Final project for the Data Mining course 2016/17

Death Mining

Spark™-based market basket analysis on USA's mortality data

Marco Alfonso 1151026 – **Riccardo Belluzzo** 1129004
Antonio Bevilacqua 1050301 – **Gianluca Marcon** 1128001 – **Davide Martini** 1151502

1. Introduction

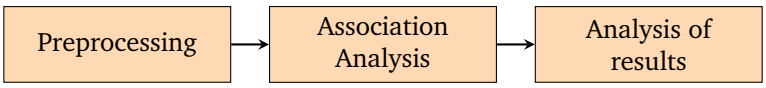
1.1. The mortality dataset

For our project we have decided to apply a Market-Basket analysis approach to a particular dataset we found on kaggle.com. It is called *the mortality dataset* and it consists of over 2.5 million records of deaths in the USA for the year 2014. It includes detailed information about causes of death and the demographic background of the deceased. We selected this dataset in order to perform association analysis which aims to retrieve interesting rules involving correlated features represented in the dataset. This choice explains the name of our project: *Death Mining*. We took inspiration for this approach from [1].
Generally speaking, the dataset is a collection of tables available in CSV format. The primary table is the one called *DeathRecords* which contains a total of 38 features, that we have interpreted as items of our market basket-like analysis. As consequence, each row of this main table represents a transaction. The majority of the values of each feature are referred to other lookup tables, where a detailed encoding can be found of each value.
As mentioned before, the main features of the dataset are:

- demographic background of the deceased, like: *Sex, Age, Education, Race, MaritalStatus*, etc.
- causes or other information related to the death of the individual, like: *MonthOfDeath, PlaceOfDeath, CauseOfDeath, MannerOfDeath*, etc.

In the following, the workflow of the project and all its steps are described in detail.

1.2. Project Pipeline and Objectives



As depicted in the flow diagram, we have organized the project into three core phases:

- **Preprocessing phase**: all the operation meant to prepare the dataset for the mining such as cleaning incomplete rows, solving redundancies and binning (see section 2)
- **Association Analysis phase**: frequent item-sets analysis and rule mining through the implementation of the algorithms seen in classroom; improvement of the results by setting up the parameters of the analysis (see section 3)
- **Analysis of results phase**: looking for interesting patterns and rules, combining the outputs with our a-priori field knowledge and visualizing the results (see section 4)

Each phase will be described in details in the following sections. The three core phases have been developed with Spark. In the final phase, we used some Python libraries for result visualization.

2. Preprocessing

The technique of association analysis does not have many parameters to set, therefore the preprocessing phase is fundamental. For instance, in the frequent itemset analysis, the *minsup* parameter has the principal role of reducing the volume of the output without affecting the results deeply. On the other side, even a couple of redundant features would produce an enormous amount of uninteresting results; all rules that contain the related information would be multiplied by all the occurrences of the redundancies.

2.1. Dataset description

The first thing to do in order to define the requirements of preprocessing is to take a deeper look into the data. The dataset has the shape of a table, therefore we adapted it to the market basket analysis paradigm by seeing each row as a transaction. We noticed that:

- All transaction have the same length, before the filtering phase. More generally, in the case where the filtering is applied, we can say at least that the maximal length of the transaction is known and fixed.
- Each feature can assume exactly one value per transaction. In particular every item has to be associated with the name of the related column. This is necessary in order to avoid superpositions of items with different meanings as well as repetitions, which are not allowed in the association analysis approach.
- The sum of the supports of the items relating to a same column equals to 1. This is an aspect to take in mind during the analysis of data and results. For instance, it may be useful to reject one value of a column and infer it by the absence of one of its complementary values (*tertium non datur* argument).

2.2. Filtering

The most important step of the preprocessing for our approach is certainly the filtering. A well designed filtering lets focus on most interesting columns, by improving the performances of the analysis. We performed two types of filtering: *by column* and *by value*.
Filtering by column is mainly based on our dataset's knowledge. It is meant most of all to remove redundancies and flag-kind attributes. It can be modified also in a second time if we notice that some columns do not generate good outputs.
On the other side, filtering by values requires also a field knowledge base. In particular, it aims to remove items that do not carry any information (e.g *Unknown*, *Not specified*, *Other*).

