BigData Project

Christophe yang
University Paris cité

Luca bresolin
University Paris cité

Amine Benmehrez
University Paris cité

April 25, 2022

I. Introduction

Chronic non-communicable diseases like cancer, cardiovascular diseases, diabetes, . . . , represent the most frequent causes of death in industrialized countries. These diseases account for 88To face the current public health problems and to prevent these chronic diseases which does not cease growing, the international organizations such as the World Health Organization recommended nutritional measures of prevention to improve the nutritional status of the consumers. Among these measures are the nutritional logos. The Nutri-Score is a synthetic and gradual system located on the front of packaging which serves as a guide to compare the nutritional quality of products in the same category (pizzas, dairy products, biscuits, etc.) or between two different brands, it is even more effective when the product has a high degree of elaboration (cereals, ready meals, etc.). Using a scale of five colors each associated with a letter, it is based on nutritional benchmarks considered simple, obvious with proven and proven health effects (sugar, salt, fat, energy content, etc.). It facilitates nutritional advice by health professionals, helps consumers to quickly know the nutritional quality of a product or food and encourages industries to improve the nutritional quality of their product. The diversity and the daily arrival of new food products complicate the choice for the consumers and they don't always know which product to choose and what are the products that are the best suited for their health, and instead they choose a product with a low nutritional value. The goal of our project is to create a Nutri-Score predictor from the nutritional data of the product in order to be able to predict whether the nutritional value of food products and better understand the nutri-score.And at the same time find the best tools to achieve our goal.This could help consumers better choose their products and therefore reduce the risk of chronic diseases.

II. Related work

In this section, we briefly summarize the related work we used for our project and why we used them. Part of the topics discussed below are about column-oriented databases.MonetDB is one example of a column-oriented database. Column-oriented database system store every column with their attribute value at different location on disk in comparison to the traditional row-oriented method that store the rows one after the other. But it also has disadvantages like the write operations and tuple construction which can cause performance issues when inserting tuples because every attribute must be written separately.Column-oriented database system accomplishes a significant speed-up in read operations.

MonetDB uses vertical fragmentation to exploit efficiently the main memories during query processing and can reach a high level of speed on column store in comparison to the traditional designs and this is possible and is due to all the innovations made in all layers of a database management system, MonetDB creates a relational table that maps an object identifier to the corresponding value. And to avoid performance degradation, it limits access to the processor cache.[8][2]

MonetDB performs well in an inherently dynamic data storage environment as it allows the physical reorganization of the data.It is the Cracking that allows an adaptive indexing. Constant and automatic cracking allows for queries to run at the same time and adapt indexes to the workload, the indexes are optimized every time we have to process more queries. It also gives the possibility to determine a limit to the storage and cracking adaptively to maintain its structure by using partial cracking.[5]

Monetdb was one of the first database to introduce database cracking.

We can design a system that is simple and flexible system and is able to handle huge sets of data and automatically organize data the way users request and allows multiples queries to run at the same time with Database Cracking. The section about database cracking shows the difference between cracking, sorting and the default MonetDB select operator, and we were able to conclude after several experiments that it is better to use cracking over the others in terms of time costs, the simple scan grows linearly with the number of queries while with cracking the more queries arrive, the less data need to be touched or physically reorganized.[6]

To increase the speed of implementation on modern hardware, the MonetDB algebra has been simplified in comparison with traditional relational set algebra and used database cost models to provide foundations for query optimizers. While constructing more complex CPUs to make the common case fast, the dynamic random access memory access latency has not changed in a significative way and this "memory wall" can reduce by two orders of magnitude modern CPUs, this is why we use cache memories. Cache memories keep among other things the recent accessed data and becomes faster when it is placed near to the CPU.[4]

Minimizing data transfer. We can distinguish two types of digital information: the input data provided by the user and the output data provided by the computer even if the processor of a computer needs the user in order to calculate or produce output data. NDP provides several benefits and impact, placing the storage in a location near to the compute units minimize data transfer and response time and can handle the execution entire queries by its computing units and also increases significantly the operation of data-intensive application.[10]

Theses paper demonstrate the advantages of column-oriented databases and let us wonder how does it compare with the traditional row-oriented databses.

