Predicting Computational Reproducibility of Data Analysis Pipelines in Large Population Studies Using Collaborative Filtering

Soudabeh Barghi

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

November 2018

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Soudabeh Barghi**

Entitled: **Predicting Computational Reproducibility of Data Analysis Pipelines in Large Population Studies Using Collaborative Filtering**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Jinqiu Yang

_____ External Examiner
Dr. Adam Krzyzak

_____ Examiner
Dr. Gregory Butler

_____ Supervisor
Dr. Tristan Glatard

Approved by _____
Lata Narayanan , Chair
Department of Computer Science and Software Engineering

_____ 2018

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Predicting Computational Reproducibility of Data Analysis Pipelines in Large Population Studies Using Collaborative Filtering

Soudabeh Barghi

Evaluating the computational reproducibility of data analysis pipelines has become a critical issue. It is, however, a cumbersome process for analyses that involve data from large populations of subjects, due to their computational and storage requirements. We present a method to predict the computational reproducibility of data analysis pipelines in large population studies. We formulate the problem as a collaborative filtering process, with constraints on the construction of the training set. We propose 6 different strategies to build the training set, which we evaluate on 2 datasets, a synthetic one modeling a population with a growing number of subject types, and a real one obtained with neuroinformatics pipelines. Results show that one sampling method, "Random File Numbers (Uniform)" is able to predict computational reproducibility with a good accuracy. We also analyse the relevance of including file and subject biases in the collaborative filtering model. We conclude that the proposed method is able to speed-up reproducibility evaluations substantially, with a reduced accuracy loss.

Acknowledgments

This work would have not been possible with the generous and continuous support of my supervisor, Dr. Tristan Glatard. I do not forget the very moment that my educational path had a significant turning point, when I asked Dr. Glatard to accept me as his graduate student. I am really grateful to him because he trusted my abilities, challenged me and let me grow both personally and professionally under his supervision. I am indebted to him for the privilege of working with him on this thesis. His dedication, devotion, leadership and professionalism is a source of inspiration for me and I would like to wish him and his esteemed family all the best in life. I am grateful to all of whom I have had the chance to work with and particularly I would like to thank my colleagues at /bin lab, especially Valérie Hayot-Sasson, Lalet Scaria, Ali Salari, and Behzad Dehghani who have helped me along the way and appreciate all the fruitful discussion opportunities that we had together. Most importantly, I would like to thank my parents and my brother who have always supported me and have inspired me to move forward at most difficult times. They are far and yet very near.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the main concerns about reliability of the findings of a study is the ability to reproduce the same results by doing the same analysis on the same data. This is defined as reproducibility. This ability could be an evidence for the correctness of the experiments. This enables the other researchers to make use of the methods and results. In other words, reproducibility is one of the ways to measure the precision of a study by measuring the ability of replicating the findings. This concept is defined and categorized in several ways [1, 19, 42]. In general, it refers to those studies that the same findings, or results within the range of experimental deviation [42], can be achieved by a different team and experimental setup. This measurement is different from repeatability in which the experiment is being taken under the same condition by a single person, instrument or team. This concept can be considered from different perspectives such as empirical [36], computational [54] and statistical [39].

This chapter explains the concept of reproducibility in relation to computational aspects of research (computational reproducibility).

1.1 Computational Reproducibility

Computational reproducibility is the ability to recompute analyses over time and space [41]. This aspect of reproducibility has become a critical component of scientific methodology as many researchers acknowledge the existence of a reproducibility crisis [7]. There are so many factors that can threaten the computational reproducibility, specially infrastructural characteristics. For instance in neuroinformatics, our primary field of interest, studies have shown the effect of the operating system on computational results [25, 26].

Neuroimaging pipelines are prone to generate non-identical results depending on the computational platform that they are computed in. According to [25] these different results achievements, arise from variations in hardware architecture and software versions. As it mentioned in [25] restricting researches to a single computing platform is not a very practical solution since: i) the computing platform is getting outdated over time and results may not reproducible; ii) there are various available platforms adaptable to various tasks. However, High-Performance Computing (HPC) is limited to alike sets of platforms; furthermore, iii) homogenizing computing platforms sometimes is not feasible like when different institutions are processing shared databases. This high possibility of not achieving same scientific results when conducting the same study with same dataset stems from the hardware and software computing platform. Several studies show that the choice of operating systems [25] and software package [10] can impact the analysis results.

In this regard, [25] presents some of the differences that are generated along the analysis of pipelines on different computing platforms. A pipeline is considered as a set of data processing elements that are connected in series. Some of the computational reproducibility issues could be caused by differences in i) hardware architecture, ii) version of code, iii) type of the library used by the code, or iv) compilation options.

A notable fact about processing of datasets in neuroimaging is the use of complicated pipelines that makes this process both time-consuming and computationally intensive. The size and type of dataset might cause the analysis process to long for several days and meanwhile the image size can exponentially grow. Therefore, conducting such reproducibility studies at scale is cumbersome due

to the computational and storage requirements of analysis pipelines.

1.2 Reproducibility of Neuroimaging Pipelines

Neuroinformatics pipelines are generally iterated on data coming from 10 to 1,000 subjects, possibly with subtle input parameter variations to adjust specific data acquisition conditions. Subject data often capture anatomical and functional characteristics of their brain, for instance through Magnetic Resonance Imaging (MRI) or Electroencephalography (EEG). The number of input files associated with a subject may vary, and these files may also be of different sizes. Typical processing times range from 15 minutes to 15 hours per subject, with input ranging from 100 MB to 15 GB per subject, and outputs ranging from 1 GB to 500 GB.

The reproducibility of a given pipeline may vary across subjects, for instance due to different pipeline branches being executed depending on the input data content. As an example, in the public database released by the Human Connectome Project [57], subjects may have one or two anatomical images of each modality; when two images are present, they are aligned together and averaged. Acquisition artifacts, for instance due to motion, may also trigger corrections not otherwise required, and some branches of the pipeline may be executed only for specific acquisition parameters. These remarks are consistent with recent findings showing that variations in the results of a functional MRI analysis are dependent on the dataset being analyzed [10]. To capture such variations, reproducibility evaluations need to be conducted on many subjects, which is unwieldy.

1.3 Contribution and Outline

In this study our goal is to predict the outcome of reproducibility evaluations in a large population of subjects, from a reduced set of pipeline executions. More precisely, we aim at predicting whether a particular file produced by an analysis pipeline will be identical across execution conditions, or if it will contain reproducibility errors. We approach the problem from the point of view of collaborative filtering. This study makes the following contributions:

(1) We model reproducibility evaluations in data processing pipelines as a collaborative filtering problem.

(2) We propose strategies to sample the training set under time constraints.

(3) We evaluate and compare our sampling strategies on synthetic and real datasets.

Chapter 2 discusses some well-know prediction techniques. The problem formulation, collaborative filtering technique, and proposed sampling strategies for the training set are presented in Chapter 3. The datasets are described in Chapter 4, and experimental results are in Chapter 5. Finally in Chapter 6, we conclude on the best sampling method to use, and on the impact, limitations and generalizability of the results. The work of Chapters 3, 4 and 5 has been accepted for publication at IEEE BigData 2018 conference (also available as an arXiv pre-print [8]).

Chapter 2

Machine Learning for Reproducibility Predictions

Equipping computers to acquire new declarative knowledge by discovering facts and theories is a challenging goal in artificial intelligence (AI). This learning process tries to map between inputs and outputs of a problem through observation and experimentation. The study and computer modelling of learning processes establish the subject matter of Machine Learning [37].

Machine Learning algorithms aim at finding a good function $F : X \rightarrow Y$ to do the best mapping between X, the set of possible inputs, and Y, the set of outputs [33]. Depending on what kind of data is provided to the learning system in order to learn the map function, Machine Learning algorithms can be classified into three categories: supervised, unsupervised and semi-supervised. In supervised methods, the learning system is provided by both inputs x and outputs y (data example of (x, y) pairs). These labelled instances can be used to learn the best map of inputs to outputs. Instead, if learning system has to discover patterns in inputs while no desired output is given, it is called unsupervised learning system. In unsupervised learning, sometimes there is a way to get the understanding about the quality of an output y by following the input x but the difference is that this time x comes from the feedback of previous experiences in the learning system. This type of unsupervised methods which works on rewards or punishments associated with actions are known

as reinforcement learning [33, 63].

The third category of Machine Learning algorithms are semi-supervised learning where the system is provided by small labelled instances of inputs-outputs pairs, like the ones in supervised learning, as well as a large number of unlabelled data similar to unsupervised learning [63].

Machine Learning algorithms are being used in recommender systems to recommend items to users. Recommender systems extract historical user ratings on items and/or implicit information in order to recommend items to users [58]. Collaborative filtering is a recommender system technique that relies on the relationship between items and users instead of analysing the content of the items or users. These relationships are encoded in a rating feedback matrix, called utility matrix, where each element represents a specific user rating on a specific item [58].

In this study we consider collaborative filtering technique as a solution for classification and regression problems. This choice is motivated by the following points:

- The problem in our study can be presented as a utility matrix of subjects and files that represents the items and users respectively.
- We were wondering if collaborative filtering approach could solve classification problems in neuroinformatic applications. In other words, we were curious to see if collaborative filtering can be used in other domains than its prior usage field, e-commerce and social networking applications.

2.1 Supervised Prediction and Classification Techniques

In this section we provide an overview on some of the well-known supervised techniques that can be used for classification problems such as k-Nearest Neighbours, Support Vector Machines, Decision Trees, and Random Forests. However, exploring these techniques in details is beyond the scope of this work.

Minkowsky: $D(x,y) = (\sum_{i=1}^{m} \lvert x_i - y_i \rvert^r)^{\frac{1}{r}}$
Manhattan: $D(x,y) = \sum_{i=1}^{m} \lvert x_i - y_i \rvert$
Chebychev: $D(x,y) = max_{i=1}^{m} \lvert x_i - y_i \rvert$
Euclidean: $D(x,y) = \left(\sum_{i=1}^{m} \lvert x_i - y_i \rvert^2\right)^{\frac{1}{2}}$
Camberra: $D(x,y) = \sum_{i=1}^{m} \frac{\lvert x_i - y_i \rvert}{\lvert x_i + y_i \rvert}$
Kerdall's Rank Correlation: $D(x,y) = 1 - \frac{2}{m(m-1)} \sum_{i=j}^{m} \sum_{j=1}^{i-1} sign(x_i - x_j) sign(y_i - y_j)$

Table 2.1: Approaches to define the distance between instances (x and y).

2.1.1 k-Nearest Neighbours

Nearest Neighbours algorithm is one of the straightforward instance-based learning algorithms that generates classification predictions by using only specific instances [6, 29]. These type of algorithms do not create an explicit model for classification, instead they classify a new instance by comparing it with other instances in the training set.

k-Nearest-Neighbours algorithm (k-NN) is one of the Nearest Neighbours methods which is based on the idea that instances in the dataset are approximately close to the other instances having similar properties [29]. To classify new instances, first it needs to identify k nearest, "similar", instances from the dataset. The class determination is achieved by identifying the most dominant class label among the k neighbours [29, 50]. There are some distance metrics that are used to determine the distance between instances; the most significant ones are presented in table 2.1 [29].

Defining k is important as the wrong choice might diminish the classification accuracy. The optimal choice of k depends on the nature of the data. The more complex and irregular structure of the data results in the lower value of k [50] being needed. Higher value of k can make the decision boundary smoother, which reduces the risk of over fitting. On the other hand, if k is chosen to be high then the ability to capture the local structure in the data would be disturbed.

According to [29], incorrect classification could occur especially when the noisy instances win the majority vote due to their locality being near to the query instance, which in this case a larger k would solve the problem. Or in another case, when the data points of different classes are very close to each other, there is a high probability to find neighbours from other classes for a query point. In

this case, having a small k may increase the chance of finding the neighbours from the same class.

To classify instance i, k-NN needs to compute the distance between i and all the instances in the training set. This high computational distance calculations, not only make k-NN sensitive to its computed distance approach, but also makes it undesirable for real-time prediction of big data problems [17, 50].