2.3. Binning

The binning procedure is meant to reduce the possible values for an item, and consequently making the outputs more readable. For example, in our dataset the deceased's age information could be expressed by its value in years or even in months (child deaths), but in this way we would end up having a column with more than hundred possible values with very low support.

This column would hardly produce any result.
If we decide for a binning, it is not straight forward to define it. It could be defined arbitrarily (e.g costant width intervals), statistically (e.g constant probability partitions) or basing on an a-priori knowledge of the field. In this case we opted to subdivide the age into the "classical" stages of human life (*Baby, Child, Teenager, Young, Adult, Old*). This appears meaningful to us, because we know that those category are very different and meaningful and are easy to locate in the age range. We behaved similarly in the *CauseOfDeath* category, where we tried to reduced the possible causes merging the similar one (e.g *Heart Attack* and *Ischemic Heart Disease* have been grouped in a more general *Heart Disease*)

3. Frequent itemset and Association Rules mining

At the core of our analysis lies the algorithm for discovering frequent itemsets and association rules. Since our goal wasn't the implementation of such algorithm from scratch, we based our work on the FP-growth algorithm, which was already available in the Spark library.

FP-Growth was introduced in [2], and differs from the A-Priori algorithm for its ability to completely avoid the candidate generation step, which can be costly both in terms of memory/storage requirements and in terms of computational complexity. As a consequence, its running time is an order of magnitude shorter than other A-Priori-like algorithms.

3.1. Issues and limitations

Spark's version of the algorithm is able to mine frequent itemsets (and their support) from a collection of transaction, where each transaction is itself a collection of key-value pairs, e.g. $\big[\langle\texttt{Sex, Male}\rangle, \langle\texttt{Age, Old}\rangle\big]$. Association rules are then generated from the collection of frequent itemsets by the Spark `AssociationRules` module. The first limitation we dealt with was that only rules with a single item as a consequent are discovered. After a closer look we decided that this could be an advantage. This structure of rules can be easier to read and understand in a manual analysis, because of the smaller number of generated rules. It also lightens the computation costs of the algorithm by reducing the number of generated rules.
The second issue was that the only interestingness measure computed by the algorithm is the confidence of the rules. Using only this parameter would complicate our analysis, so we decided to write a Spark program to extend the functionality of the predefined algorithms in order to compute Support, Lift and Conviction of each rule.

3.2. Spark implementation

Given an association rule $r : X \Rightarrow Y$, where $X, Y \in I$ and I is the set of items, we can define the above mentioned measures as

$$\text{Conf}(r) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X)} \tag{1}$$

$$\text{Lift}(r) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \cdot \text{Supp}(Y)} = \frac{\text{Conf}(r)}{\text{Supp}(Y)} \tag{2}$$

$$\text{Conv}(r) = \frac{\text{Supp}(X) \cdot \text{Supp}(Y)}{\text{Supp}(X \cup \overline{Y})} = \frac{1 - \text{Supp}(Y)}{1 - \text{Conf}(r)}\,. \tag{3}$$

We note that given the confidence of the rule its support, lift, and conviction can be easily determined if the support of both antecedent and consequent is available. Recalling that `AssociationRules` generates rules from the frequent itemsets (of which we know the support), we can associate each rule with the support of its antecedent and of its consequent through a series of `map` and `join` transformations, fully taking advantage of the MapReduce paradigm. The procedure is illustrated in Fig. 1. We also note that there's no need to store additional collections of data.