A data structure is a particular way of organizing data to be used effectively, all the algorithms that are used to deal with data starts by defining its data structure. There is so many ways to design a data structure but we can't have perfect data structure design, every design is a compromise between the fundamental tradeoffs and every design has its strengths and weakness.[7]

In this section, we are going to discuss about partitioning a set of feasible solution in linear regression who has a wide range of application into smaller subsets of solution with the objective of finding the best independent variables that can perform the tasks in an optimal way and able to explain most of the information in a moderate computation time, improve the precision of a model fit and prevent overfitting. When the predictor vector is too long and contains many variables, variable selection becomes necessary. Instead of wasting a lot of time trying several values for each variable which in the end may not bring a significant change, it will save us a lot of time and will allow us to apply it in several fields such as climate prediction and genetics and many other fields. The section has as objective to develop methods able to find better solution than traditional methods and compares the Branch Bound method with a new variant created by adding tools to the original method (Branch Bound method) who use the information provided by a heuristic and other variable selection methods. After a series of computational experiments to analyze the performances of both methods, the study shows that these methods can be used in large datasets with and still obtaining good results and does not depend on the size of the dataset but rather on the number of initial variables.[9]

III. Problem

There is a lot of database management systems available for the public. But we want to know which one is better for our use cases? That is data acces time and queries. So, we picked 2 popular dbms : monetdb and postgresql. MonetDB is an open-source column-oriented relational database management system. Postgresql is a free and open-source row-oriented relational database management system. So we want to highlight the cases when column-oriented database or row-oriented databases is better. So, we have to install theses dbms and load the data in the databases.

In parallel, we also want to find out if there is any useful data to extract from this database. Such data could inform us about the link between nutriscore and calories for a given product or about the evolution of sugar quantity in products over time. Nevertheless, finding relation between chunks of data in such large proportions is really hard to do manually hence we will probably need to use machine learning models and data visualisation techniques in order to do it efficiently.

IV. Solution

i. Technologies overview for dbms

Column-oriented database stores data tables by column.So when we need to use only a subset of the database columns, the reading operation is faster because we don't read irrelevant column. But the downside compared with row-oriented database is that the insertion operation is more expensive.

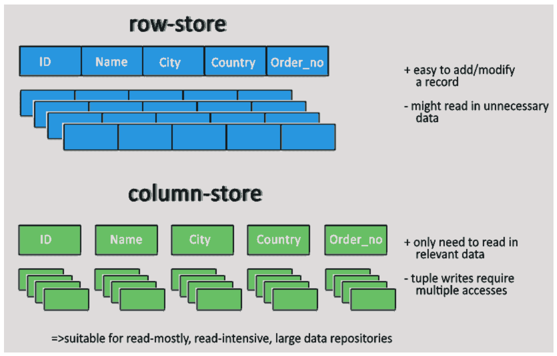
row-store
ID
Name
City
Country
Order_no
+ easy to add/modify a record
- might read in unnecessary data
column-store
ID
Name
City
Country
Order_no
+ only need to read in relevant data
- tuple writes require multiple accesses
=>suitable for read-mostly, read-intensive, large data repositories

Figure 1: *row-oriented versus column-oriented*

We need to compare monetdb and postgresql on different usecases. There is some usecases where we want to do some linear regression.

