A closer look at Machine Learning Pipelines in Python and SQL

Christian Fuchs

Otto-Friedrich-Universität Bamberg

Abstract

Preparing information is a necessary step in training machine to compile tasks with a high successrate. This data often has different sources and is in better cases already stored in database systems, which are controlled by a DBMS. It's common to extract the data and through multiple operations first prepare, then train and examine the data after the machine learning happens. Often libraries compatible with Python are recommended to compile these operation. Preferred is Python Panda a library, which combined operation specialise to prepare and combine data. In the following we look at different may disruptions and appearing problems in the steps of preparing data, at countermeasures and in the end compare the systems of compiling pipelines in Python and in SQL and similar Queries.

1 Introduction

In prepossessing pipelines data is gathered through risking biases and the amount of time often is the most damaging element to make modifying of data more difficult and disrupt the use we gain from compiling data. Both fails can happen whether code is compiled in Python or in SQL. In SQL to fight biases it's often necessary to inspect intermediate results between operations through View and CTE, which then has an negative impact on the runtime, in Python there is the framework mlinespct, which brings two operation to identify biases, especially technical biases and works well with not too complex panda operations. This difference will be looked in section two, we analyze how biases appear, how to differentiate them and what operations helps us to still filter the data without appearing biases. Why translation?

Expected Advantages:

- Performance benefits
- Easy readable code
- Enables in-memory performance
- Eliminates the overhead of function calls

Expected Disadvantages:

- Need to materialise subresults.
- Existing libraries can't be used

2 Prepossessing Pipeline

Prepossessing Pipelines are steps of operations to manipulate and comprehend data sets before using them in training methods. Therefor it is often taking big datasets in following comparisons it's between 2000 to 9771 tuples every tuple containing information. [1] To measure the performance it's necessary to take a look at different pipelines starting with various amounts of tuples and with a different number of compiling steps. After the last step finished it becomes possible to compare all tuples with each other and to compare runtime. By comparing results we cancel out biases, both Python and SQL have often different ways to get to a train set. "Blue Elephants inspecting Pandas" especially looks at the panda library, which is through frameworks possible to be replaced without writing SQL Code.
Correctness is verified by comparing the intermediate results, in this steps it's possible to detect biases and have an overview of amount of data gained. Often the SQL snippets, are longer, but genuinely easier to understand as the python functions seen on the left. Both functions are compiling the same operations and work on the same datasets.[2] Still both codes snippets access databases and manipulate, SQL in this case allowing the user to specify what to do rather than caring about optimisation details. Often the following operations are used in preprocessing piplines:

- join
- aggregate
- filter
- split

[1] https://openproceedings.org/2023/conf/edbt/paper-168.pdf
[2] https://streamsets.com/blog/python-vs-sql