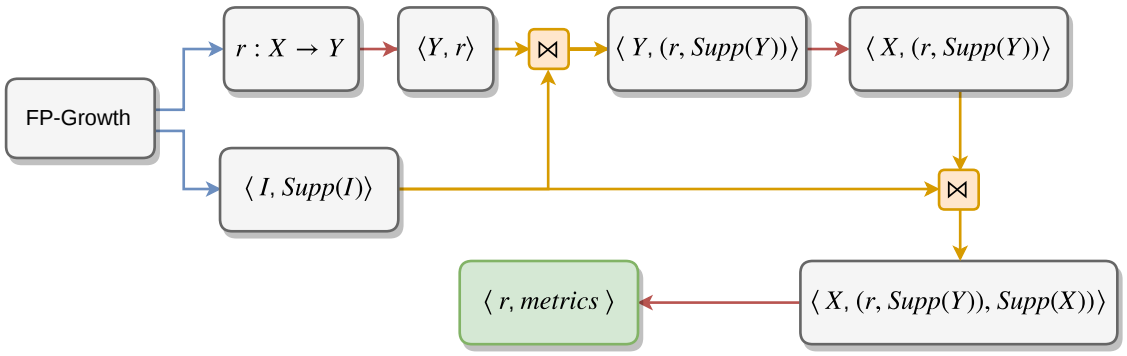


Figure 1: Block diagram illustrating the Spark algorithm to compute support, lift and conviction of the mined rules. Each gray block represents a Spark RDD (*resilient distributed dataset*). `map` and `join` operations are depicted respectively in red and yellow, while the blue arrows coming out of the FP-Growth block indicate a series of unknown operations implemented by the algorithm.

4. Analysis of the results

The first results we obtained running naively the algorithm on the dataset without any modification other than the standard preprocessing were not satisfying. We set the minsup parameter in the frequent itemset mining step to 0.1. We tried to lower it as much as possible by considering the computational issues that this implicated.
Despite of the metric we chose (confidence, lift or conviction) the resulting rules did not seem to be interesting, because they were all obvious and very repetitive.
The limit of lowering the minsup parameter caused the loss of a lot of less frequent items, that could be part of some patterns linked to some less common sub-population. The majority of the top rated rules (in the sense of high confidence, lift and conviction) were "monopolized" by the most common categories. To better understand this issue, let's have a look at the histograms of Fig. 2. As the reader could notice, there are some items such as *Old* or *White* that have a very large frequency and will certainly have a bigger influence on the output. At the same time, noticing that *Heart Disease* is the most frequent death cause, we can easily realize that the majority of the dead people in USA, year 2014, are old white people (>65 years old) that died for a quite common natural cause: the heart attack.