To handle its real-time issue, [17] proposed a new k-NN method that uses k-means [50] algorithms to cluster the dataset into several parts. Then k-NN classification can be conducted on the nearest cluster that now is selected as the training samples.

k-NN method requires less computation time during training phase in comparison to its classification phase process [29]. When the training set is large enough k-NN could perform very well and even much better if each class is characterized by multiple combination of predictor values [50].

There is no surprise that this method requires large storage volume since for predicting every record, its distance from the entire set of training should be calculated [50]. This method is not robust in terms of missing values as it requires complete records for doing its work [29]. Also, irrelevant features can corrupt the similarity measures used in this method.

2.1.2 Support Vector Machine

Support Vector Machine (SVM) is a classification algorithm that finds the hyperplane (decision boundary) with the maximal margin to the data classes. Margin is defined as the distance of the closest instance of the class from the hyperplane. In SVM the goal is to maximize the margin, which means the instances in either side of the hyperplane are in their furthest possible distance from it [16, 29]. This hyperplane with maximal margin is called optimal hyperplane. In the case of linearly separable data, those data points located on the margins are known as support vector points and the solution is the linear combination of only these points [29]. Those other data points that are not located on the margins are dismissed. Figure 2.1, extracted from [29], presents an example of a linear SVM model for classification problem in a two dimensional space.

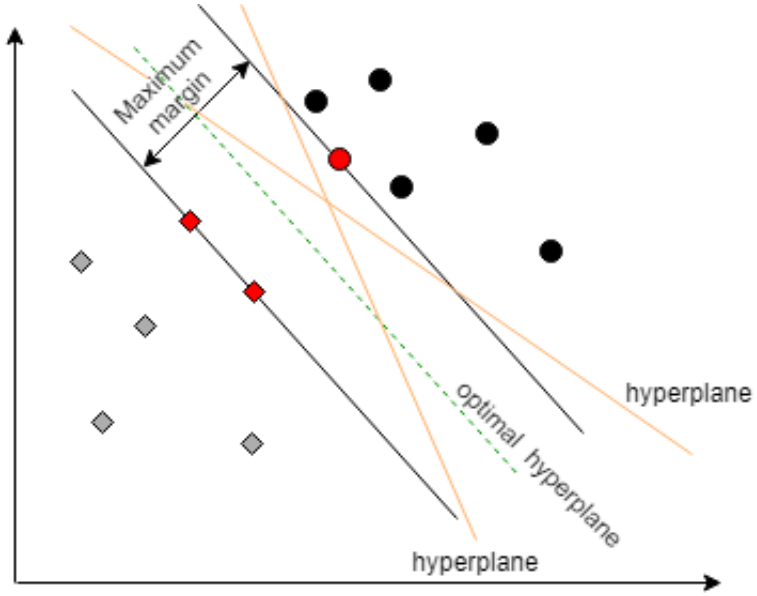


Figure 2.1: An example of a linear SVM model. The support vectors in red colour define the maximum margin between the two separable classes.

According to [56], when the data domain can be divided linearly, the SVM steps are first, mapping the data domain into a response set and then, dividing the data domain. But sometimes this linear division is not possible. In such case, the data domain is transformed into a feature space where the data domain can be linearly divided into separate classes. Therefore SVM steps are considered as mapping data domain into the feature space, using kernel function [49]. This function is used for mapping feature space domain into the response set [56], and then similar to what linear SVM, it divides the data domain as well. It is notable that this kernelized space hyperplane is based on the features that ensure the high generalization ability of the network [16].

Sometimes it is possible that SVM cannot find a hyperplane since the data set contains misclassified instances. This problem can be resolved by using soft margin solution which is well described in [29].

This algorithm is versatile in a way that not only common kernel functions are provided for the decision function, but also custom kernels are possible to be specified. Moreover, it is memory efficient [5] since it just uses the linear combination of support vector points and ignores the other data points. On the other hand choosing the right kernel functions and regularization term would be

critical when the number of features are much greater than the number of instances.

[5, 29] are some of the valuable sources of information about SVMs especially in terms of its performance. However, the general and critical comparison of supervised Machine Learning methods presented in [29] is a worth reading study in which the advantages and disadvantages of this technique are well-explained.

2.1.3 Decision Trees

Decision Tree [13] is a non-parametric algorithm which uses a tree-logic to make predictions. It uses divide-and-conquer approach, a recursive partitioning of the data set, to define the classification rules. This approach can use various metrics [59] to grow the trees such as Gini index and Information Gain. These methods evaluate variables in order to find the best attribute to split on at each steps.

Gini Index

Decision Trees could be generated by Gini index which measures the probability of incorrectly labelled random selected element from data set if this element was randomly labelled according to the distribution of labels in the subset. If p_i consider as the fraction of elements with label i in a set of items with J classes, $i \in \{1, 2, ..., J\}$, then Gini index is computed as below.

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = 1 - \sum_{i=1}^{J} {p_i}^2$$

Classification and Regression Trees (CART) [35, 53] is one of the well-known algorithms that uses Gini index technique to choose the best attribute in each splitting stage.

Information Gain

This technique is based on the entropy concept and the fact that how much information is gained by a random attribute when it is chosen. We can compute entropy as bellow, when there is a set of j classes where the selected element labeled in class i with the probability of p_i.

$$H(T) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

In process of building a tree, the calculation of Information gain for each possible first split (parent node) is needed. This Information for a selected element a of class T is gained as the subtraction of the parent's entropy $H(T)$ and the sum of children's entropy$H(T|a)$:

$$IG(T, a) = H(T) - H(T|a)$$

or

$$IG(T, a) = -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{J} - \Pr(i|a) \log_2 \Pr(i|a)$$

Some algorithms such as ID3 [43], C4.5 [44, 46] and C5.0 [40] use Information Gain to measure the quality of the split in order to keep the tree simple.

Regardless of which technique is used to choose the best attribute in each split stage, generally decision tree approach is to select an attribute and starts splitting the dataset into two nonoverlapping, smaller datasets based on the outcome of a test on the value of the selected attribute. The reason for this splitting is for increasing the chance of homogeneity of the resulting smaller datasets with respect to the target variable. This process recursively applies to the partitions, for instance until pure subsets are reached (all members are classified to only one class). To generate the classification rules it just needs to go through the paths of the tree from the root to each leaf [30, 55]. In other words a decision tree is a tree structure flowchart where the topmost node is the root node and each leaf one (terminal node) has a class label. The internal nodes indicate a test on an attribute, and finally branches show the outcome of the test [15].

Decision tree classifier works well on noisy data and does not require any prior knowledge of the data distribution [38]. They are easy to build, use, and interpret. The prediction values can be generated easily by backtracking from the leaves to its root. This algorithm is commonly used for pattern classification.

This tree-logic approach is based on a single basic idea or principle, which considers the utility of individual attributes one at a time and this could disregard the case when multiple attributes begin to be strongly predictive whereas their weak predictive utility in separate cases [38]. In addition, they tend to cause overfitting problem as they work well with the data they created by, but they are not flexible when it comes to classifying new samples.

2.1.4 Random Forest

Random forest [4, 12] is another successful Machine Learning model for classification and regression. It is based on aggregation of many decision trees that aims to control the overfitting issue.

A random forest consists of a number of, k, tree-structured classifiers $\{h(\boldsymbol{x}, \Theta_k), k = 1, ...\}$ where a random vector Θ_k is generated independently but with the same distribution from the past random vectors, $\Theta_1, ..., \Theta_{k-1}$, and each tree casts a unit vote for the most popular class at input vector $\boldsymbol{x}$ [12]. Contrary to the classification problems where a committee of trees' votes predicts the class, in regression problem, averaging the trees' results is the result of prediction [23, 50].

As detailed in [23], forests can be generated through different techniques such as: bagging, random split selection, and boosting [12]. According to [12], depending on the type of injected randomness, some random forests have lower generalization error than others. Based on the other studies ([14, 18, 22]) mentioned in [12], those algorithms that make the forests by adaptive reweighting of training set (such as Adaboost), outperform the ones that use other techniques (like bagging). The reason behind this better performance could be the fact that they grow trees in an adaptive way in order to remove bias [23].

Compared to simple tree, random forests are not interpretable since their results cannot be displayed in a tree-like diagram [50]. An appealing feature of random forests is its ability to produce variable importance score, that measures the relative contribution of the different variables. The importance score for a particular variable is computed by summing up the decrease in the Gini index for that variable over all the trees in the forests [50].

When the fraction of the number of variables over the number of relevant variables is high, random forest might perform poorly if the number of selected variables be small; there would be a low chance of having relevant variables at each split [23].

2.2 Collaborative Filtering

Collaborative filtering is a technique for predicting unknown values of a matrix called "utility matrix", from the known ones. Traditionally, the matrix represents the ratings of items (considered as columns) by users (considered as rows). Ratings might be explicit, when users provide ratings through a dedicated system, or implicit, when users' behaviours are analysed to estimate their preferences. An overview of collaborative filtering is provided in [31]. Several methods have been proposed to implement collaborative filtering such as Item-Based and User-Based collaborative filtering. Both methods have been used extensively for e-commerce applications.

2.2.1 User-Based Collaborative Filtering

User-Based collaborative filtering [11] predicts the rating of item i by user u from the ratings of item i by users similar to u. There are several techniques to compute the similarity of the co-rated users like cosine-based, correlation-based and adjusted cosine similarities [60].

Pearson correlation is one of the popular similarity measures between two users' ratings. For a set of items P, if the rating of items $I_1, ..., I_p$ by user U_1 presented as $r_{1,1}, r_{1,2}, ..., r_{1,p}$ and their average denoted by $\bar{r}_1$ and similarly rates of $r_{2,1}, r_{2,2}, ..., r_{2,p}$ by user U_2 with average of $\bar{r}_2$, the correlation can be calculated by:

$$Corr(U_1, U_2) = \frac{\sum_{i \in p}(r_{1,i} - \bar{r_1})(r_{2,i} - \bar{r_2})}{\sqrt{\sum_{i \in p}(r_{1,i} - \bar{r_1})^2}\sqrt{\sum_{i \in p}(r_{2,i} - \bar{r_2})^2}}$$

The summation is calculated for the items that are rated by both users. Cosine similarity is another popular measure that is not subtracting the means:

$$CosSim(U_1, U_2) = \frac{\sum_{i \in p}(r_{1,i})(r_{2,i})}{\sqrt{\sum_{i \in p}(r_{1,i})^2}\sqrt{\sum_{i \in p}(r_{2,i})^2}}$$

It is notable that when the data is binary, all the items regardless of being co-rated by both users or not, should be considered in the calculation of Cosine Similarity [31].

User-Based collaborative filtering has difficulty in measuring the similarities between users and the other when the number of users grows. This issue can exponentially increase the computation time of the algorithm [60]. Since the nearest-neighbours approach can be computationally expensive, clustering methods can be another solution. In this method, users can get into alike clusters in terms of preferences and the distance measure could be done between user and each cluster. This clustering approach could be less accurate in comparison with the nearest-neighbours approach [50].

2.2.2 Item-Based Collaborative Filtering

Item-based collaborative filtering [11, 32] predicts the rating of item i by user u from the ratings of items similar to i by user u. Whenever a user expresses interests in an item, the algorithm finds the items that are co-rated by any user and computes the similarity between items.

The similarity of the co-rated items can be computed with the same aforementioned techniques in User-Based collaborative filtering, except that $\bar{r}_1$ defines as the average of rates for our specific item. According to a report by developers of Amazon item-to-item recommendation system [32],

Item-based algorithm produces recommendations in real time and generates high quality recommendations. But despite of the high diversity of users' taste in User-Based recommendation, the number of items in Item-based recommendations is not very varied and therefore recommendations are almost obvious [30]. This method could also calculate item similarities in an offline basis, and this makes it to be able to handle the massive computation caused by rapid increase in the number of users and items [60].