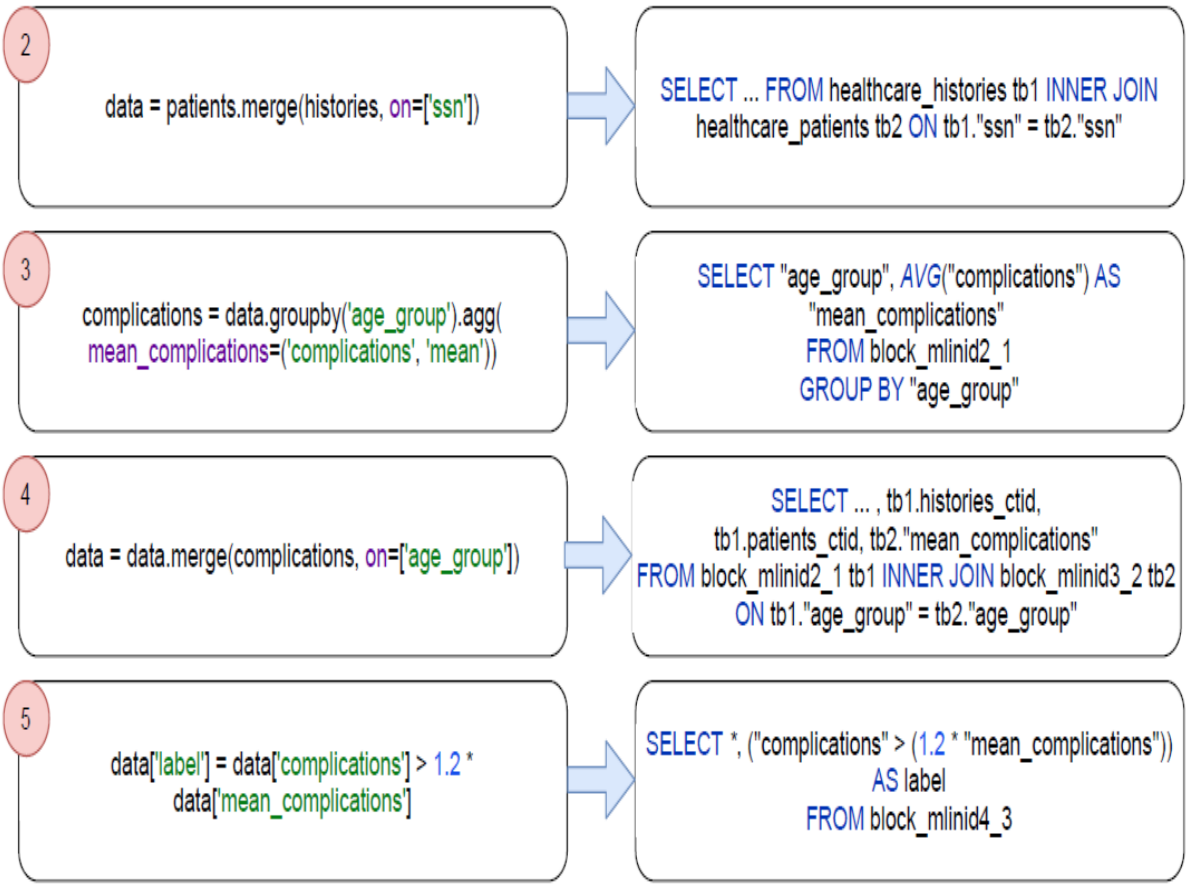

Figure 1: Translation "Blue Elephants Inspecting Pandas" 2020

In the above graphic it shows typical steps in a prepossessing pipeline in Python as well as in SQL. It's possible to translate the operation one by one. In the SQL Code to look at steps between the SELECTs is necessary before compiling the next operation. The panda framework enables the following operations:

- Read from CSV
- Merge/Join
- Selection and Projection
- Arithmetic/Boolean Operations
- Group-By and Aggregation
- Drop Null Values
- Replace
- Row-Wise Operations

In a runtime comparison we see, with an increasing number of tuples all database systems execute the additional inspection steps faster.[3]Therefor we look at different ways to order the steps of the pipeline and to compile them. On the one side it's possible to translate each function call, but in order to reduce dependencies it helps to gather all operation, which lead to current result. One the other side it helps to have a first overview over necessary functions to safe possible time lost and prevent redundancies. The different runtimes are going to be compared in graphs in section three. [4]

Lookup table help to map the panda functions, which often are named different to their counterparts in SQL. Not only the function get translated, but SQL maps the information contained in queries and tuples. It's important to keep the list of identifiers containing tuple identifiers, that biases detection can work efficiently.

[3]https://github.com/pandas-dev/pandas/tree/main/scripts
[4]https://openproceedings.org/2023/conf/edbt/paper-168.pdf page41

2.1 Operations

In comparison to the panda function, machine learning functions can be more complex and often have arithmetic operations. Still it's possible to translate following Sci-Kit Learn Functions, which often are used after the prepossessing and help train machine learning: Typical Sci kit-Learn Functions:

1. Simple Imputer
2. One-Hot-Encoder
3. Standard Scaler:
4. KBins Discretizer
5. Binarize

5.2.3 *Standard Scaler.* The standard score[6] z of a sample $l \in X$ is calculated as $z(l) = \frac{l - mean(x \in X)}{stddev(x \in X)}$. The mean and standard deviation is calculated in the fitting step (Listing 17) and reused for any other transformation.

```
SELECT (("label" - (SELECT AVG("label") FROM origin)))
      / (SELECT STDDEV_POP("label") FROM origin) AS "label"
FROM origin
```

Figure 2: Scaler in SQL
"Blue Elephants Inspecting Pandas" 2020
The translation of all those features is described on page 47. [5]

2.2 Biases

Biases often appear in prepossessing pipelines and having an impact on those pipelines and the different outcome. There is to be differentiated between two different types of biases the technical and the introduced bias.

- Technical Bias: Appears in SQL and python scripts. Often after a preprocessing pipeline miss-classifies data.
- Introduced Bias: Systematic error, whether through dataset with already existing problems or appearing in process.

Materialising View/CTE helps detecting them. It's possible to use Mlinespect to detect them.

2.3 mlinespt framework

Provides two checks:

- NoIllegalFeature: Verifies that none of the used features in provided dataset are contained in blacklist of illegal feature names
- NoBiasIntroducedFor: Targets pre-existing and technical biases

Mlinspect looks at data pipelines created with pandas and scikit-learn.
Mlinspect is designed to understand the semantics of preprocessing operations of popular Python frameworks from the data science space like scikit-learn and pandas. [6]

3 Performance test

3.1 Datasets

On the basis of three different datasets four different pipelines get tested, partly just prepossessing pipelines, partly pipelines developed for training and testing. Look at following pipelines: healthcare, compas, adult simple and adult complex. They differ in tuple size and used operations. Except adult complex, all use pandas and scikit-learn operations. While healthcare works on the smallest dataset with just 889 tuples, the adult dataset contains up to 9771 tupels. The instrumentation based on captured function calls described so far is independent of the specific library. Just comparing panda operation no view/CTE needs to be shown, as all operations are only executed once.