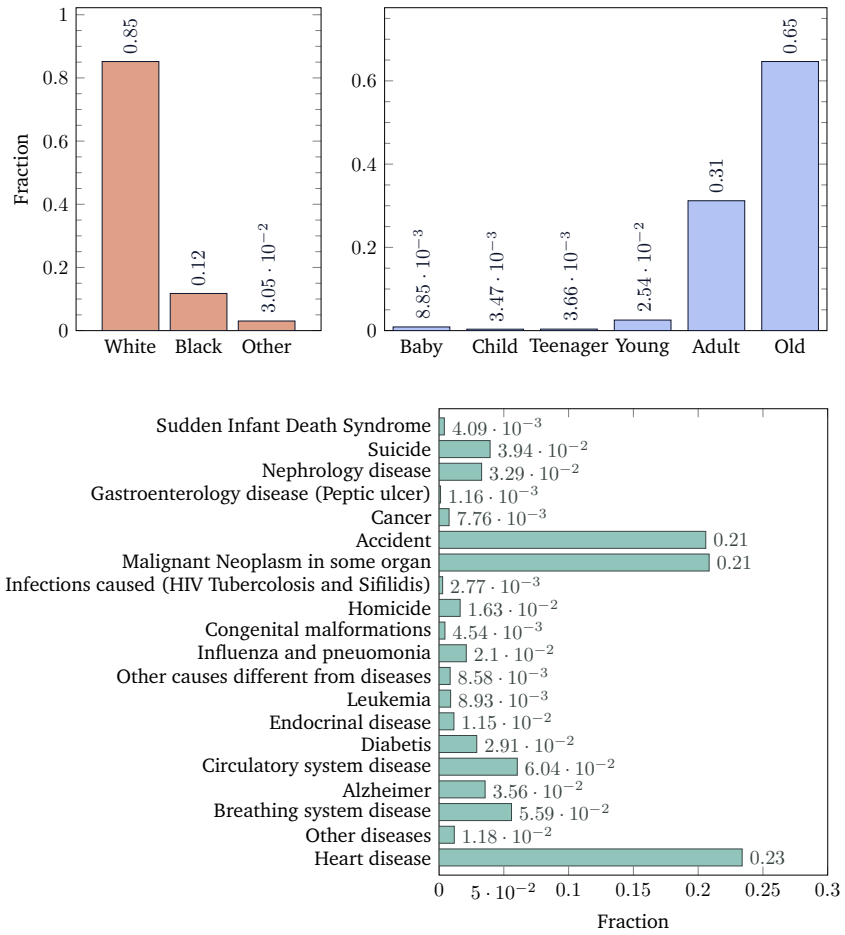


Figure 2: Histograms highlighting the distribution of each feature.

Our first attempt to contrast imbalance between classes has been to exclude the most common items (e.g $\langle$`Race, White`$\rangle$, $\langle$`Age, Old`$\rangle$). This solution seemed good for exploring the dataset more deeply. It did not influence metrics in the discovered rules as it did not affect the number of transactions, but only their composition. If the threshold of maximal frequency for the items was lowered enough, than it was also possible to lower the minimum support in the frequent mining. The big issue in this approach was that we could never have rules with both common and uncommon items at the same time (e.g $\langle$`Age, Old`$\rangle$ and $\langle$`Death Manner, Suicide`$\rangle$). Another way to solve the computational limit issue is to reduce the number of different items in the dataset, reducing then also the number of frequent itemsets and rules. We first selected columns that brought generally uncorrelated data. Those were in particular *DayOfWeekOfDeath* and *MonthOfDeath*. These columns generated a lot of uninteresting rules (lift = 1), as they are uncorrelated from most of the other features. The fact that those had a high number of possible values (*DayOfTheWeek* 7, *MonthOfDeath* 12) increased the number of different items added to the transaction list by a lot.

For this reason we decided to exclude those columns from the analysis. After this we could lower the minimum support parameter to 0.001 without stressing too much on the execution time or on the local memory utilization.

Rule	Lift
(Dead at the Hospital) ⇒ (Wednesday)	1.0136912799
(Dead at the Hospital) ⇒ (Tuesday)	1.014864294
(Dead at the Hospital) ⇒ (Thursday)	0.9990256068
(Dead at the Hospital) ⇒ (Sunday)	0.9833063797
(Dead at the Hospital) ⇒ (Saturday)	0.9820444323
(Dead at the Hospital) ⇒ (Monday)	1.0083992171
(Dead at the Hospital) ⇒ (Friday)	0.9989123458

Table 1: Values showing samples of uncorrelated rules including *DayOftheWeekOfDeath* feature

4.1. Lowering minimum support parameter: an opportunity or a curse?

With a lower `minsup` value, very different results are obtained. In this paragraph we explain which are the advantages, which the issues and how we tried to solve them. The biggest advantage is that many new strong patterns with very low support are found. For instance the most strong rules with the highest metrics have support included in the interval $[0.001, 0.1]$ (e.g those about homicides, suicides, neonatal deaths, rare illnesses). These rules are way stronger than those that we found before and this solved our issues about imbalanced classes, as they are especially highlighted by the lift parameter. On the other side, if the `minsup` is too low, less than 0.001, the number of generated rules is so big that it is difficult to analyze the output.
Another issue we found is that the algorithm produced a lot of duplicated rules. These can be related to the same patterns, but are distinguished by only some items that do not have a heavy influence on the metrics. It would require a lot of manual filtering to extract interesting rules. This is not acceptable, mostly because the purpose of this method is to help us explore a dataset more easily.
The best way we found to read the outputs is to plot rules by support, confidence and lift on a heat-map, as is shown in Fig. 3.