2.2.3 Matrix Factorization

A third class of collaborative filtering methods, the one that we used, is based on the factorization of the utility matrix to estimate latent factors along which users and items are represented. This method is described in [28] and became famous as it contributed to winning the Netflix prize in 2009. This method tries to predict the rates by characterizing both users and items according to inferred factors from rating patterns. Such factors might measure obvious dimensions or well-defined ones but in some cases it could be completely uninterpreted dimensions. This model basically maps the users and items in a joint latent factor space. In matrix factorization, first a relative number of latent factors f is getting fixed. Then each user u is summarized with a f dimensional vector p_u, and each item i is summarized with a f dimensional vector q_i [2]. In this case, the rating of item i by user u can be predicted by $r_{ui} \approx q_i^T p_u$. This rating predictions can be presented in matrix form. The $f \times n$ user matrix P , and the $f \times m$ item matrix Q are defined by:

$$P = \begin{bmatrix} | & & | \\ p_1 & \dots & p_n \\ | & & | \end{bmatrix}, Q = \begin{bmatrix} | & & | \\ q_1 & \dots & q_m \\ | & & | \end{bmatrix}$$

Where $p_1, ..., p_n \in \mathbb{R}^f$ and $q_1, ..., q_m \in \mathbb{R}^f$ are the factor of users and items, respectively. The aim of this method is to complete the rating matrix, and the predicted rating values, $R \approx P^T Q$. This problem can be formulated as an optimization problem in which the aim is to minimize an objective function while P and Q are optimal. In other words, the goal is to minimize the least square error of

the observed ratings.

In practice, the optimization involves a regularization term to avoid overfitting particular users or items. The method finds q_i and p_u that minimize the following objective, where λ is the regularization parameter and $\mathbb{T}$ is the training set:

$$\sum_{(i,u)\in\mathbb{T}} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{1}$$

Two good techniques to perform the optimization are Gradient Descent and Alternating Least Squares (ALS). Gradient Descent [9, 45] turns to be slow and cost lots of iterations [2]. In ALS approach, the one that we used in our study, the set of variables P are fixed and considered as constants; then the objective is a convex function of Q and vice versa. In other words, ALS factories utility matrix R into a product of matrices P and Q of dimensions $p \times f$ and $f \times q$ respectively. Therefore each u and i of matrix R has a f-dimensional feature vector that describes its characteristics. The r_{ui} is the inner product of feature vector of u and feature vector of i. ALS uses following iterative process in order to computes $P \times Q$ for predicting the unknown values of matrix R [62].

(1) Initialize P to a random value;

(2) Optimize Q given P to minimize error on R

(3) Optimize P given U to minimize error on R

(4) Repeat steps 2 and 3 until convergence

It is also common to include user biases b_u and item biases b_i in the optimization, defined as the average deviation of user u and item i to the global average μ. The problem is then to find q_i and p_u that minimize the following objective:

$$\sum_{(i,u)\in\mathbb{T}} \left(r_{ui} - \mu - b_u - b_i - q_i^T p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2\right)$$

In our study we aim to predict if a particular file produced by an analysis pipeline is identical

across execution conditions, or if it contains reproducibility errors. We approach this problem from the point of view of collaborative filtering, inspired by works that applied this method outside of its initial application domain.

Unfortunately, there is not enough previous research work that has been done about the potential application of collaborative filtering in other domains than its prior domains (e-commerce, social networks, or other personalized recommendation purposes). To our best of search, there is just one study [21] that used collaborative filtering to minimize the number of probes in the monitoring techniques. This minimization conducted by the combination of Maximum Margin Matrix Factorization (MMMF) and their proposed selection methods.

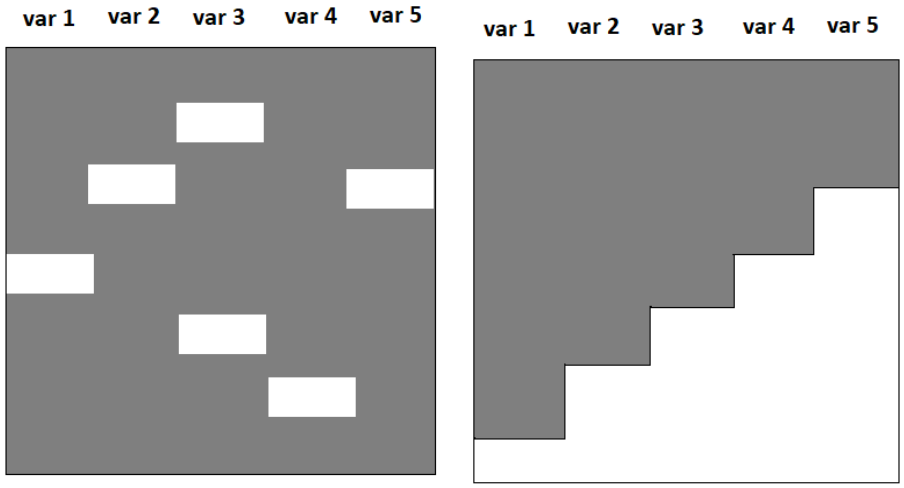
2.3 Sampling Concerns

Data sampling for training set is a critical matter that can affect the accuracy of the predicted values. Sometimes the nature of the problem makes it impossible to sample the training set randomly. For instance, the statistical nature of the problem in [51], is the reason behind not splitting data randomly for model validation and selection. As mentioned in their study, random sampling of training and test sets could also accentuate the variance of predicted results in small data sampling. They present an algorithm that uses Euclidean distance for finding furthest samples in the dataset according to the already sampled data in the training or the test sets. This sampling method is being taken in turn for sampling the two training and test sets. Therefore, the ratio of these two sets are mostly the same (50% of dataset is sampled in each sets). Their results show that the sampling technique can control (decrease) the variability of prediction accuracies across different classification models.

Typically, in recommender system a small fraction of available items are rated by users, and most of data is missing. Missing data can happen in different mechanisms: missing completely at random (MCAR) [34], missing at random (MAR), missing not at random (MNAR). Missing at random in recommender systems context, refers to the probability of a rating being missed that does not depend on its value [52]. In general, when using only observed ratings, matrix factorization of

collaborative filtering assumes that randomness of the system is a MAR mechanism [27]. Sometimes data is not missed at random. For example when i) the users only buy the items that they like and just rate the items that they bought; ii) user is an extremist in giving rate, who just gives the feedback of good or bad for items; or iii) some items are assumed to function properly all the time and they mostly are rated when they are flawed.

Figure 2.2 illustrates the two kinds of missing data pattern. Depending on the patterns of missingness, there are some approaches that can deal with missing data problem. The required approach could be a simple method such as Pairwise Inclusion [20, 48] or an advanced one like Imputation [48].



(a) Arbitrary (b) Monotone

Figure 2.2: Missing value patterns. White indicates the missing value in the data.

Several studies like [27, 52] try to optimize the accuracy of the prediction results in missing not at random mechanism with different approaches. [52] presents appropriate surrogate objective functions for efficient training on MNAR data and [27] propose a probabilistic matrix factorization model for collaborative filtering that learns from data that is MNAR.

Identifying the nature of the problem before applying any learning models is important. According to our research on data sampling and its effect on accuracy of the results, selection and evaluation sets of learning model should be sampled according to the structure of the problem.

Also, ignorance of missing data not at random (MNAR) can lead to a significant reduction in the resulting recommendation accuracy. Missing data issue can be handled by different methods [3], depending on the type of entered variable (continuous, ordinal, nominal), number of missing data, and missingness pattern.

2.4 Summary

There are numerous learning models in Machine Learning that try to predict values in real classification and regression problems. We tried to cover some of the well-known supervised algorithms that are being used in these kind of problems, however a very good comparison between most of these algorithms is done by [29]. In our context, we consider our problem, detection of reproducibility errors, as a kind of classification problem and [21] inspired us to apply collaborative filtering as a potential solution for tackling this problem. The main issue, and originality, of our problem lies in the fact that the training set cannot be arbitrarily sampled from the utility matrix defining the collaborative filtering problem. Instead, the training set has to respect time constraints imposed by the file creation order during pipeline executions. In this regard, we propose some sampling methods for building the training set with respect to the structure of our problem. In the next Chapter we present these methods in detail.

Chapter 3

Collaborative Filtering and Training Set Sampling Methods

This chapter describes our proposed framework to evaluate a pipeline. First, we formalize the problem considered in our study. In order to make our training set we proposed several sampling techniques which are explained in the last section.

3.1 Problem Formulation

The pipeline to be evaluated is represented by a matrix U of size $N_f \times N_s$, where the N_s columns represent data coming from different subjects, and the N_f rows represent the files produced by the pipeline. While subjects are not ordered, files are, for instance from their last modification time in a sequential execution, and we assume that this order is consistent across subjects. $U_{i,j}$ measures the reproducibility of file i produced during the processing of subject j in two conditions, for instance two different operating systems. In our experiments, we restrict $U_{i,j}$ to be boolean, but our methods can be applied for real values as well.

Our goal is to predict the test set $\mathbb{T}'$ of missing values of U from a training set $\mathbb{T}$ of known ones, where $\mathbb{T} \cap \mathbb{T}' = \emptyset$ and $\mathbb{T} \cup \mathbb{T}' = \{U_{i,j}\}$. In contrast with the traditional collaborative filtering

method, we have control over the construction of the training set. For instance, we can choose to include only files produced by specific subjects, or the first files produced by the processing of every subject. Moreover, the construction of the training set is constrained by the order of the matrix rows, which is formalized as follows:

$$\begin{aligned} &\forall (i,j,k) \in [\![1, N_f]\!] \times [\![1, N_f]\!] \times [\![1, N_s]\!], \\ &(U_{i,k} \in \mathbb{T} \quad \text{and} \quad U_{j,k} \in \mathbb{T}') \Rightarrow i < j. \end{aligned} \tag{2}$$

Our problem is as follows:

> Given a training ratio α, find a subset $\mathbb{T}$ of $\{U_{i,j}\}$ of size $\alpha N_f N_s$ such that (1) $\mathbb{T}$ and $\mathbb{T}'$ respect Equation 2, and (2) $\mathbb{T}'$ can be predicted from $\mathbb{T}$ with high accuracy.

In this study, we use ALS [28] as implemented in Apache Spark's MLLib version 2.3.0. We also round predictions to the nearest integer to obtain binary values.

3.2 Training Set Sampling Methods

As explained before, the training set in our problem cannot be constructed by unconstrained random sampling of the matrix. Instead, the training and test sets have to comply with Equation 2. To address this issue, we investigated the following sampling techniques, illustrated in Figure 3.1.

3.2.1 Complete Columns

The training set is sampled by randomly selecting complete columns in the utility matrix, that is, complete subject executions (Figure 3.1a). The last selected column might be incomplete to meet the exact training ratio. This method respects the time constraints. It corresponds to a situation where the collaborative filtering method will predict the reproducibility of the subjects in the test set from the subjects in the training set.

3.2.2 Complete Rows

The training set is sampled by selecting complete rows in the utility matrix, that is, the first files produced by every execution (Figure 3.1b). The last selected row might be incomplete to meet the exact training ratio. This method respects the time constraints. It corresponds to a situation where the processing of all the subjects is launched and interrupted before the execution is complete. The collaborative filtering method will then predict the reproducibility of the remaining files.

3.2.3 Random Subjects – RS

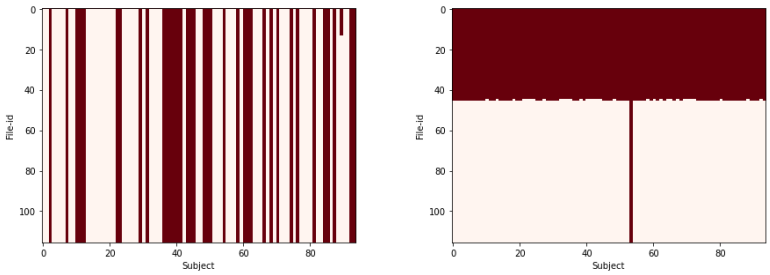
RS method builds the training set by selecting the files from random subjects until the training ratio is reached (Figure 3.1c). The file selected in a subject is the file with the lowest index in this subject that has not been already selected in the training set, which respects the time constraints.

