene expression data. Biostat. Biom. Open Access J. **1**(2), 555557 (2017)
7. Jaseena, K.U., David, J.M.: Issues, challenges, and solutions: big data mining. In: Sixth International Conference on Networks & Communications. https://doi.org/10.5121/csit.2014.41311
8. De Jay, N., Papillon, S., Olsen, C., El-Hachem, N., Bontempi, G., Haibe-Kains, B.: mRMRe: an R package for parallelized mRMR ensemble feature selection. Bioinformatics **29**(18), 2365–2368 (2013). https://doi.org/10.1093/bioinformatics/btt383
9. Zhang, Y., Ding, C., Li, T.: Gene selection algorithm by combining reliefF and mRMR. BMC Genom. **9**(Suppl 2), S27 (2008). https://doi.org/10.1186/1471-2164-9-S2-S27
10. Ramírez-Gallego, S., et al.: Fast-mRMR: fast minimum redundancy maximum relevance algorithm for high-dimensional big data: fast-mRMR algorithm for big data. Int. J. Intell. Syst. (2016). https://doi.org/10.1002/int.21833
11. Yang, Y., Li, H., Lin, X., Ming, D.: Recursive feature selection based on minimum redundancy maximum relevancy. In: 2010 Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP) (2010). https://doi.org/10.1109/paap.2010.52
12. Mandal, M., Mukhopadhyay, A.: An improved minimum redundancy maximum relevance approach for feature selection in gene expression data. IEEE/ACM Trans. Comput. Biol. Bioinform. (2016)
13. Chang, Y.-W., Lin, C.-J.: Feature ranking using linear SVM. In: Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008, PMLR, vol. 3, pp. 53–64 (2008)
14. Mundra, P.A., Rajapakse, J.C.: SVM-RFE with MRMR filter for gene selection. IEEE Trans. Nanobiosci. **9**(1), 31–37 (2010)
15. http://mldata.org/repository/data/viewslug/ovarian-cancer-nci-pbsii-data/