We did not use the conviction for this representation because it is difficult to plot clearly as it can easily tend to infinity. Those plots are very suitable for this case, in fact they let us see how the duplicated rules were grouped in clusters with close values in the represented metrics. This approach is unfortunately still vulnerable to all the "obvious" rules; for example, the yellow cluster at the top of the picture is composed by all those rules containing the items ⟨`Age, Baby`⟩, ⟨`Marital Status, Never married`⟩, ⟨`Education, 8th grade or less`⟩ which are expected to be correlated. A very useful feature of the tool we used is the possibility to interactively show the rule related to a point in the chart by pointing it with the cursor. This gives the possibility to explore rules in a very intuitive and natural way.

4.2. Sub-group mining

Another analysis that we performed on the dataset has been to apply the whole process to only a part of the dataset, depending the value of a feature (e.g ⟨`Age,Young`⟩, ⟨`Sex,Female`⟩, ⟨`Race,Black`⟩). This approach is useful to focus on a specified sub-population due to a particular interest or even to compare more complementary sub-groups (e.g ⟨`Sex, Male`⟩ with ⟨`Sex, Female`⟩ , ⟨`Race, White`⟩ with ⟨`Race, Black`⟩). We show here an example of this analysis on Fig. 4 using the solutions presented above, where we compare the rules regarding males and females. The same comparison was made for different races, age groups and death causes, but we couldn't include them due to our page constraints.

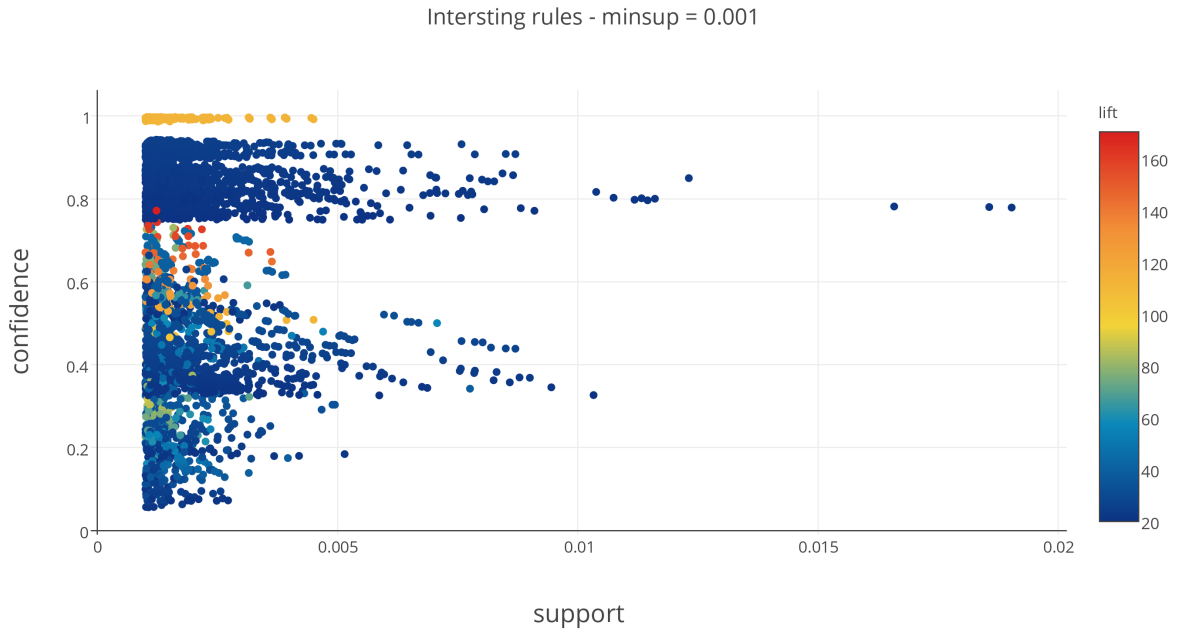


Figure 3: This heatmap has been plotted from rules with $\texttt{minsup} = 0.001$. The total number of generated rules were around 100000 and the great majority of the rules had low interest metrics. Therefore we filtered them by those with lift > 20 and we took a sample of 4000 rules.