3.2.4 Random File Numbers (Uniform) – RFNU

In RFNU method as Figure 3.1d shows, the number of files selected for every subject is randomly selected in a uniform distribution $U(a, b)$, where b is set to the total number of files N_f and a is set according to the training ratio α as follows:

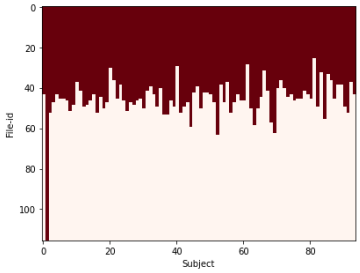
$$\begin{cases} a = 0 & \text{if } \alpha \leq 0.5 \\ a = (2\alpha - 1)N_f & \text{if } \alpha > 0.5 \end{cases}$$

For $\alpha \leq 0.5$, we ensure that the average number of selected files by subject is αN_f by sampling the number of files in every subject from $U(0, N_f)$ with probability 2α, and setting it to 0 otherwise. For $\alpha > 0.5$, this is ensured by the value of a, which leads the average of U(a, b) to be αN_f.

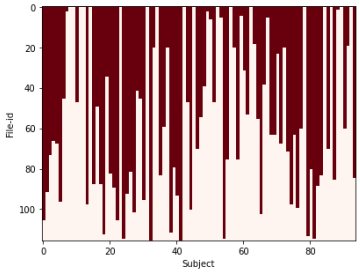


(a) Complete Columns

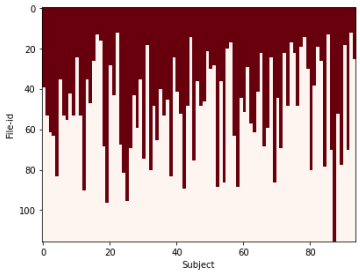
(b) Complete Rows



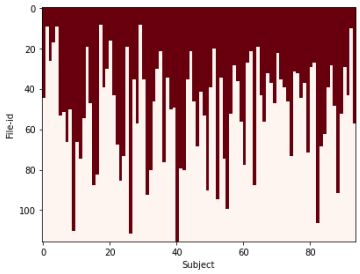
(c) Random Subjects (RS)



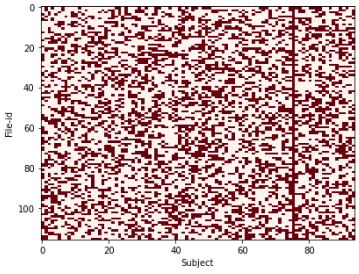
(d) RFNU



(e) RFNTL



(f) RFNTS



(g) Random Unreal

Figure 3.1: Training sets for different sampling methods, $\alpha = 0.4$ (dark elements are in the training set).

3.2.5 Random File Numbers (Triangular) – RFNT

In RFNT the number of files selected for every subject is randomly selected in a triangular distribution $T(a, b, c)$ as in Figure 3.2. The mean of the distribution is $\frac{a+b+c}{3}$. We set c to N_f and we set a and b with the following two approaches.

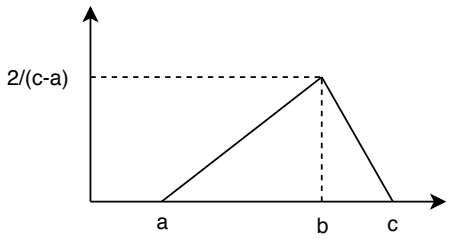


Figure 3.2: Triangular distribution $T(a, b, c)$

Largest a

In this approach known as RFNTL , an example shown in Figure 3.1e , a is set to the largest possible value, i.e., b, and b is set accordingly to ensure that the average of the distribution is αN_f. Two cases occur:

- When $\alpha > 1/3$:
$$a = b = \frac{3\alpha - 1}{2} N_f$$
- When $\alpha \leq 1/3$: $a = b = 0$, the number of files selected in every subject is sampled from $T(0, 0, N_f)$ with probability 3α and set to 0 otherwise.

Smallest a

In this approach known as RFNTS, an example shown in Figure 3.1f, a is set to the smallest possible value, i.e., 0, and b is set accordingly to ensure that the average of the distribution is αN_f. Three cases occur:

- When $\alpha < 1/3$: $a = b = 0$, the number of files selected in every subject is sampled from $T(0, 0, N_f)$ with probability 3α and set to 0 otherwise.

- When $1/3 \leq \alpha \leq 2/3$:

$$a = 0 \quad ; \quad b = N_f(3\alpha - 1)$$

- When $\alpha \geq 2/3$: $b = N_f$, the number of files selected in every subject is sampled from $T(0, N_f, N_f)$ with probability $3(1 - \alpha)$ and set to N_f otherwise.

As illustrated in Figures 3.1e and 3.1f, the point of the RFNTL method is to guarantee that, for large enough values of α, all subjects will have at least a few files in the training set (no empty column), which is not the case for RFNU or RFNTS.

3.2.6 Random Unreal

The training set is sampled in a random uniform way, regardless of the file creation times (Figure 3.1g). This method does not respect the time constraints in Equation 2, however, it will be used as a baseline for comparison.

In each method, we also included the first row of the matrix (first file of each subject) and a random column (all files of a random subject) to avoid cold start issues (having no initial information for specific item or user) .

3.3 Summary

As it was mentioned in the previous Chapter, data sampling strategy of training set can affect the accuracy and quality of the predicted value. According to our limitation, the time constraint, in our study we proposed six sampling methods that have different strategies for sampling the training set and all of them are respecting the training ratio. Some of these methods try to sample data either with having all samples of few subjects or few samples of all subjects. But there are some

methods that try to maximize the number of selected subjects while the most possible number of samples from these subjects are selected for training set. This maximization is done by two kinds of distributions; uniform and triangular. In the next chapter we describe our two types of datasets that we used to apply our sampling methods on.

Chapter 4

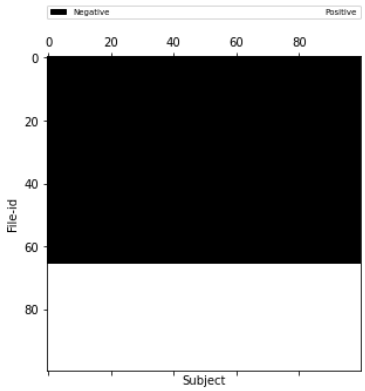
Data Collection

In this chapter we describe the two types of datasets used in our study. First, a synthetic dataset simulates different possible cases that might be observed in the analysis pipelines. Second, a real dataset was acquired using neuroinformatics analysis pipelines.

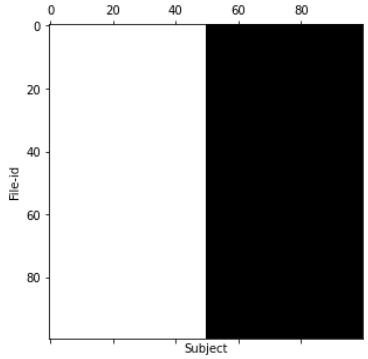
4.1 Synthetic Dataset

We generated synthetic matrices as shown in Figure 4.1. Each matrix has 100 files and 100 subjects of different types. Subjects of the same type behave identically and all types contain the same number of subjects $\pm$ 1. Such a decomposition by subject type corresponds to a situation where different subjects may have data of different nature, as is the case in the Human Connectome Project data [57] where not all the subjects have the same amount of images.

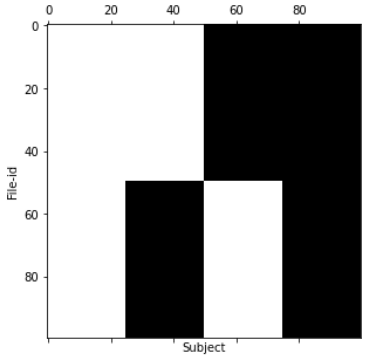
For subjects of a given type, the matrix consists of $log(n)$ blocks, where n is the number of types. Blocks are defined with all the possible variation patterns: some types do not vary at all, while other ones vary between every block. Such patterns are meant to mimic the logic of data processing pipelines: each block of files represents the files produced at a given stage of the pipeline, which may or may not contain reproducibility errors depending on the subject type.



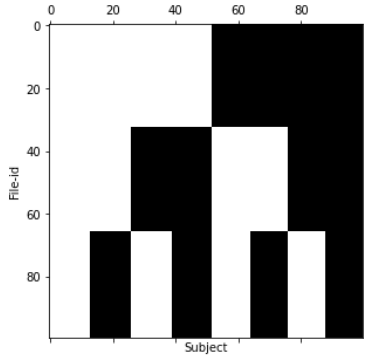
(a) 1 type



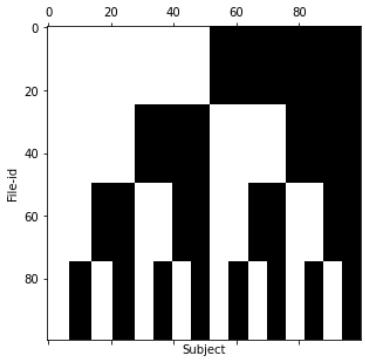
(b) 2 types



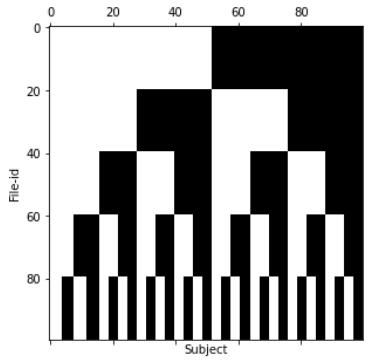
(c) 4 types



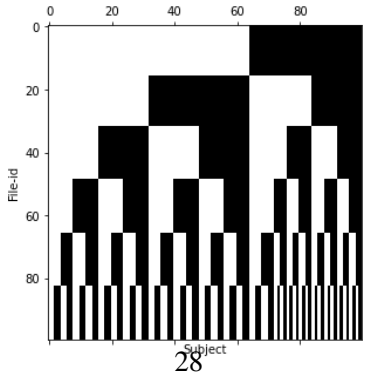
(d) 8 types



(e) 16 types



(f) 32 types



(g) 64 types

Figure 4.1: Synthetic reproducibility matrices. White cells denote reproducibility errors.

4.2 Human Connectome Project (HCP)

Human Connectome Project (HCP) pipelines aim to provide free high quality of neuroimaging data for characterizing human brain connectivity and function [57]. Figure 4.2 shows an overview of the HCP Minimal processing pipelines and the overall workflow for preprocessing and analysis in the HCP. HCP contains three structural pipelines (PreFreeSurfer, FreeSurfer, and PostFreeSurfer) and also has two functional pipelines (fMRIVolume and fMRISurface) as well as a Diffusion Preprocessing pipeline [24]. According to [24] the overall goals of the HCP Minimal processing pipelines can be listed as below:

(1) Remove spatial artifacts and distortions

(2) Generating cortical surfaces, segmentations, and myelin maps;

(3) Making the data easily viewable in the Connectome Workbench visualization software

(4) generating precise within-subject cross-modal registrations;

(5) Handling surface and volume cross-subject registrations to standard volume and surface spaces;

(6) Making the data available in the CIFTI format in a standard "grayordinate" space

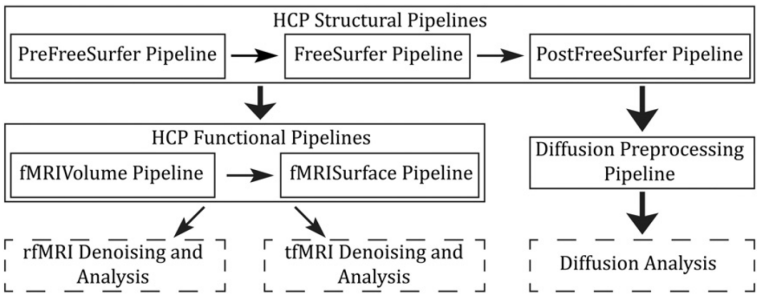


Figure 4.2: HCP minimal preprocessing pipelines.