We will use python to work with theses 2 databases because python have easy-to-use machine libraries and librairies that can interact with these two databases systems. We will use theses python libraries :

- pymonetdb : connect to monetdb DB
- psycopg2 : connect to postgresql DB
- seaborn : data visualization, linear regression

ii. Procedure

ii.1 Installation

We installed monetdb on a linux machine. In our case, we suceed to install monetdb on debian buster. But, the installation was difficult on ubuntu 20 because there was security problem in the installation.

ii.2 Datasets

For our experiments, we used the dataset from OpenFoodFacts. It is a open, collaborative dataset. The dataset was initially in CSV.The dataset size is 5.7GB.The dataset have 186 columns and 2 251 894 rows. In monetdb, we can feed our csv to our database with the help of the COPY INTO STATEMENT. The downside of this method is that for each column we need to manually specify their datatypes. We also meet some issues due to the collaborative nature of the dataset. In fact, the dataset isn't homogeneous that is to say that it doesn't respect a standard. So we can see in the same column different encoding for STRING records. In this case, we needed to filter out the non UTF-8 encoded records because monetdb doesn't have a datatype that accept non UTF-8 STRING. There is also some column that obviously is of one data type like "nutriscore_score" which is "INT" but we can find in the column some record that of string type. For the nutriscore_score column we casted it in STRING then after the copy operation we recasted it in INT. There is also some missing value in the data, so that some records doesn't have all the 186 columns. We removed theses records to clean the dataset with the help of unix commands. Finally, we got a dataset that can be copied into a monetdb database table. But, the installation was difficult on ubuntu 20 because there was security problem in the installation.

ii.3 Comparative study

We want now to do a comparative study on perfomance on multiples uses cases for monetdb and postgresql. So we will run multiples use cases on monetdb and postgresql then store the response time for the two. We will do it multiple time then take the mean time for each use cases. We will use algorithm 1. Later we will analyze the results.

iii. Queries selections to test

1. We want to see the distribution count of nutriscore
2. We want to see the average fat proportion by countries
3. We want to see the average number of additives in each nutriscore_grade

```
1.SELECT "nutriscore_score" FROM
    opf WHERE "nutriscore_score"
    IS NOT NULL
2.SELECT "countries",AVG("
   fat_100g") FROM opf WHERE "
   countries" IS NOT NULL AND "
   fat_100g" IS NOT NULL GROUP
   BY "countries"
3.SELECT "nutriscore_grade",AVG(
   "additives_n") FROM opf WHERE
    "nutriscore_grade" IS NOT
   NULL AND "additives_n" IS NOT
    NULL GROUP BY "
   nutriscore_grade"
```

Algorithm 1 Monetdb / postgresql

```
Require: monetdbConnect, postgresqlConnect, SQLStatementList
0: $m \leftarrow monetdbConnect.cursor()$
0: $p \leftarrow postgresqlConnect.cursor()$
0: $mTime \leftarrow []$
0: $pTime \leftarrow []$
0: while $SQLStatementList \neq \emptyset$ do
0:   $s \leftarrow SQLStatementList.getOne()$
0:   $start \leftarrow time.time()$
0:   $m.execute(s)$
0:   $end \leftarrow time.time()$
0:   $mTime.append(end - start)$
0:   $start \leftarrow time.time()$
0:   $p.execute(s)$
0:   $end \leftarrow time.time()$
0:   $pTime.append(end - start)$
0:   $SQLStatementList.remove(s)$
0: end while
   return =0
```

iv. Data analysis

To analyze our data efficiently we decided to use Python with libraries like SKLearn, Numpy, Pandas, Matplotlib and Seaborn. When we were together we used our personal computer to also use databases from the data management part of this project and gain time, but we had to use GoogleColab (free version) otherwise, which is working well if we do not take into account the time it takes to download and set up our data in the environment before we can use it.

We wanted to look both at trends over time and to understand how the nutriscore is calculated and whether or not it is predictable. Common models such as LinearRegression, Support-vector machine regressors and RandomForest regressors came to our mind to try to emphasize correlations that may exists between different features.