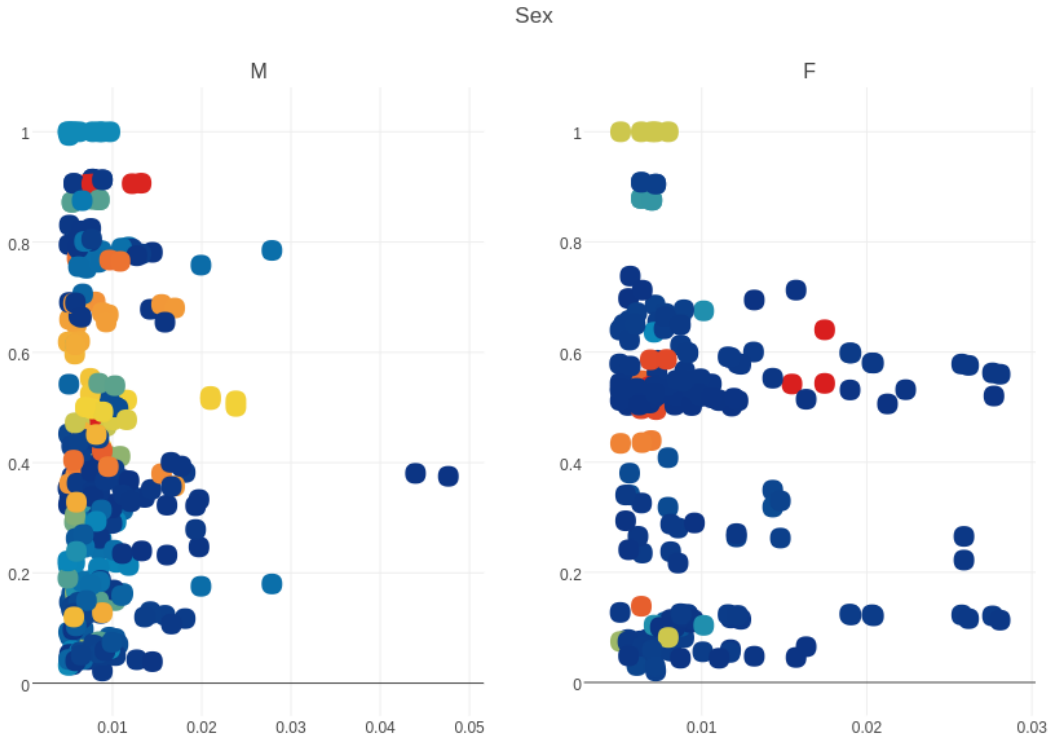


Figure 4: This comparison shows the different distributions in the two classes of Male and Female. It is very clear how the main blocks of rules aggregate near different *support-confidence* coordinates. The rules in this case have been filtered by lift between 2 and 20, so the gradient scale is the same for the two plots.

5. Conclusions

We tested various methods of preprocessing and filtering to extrapolate interesting relationships in the proposed dataset. The major obstacle was the presence of "obvious rules", relating features that have high correlation. This problem can be in part mitigated by considering rules with moderately high lift value though a final manual screening which is inevitable. Filtering those rules with a huge lift resulted in a more meaningful output. We list in Table 2 a sample of those results.

Rule	Confidence	Lift	Conviction
Adult White Females ⇒ Breast cancer	0.1226	3.4486	1.099
Males that died at home ⇒ Never married	0.382	3.0197	1.4138
Education:9-12th grade, Never married ⇒ Black	0.3463	2.945	1.35
White males that never married ⇒ Suicide	0.1199	3.042	1.091
Adult white men Education:9-12th grade ⇒ Divorced	0.3246	2.1302	1.255
Adult white females ⇒ Suicide	0.053	1.346	1.0144
Males injured at home ⇒ Homicide	0.3021	18.54	1.409
Adult Males Suicide ⇒ Divorced	0.3146	2.064	1.236

Table 2: List of meaningful rules.

References

[1] M. MacDougall, “Shopping for voters: Using association rules to discover relationships in election survey data,” *SUGI*, vol. 28, no. 122, pp. 1–5, 2001.

[2] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” *SIGMOD Rec.*, vol. 29, pp. 1–12, May 2000.