4.3 Real Dataset

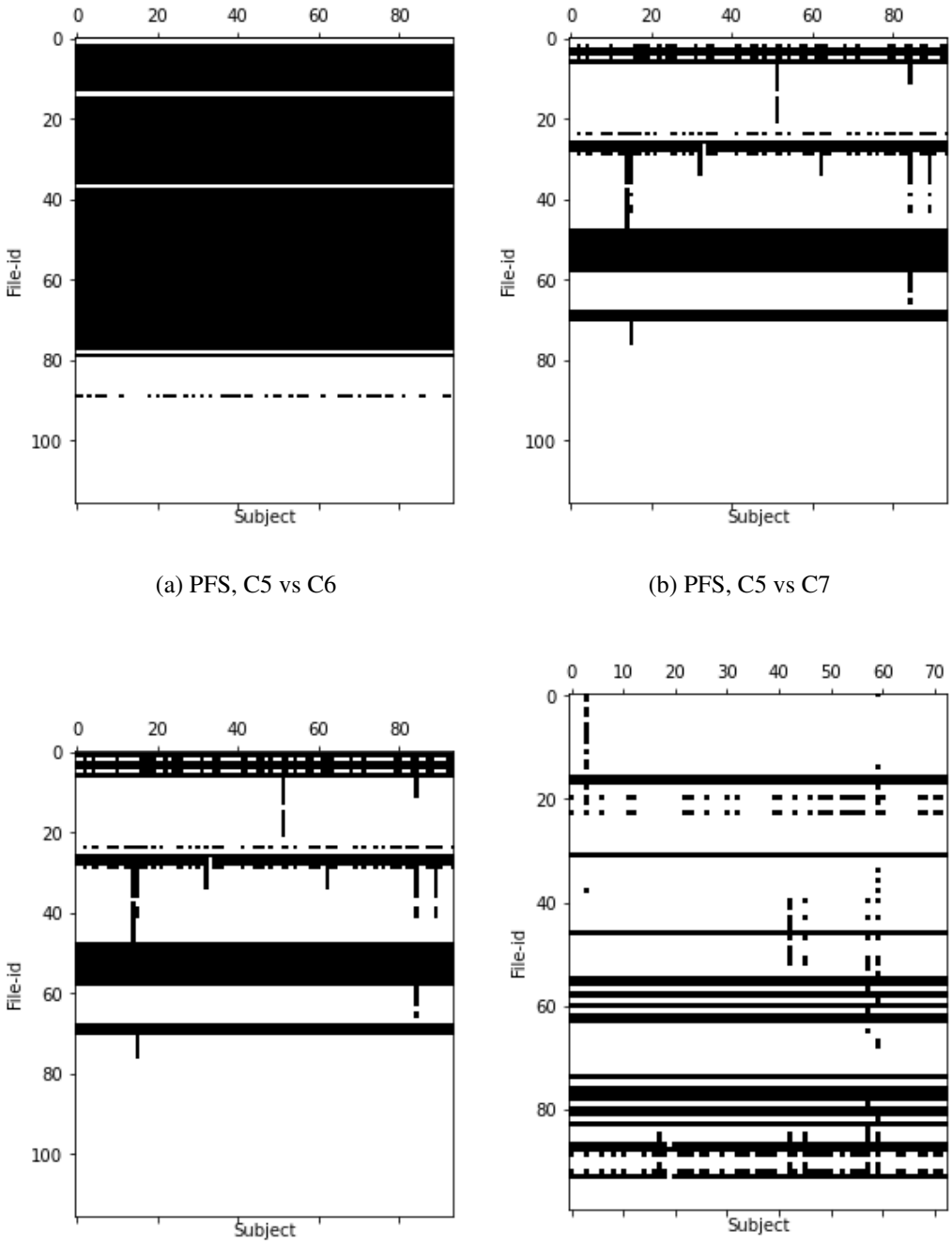
We processed a set S of 94 subjects randomly selected in the S500 HCP release[1] of the Human Connectome Project [24], in three execution conditions with different versions of the CentOS operating system (5.11, 6.8 and 7.2 – referred as C5, C6 and C7), using the PreFreesurfer and Freesurfer pipelines described in [24] and available on GitHub[2]. For each pipeline, we identified the set F of files produced for all subjects in all conditions. For each condition pair and each pipeline, we computed a binary reproducibility matrix U of size $|F| \times |S|$, where $U_{i,j}$ is true if and only if file i of subject j was different in each condition. Rows of U were ordered by ascending file modification time in a random subject in S.

Figure 4.3 shows the matrices for PreFreesurfer and Freesufer. The reproducibility of these pipelines varies across subjects, but most files behave consistently across all subjects, leading to complete black or white lines.

4.4 Summary

We conduct our study on two different types of datasets. Synthetic datasets are built to have a better understanding of each proposed sampling methods since since their utility matrices contain clear patterns. Real datasets are built for having a real case for use our sampling methods. we used HCP pipelines since they have high impact in the neuroimaging domain. For making our real datasets we used the comparison results [47] getting by a framework [3] for processing the data along the two HCP structural processing pipelines (PreFreeSurfer and FreeSurfer).

[1] https://db.humanconnectome.org
[2] https://github.com/Washington-University/Pipelines/releases/tag/v3.19.0
[3] https://github.com/big-data-lab-team/repro-tools



(a) PFS, C5 vs C6

(b) PFS, C5 vs C7

(c) PFS, C6 vs C7

(d) FS (100 files), C6 vs C7

Figure 4.3: Utility matrices for PreFreesurfer (PFS) and Freesurfer (FS). White cells denote reproducibility errors.

Chapter 5

Results

We conducted two experiments for each reproducibility matrix, to evaluate the performance of our predictions using (1) ALS without biases, and (2) ALS with subject and file biases. We compare the performance of our sampling methods to (1) a dummy classifier that always predicts the value in the majority class and (2) the Random Unreal method, used as the baseline sampling technique. All reported values are averages over 5 repetitions. Due to sampling issues, it is possible that the actual training ratio obtained with some of the sampling methods does not exactly match the target one. We checked that the difference between the target and real training ratios was lower than 0.01. We used Spark's ALS model as available in `pyspark.ml`, with 50 factors, a regularization parameter of 0.01, a maximum of 5 iterations and non negative constraints set to true. The code and data used to obtain the results are available through GitHub at https://github.com/big-data-lab-team/paper-reproducibility-collaborative-filtering.

5.1 Accuracy on Synthetic Dataset with ALS without Bias Technique

Figure 5.1 shows accuracy results for ALS without bias. By construction, the accuracy of the dummy classifier is close to 0.5 for all subject types. Random Unreal performs well for all subject types, which confirms that ALS is working correctly. All the other methods perform better than the

dummy classifier, and their accuracy decreases as the number of subject types increases. However, only 3 methods can provide accuracy values above 0.85 for all subject types: Random Subjects (RS), RFNU and RFNTL. Surprisingly, RFNTS does not perform well for more than 2 subject types, even for $\alpha = 0.9$.

5.2 Accuracy on Synthetic Dataset with ALS with Bias Technique

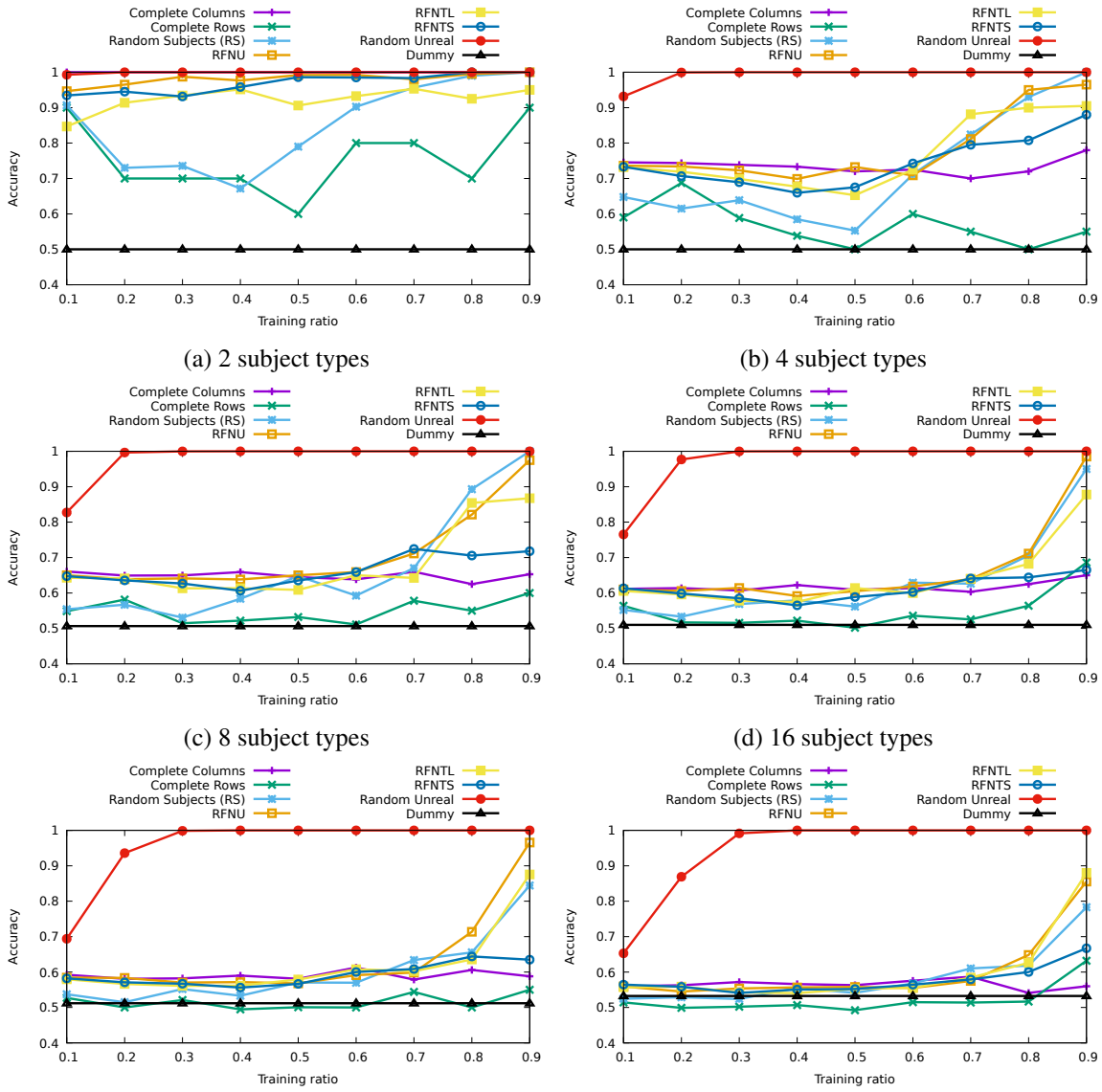
Figure 5.2 shows accuracy results for ALS with bias. RS, RFNU and RFNTL remain the methods that best compare to Random Unreal, but their accuracy is much lower than without biases. This is presumably due to the fact that file biases are all close to 0.5. In such a situation, biases are detrimental and should not be included.

5.3 Accuracy on Real Dataset with ALS without Bias Technique

Figure 5.3 shows the accuracy for ALS without bias. Among the methods that performed well in the synthetic dataset, RFNU performs the best, with an accuracy higher than 0.95 for all datasets when the training ratio is larger than 0.5. Complete rows and complete columns do not reach the accuracy of the dummy classifier. Random Unreal has an accuracy close to 1, which shows that collaborative filtering works well on this dataset too.

5.4 Accuracy on Real Dataset with ALS with Bias Technique

Figure 5.4 shows the accuracy for ALS with bias. From a training ratio of 0.5, all methods perform well for all datasets, except complete columns in Figure 5.4b. All methods except RFNU, RFNTL and RFNTS also perform well for a training ratio lower than 0.5. Overall, the accuracy is much higher than without bias, due to the fact that the file biases are very strong. In the remainder, all the results are obtained using ALS without bias.