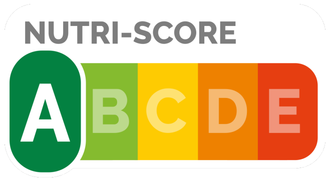

Figure 2: *Nutriscore is a label attempting to easily identify a product as healthy or unhealthy for consumers*

Before analyzing it, we looked into Nutriscore and how it is supposed to be computed, according to this quick article[1] , nutriscore is positively influenced by both fiber, protein, fruit, vegetable and nuts quantity (among others characteristics) and negatively influenced by sugar, fat, energy density and salt. We will verify these relations in the first place and then use these information to try and predict the nutriscore of products as precisely as we can.

Looking at trends appears to be feasible but hard, as products are only tied to the date they were added to the database, and not the one they were commercialized. We will try our best to make it possible.

For every information we will extract from the dataset, we will need to look first at the quality of the data, as it seems very uneven : lots of not assigned fields, outliers, typo errors, multiple categories for the same meaning. (e.g France, Frankreich, en:france for french products)

V. Experiments

i. Hardware specifications

We launched our experiments on debian buster (intel i5-8300H) 8GB RAM. Specifically, we used our computer for the benchmarking of dbms. But we used Google collab to simulate raw file access because we didn't have enough ram.

ii. Results

ii.1 Acces time

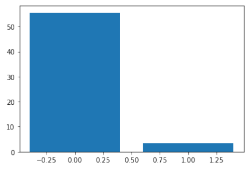

Figure 3: *monetdb acces and pandas acces*

We can see that reading through monetdb is 16 times faster than reading through the raw file.Using a database is definitely better than raw acces data. We could also test raw file acces and queries with NODB[3] but we didn't have time.

ii.2 SQL queries

1.We want to see the distribution count of nutriscore

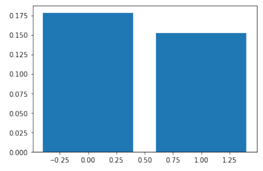

Figure 4: *1 statement*

We don't see much difference between the 2 dbms.
2.We want to see the average fat proportion from french product

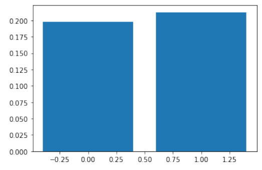

Figure 5: *2 statement*

We don't see much difference between the 2 dbms.

3.We want to see the average number of additives in each nutriscore_ grade

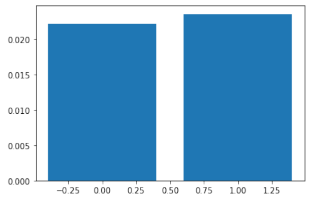

Figure 6: *3 statement*

We can see that there isn't such a big difference between postgresql and monetdb. We can't see results probably because 5.7GB isn't that big. And our computer isn't powerful.

iii. Data analysis

In the Data Analysis part, we had some nice results and some disappointments. Our first deception was for everything related to trends over time, we were unfortunately not able to do anything valuable because the only way we could analyze trends was using the column called "created_t" which is basically the time when the product was added to the database, but because of how OpenFoodFacts works, some dates are overloaded with products while others are empty, add to this the fact that a large amount of product was created way before it was ever added to this database, it makes everything fail when it comes to analyze trends in product composition or anything time related. See below an example of what kind of relation we can expect from such analyses :

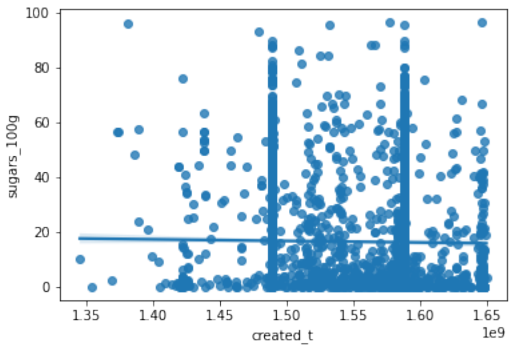

Figure 7: *Regression and plot on sugar per 100g through time*

Hopefully, other tries to analyze the data were more fruitful, like analysis done around Nutriscore :

Before looking at patterns, we noticed that nutriscore in our database was divided in two variables : scores and grades. We decided to first look into it by looking at the distribution of the scores (colored by grades) and the link between scores and grades :