3.2 Graphs

To prove translation is possible the results of different Pipelines is shown in the following graph. In the Analysis we differentiate between the runtimes by only panda operations, additionally Sci-kit-learn operations and additionally inspection operations. A look at the four datasets results show that database systems perform data prepossessing faster than using dataframes in pandas.

[5] https://openproceedings.org/2023/conf/edbt/paper-168.pdf page47
[6] https://www.cidrdb.org/cidr2021/papers/cidr2021$_paper$27.$pdf page$3

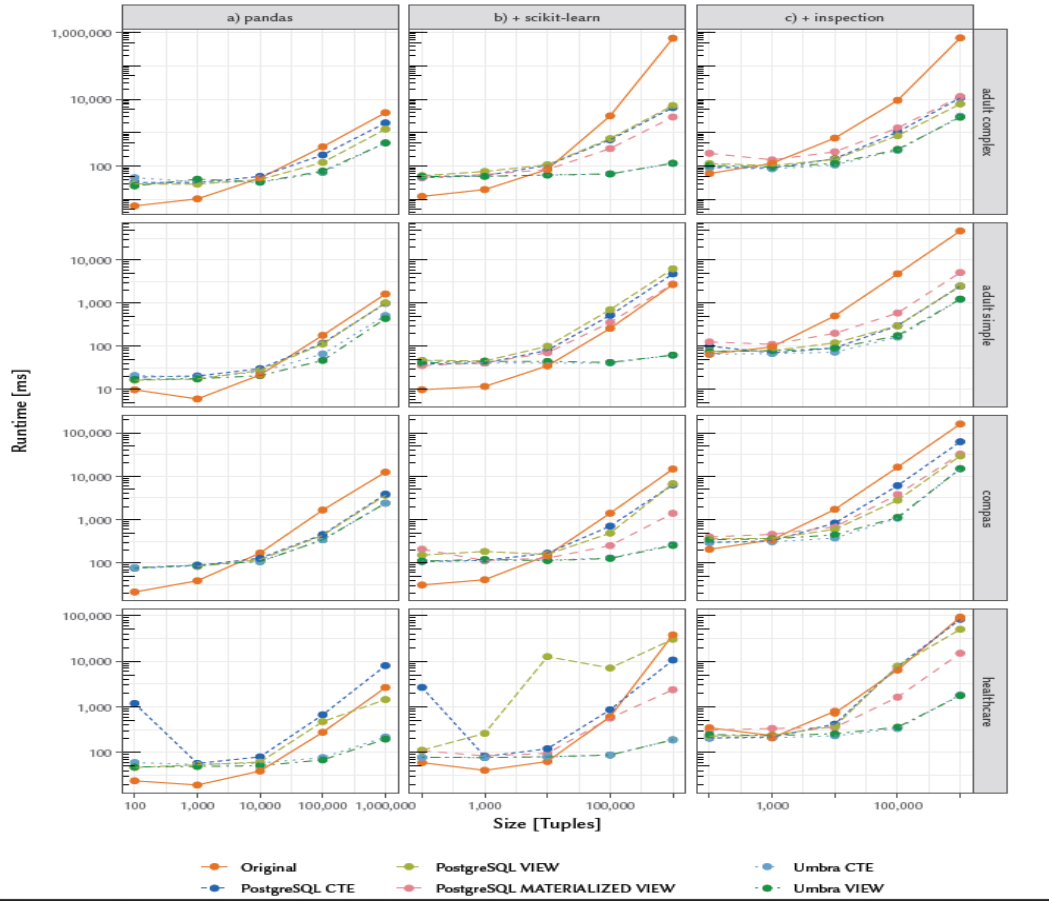

Figure 3: Runtime Analysis
"Blue Elephants Inspecting Pandas" 2020

4 Conclusion

After the Inspection of prepossessing pipelines in SQL, covering the frameworks panda and sci-kit and a runtime comparison, the advantage of python code be translated into SQL is visible. Therefor even with using supplemented tools for the mlinspect framework and with utilizing of CTE and View SQL improves the working with machine learning pipelines. It'S necessary to prevent the occurrences of technical biases and use modern data systems like Umbra to maximize performance improvement. Further success would be to prevent PostgreSQL to lose time due to additional data loading and extraction. This is eliminated when all computations happen inside database systems. [7] Still the out-performance between PostgreSQL and original inspection is significant, therefor it's worth to further follow the lead of translation of Python code into PostgreSQL.

[7] https://openproceedings.org/2023/conf/edbt/paper-168.pdf page51

5 Sources

https://www.cidrdb.org/cidr2021/papers/cidr2021_paper27.pdf
https://ubc-cs.github.io/cpsc330/lectures/05_preprocessing-pipelines.html#
https://openproceedings.org/2023/conf/edbt/paper-168.pdf

https://streamsets.com/blog/python-vs-sql/#Performance