(a) 2 subject types

(b) 4 subject types

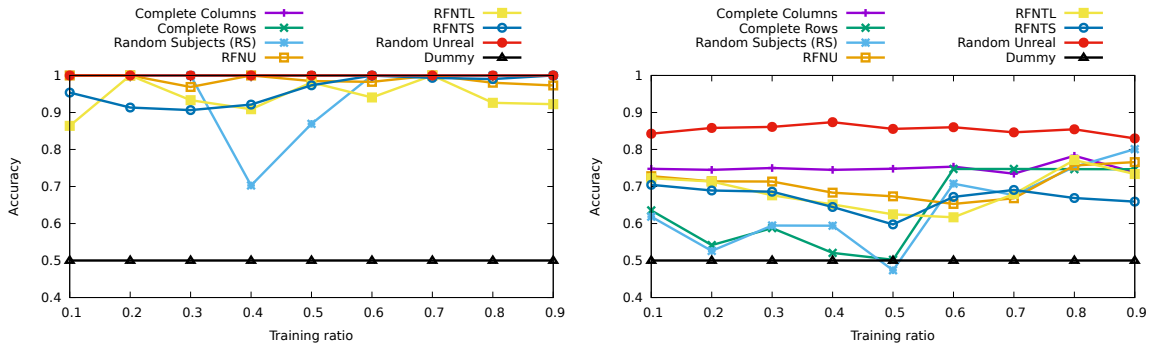
(c) 8 subject types

(d) 16 subject types

(e) 32 subject types

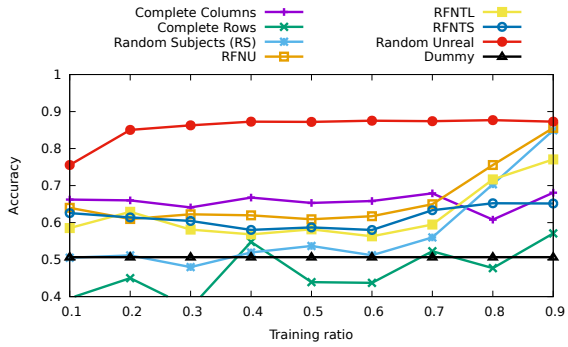
(f) 64 subject types

Figure 5.1: Accuracy results on synthetic data, ALS without bias.

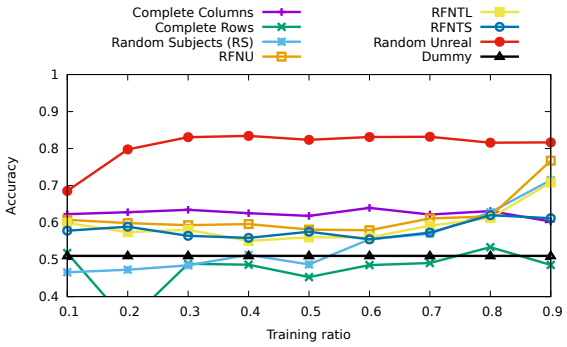


(a) 2 subject types

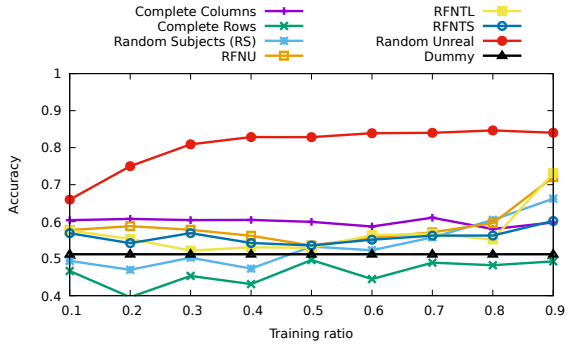
(b) 4 subject types



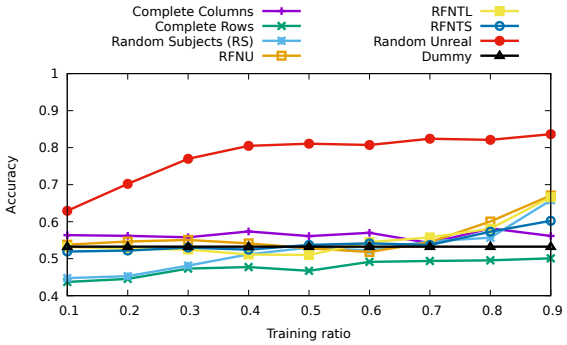
(c) 8 subject types



(d) 16 subject types

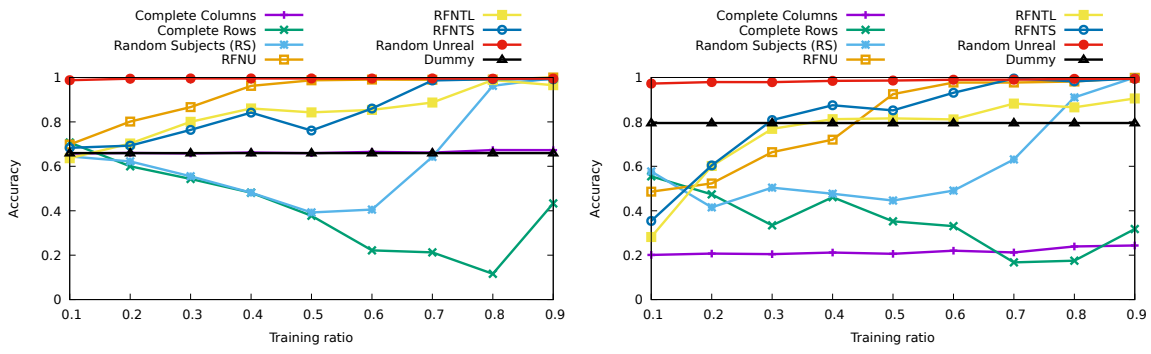


(e) 32 subject types



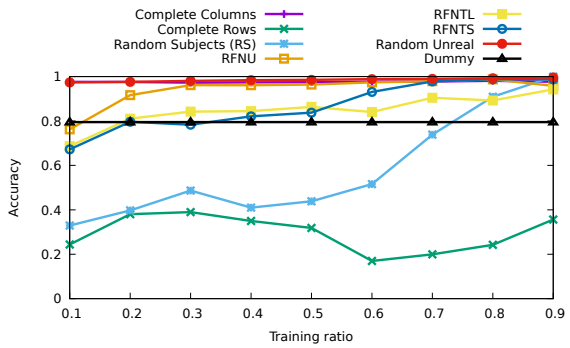
(f) 64 subject types

Figure 5.2: Accuracy results on synthetic data, ALS with bias.

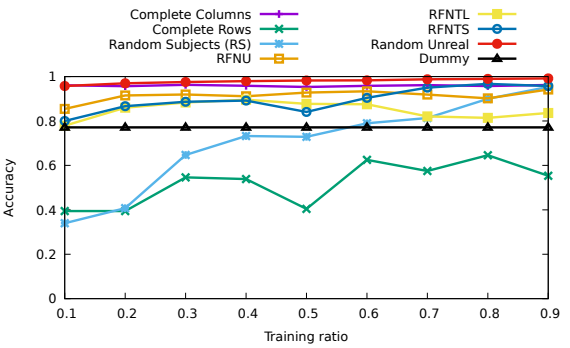


(a) PFS, CentOS5 vs CentOS6

(b) PFS, CentOS5 vs CentOS7

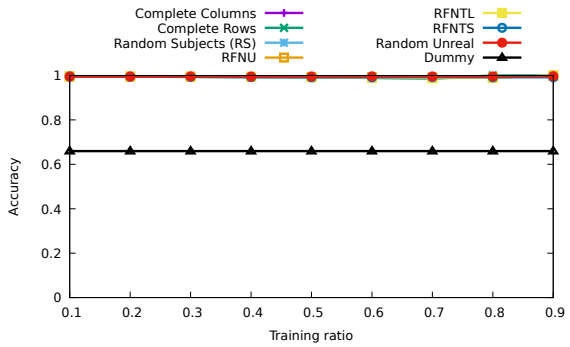


(c) PFS, CentOS6 vs CentOS7

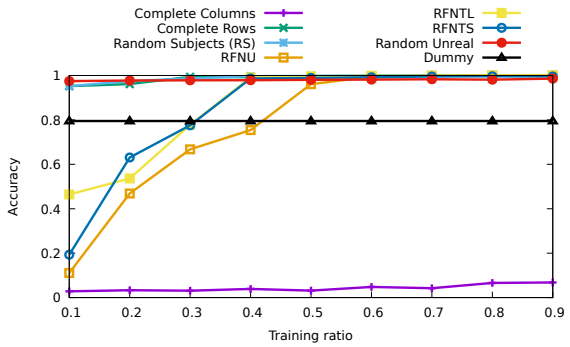


(d) FS, CentOS6 vs CentOS7

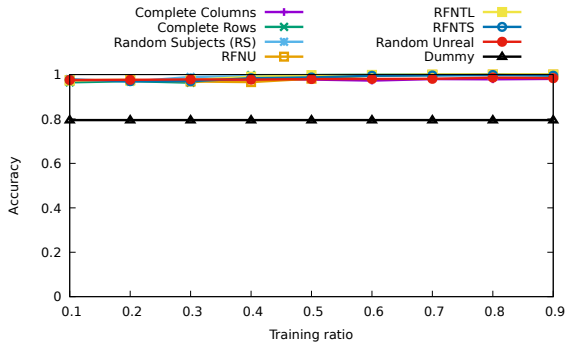
Figure 5.3: Accuracy results on PreFresurfer (PFS) and Freesurfer (FS) data, ALS without bias.



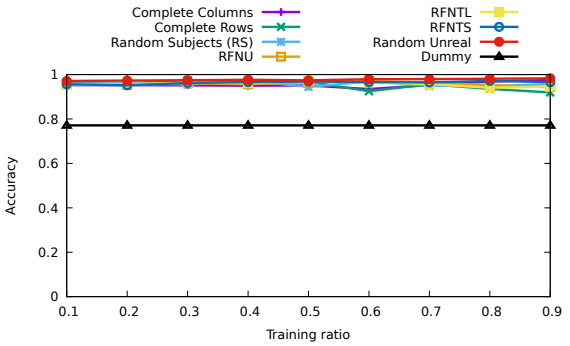
(a) PFS, CentOS5 vs CentOS6



(b) PFS, CentOS5 vs CentOS7



(c) PFS, CentOS6 vs CentOS7



(d) FS, CentOS6 vs CentOS7

Figure 5.4: Accuracy results on PreFresurfer (PFS) and Freesurfer (FS) data, ALS with bias.

	Sensitivity	Specificity
Complete Columns	0.53	0.97
Complete Rows	0.43	0.89
Random Subjects	0.88	0.99
RFNU	0.92	0.99
RFNTL	0.81	0.97
RFNTS	0.65	0.93
Random Unreal	1	1

Table 5.1: Average sensitivity and specificity for the synthetic datasets and Freesurfer ($\alpha = 0.9$, ALS without bias).

5.5 Receiver Operating Curve (ROC) Analysis

ROC curve is depicting the performance by an anomaly detection technique [61] The closer to the top-left corner(representing the false positive rate 0 and the true positive rate 1 in figure 5.5) of the chart, the better performance the anomaly technique produces. Figure 5.5 compares the sampling methods in the ROC space, for a training ratio of 0.9, on the synthetic and Freesurfer dataset. The PreFreesurfer dataset is not included since specificity of RFNTL, RFNU, RS and complete rows is undefined at this training ratio (the test set only contains positive elements). The average sensitivity and specificity values are reported in Table 5.1, which confirms that RFNU is the best performing method on average.