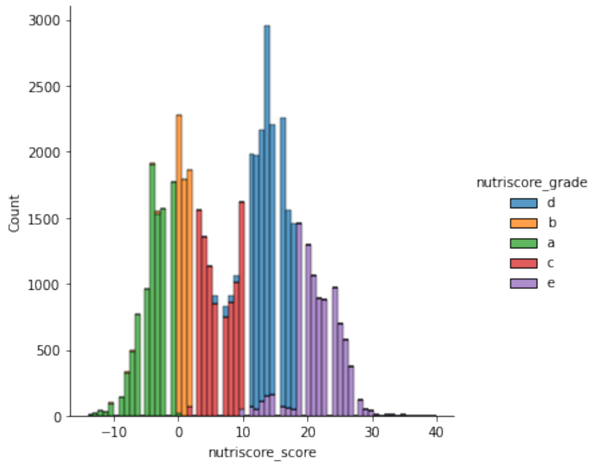

Figure 8: *Distribution plot of nutriscore score colored by nutriscore grades*

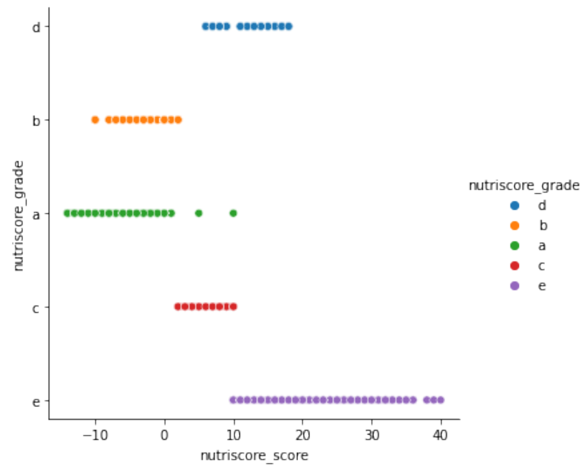

Figure 9: *Nutriscore score in function to grades colored by grades*

Lower grades seems to have more uniform score since there is less overlap between grades and the higher the grades and the less uniform the distribution gets and the less correlated score and grades become. We can observe that a good nutriscore (i.e A,B grades) is represented by a low score while a bad one (e.g E,C grades) is characterized by a high score. Thus, the higher the nutriscore number, the less healthy is supposed to be the product.

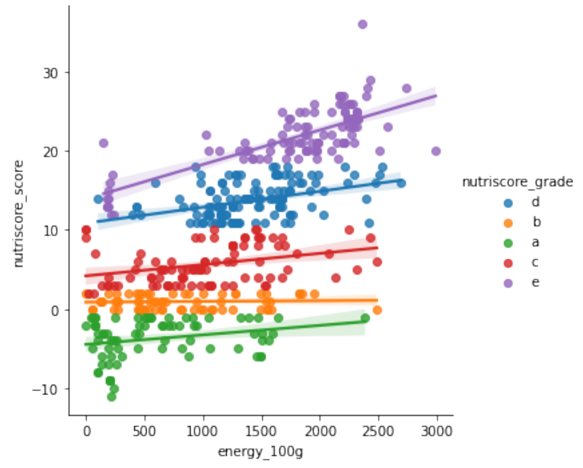

Figure 10: *Regressions and plots on calories per 100g in function of nutriscore score colored by nutriscore grades*

We then decided to work with nutriscore score as we figured its numeric nature would make it easier to use than grades in order to work with with different machine learning models and especially to highlight correlation and regression. Afterwards, we moved on to check whether or not influences mentioned by our sources [1] really had a great impact on nutriscore.