5.6 Effect of the Number of Factors

Figure 5.7 shows the effect of the number of factors used in the ALS optimization for the RFNU method and the synthetic dataset with 8 subject types (3 blocks of files). For training ratios greater than 0.5, the accuracy with 3 factors is substantially higher than with 2 factors. Beyond 3, the number of factors does not have any effect, which shows that the data is best explained by 3 factors. Figure 5.9 confirms that 3 factors are enough to separate the files of this dataset in 3 blocks, and the subjects in 8 types.

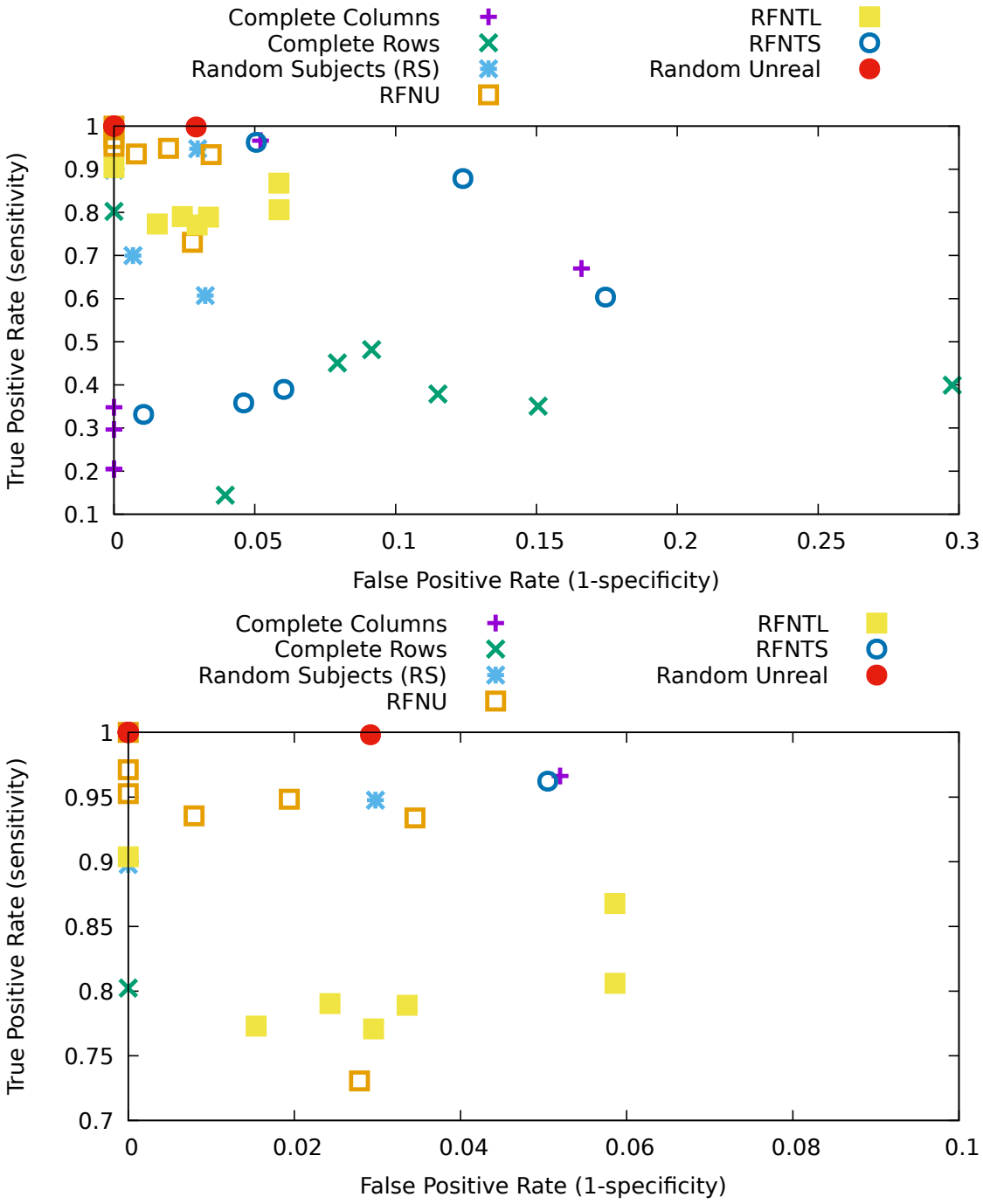


Figure 5.5: Comparison of the sampling methods in the ROC space for the 6 synthetic datasets and Freesurfer (ALS without bias). Right: entire dataset. Left: close-up on the top-left part.

5.7 Effect of the Maximum Number of Iterations

Figure 5.8 illustrates the effect of the number of iterations used in the ALS optimization, for the RFNU method and the synthetic dataset with 8 types. From a training ratio of 0.7, the number of iterations has a moderate effect on the accuracy. We used 5 iterations in our experiments, which only slightly degrades the accuracy compared to 15 or 20.

5.8 Prediction Error Localization

Figure 5.10 compares the prediction errors made by RFNU, RFNTL and RFNTS. The training set is represented in black (negatives) and white (positives), and the test set is in green (true positives), yellow (false negatives), gray (true negatives) and red (false positives). This representation provides insights regarding where, and perhaps why, prediction errors occur. At this training ratio, RFNU (Figure 5.10a) uniformly samples the number of files per subject between $0.8N_f$ and N_f, which enables the training on the last files of the pipeline, while maintaining a low number of columns with a low training ratio. On the contrary, RFNTL (Figure 5.10b) samples the number of files per subject from $0.85N_f$, but the probability to have a complete column is 0 (the one complete column in Figure 5.10b is the one included to prevent cold start issues), which leads to a "stripe" of prediction errors at the bottom of the matrix. RFNTS (Figure 5.10c) does not have this issue, as it includes many complete columns in the training set. However, it comes at the cost of several columns with a number of files lower than 60%: in such columns, the prediction is often entirely wrong, which explains the reduced accuracy compared to RFNU.

Figure 5.6 shows the RFNU results on the Freesurfer dataset for a training ratio of 0.6. The prediction error is localized (1) at the bottom of the matrix, which corresponds to the end of the execution, and (2) in the regions where file reproducibility varies across subjects, i.e., lines are not entirely black or entirely white. This is consistent with our expectations and confirms the validity of our results.

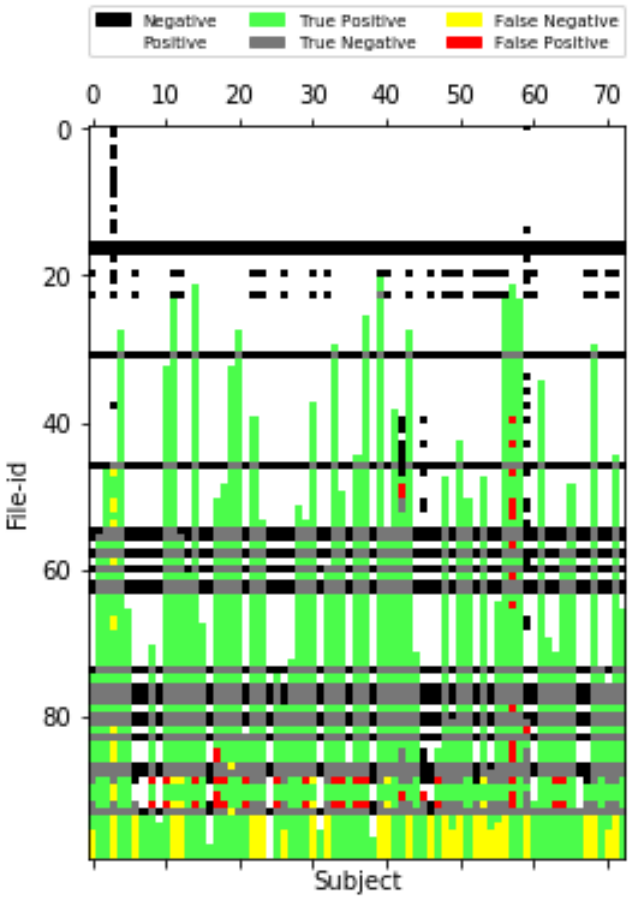


Figure 5.6: Prediction results for RFNU on the Freesurfer dataset, $\alpha = 0.6$.

5.9 Summary

Besides evaluating the accuracy of each sampling method we compared these methods in the ROC space. The average sensitivity and specificity values support the general outperforming of RFNU. Also we had some experience about the effect of the number of latent factors and iterations specifically for RFNU method to find the optimal number for each of them. Depicting the subject and file latent factor vectors ensure that by use of RFNU, matrix factorization can effectively determine the different types of subjects or files of the dataset.

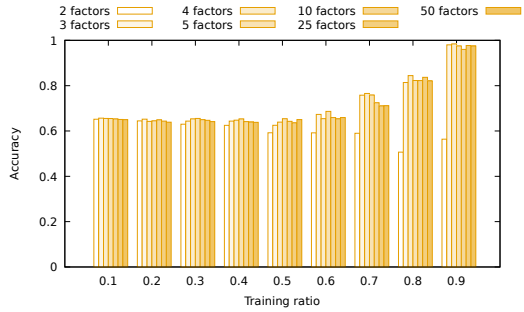


Figure 5.7: Effect of the number of factors used in ALS, (RFNU, synthetic dataset, 8 types, 5 iterations).

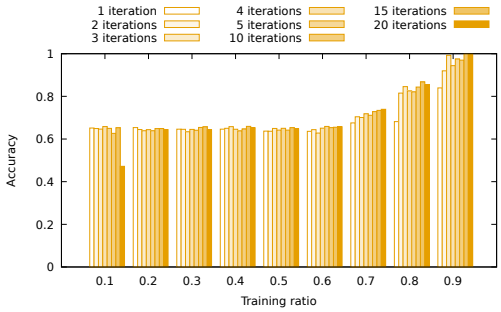
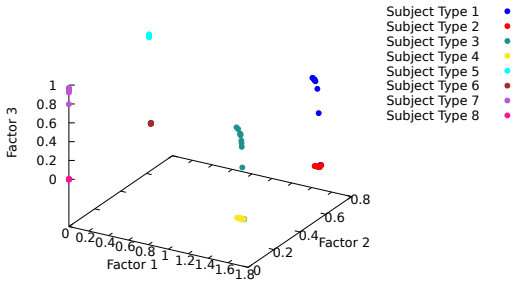
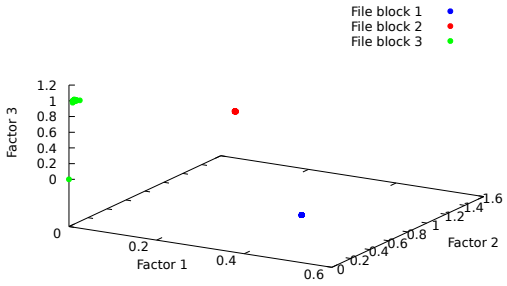


Figure 5.8: Effect of the number of iterations used in ALS, (RFNU, synthetic dataset, 8 types, 50 factors).

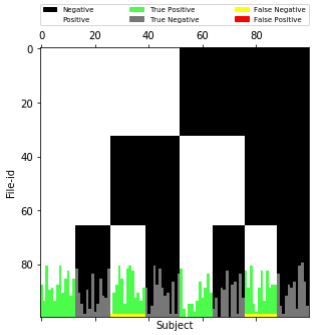


(a) Subject factors

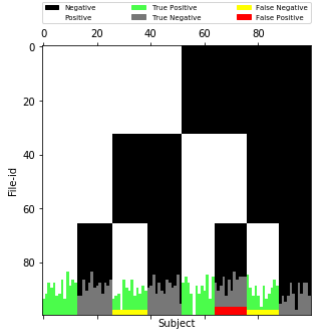


(b) File factors

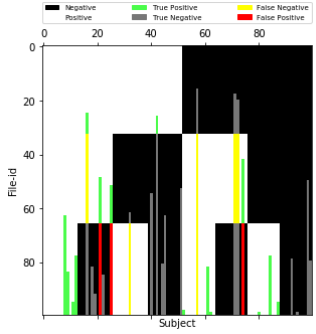
Figure 5.9: Subject and file factors produced by collaborative filtering (RFNU, synthetic dataset, 8 types, 3 factors, 5 iterations).