Firstly, as expected more calories, more fat and more sugar are negatively correlated with a good (i.e high) nutriscore :

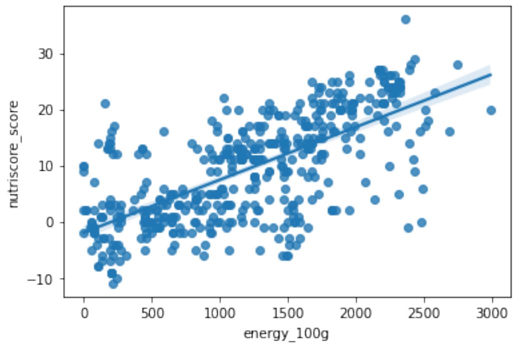

Figure 11: *Regression and plot on energy per 100g in function of nutriscore*

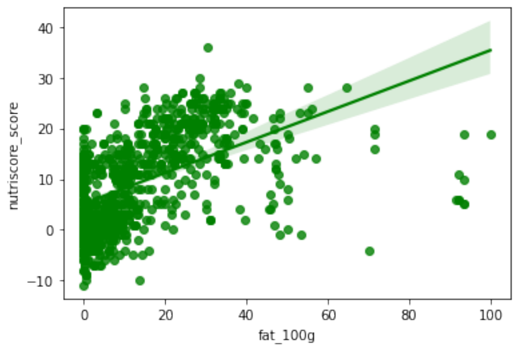

Figure 12: *Regression and plot on fat per 100g in function of nutriscore*

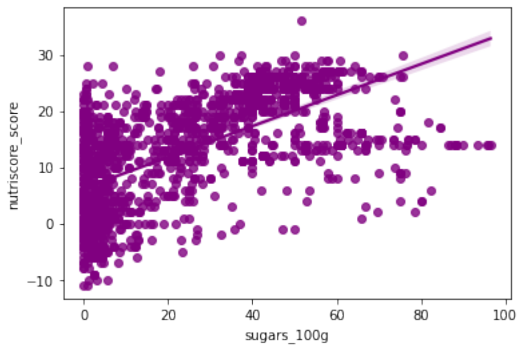

Figure 13: *Regression and plot on sugar per 100g in function of nutriscore*

Despite what the article says, protein seems only weakly correlated with a good (i.e low) nutriscore. However, fiber density and fruits, vegetables and nuts density seems a little bit more correlated to it.

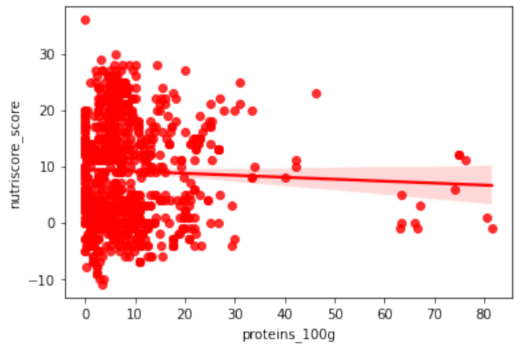

Figure 14: *Regression and plot on protein per 100g in function of nutriscore*

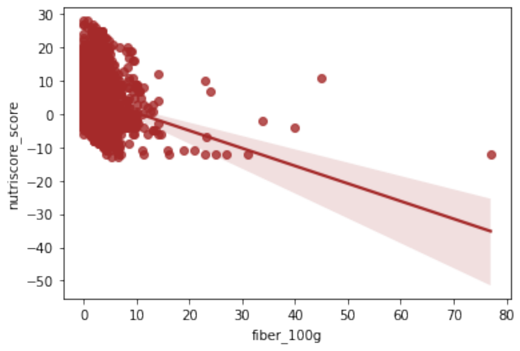

Figure 15: *Regression and plot on fiber per 100g in function of nutriscore*

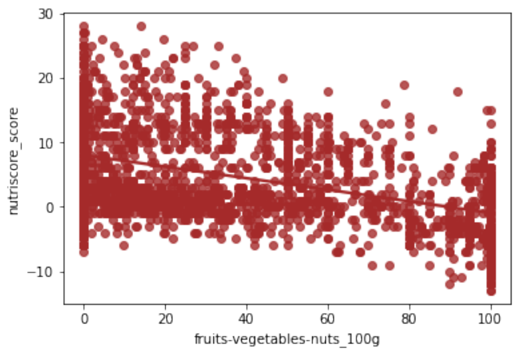

Figure 16: *Regression and plot on fruits, vegetables and nuts per 100g in function of nutriscore*

With this analyzes done, we tried to uses several characteristics of products, which we heard and verified had a link with nutriscore to try and make a classification model able to classify products in five different classes (i.e five different nutriscore grades). We compared different models and chose to use a RandomForestClassifier from the sci-kit learn library with 500 trees. His only major downside is that it probably is overfitting a bit much but he is way more precise than the others models we have tested so far (88 percent against models with accuracy as low as 40 percent).

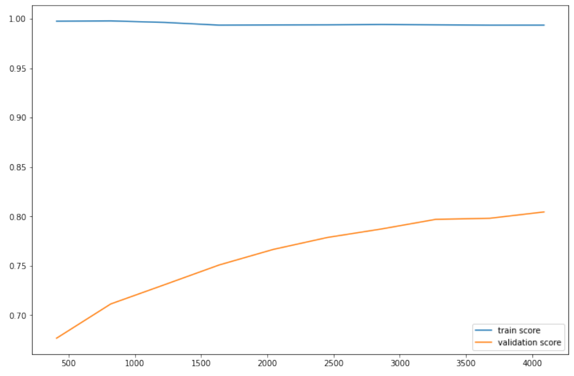

Figure 17: *Learning curve of the model*

VI. Conclusion

The experiments didn't show a winner between monetdb and postgresql but it was probably due to the low size of our database for the dbms or we didn't do the experiments that can showcase such difference.One big and time consuming part of the project was cleaning the dataset because it was dirty (written by different sources). Perhaps, in our case where preparing and loading data in the database was a big problem using nodb[3] was the right call.

We were not able to analyze trends in products composition due to inherent problems of the dataset. Nevertheless, we ran some analyzes and established an average performing model to classify products into their right nutriscore-grade with only some information on their composition such as their caloric density or their saturated fat density to only name a few. Considering the poor quality of the data and the presence of numerous outliers and errors in the dataset, our result of 88 percent seems satisfying. call.

References

[1] Nutriscore : informations et réglementations. 2022.

[2] Daniel J. Abadi, Peter A. Boncz, and Stavros Harizopoulos. Column-oriented database systems. *Proc. VLDB Endow.*, 2(2):1664–1665, aug 2009.

[3] Ioannis Alagiannis, Renata Borovica-Gajic, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. Nodb: Efficient query execution on raw data files. *Commun. ACM*, 58(12):112–121, nov 2015.

[4] Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. Breaking the memory wall in monetdb. *Commun. ACM*, 51(12):77–85, dec 2008.

[5] Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.

[6] Stratos Idreos, Martin L. Kersten, and Stefan Manegold. Database cracking. In *In CIDR*, 2007.

[7] Stratos Idreos, Kostas Zoumpatianos, Subarna Chatterjee, Wilson Qin, Abdul Wasay, Brian Hentschel, Mike Kester, Niv Dayan, Demi Guo, Minseo Kang, and Yiyou Sun. Learning data structure alchemy. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 42(2):46–57, 2019.

[8] Artur Nowosielski, Piotr A. Kowalski, and Piotr Kulczycki. The column-oriented data store performance considerations. 09 2016.

[9] Joaquín A. Pacheco and Silvia Casado 0001. Variable selection for linear regression in large databases: exact methods. *Appl. Intell.*, 51(6):3736–3756, 2021.

[10] Tobias Vinçon and Ilia Petrov. Near data processing within column-oriented dbmss for high performance analysis. In Dieter Hertweck and Christian Decker, editors, *Digital Enterprise Computing (DEC 2016), Böblingen, Germany, June 14-15, 2016*, volume P-258 of *LNI*, pages 235–239. GI, 2016.