(a) RFNU: $U(0.8N_f, N_f)$



(b) RFNTL: $T(0.85N_f, 0.85N_f, N_f)$



(c) RFNTS: $0.3T(0, N_f, N_f)$+$0.7N_f$

Figure 5.10: Comparison between RFNU, RFNTL and RFNTS on synthetic dataset with 8 types, $\alpha = 0.9$.

Chapter 6

Conclusion

While collaborative filtering, perhaps unsurprisingly, correctly predicts the missing values in a matrix modeling reproducibility evaluations, the usual random sampling method cannot be used in time-constrained processes. Processing of each subject in neuroimaging pipelines produces several files sequentially. This time generation characteristic of files in the pipelines is known as time-constrained for sampling the training set in our study. Basically random training sampling is impractical in this type of problems since it corresponds to the situation that a file has been chosen to be sampled in the training set while it might not have been produced in the process so far; just because the training set size has been met. Also the dependency of some files to their previous ones in terms of their generation in the processing of the subject makes prediction unrealistic if a dependent file is sampled in training set which that means its reproducibility value is known while the independent file assume is not generated yet and its value is going to be predicted.

We proposed six sampling methods to address sampling issue. Two ordinal sampling methods, Complete Columns and Complete Rows, one Random Subject (RS) and three distributed random sampling methods, RFNU, RFNTL, RFNTS. By comparison results of all methods in two kinds of datasets, synthetic and real, we found that one of them, RFNU, performs better than the other ones on average. We explain that by the fact that RFNU builds the training set using a balanced mix of nearly complete and nearly empty columns, with a continuum of intermediate configurations. On

the contrary, other methods, including RFNTL and RFNTS, bias the sampling toward complete or empty columns, which is sometimes detrimental to accuracy.

Besides biases effect, dominant value of the first execution files that are sampled in the training set plays important role for the accuracy of some methods such as complete columns, RFNU and both RFNT approaches. As much as the ratio of reproducibility error and not evident error values be the same the accuracy of these methods decrease whereas RS and complete rows methods which have almost the same behavioural performance and in this case they have better accuracy for very low training ratios. Complete columns is the most vulnerable method to this equivalent of classes value related to the first executed files, specially when there is high file bias in the dataset. This may relate to ALS optimization parameters(the number of iterations or latent factors)that might base happen on the first sampled files.

For datasets that are strongly dependent on row bias, such as the PreFreesurfer and Freesurfer ones, RFNU provides accuracy values consistently higher than 0.95 for training ratios higher than 0.5, even when biases are not included. For the synthetic dataset, RFNU still performs very well when biases are not included, but it does very poorly when biases are included. For this reason, we recommend to not include biases in the collaborative filtering optimization to solve this problem. Even though biases provide a slight accuracy improvement when datasets are strongly biased (constant lines in the matrix), they can also be very detrimental for more complex datasets such as the synthetic one used here.

From a practical standpoint, our study shows that reproducibility evaluations of the PreFreesurfer and Freesurfer pipelines can be conducted using only 50% of the files produced, with an accuracy above 95%. Potentially, this could reduce the computing time and storage required for such studies by a factor of 2. However, such studies could not be conducted by processing only half of the subjects entirely, which would correspond to the complete columns sampling method. Instead, the processing of all the subjects should be initiated and terminated in a random uniform way, assuming that files are uniformly produced throughout the execution. On more complex matrices, such as the synthetic dataset studied here, the training ratio required to get a 95% accuracy increases to 0.85.

This study can make a contribution to solve the reproducibility errors. Prediction of values of reproducibility evaluation matrix can be helpful to identify the files with possibility of not being reproducible over the problem's conditions.

Our study could be extended to real-valued reproducibility matrices instead of just binary ones. It is indeed common for file differences to be quantified using specific similarity measures or distances, such as the Levenshtein distance between strings, or the sum of squared distances among voxels of an image. Our sampling methods could be directly applied to real-valued matrices, and we expect our conclusions on the best-performing sampling method (RFNU) and the inclusion of biases in the model to remain valid.

The method described in this thesis could possibly be used to predict the outcome of other time-constrained processes, for instance markers of chronic disease activity.

Sampling method could be considered as a missing data problem. The value of those files with no data in the training set could be equivalent to unobserved ratings in the data missing not at random situation; [52] is an example of data missing not at random. In another words, the training set output of the sampling method can be a utility matrix in the data missing problem that the prediction is not occur only by use of observed values but also the non random missing data.

Glossary

ALS Alternating Least Squares. 16

HCP Human Connectome Project. 29

RFNT Random File Numbers with use of Triangular distribution. 24

RFNTL Random File Numbers with use of Triangular distribution when parameter a set to the Largest possible value. 24

RFNTS Random File Numbers with use of Triangular distribution when parameter a set to the Smallest possible value. 24

RFNU Random File Numbers with use of Uniform distribution. 22

RS Random Subjects. 22

Bibliography

[1] Artifact Review and Badging. Available at https://www.acm.org/publications/policies/artifact-review-badging, Accessed on: April 20, 2018.

[2] Matrix completion via alternating least square(als). https://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf. Accessed: 2018-10-29.

[3] The mi procedure imputation methods, sas institute. https://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_mi_sect019.htm. Accessed: 2018-11-04.

[4] Random forests leo breiman and adele cutler. https://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm. Accessed: 2018-10-30.

[5] Support vector machines. http://scikit-learn.org/stable/modules/svm.html. Accessed: 2018-10-19.

[6] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. Machine Learning, 6(1):37–66, 1991.

[7] Monya Baker. Is there a reproducibility crisis? A Nature survey lifts the lid on how researchers view the crisis rocking science and what they think will help. Nature, 533(7604):452–455, 2016.

[8] Soudabeh Barghi, Lalet Scaria, Ali Salari, and Tristan Glatard. Predicting computational reproducibility of data analysis pipelines in large population studies using collaborative filtering. arXiv preprint arXiv:1809.10139, 2018.

[9] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer, 2010.

[10] Alexander Bowring, Camille Maumet, and Thomas Nichols. Exploring the impact of analysis software on task fMRI results. bioRxiv, page 285585, 2018.

[11] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[12] Leo Breiman. Random forests. Machine Learning, 45(1):5–32, 2001.

[13] Leo Breiman. Classification and regression trees. Routledge, 2017.

[14] Leo Breiman et al. Arcing classifier (with discussion and a rejoinder by the author). The Annals of Statistics, 26(3):801–849, 1998.

[15] Priyanga Chandrasekar, Kai Qian, Hossain Shahriar, and Prabir Bhattacharya. Improving the prediction accuracy of decision tree mining with data preprocessing. In Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual, volume 2, pages 481–484. IEEE, 2017.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine Learning, 20(3):273–297, 1995.

[17] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, and Shichao Zhang. Efficient knn classification algorithm for big data. Neurocomputing, 195:143 – 148, 2016. Learning for Medical Imaging.

[18] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning, 40(2):139–157, 2000.

[19] Chris Drummond. Replicability is not reproducibility: Nor is it good science, June 2009.

[20] Craig K Enders. Applied missing data analysis. Guilford press, 2010.

[21] Dawei Feng, Cécile Germain, and Tristan Glatard. Efficient distributed monitoring with active collaborative prediction. Future Generation Computer Systems, 29(8):2272–2283, 2013.

[22] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In Icml, volume 96, pages 148–156. Citeseer, 1996.

[23] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[24] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. The minimal preprocessing pipelines for the Human Connectome Project. Neuroimage, 80:105–124, 2013.

[25] Tristan Glatard, Lindsay B Lewis, Rafael Ferreira da Silva, Reza Adalat, Natacha Beck, Claude Lepage, Pierre Rioux, Marc-Etienne Rousseau, Tarek Sherif, Ewa Deelman, et al. Reproducibility of neuroimaging analyses across operating systems. Frontiers in Neuroinformatics, 9:12, 2015.

[26] Ed HBM Gronenschild, Petra Habets, Heidi IL Jacobs, Ron Mengelers, Nico Rozendaal, Jim Van Os, and Machteld Marcelis. The effects of FreeSurfer version, workstation type, and macintosh operating system version on anatomical volume and cortical thickness measurements. PloS one, 7(6):e38234, 2012.

[27] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In International Conference on Machine Learning, pages 1512–1520, 2014.

[28] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, (8):30–37, 2009.

[29] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. Emerging Artificial Intelligence Applications in Computer Engineering, 160:3–24, 2007.

[30] Johannes Ledolter. Data mining and business analytics with R. John Wiley & Sons, 2013.

[31] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets. Cambridge university press, 2014.

[32] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, (1):76–80, 2003.

[33] Pierre Lison. An introduction to machine learning, 2015.

[34] Roderick JA Little and Donald B Rubin. Statistical analysis with missing data, volume 333. John Wiley & Sons, 2014.

[35] Wei-Yin Loh. Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1):14–23, 2011.

[36] Marcia McNutt. Reproducibility. Science, 343(6168):229–229, 2014.

[37] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

[38] Sushmita Mitra and Tinku Acharya. Data mining: multimedia, soft computing, and bioinformatics. John Wiley & Sons, 2005.

[39] Regina Nuzzo. Scientific method: statistical errors. Nature News, 506(7487):150, 2014.

[40] Su-lin Pang and Ji-zhang Gong. C5. 0 classification algorithm and application on individual credit evaluation of banks. Systems Engineering-Theory & Practice, 29(12):94–104, 2009.

[41] Roger D Peng. Reproducible research in computational science. Science, 334(6060):1226–1227, 2011.

[42] Hans E. Plesser. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. Frontiers in Neuroinformatics, 11, January 2018.

[43] J. Ross Quinlan. Induction of decision trees. Machine Learning, 1(1):81–106, 1986.

[44] J Ross Quinlan. C4. 5: programs for machine learning. Elsevier, 2014.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[46] Steven L Salzberg. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Machine Learning, 16(3):235–240, 1994.

[47] Lalet Scaria. A framework to evaluate pipeline reproducibility across operating systems. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Quebec, Canada, 10 2018.

[48] Joseph L Schafer and John W Graham. Missing data: our view of the state of the art. Psychological Methods, 7(2):147, 2002.

[49] Bernhard Scholkopf, Sebastian Mika, Chris JC Burges, Philipp Knirsch, K-R Muller, Gunnar Ratsch, and Alex J Smola. Input space versus feature space in kernel-based methods. IEEE Transactions on Neural Networks, 10(5):1000–1017, 1999.

[50] Galit Shmueli, Peter C Bruce, Nitin R Patel, Inbal Yahav, and Kenneth C Lichtendahl Jr. Data mining for business analytics: concepts, techniques, and applications in R. John Wiley & Sons, 2017.

[51] Dan Sprevak, Francisco Azuaje, and Haiying Wang. A non-random data sampling method for classification model assessment. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 3, pages 406–409. IEEE, 2004.

[52] Harald Steck. Training and testing of recommender systems on data missing not at random. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 713–722. ACM, 2010.

[53] Dan Steinberg and Phillip Colla. Cart: classification and regression trees. The Top Ten Algorithms in Data Mining, 9:179, 2009.

[54] Victoria Stodden, Jonathan Borwein, and David H Bailey. Setting the default to reproducible. Computational Science Research. SIAM News, 46(5):4–6, 2013.

[55] Sang C Suh. Practical applications of data mining. Jones & Bartlett Publishers, 2012.

[56] Shan Suthaharan. Support Vector Machine, pages 207–235. Springer US, Boston, MA, 2016.

[57] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The WU-Minn Human Connectome Project: an overview. Neuroimage, 80:62–79, 2013.

[58] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. Expert Systems with Applications, 69:29–39, 2017.

[59] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2017.

[60] Guanwen Yao and Lifeng Cai. User-based and item-based collaborative filtering recommendation algorithms design. 2015.

[61] Nong Ye. Data mining: theories, algorithms, and examples. CRC press, 2013.

[62] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. HotCloud, 10(10-10):95, 2010.

[63] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. Neurocomputing, 237:350–361, 2017.